KNOWLEDGE-BASED WEB SERVICES FOR CONTEXT ADAPTATION

A Dissertation Submitted in Partial Fulfilment for the Degree of Doctor of Engineering

By

Karen L. Lawson

BS, Applied Physics, University of California, Davis, 1985

MS, Environmental and Biomolecular Systems, Oregon Graduate Institute, 1994

Academic Supervisor

Anthony Finkelstein, Professor and Head

Computer Science Department

University College London

University of London

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ACKNOWLEDGEMENTS

There are several people who have contributed to the completion of this thesis both directly and indirectly. I would like to express my sincerest thanks to the entire team from the Systems Science and Software Laboratory at Kodak Limited for their camaraderie and the effort expended on creating next generation technology, many of the ideas and concepts were motivated in part by the discussions and issues found in our projects. Specifically, I would like to thank and recognise the contribution of Patrick McNulty, for his willingness to discuss and provide technical guidance, without which many of the experiments would have taken significantly longer. I would also like to thank Kodak Research and Development management for their support. From a technology perspective, I have had the great fortune to have collaborations with Dr. Cecilia Mascolo, Prof. Jeff Magee, Prof. Jeff Kramer, Prof. Wolfgang Emmerich, and Prof. Austin Tate. I have learned much from them, all of which has influenced my thinking and exploration in several areas in this programme. I would especially like to thank my supervisor and collaborator, Professor Anthony Finkelstein, who has provided excellent advice and support in many ways with great patience and persistence. I would also like to acknowledge the extra effort expended and excellent feedback provided by Professors George Spanoudakis and George Roussos, your comments and suggestions are greatly appreciated.

Finally and most importantly, to my husband and children who have endured many bad-tempered moments over the last five years with great humour and have continually provided encouragement.

ABSTRACT

The need for higher value, reliable online services to promote new Internet-based business models is a requirement facing many technologists and business leaders. This need coupled with the trend towards greater mobility of networked devices and consumers creates significant challenges for current and future systems developers. The proliferation of mobile devices and the variability of their capabilities present an overwhelming number of options to systems designers and engineers who are tasked with the development of next generation context adaptive software services. Given the dynamic nature of this environment, implementing solutions for the current set of devices in the field makes an assumption that this deployment situation is somehow fixed; this assumption does little to support the future and longer term needs within the marketplace. To add to the complexity, the timeframes necessary to develop robust and adaptive online software services can be long by comparison, so that the development projects and their resources are often behind on platform support before the first release is launched to the public. New approaches and methodologies for engineering dynamic and adaptive online services will be necessary and, as will be shown, are in fact mandated by the regulation imposed by service level guarantees. These new techniques and technology are commercially useless unless they can be used in engineering practice. New context adaptation processes and architectures must be capable of performing under strict service level agreements; those that will undoubtedly govern future business relationships between online parties.

This programme of engineering study and research investigates several key issues found in the emerging area of context adaptation services for online mobile networks. As a series of engineering investigations, the work described here involves a wider array of technical activity than found in traditional doctoral work and this is reflected throughout the dissertation. First, a clear definition of industrial motivation is stated to provide the engineering foundation. Next, the programme focuses on the nature of contextual adaptation through product development projects. The development process within

these projects results in several issues with the commercial feasibility of the technology. From this point, the programme of study then progresses through the lifecycle of the engineering process, investigating at each stage the critical engineering challenges. Further analysis of the problems and possible solutions for deploying such adaptive solutions are reviewed and experiments are undertaken in the areas of systems component and performance analysis. System-wide architectural options are then evaluated with specific interest in using knowledge-base systems as one approach to solving some of the issues in context adaptation. The central hypothesis is that due to the dynamic nature of context parameters, the concept of a mobile device knowledge-base as a necessary component of an architectural solution is presented and justified through prototyping efforts. The utility of web ontologies and other "soft computing" technologies on the nature of the solution are also examined through the review of relevant work and the engineering design of the demonstration system. These technology selections are supported directly by the industrial context and mission.

In the final sections, the architecture is evaluated through the demonstration of promising techniques and methods in order to confirm understanding and to evaluate the use of knowledge-bases, AI and other technologies within the scope of the project. Through the implementation of a context adaptation architecture as a business process workflow, the impact of future trends of device reconfiguration are highlighted and discussed. To address the challenge of context adaptation in reconfigurable device architectures, an evolutionary computation approach is then presented as a means to provide an optimal baseline on which a service may execute. These last two techniques are discussed and new designs are proposed to specifically address the major issues uncovered in timely collection and evaluation of contextual parameters in a mobile service network. The programme summary and future work then brings together all the key results into a practitioner's reference guide for the creation of online context adaptive services with a greater degree of intelligence and maintainability while executing with the term of a service level agreement.

TABLE OF CONTENTS

APPENDICES

List of Tables and Figures

Chapter 1 Introduction

Much has changed in the world of business and technology since the well publicized collapse of the Internet "boom" economy. Online businesses in particular can no longer expect to sustain their viability based on the number of potential customers who visit their site or evaluate their services. They must turn their visitors into buyers and those buyers into repeat customers for higher margin services. The online business climate is now driven by quick deployment of innovative services those that are enabled by computing hardware, software engineering, hosting services, and networking infrastructure. These enabling technologies continuously evolve to support the mission of attracting buyers to online services and products. Coupled with this change of business climate, new technology and capabilities are emerging which promote the ability of "many-to-many", real time commerce with services and products available and accessible across the networked world. Availability and accessibility has now become table stakes for online business. Success in this competitive environment is no longer about having the ability to reach the multitude of customers; it is about the measure of the value proposition that is offered to each customer. This value proposition needs to be composed of important and relevant business features which are directly related to the need of the consumer. In addition to these features are those non-functional elements, such as service level guarantees surrounding the business interaction. Elements such as response time guarantees, quality of service guarantees, security, privacy and ubiquitous access are all a part of the total proposition offered to the customer. The goal and objective for many companies is to maximize this value while maintaining cost-effective development and operation. Arguably, there are many ways to increase value through novelty and relevance; one way however which promotes the proven method of personalisation is the addition of contextual or situational awareness and adaptation to the overarching design of services.

This programme of research was designed to cross the spectrum of engineering challenges found in context-aware system development and to isolate areas for investigation and contribution. There are many research areas within systems engineering which seek to create generic methods and tools for context-based applications. Equally there are several sub disciplines of software engineering concerned with the usability, security and privacy issues with such systems. This current work complements these efforts by investigating the engineering challenges of context adaptation in service oriented systems. The programme goes beyond the current set of available solutions to identify new and novel methods and techniques that provide alternative architectures for context adaptation. Much like the systems that this research investigates, the programme is a continuous thread of system prototype to analysis; from technology to architecture to problem identification to feasible solution, with contributions along the way in the form of new architectural approaches and validated techniques for the practitioner. Even the most esoteric component of the research programme, that of the evolutionary computation approach to service and context mapping, is systematically focused down to a practical method for the prediction of potential context in engineered systems.

1.1 Industrial Context

Before the details can be presented, it is important to set the stage and clearly articulate the business motivation that provide the foundation for the entire programme. This business foundation reflects both the need and implies the opportunities for new innovation in online service provision. The programme is sponsored by Kodak Limited Research and Development as a part of the activity centred on expanding and growing its core competency in advanced mobile networking and software as services research in Europe. This effort was a part of a broader portfolio of activity in the following areas:

- Mobile networking for imaging services
- Image management in mobile systems
- Adaptive computing systems
- GRID-based Imaging services

The programme was funded for the purpose of providing technology and knowledge useable in a commercial setting but which included novel and academically-based features and components. The primary requirement of the business was that of a full suite of adaptive services that offered the Kodak subscriber a greater degree of value delivering optimal imaging services to any device with end-to-end quality of service. This suite of imaging services would be offered through partnership with appropriate telecommunication and Internet service providers. As the programme progressed, further requirements were placed on the services developed including the desire to investigate the inclusion of the appropriate aspects of computational intelligence technology into the overall service architecture. To summarise, the industrial setting for this work is governed by the following needs:

- A suite of novel imaging services for the mobile consumer
- Adaptive content and application delivery to the Kodak subscriber
- Revenue generating services which can be deployed through existing channels within a service level agreement framework
- Offering possible competitive differentiation through a foundation of computational intelligence techniques as appropriate to deliver the service

One of the first engineering challenges that must be addressed is how best to implement extensible service agreements in the deployment space. The future success and growth of the service provision business depends on reducing the subtle and intangible nature of the transaction in order to form a more concrete "contract". This contract is either be formed through the execution of a software application or through other paper means. In either case, an implementation is created which is used as a virtual sales environment and within it, there is contained a stated quality of service agreement. Quality of service or QoS, provides the transactional framework of guarantees which are critically important to a software as service business. A central assumption in the current work is that service level or QoS agreements are a prominent part of Internet development, thus

all engineering practice must take these agreements into account as a part of the design constraints. It will be required that new services adhere to the level of service which is agreed, and that the older services interact with those that are developed and that the entire service provision be predictable and reliable.

Providing yet another operational challenge to the goal of provisioning context adaptive and higher valued services, is the dynamic operational environment itself. The business of online service provisioning is built on an ever-changing software and hardware infrastructure. Providing such services with added value yet cost-effectively requires new methods and mechanism to build in that value that customer's will be willing to pay for. Additionally new architectures will be required to support the provision to the ever-growing number of devices that a consumer will be using within this networked infrastructure. The uncertainties of resources on the destination platform and in some respects, the attributes of the platform itself are significant challenges for every phase of the systems development cycle. Contextual adaptation is the ability to utilize information about the goals, environment, intention and preferences of the service requester in order to provide an optimal service within these variables [1]. At its most general, this notion is defined as the ability of the user to access any information over any network from anywhere through any type of client device. This statement forms the direct basis for the connection between the Kodak business needs as stated above and the use of context adaptation technology. The premise presented is that providing an optimal imaging service to any device with a defined quality of service is precisely equivalent to providing image-based context adaptive services within a service level agreement.

One of the most difficult engineering challenges in this area is to isolate and agree the appropriate definition of context which is important to the desired suite of services to be offered by the business, those that have the highest return on investment. It is also important with the agreed definition of context for the domain, such as imaging service, to subsequently fix the level of context granularity that is necessary to provide the value of the service. The assumption to be proven in the first phase of this research is that

context adaptation, as a service provides valuable features to add to an existing service and is a valuable standalone service in and of itself. Only be building prototypes of these services within model architectures can the extent of the value proposition be documented. In this work, the implications, performance and architectural alternatives of context adaptation will be investigated within a service-oriented environment and from a service developer/provider point of view.

1.2 Network QoS and Enabling Infrastructure

A network service is only as good as it is underlying infrastructure allows it to be. This has been proven in the mobile services area where the service developer is both constrained by and enabled by the network provider when it comes to the richness of the features that can be offered to the end consumer. Therefore, it is critical to understand the possibilities and limitations imposed on context adaptive services by the network. The Internet and its broad consumer reach have brought with it a wealth of new ideas regarding services to be offered. A key enabler for these new services is the expanding capabilities and intelligence being built into the very fabric of the network itself. This new intelligence can assist or detract from the operation of the network and will have particular implications of the performance and operation of context adaptation. Rudimentary forms of computational intelligence in the form of routine functions like optimal route discovery, bottleneck detection and avoidance [2] are commonplace in today's engineered networks. Much research now is focused on increasing the network capability beyond these functions and into the application or service domain itself by instilling even greater computational intelligence in the network components. This technology direction is constrained however by the need to provide the services within a specified level of quality or response time in order to satisfy the customer agreement as previously mentioned. Creating an intelligent network to provide services which cannot meet the demands and contractual obligations that are needed for a transaction is not a viable business model. The objective is to engineer a level of intelligence and computational capability sufficient to provide the higher valued services required by the service level agreement management framework[3]. This is a foundational tenet for context adaptation and one of the design elements which will be defined in greater detail

in subsequent chapters. To operate services within a service agreement constraint requires that some common understanding of what the constraint actually means to the interaction is necessary. Quality of service in the internetworking environment is a term most often used to denote a software and hardware solution which effectively manages the resources of the network in order to meet a level of service specified by some agreement. This traditional definition has not often considered the quality of service apparent to the user or permitted through the architecture of the application itself [4]. This is an interesting artefact of the telecommunications business environment where the service level agreement was a legal contract between the service providers within the network and was not necessarily a customer facing document. However, the future value of network provided services depends on delivering high quality to those willing to pay more for this privilege and as such, the fundamental constructs of service level agreements will inevitably need to make their way into the customer arena for service differentiation to become a revenue opportunity.

This research programme will be set within a QoS environment where there is a degree of "regulation", or a set of operational constraints enforced through a legal agreement, over the conduct of the services within the networked environment. This current work will look at challenges imposed by working within a regulated environment which sets specific performance attributes. These constraints pose a realistic framework and one which is encountered often when developing systems for commercial use. Currently in engineering practice, the formality of a service level agreement is often represented by the indirect customer retention statistics; services which cannot deliver as promised within an acceptable service level, meaning service at all levels from network through to application, are not viable and certainly will not drive a consistent revenue generation scheme. This indirect metric of QoS is insufficient to force real requirements into the service engineering process. It is an important to make the QoS constraints explicit within the engineering design process of value added services.

1.3 Context Aware Adaptive Architectures and Revenue

Both research results and general consensus suggest the need and possibilities for context-aware computing, specifically in support of new mobile platforms. It should be equally clear that context-aware and context-adaptive services can bring a new value to the online service consumer by providing highly personalized or situation appropriate features. An important analogy to this can be found in the assemble-to-order manufacturing sensation which brought a new sense of the personalized product to the PC computer market through Dell Computers and others. By correlation, context-adaptive services are the service analogues to the product-based assemble to order concept. Before continuing though, it is important to outline the definition of and difference between context-aware, context-sensitive and context adaptive. These terms are often used repeatedly and sometimes interchangeably as a result of their subtle differences. Context-aware and context sensitive represent software architectures or deployments which identify the environmental conditions and which may provide the means to alter operating parameters changing behaviours in response to some set of conditions. A context adaptive system goes a step beyond this to the proactive collection, evaluation and goal-driven adaptation of the application itself or the content of the application or in some cases both. In this work, the terms *adaptive context-aware* and *context adaptive software* are synonymous; as those application systems that provide the means to deliver a truly unique experience to the service requester which is optimal for the time, place, goals and preferences. The basic premise given is that the value of such an experience is greater than that found in a standard software as service. These architectures provide valued features to the overall service, but the cost of the increased value is the increased effort in engineering and performance issues. These key issues of context-based media adaptation in pervasive computing are well summarized in [5]. The issues and implications span the range of disciplines in both systems and network engineering. From understanding and implementing the right information model to representing the essence of the adaptation to be offered through to the understanding and proper handling of the current and future capabilities of the myriad of devices encountered in the deployment of the services to a global market. It was this latter issue

that was the prime motivator for this research programme and this aspect will be another thread of the investigation which reappears throughout the programme.

Yet another aspect of engineering context adaptation into the online service is the changing trends in software engineering field itself. As mentioned, the tools and technological components will change dramatically in ever decreasing timeframes. Of specific interest to this programme of research is the movement toward fine-grained service composition making logically independent services operate together in order to deliver one complete consumer solution [6]. It will be shown that an example of such a composite service approach is possible for context adaptation, a fact which will be outlined in detail in the subsequent chapters. There are a multitude of services that might be offered which are composed of still finer level services, creating a complex interaction network of services brought together to fulfil a customer need. This "dynamic assemble-to-order" approach to delivering software and services is a challenging software engineering problem and one which will require the further development of structured approaches. Contextual handling of requests will not override quality of service requirements, but must be made to fit with the regulated environment which is formed by the QoS contract. This additional requirement increases the difficulty still further, making this a problem formulated with many competing objectives each with their own set of goals and constraints. Balancing all the parameters of adaptation within an agreement framework requires intelligence and negotiation. Current software engineering practices in industry are not necessarily designed to handle the cognitive science issues which may be inherent in the use of context-aware adaptation architectures. Collection, interpretation and adaptation across a set of highly variable parameters in the span of a controlled time allocation is a relatively new commercial engineering challenge and one without time tested techniques to rely upon. Context-aware adaptation is most likely one of the first application techniques, outside of scheduling and process control, with large commercial value which requires the use of concepts taken from the intelligent systems areas of computer science.

1.4 Programme Objectives

The primary objective of this programme of research is to investigate methods and mechanisms to improve the design and engineering of mobile context-adaptive systems, and in particular mobile imaging services. However, it is not possible to cover the full spectrum of issues in this emerging technology arena, so in order to focus the work and deliver a measure of novelty, the programme is centred on three core objectives; performance modelling of context-adaptation subsystems, architectural suggestions for improving context-adaptation performance and finally, a case study of the design of an context adaptation service. Additional topics of importance to "future proofing" the results are discussed in the area of the trend towards reconfiguration and the use of evolutionary computing for implementing context requirements to capability mapping. A diagrammatic view of the thesis is provided in Figure 1-1 below:

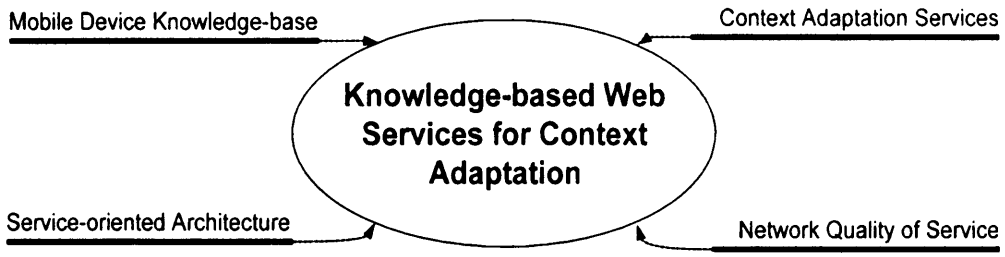

Figure 1-1: Programme Overview and Key Areas

In order to meet these core research objectives, it will be necessary to provide supporting information including definition of contextual attributes, proper description of capability versus raw attributes and so on. It will also be required to show the higher value of context-based services and that they are in fact, largely dependent on the device features and capabilities. A final supporting objective will be to show that there is a mapping or correlation between the devices features and the resulting context possibilities afforded by those features. The transformation from features to context will be enabled through the use of a persistent repository and a set of service relevant categorisations and translations. The repository and its translations are then made available to other

developers who may use their own domain specific problem solving methods in order to create the next generation context adaptive applications.

The overriding objective in an engineering focused research project though is to be able to produce tangible work product from the various investigations, evaluations and experiments.

> *All rules for study are summed in this one: learn only in order to create.*
>
> *Friedrich Schelling*

Although this is perhaps overly secular in nature, it is with this statement in mind that the technical objectives were formulated. A series of explorations are motivated in part by the technical objectives, relevant research questions and by the subsequent engineering issues encountered. A continuous thread of issue—analysis—study—design—prototype is followed in each of the engineering studies. This will be shown in the subsequent chapters where the reader will find the performance modelling includes mathematical modelling of the context adaptation subsystem after a formal tool selection, the alternative architecture includes the implementation of the device ontology knowledge-base with rules specification and the final component includes a full working prototype of the process-based, composite web service for context-adaptation and an associated case study. Through these different avenues of the research programme, all of which are intended to be reusable, the utility of the engineering effort is coupled with academic foundation. The outcomes of the programme are manifested in new architectures for context adaptation, concepts for reconfiguration support for future dynamic services, methods for design visualization of the reconfigurable services and finally, a business process execution language model for context adaptation developed using the new concepts outlined.

1.5 Thesis Structure

The thesis is organized starting from the description of the service architectures currently in practice and the issues with engineering context adaptation systems in these environments. Quality of service guarantees implemented as service level agreements are then introduced as a constraint for future engineering efforts of context adaptation particularly as it is implemented in cross layered architectures. Results of the analysis and investigation in this area are outlined and provide the motivation for an architectural alternative study. An alternative is formulated to meet the requirements and incorporate intelligent systems elements providing a new mechanism and thinking for context adaptation. Using this architecture, a prototype is designed using the business process execution language and web services programming paradigm. Along the way, new concepts and design considerations are investigated such as the nature of future services architectures in the environment of reconfigurable devices. In the final sections, a discussion and some analysis of potential methods for creating predictive models for device specific context using evolutionary computing theory is presented. The motivation for such a predictive model is a direct consequence of the commercial experience of context adaptation services development within the Kodak Limited Research and Development laboratories during 2002.

Chapter 2 – provides a survey of a multitude, cross-sectional research efforts in the area is presented. This chapter review the findings and analysis of an extensive literature review of the significant published works of study, practice and thought through the major topic areas of the programme.

Chapter 3 - provides the foundation for key elements and concepts to be investigated within the programme of research. In Chapter 3, a series of key concepts are defined and demonstrated. First, the definition of the context adaptation is presented as well as the key supporting topic of QoS regulation. A description of two context adaptation demonstration systems are then described in detail with extended analysis into their faults and opportunities for improvement. Finally, the use of network of queues

modelling for diagnosing issues with the presented context adaptive systems is discussed and further defined.

Chapter 4 – presents a novel architectural alternative of a knowledge-base is used to hold the evolving feature information for each device encountered in a context adaptation service deployment. The chapter covers the basics of knowledge engineering, outlines its relationship to ontological engineering, and discusses the selection and development process for the knowledge-base as well as the method for validation. In Chapter 4, the method of using the technology standards process as a proxy for the knowledge engineer will be discussed as a viable technique for incorporating intelligent systems into commercial applications.

Chapter 5 – provides the validation of the concepts and architectures developed within the programme of research, Chapter 5 will describe the reduction of many of the concepts discussed into actual practice. The business process execution language (BPEL) for web services will be described, and how it is a relevant method to be applied for instantiating the concepts developed in previous chapters and what advantages such a methodology might afford the engineer will also be presented. Also in this chapter, the details of the prototype requirements, design, development, and its subsequent evaluation including the description of a mobile device case study using a simple adaptation scenario will be presented and discussed at length.

Chapter 6 – describes and presents analysis on the future technology implications and opportunities of context adaptation web services. Questions on how future computing architectures at the lower layers of the systems architecture will impact the proposed context adaptation system are reviewed. Of specific interest is the emerging notion of reconfigurable devices and their impact on how service developers will need to adapt design and engineering techniques to accommodate reconfiguration.

Chapter 7 – discusses the issues with context adaptive systems where there is required an optimal match between the elements of context and the needs of the service

requested. Chapter 7 will cover an innovative approach to this problem by using co-evolutionary computation and techniques to evolve a set of device configurations designed to meet the service requirements.

Chapter 8 – provides the summary and unification of the themes and discoveries within the programme of work. As a series of engineering investigations, this work requires focus on how the results support the industrial context. In Chapter 8, the overall programme conclusions and learning will be presented in an integrated manner. The chapter will also review the beneficial future work based on the engineering results.

Chapter 2 Relevant Work

A review of the relevant work in a traditional doctoral programme digs deeply into the chosen topic and explores all aspects of the singular problem statement. Conversely, in an engineering doctoral programme such as this, the review is conducted differently as the problem statement is broader and generally less focussed on a singular advance in one area, but more on the advances towards an overall practical solution. As a result, the review can cover several major areas of study representing the many sub disciplines of the systems engineering field. For example, the literature collected and analysed throughout this study contains more than three hundred references spanning four decades of computer science research in seven different topic areas. This chapter describes the literature through the use of the key topics and categories of interest. Within each topic area, important literature from both academic and industrial researchers is reviewed, discussed and the most prominent research in the area is selected for further analysis.

Context-aware adaptation services

Central to the main thesis of this programme is the discovery and solution of basic engineering challenges in the area of context adaptive services, particularly those found in the mobile computing. To do so, requires an awareness of how context is defined viewed, handled and managed in the software development and research. Context awareness and adaptation as a computing research topic has amassed a significant and diverse set of literature, outcomes and publications. But as stated in the introduction to a special issue devoted to context, "Context awareness is fine in theory. The research issue is figuring out how to get it to work in practice" [7]. This statement pervades the literature in this area as new architectures are brought forth to address the complexity,

the issues of security and robustness, and to describe how best to retrofit hardware to collect valuable contextual data [8]. There is little debate in the literature about the value of adding information about the many environmental, situational and very personal aspects of the user into the delivery of networked services. The influential work in the description of context was written a decade ago and still reflects the needs that have yet to be met in any reliable way today [9]. The importance of context awareness is apparent in the literature on the future potential of Internet itself. The recognition that a new future Internet architecture will be required to provide the type of adaptation that real-time service customers will require was published in 1995 [10]. Over the last ten years, mechanisms, architectures and application frameworks have been researched, designed and developed, yet there still is no clear or dominant technology available to the developer today. In particular, the large commercial systems and software providers focus their research toward the creation of tools and technology that provide a saleable platform from which to develop services that allow universal access to collections of media assets [11-13]. Both the computing popular media and industrial researchers report on context awareness as providing motivation for greater intelligence for distributed application and in fact, while this may be true, the most often cited means of implementing context-awareness involves a server directed approach, one which uses a number of message exchanges to negotiate functionality and involves a table lookup of possible attributes on which to adapt the service to meet the user's needs. This approach, advocated through a standards point view, uses agent profiles and a capability profile message exchange. This current approach of contextual message passing provides basic functionality through the use of a static statement of capability and the return of a response from an interacting service at the other end of the message. The need for an increase in application intelligence to support context adaptation is directly addressed in this work where architectural elements taken from the AI field are implemented within the adaptive process flow and the use of a systems architecture composed of a centralised knowledge-base with reasoning components is proposed.

From the collection of literature reviewed, it appears that there is a gap in research reported on the concept of including computational intelligence within in the [illegible] of handling context adaptation in future services. However, it is also clear that computational intelligence impacts the performance of systems by adding expensive computing algorithms to the overall process. Tools for contextual systems and services development are emerging, but methods and mechanisms are still forthcoming with academic and government sponsored project efforts resulting in new forms of context aware possibilities. Little is available on the performance implications of context adaptive architectures using computational intelligence although anecdotal evidence is available as a by-product of architectural building projects. Further still research outcomes on the methods of delivering context adaptation within a performance constraint are not to be found. There are numerous architecture projects around context-awareness and adaptation [9;14;15]. These papers provide detailed design and plans for extracting context from sensors, use of the collected context and providing insight into the difficulty of interaction modalities with contextual objects. In the descriptions of research work that investigate performance aspects, it is the performance on the context attributes acquisition that is most often the focus as in [15], not the performance of the system as a whole. The issues of using and then evaluating the effective use of contextual information over a long run history of the system are not investigated or have not been reported. Even in the more industrially directed literature, there are discussions on the opportunities to make applications "smarter" through the use of a multitude of context variables and how context might be integrated or implemented into traditional software architectures, but little if any reference to the practical operation and performance implications of such systems is presented [1].

While results in the area of performance and engineering evaluation are not plentiful, relevant work on systems for the aggregation of context are available. This literature is motivated by and forms the foundation for content caching strategies which has made its way into commercial practice. The implementation of forms of aggregation of context as an architectural requirement describes the issue with a multitude of dynamically

varying contextual variables in a commercial system. Methods for handling the aspects of context aggregation are found in [16] which introduces the need for context aware structures in order to support the notion of "ubiquitous" computing and they propose an acyclic graph mechanism to manage the context through its lifecycle making sure that it arrives at a computational node to be used in the adaptation step which is the action step in the context process cycle. The aggregator gathers all the information about a specific entity from all the sources or sensors "behaving like a proxy to context for applications" [17]. This input context represents various definitions and levels of information from embedded locational and movement sensors to historical usage profiles representing network access, human interaction and activity level [18]. One of the more relevant papers in this area is a discussion around component-based architectures for integrated and extensible context-aware systems by [19]. While this work begins the process of analysing the issue of performance, it falls short of actually reporting any aspects on the trade-offs of performance in context-awareness as a computing style. Architectures are also designed, prototyped and tested from the adapted content perspective, meaning that the acquisition of contextual information is with the primary purpose of modifying content to suit the target or destination device. This is particularly true for those research groups which are interested and experienced with dealing with digital multimedia or other rich forms of content. The recognition that the delivery of optimised content in itself increases the value of the service tends to be the motivation for this segment of the context adaptation research community. Researchers in this area focus on transcoding, defined as the transformation of content, and structural layout modifications formulated in response to a specific request given a specific situation [20]. This style of context adaptation centred on content rendering is the same as that designed within Kodak and was in fact, the partial motivation for this engineering research programme. The key issues with content adaptation in this manner, as will be outlined in detail in Chapter 3, are its lack of openness, flexibility and extensibility.

Beyond the notion of aggregation and caching as a context adaptation architecture, is its use and management throughout the service delivery lifecycle. Many of the content-

based adaptation schemes use specialised rules for adaptation which are customised to the business model of the service to be deployed. These rules on adaptation are tightly coupled to the context variables and the nature of the service design, which means that minor modifications to the variables or the service necessitate careful modification of the rules. To help alleviate this problem, several researchers introduce the concept of a context-aware middleware to abstract the details of the service, context and device so that the overall system is more tolerant of change. Several middleware architectures are available from the research community and in particular, the QoSDream architecture from [21] which supports the collection, usage and management on contextual components using an event-based paradigm. Their architecture is somewhat unique in its incorporation of a quality of service management subsystem for ensuring service consistency when target configuration changes are required. Taking advantage of middleware for context adaptation is the recent work to support mobile, context-awareness which essentially extends the support for the unreliable routing, connection and unpredictable bandwidth found in networks of mobile devices. The interplay between mobility and the need for adaptation is clear; the computing platforms most often found in mobile environments are also those that require the most extensive adaptation support. The market demand for services in the mobile communications segment is also well established and recognised thus promoting the need for scalable, robust and deployable service architectures. Consistent service delivery coupled with temporally variable resource availability provides a few of the key requirements for most of the context adaptation architecture work in this area [15;22;23]. Several mobility middleware solutions involve the implementation of a relatively fat client and directed adaptation on the server. The reason for this approach is the need to collect from the client device all elements or forms of possible contextual variables and make those available to the server, so that it may adapt against these relevant components [24]. This approach proves to have serious implications in a more dynamic setting where time sensitive capability is important. This will be discussed in Chapter 3 using the evaluation of the adaptation process as a series of network queues.

Other architectures are also seen in the relevant literature in the area, of particular note are those using mobile middleware that partition the functionality of the service requested based on the available capabilities of the known target destination device and allow the user to interact with the service in a proactive manner [25]. This style is an improvement on the fat client approach for mobile devices, yet the concept of asking user to select the appropriate adapted content from a series of pre-defined and formatted execution links does not use the machine intelligence and user history to formulate automated context adaptation. It instead depends on the knowledge the user maintains about his/her own device capability. Understanding the requirements that mobility support and management imposes on context adaptation is the subject of several papers, in particular [26;27]. In eliciting and formulating requirements for context adaptation, the two principle challenges of device level dynamism and service complexity are clearly seen and uniformly reported. This is consistent with the realisation in development and testing of the performance issues with such systems; the context changes too quickly to properly adapt and deliver the desired service. Secondarily, the extent of the contextual input variables is often ill-defined or too great to efficiently create appropriate adaptation directives on the timescales permitted. A particularly important requirement which forms the basis for the research described in Chapter 4 of this work is the inherent lack of a "fixed" standard deployment platform and the implications of dealing with this in an intelligent manner over longer time scales. Basically, the trade-off is that by not fixing the target device as is normally done in traditional systems development to reduce complexity and improve performance potential, another static component in the design is necessary to maintain the details of the now non-determinant deployment environment. The importance of device specific context is a critical parameter for accurate context adaptation services, understanding the situation and the device means that opportunities for improvement can be implemented in real-time within the service level agreement[28]. The addition of a service level agreement as a core component of a context adaptation architecture providing further constraint on its operation is often not stated in publications on context adaptive network research. The difficulty with this approach however, is that architectural design and fundamental

technical approaches are altered as a result of having to meet specific legal frameworks and this will affect the utility results of the original reported research.

As with all research, problems and opportunities should form the foundation on which the work is based. One central problem of context adaptation is complexity of interaction involving the dynamic and complex interaction between the physical world of hardware, the virtual world of software and most importantly the unpredictable world of human users. The first step in understanding a truly complex engineering problem is often the creation of a simplification, usually in the form of a small scale model. This process is consistent in the traditional engineering disciplines such as civil engineering where understanding the solution is a series of physical, architectural drawings followed by a 3D model. This process of modelling and in particular modelling network systems is well-established through its roots in planning for large and expensive telecommunications networks. It is a step often left out of fundamental research studies as they have no working system to model as a starting point. There is significant impetus and benefit for starting a research project with a problem posed in the form of a poorly performing system, and using modelling as a tool to outline the most prominent issues as it provides an immediate focus for engineering research. However, it is well documented that performance in distributed network systems is difficult to properly assess [29]. This is particularly true for application services which are run across the public Internet. It will be subsequently presented that the network of queues theory fits very well the process of context adaptation and offers insights into where the overall process is most vulnerable to bottlenecks and other performance issues. The work in queuing network modelling is a subset of the broader field of queuing theory, where networks of queues are analysed through the application of probabilistic and formal statistical methods [30]. The foundation for these theories as they apply to computational networks was established in the late 1950's and early 1960's. Of particular note is the work by [31], whose network studies and treatment provided the foundation for performance analysis of networked computing nodes that could be represented as a series of queues, where the input to the queues can be statistically characterised and the execution times within the system are known on average. These

networks have had problems with consistent and inexplicable results under certain events within the network, in particular one relevant in this work, the condition of network overflow [32]. Still the simplicity and well-established theoretical basis coupled with the direct applicability of the concept of interconnected queues to the design of context adaptation makes this analysis relevant. Using this approach as a means to determine performance issues and to outline the concurrent behaviour of software systems is known and documented, and it is increasingly being used to evaluate software architectures prior to development [33]. This more recent adaptation of the traditional utility of the queueing network model (QNM) methodology provides the motivation for its inclusion in this engineering programme.

Cross Layer Networking for Quality of Service

Modelling network performance requires that a set of assumptions be made concerning the underlying infrastructure and network engineering itself. Just as trends in systems engineering change rapidly, so do those found in network engineering. One of the most important recent trends in this area is in the use and power of cross layer network information. The concept of cross layering involves the intentional interaction between the architectural layers of the network model which had been previously specified through the Internet standards process as distinct. The distinct and segregated function of the layers made for a reliable and consistent stateless operation which suited well global access to [illegible] ormation retrieval. With the advent of new applications and the motivation of the Internet as a broad service distribution channel, the network needed to evolve to accommodate the new requirements for real-time, rich media services that require state and knowledge from across the different layers within the network. A particularly important motivation for knowledge sharing across the system layers is that of quality of service and specifically, service differentiation. Those revenue generating network services which offer increased value through experiences such as context adaptation will need to be able to utilise information found in deeper layers of the

network. For example, a service will need to understand real-time traffic conditions and device battery resources in order to complete the delivery of a service in a specific amount of end user clock time. The work by the GRACE team at the University of Chicago at Urbana-Champaign contributes directly in this area [34;35]. Their approach is to investigate the cost versus utility of the service where cost is the resource expense of adaptation including battery, bandwidth and reconfiguration against the total utility of the service in terms of meeting the quality of service and importance of the service to user [36]. In their work, all layers are involved and in this global adaptation all must cooperate to achieve a consistent and cohesive service delivery. This is in part true although service engineers can isolate specific information needed from one layer and then adapt their application functionality to meet a set of constraints pertaining to the network layer without actually modifying the network layers themselves. This is the approach taken by [37] where the importance of network status information requires a separate module that is responsible for interlayer communication of status messages only, thus preserving to the separation of system architecture layers model. While this may limit the future extensibility of the service, it is not often the case that all layers will have impact on the delivery of specific solution so forcing a global adaptation which incurs some severe performance penalties may not be the most effective engineering solution. In engineering a service-based solution, cross layering is based on the information exchange needs that will enable the enhanced delivery of the service so only those layers that are actually required are involved. This is the approach taken in [38], where the requirements for the provision of multimedia application services over a wireless link drove the specific selection of the layers that will interact to form the solution [38]. In this case the application, link-level and transport layers are connected via message exchange to provide error protection and bandwidth estimates to the application layer for its use in adaptation to meet quality of service levels. In both the work of the [36] and in [11], the adaptation required a centralised intermediary controller, either a proxy or a resource manager to coordinate and manage the adaptation. An important observation taken from the GRACE results are their finding on global versus local adaptation. Global adaptation being the server directed adaptation where the system assesses the needs and the nature of the new service before

submitting it for execution. Global adaptation is found to be feasible in their initial results given the use of a centralised resource manager to manage the application adaptations. The local adaptations, those that occurs within in each architectural layer, are found to be too resource intensive to function properly which is consistent with empirical work done in this programme and outlined in detail in Chapter 3.

It is instructive to look at how cross layer design is a feasible approach and what are the architectural implications of its use. To understand this it is necessary to understand the direction of networks, and specifically the trend towards computational intelligence in the nodes of the network. This increasing computational intelligence means that through the interaction of the application layer with the network and transport layer, current information which is reported by these newly intelligent nodes can be utilised by the application. The programmability of the network nodes gives them functionality that enables the cross layer design to occur and in many ways, supports the greater possibilities for service engineering beyond what is possible today. Programmable networks allow, through the controlled use of application programming interfaces, modifications at runtime to support traffic management procedures at the active nodes within the network [39]. The range of enhanced functionality available through the APIs can be as simple as a route forwarding mechanism and as complex, as the context aware, QoS-based adaptation that is mentioned above. Without some form of programmable or intelligence in the node, cross layering is possible but network related adaptation is not. For example, modifying the nature of a load balancing route operation within the network requires the ability to send a form of mobile code to the routing infrastructure to update the management policies to match the current service level offering. Many researchers separate the concept of adaptive software at the application level from that which operates at the network layers primarily it seems because these are traditionally two different types of research groups, those working at the lower network level and those working in the application area. Research in applications have largely been client and server systems and architectures supporting adaptation [40;41], whereas in the networking research community the focus may be on active nodes and mobile code distributed to computation nodes [2;3;42;43]. There are several different approaches

between these researchers from middleware for management of programmable networks in [42] to the work found in [43] on a unified architecture for real-time applications over packet networks. This latter work alluded to many of the issues that are still present today, that of delay, service commitments and admission control which is notable as this paper dates from 1992. The need to understand the complex network architectures required to support new applications services tend to focus on two elements of the system, the load on the network and the integrated ability for the application and the network to balance the service quality against the predicted or detected load [44],[45]. This is the early picture of context adaptation which does not mention cross layer design or even the requirements of reconfigurability in the destination devices. As application developers and those interested in service engineering start to investigate how to utilise the network more efficiently and flexibly the notion of standard access to cross layer functionality starts to appear in the literature.

The notion of cross layering in support of enhanced services unifies the domains of active networks and context adaptation. Cross layering however, inherits fundamental issues due to the fact that it is essentially altering the foundation of the Internet technology approach. First, it breaks the standards for internetworking, which introduces concerns about the creation of closed networks and creating proprietary sub networks designed for clusters of users who are able to execute the now specialised application and network layers. Cross layering for adaptation within an active network also has several potential issues regarding performance and security which are frequently cited as the major constraint against the deployment of truly intelligent computational nodes across the Internet infrastructure. To date, these issues have not yet been resolved in a standardised way and the interest of the research community now appears to have moved onto service-oriented mechanisms for achieving new value-added online offerings. Architecturally there is a multitude of diverse approaches to implementing context adaptive network services in programmable networks, but the difficulty is in gaining agreement across the consortium of partners required to actually put such a working system together in a commercial setting. Overlay networks, open application programming interfaces for application providers, and a host of standards

from the IETF and industrial consortiums like Parlay all have had successful prototypes and proof of concept implementations in adaptive systems starting back in 1999 [46]. Many of the proof of concept systems are motivated by and demonstrated using multimedia applications which require real-time constraints on transmission and which many feel are complex enough to have a greater value to the consumer therefore the cost of service can support the increase in expensive infrastructure. It is still unclear how and when cross layer applications supported by the associated adapted networks will reach the mainstream commercial market.

The technology of active networking and cross layer design can be tied to value-added services through the need to accommodate some level of quality of service as mandated by a service level agreement (SLA) [3;47]. Much of the published work in this area cites the need for management and monitoring of service levels as a key requirement for programmable networks. This is particularly important when the business model of online service is predicated on an increased value of service through a more complicated and complex application solution. The service level becomes a critical factor for new services and in particular, the SLA is important in the "software as service" subscription model where user's purchase application functionality at the time of use, and on demand. The literature is relatively sparse at this time in the area of technical research into service level agreements and their verification as it is a broad and difficult business issue involving many partners, transfer or exchange of value and automated negotiation. These elements make the technical problem challenging to define specifically, and even more difficult to codify in any consistent or universally acceptable way. There is work in this area which attempts to define an ontology of terms and a structure set of actors and vocabularies [48-50]. In [50], the focus is on a framework for implementation of a service level management system where the detail of the service level is simplistic. In this work, it shows empirically the direct interaction of admission and route control within the network directed by service level agreement through the use of policies. The details and structure of the SLA are the primary discussion point and its implementation is intended to be through integration with a programmable framework as found in [42;51]. The work by [48] concentrates on how to apportion the details of the service level constraints

through the layers of the network in order to make an end-to-end service level contract which represents the intent of virtual agreement between the user and provider. It is clear that the business of adaptive service deployment is a complex one if it is to be driven by a standardised individual service level agreement instead of a subscriber bundle. As well as QoS mechanisms themselves, there is a fair amount of attention paid in the literature to the user perceived quality of service. It is important to recognise that in many cases, the numeric or deterministic operation of the service level will differ from the user's perception of value from the service. The implication is that the true value is a variable mixture of known service parameters and the user's experience of how the service was delivered [4]. Some studies look specifically at transmission rates and service utility as representing QoS from a user perspective which is most likely true for some simple application or even for voice telephony [52], but clearly the diversity of complex, rich media applications with higher revenue potential will have a QoS composed of the many context sensitive variables associated with such services.

Knowledge-Base Systems for Open, Standard Context Management

Bringing computational intelligence into any engineering research programme brings with it a vast amount of literature and prior work. The fascination with the fundamental details of human knowledge from acquisition to manipulation and into validation and visualisation is astounding. This is perhaps due to the early desire by computer scientists to use the human thought process as a model for computing and by doing so, the requirement to understand and apply this information to computing was heavily researched and published. The approach taken here is twofold. First to review the broader, general understanding of computational knowledge management and then, based on this high level understanding, to select specific relevant work based on the more directed engineering questions to be answered. A broad perspective on knowledge engineering is found in artificial intelligence texts, most of which provide the foundation for the terminology and general principles that are accepted in the AI,

cognitive science and in some case the psychology communities all of which are interested in promoting varying views of learning, knowledge and how to automate the process [53;54]. The research community and authors are generally either apologetic or defensive about the state of AI and knowledge engineering given high expectation of commercial applicability have yet to be widely demonstrated. Practical applications are outlined most in more recent publications and there is a natural synergy documented in AI texts around the practical use knowledge-bases formed from well-defined ontologies and using mobile agent code as the functional interaction and execution layer. However in many cases, these same texts seem to strive for an all or nothing approach. This means that it is often thought to be better to fully automate the acquisition, reasoning and decision-making processes computationally as a measure of success of AI. This is quite a high expectation for a computer system and so the practical application even using partial results appears to have been limited until recently. The semantic web has brought machine reasoning, ontologies and knowledge-base systems to the forefront of development although these research topics have been continuously studied in the last decade [55]. The concept of the Internet as a vast knowledge-base where an individual can contribute, rate and maintain information is well known although there is mention that the quality of the information is occasionally in doubt or at least subjective. A resurgence of interest in global knowledge acquisition through a network of expert contributors is common in the literature [56-58]. Contributory knowledge acquisition is a reality as is evidenced by the work by Singh, et al at MIT in the project called OpenMind Common Sense, which uses the input of the public and automated knowledge validation through questioning to build a common sense knowledge base [59]. This project demonstrates the willingness of many to contribute knowledge or to correct knowledge through validation questioning processes in the interest of improving a repository with no other compensation. This basic premise of a common repository for globally important information, in one case that of common sense, and in the current work, that of contextual variables, is an underlying theme found in this work.

A shared, open repository may provide the cornerstone for efficient retrieval of complex information, but it is not necessarily sufficient for dynamic adaptation parameter

storage. Acquiring knowledge through an automated means is a secondary step in the knowledge engineering process. The primary activity that is necessary before populating the knowledge-base is defining the structure and vocabulary in use with the domain of interest, an activity referred to as ontological engineering. This step is universally accepted although its prominence and importance in the overall process varies depending on the research group focus. For those in cognitive science, the understanding of the knowledge structure, axioms or connections between concepts is paramount. To those in computational intelligence areas of systems research, it is usually a step within a greater overall knowledge systems engineering process [60]. The process of developing a domain ontology provides concept organisation and forces an agreed terminology between stakeholders, it can represent the agreement on the description of and relationship between concepts of interest [55]. Ontological engineering as the basis for a knowledge base development is incomplete as it is impossible to describe every possible axiom and every possible concept within a domain unless it is one that is artificially small. To this end, the process is iterative and continues throughout the life of the system. Validating the ontology is therefore equally difficult as there is no conclusive answer to whether the concepts are fully represented or well represented although there is work in the area around more formal methods for this validation [61]. The importance of valid and structurally correct ontologies directly impacts on the efficiency of information retrieval which is a key issue for commercial application of knowledge-base systems [62]. While this is mentioned in several areas of KB systems development, there are little practical results published on the improvement of the information retrieval. Even structural validation of ontologies becomes increasingly more difficult as the number of concepts and relationships grow. With the use of visualisation techniques, the process of unifying, building and maintaining ontologies can be enhanced, however it is doubtful that for any real domain, such as the medical informatics domain, that a visualisation mechanism is an accurate method for validating the concepts in the ontology at anything but the highest level[63]. This is counter to published results in [64], but much work in the literature is done in an isolated or a subset of an actual application domain. There is essentially no completely correct ontology for it completely defines a domain in its own precise frame of reference

and this fact is in part why they are difficult to use in practical engineering applications [65].

Using computational intelligence techniques in context adaptation is a concept motivated by industrial need and interest in this cases, and by the requirement to deliver a more computationally intelligent set of architectural components for context adaptation. Knowing the basics of knowledge base systems does not define the domain specific knowledge needed by the system, as there is still additional expertise required in the content area itself. It is often necessary and efficient to rely on other means to help with this process. Through the technology standardisation process, a measure of correctness is possible in content knowledge. Standards bodies engage experts for the purpose of defining and refining terminology, concepts, components and systems. This is where much of the recent semantic web work appears in the literature through standards bodies. The semantic web standards process has resulted in several iterations of ontologies for services which are openly debated and ultimately agreed by legions of technical experts. The underlying desire in this case is to allow automated processing between machines in the network through the use of intelligent services, such as matching specific context attributes to specific adaptation actions. For semantic services to work, the machines in the interaction must have a common language just as humans require one when interacting amongst themselves. The basis for this common language is the service ontology and for the interaction semantics it is the process model [66-69]. This is an active area of research as the service-oriented approach to networked systems continues to emerge. One of primary issues with the direction of service ontologies and capability description is the degree to which a system engineer is required to specify such attributes. In one simple example of a single location aware service, the descriptive code required to describe the service is more than fifty lines of text-based XML code, a mistake in which would render the service useless in a semantic network [70]. Another important aspect of knowledge-base systems that is often mentioned in the literature, is that the structure or ontology is only part of the complete picture. It is the reasoning that operates on the concepts and the constraints that control the possible reasoning which in turn makes the knowledge-base a decision-making

systems component. This is where simple knowledge becomes intelligence given rational decision-making ability [71]. It is critical to note that many of the same issues with context adaptation are also issues in computational intelligence. The fact that making an intelligent decision in a computing environment is based on many contextual parameters, in particular the situational environment coupled with the goals and desires of the entity attempting to make the decision. Even with the same set of structural concepts, given a different situation, the reasoning outcome may be quite different. This is consistent between both knowledge-based systems and context adaptation systems which is one reason the study of one implies the other and to use one for the implementation of the other assists in the further understanding.

A central topic of study in the use of knowledge-bases in engineering is the selection of the proper knowledge representation. It is also a critical design factor in these systems, so it is not enough to know how to structure and define terminology but how to represent the concepts, axioms and relationship is a central engineering question. Knowledge and concepts can be organised as networks of patterns, hierarchical taxonomies, or frames which represent a subset of semantic networks [72-75]. The foundations of knowledge representation from an engineering perspective must be relevant and applicable to the nature of the semantics of the domain in which the system operates. In a very practical sense, the representation used can be a decision implied by the use of tools and techniques necessary to create the solution itself, rather like the physical form possibilities that an architect accepts as a part of the decision to use a specific material in the design of their structure. In summary then, representation and structure are the infrastructure elements that enable the machine learning and reasoning process in knowledge engineering. The computing parallel to human reasoning in a machine to machine interaction is found in literature as *grounding* in OWL-S, or business process workflows in BPEL [[67];[76]]. In these areas, the interaction between machines or the components of the services running on the networked nodes operates on the knowledge or concepts through permitted actions and transformations utilising a set of message exchanges. When the situation or environment changes, so must the actions or messages. It is here that problem solving methods and reasoning become important. In

an interesting cycle of conceptual topics, this goes back to the roots of artificial intelligence where much work in formulating, developing and validation modular problem solving methods for machine execution has previously been done and published [77]. The extent to which these can be applied in real-time systems is still questioned although with the help of service ontologies and the semantic web, problem solving through orchestrated services or patterns of actions is a probable next step. Chapters 4 and 5 will describe in detail the use of a set of orchestrated web services operating with a ontology-based knowledge structure for context adaptation.

Reconfiguration in Service Engineering

Context adaptation as an orchestrated service is set within the broader technical environment which involves the user, any relevant device, network, and of course, the detailed nature of the application features itself. This broader setting also includes possible future technology disruptions which may occur in any of these technical areas. One such disruption which has already started to have impact in the computing industry is dynamic reconfiguration; the ability to have fundamental changes in the configuration of the devices below the application layer and into the traditional hardware layer while the device continues to operate. The motivating factors for reconfiguration are many and several of them are quite important to industry. First, the number of unique computing devices that a consumer will purchase, interact with and maintain is continuing to grow. Although this is a state which many industrial entities may find appealing, the sheer complexity of many connected devices all with different and very specific functions is not a welcome condition to many users. The literature expresses this as the multitude of unique platforms that software and systems must be adapted to and support [78]. The benefit of low level reconfiguration is that one generic device which supports fundamental reconfiguration can take the place of several specialist devices reducing the number of devices that require maintenance, reducing the number of nodes in one's personal network that require management and security and reducing the waste

associated with discarding generations. As an example there is an estimated 130 million mobile phones that will be discarded in the US in 2005 alone, clearly a problem of environmental waste is inherent in the continued paradigm of specialist devices. Generic, programmable devices are often called 'do-it-all-devices" and are in fact only possible through low level reconfigurable architectures [79]. A second commonly agreed motivation for the reconfigurable device proliferation is research outcomes pointing to improved energy efficiency when hardware reconfigurability is used to match the demand of the application [80;81]. Dynamically adapting the device configuration to address the needs of the application means that only the necessary hardware and software resources required to execute a specific application are powered and enabled at the time of execution [81]. Theoretically, this will reduce the overall energy consumption; removing a certain percentage of processing which is expended to maintain an operating environment which is not needed. This dynamically configured system requires a new approach to hardware and software design at the processor level and hybrid systems supporting differing levels of reconfigurability are available today [82]. As many opponents to the concept of reconfigurability can be found as those that support it, perhaps because it is requires significant investment in new foundation technology in the hardware manufacturing sectors as well as new software mechanisms and methods all of which come at a significant cost. There are also significant low level issues that are outlined in [83]. However, the basic assumption and general conclusion for future computing trend is that context adaptation services will encounter greater and greater reconfigurability in service endpoint devices so that an understanding of this emerging technology is important and relevant to the future usability of any outcomes of this programme of research.

If the future of computing involves reconfigurable, mobile devices then clearly context adaptation services and their architectures will need to accommodate them. To do so properly will require an understanding of their basic operation. Reconfigurable devices today are generally hybrids with both application specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) on board with specific tuning to allocate different computational tasks between the general purpose computer and more

application specific processing units. It is generally agreed that the manner is which one allocates processing cycles and the nature of the processing cycles is critical to the success of the device. An example of hybrid configuration is the Gecko prototype from [84]. It is a clever mix of both programmable circuits and standard ASICs where the ability to reconfigure the device in the field allows the modification of multimedia codecs to execute video streams, for example in one part of the processor while in another a different configuration may be executing a downloaded set of encryption routines for use within embedded communication protocols. This view of a partitioned system is shared amongst several of the key researchers in the area including [79;81;85;86]. Of particular note, is the large investment and interest in this style of dynamic reconfigurability from such governmental research entities as NASA who have spent considerable amounts of research funding on investigating mission critical reconfigurable architectures [87]. Another concept which is agreed in the literature is that of the distinction in reconfigurable systems that support runtime configuration changes versus those that at design time are pre-defined to have a limited set of configuration options. The implication of interest for the work in this engineering programme is runtime reconfiguration as that promises to fundamentally change the context for adaptation services, perhaps within the timescale of the adaptation service session itself. In runtime reconfiguration, changes to the configuration are devised, deployed and re-linked in field after device manufacture. Key aspects to this concept as it regards context adaptation are the processes by which the change is initiated, how long it takes to complete and the effect of the change on the operation of the application service itself [85;88]. In [85], the process of knowing what to deploy and run and where it is best to run it, is tightly coupled with the power state of the device evidence of one of the primary motivating factors for reconfiguration in mobile devices. These decisions will have significant impact on any higher level service executing or planned to execute.

It is instructive to discuss the convergence of the trends in reconfigurability and service engineering. Reconfigurability is a technology usually associated with lower level hardware design. Dynamic service composition is associated with an architectural approach at the higher application level. However, both academic and commercial

research in these two areas are moving along a technologically parallel line, with each topic area investigating the need for flexibility and adaptability in the composition of the execution environment while recognising the constraints of concurrency and performance. In the service engineering topic area, many of the same concepts are expressed but at the higher level of abstraction. Concepts like service discovery, or the process of finding through semantic query the needed service to execute to provide the results desired is itself a process of knowing what to run similar in concept to in the reconfigurable computing topic area [89;90]. The notion of deciding what is best to run in given the resources available occurs at the higher level. This conceptually similar to the notion of web services workflow and orchestration research where the pre-defined workflow moves towards a dynamically composed workflow that reflects changes in the situational context of the service execution [6;68;91;92]. At a higher abstraction level, certain aspects of the service computing environment are mandated by the provider during the service composition. It is possible though that the two levels of service architecture will come into conflict as the higher level application seeks to adapt to an environment which the lower level hardware reconfiguration continues to modify. The current message passing paradigm found in the XML capability classes approach to create optimised adaptation architectures are not feasible in this instance as the continued reconfiguration could result in a high level "spin" lock resulting from the continuous cycle of reconfiguration and re-specification of capability, most likely to end in the user abandoning the request altogether [93]. In [94], the authors allude to this condition and to the need for the system layers to communicate to allow for the adjustment of overall behaviour in line with the overriding execution environment objectives [94]. It is here again that the parallel research paths are apparent between service engineering and reconfiguration as this information exchange between system layers needed for coherent reconfiguration is roughly equivalent at the higher level to the cross layer network design topic mentioned in the first part of this chapter. Attention must paid during any system design to the potential for conflict between the application or service and the reconfiguration, this is specifically as it pertains to composite lio where terminal devices can support different access methods depending on the most efficient technology available and their configuration in the field

is adjusted in real-time to the appropriate access capability for the service area [95]. The work by [86] investigates reconfigurability for radio access through the software defined radio initiatives common in 2001. In this research the authors report on the design, not the practical implementation of, an adaptive baseband processing system within the mobile terminal which uses runtime reconfiguration capabilities for multi-access radio functionality [86]. While the design and architecture reported appear as a feasible design, closer inspection on the subsystem show an extensive amount of processing capability required within the terminal to manage the reconfiguration process including a primary and secondary configuration simultaneously resident inside the device.

It is clear from the progress of the research outlined in the literature that reconfigurable devices, architectures and prototype implementations are continuing to grow in number. One aspect that is also quite evident is that a unified approach to the design of such systems which incorporates all the layers of the system architecture and extends out into the domain of distributed computing will be important for the viability of reconfigurable devices. New approaches to the design and to the simulation of how these complex device and system configuration will be initiated, managed and terminated during the execution of a higher level application service will be critically important to the future success of reconfiguration. There is little literature on this topic available today and what is available refers primarily to the specifics of code partitioning onto a hybrid hardware component or methods for compilation of reconfigurable algorithms [96;97]. Code partitioning is a central research theme in this area although very little is mentioned about the issue of moving from one configuration to the other in short time spans and how the efficient spatial reconfiguration will also be a factor in deciding the most optimal partitioning algorithms between hardware and software. There are also important issues of timing of reconfiguration and in fact, the concept of reconfiguration induces a new spatial and temporal design problem in hardware as pointed out in [98] with their work on co-design of embedded systems. They describe yet another dimension in this area that includes the timeframe of reconfiguration during the execution of a specific application and how that may affect the overall efficiency and consistency experienced by the user. While architecture description languages (ADLs)

are the traditional method for designing and verifying such systems, the addition of time and dynamic transformations of architectural components onto a spatially segmented processing platform will tax most ADLs and may require completely new approaches. These approaches will need to be suited to three dimensional mapping of control and datapath components onto platforms where part of the mapping process involves a least cost optimisation function involving the needs of the runtime execution coupled with the time and energy needed to get it to that state. To properly design for such future environments, engineers will need formal tools and techniques taken from distributed systems concurrency theory as well as visualisation methods to ensure that distributed deadlocks, interlayer conflict or inconsistencies do not occur.

Evolutionary Computing in Service-Oriented Architectures

A final area of interest in this programme is in automated service requirements to device capability matching process. The extensive parameterisation of context coupled with the dynamic reconfiguration of the mobile device in an environment where services can be composed in real-time provides for a large solution space for the service provider interested in delivering optimised services. Few problem solving methods in the literature can deal with this type of high dimensionality problem space without initiating a search for an optimal solution within a fixed set of potential solutions [99]. Three elements of the solution are of keen interest to commercial engineering in the next generation network environment; how best to configure those configurable entities in the transaction for an optimal experience, how best to match the configuration objective with the request target, and how to achieve these tasks within a specific timeframe. One technique that is heavily researched and published is that of multiobjective evolutionary algorithms and strategies which are able to use relatively simplistic mathematical operations to find an optimal set of potential solutions such that no other solution can be found that will not degrade one objective. This set of solutions, also called the Pareto optimal set, provides a set of states given the service requirements and the end point

device to achieve together, in order to successfully complete a service request [100]. The use of multiobjective exploration of a possible set of solutions is used extensively in many technical areas from circuit design, cellular communications configurations, local area network engineering and even digital art and product design [99;101-104]. In this current work, the problem is posed in terms of the extensive number of variables, each impacting more than one objective in the problem space and all the objectives are equally important. Such problems are difficult to solve with mathematical techniques, e.g. linear programming, as the solution is both non-linear and often non-deterministic. The question remains why use these evolutionary techniques at all? First, the multiobjective nature of the context adaptation service engineering task makes it a good fit for such problem solving techniques and there is excellent support for using it in this way throughout the EC literature [105]. Second, given the proper representation is found and the algorithm is tuned appropriately, good performance can be obtained for problem spaces with many variables. Lastly, there are several excellent tools and experimental platforms available to the engineer based on solid and well-published research [106-109]. This is an important aspect in an engineering study which is not a pre-requisite in more fundamental research programmes whose objective is to develop the necessary tools to solve the problem at hand. Engineering work is traditionally constrained to using existing tools and enhancing or extending them to fit the specifics of the problem to be solved rather than inventing again new tools. Evolutionary computation methods are not without their issues and opponents. They can suffer from problems of convergence in some areas where the algorithm fails to converge on any feasible solution over time or it converges to a value which is suboptimal. Thorough analysis of genetic algorithms and evolutionary computation within the context of search and problem solving methods can be found in [110;111]. Both publications address the technique by analysing its effectiveness in working with large population-based problems set in complex adaptive environments. A discussion of the problems with applying EA's to complex problems can also be found in [103] where the issues of premature convergence, optimisation of noisy functions and prioritised multi-objective problems are outlined. In [112], a deeper discussion is found on the issues encountered in the multi-objective function optimisation domain.

Understanding the nature of the populations involved in a problem space is a critical element in deciding on the application of evolutionary computation methods. These populations can be biological or in the case of context adaptation, they can be non-biological as long as they behave in a distinctly evolutionary manner. In context adaptation, these populations are devices, users, networks and even the software itself. This remains a consistent trend in the use of evolutionary computation; to delve into the nature of the non-living populations involved in new problem domains. In almost all cases with the notable exception of the first unicellular creatures, evolution is not a single species evolution but rather a co-evolution of multiple populations competing for resource to varying degrees [113]. Co-evolution provides for the evaluation of the fitness of one species by the other rather than through manually defined evaluation function and in some studies this has resulted in significantly better problem solving performance [114]. Although the service level agreement will provide guidance on how to achieve the optimal service, it does not provide a means to match configuration with requirements. Specifically, the service level will dictate the performance characteristics of the overall service, not how that performance is actually achieved. Without a clear understanding of what makes one solution better than another, the notion of co-evolution where one species or population decides the fitness of others, enables an interesting adaptive problem solving method. The techniques in this area are new and there is still much work needed to understand the complex dynamics between populations which can sometimes be cooperative as in the symbiotic biological relationships found in nature [115-118]. Alternatively, co-evolution between the populations can be an antagonistic or competitive situation where the various populations are competing for a scarce resource. The competitive co-evolution is useful in evolving solution to strategies in economic markets and in competitive game playing [117]. The principle difference in co-evolution techniques is the use of one population as the evaluation function for the other, independent of any manually defined evaluation function. This type of evaluation is usually in the form of positive and negative interactions between the two populations [119]. The application of a co-evolutionary approach will be useful when analysing the change of the service requirements as they

might change relative to the device population, or perhaps in representing how the device populations evolve over time relative to the market demand or user requirements.

Context adaptation services must resolve many parameters and take optimal actions in a very short and constrained time frame. This set of actions can be based on a large number, e.g. population, of prior decisions when appropriate problem solving methods are used. A key to using population-based evolutionary theory in problem solving is fitting the practicalities of the problem domain with the appropriate set of technologies that occur with the broader evolutionary computation. By analogy, the evolutionary computation field offers a wide array of mathematical and statistical optimisation tools to the engineer and the design work is to then compose the solution space in the most appropriate manner so that when the algorithm is run the optimal solution subspace is presented. There is little disagreement in the literature about the importance of representation in the formulation of an EC problem [120-122]. Throughout the literature, how to encode the intrinsic nature of the problem into the gene is the central issue for getting good performance from the algorithm itself. As pointed out in [120], the representation is so important that it may be appropriate and sensible to use evolutionary computation the problem's own variables to evolve the appropriate representation rather than leave it to the interpretation of the engineer. Other aspects of the representation are particularly interesting and appear in the literature where practical applications of the method are outlined. Variable length genomes which can evolve over time, as their biological analogues do in nature, are researched heavily with varying degrees of success in their implementations [123;124]. Some lack of success is noted in variable genomes when dealing with mathematically based evaluation functions that have positional dependence built into the genome itself. This is a difficult problem as variable length and mixed variable representation must have gene position independence. The use of start and stop sequences is a feasible approach also, in order to allow the genome to change over time and yet to still maintain an accurate representational mapping back to the solution space. As will be discussed, the concept formulated from the problem statement and the literature, is of a context adaptation

service environment using the knowledge-base as a possible repository of the phenotype itself describing both the device and services in the problem domain. By using this information, and developing a genotypic representation as a reverse mapping from the phenotype, a first attempt of co-evolving the two populations can be attempted. It is possible to see in the literature the use of such test-based co-evolution as means to validate a learning population against a pre-defined or evolved testing population. In some cases, this is referred to as partnering strategies where two sub-populations are evolving independently according to a set of fitness functions which creates an optimal set of solutions in a subset of the overall space. These sets of solutions are then co-evolved against each other to decide the most optimal set of solutions within the overall space and across the sub-populations [125]. This is consistent with the concept in other work of evolving sub-components to a more complex problem with the objective of using cooperative co-evolution to then evolve the population "leaders" to form a higher level of overall solution fitness [116;118].

Through the review of relevant work, it is clear that there is significant evidence to prove that multiobjective evolutionary computation is a viable technique for the context adaptation multi-dimensional problem solving. The choice of the technique for this study will be discussed in Chapters 1 and 7 as it is guided both by the industrial context of the entire programme and the direct analogy of the mobile device and adaptation service as two dynamic, evolving populations each seeking a scarce resource, that of market share. Co-evolution provides a means to evolve solutions without the design of a evaluation function, and that this style of optimisation is extensively researched and used in practice when attempting complex configuration-based design problems. One approach taken from the literature involves three steps; decomposition of the problem, creation of models for each decomposition space and integration of the models [126]. Published results from experiments in the runtime use of EA results for real-time adaptation in complex environments are sparse and the concern with the use of them in this type environment is the condition where there is a failure to converge in a production execution timeframe. Current research in this area involves the use of distributed evolutionary algorithms which are deployed through a network

environment and can be executed in parallel. There is published work as of this date in the area of performance analysis and these point to favourable results in execution when algorithms are properly designed and tuned, and specifically that the convergence is repeatable and robust [118]. However, even in light of these realistic concerns, the benefits of using a properly tuned evolutionary algorithm are summarised as follows:

- Provides an optimal of device to service configuration matching function with a simple co-evolutionary evaluation technique – a constrained but continuous variable optimisation problem [110]
- Handles sparse or missing configuration information through partial matching functionality – allows new device and service components to appear in the environment without manual specification and publication

It is clear from the literature survey and review that context adaptation research spans many areas of systems engineering. From issues of hardware, software and networking to the business model necessary to support the robust deployment of quality adaptation services, context adaptation as specified by this engineering research programme is more about creation of a services framework and architecture than about delivering one individual service set. The published literature available sets this engineering programme in "context" itself. It provides positioning and guidance about opportunities in enhancing the current paradigm of service delivery to include adaptive computational intelligence into a commercial architecture. It also provides the motivation to pursue a design such that the performance of the overall system can be automatically managed by a complementary set of web services.

Chapter 3 Contextual Systems Engineering in a QoS-regulated Environment

The original motivation for this programme of research was the desire to understand and potentially exploit the available contextual information enabled by more computing power within the network components themselves. Networking has moved well beyond the simple bridge and switch architectures now control elements within the network itself are assuming a more intelligent role in the way in which services and applications can be run. The intelligence in the underlying transport mechanism of the network can act to enable more complex and potentially higher valued services. The great difficulty is to decide the amount and nature of the partitioning of intelligence as there are several disadvantages with the addition of computational ability throughout the interconnected nodes of the network. Several researchers have attempted to investigate the generic attributes of this trade-off of machine intelligence for performance or distributed network computation versus security. However, there has been little work around the actual commercialisation of such solutions that would provide evidence of conversion from academic study and findings into a usable body of engineering knowledge. The question is then, while it is feasible to "activate the network" by distributing some measure of computation or logic into the network nodes themselves, what applications are possible and what are the engineering challenges that will be faced when attempting to commercially implement these architectures into consumer products?

This chapter will review the underlying network engineering that supports the delivery of context-aware adaptive systems at a more fundamental level. The topic of active networks will be outlined, as well as a discussion of the value of cross layering between the network layer and the application layer which is an essential part of the adaptation architecture. The details of the Middleware for Active Networks in Image Adaptation to

Context or MANIAC, a collaborative project between Kodak Limited Research and Development and University College London, will be discussed at length and the results presented. In the final sections of the chapter, the details of the context 4-tuple and the separation of the static from the dynamic context will be introduced. The resulting challenges will be then outlined as a precursor to the further investigation of a new architecture for context adaptation.

3.1 Adaptation in Active Networks

Context adaptation is a higher level systems concept truly dependent on the support for adaptive processes found in the deeper layers of the traditional system architecture stack. In this section, the field of active networks will be introduced and a basic discussion presented on the need for its inclusion as an enabling technology of adaptive services. In 1995, Scott Shenker who was at Xerox PARC at the time published his paper on *"Fundamental Design Issues for the Future Internet"*. In this work, he summarizes and presents some of the more fundamental issues with the traditional architecture of the Internet itself. Prominent among these issues are the concern with the best effort nature of the internetworking topology and the lack of management control and capability around resources that are inherent in its design [10]. It was recognized then for the Internet, as it is now for private and public networks that while best effort is simple and straightforward it lacks the ability to provide differentiated services which would serve as a network infrastructure version of a free market force for the network resources control. Differentiating services also provides the ability for new business models as services such as imaging services which can provide variable levels of resource -- computational, storage, or other -- based on the subscribers levels or even based on a point of use or time of use payment schemes. Commerce and transaction potential are not the only impetus for investigating increasing the level of distributed computation embedded within the network nodes. Flexibility provides a second substantial motivation for pushing the execution environment out into the nodes of the network. It is generally true that computing paradigms have a recurring pattern. One such pattern is that of the movement from monolithic, tightly controlled resources maintained by specialists in the early stages of a computing domain to a model of distributed,

encapsulated components accessible to all in later, more mature stages of technology evolution [46]. This is observed and documented readily throughout computing trends, from mainframes to personal computers; in telephony from monolithic hardware switching with tightly controlled features to the software components based multi-tier intelligent switching networks of today. Systems software itself has moved from the world of tightly controlled specialized languages and operating systems for controlling the low level nature of the computing hardware, to graphical development environments accessible to all with power and features that far surpass that of the mainframe systems programmer of just fifteen years ago.

It is quite natural then to combine these views and see that pushing intelligent components into the network in order to make informed choices during transmission is another example of this generally accepted evolution of computing. The flexibility and differentiation come with a cost, however, much as the shift to personal computing and its increased personal control over computing resource was a huge benefit, it also increased each individual users own technology management responsibility. This same increase flexibility as the intelligence moves to the "leaves" means an increased computational workload in the network components which can decrease performance and reduce the overall security and integrity of the network as the programs may now be executing on nodes and accessing packet level information on behalf of the users[127]. This type of network where the execution environment or computational elements are inside the network is commonly referred to as an active network. Active networks are an area of extensive research within network engineering, particularly around concerns of security and performance. Of particular note is the concern that intentional damage through virus and worms can be far more devastating to the operation of a network using the active network architectural style. In these cases, it is no longer that a single individual or group is affected by this, but due to the interconnected nature of the Internet, any traffic from any where on the globe routing through a compromised router will be affected. Much work has gone on to understand these two areas of AN vulnerability, in addition to other operational issues, and there is still much to be done.

Application service providers are dependent upon the underlying transport layers of the network in order to deliver their services and collect their revenue. The opportunities and challenges of developing systems and services that are not only compatible with the new active network infrastructures, but that take advantage of the benefits of such enhanced technology can be significant online business enablers. The issues for many commercial service developers and providers are that their domain of expertise resides only in the features of their services, not in the overall technical details of the active network architecture. Additionally, the ability to affect any changes in the operation of the underlying network infrastructure is extremely limited unless there exists an open application interface to the network provider [128]. To understand the technological possibilities of these networks, any commercial investigation must look at all layers of the network, the elements of the infrastructure that supports the creation and management of "activated services" as well as the core service application components themselves. This was the mission of the industrial study done as a part of this programme of research to investigate this as a first step towards understanding the commercial implications and benefits of the active network.

3.2 The MANIAC Project

Before presenting the details of the Middleware for Active Networking in Image Adaptive Computing (MANIAC), a collaborative study, it is important to review certain foundational concepts concerning the systems engineering in this area. Middleware and middleware-based architectures are probably the most important systems engineering concept when dealing with heterogeneous environments and dynamic services. Middleware as technical concept is a dominant design metaphor in current software engineering both in academic and commercial projects. The value of middleware is many-fold; it creates an abstraction layer which permits the interconnection and translation of services between different layers of systems and between different system components. Layering middleware systems over a differentiated services network has been done in several different research projects for various reasons from solutions of specific problem to provision of generic architectural support. In this work, we were specifically interested in applying the knowledge learned from the generic use of

middleware architectures in the active network topology to performing content adaptation or "transcoding" for imaging applications. A good, functional description of transcoding is provided by Knutsson, et al in their discussion of server-directed transcoding [14]. Transcoding in the most general terms is the transformation of content from one form into a representation which more suited to the user's needs and device capabilities. Although this may sound like context adaptation, transcoding is used most frequently to describe the specific transformation of object formats rather than the entire content "package" which includes objects, navigation, or GUI components. Conceptually transcoding is a component of the overall context adaptation, providing the raw translation of the rich content based on specific parameters. It is important to separate the service feature such as transcoding from the middleware. The middleware is the vehicle across which the feature is delivered. This will be seen clearly in the description of the MANIAC prototype. While transcoding represents the marketable feature in the online service, the system must do more than deliver one feature and that is often the domain of the application which is responsible for putting together all the features and functions in order to deliver a higher valued service. The MANIAC investigation used transcoding of images as one of its features and combined this notion of transcoding with the increased computational capability found within the nodes of an active network. The specific instance of the active network was designed to provide differentiated services as mentioned above. The programmable middleware utilised as the framework for MANIAC is the PROMILE system is detailed in [42]. This middleware solution provides the lower level network integration points to the custom designed and constructed application service.

The use case scenario for this study starts with the concept provided by a structured service agreement between the three traditional tiers of telecommunications; telecommunications provider – service provider – consumer. Using the bidirectional relationship at the application level to drive the QoS parameters at the network level. These QoS parameters are then fed into the active network in order to create a logically consistent model for adaptive service deployment.

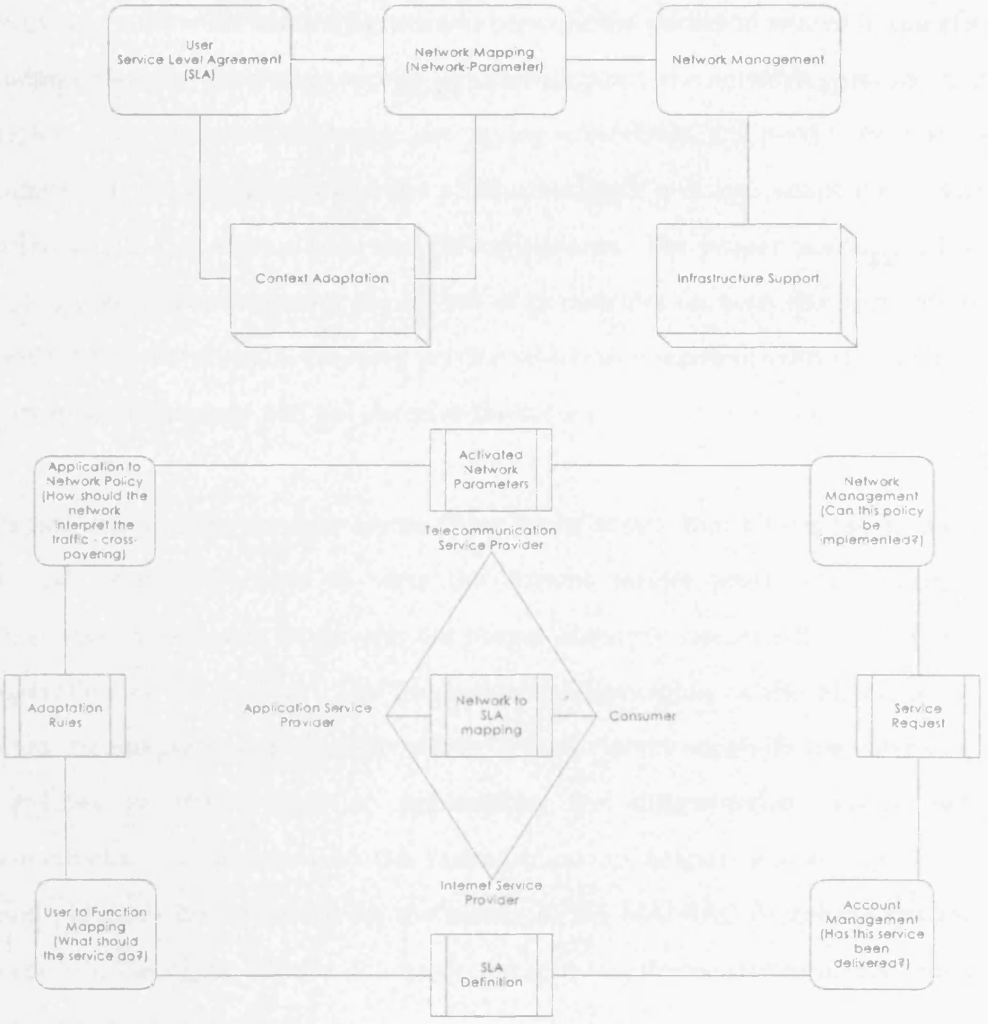

Figure 3-1: MANIAC System Model

The novelty of this system is that it integrates the cornerstone components of the adaptive service delivery model into one architectural framework and includes the ability to administer the network services in conjunction with the application rules which are derived from the overall service delivery business models. This is important because administration of future content delivery networks promises to be extremely challenging given the interconnected nature of the content services with the differentiated service networking engineering. For example, the future online services company may need to design and deploy whole new services very quickly to stay competitive, and part of this service deployment may include network adaptation at the layer 2 or layer 3. In order to do this, a new service level agreement would need to be

developed to outline the mutual agreement between the parties in regard to the offering of this new revenue-generating service. From that point, the network parameterisation, perhaps the delay time inflicted on lower paying subscribers, will need to be matched to the offering of the application and the application itself will also adapt the services in accordance with this service level and its components. The proper operation of such a service will be dependent on a logical set of parameters on both the application and network layers that forms a cohesive service which is consistent with the needs of the consumer otherwise they will not perceive the value.

To deploy and modify services across these levels means that all entities in the value chain will need to be able to view the current service attributes, routing, and parameterisation but only those with the proper authority can actually modify or affect the operation of the service. The fundamental construction of the MANIAC system involves the integration of a quality of service subsystem which defines and monitors the policies of the service or application, the differentiated service network implementation architecture and the management application which implements the defined policies. Before reviewing the nature of the MANIAC prototype system, it is important to detail the nature of one of the four key cornerstones of the study, the quality of service agreements.

3.2.1 Quality of Service Agreements

It is now relevant to review the details of some of the novel components of MANIAC. Central to the work in this programme is the notion of implications of engineering in a regulated and monitored environment. For the purpose of this programme, this regulation takes the form of the service level agreement between the Internet service provider representing the application service provider and the customer. The quality of service (QoS) structure is essential to the operation of next generation service-oriented architectures. The primary means of this operation is through the use of a service level agreement which provides the contractual bounds on the system-user interaction. The difficulties in implementing service agreements in an automated service provision environment are threefold; the ability to define them at a manageable level, the ability to

force compliance in an automated way and the ability to monitor and report on the activity which occurs under the jurisdiction of the service agreement [129]. Today, service level agreements with the consumer of the services tend to be an all-inclusive arrangement where there is an implicit quality of service measure which is predominately predicated on the network availability. In the small to medium business environment, service agreements tend to be more service specific but again, are primarily around the provision and guarantee of a specific rate, availability and price of transmission capacity. Many market analysts believe that consumer service provision will also continue in this way, but that value-added services which have additional QoS parameters will become increasingly more important to end consumers. Concepts like buying improved more secure VPN services, automated archival and other such services will become those services that the average connected consumer will purchase much like they purchase the enhanced telephony services today. With the emergence of these value-added services, there will need to be an accounting framework in place to ensure that the value provision is as it promises and that the consumer can have recourse in the event it is not. Such a framework will inevitably resemble the service agreement and as such will provide the quality of service environment for online services themselves. A formal structured approach to service level agreements then provides the ability to automate the control and management of these activated services, so that like your telephone bill, your services invoice can be generated. The service level agreement notation generator, or SLAng from [49] provides such a structured approach using XML. A thorough understanding of the changes to the engineered solution when introduced within a SLA constrained environment is a key result from the MANIAC project itself. The success factors of the project itself now had to incorporate adherence to the service level rather than purely the delivering of the service itself. This addition significantly influenced the design and engineering as will be detailed in the following sections within this chapter. It is important to note though that a key result was the fact that for commercial systems to operate and deliver on the specification, the engineer will need to have either a clear understanding of the structure and potential constraints of the SLA for a general class of the intended market or there will be an additional step to properly specify how the service level will impact the requirements of the system. This step in

service engineering is not currently a standard task in practice as of today. Given knowledge of the details of the agreement though, it does help to provide a suitably robust and definitive framework in which to investigate the operation and performance of non-contextual and contextual services.

3.2.2 SLAs and Differentiated Services Networking

The relevance of the programmable network within the MANIAC system architecture is to provide the means for cross layer design and to implement the transmission adaptation. While the SLA structure forms the basis for the transactional control layer, the differentiated services network forms the underlying infrastructure to support and implement the control provided by the SLA. An active network and its unique functionality and capabilities form the second key cornerstone of the MANIAC research project. The implementation of the service agreement between two parties takes place on two different levels. The differentiated service model that is referred to in this work is an example of a programmable network which operates in a standardized way of providing services, algorithms and management capabilities within the network in order to offer new complex services and capabilities to the user [39]. Examples of such services are the imaging services which were designed to exercise the programmability in two ways. First, in the ability to flexibly incorporate the changing service agreement parameters via a management interface. Today the service agreements are necessarily broad with a generalized subscription bundle covering the various services the user is interested in from basic connectivity and bandwidth through to email, websites and other such service offerings. The pricing and competitive positioning of such bundles is a critical success factor for Internet service providers. In the current environment, changes to the services within the bundle on a person by person level is possible but prohibitively difficult to manage and impossible to properly account for. By abstracting the control and management of the service level with its associated parameterisation in the software within the programmable network, then real-time modifications, transactions and most importantly personalized service provision is possible.

Media and content service providers are equally interested in having the opportunity to compose customized service bundles to run across the future "open" network. From this research study, it is clear that the selection of differentiation implies the need for a specific network model. The Diffserv network model promotes the integration of the service agreement and creates a means to implement the service agreement. This network model itself is the result of many years of standards work within the IETF. It essentially specifies a model which has a set of core functionality and a set of edge functionality. The edge functions to classify and "condition" the traffic, while the core functionality insures that the transmitted packets are forwarded appropriately [130]. This is the fundamental starting point of the control framework needed to implement the service agreements structure, but it is not sufficient to implement the entire service agreement which regulates other behaviours outside of those concerned with purely the transmission of traffic. The middleware and application code is also required to control the higher level system behaviours such as specific service features, for example in the imaging services the network service parameters which control transport are managed within the Diffserv model, while the appropriate image resolution to be delivered are managed at the application level.

In the MANIAC system, the QoS levels are managed within the DiffServ network and are coordinated with the web services architecture of the imaging application itself. These two levels work cooperatively to implement the agreement and provide the services as specified. As previously mentioned, the MANIAC system utilizes a programmable network environment from University College London called XMILE which provides an XML programmable router architecture [3]. The significance of this choice in the overall context adaptation system is that it implements a differentiated services network model and provides the infrastructure support to allow XML specified rules about the service differentiation. The XMILE architecture is expanded further by the same group of researchers to include a management function in another system called PROMILE. The PROMILE system provides an application overlay which permits distributed administration of the active network from an operator console with overall

goal to achieve differentiation with the XMILE architecture integrated with the programmable management function to achieve maintainability. While XMILE allows the use of standard XML for the specification of the network parameters, PROMILE extends this to include functions for real-time management of the policies for the network operation [51]. This foundational networking infrastructure provides a controlled, yet adjustable connectivity layer over which application adaptation can be implemented. By connecting or "cross layering" the traditional network layers the system designer can enhance the adaptation capability and hence the perceived value of the services. This is not without risk and cost which is a subject covered well in several research reports. The importance of this middleware for imaging adaptive network investigation was to provide solid engineering and operational evidence of the feasibility of such a solution in a web services model.

3.3 MANIAC Architecture

The MANIAC prototype was the combination of three separate, but integrated activities. The work of the author was to design the architecture for the overall system, the interaction, the feature itself. Researchers at Kodak Limited R&D later took the design specification and developed it into a mobile web services demonstrator and a team from UCL provided the ability to use an active network. The web services application itself was composed of a JPEG 2000 server and transcoding engine called Kakadu from Professor David Taubman of the University of New South Wales, Australia [131]. The selection of the JPEG2000 was based on the new formats ability to dynamically render through the use of wavelet transforms, variable resolutions of the same image without creating a new file. The MANIAC system was designed to demonstrate the following:

- Implement an "indirect" cross layered application using a web service which adapts the delivery of content in two ways – providing functionality to adapt at the network layer through expedited packets in a differentiated services network and at the application layer through the change of resolution of content

- Demonstrate the ability to have independent yet consistent control of the two layer adaptation by using the service level repository as the mediator
- Demonstrate the use of a single SLA structure and a network management platform to control adaptation definition and deployment

The architecture designed to achieve the demonstration of these attributes appears as shown in Figure 3-2.

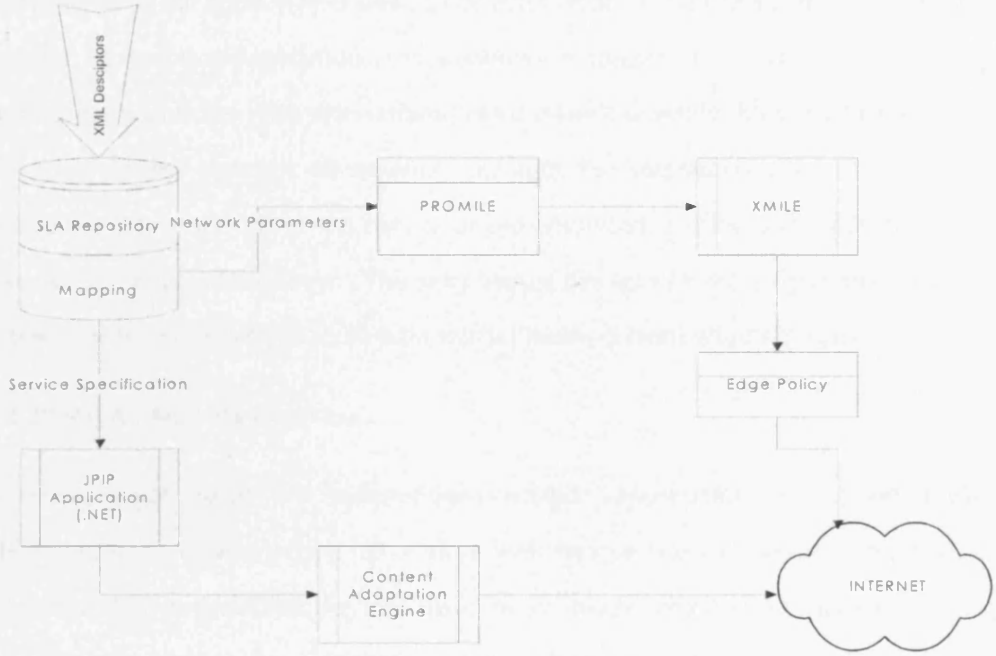

Figure 3-2: High Level System Diagram – MANIAC

The architecture uses the SLA as the governor of the service deployment which effectively links the active network to the adaptation service creating an "activated service". The service level itself uses service categorization and mapping to provide the layer by layer instruction. Under this process, services can be bundled into a single shared identifier or it can be specified for each individual service that a user might request. This allows for future personalisation of discrete service components. The critical requirement is that the service specification components must be logically consistent with each of its constituents for the network and application capabilities. This means that the SLA must not specify parameter values that direct the network to attempt

some adjustment that does not support the goals of the service category itself or indeed vice versa. Driving the service bundle definition from the agreement itself and then distributing the layer specific rules means that the business service manager and the network system administrator have one specification in common at the highest level and then options for each layer can be derived based on the features of interest. For example, if in our case, the "gold" service level was specified as the most valued level, then a hierarchical decomposition of the service can be specified through the system layers in a manner forcing the service operation to be consistent. With the agreement as the central governor of service description, the business manager is not free to design service offerings for which the network administrator cannot possibly deliver adaptation. While this is not a new concept necessarily, through the implementation of the MANIAC system, the experiment into the nature of the engineering effort demonstrated the need to work from this shared view. The selection of the service level agreement structure as the unification point proved to be a beneficial method from which to start.

3.3.1 JPIP – An Activated Service

To demonstrate how an independent service application might be built and implemented in such an active network, a web service was devised which provided the means to adjust in real-time the resolution of an image which was requested by the user based on the service level that the user had purchased and the allowed network bandwidth associated with that service level. The JPEG2000 reference software development toolkit provides the basic module support for encoding and decoding the file format, has functions used for retrieval, and rendering for client display including the variable resolution functions [131]. The .NET framework from Microsoft was used to build the core MANIAC application level functionality; wrapping the Kakadu executable functions in web services and using a simple thin client interface to both permit the user and the SLA administrator to interact with the system. The application level prototype was designed and built in the research laboratory of Kodak Limited, the service level agreement structure was constructed by researchers at UCL as was the underlying active networking infrastructure [49]. The images were uploaded in traditional JPEG format and encoded automatically on the server into the new JPEG2000

format and stored for later retrieval. Depending on the service level of the subscribing user, the transmission of the image packets was conditioned either expedited or delayed via the PROMILE network administrator console. Once converted to the JPEG2000 format, which is also offered as a web service, the image is available for retrieval in the multi-resolution format. On the receipt of a user request to retrieve the image, the image resolution level appropriate to the service level subscribed to is extracted from the multi-resolution format and the image transmitted with appropriate level of packet conditioning. The result is a system which operates independently preserving indirect cross layering but in the operation of the layers work concert to provide a cohesive adaptation service.

3.3.2 MANIAC Prototype Operation

Once the architecture, tool selection, and interaction design were completed, the prototype was constructed and operationally tested. A simplified SLA interface was developed to mimic the SLA management console. A user interface was also developed in order to operate the MANIAC system for testing, simple request-return timer was added to the interface to confirm the relative expectations the user might have based on the subscribed service level. In Figure 3-3, the .NET IDE is shown containing the assemblies which make up the custom components of the project.

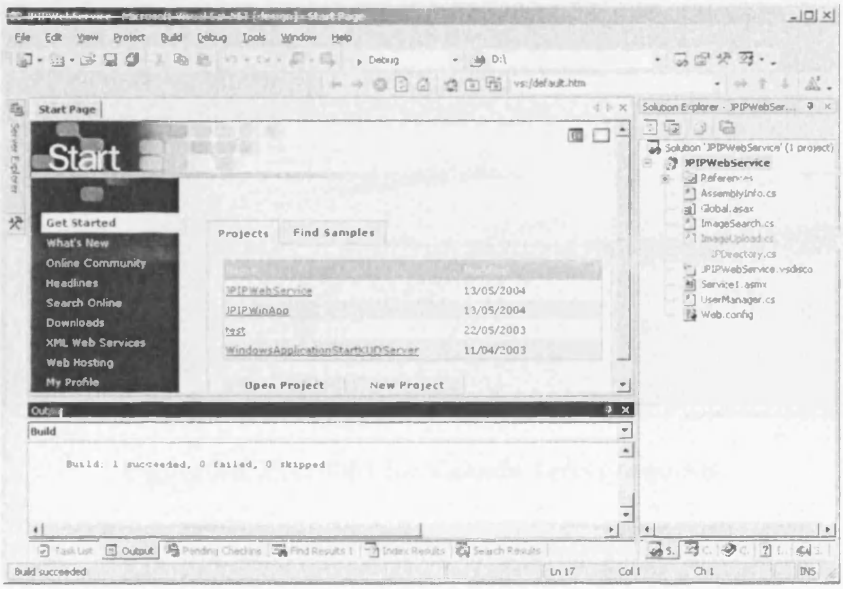

Figure 3-3: JPIP .NET Assembly Web Service

It is instructive to present a walkthrough of the system operation. On an implementation level, the application consists of simple set of web services to support the start of the Kakadu server. This opens two TCP ports, and it is this service that receives the image requests and returns a response. A second set of services is then responsible for the retrieval and encoding services. A very simple interface is used to start the two Kakadu listeners on the server, one for incoming images and one for those which have been adapted to the user's needs.

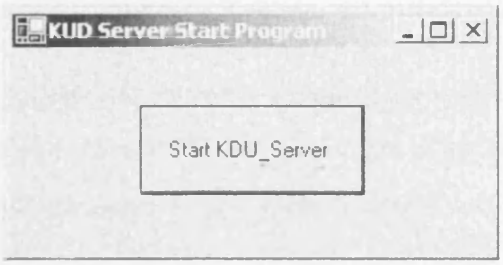

Figure 3-4: Kakadu server startup service interface

The services are each started as depicted in Figure 3-5 and Figure 3-6 below. Each service maintains a running log of the requests and the tasks taken on those requests so that the chain of events of the request can easily be visualised by the engineer.

Figure 3-5: Port 8001 for Kakadu server requests

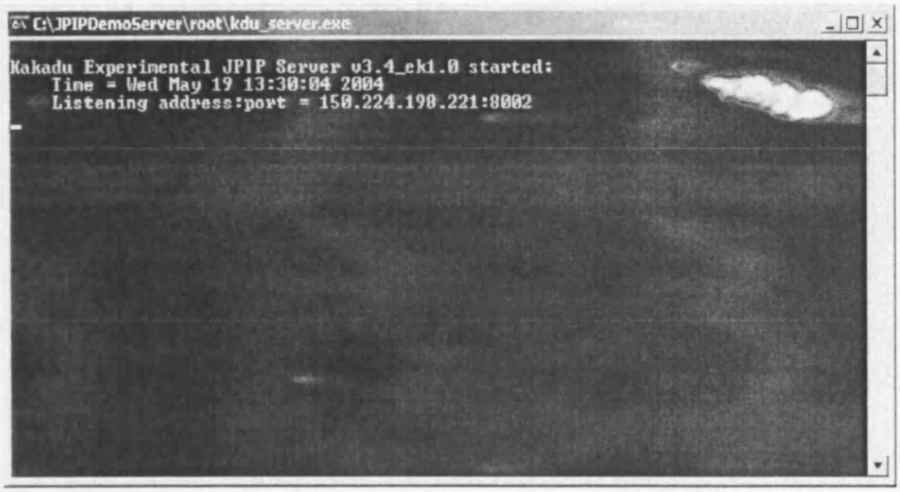

Figure 3-6: Port 8002 Kakadu server for replies

As mentioned previously, response times for a request are visible from the interface and roughly indicate perceived time. These times are not intended to be a true representation of network transport time. Three levels of service were defined; gold, silver and bronze. Each level had both network and image adaptation parameters defined as well in accordance with XML definition as structured within SLAng [49]. These parameters were agreed between the network administrators managing the programmable network and the application developers acting as business managers developing the new service offering. Once complete the network engineering and policies were implemented in an independent step through the PROMILE environment. The complete application was implemented in three phases; one for content adaptation which was the construction of a web service without any programmable network infrastructure. In a second but parallel phase, the PROMILE infrastructure including the XMILE execution engine was implemented within Kodak's research laboratory. The final phase of the experiment is to test the combination of the two with the linkage of the SLA repository and mapping as the only point of interaction between the two projects.

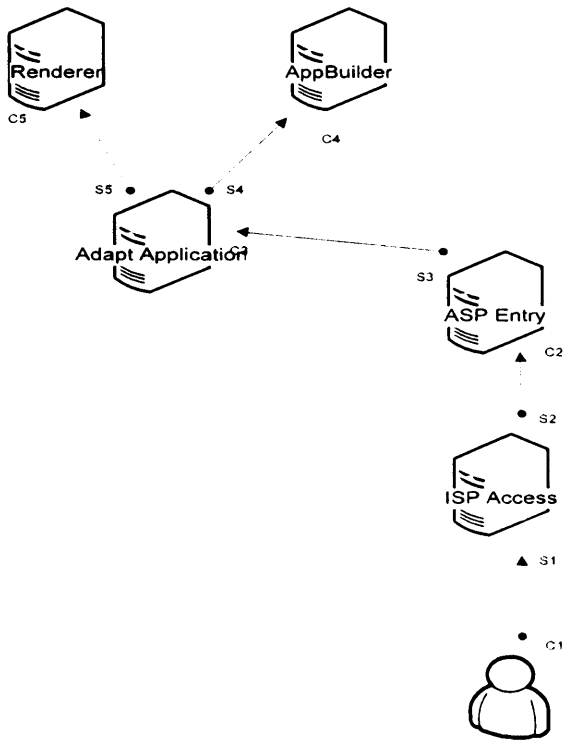

Figure 3-7: Service Level Agreement Overlay on Context Adaptation System

The SLA repository represents the mechanism by which the network engineering and the service context can be joined. The SLA implemented can and would mostly likely be comprised of several sub SLAs, with each representing a different view point in the business transaction. In all, it is clear from Figure 3-7 that there are in this one pathway of application flow, five different service level agreements possible. An additional service level agreement could be the integration of all five to act as the service level for entire service execution; however this is a point of research outside the scope of this programme. Each agreement can be resolved using the method developed by [49]. The result of the resolution would look like the sample depicted in Figure 3-8. The figure below depicts the XML policy as implemented PROMILE and the content rules taken from the .NET JPIP application.

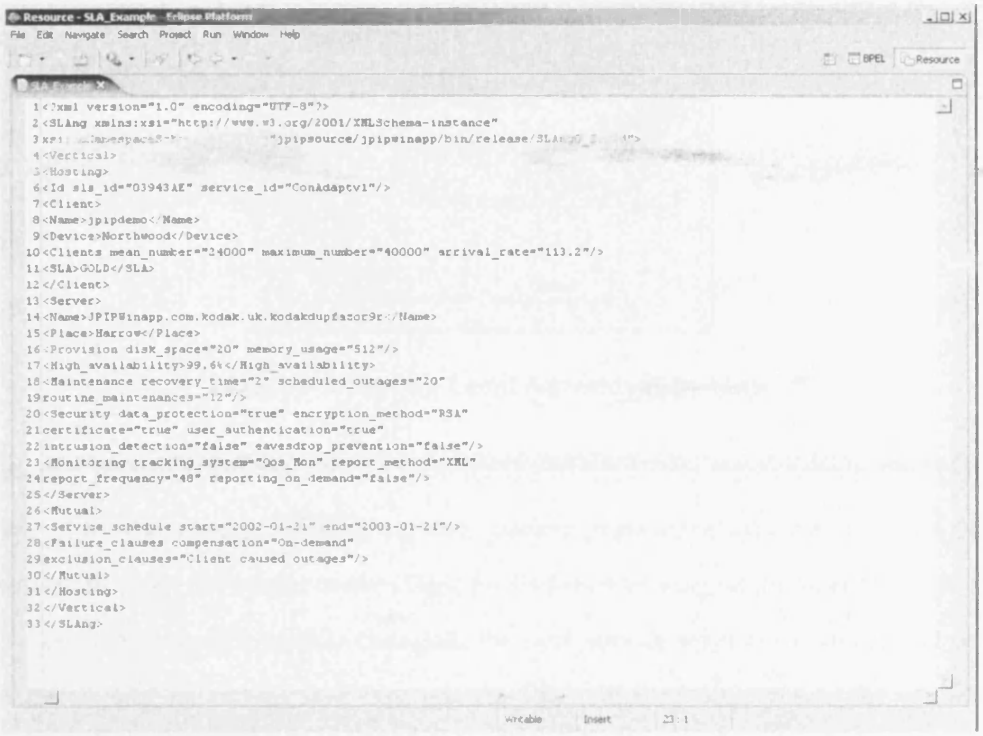

Figure 3-8: Example of one vertical SLA

This reflects the nature of the service delivery environment as envisioned by the designers. For example, there is specified a monitoring service which will act as the accounting entity to ensure that the service level components are tracked and deviations are reported. The key component that will be picked up by the lower layers in the architecture is the specification of the GOLD level designator which resolves to a common term at the two layers that are involved in the system. The other structured tags are those found in the basic language as described in the SLAng specification and are there to indicate how further aspects of the service relationship can be included. This connection between the application service and the programmable network is initiated through the use of an administration function, a simple menu which is provided to allow the service provider to manage users and the subscription level associated with them. This is depicted in Figure 3-9.

Figure 3-9: Service Level Agreement by User

This simple interface sets the service level for the individual services within the JPIP web service. In the case of a gold service, packet expediting will be in force for all transmissions from the server to the client for this user as long as the user remains at the gold level. If the service level is changed, the new service level at its associated service specification will be propagated through the PROMILE system which can update the edge routers in its administrative domain in real-time without loss of service.

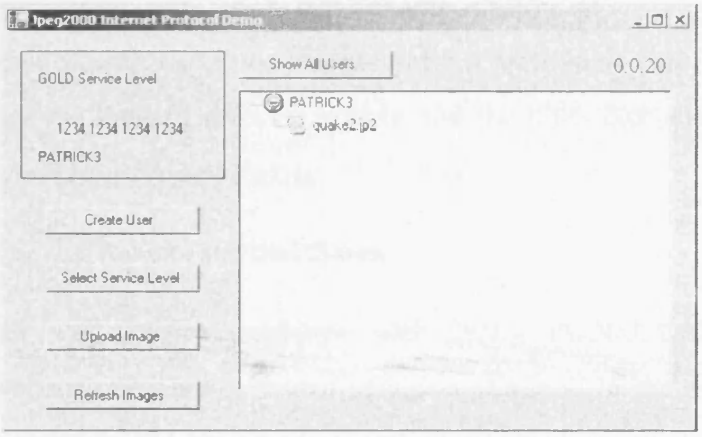

Figure 3-10: MANIAC Application Interface

In Figure 3-10 is shown the interface for the system which allows both user and administrator functions. Images are retrieved from the file system once they are uploaded by double-clicking on the image file name under the user directory in the normal tree structure. The timer in the upper right hand corner is then incremented based on the appearance of the image in a separate window using the KDU_SHOW application.

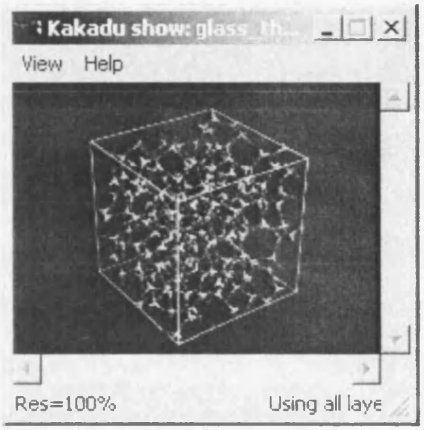

Figure 3-11: Example KDU_SHOW window with resolution scaling indicator

The final application screenshot is that of the display of the resulting image based on the request. The window is a separate interface which indicates the resolution scaling of displayed image and the interface allows only limited interaction with image itself. This ability to limit resolution of the displayed image without re-rendering coupled with the ability to remove certain features of the Kakadu window allow great flexibility in providing a range of features based on the service level itself [132]. A wealth of information about the Kakadu software system and the JPEG 2000 file format can be found at http://www.kakadusoftware.com.

3.3.3 MANIAC Prototype Results and Discussion

The Kodak JPIP web service combined with UCL's PROMILE/XMILE network infrastructure demonstrated one activated service application in the area of imaging. The system was tested by three users on several sets of standard jpeg image sets which totalled more than 40 images. The image tests were conducted over a period of several weeks. The system operated as designed with the levels of service being applied correctly to the different requests. The highest level of service delivering the high resolution images with expedited packets on a closed, uncongested network segment. Due to the nature of the active network and the potential for impact to other traffic, the MANIAC system was only permitted to operate in Kodak on its own private segment which did not permit the analysis of the quality of service in an environment with

realistic traffic. One observation on the system that was confirmed, however, was that highest levels of service subscription did not necessarily complete the request in the shortest user time as has often been reported in the literature in active networking. The simple explanation of this is that the higher resolution image required more time to send even with expedited packet handling and the expedited packet improvement was less than the increase in the transfer size. On average the gold level transmission time for the sample set of images was 10% longer than the bronze level although the issue was that although the bronze user may have completed the request to receive the image sooner, the image was of such a resolution that it would not make a suitable photographic print. The testing and feedback from the system was that the user needed to be able to change the service level for a request based on the desired output format for the image. So, essentially in some cases, the low resolution version was all that was needed and speed was in fact the important parameter while in other cases, a print of the image is desired and the user is then interested in upgrading the service to get the higher resolution. This fact confirmed the need for some capability to be unbundled from a set of services and to be available a per use fashion from time to time. The conclusion was that depending on the context of the request, the basic set of services may have to be unbundled temporarily to allow an increase in feature set and the user was, in fact, willing to pay more at that time for this temporary increase in functionality.

The experimentation, in the form of the prototype confirmed several key concepts for the active networking and cross layering in a QoS environment. The prototype demonstrated that a form of network cross layering using independently developed web services could be achieved commercially. It was also noted that these individual services can be delivered over an active network or over the traditional network. In the traditional network the lower level network adaptation is missing, however it does demonstrate the backward compatibility. The design of the overarching service level agreement provided the process step which unified the network engineering with web service development although once in place, the teams could then implement their component independent of each other and even geographically separated. This is again a benefit to commercial enterprises that operate distributed development groups. The

critical design element was a shared view of the adaptation rules and a shared language for rule description. Once the system was implemented, the use of the structured SLA provided a means to quickly validate the rules at the layers to ensure consistency. This shared view was facilitated by the construction of the service level agreement and its representation in XML.

3.4 Context Collection and Adaptation Subsystem (C2Adapt)

A direct result of the work in the MANIAC project was a clearer understanding of the possibilities of network adaptation for imaging services and insight into the higher value service which could result. However, it was also clear that the adaptation is driven by the needs of the user and their preferences as well as an understanding of the environment in which they would received the services requested, not just the network or the device. From MANIAC, it was clear that adaptation on behalf of the user was possible and it created a new vehicle for service deployment. It was also possible to coordinate separate aspects of adaptation, each operating on different components of the complete solution. But this was a first step only. Adaptation though, requires one or more fundamental variables upon which to transform the original content or service into its higher valued, adapted form. In the MANIAC system, these variables were the image resolution and the packet handling. The next research and engineering challenge was to determine within the domain of imaging services what these variables for adaptation are, or could be in order to deliver value-added services.

After an initial review of the literature and a framing of the general question into a research problem, a second joint investigation was started to look into the nature of context for imaging applications and context as a part of the software engineering design process. A project called the Network Ease of Use (NEU) was sponsored by Kodak and involved a team of research scientists in Harrow, England and Paris, France who were tasked to research, design, develop and evaluate a context-aware imaging application. The details of this project are reported in other publications which are internal to the company; however the author's role and direct experience in this effort provided subsequent motivation for the performance analysis component of this research

programme [133;134]. The focus of the NEU project however, was to experiment with the level of adaptation possible in a dynamic, mobile device environment where the features and details of the mobile device might change not only day to day, but perhaps even during the service delivery timeframe itself.

3.4.1 Context Adaptation Application for Imaging Services

As part of the NEU project, a context-aware architecture was developed by a team of researchers including the author. This architecture provided the foundation to dynamically compose imaging-centric web services. These services were designed to collect context from a destination device and other static preferences, from an information store called the user profile, to evaluate the input based on specific mapping rules and render the resulting set of content and code for the requestor's environment and situation. The general description of the resulting architecture of the "simultaneously heterogeneous" type, with the functions implemented as independent components which communicate through message passing but mediated through the use of a rule-based system or virtual machine [135]. The architecture was implemented as a set of fundamental services using the web services model. The service which was ultimately offered to the consumer was a composition of these fundamental services based on the task and the context surrounding it. For example, the display of an image was a consumer service composed of several sub-services each of which could be called from yet another service. The design called for the use of centralised controller of this service configuration module which was essentially responsible for preparing the equivalent of a software "bill of materials" and then ensuring the services were executed according to the instructions within it. A critical component of the design for the overall system was the domain specific definition of context awareness that was developed. In order to design the architecture and later build the prototype itself, it was clear that a shared understanding was needed among all the developers and designers involved on what contextual information was going to be collected, how it was to be collected and most importantly how it was to be used within the system. There are many context-aware architectural studies and research reported in the literature. Like other such systems in software development, context-aware architectures are many times tightly

designed to the domain under study, linked by the descriptive semantics of the important elements of context. Unfortunately, each project often addresses the specific needs of the problem area under study and if the elements of context are embedded in the solution, it is often impossible to reuse the system in other domains. True separation of the "form from the function" of the system is difficult and is commonly mentioned as the reason why so many system architectures and systems exist. This holds particularly true for context adaptive systems as inherent in the design of the system is its responsiveness to an informational subset of the real world which is highly domain specific.

One of the most significant gaps in published results from engineering research in the context adaptation area is that of performance analysis of systems as architected. The importance of this research on engineering projects cannot be overstated due to the fact that when creating novel services, knowing the possible bottlenecks in these systems can save valuable time in development. It was learned from the parallel effort in the NEU project that collecting and analysing contextual information imposes some significant performance implications on online services and as will be shown, this is directly proportional to the traffic entering the contextual system and to the time it takes to perform the collection and adaptation. The problem is that context parameter definition in the commercial service grew very quickly without the knowledge of performance implications. It was also observed that the requirements on the operation of the context-aware systems are problematic given the dynamic nature of device, users and information involved in such systems. The changing location, preferences, network conditions and other information that a system must adapt to in a very short span of time in order to deliver optimal services present a formidable design and engineering problem [26].

The NEU context design cornerstone is a 4-tuple with four main contextual elements; device attributes, network characteristics, user preference. It should be noted that this was not intended to be a generic approach for context-aware architectures but was specific to an imaging service intended to be delivered as a composition of web services.

These specific elements are taken from the internal project report, modified and presented in the table below.

Contextual Function	**User Preference** ***(content)***	**User Environment (location/time specific) (user)**	**Device Capabilities (device)**	**Network resource (network)**
Capture/ Upload	Flash/No flash Orientation Scale	Conditions, Location, Calendar Events	Sensor data, Vocal caption, Surround, Image sensor characteristics Power conditions	Location, network storage solutions availability
Organize	Local indexing Preferred objects selection	Noise (short movie relevant Light levels	Interface (stylus, vocal, keypad...) Screen characteristics Power conditions	Image source search Preferred image library sources
Share	Contact List Preferred method	Calendar Local events	Interface (text or vocal input) Connectivity with other devices Power conditions	Supported image protocols (SMS, MMS, email...) Bandwidth Connection Opportunities

Print	Layout Preferred supplier Product	Conditions Location	Pairing status PAN, LAN Peripherals Power Conditions	Available and preferred printers/kiosk
Edit	Layout Orientation Scale Navigation	Light levels	Screen characteristics – bits per pixels, palette, size, resolution, orientation Power conditions	Available and preferred access
Browse/ Search	Layout Orientation Scale Navigation	Light levels	Input methods possible	Available and preferred access

Table 3-1: NEU Imaging Service Contextual Elements

The system was designed around the use of a 4-tuple as mentioned previously, which is composed of the application specific elements under context, user, device and network. The 4-tuple concept helps to aggregate the elements of context into a standard form, much as the service level agreement helped to unify the two layers of the cross layered active network system in MANIAC. The 4-tuple also helped the engineering team conceptualise the possibilities in a form much like the triplet structure in RDF which allows for machine integration of concepts, thereby easing some of the complexity in the development process itself. From this set of potential context variables by function, a set of application specific variables were created and are described in the XML presented below.

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getContextResponse xmlns="http://tempuri.org/">
      <getContextResult>
        <device>
          <owner>string</owner>
          <oemManufacturer>string</oemManufacturer>
          <keyboard>string</keyboard>
          <mouse>int</mouse>
          <touch>boolean</touch>
          <audio>boolean</audio>
        </device>
        <capabilities>
          <technology>string</technology>
          <screenWidth>int</screenWidth>
          <screenHeight>int</screenHeight>
          <screenWithClientArea>int</screenWithClientArea>
          <screenHeightClientArea>int</screenHeightClientArea>
          <horizontalSizemm>int</horizontalSizemm>
          <verticalSizemm>int</verticalSizemm>
          <bitsPerPixel>int</bitsPerPixel>
          <numOfPlanes>int</numOfPlanes>
          <numberColours>unsignedInt</numberColours>
          <pixelRatio>int</pixelRatio>
        </capabilities>
        <osVersion>
          <dwMajorVersion>int</dwMajorVersion>
          <dwMinorVersion>int</dwMinorVersion>
          <dwBuildNumber>int</dwBuildNumber>
          <dwPlatformId>string</dwPlatformId>
          <szCSDVersion>string</szCSDVersion>
        </osVersion>
        <powerStatus>
          <ACLineStatus>string</ACLineStatus>
          <BatteryLifePercent>unsignedInt</BatteryLifePercent>
          <BatteryFullTimeLife>unsignedInt</BatteryFullTimeLife>
          <BatteryLifeTime>unsignedInt</BatteryLifeTime>
          <BatteryCharge>string</BatteryCharge>
        </powerStatus>
        <systemInfo>
          <dwNumberOfProcessors>int</dwNumberOfProcessors>
          <memory>int</memory>
          <wProcessorAchitecturec>string</wProcessorAchitecturec>
          <wProcessorTypec>string</wProcessorTypec>
        </systemInfo>
        <userAgent>
          <userAgent>string</userAgent>
          <manufacturer>string</manufacturer>
        </userAgent>
        <network>
          <CurrRate>int</CurrRate>
        </network>
      </getContextResult>
    </getContextResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 3-12: NEU Imaging Application Context Schema - Baseline

The method for composing a complete instance of the 4-tuple for use in adaptation involved the following steps:

- Identify the service requested
- Lookup the service context components
- Populate the 4-tuple – call dynamic discovery modules – using proprietary multimedia messaging process
- Pass a message to the application builder with the elements of the 4-tuple
- Evaluate the rule base, retrieve appropriate content
- Pass pointer to content to rendering engine to render based on dynamic context
- Render content and produce navigation elements
- Pass pointer to application components
- Present service to destination device through a browser

The full detail of the operation of the overall NEU prototype are reported within Kodak technical reports, which are not public documents. However, from the investigation it was found that the performance of the system even with the restriction of just four variables of adaptation is insufficient for a commercial service even in a private, controlled local area network environment without the latency incurred in crossing the public network or even worse in the performance of mobile and variable signal strength wireless connection. The average time for the system to provide a complete response involving full dynamic collection using a proprietary multimedia messaging service (MMS), context evaluation and rendering was in excess of 96 seconds and in some cases was documented to be as long as 121 seconds. The network environment for test system is shown in Figure 3-13.

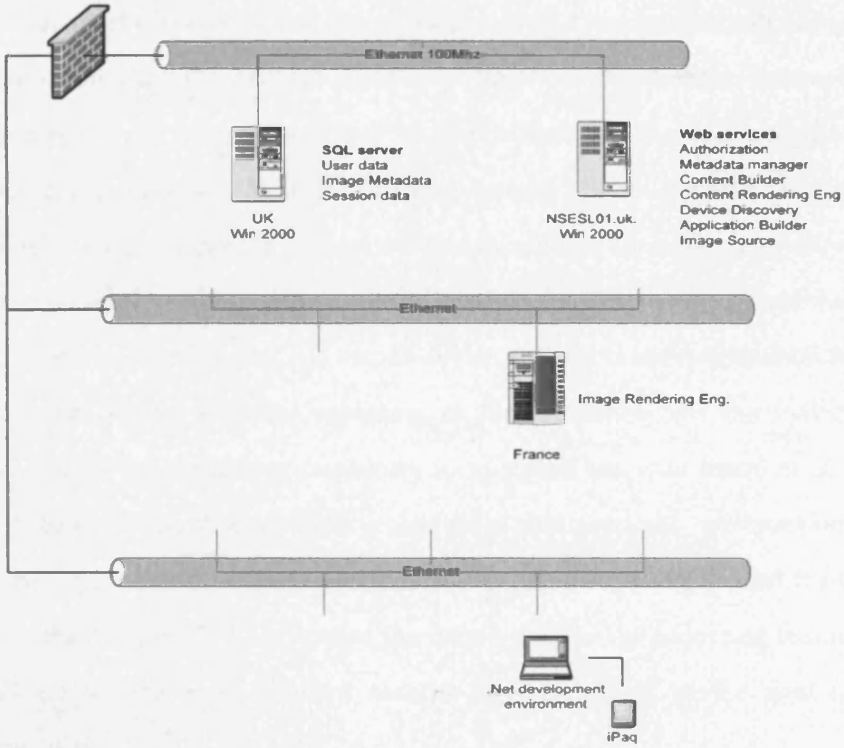

Figure 3-13: NEU Test Network

The initial results as interpreted and analysed by the author showed that the adaptation using web services functioned although further work would be necessary to understand and redesign the systems architecture to reduce or eliminate the performance issues before deployment into a commercial environment would be possible.

3.4.2 Context Engineering Analysis and Re-Design

For the conclusion taken from the NEU investigation concerning performance of context adaptation it was found that the two greatest contributors to the value and the quality of imaging service were found in the context variables from the device and from the network itself. Although the user context preferences were also important, these were captured via the use of a profile directly from the user and specified for the service. It was observed that once these preference were set for a service, it was easy to allow the user to direct the contextual change. For example, the user has access to preference changes on the main application interface so that as the situation changes, they were easily able to direct the application to change orientation, lightness, navigation method,

etc. However, all of these potential directives required a precondition of accurate device attribute information. The second most important context 4-tuple component in the NEU architecture was the network and its parameters which represent the perceived conditions. It was observed that the dynamic nature of the network conditions made adaptation challenging but that it required at a minimum an accurate understanding of the device's network configuration. Additionally, the NEU project outcome reported that within the discovery phase, the actual device specific context collection had several significant faults in the accurate reporting of the attributes and the inability due to proprietary nature of the device hardware to ascertain the true features of the device itself [133]. From these observations, it was clear that accurate information about the nature of the deployment device was critical above all else and supported the population of the rest of the 4-tuple. It was decided the detailed nature of acquiring this information accurately and in the most efficient manner was the focus of the next step of the engineering work.

Before continuing with a deeper investigation of device-specific context, it was important to understand the issues with system operation. A further analysis was completed by the author into the functional components and the prototype infrastructure to understand the possible bottlenecks in the system. It was uncovered and reported that through the assessment phase of the prototype implementation it was found that the device discovery and the adaptation engine were the two key areas where performance was unacceptable. These two major results cast doubt on the commercial viability of context-aware adaptation services using the NEU design and architecture. The work within this engineering doctorate was then separated to focus solely on a deeper study of the context collection subsystem as the principle performance bottleneck and on devising new methods for engineering the context-adaptive imaging services.

With the focus now on the enhancement of context adaptation architecture for novel services, the first step was to analyse the issues by creating an abstract model of the most common design paradigm in context adaptation. The context collection and adaptation

subsystem schematic was created as an abstract model for the purpose of further analysis. The model appears in Figure 3-12 below:

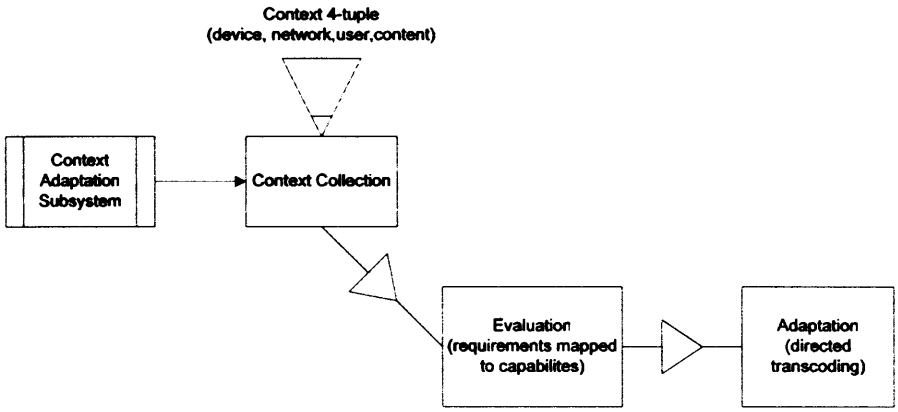

Figure 3-14: Context Collection and Adaptation

The subsystem model depicts three individual components - the first is responsible for collecting the context as necessary for the service triggered by the request, the second composes the 4-tuple and evaluates the adaptation necessary through the call to the adaptation rule-base and the third is responsible for directing the actual adaptation and passing a pointer to the adapted content including any navigational or control items to the dispatch subsystem. While there are many architectural styles capable of implementing a context collection and adaptation subsystem, there are several key requirements for operation of the system that are subtle. One major implication around context is its temporal aspect, not just that the total context variable changes over time, but that the right variables within the 4-tuple are accessible and available on timeframes dependent on the context itself. Meaning that in the case of dynamic discovery of context elements, the availability of that variable is a function of perhaps another variable within the context 4-tuple. By the virtue of this statement then to use the context 4-tuple for a complete optimal solution will require its population to be complete prior to evaluation or the acceptance that a sub-optimal solution is acceptable. The detailed investigation of operational trade-offs or sub-optimal solutions were not a part of this research programme and were left for subsequent future work.

As a starting point, this new abstract model makes several simplifying assumptions. These assumptions serve to make the model computationally feasible with the recognition that they also detract from its scope and limit its utility as a true representation of the system. The first assumption is that in the model the context 4-tuple, *V* contains four sub-elements of context – one variable each for the network, content, user and device, namely *v1-v4*. However, it is an ever increasing number of variables that could be input to the adaptation process under each of these four categories. Artificially then a limit to the extent of the context needs to be established to make the service engineering project tractable. It is the 4-tuple of required elements which is required for the specific contextual service requested. The structured details of the device embedded within the service context specification at the time of the request is currently the dominant design in use in such standard as CC/PP and others [93]. In these standards, a request is made and within the header or within the supporting user defined components of the request itself are the details of the context needed for the service request. This implies that with each new request for service, information will be provided in order to provide the subsequent service, and if context is dynamic enough then there will be issues with accuracy of the context information embedded in the request. A simple example is in the case of responding to simple audio levels found in the area of the destination device. In a request style context reporting system, an audio level will reported for a service which by the time the service has adapted to the audio levels reported and returned the contextual response, the actual audio level will now be quite different. While this is dynamic to some degree, the overhead to provide detailed context with each new request is not scalable in two ways. As the number of requests increases, so does the overhead of providing context and as the amount of context variables increases so does the overhead of the total request.

Principally then, the subsystem of most interest for this component of the study is one which has fundamentally three components; context collection both static and dynamic, evaluation and adaptation. For each service request, the first salient question is whether contextual elements are even required for the QoS agreed with the consumer. If contextual information is required, it is assumed that some level of collection, evaluation

and adaptation service will also then result and all activity must be done within the QoS guarantees as specified in the service level agreement. The nature of the context 4-tuple as defined in the NEU project is an important aspect of the engineering solution and in fact, is so designed as to permit a dynamical systems approach to the matching of service requirements with context elements, a subject which will be discussed in subsequent chapters. It is important also to realize the limitation of the study and its resultant architecture so far. At this point, the focus is primarily the device context itself, but context is a far broader topic which includes any number of outside parameters. This context 4-tuple represents a point in "context space" which is essentially n-dimensional. As context changes in time, the context 4-tuple will create a sequence and one path of the sequence will represent the device changes over time. The changes of the device will create a device subsurface which is a subset of the overall context surface which could be unique for each user and for each service. This is an interesting area for future work, a topological study of the usage of context by user, service and device and it is of particular note as it may have a consistency which would permit predictive inferences to be made over time. The context 4-tuple is more precisely a context matrix with the rows representing each instance of a context variable set. The columns are the context categories and sub-categories of the 4-tuple.

The matrix rows of context elements are not necessarily linearly independent in terms of the data elements within each row in the lower half of Figure 3-14 as there are more components of the same type available for each context element. This creates an issue with the current method for handling context as it requires an intelligent choice be made, a requirement further discussed in Chapter 4. At this point, the context collection and adaptation subsystem has been suitably abstracted out of the NEU architecture and others and was ready for analysis via the appropriate form of performance evaluation model and study. For the purpose of clarity, this abstract model of context adaptations was called C2Adapt.

3.5 Queuing Network Models (QNM)

As mentioned earlier, one of the fundamental engineering challenges for context adaptation services is a clear understanding and prior references on the performance of such systems. There are many architectures and designs, but very little analysis and reporting on the actual performance of the services in those architectures. The objective of the work in this chapter was to take the dominant context adaptation design, abstract the design for analysis and then perform some assessment of the operation of that design. There are two primary methods for evaluating systems such as the context collection and adaptation subsystem (C2Adapt), through simulation or through analytic mathematical models. While simulation enables various forms of experiment with the system in question, the analytic model provides a straightforward answer even in the case of stochastic models [136]. An analysis of the system in question and the objectives of the assessment in terms of measures of performance made the analytic method the best vehicle for the investigation. The goal was to understand the nature of the performance issues of the context collection and adaptation system as it was originally designed and empirically tested. It was clear that every model of a system whether analytic or simulation must make simplifying assumptions in order to be tractable. The empirical data from the operation of the NEU prototype would be used as input to the model. The use of analysis coupled with simulation can improve the overall understanding of the system dynamics and is particularly useful in the study of context-based systems [137]. The simplified high level subsystem model would be used as the "mathematical model" [136]. One further question was how best to model the interaction between the individual components of the subsystem. It is clear that the overall system corresponds to the basic operation of a queue with service requests triggering the need for context collection. These requests arrive to the system, hopefully, from some very large population of potential users or other machines within the network. These requests are independent of each other as is the case of Internet traffic arriving at a website requesting a page. The requests are serviced as a function of their arrival in a first-in, first-out (FIFO) basis. The arrival times can be modelled similarly to that of general web traffic by assuming that the imaging services are offered as a set of

generalized services to the broader Internet community. The most important part of the evaluation is to understand the nature of the delay caused within the three component system. From the literature and analysis, the performance model most suited for this is the queuing network model or QNM. A queuing network model is an example of a continuous stochastic performance model based on Markov's assumption. This means the probability of the context service request moving through the subsystem components is independent of how it arrived there in the first place, it is also independent of what happened to it within the individual components and the total probability of it entering and leaving the total subsystem is unity [138] . This description fits closely the perceived operation of the subsystem as designed and abstracted.

3.5.1 Tandem Network Queuing Model

The C2Adapt abstract model of the engineered context adaptation system is the basis for the operational assessment. This system approach can be viewed as a series of queues where a service is provided a stream of requests which are entering the system from some initial population. The concept of tandem queues can then be used to create the model for evaluation. The question remains as to whether the summation of the performance measures represented by each individual queue, for example average wait time or queue size, can properly represent the behaviour of the subsystem itself. This behaviour has been previously observed in the systems testing of the NEU prototype, so given the limited resources of this study this empirical basis is deemed sufficient. The subsystem model is depicted in the figure below. This represents a simplified context collection workflow sequence; however, in later sections in this chapter, the concept of a parallel set of services implementing the overall subsystem will be explored.

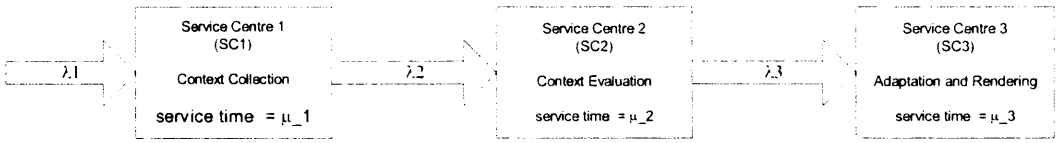

Figure 3-15: C2Adapt tandem queues model for assessment

Within service centre 1 (SC1), the context collection process begins based on the nature of the request. In C2Adapt, contextual variables are defined and are considered to be dynamically acquired from the remote device. The term *dynamic* is defined here as a variable which cannot be assured to has the same value over the timeframe of one service request session. The service request session is defined as the time interval between the request, the adaptation and the delivery of the requested content within context. In this regard and as already presented, device attributes and operational aspects are considered to be a primary source of dynamic context. As a specific example, during a service session, a user may be forced or may choose to change some key software components on the device from which the request originated. One of the most frequent causes of this is loss of power, particularly in smaller, more mobile platforms such as PDAs, mobile phones and the new hybrids of the two. The second component in the system is service centre 2 (SC2), it is tasked with taking the 4-tuple of context and looking to the rule-base to determine the extent of the modifications necessary to compose the appropriate adaptation. This operation is done as a separate step and the pointer to the content with the necessary steps is then passed as a message to the adaptation engine. The adaptation engine, or service centre 3 (SC3), takes the necessary modifications or instructions and the pointer to the original content element and passes control to a module which performs the modifications and transformations storing the transformed content and passing the pointer to the application where it supplies the new content as requested by the user. The model represents largely a sequential operation with one set of servers instead of a set of parallel servers operating on a service request stream. The contribution provided by this model is an understanding of the interactions and performance of the components of the typical context collection and adaptation subsystem, this is not intended to be a performance evaluation of how to provision the appropriate computing resources to handle a specific workload.

At this point, the inner workings of each service centre are not required given the empirical service times provided from the prototype itself. These timings serve as the deterministic or fixed variable in the queuing network model. The mathematical model is constructed using a purpose built analytical tool for QNM analysis called QTSPlus

[139]. Several experiments were conducted maintaining the service time, μ but varying the arrival rate. The results are shown in Figure 3-20 through 3-23.

3.6 QNM Analysis

Queuing network models are mathematical abstractions themselves, developed to assist with the understanding of complex network traffic. Understanding the different possible QNMs is important as each is applicable to different circumstances and for studies of specific objectives. Using the context collection and adaptation model outlined above, a set of analytical assessment runs was planned, but the first activity in the experiment was to determine the set of parameters which most closely model the operational behaviour in the C2Adapt system. From the parameters of interest and the desired results, standard models of queuing network analysis were compared to ascertain fit. Several models of queuing systems could be feasibly used for the analysis in this case, from simple Markovian models, Erlangian Models and Jackson Network Models [30]. The following variants were reviewed for use in the assessment:

Infinite Population Markovian Models – essentially Poisson arrivals with infinite calling populations. Queue disciplines in these cases are first in, first out (FIFO). These are steady state models which are most accurate over long time periods and in those time periods, the system in question is assumed to be in equilibrium. Systems of these types are also assumed to have unlimited capacity.

Erlangian Models – a model often used in the study of telecommunications system and a likely candidate for the simulation of the C2Adapt. Erlangian models are a more advanced example of the Markovian model and are specialised in that they provide a greater degree of modelling flexibility to model the interarrival times. In simple Markovian models, the arrival and service times are exponential; in Erlangian these rates are variable and are represented by the Erlang set of probability distributions. The Erlangian model is closely related to the exponential Markovian, however it is particularly useful when describing cyclic or repeating events [30]. However, the

Erlangian applies mostly to service centres that are independent and have parameters which are exactly the same in all cases. These aspects of the Erlangian queuing model make it unsuitable for the context model which can have some dependence between the services and the attributes of each service centre are different.

Jackson Network Models - a set of network nodes, say n in number, where each of the nodes acts a service centre and is largely independent of the other nodes within the network in terms of the service and arrival times. The requests or customers arrive from outside of the nodes themselves following a Poisson process. It is important to note that queue times are independent and that they are exponentially distributed around the mean service time for each service centre. This mean service time for each service centre is also an independently varying variable. The nature of these networks was developed by Jackson and in fact, he showed that the attributes of each node could be multiplied to express the behaviour of the system as a whole in what is referred to as the product form solution. While this is an extremely useful result, it does not imply that a simple mathematical solution is available for these models however, rather an approximated solution formed by the stochastic result of each queue [30]. There are several variants of the Jackson network models which have evolved over time to accommodate the needs of system modellers. Examples of such variants are the open Jackson models which allow requests/customers to enter at any queue in the tandem network from outside, they then complete service at some time which exponentially distributed and will route to the next node based on some probability matrix. There are also closed Jackson models which essentially limit access to the service centre-based systems through an originating node and then allows requests or customers to proceed in a stepwise fashion through the service centre, neither flowing backward nor entering a queue out of sequence.

By comparing the operation of the abstracted model, C2Adapt, with the operational aspects of the different queuing models showed that the family of Jackson network models was the most appropriate for the analysis and modelling of the context adaptation system. The Jackson model provided several important benefits to the study.

First, the Jackson network is a series of independent nodes each following a Poisson traffic distribution. Second, the technique also has to model the parallel service centres at each stage of the context process which is key to design of the overall service. And lastly, the modelling technique also has the ability to represent an environment with multiple service or customer classes which is a key consideration for C2Adapt which implements differing levels of service.

3.6.1 Tools for Analysis of Queuing Network Models

With the modelling approach formulated and some of the abstract components already ready for interpretation, the next step in the process is to evaluate the available tools for the modelling environment. An exhaustive list of tools for stochastic and probabilistic analysis exists for computing systems and networks which are engineered for performance evaluation. The first choice of a tool was the MATLAB® development environment from Mathworks. Using this, the network models can be implemented through technical programming. Another class of tools for use in the analysis are purpose-built queuing network modelling packages which are designed for experimentation with queuing network models to solve specific design and engineering problems. Due to the great number of choices in this area, the selection of a tool to assist with the assessment was quickly needed. Those listed below were reviewed thoroughly and simple network models were created in each to determine the effort necessary to fully assess the C2Adapt subsystem:

- MATLAB® R13 from The MathWorks
- Mathematica® R6 from Wolfram Research
- QTSPlus v 2 from Gross and Harris

Each tool had a slightly different set of advantages and disadvantages as one would expect. Both MATLAB® and Mathematica® are excellent mathematical development toolkits providing a high degree of accuracy and flexibility. This is at the cost of time to implement and develop the basic functions and mathematics of the QNM model itself as

these tools require programming to implement the systems' components. QTSPlus, however, provided a robust environment with a good degree of accuracy and pre-programmed models supported by academic foundations, the classic queuing network model text by Gross and Harris [30]. To start, the level of performance assessment is at the overall application level, it is not intended to research or further development of the theory of Jackson network models. The use of the network models is to understand the nature and difficulties in systems design and performance, not in the detailed assessment of the nature of the mathematical treatment of such queuing models. Hence, it is necessary to find a tool to use which could be specifically applied to the problem at hand, and did not need to be developed further.

The tool selected for the experimentation is QTSPlus from Thompson, Harris and Gross[139]. The QTSPlus software provides an interactive framework for the execution of several type of queuing network models and provides a means to create a graphical representation of the results as well. It also provides a multitude of models, as depicted in part in Figure 3-18 and Figure 3-19, approximately seventy-eight (78) in total. This allows the systems analyst to quickly compare the different models to understand the implications of differing parameters and design decisions on the outcome.

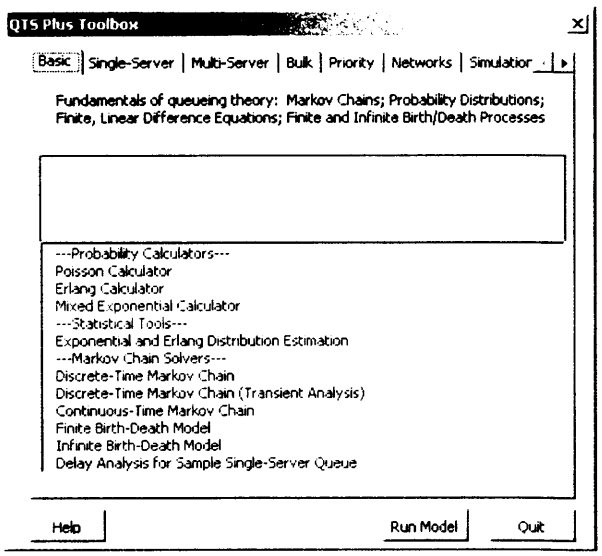

Figure 3-16: General queuing theory models from QTSPlus 2.1

The QTSPlus tool provides specific models in the area of Jackson network for tandem or series queues, namely M/M/C, Open, Closed (Buzen or MVA approaches). It also provides the ability to analyze queuing networks with more than one class of service which is an important area for context adaptation research in QoS networks where users can subscribe to higher levels of service.

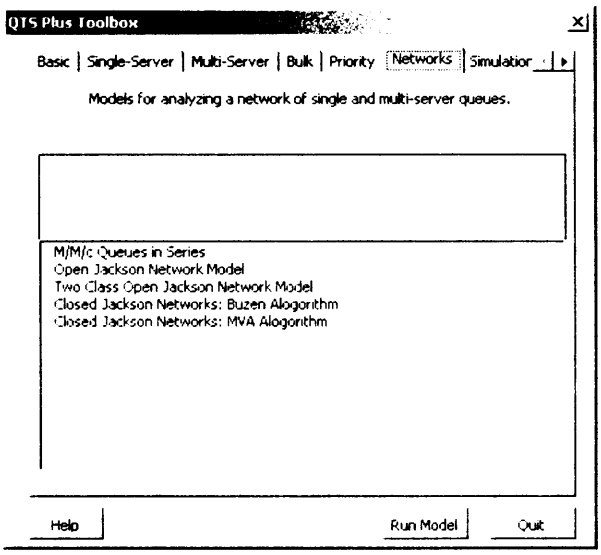

Figure 3-17: Queuing Network Models in QTSPlus 2.1

The key parameters of interest from the analysis are the total long running wait time (W) as experienced by a service request. This result is often called the sojourn time as it represents the average steady state time necessary for a request to complete through the three node system. The additional key parameters which are used in the analysis are:

μ = the service rate for each server

γ = the average arrival rate to the node

λ = the calculated total mean flow rate of the request in seconds – includes both γ and other flow to the node

ρ = the calculated utilisation of the servers

L = the long run average number of requests in the system

The assessment methodology was designed with two sets of parameters. The first set, *A*, which corresponds to the average service times for empirically observed for the context collection, evaluation and adaptation process. The service times include the complete roundtrip time and are not separated out to include the network transmission or the individual component operation times. The reason for this approach is that one very important parameter in the study is the mean service time per stage as perceived by the service request at each node or stage of the overall process. The high level analysis of each stage is then combined to indicate the overall performance of the system. This approach is supported by the mathematical foundations of the Jackson network models [30]. An analysis of the operation of the finer-grained components of each stage requires significant effort, and the details of such analysis were not of interest at this stage. The exact implications of the performance of each constituent part of the service operation are outside the scope of this systems evaluation and are left for future work.

The second parameter set, *B*, represents the parameterisation when operating in an alternative architectural style which increase the service rate of the collection phase. The purpose of the second parameter set is to see the effect a slight improvement in service rates has on the three service centres and on the ability to meet a selected, but properly scaled QoS target. This parameter set will also demonstrate that a slight improvement in performance of such a network can have a significant impact on the cost of running such a network service centre and on the ability to meet the constraints of operating in a QoS business model. For each set of service centre parameters, two separate runs of the simulation are performed. The first set is in the unconstrained business model, one without a total sojourn time limited by the service level agreement and one with the sojourn time limited by the SLA. In the Jackson network model, the sojourn time represents the time necessary for the service request to complete the tandem network of nodes and exit the system. This is then mapped against the service level agreement. In this study a generic performance agreement is used which includes the service required to retrieve, transcode and present a high resolution image with average media adaptation time as represented by the parameter set described in detail in Chapter 3. The first in the series of simulation parameters is given by the figure below:

OPEN JACKSON QUEUEING NETWORK

To setup new problem, enter number of nodes in queueing network.

Number of Nodes: 3

Enter maxium size for marginal probabilities.
Max Marginal Prob: 10

Press "Solve" button to compute Jackson performance metrics. **Solve**

γ	μ	Threads	Node	P	ble	
1.	0.0333	100.	COLL	0.	0.9	0.1
0.	0.056	100.	EVAL	0.05	0.	0.95
0.	0.025	100.	ADAPT	0.	0.05	0.

Figure 3-18: Network with 1 request per second

An example of the basic simulation run details are presented in Figure 3-20, the service times for each stage of the context adaptation system are given, the interarrival rate of 1 service requests per second and then the routing matrix for the open Jackson network which indicates are largely feed forward network with only 1% of the traffic flowing straight back to the collection stage and 5% of request requiring further context collection. The notion of servers was converted to computing process threads as in [140]. All subsequent simulation runs were done using the same routing matrix, but adjusting the inter-arrival rate and the service rate.

System Performance Measures

Total number in the network (L)	91.445928
Total sojourn time through the network (W)	91.445928

Node Performance Measures

Node	COLL	EVAL	ADAPT
γ	1	0	0
μ	0.0333	0.056	0.025
Servers	100	100	100
λ	1.049876	0.99752	1.0526316
ρ	31.53%	17.81%	42.11%
L	31.527808	17.812857	42.105263
W	30.03003	17.857143	40

Marginal Probabilities

0	0.000000	0.000000	0.000000
2	0.000000	0.000003	0.000000
3	0.000000	0.000017	0.000000
4	0.000000	0.000077	0.000000
5	0.000000	0.000274	0.000000
6	0.000000	0.000815	0.000000
7	0.000000	0.002073	0.000000
8	0.000000	0.004617	0.000000
9	0.000002	0.009137	0.000000
10	0.000005	0.016276	0.000000

Figure 3-19: C2Adapt Representative Results

A set of representative results from the QTSPlus analysis run is shown in Figure 3-21. The details of all simulation run results can be found in Table 3-2. Each run for each parameter set was done assuming a target QoS response time of 90 seconds, meaning that each request should experience a total *W* of less than 90 seconds in order to be acceptable to the user.

Constants	
$\mu 1$	0.0303
$\mu 2$	0.056
$\mu 3$	0.025

γ	$\lambda 1$	$\lambda 2$	$\lambda 3$	L	Threads	W
1.000	1.044	0.888	1.053	92.434	100.000	92.434
2.000	2.088	1.776	2.105	185.198	100.000	92.599
3.000	3.133	2.664	3.157	277.302	200.000	92.434
4.000	4.177	3.552	4.210	369.794	200.000	92.448
5.000	5.222	4.440	5.263	462.170	300.000	92.434
6.000	6.266	5.329	6.315	554.615	300.000	92.436
7.000	7.310	6.217	7.368	684.644	300.000	97.806
8.000	8.355	7.105	8.421	739.473	409.000	92.434
9.000	9.399	7.993	9.473	832.947	409.000	92.549
10.000	10.444	8.881	10.526	924.341	500.000	92.434

Table 3-2: C2Adapt Models Performance

3.6.2 Analysis of QNM Assessment Results and Discussion

The primary motivation for the analysis was to understand the operational and performance-related issues with the standard architecture for context adaptation services. The complexity of the system designed to perform context adaptation was quite high and therefore, to achieve a first understanding required an abstraction of the most important details only. Several simplifying assumptions were made in order to gain insight into the NEU system operation, the model developed in this work defines a plausible system model which can represent the systems behaviour. The model exhibits the same general behaviour as the empirical evidence on the operational prototype NEU system. Clearly the system is unable to meet the QoS response time requirement of 90 seconds in all cases. It is also noted but not a part of the QNM analysis, that at least three (3) of the thirty-four (34) context variables were improperly reported by the device or not available at all. The assumption in the QNM model is that all information is delivered and is usable to deliver on the quality aspect of the service. that without further optimization and architectural re-engineering, this system will be unable to cost-effectively operate within an environment governed by a service level agreement.

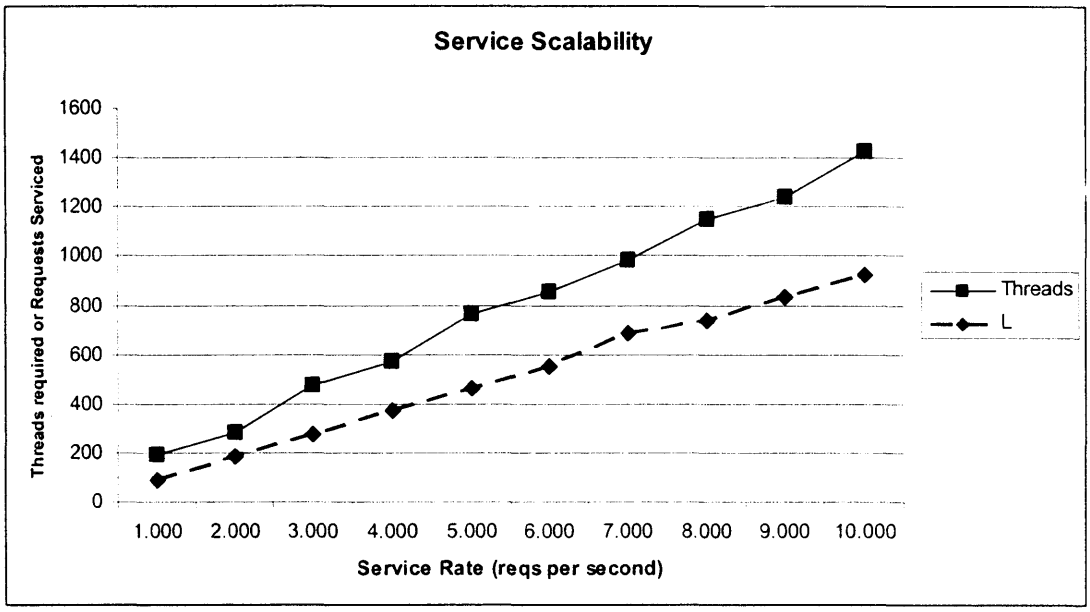

Figure 3-20: Parameter Set A: Unimproved Metrics in System Operation

The overall operation of the model system and, in particular, the long service times for context collection and adaptation stages were clearly a bottleneck and prevented the context-aware adaptation from occurring within the preset response time objective. The detailed results showed that the tandem queuing network was blocked at service centre 2 for a large proportion of the time as the number of arrivals increased into the queue for service centre 1. Although the fact remains that the number of servers increases linearly which is not a desirable performance situation. It is worth repeating that the number of servers in this analysis is not to be construed as the number of hardware devices commonly called "servers" but rather the number of threads or processes that are capable of serving a request. For example, most multi-threaded web servers are capable of serving hundreds of simultaneous requests, e.g. Linux servers have a max thread limit of 512 per server. It is shown that the number of servers in this study would increase to a greater extent then the number of requests serviced as depicted in Figure 3-22. This is a significant result and clearly shows a lack of scalability in the C2Adapt model as thread and process resources are tied directly to hardware expense. However, the predicted performance and the issues with scalability provide sufficient motivation to investigate alternative system architectures.

It is clear, however, that the design of the overall C2Adapt system model had to be done differently. Optimization is necessary in order for a system such as this to move into the next stages of product development and hopefully further into commercialisation. The result of the engineering study served to indicate that to improve the ability of such a system in a commercial setting, it was critical to find alternative designs to improve the overall performance of the system. Three primary objectives for a new architecture for context adaptation were developed and a new design was formulated that would provide the following:

- Reduce the overall sojourn time by improving the service rate in one of the stages
- Improve the accuracy of contextual reporting
- Improve the scalability

Alternate solutions were reviewed over several weeks including the re-design of the implementation architecture, optimisation of the code particularly in adaptation, load balancing and partitioning of the software components in a manner to reduce the issues with scalability, were all considered. While the optimisation of the adaptation service promised to improve the service rate and improve scalability, it would not correct the accuracy of context variable reporting. In fact, the adaptation service is completely dependent on accurate contextual variables to perform its service to the agreed service level. Enhancement of the context collection service could improve its service rate, improve the accuracy of variable deliver and ultimately improve the systems scalability. It was decided to focus on enhancing the context collection service by a re-design of the architecture and optimisation of the process. It is important to review possible architectural alternatives to this style of context reporting in the field [82]. It is clear then the hybrid architectural alternative of static and dynamic device context discovery is a potential feasible approach for future context adaptation systems such as the one subsequently designed in the programme. To summarise then, a design conclusion is reached that proposes that some aspects of context will remain realistically fixed over

long time cycles and are usually only changed with new model introduction. These are termed static context attributes, the new context 4-tuple depicting this partitioning is shown in Figure 3-21.

Service Name	Unique ID	Service Level	Context Category	Dynamic Element	Static Element
Media_album	03943AE	Gold	Device	OS Level	Screen_Size
				Browser Version	Palette
					Bits Per Pixel
			Network Primary	Current_Interface	

Element	Type	Value
OS Level	Dyn	Linux_RH_7.2
Browser	Dyn	Moz_
Network Primary	Dyn	802.11_b

Figure 3-21: Sample Linked List of Partitioned Context

This alternative architecture schematic where context is queried from a centralised data source in part and discovered in part by a remote service is shown below:

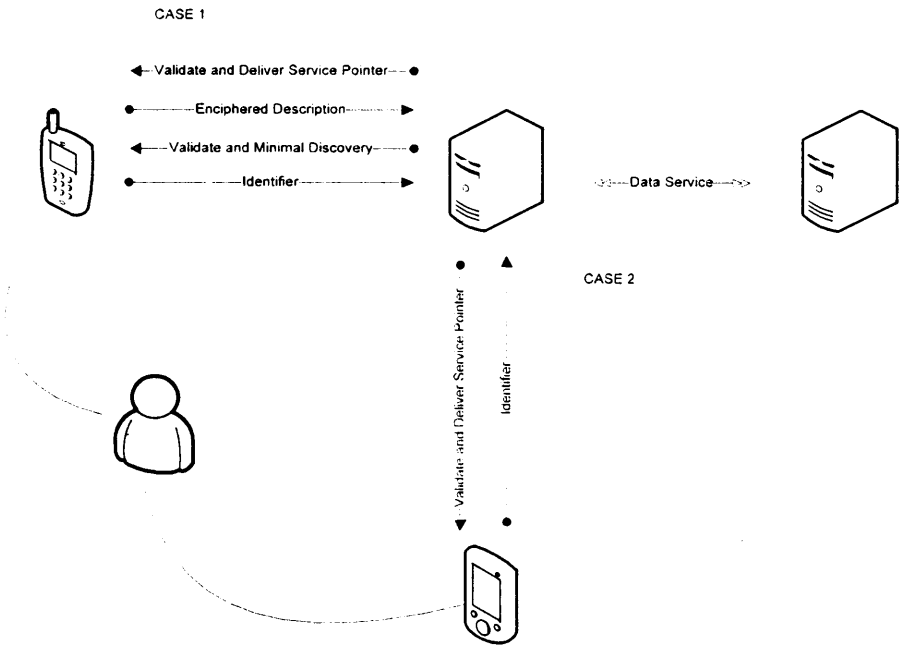

Figure 3-22: Basic contextual partitioning concept

The partitioned model works using a combination of static and dynamic context much in the same style as the context widget approach coupled with the discoverer agent as reported in [19]. Yet another alternative method for context information gathering is that of the centralized listing of features of a device for example which can be used for adaptation. This type of model is in use today in several online content delivery networks. In this model, a subscription to the device description service is provided to a developer who can use it to validate the device context against the request and provide adapted services. These lists usually contain static features of a device which can be used by the application developer through a series of lookup tables. The service engineer/developer can then develop specific evaluation and adaptation modules around the device context repository concept. The most obvious issues in this partition device context lookup approach are questions of scalability and the ability to support true dynamic context information. Using the second parameter set, B and the QNM models provides guidance on the improvement that would be necessary to meet the response time set. The model was updated with a 10% improvement in service rate in context collection and the results are shown in Figure 3-23.

Constants

$\mu 1$	0.0333
$\mu 2$	0.056
$\mu 3$	0.025

γ	$\lambda 1$	$\lambda 2$	$\lambda 3$	L	Threads	W
1.000	1.044	0.888	1.053	89.328	100.000	89.328
2.000	2.088	1.776	2.105	178.987	100.000	89.493
3.000	3.133	2.664	3.157	267.986	200.000	89.328
4.000	4.177	3.552	4.210	357.374	200.000	89.343
5.000	5.222	4.440	5.263	446.644	300.000	89.328
6.000	6.266	5.329	6.315	535.984	300.000	89.330
7.000	7.310	6.217	7.368	662.907	300.000	89.323
8.000	8.355	7.105	8.421	714.630	409.000	89.328
9.000	9.399	7.993	9.473	805.000	409.000	89.444
10.000	10.444	8.881	10.526	893.288	500.000	89.328

Figure 3-23: Enhanced Context Collection

The long run sojourn time, W is now within the 90 seconds response time while the thread or processes are the same as the original C2Adapt model. With the enhancement of service rate, the overall service can now meet the service level in all cases for all

service request rates as long as the thread resources are allocated appropriately. The next step was to look at the scalability of the service as was done before for the original model. The service scalability diagram which depicts the general trend of threads required to service requests processed for different interarrival rates is shown in Figure 3-24.

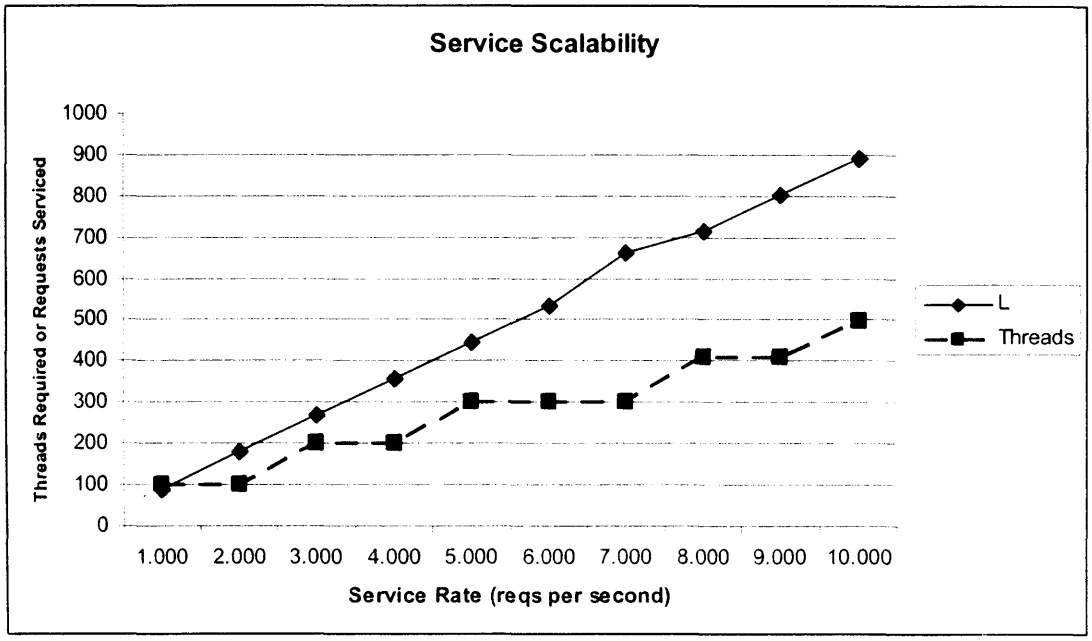

Figure 3-24: Service Scalability with Enhanced Collection

It is clear that the trend now is that as the service requests increase, the thread or processes also increase but at lower rate, thus making the service more scalable as more requests can be serviced with a given set of resources. The conclusion is that enhancing context collection by 10% will improve service rates and overall scalability so that the overall service meets the specified service level. The final step is to show that the context partitioning architecture provides this level of improved service rate and that it improves that accuracy of contextual variable reporting. This will be the subject of the next chapter.

3.7 Summary

There are several small scale investigations described in this chapter; all are focused on the goal of understanding the inherent engineering problems found with the use of

cross-layered context adaptation architecture. From the analysis of the current state of the art in context adaptation architectures to the development of a feasible prototype for the cross layer adaptation, these efforts each open a new path for investigation. The scenario of an image transcoding system implemented with standard web services development tools, in this case, Microsoft's .NET running over the PROMILE active network architecture provided clear insight in the need for a consistent definition of context and the integration with the terms of the service level agreement. The proof of concept also demonstrated that connecting the terms of the service level at both network level and the application level simultaneously was an acceptable method to adapt content both to the bandwidth and to the context of device within the service level. Although a limited single element adjustment, the MANIAC framework did provide the ability as specified in the user scenarios.

The realities of deployment, however, could not be overcome. The fact that as a service provider using an existing networking infrastructure, there were severe constraints on the ability to create novel services that were integrated deep within the network itself. The ability to provide innovative, value-added services across the network today requires a close alliance with the infrastructure provider and it is well published that this is still largely a proprietary, closed system in many respects. The most striking result was the limitation on the resulting perceived quality of the rendered content. Although it was adapted to meet the purchased service level, and could be adapted to deal with bandwidth variations, the quality of the content was still largely in control of the capability of the device. This was of course, thought to be a consideration but it was observed to be so variable amongst the test devices that it raised the question of the benefit of adaptation in QoS-constrained environments if the quality was so negatively affected by the capability. In some cases, the observations indicated that given this, the QoS target was still not attainable as it may have arrived within the transmission time limits but the quality side of the equation was still not met.

This conclusion led to a deeper investigation into nature of a QoS constrained system, which would look into problems of accurately capturing the capabilities of the device as

well as operating within the limits of the service level agreement. Research into such contextual systems revealed a considerable body of work, some solutions in the market and still many questions concerning the engineering and commercialisation of such systems across the networks available today. Through project work internal to Kodak Research as well as independent network modelling, the results indicated that device capability and in particular dynamic device discovery of capabilities was inaccurate and could not be trusted to do the adaptation of content. This was evident in display palettes and in other areas of device features which negatively affected the quality of the imaging application in particular. The very nature of the specification of capabilities was disappointingly imprecise and slow, even using different styles of discovery from server initiated client reporting to a permanent thin client which reported capabilities as needed. Each approach suffered from performance difficulties, or an architectural flaw or from fragmented, incomplete standards associated with the technology. This results in an implementation which would not scale well to the diverse numbers of devices that a content service provider would be required to deploy to realize an acceptable business case. A further understanding into the nature of the context adaptation service-oriented architecture was gained through the analysis and modelling of simplified C2Adapt system using a queuing network model. Using queuing network theory, a model was developed to outline the sensitivity and issues a service provider might be faced with based on some of the limited empirical data available. The result of this analysis was again interesting as it depicted a situation where the scaling of the service within a service level constrained environment would basically follow the rule of "capital optimisation" or the need to increase the server farms for the service centres which perform the work to the point where no business case would be viable. The network model also indicated that an improvement of 10% in the service rate would improve the overall system performance allowing it to meet the response time target set with the service level. It was clear that alternative architectures were necessary which would improve the possibility of getting reliable reporting on the device capabilities and in reducing the time necessary to collect the elements of the context both of which are preconditions for the subsequent evaluation and subsequent adaptation phases.

In the next chapter, a new strategy and architectural approach for context adaptation systems will be presented which is based on the findings and on general computing systems trends. This approach will take the observations on operation and performance coupled with the emerging trends in distributed computing to formulate a more computationally intelligent and flexible model for engineering context adaptation.

Chapter 4 Knowledge-Base Systems for Device Context

There is a significant number of possible solutions to most engineering challenges with many of those solutions following familiar patterns. Patterns, particularly time tested patterns, can aid in the engineering of a solution without the need for trial and error experimentation [141]. When faced with a systems performance issue of the type identified in Chapter 3, there are generally two classes of action that can be undertaken. The first is an optimisation process which seeks to identify and streamline the software systems as it is implemented. The second option is to re-architect the components of a given system, substituting in modules or components which improve response time or reduce bottlenecks. The choice of which approach to take often involves other decision variables such as the extent to which code can be optimised to achieve the performance goal, availability of source code for all the modules in question for optimisation and the possible other benefits that a developer might "inherit" if an alternate design is chosen. These are weighed against the effort in rework and the skill gap that might also be inherent in the selection of a new architectural design for a system. In many cases, there is a global benefit beyond the singular increase in performance that can be realised from a local code optimisation exercise. These global benefits can be difficult to see when dealing at the program level, but through an understanding of the broader process within which the implemented system operates; one can improve the overall architectural style as well as improve the performance to meet the goal. From the results of the context adaptation experimentation outlined in Chapter 3, an opportunity was provided to create new understanding and possibly innovative technology to provide a foundation for moving the prototype concept into a position of being commercially viable. The Kodak European strategic research direction supported the decision to investigate the use of intelligent systems techniques and constructs and specifically, knowledge-based systems architectures to address the issues uncovered with context adaptation. This decision was an important one, as it determined not only the domain of the solution but also the design and implementation of the prototype. A key reason for

the investigation into the knowledge-based system for context was the desire to have the new architecture act rationally and intelligently to evolving conditions, a requirement which implied the use of some form of automate computational intelligence. This decision also followed directly from the industrial research portfolio directives discussed in Chapter 1.

It has already been established that the mobile device marketplace and mobile application technology segments are extremely dynamic and these devices change features and functions very quickly. The implication of this observation is that a traditional business systems engineering efforts do not and have not necessarily resulted in a core knowledge structure which can evolve readily, but that new methods for dealing with rapidly evolving contextual components were critical success factors for future mobile context adaptation services. The general needs for the future context adaptation system are summarized superbly by Rodney Brooks in his short text "Cambrian Intelligence". In discussing the desired path for evolution of artificial intelligence, he outlines needs for future new creatures, or software modules. The creatures must do the following [142]:

- They must cope well with dynamic change
- They must be "robust" to changes in the world , meaning in the context adaptation model, that the service should continue to operate even in the face of dynamic context changes
- They must be able to work towards several possible and equally rational goals and it has an important and very relevant attribute – that "adapts" to its to surroundings and take advantage of opportunities that might be arise serendipitously
- They must achieve something in the domain of interest

Knowledge-based systems offer the promise of automated extensibility through general conceptualisation of the architectural solution space coupled with automation of

learning about the new features of the world. The context adaptation subsystem also showed that not all aspects of the knowledge were equally dynamic but in fact there were elements that were reasonably static at least within the time frames of the service request session. The use of artificial intelligence and automated knowledge learning to achieve these requirements means a new approach to context adaptation and to the architectural design of the C2Adapt. The evolving nature of mobile device attribute driven context is apparent at all levels of a system's architecture. It starts with new functionality for changing low level characteristics after manufacturing. At the next higher level in terms of frequency, is the OS and firmware level where changes are made to device functionality already deployed in the field. At an even greater level of dynamic change is that found in the future reconfigurable devices which are the topic of Chapter 6. Given the dynamic nature of change in these systems, the alternative architectural design was meant to accommodate the dynamic nature in an intelligent manner by modelling the device as an object whose aspects, attributes are inherent knowledge related to the object. The ability to implement the context adaptation in a way which incorporates the evolution and object dynamism is believed to be an intelligent approach in the manner consistent with [54].

With the notion of a new intelligent repository for device context, it is important to review the keys aspects of knowledge-bases themselves. This study investigates the practical use of KB's in a service engineering context. As such, existing technology will need to be understood, evaluated and utilised. A knowledge-base represents a physical storage repository which is commonly the output of a knowledge engineering process [54]. Well designed knowledge-based systems allow the continual collection and implementation of a concepts and relations such that these may be used for various purposes. One of the principle purposes for such repositories is to allow some form of logic to be applied against the contents of the knowledge-bases such that one can make a decision and/or further enhance the amount of knowledge maintained relative to a specific instance, event or concept. A knowledge-base is often used as a repository of conceptual information drawn from a pre-defined ontology and is the central element of many AI systems. One of the key differences between a knowledge-base and its near

cousin, the relational database is in its usage and extensibility. The structure of a knowledge-base is designed to allow the acquisition of new knowledge and to extend the repository automatically to support this newly acquired knowledge. In the relational database structure, new knowledge must fit into the existing data structure as previously designed and all previously determined knowledge is then appended with the new knowledge. Knowledge-bases are designed to extend so that newly acquired facts are accommodated with it structural renovation. This fact is the reason that knowledge-base designs are preferred for systems engineering efforts which require knowledge acquisition and learning.

The use of a knowledge-base centric system requires the selection and utilization of the relevant KB development methodology which differs slightly from that of the traditional systems development methodology. It is important to understand that the process is not the same between the two sub disciplines of computer engineering and that there are subtle yet significant differences which affect the overall outcome [65]. A study of several methodologies for knowledge-base systems development reveals that in general, an iterative process is most effective. It is also clear the development of a seed ontology is preferred as the engineer can learn more about the domain and incorporate that learning into the system iteratively. A seed ontology is, as its name implies, a knowledge representation which acts as a starter for future iterations. The seed ontology usually presents the outwardly obvious information and relations within the domain. Analysis and further investigation of the real world setting of the system act to refine the seed ontology further, expanding and correcting additional facts which in turn provide additional knowledge to the overall system. The methodology for the development and implementation of the mobile device knowledge-base, or MD-KB as it will be called hereafter, is outlined as follows:

Phase 1 - Feasibility – study of the domain, the relevant knowledge in the public domain and determination of the nature of the knowledge-base extent

Phase 2 - Requirements and Competency Questions – use case scenario development and development of the key competency questions which will form the summary level requirements

Phase 3 - Tool Selection – determination of the best tool for the development of the KB development technology available

Phase 4 - Design and Refinement – knowledge engineering in an iterative fashion

Phase 5 - Implementation – transformation from the model of the knowledge domain as engineered into a usable and accessible set of axioms

Phase 6 - Evaluation and Validation – formalized validation of the knowledge-base through evaluation and analysis

Phase 7 – Maintenance and Operation – the proposed mechanism for keeping the knowledge-base current and adding new knowledge over time

This methodology is adapted from the work of the On-To-Knowledge (OKT) project and is found published as a set of deliverables from the group from University of Karlsruhe [65]. These steps in the process are described briefly to set the stage and then the details of the MD-KB are outlined further in this section. A first review and detailed study of the conceptual elements of the system domain of the specific problem is necessary to form an ontology. This ontology provides the basis for the concepts and the relations between the concepts themselves. In the MD-KB, the concepts of the specific domain in question are the components of the device and the situational context classes, for example. The purpose of the first feasibility phase is to supply the background domain information for the project. This includes research on the nature of knowledge representation, ontology creation, and a determination of applicability from the current state of the art in knowledge engineering.

In a computing system implementing machine intelligence, an evolving knowledge repository is not the only required element. There is also the need to be able to use the evolving knowledge in a predictable and rational way. The choice of a core ontology for

the MD-KB is in anticipation of using it as input to inference of context for the purpose of performing the actions necessary for context adaptation. The central question is whether the rules which drive the context adaptation are logical and consistent and whether the delivery of the services adapted based on these rules will provide a higher quality representation of the desired content is the central point of this chapter's work. It remains to be shown if this question can be accurately answered. These philosophical questions are less easily answered then the basic question of whether or not a knowledge-based system can provide the basic facts for the adaptation. But at a deeper level, it is worth noting some foundational theorems in cognitive science which will help to understand the possible limitation on the results. As Jacob Bronowski discusses in his classic text "The Origins of Knowledge and Imagination", *Kurt Gödel determined universally that if you have an axiomatic system (or in this case a system of rules and facts), and that system is consistent in meaning so it can infer that certain implications are in fact true, then your system will have some statements that it cannot determine the truth of and in fact, there will be conditions which are true that the system will not be able to determine.* This may at first appear to imply defeat in goal of providing an accurate and usable system. However, the Gödel assertion sets limits on a **global** axiomatic knowledge system. The MD-KB and its associated prototype will be shown to be constrained already to a "micro-world" or a narrowly defined subset of the global knowledge which will serve to increase its likelihood of accuracy. While an understanding of this limitation imposed on the performance of our device context knowledge system is important, the extent to which this affects the overall solution effectiveness is assumed to be lower than the improvement obtained by the use of the structured knowledge in the solution. It is important to observe this fundamental principle however, so as to set the appropriate expectation level. Simply put, there will always be a set of conditions, of context variables in this case, for which a knowledge-base will not be able to assert viable actions. The goal then is to design and develop a system which minimizes the potential of this condition.

4.1 Requirements for Mobile Device Knowledge-base System (MD-KB)

The summary level requirements for the MD-KB are straightforward and primarily based on the results of the previous phase of the project as outlined in Chapter 3, where performance bottlenecks and accuracy of contextual variables hampered the utility of the prototype system. The table below represents a prioritised list of requirements and those associated metrics for each at this summary level:

Req_Id	Category	Description	Priority	Metric
1	Core	Allow the input/edit of ontology – class, instances, etc.	H	Task completion
2	Core	Allow hierarchical model of conceptual domain	H	A suitable ontology is created which models the knowledge through validation
3	Acquisition	Provide the means to automate acquisition on new knowledge from public sources	L	Using knowledge extraction techniques, is it possible to take manufacturing specifications directly into the system

4	Constraint	Allow the implementation of constraints on the knowledge	M	Create meaningful rule using an abstract rule engine integrated
5	Constraint	Provides a means for the validation of the model	H	Provide a set of tools which can be used to positively validate the model
6	Core	Provide the means to implement new features of devices without re-structuring	H	Add features to existing devices or whole new subassemblies
7	Access	Provide a means to access the instances of the knowledge-base programmatically	H	Specification of API for query engine
8	Core	Provide a suitable framework to represent the knowledge in the case of situated context	H	Proper model for context
9	Validation	Allow the use of visualisation tools to assess the KB	M	Demonstrate the visualisation tool output
10	Validation	Allow access to structure of	M	

		KB to assess quality of classes, instances in textual format		
11	Standard	System must be capable of being accessed from third party tools	H	Utilize 3rd party plug-in
12	Standard	The resulting knowledge-base must adhere to the open knowledge-base connectivity standards	H	Validate through OKBC standard
13	Performance	Allow sub second retrieval of instance data	L	Test extraction of context variables though query tool
14	Capacity	Provide managed storage for mobile device instances in RDF-S	L	None

Table 4-1: Knowledge-base Requirements

The requirements were derived from the market objectives of the NEU context adaptation system and coupled with the intelligent systems architectural vision provide the basic inputs for tool selection and act as a guide for the prototype design process. This is not an exhaustive list and does not represent the requirements for overall context adaptation services as it does not include all the auxiliary functionality necessary to create a viable public web service but not necessary to answer the research question here.

The next phase for MD-KB is the selection of the appropriate tools and knowledge representation for the system itself. Unlike traditional systems development, knowledge systems must encode the information in a representation which will adequately support the processes involved. There are many ways to encode domain knowledge although two major schools can be identified in the literature. Those are knowledge-base formulations based on predicate calculus and those that are based on a taxonomy of knowledge often times referred to as knowledge networks [53]. Predicate calculus represents knowledge as a set of facts which can be true or false. Networks represent knowledge in a hierarchy of facts all originating from one central object called "THING". A variant of the network is the frame which extends the network node concept to include an internal association of information with slots and facets which further describe the class which is the substantive component of the frame [53]. The benefit of the frame is that it was developed in order to represent the situation of a central object or environment so it handles the meta-aspects or meta-information associated with the primary object of interest very well [74]. This latter representation precisely matched the central goals of the alternative architecture as it represents the device and its associated context potential as the central object whose meta-aspects change through time.

4.2 MD-KB Design and Competency Questions

In order to deliver to the requirements, the device context knowledge-base must be able to evolve as the devices themselves change in the field. The KB must support then the ability to provide a feasible configuration answer to questions posed in the form of service requests. With the best representation approach for the MD-KB selected, the design stage of the MD-KB implementation was broken into two components of work, an upper system architectural level and a lower knowledge-base specific implementation design level. On the higher level, there is the design of the overall KB-based architectural alternative which will be discussed in greater detail in the following section. At the lower level is the design of the knowledge-base itself, which sits at the core of the new context adaptation system. In design of context-based systems, context as mentioned above has been defined by several researchers across several different domains. In fact, it is clear that context in software engineering systems follows the

many of the same rules as ontology development, in that it is only meaningful as it relates to the specific domain of interest or application. Therefore, to speak of context-aware systems is to specify that a system is aware of a set of domain-specific entities about which adaptation for improved quality of service may occur. To link the context definition to some idea of feature specification requires that we define either endpoint and draw the relationship or transformation from one endpoint to the other. [17], a work which is widely cited in this area, defines the context as "any information that can be used to characterise the situation of an entity". This is quite a broad statement about the nature of context itself although the use of the term "situation" is very important and represents a statement of the approach taken in this work. The work described in [22] takes a more focused approach on the definition of context which is essentially the information provided by the network itself as a means to adapt the application in a mobile peer-to-peer system. The disparity between the approaches is clear; there are several interpretations of context and context adaptation. Engineering contextual adaptation can be both a conceptual approach at the highest level or a specific capability within a system at a lower level of operation, or in multiple levels within the architecture.

Further thinking along these lines is instructive as it help to form a clear picture of the both the detailed KB design set within the broader context adaptation architecture. Another approach to context architectures mirrors the work done in this project. The work reported in [28] investigated the use of context within mobile systems from a human computer interaction perspective. Most interesting to note is that the results of this work and the segmentation of context although significantly different in some respects, aligns with the taxonomic treatment in the present work [28]. In the paper "Exploiting context in HCI design for Mobile Systems", they segment their context into **infrastructure, application, system, and location**. Each of these segments play a different role in the context state and hence can play a different role in the adaptation to suit the needs of the user. In a manner similar to the work outlined in this thesis, the device within its environment is the central theme and forms the linkage between the physical and "virtual" space [28]. The model of this world will then reflect this central

position by placing the central or backbone taxonomy around the nature of the device itself. The linkage between the physical and virtual world provides the first insight to the appropriate treatment of the next step in the process of context and feature mapping procedure. Just as devices are clearly real or physical artefacts in this case, so are the elements of context, virtual or conceptual entities subject to the domain of relevance. By the same rules of knowledge engineering then it is possible to use the process of knowledge engineering to resolve these physical features into virtual context in much the same way as the knowledge engineer takes physical entities in the world and resolves them into virtual concepts within a knowledge-base. In this section, the focus will be to isolate specific elements of these architectures which will form part of a new intelligent systems context adaptation solution, one which will improve performance and help the scalability of such systems.

To re-iterate, there are several architectural design patterns which have emerged from the literature and are relevant to the knowledge-centric remodelling effort. The context widgets described [143] are a classic example of one architectural approach. These widget-based systems are an example of a multi-component, modularised context architecture. The author of [144] took a different architectural approach from widgets, not necessarily predicated on the need to improve performance but for other reasons, when he introduced the blackboard style of systems architecture as a mechanism to maintain context information and to efficiently recognize the changes in the context. Another architecture called QoSDream developed in [21] utilises a multi-tiered, event-based message passing architecture to implement context adaptation. The approach outlined in this implementation appears to be similar to the NEU architecture which was described in detail in Chapter 3. The QoSDream system uses a series of application programming interfaces (APIs) which allow applications to subscribe to the relevant context services and then to receive notification when an event of interest occurs. These same APIs are also used by an application to broadcast out contextual or other programmatic events that other services or applications may be interested in. There are two higher level abstractions within this architecture which represent the context collection. The first is a sensor subsystem or device register with a context aggregator in

QoSDream, where a module called the "Federator" is responsible for aggregating and creating events to be passed into the centralized system. The federator creates a fairly lightweight event message indicating the id, the type of event, as in an add or change to context and some specific details about the context change itself. The complete details of the QoSDream architecture and system are not covered here, but can be found in [21]. The observation made from the literature regarding context-aware architectures is the need for some form of persistent storage to hold domain specific elements of the context as in the case of the coordinates for a mapping system in a location-aware application. This persistent storage is also used to hold other forms of contextual information such as content attributes and user preferences, or generally other elements of the context 4-tuple. The MD-KB will represent a contextual information store as a persistent, yet evolvable knowledge store.

The contribution of this part of the overall programme is in the use of existing architectural patterns around persistent storage of context in very different ways, to use this pattern as a means to improve performance and scalability, not as a temporary storage location or database management system for the varying element of content. The novel concept here is to partition the context in a temporally static and dynamic context and to utilise persistent storage with machine learning for the static context. As previously mentioned, the nature of adaptation in context-aware systems is that of change, change to device capabilities, location, preferences, environment and more. Adapting to change in all these areas is not straightforward in a QoS constrained environment as has been already shown, but the greater the context adaptation, the greater the value and the greater the revenue potential. This makes the nature of this work around an innovative alternate architectural style well worth pursuing. This complex issue creates the need to balance all these factors in order to provide the optimal context adaptation within the QoS constraints. If we assume that the 4-tuple of context is fixed by service requirements in terms of the numbers of variables needed to properly adapt, and also assume there exists a specific and fixed time frame in which one can perform the adaptation and return the content. One of the parameters left to tune is the architectural design of the system itself and seek to increase or optimise components to

improve the QoS. By partitioning the context into static variables which is stored efficiently in a structured knowledge-base the time to retrieve that context will be improved over and above if all the same context had to derived, inferred or dynamically queried. Conceptually, this hybrid architecture with the MD-KB at its core, is proposed to store what is mostly likely to be session static and only discover dynamically that which is actually time varying in the lifetime of the service itself. The benefits of this style are accuracy, reduced overhead in transmission and improved performance.

To integrate into this architecture, components of the context 4-tuple must be analysed and the temporal nature of the attributes determined. It is obvious that some elements of context are suited to persistent storage as they have suitably long life times over which their value is still relevant. It is equally clear that other elements of context are very dynamic in nature and may need to be determined as close to the request and response as possible.

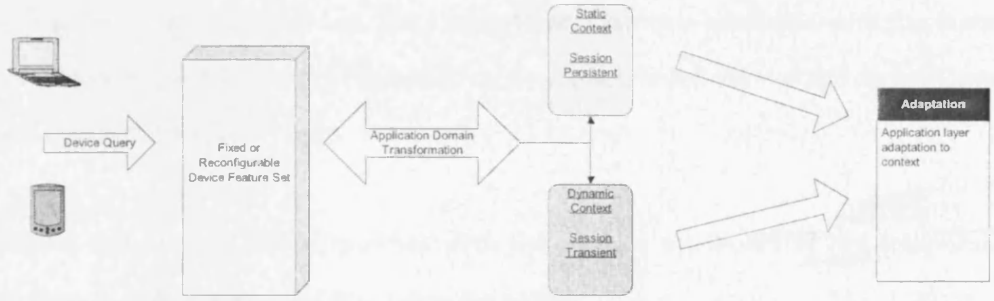

Figure 4-1: Static and Dynamic context in new architecture adaptive systems

Given the expansive number of contextual variables, it is impossible to analyze all the context partitioning scenarios in a work such as this. Ideally, the truly intelligent system will learn which attributes are invariant over relevant timeframes and this would not be part of the system design. Unfortunately due to resources constraints, this possibility could not be investigated rather the method here is to analyze the component of the context 4-tuple which has the greatest effect on the outcome is taken from empirical work with the context-adaptive architectures for imaging. A key conclusion from all work on context adaptation done in the laboratory is that device related contextual

attributes above all else contributed the most to adaptation of content. Specifically for imaging applications obvious attributes such as display size and display type contributed significantly, but equally as important for context adaptation for visual media are the more difficult attributes like palette type, brightness level, contrast and so on. These elements are not reported in user agent headers and are not consistently provided to the browser in a HTTP session. Taking these three as an example, palette type is an inherent attribute of the device and will not change from session to session and certainly not within the time frame of a request cycle. This is denoted a static attribute of the device. The other attributes which are mentioned, brightness level and contrast, while are inherent to the device, their actual setting can be quite transient and changing depending on the lighting in the environment. These are dynamic attributes of context. Static attributes can be made available to the system via a database management system or through an online Internet database where a query is run for all static components or specific components that are needed. The dynamic attributes can be acquired from the device which may be over a slower transmission line. By partitioning the context in this way, the longer term context is available over the fastest medium and only the most critical dynamic context is retrieved via the slower and more complex dynamic discovery route.

The MD-KB architecture design process with the desired attributes of the temporally variable device context provided the following model:

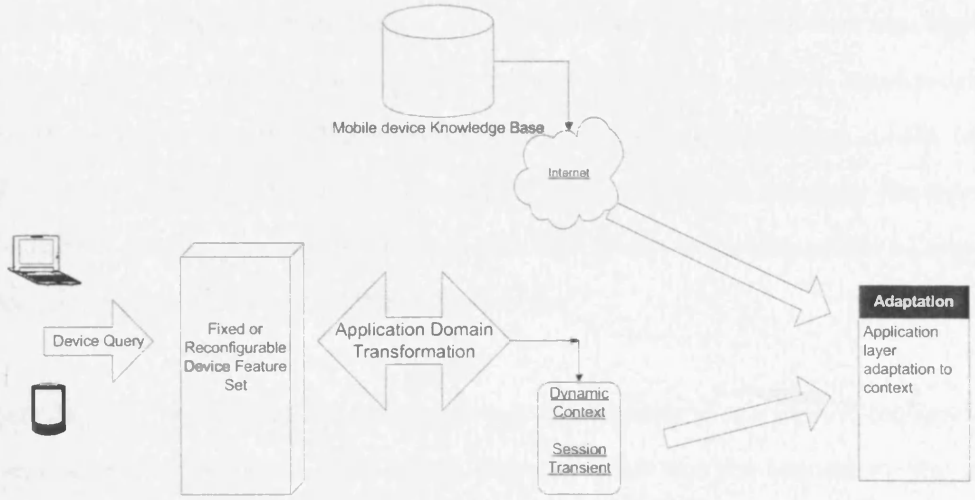

Figure 4-2: Alternate Architecture for Temporally Variable Context

The system as designed calls for the replacement of the device discovery or feature extraction step with the combination of a knowledge-base for those static components of the device related context coupled with a shortened dynamic discovery header which in total reduces the amount of information to be acquired dynamically from the device itself. Another positive aspect of this architectural design is in its potential for caching of the static information. This would further increase the performance of the system as the variables could be held in a local server closer to the edge of the network itself. Caching and in particular, caching proxy architectures are prominent in the active network architecture as mentioned in the previous chapter. This architectural component is particularly important in context-aware systems as it provides a performance improvement by holding important and frequently referencing information closer to the potential user. Re-use and reduction of network hops improves the performance through reducing network latency. Given that properly implemented and tuned caching in a network improves the service performance, the architecture is then designed so that it will support the caching of the elements from context 4-tuple. The cache is then populated in two ways, one for the static elements as specified for the service request and one for the dynamic information as acquired by the system from the source itself. As described, this is a different variant of context aggregation, a common topic of research in networking performance. This version is one which allows session relevant context to converge with dynamically acquired device context in a manner quite

consistent with that found in [145]. The knowledge-base centric services approach differs slightly in that it follows the design philosophy of the service-oriented architectures rather than the object-based component architectures found in [23]. Special focus is on the nature and extent of the device context elements; however the extension to the further 4-tuple elements is foreseen although future work is required to assess the performance degradation as the elements increase.

The concept of "session" in this architecture is the lifetime of the service request where the service request is atomic. The atomic nature implies that the request for the service may incur several sub services or other activities, but the overall service is not considered complete until all these supporting activities have reported their result. The dynamic attributes in this model are those attributes that may change during the session timeframe. Conversely, there exists a set of attributes which are guaranteed, or at least have a high probability of remaining the same during the session timeframe. These categories coexist in the execution period of a service request. In Figure 4-4, the architecture for an alternative approach to context adaptation takes the dynamic context from the device which would effectively self-report specific short lived variables and merges these through the context proxy with the static variable retrieved through the knowledge-base in an approach designed to improve accuracy and reduce time of variable collection. Given this systematic approach, populating the caching proxy would follow the schematic in the figure below.

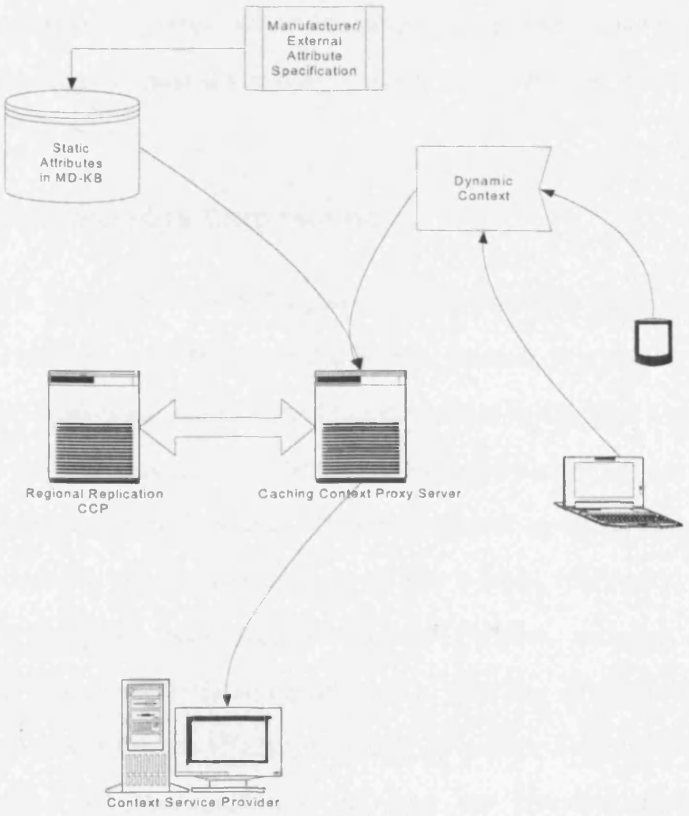

Figure 4-3: High Level Context Caching Proxy Architecture

With the architectural design described at a high level, it is next important to look into the nature of the knowledge-base itself, and to design the ontology which will support the creation of the core knowledge-base. This work requires study and analysis in order to properly select the appropriate knowledge representation framework. All these tasks in this next step are traditionally referred to as "knowledge engineering". It is worth noting that a hybrid architecture such as the one proposed requires processes of traditional software engineering in terms of design to be merged with those in the knowledge engineering world. This is all then specified using web services engineering techniques which again has a subtlety different approach to design. One of the key challenges with providing an engineered solutions which touches all these domains is in understanding and rationalizing the key process steps. For example, in the development of the service itself, there are traditionally built objects and components which each need to be developed and then integrated with the knowledge-base which is also developed with its own set of processes. The result is that additional time is necessary for the

process integration required when spanning disciplines such as these, so the decision to use the technologies should include thought and planning for this additional work and time.

4.3 MD-KB Knowledge Engineering

To move this work forward in its goal towards enabling contextual adaptation through open and available knowledge of the device context, two concepts must be linked in a robust way. First, the context 4-tuple as described in Chapter 2 and further discussed in Chapter 3, depicts four independent elements of which there are 1 to *N* sub-elements of the actual context variable itself. This component of the programme of study focuses exclusively on the device component of the 4-tuple hierarchy and its numerous sub elements; only lightly touching on the user profiles, network attributes and content adaptation. It has been shown empirically, that the device-related context represents those attributes which can affect the quality of service delivered to the service requester. There exists a bi-directional link between the physical device and its contextual adaptation potential. This link is critical to the development of adaptive systems as a standard descriptive language is necessary for consistency in coding even within development teams themselves. Simple problems such as whether the output of the application is sent to the "display" or to the "screen" can cause errors in coding and confusion in on-going maintenance of such system. A standardized language of device features is also a necessary pre-requisite for open systems. This need for a standard description and semantics for device context is not new and much work has been undertaken through the IETF and others to attempt to achieve standard contextual descriptions. While the descriptions and the definitions have resulted from these efforts, the validation and implementation of the work is left to the application or service developer who then interprets these in a very non-standard and often proprietary way, creating new databases of features which can only be used for one set of closed services. These schemas provide the framework in many cases, but they lack the ability to govern how the definition and descriptions evolve over time.

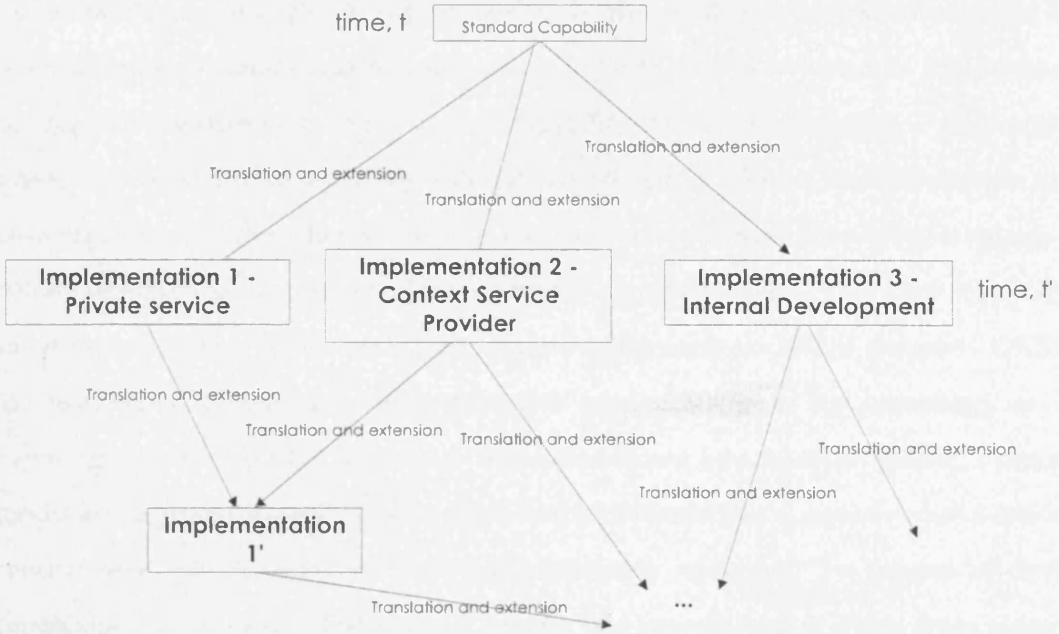

Figure 4-4: Free Form Propagation of Systems from Open, Standards Definitions

Using a well-defined, and flexible yet constrained KB reduces the rampant expansionism of possible context descriptions by providing a means to gather the intrinsic attributes and capabilities into one common semantic source. While service deployments will generate new ways to use the context, they will not generate new definitions and descriptions of the context itself. The design of a standardised language of devices features is synonymous in many respects with the development of an ontology. The key question is then can the standards development process for device specific context be used as the basis for the ontology development process and ultimately be converted to usable knowledge-base? As previously mentioned, there are several standards such as CC/PP, UAPROF which are in development to help with the descriptive nature of the device attributes as a variable of context. Analysis of these standards indicates that there is some subset of the context information contained in the user agent header [146]. However, the list of attributes and features which are outlined within emerging standard is relatively low and does not include any provision for future reconfigurability of devices, a topic which will be addressed in Chapter 6. The standards indicate the features and the possible values but as one would expect they cannot provide the resulting contextual variables that an engineer might possibly call on in order to adapt

the content or the application appropriately. Further analysis of the standards and the possible implementation which would result indicate that as written, the standards do not support the determination of service capability, but device capability. Although a subtle detail, the device capability will indicate the set of features from which one may infer capability but they do not have a mechanism or a standard descriptive language to denote whether that device capability can match a specific service. There are yet another set of description services-based standards emerging, such as OWL-S (formerly DAML-S). In OWL-S, the web services metaphor is used as the basis for automated service handling which breaks the functional description down into a service profile, a process model and a grounding [67]. As its origin would indicate, this approach takes a service-centric view, but its early versions had previously addressed the impact of device capabilities on services. The current version has moved further away from issue of device and now takes the approach of specifying the needs of the service although the treatment of device capabilities is at this time quite limited. Parameters and information external to the device itself also are not covered completely in the standards around capability and feature expression. With the exception of those standards focusing on mobile content delivery, most standards work is focused on the intrinsic nature of the content itself as in the MPEG standards or in the details of the software within the services as in the web services standards process, but not necessarily on guarantee of the quality of the delivery of both. This is an important and relevant gap that needs to be addressed before service engineering can build up momentum in deployment.

The basis of this chapter though, is on the use of the appropriate standards derived device descriptions as the foundation for the design and development of the mobile device knowledge-base. In the knowledge engineering process, it is important to consider how extensible the knowledge-base will be in order to accommodate new information and other associated information. For example, in the device context area alone, there are sufficient challenges in modelling the proper structure for basic device features and a minimal set of contextual categories. However, as was discussed earlier, the context 4-tuple that has been defined includes three other major context elements beyond device, namely network, user preference and content attributes. Another level

of adaptation to be considered in the design and one which may not be suited for inclusion in the MD-KB, is that of the context related to the real-time network performance. Many researchers conclude that network characteristics such as delay can be determined and stored as context. While this is certainly possible, it is an optimistic assumption given the bursty nature of Internet traffic; it is particularly optimistic in the mobile device domain where these devices can lose a signal entirely without warning. The NEU context adaptation system prototype demonstrated that using network probes as the requests are made gave a reasonable indication of traffic on those requests which required network related context adaptation. There is inherent in contextual variables the notion of "time to live" or the amount of time after which the value of the variable is no longer valuable and may in fact be quite misleading if used for adaptation. This is specifically true of the network elements of the context 4-tuple. The issue with this in the design of the knowledge-base is that these individual facts lead quickly to an unmanageable set of requirements. Therefore, those variables which contribute the greatest to the service value were to be included in the MD-KB, otherwise a design by fact approach leads to a lack of clarity in conceptualisation as the individual facts may be in conflict or may be applicable only in one specific instantiation of the domain. One of key design criteria for an ontology, and hence its resulting KB, is that in order that it be suitable for reuse amongst several different users, the ontology must be clear and coherent, meaning the concepts of objects must be complete and largely unambiguous [60]. In this first prototype of knowledge-based web service for context adaptation, it was important to focus on one contextual area to prove the concept rather than dilute the architecture by trying to address all context variables and then not proving even the base concept. To ensure a minimal level of coherence in knowledge, the MD-KB's basic ontology will use the integrated knowledge provided by the standards formation process. Using the expert opinion, yet heterogeneous knowledge represented by a number of different standards groups, the ontology will represent knowledge from several different participants in several different industries. The ontology will then include knowledge of features and attributes provided by a diversity of sources and through that diversity have better clarity than through a single knowledge expert [58].

At this stage, the design can be validated to some degree using a set of basic competency questions. The competency questions, or CQs, are those set of questions extracted during interviews with experts from the specific knowledge domain of the system [55]. As mentioned, the questions are those that the knowledge-base must be capable of answering once it is constructed. These questions can also form a foundation for the requirements of the knowledge-base and can imply the structure of the ontology underlying the knowledge-base. There is an important distinction to make at this point in the knowledge engineering process about the difference in meaning between the terms ontology and knowledge-base within this chapter. While often used interchangeably, the correct interpretation of these two in the scope of this thesis is best expressed by Thomas Gruber in his excellent paper entitled *"Toward Principles for the Design of Ontologies Used for Knowledge Sharing"* wherein he outlines an engineering approach to the design and implementation of ontologies for application needs. Gruber's definition is "A shared ontology need only describe a vocabulary for talking about a domain, whereas a knowledge-base may include the knowledge needed to solve a problem or answer arbitrary queries about a domain" [60]. It is evident from this statement that the MD-KB is the knowledge-base implementation resulting in part from the ontology derived from the standards for device management and context. A subset of the competency questions posed to the MD-KB can be found in the table below. These questions will be used again in a subsequent section as part of the formulation of the validating queries.

Question	Competency Questions
1	What device will support imaging applications?
2	What device will support the distribution of a new format of media?
3	What image will be feasible on a Nokia™ 6310i?
4	What device allows storage?
5	What device will allow a dynamic discovery program to be installed?

6	What device supports local wireless connectivity?
7	Will a location-aware application work on this device?

Figure 4-5: Sample MD-KB Competency Questions

The final component of the knowledge engineering process involves the critical decision around the knowledge representation framework. In this section, the primary concern is which knowledge representation framework suits the knowledge domain. To answer the question of representation from an engineering perspective is a daunting task, however. The resources, the history and depth of the field of artificial intelligence are built on this basic question of how to accurately and completely represent knowledge. The issue is then how to distil this vast quantity of information, opinion and experiment into a manageable set of facts from which an engineer can decide on a path forward. Simply put, a knowledge representation scheme can generally fall into either predicate logic or situated knowledge. The ***predicate logic*** category follows the "good old fashioned" AI path which constitutes a series of facts which ultimately can be combined and evaluated in a mathematical manner in order to represent a state of knowledge. The ***situated knowledge*** category is a later invention where knowledge can be represented as set of interconnected facts all of which combine to make a consistent view of a given environment or situation [73]. The decision taken for the MD-KB is to use a situated knowledge approach and not one of predicate logic as the CQs and the nature of the contextual adaptation itself is intrinsically about a situation or point in time of a service request, a complex situation which would take a formal logic system an inordinate amount of fundamental facts. Using the connectedness style of AI for the knowledge representation scheme brings about additional choices for the engineer. Since the MD-KB is designed as the core of an operating application it will need to provide some means of reasoning. With the knowledge representation scheme following the connected node or semantic network style, the choice of a reasoning environment was primarily between frame systems, semantic networks, or description logics. Given the system domain for MD-KB, a review of the fundamental details of three representation choices led to the selection of the frame system approach. In modelling which

representation style would be appropriate for the context adaptation environment, the frame theory developed by Minsky fit very well with the known and observed operating parameters of the mobile device world. By definition, *"a frame is a collection of questions to be asked about a hypothetical situation; it specifies issues to be raised and methods to be used in dealing with them"* [74]. This fits the environment of context adaptation where the service request reflects the needs of a given situation in time and the contextual needs of the service request require specific actions and methods to be taken in order to properly address the request. The analysis of this statement finds that contextual adaptation in services can be broken into a series of questions – What is the nature of the service? What are the constraints that the customer has subscribed to? What is the known context? What is the needed context that come from user? The answers to these questions lead down a hierarchical pathway of nodes to a series of appropriate actions. So, one situation in time, or service request, resolves into many questions that create each a series of actions. This corresponds to the knowledge framework of a frame system nicely and as such was the best candidate for the representation scheme for the MD-KB.

4.4 Development Tool Selection

With the basic foundation of the new model outlined, it was necessary to turn the focus towards the reasoning functions of the system. In intelligent systems development, the process of modelling the reasoning process within the human thought cycle is critical to the success of any knowledge-based system. As a matter of fact, the reasoning process within many AI systems is in itself a vastly complicated subsystem. A considerable volume of research work into automated reasoning has occurred over the last eighty years starting with a particular focus on symbolic logic. While there is debate about whether this focus on symbolic logic was in fact to the detriment of the field itself , there is no doubt that the dominant theme in reasoning systems is based on the symbolic logic work. Automated reasoning systems are the users of the machine understandable knowledge maintained within the knowledge-base, reasoning systems provide the mechanism for the formulation of an appropriate decisions which leads to some form of negative or positive action. Therefore, the nature of evolving knowledge-bases require some form of reasoning system in order to acquire and classify new knowledge

appropriately. The oldest of these systems are the theorem provers which seek to take in a mathematical thesis and either prove or disprove it through some form of intelligent plan or resolution technique. Theorem provers were the first of these systems which were designed to answer a question which is posed in some logic-based format. Other tools such as artificial intelligence programming languages such as Prolog provided the means for programmers to compose programs which would interrogate a knowledge-base, use the facts that are retrieved and combining this with highly constrained logic language constructs, they could perform some inference to derive possible solutions[144]. These solutions can then be added back into the knowledge-base allowing the system to acquire new knowledge in an iterative fashion. Three more modern types of reasoning systems exist and are relevant to the device capability knowledge system which is the topic of this chapter. Production systems, frame systems and description logic systems are variants of systems all concerned with the retrieval of structured information from a knowledge-base in order to achieve some pre-defined condition. It is instructive to provide a brief review of these knowledge framework systems and their differences as outlined below:

PRODUCTION SYSTEMS

A production system is composed of a set or list of rules which are called the production rules. These rules are commonly written as a condition, c and the subsequent action, a. In most production type systems, conditions are continually monitored through the use of sensors. The resulting actions are either direct actions, calling other programs or can even be calling other production systems to execute further conditions and actions [53]. In the context of the knowledge-base systems, production systems will be that part of the knowledge-base that maintains the rules which can be applied against the concepts within the ontology. Some production systems contain knowledge-base repository capabilities, while other rely on semantic and most often, manual integration with external knowledge-bases.

FRAME SYSTEMS

A frame based system defines concepts at the nodes or junctions of graph-like structures where the concepts are defined in detail through the use of taxonomy. Semantic networks are the cousin of the frame-based systems, but instead of being a nesting of concepts in a taxonomic representation, the semantic network is a series of arrow connected nodes. The frame itself is the core of the architecture in a frame-based system and it is usually a full representation of the state of a given space at a give point in the entire problem space in question. The dimensions in this space are defined relative to the problem, so in some cases it may be a point in the life cycle of an object or may be an intangible attribute of some central entity.

DESCRIPTION LOGIC

Description logic systems are a variant on the semantic network where the more formal symbolic definition of the knowledge is represented and the structure still maintains its taxonomy. The description logic representational system combines the nodal representation of the semantic network with the description in formal terms of the concept themselves.

With this background and further analysis, a selection of technology tools was performed and the selection of the Protégé 2000 ontology editor was completed. Protégé 2000 is a software tool developed by Stanford University in its department of Medical Informatics [147]. There is extensive support and ongoing development work on this software tool which enables easy experimentation while providing a measure of technical robustness. Protégé 2000 is implemented using Java and the current version 1.9 supports a variety of third party plug-ins to enhance the ontology editing capability of the core system. The editor itself can be extended through the use of plug-in applications which are contributed both from the original team of developers and the community of Protégé users. Protégé also provides a set of visualisation tools which will be used in the evaluation and validation phase of the KB development.

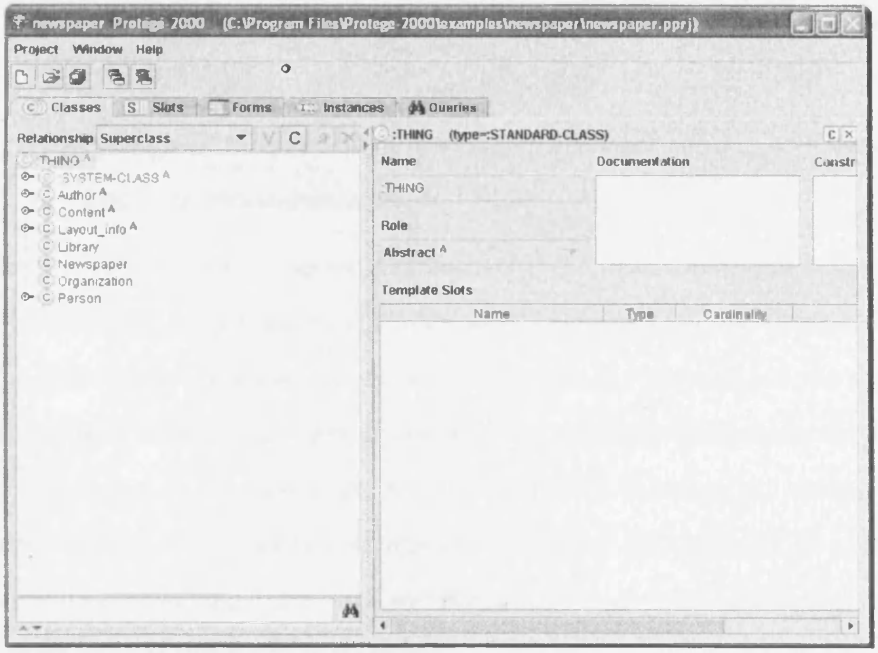

Figure 4-6: The user interface and sample project in Protégé 2000 v1.9

Figure 4-7 shows the Protégé 2000 primary ontology editing interface depicted here with the sample Newspaper project, the tab metaphor gives the developer access to the components of the frame based system – the classes, slots and instances as well as the supporting tools for the use of input forms and composing queries. In addition to the basic ontology building toolkit, there is a second set of tools necessary to properly complete the knowledge-base development process. A set of visualisation tools is necessary to perform the evaluation and validation phase, a topic which will be discussed in detail in a later section. These tools integrate into the ontology editor so as to provide native support for the iterative refinement process. In knowledge engineering, it is not sufficient to design the ontology and rules without validating the utility of the resulting KB through a variety of means. Using more than one validation measure enhances the credibility and technical robustness that results from any engineering effort. As such, the selection of the ontology editor is only a first step to the complete development of the MD-KB system; additional tools and processes will be required and are summarised as follows:

- Visualisation tools integrated with the structure of the KB to help the engineer understand and validate the domain concepts

- Rules engines for incorporating new interconnections between the KB facts
- Query engines to make the knowledge (facts and rules) easily accessible to the service developer/engineer

Additional methods are also required to ensure that the tools are applied appropriately and that the resulting knowledge model is of sufficiently high quality. The quality of the knowledge-base design directly affects the quality and the accuracy of the knowledge which will directly affect the proper adaptation actions determined from the reasoning engine. Validation of the structure and the contents therefore is critical. In the forthcoming section on validation, details of the use of visualisation as a method of validation will be presented. One such technique, called graph visualisation, is a subset of the broader field of information visualisation where researchers work to provide new and automated mechanisms to visually depict data sets and information in the most efficient manner without undue distraction. There is extensive work in information visualisation around the optimal representation of data in order to enhance a user's ability to analyze and understand the data efficiently. In the case of the MD-KB system, there were two types of visualisation tools required to meet the needs of the refinement and validation phase. In both cases, these tools use the graph model to depict the objects, the relationships between objects and most important, the patterns within the ontology. A tool already integrated with Protégé is Ontoviz which conveniently provides the basic graph-based visualisations of the MD-KB ontology [148]. The visualisation itself can be configured dynamically to do the following on behalf of the user:

- Display a set of classes, subclasses to validate the relationships between the objects
- Display the slots themselves and the slot information
- Display the entire ontology structure showing its full hierarchy in a form which is similar to the traditional software engineering entity relationship diagram or ERD

Ontoviz uses the Graphviz set of visualisation tool primitives from A.T&T Research [149]. This tool provides the transformation of structured information into a 2D geometric representation. The geometric representation is created from a set of graph layout algorithms and their associated filters. There are many graph layout tools provided within the Graphviz toolkit from which a developer can choose to create the appropriate graph. The Graphviz tool can create layouts using hierarchical trees, directed acyclic graphs and virtual physical or "spring model" layouts. The choice of the visualization can depend on many important factors related to the information to be displayed, the type and nature of the display requirements, the user preference and so on. Through the Graphviz tool, Ontoviz operates as a plug-in to Protégé 2000 essentially connecting the concepts defined in the ontology to the node and representational detail of the graphing algorithms in the tool. In this case, the choice of Ontoviz and the hierarchical layout was largely dictated by the fact that it was available as a plug-in for Protégé 2000. This meant that there would be no additional steps to export the ontology to a file and then import it into a visualisation tool, but instead it would be available with just one click through the tab interface in Protégé. As well stated in [149], the use of such an interactive representation helps the analysis and development of the ontology by the fact that it can react immediately to the changes within the ontology. By providing an immediate visual representation, the engineer can quickly see that new slots or facets are created, positioned and associated correctly. The graphing tools increases the accuracy of the engineering process by providing an improved "perceptual process" through visual patterns and relationship [150]. A second visualisation tool was also selected to provide yet another view of the knowledge-base. This tool was another slightly different graph-based visualization called TGViz. TGViz uses a set of algorithms from the Touchgraph library of layout algorithms to represent the ontology in a 2D form. The Touchgraph library provides primitives which enable the software to render an interactive graph visualisation. This is a slightly different visualization from that of Ontoviz which uses the undirected graph layout [149]. The TGViz software supports configurable visualisations displaying classes, subclasses and instances directly from the Protégé 2000 tabbed interface. Additionally, TGViz supports the concept of zooming into various levels of the network depths, this is particularly useful for very

large knowledge-bases which have many level of class hierarchy [151]. As will be discussed later, the MD-KB prototype is a rather sparse ontology, also called a "lightweight ontology", due to the restricted domain of interest so the zooming aspect is used primarily for aesthetic layout.

All the tools selected are Java-based, and all tools were installed on a platform using an Intel Pentium 4 2.4 Ghz processor with 512 MB RAM running MS-Windows XP. Java 2 SDK was installed on the machine and the runtime environment was fixed at 1.4.2_03. The next phase of work was to complete the backbone taxonomy for the new ontology and then to implement it into the Protégé environment.

4.5 Implementation and Refinement

After completion of the tool selection phase, the KB was now ready to be iteratively designed, developed and validated using the standard processes. There are essentially four steps that are used in development and implementation phase of the knowledge-base [54].

Domain expertise – knowledge-base systems deal with broad knowledge about a topic or domain of interest therefore it is essential to have a clear understanding of the area involved with the knowledge-base itself

Defining the vocabulary or ontology of concepts – it is necessary then to take the conceptual definitions and formalize these in structured way in an ontology

Build the relationships or axioms – in addition to the conceptual knowledge composed of lists of "things" and their unique properties, it is necessary to form relations through the use of facts about the ontology components. For example, an axiom about our ontology might be that "All Nokia devices are mobile phones". This fact or axiom is a connection between the concepts of the ontology in this case an instance of the device type called a "Nokia" and how it is categories. It is clear that axioms can be considered "true" or "false" and it is this fact that leads to the next important step in the process

Validation of the knowledge-base – after encoding of the relevant domain facts or axioms and the creation of specific instances within the knowledge-base, it is important to have an appropriate method of validation for the system. Various techniques exist for validation from the highly manual process of human expert evaluation to partially automated tools for validation. The topic of validation will be discussed at length in a subsequent section of this chapter.

4.5.1 Domain Expertise

By far the largest amount of time in building a knowledge-base or even the backbone ontology is spent in either study of the domain or extraction of understanding of the domain from a seasoned, expert in the area. A key challenge in engineering a knowledge rich solution in this area is that it is impossible to understand all the disciplines that comprise it to the level necessary to create a truly "correct" ontology. After reviewing a considerable number of resources on mobile devices -- attributes, manufacturers and their unique capability, it was clear that prior work in the standards area had created some excellent structures which could be used with some thought and modification. Rather than create new expertise or recruit and validate "experts", the novel concept is to use the open standards development process as a virtual expert and to utilize what was developed for each standard and rationalize them into a backbone ontology as a starting point for the KB. Several standards were reviewed against the needs of the ontology and two were particularly well suited:

- Open Mobile Alliance (OMA) – User Agent Profile (UAProf) Candidate Version – 12 December 2002 [152]
- FIPA Device Ontology – www.fipa.org [153]

It became clear that while these two standards based device specifications provided a good foundation for the ontology there was a need to express context as defined, not just features. A third element of input was therefore devised that included this aspect which was then called context implications. This element is an addition of contextual attributes that can be used to infer context opportunities based on the device slots within the

knowledge-base instances. So while the two standards could be merged together and validated, neither standard specification addressed the context as an element unto itself. The conclusion from this is that while discrete specifications of the device could be known and documented, the opportunities and the context that these specifications can afford is not readily known, and therefore it is left to interpretation by the individual service developer or engineer. This is consistent with the earlier assertion on the nature of context adaptation in terms of propagation of adaptation architectures. The use of the OMA UAProf specifications provides the basic class and slot structure for the component level of the knowledge-base. The initial ontology specification is depicted in the Figure 4-8 below.

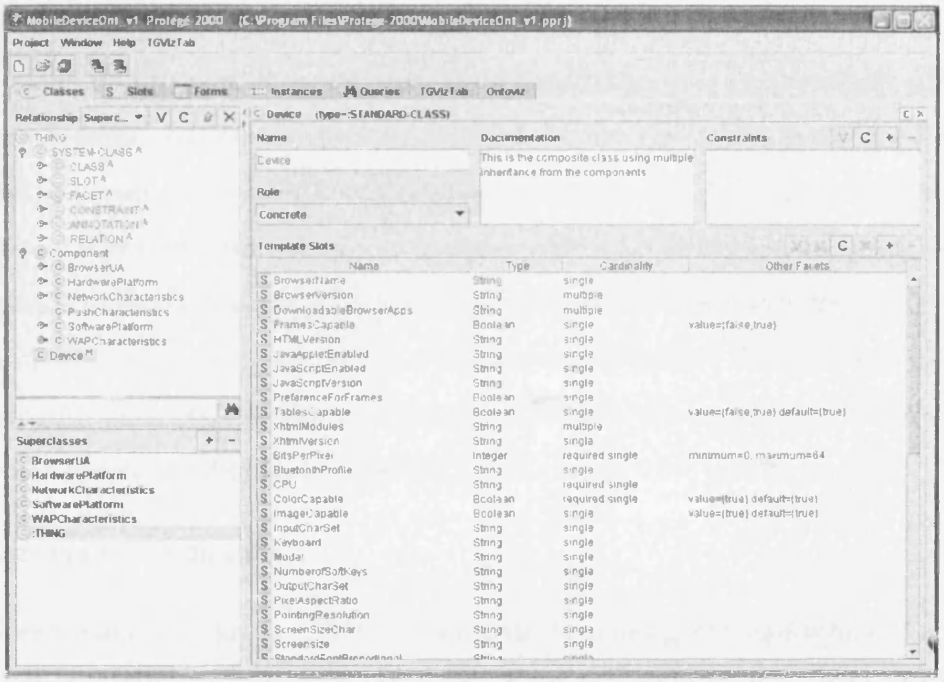

Figure 4-7: Mobile Device Ontology from FIPA and OMA UAProf Candidate Version

Two important aspects are missing from the UAProf description which are essential to the context adaptation process. The two items are the resulting context description which is available from any one instance of the device and a class which specifies user behaviour. The OMA specification of UAProf is sufficient for describing the situation

from a device and network perspective, but does not provide the foundation for the user preference component of the context 4-tuple.

Using the principles of knowledge engineering coupled with the development work done to produce a context-aware adaptive prototype clearly demonstrated that this gap would prevent the full adaptation at the level necessary to deliver the value added services. The second outcome from the knowledge engineering effort was the discovery that the frame based architecture coupled with the use of the OMA specification does not actually specify context, but in fact supplies data elements that can be interpreted by the developer to formulate a view of potential adaptation. The knowledge-base itself however, does provide the means to express at least at the component class level the nature of the context adaptation opportunities. By using a slot under the component subclass, the KB can provide context potential specifications to the developer based on the input features. As mentioned in the previous section, the feature to context mapping may still be a manual knowledge engineering task at present. In Chapter 7, the concept of using evolutionary computation to perform string-based context mapping will be discussed. Future work on an automated method using evolutionary theory as a model for feature to context mapping will be presented in Chapter 8. This concept combined together with the mobile device knowledge-base means that a fully automated context adaptation system would then be feasible.

4.5.2 Defining the Vocabulary

The second major task for the KB implementation is defining the vocabulary. To define the vocabulary of an ontology is to understand the intrinsic knowledge that it is representing and to determine how to represent the intrinsic meaning of the domain. The vocabulary for context adaptation is more complex than at first it might appear. The difficulty lies in the fact that as previously stated context is like beauty, it is in the eye of the beholder. The definition is variable across projects and even within projects. Defining a meaningful vocabulary is then an iterative and domain specific exercise. It is the combination of the ontological concepts as the words of the vocabulary coupled with the nature of context as it is inherent in the knowledge of the domain of the ontology

that defines the vocabulary. To understand how to model the context opportunity into the knowledge-base structure it was useful to look for how others determine context from various sources. By investigating the source of context derivation, the implications of the derivation can be properly modelled into the knowledge-base. There is limited research in this area, but one promising approach is called "feature extraction". Although this term is used in many areas of imaging to mean various algorithms to extract meaning from a scene, in this instance the term is used to indicate the extraction of information about features of a device from various sensors on the device itself [154]. This work is philosophically similar to the work in this thesis in that it attempts to derive context potential from a set of low level sensor signals from a device. The general process for this understanding or inference of context is to acquire the available low level data, extract the feature relevant information and then use some form of classifier to convert these informational elements into a meaningful context for use by an application in the adaptation process. These steps are clearly consistent with the process developed and used here. An exception to this is the separation between dynamic and static context. In [154], all contextual adaptation is assumed to be dynamically inferred from the sensor data. This architecture is an ideal one for mobile and wireless devices, however, it is difficult to see how this approach is feasible in light of the finding above in a QoS-constrained environment.

While there is some empirical evidence of feature to context variable mapping as mentioned above for the purpose of adaptation, there is little formally published work describing a mechanism for this process. This mapping is done as a part of the application design procedure with the development team using domain experts. In many cases, these experts are working with the developer of the application or in fact, the domain experts can also be the developer for small scale systems which are within their own area. Alternatives to the use of a team of human experts exist although there is still manual intervention in the process. The alternative of using a public standards process to develop the core ontology is one such example. The work in this chapter started with the outcome of the standards process as the fundamental structure and then through evaluation and analysis, refined this work to fit the requirements of the mobile

device knowledge system. The reduction of the amount of the human expertise required does enable the technology towards the direction of machine automation. The ideal systematic situation would be the knowledge component of the knowledge-base populated through the use of automated learning from web-based content or via direct file-based input from industry rather than the highly manual process undertaken to populate the KB in this work.

4.5.3 Building the Axioms and Relationships

It is useful to look at this step in comparison to the human thought cycle where there is knowledge acquisition, reasoning and action or outcome. While action is not often equated to maintenance in the life cycle of a knowledge-base system, the outcome of a reasoning process produces knowledge as well as a specific call for action. The system can be directed to do a certain function, in this case, perform a series of specific context renderings from which new knowledge is acquired and needs to be added to the knowledge-base. This parallels the previously cited human thought cycle. When the three component thought cycle model is working together you obtain what has been previously called an "expert system". The details of the expert systems are widely published elsewhere, but the key point of interest here is that although the expert system closely models the device capability knowledge-base system, the main difference is in the manner in which the system is initialised with knowledge and how it is maintained and updated. Expert systems as defined in [155] back in 1988, require the use of the domain expert to populate the knowledge, the knowledge engineer to build the knowledge repository and to develop the formalized language with which decision can made. For the device capability knowledge system, the acquisition of the knowledge will be through the manufacturer standardized specification, and the rule inference will be learned through information extraction from various sources. No knowledge-base can be complete without a process for acquiring and rationalizing new knowledge so although this is often done as the last component of development projects, in knowledge engineering it is important to include the process of maintaining and assessing the accuracy of the knowledge-base during the design and implementation phase. So far, the MD-KB represents only a shared global "memory" of currently accurate and

evolvable device and contextual facts. These individual facts need to be connected, unified and evaluated to create a representation of the context situation that the service would operate within. A method is needed to construct this frame of context potential from the relevant facts. In modern AI theory, the process of connecting facts and developing consequences that may result in a rational action is called "reasoning". The concept is that the current state of the object or situation in question will be inferred from the context of the knowledge-base through an application of a suitable reasoning process [53]. The details of reasoning systems can found throughout AI literature and in these references [54], [144], and [156]. The foundation of automated reasoning is an expansive topic in intelligent systems and cannot be covered adequately here. The principle interest in reasoning systems for this work is to complete the engineering of the alternate architecture, which will be abbreviated C2Adapt2. This new architecture and its resultant system requires the inclusion of an appropriate reasoning system, so although fascinating the study of the fundamental nature of reasoning systems is beyond the scope of this work.

The MD-KB, which forms the central component of the new model, C2Adapt2, requires a method to combine the static and dynamic context together and provide a way for the service to infer the contextual potential. For example, the service developers will need the ability to determine if a device is "image capable" from the combined static and dynamic context representation before advertising the appropriate service endpoints to user's device. The state "image capable" is a set of values of the context 4-tuple that matches the definition of the term within the knowledge-base itself. Even the phrase "image capable" is open to interpretation or reasoning by the service developer depending of the requirements of the service to be deployed and the quality of service required. These activities as outlined require reasoning ability about the action necessary in light of both the frames extracted from the knowledge-base and other dynamic information acquired from the service request header. In choosing the frame-based knowledge representation approach, the choice of reasoning engine is in some ways already implied. Frame-based systems are reasoning models in themselves because the frame structure or semantic network embeds a logical relationship within it.

This fact makes the frame-based system straightforward to use when it comes to its inference execution model [54]. The third element of the thought cycle is that of the resulting action. The action in this case, is the resulting evaluation of the context which then will provide a set of adaptation instructions for the delivery of the service. The action that comes for the MD-KB reasoning is not the delivery of the services, but the action of informed evaluation of the context for the purposes of context adaptation. As discussed in Chapter 3, the evaluation of how to delivery the service needs to be informed by context mapped against requirements and this is the action that results from the reasoning out of the frame knowledge-base. For example, in the imaging services scenario, when the device context resolves to "image capable" it is left to the service provider to then initiate the desired action. That action may be to further define the extent of the quality of the image possible, combine it with other necessary service attributes or it may be to simply invoke the remote method or code to deliver the service itself. Ideally, the reasoning component of the MD-KB will be capable of handling not only the clear, straightforward resolution of device capabilities against the service requirements, but it should also be able to handle some measure of uncertainty in the mapping. The principle cause of the uncertainty is the specification of the level of quality and the extent to which the capability is matched to the requirements. It is important from an engineering perspective to determine, not only if the device properly handle the service, but how well it will match the requirements. A simplistic mechanism would assign a rank ordering summation based on some point score for each applicable attribute for a given context need. Using this mechanism, the MD-KB provides a method for dealing with the uncertainty of service-context mapping. The need to handle uncertainty in inference is well-documented and a very good summary of the issues in this area is found in [157].

While the MD-KB reasoning subsystem is not necessarily intended to explore the detail of inference systems, it is important to understand the state of the art in inference systems and what methods might be most applicable to the problem at hand. One way to implement inference and specifically inference in an uncertain world is to use the probabilistic Bayesian networks integrated with the frames within the system. The

techniques outlined in [158] essentially connect the slots in a frame to the nodes in the Bayesian network and construct a table of probability values which indicates the probability for each set of slot values being the inferred value [158]. The work done in [158] involving the creation of a linkage between the frame-based representation of knowledge and the Bayesian network using OKBC is precisely what is needed for the MD-KB although their approach is at a level of granularity lower than MD-KB. In the MD-KB, there is a presumption of slot cohesion or low uncertainty between the KB slot values. This is because the slots are describing elements of an integrated, uniform physical product already so the slot values are necessarily related in the real world. The uncertainty in reasoning for the MD-KB is at the higher level than slot values-type; rather it is at the semantic interpretation between the combined frames represented by the instance. The question isn't whether a slot belongs to a frame in MD-KB, the reasoning subsystem must decide what combination of slot values actually properly match what combination of context requirement slot values.

With the basic foundation for the reasoning subsystem established and a clearer understanding of prior art in the area, a choice of a specific engineering approach to the reasoning engine is necessary. A review of those which would integrate well with the Protégé 2000 ontology tool showed one of the best and most flexible reasoning subsystems for Protégé is one called "Algernon". Algernon is a plug-in based on the Algernon abstract reasoning machine developed at the University of Texas in Austin, Texas. Complete details and description of the academic foundations for Algernon are not covered here but can be found in [159]. The rationale for the use of Algernon and not, for example, the FRS integrated Bayesian network of Ontolingua is based on engineering and accessibility of useable tools for the environment in question. What was needed in this implementation was not a complete, bespoke application since MD-KB was already capable of providing the ontology editor, what was needed was a standalone reasoning engine which could integrate into an existing frame-based system, but not completely replace the components which had already been constructed. Algernon provides some essential capabilities for reasoning within a frame-based logic system, the most significant of which is its fundamental implementation of access

limited logic, or *ALL*, and its ability to do both forward and backward chaining of rules. The academic foundation of the access limited logic is well-documented and added to the convenience of using the Algernon tool. The greatest disadvantage in using Algernon is that the underlying architecture of the reasoning engine does not permit global searching within a knowledge-base, meaning the system requires that a search must be started with a known frame. This is only a small issue for context collection and adaptation system because the service request requires initiation from the destination device which will provide the linkage into the frame of the MD-KB thus a global search is not a pre-requisite although it does provide flexibility for other purposes, for example, searching for the nearest similar device in the event that the specific device is not found in the repository. Algernon-J, which is the most recent version of the Algernon system, is provided both as a plug-in to Protégé and as a standalone inference engine [160]. It is implemented in Java, although its predecessors were written in CommonLisp and C++. Algernon is also licensed and implemented into commercial products. An important aspect of the access limited logic language foundation in Algernon is its ability to prevent exhaustive searching. The MD-KB will possibly contain thousands of devices, each with hundreds of permutations of features and context potential. As the KB grows, some method of maintaining the ability of efficient query and retrieval will be necessary. Algernon, through the use of *ALL*, was designed with this purpose in mind and will permit efficient searching of very large and dynamic KBs.

4.5.4 Inference in MD-KB

The Algernon tab widget plug-in installs easily within the Protégé 2000 ontology development environment. For the MD-KB, the Algernon-J version 4.0.2 beta is used and further details of this version can be found in [161]. After installation, sample queries and rule chaining are tested to ensure the plug-in is operational. The first step with the MD-KB ontology itself was to assert truths about the application domain. Simple facts were easily inserted and were based on the observable properties of the devices and their resulting context potential from a user perspective. The source of many of the base assertions is the set of device specific requirements that the service developer is working with when creating the service for a particular market segment. In

a commercial setting, product requirements derived from marketing specifications and market research are used as a direct source of such assertions. For this work, however a set of starter assertions is entered manually into Algernon interface provided and shown in the figure below:

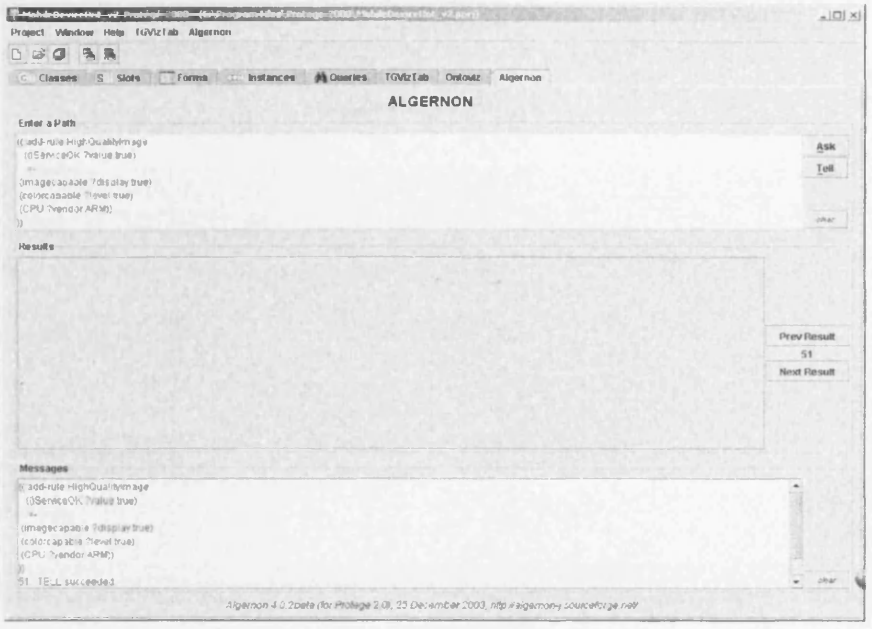

Figure 4-8: Algernon Rules Entry

Basic assertions are service specific, or in the case of the more traditionally constructed systems they are application specific. For example, the imaging service for context adaptation may require the following in order to adapt accordingly:

- A display no smaller than 5 cm by 5 cm
- A display of at least 8 bit colour
- Software module for viewing jpeg 2000 file, i.e. MIME-type registration
- A browser of some general type and support for HTTP v1.1 or greater

These requirements will form the basis for the ground truth assertions which when combined together will create the context potential category called "image capable". The

assertions will be in the ALL format and the example in Figure 4-8 below shows the assertion for an operational CPU which is the combination of a validate battery power level indicator at an "3.5" level and a system power status of "Good".

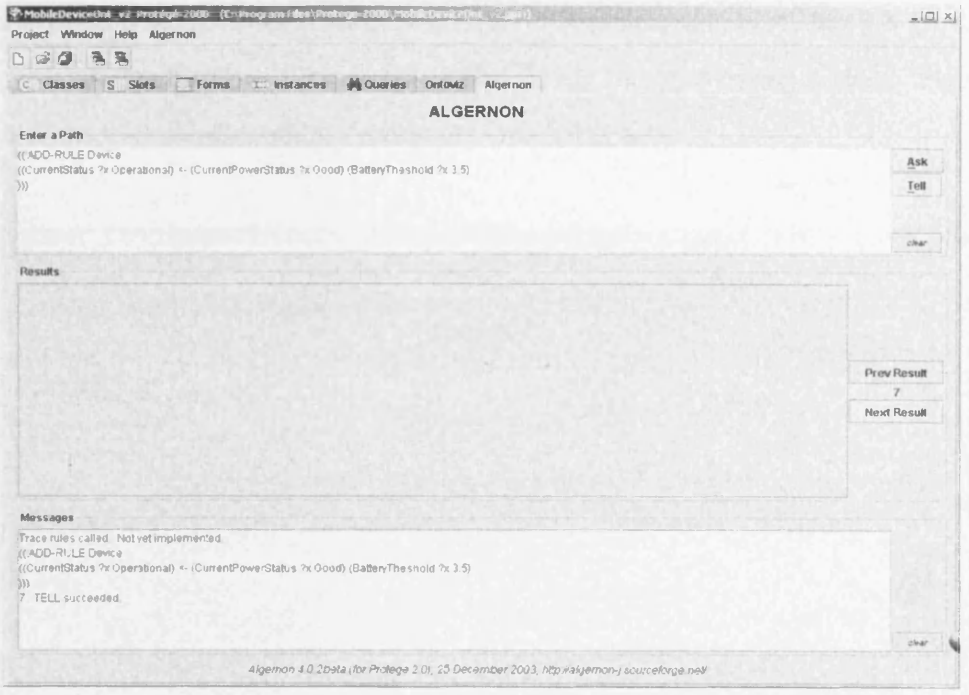

Figure 4-9: Algernon device level assertion

Architecturally, these assertions can either be private to the specific service developer's environment or they can be shared amongst the service developer's using the open knowledge-base. This is similar to the descriptive assertions found in the CC/PP standards which can be private or public. As will be outlined in the next chapter, these definitions of service definition are handled through XML namespaces representing the descriptive nature of the service components and the relationships between them. Given the Algernon java implementation and its API, an XML namespace representing the descriptive assertions or relations could be read in from either private or public service description and could be directly useable by third parties as long as the semantics of the service are universally consistent. A set of assertions were entered into the KB along with the basic set of mobile device frames. Several inference queries were made based on the test data in the Protégé MD-KB. The principle method for testing the inference

was to use the competency questions outlined earlier in the chapter in the form of Algernon queries. The information to answer the queries was easily retrieved from MD-KB and answered the competency questions. Using backward chaining of rules, the "HighQualityImage" capability was created as a rule having learned this from the entry of the ground truth assertions. The rule was subsequently stored in the KB for future use in inferring the nature of the quality of the image required by the service. Figure 4-11 shows the backward chaining executed to create this rule:

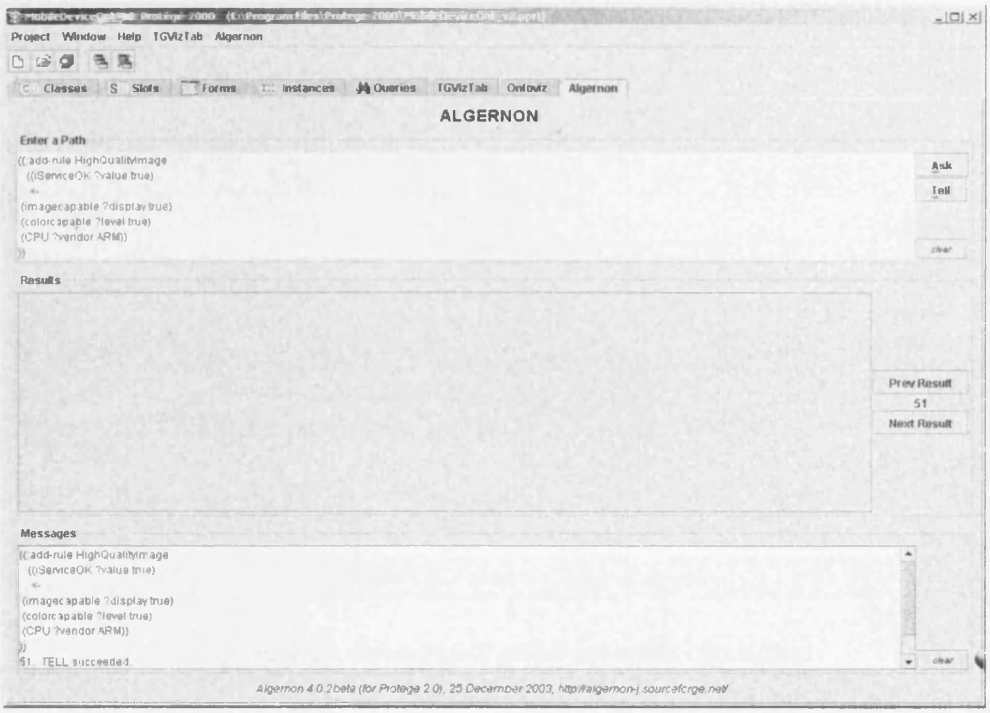

Figure 4-10: High Quality Image Rule specification from service context

Several other rules, derived from the initial set of service requirements were also implemented through manual transformation of requirement -> rule. Forward chaining inference for "Current State-> Operational" was shown in Figure 4-7. One added benefit of the inference investigation is that it demonstrates the "active" specifications that were not a part of the OMA or FIPA standard very clearly. By identifying the additional power descriptive slots for context adaptation in power scarce situations, the power sufficient rule is inferred and then stored in the KB. Automated inclusion of both new instances of devices themselves and the associated frame slots describing the new model

of the device is well supported. The concept of using rules to determine that a new occurrence of a device is actually worthy of a new device frame is simple to implement. Additionally, by the theory of Socratic completeness which Algernon has been proven to implement, these new instances will be accessed in the KB through query constructs that are time efficient [162].

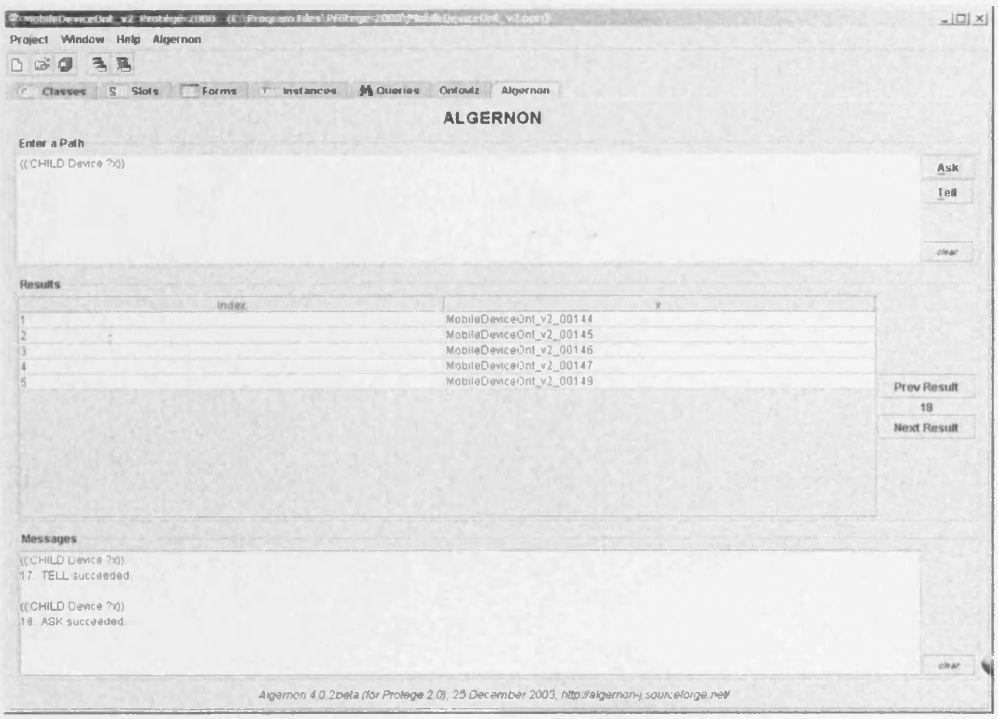

Figure 4-11: Example of child instance creation

As Figure 4-12 depicts, new contextual element can trigger a new unique frame to be instantiated that reflects the change. In this way, state changes in the device are automatically reflected in the KB when triggered by new static information or by the passage of dynamic information relating to a specific instance of a subscriber device. The advantage of this approach mobile device state change allows the development of a transition diagram similar to the one shown in Figure 4-11. Over time the series of transitions diagrams can visually demonstrate the most dynamic elements of a market segment of devices which are compared to a set of service deployments. The use of context change trajectories would be a beneficial analysis tool for market research, development testing and product planning for future service deployment.

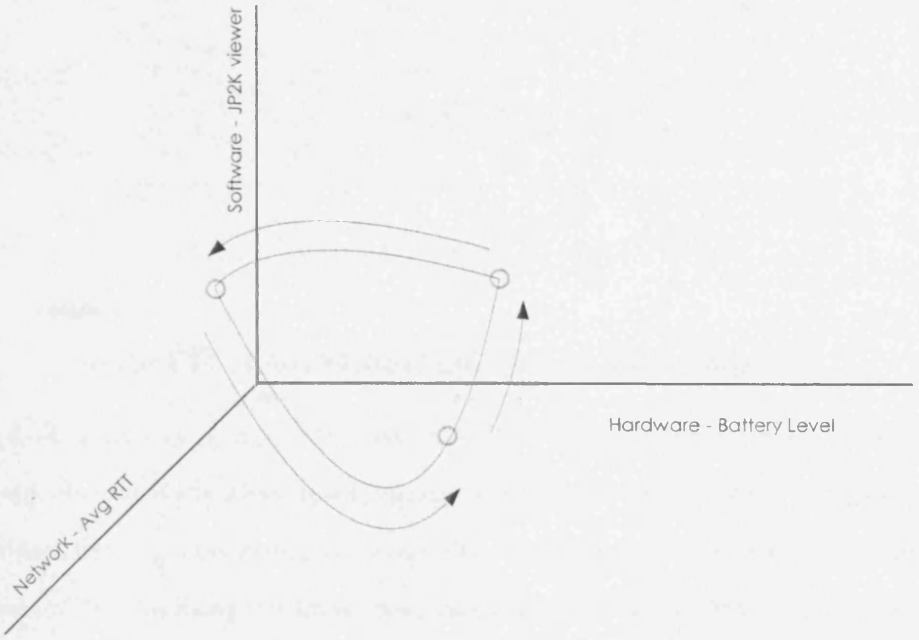

Figure 4-12: Possible device context transition diagram

The fact that the knowledge-base can track such changes, allows the addition of entirely new features sets and devices. This ability to modify or completely add new families of devices provides the required feature; to accommodate the dynamic nature of the mobile device market. The structural and operational validation of the MD-KB including the reasoning engine provides the necessary proof of this ability.

In order to take the next step towards applying a knowledge-base approach to the specific instance of an imaging context-adaptation application, a discussion around the formal foundation of the MD-KB is beneficial. The work so far has provided a feasible alternate architecture to deal with the static and dynamic components of the context 4-tuple but has not manifested this in a usable form. The issue of device feature to context variable mapping has been discussed and a method for the conversion of this information was provided. This next step is knowledge engineering where the ontology in the mobile device knowledge-base is created.

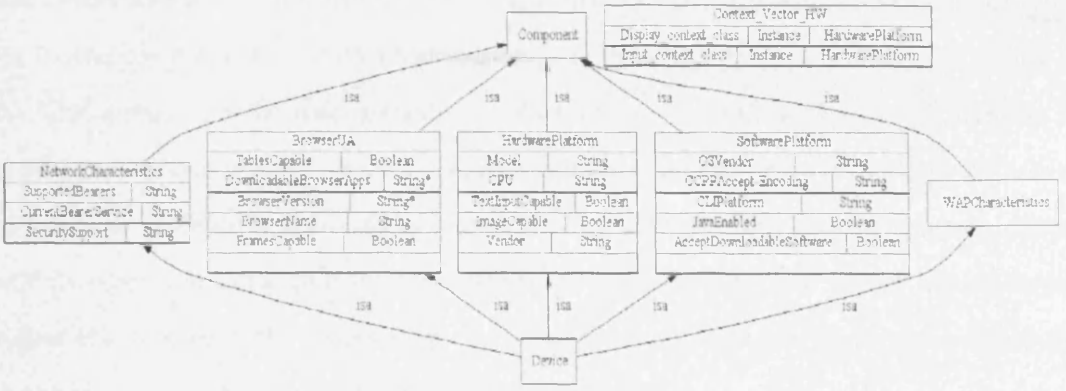

Figure 4-13: Initial Mobile Device Frame-Based Structure

In Figure 4-14, it should be noted that the benefit of using the frame based system is the ability to use the multiple inheritance feature to create a virtual subclass called **Device**. This subclass although concrete, is completely composed of inherited slots from the superclasses which logically comprise the component. From a modelling perspective as well as from a development perspective, this was not necessarily intuitive. In most OO-based designs, the **Device** would have been at the superclass level followed by the components and then followed by the specific categories of component such as **HardwarePlatform.** This hierarchy makes logical sense in the way the real world of devices are thought of and for the most part, designed. However, it becomes clear as instances are created that this structure will not function as needed. It is also apparent that the opportunity to use the inheritance in the knowledge representation presents some advantage in the design of the KB. The model itself means that that as the **HardwarePlatform** slots change or if there are additional slots added, the **Device** class will inherit those as well. This will provide a superior level of flexibility to the architecture as compared to the OO-approach that would have specified the seemingly more logical model. The initial structure was primarily hierarchical in nature, taking a relational approach to the knowledge-base; this was soon shown to be an engineering mistake. The iterative validation of this first KB was done with the creation of instances and the comparison of those instances against a test set of current production mobile device models. Simplification of the structure was done through the evaluation of each

slot, each class and the relationship between the classes. Completing the knowledge-base involved the addition of the context and the specific difficulties around specifying this intangible property. This additional "spur" structure is shown clearly in Figure 4-15. The device specific information is linked at the highest level through the use of context and device frames. In the visualization this appears as two connected spurs of information essentially showing a loose coupling between the concepts which allows both to expand without unduly constraining either. Through the use of the reasoning engine the two central concepts can be related in real-time, or as in the case of the prototype, constructed statically and evolved over time.

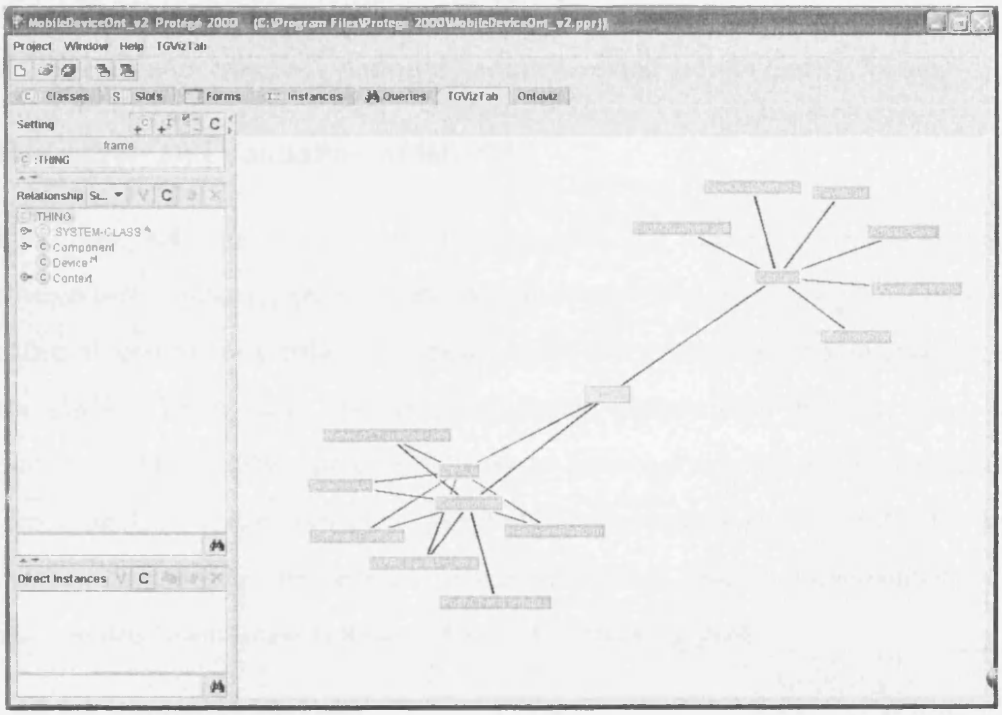

Figure 4-14: TGViz visualisation of initial MD-KB

The mobile device resultant frames using the Algernon inference entry system can be seen in Figure 4-16 with the actual information supplied in a representation more like the ERD diagrams traditionally used in software development. The instances that were entered using Algernon forward chaining are indicated in the diagram while the previous KB ontology are denoted "original".

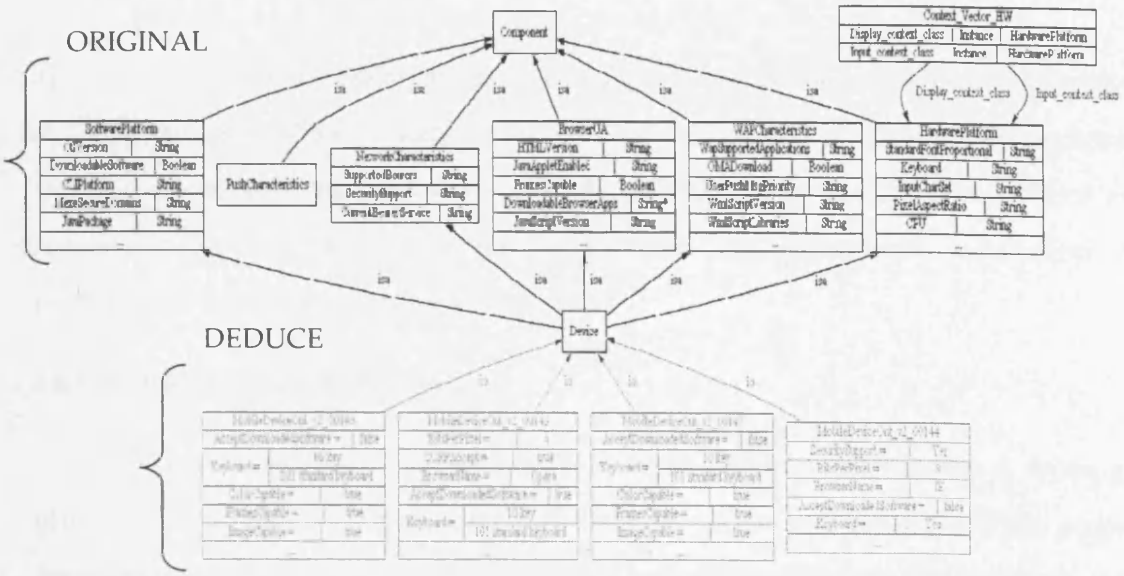

Figure 4-15: Algernon reasoning engine created mobile device frames

4.6 Evaluation and Validation of MD-KB

With the basic KB structure substantially complete, the next phase in the process of knowledge-base system construction is validation and evaluation. Ontological validation is one of four critical components for the evaluation of a knowledge-base system [163]. Verification, usability and utility assessments are the other three components. The validation process itself can be broken down still further based on the domain of the knowledge system and the design requirements known to the developer. To evaluate the mobile device knowledge-base system that is implemented in this chapter, the validation phase is designed with the following goals:

- Validate sample data and the resulting instantiation of the ontology is consistent
- Design a method for suitable visual inspection of the knowledge-base populated with sample data to reconcile the representation with the known facts relevant to the concepts
- Successful application of a methodological tool which will assess the quality of the ontology and its resultant translation into the Protégé 2000 knowledge-based structure

This list of goals for the validation process leads to a specific and customized evaluation of the knowledge system which is the result of this work. As an engineering effort within a specific domain, this work is not intended to develop new generic methods for ontological analysis, but rather to demonstrate the appropriate application of foundational techniques developed elsewhere.

4.6.1 MD-KB Validation Methodology

As previously mentioned, knowledge-base systems development requires a thorough process of validation. The critical importance of accuracy coupled with the proper representational semantics means that validation is necessarily a multi-faceted process. The first phase of the validation is the consistency checking of the contents of the knowledge-base as instantiated. There are a multitude of methods for consistency checking of the information in the knowledge-base which are largely dependent on the type of logic used to construct the knowledge-base itself. A number of these methods were reviewed and three methods were selected. The application of the validation methods and the outcomes of the analysis are outlined in the following sections. The goal in the validation process was to iteratively validate the current facts in the KB as it was designed and to ensure that the foundation of the system which is represented by the backbone ontology accurately reflected the real world model.

4.6.1.1 Validation through Instance Analysis

The creation of instance data in the knowledge-base provides a first level evaluation of the design. This process involves taking real world samples of mobile devices, retrieving the available specifications for these devices from publicly available sources, and creating instances of these samples within the structure of the knowledge-base itself. The first pass of this process took a sample set of twelve commonly available devices from various manufacturers. The specifications were retrieved off the Internet from any available source including the manufacturer, any local distributors or third party product evaluators. The sample devices were taken from all platform levels – mobile

phones, personal digital assistants (PDAs) and notebook computers. A list of the target devices is provided below:

- Nokia 7650™
- Nokia 6310i™
- Sony Ericsson P800™
- Sony Vaio™ PCG
- Compaq IPAQ™ 5550
- HP Jornada™ 720

Instances of the devices were easily added to the knowledge-base using the Protégé instance menu system. Although the process was straightforward to create the instances, the interface showed that the number of slots for a complete record was considerable and that for an operational system; a useful function would be default models which would provide a mapping for similar device types and using those default values which would improve the overall maintenance of the system. The maintenance aspects of the MD-KB system will discussed in further detail in a subsequent section of this chapter. The output of the instance analysis work is seen in the figure below:

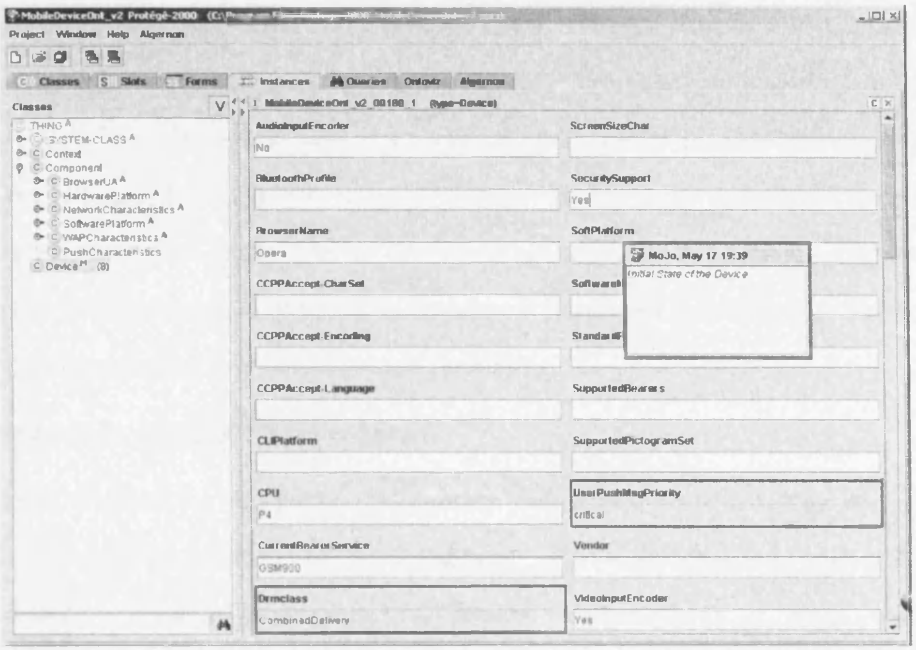

Figure 4-16: Instance Entry Form with Required Fields Highlighted

Figure 4-17 depicts the basic entry form for each instance of the mobile device. In the red outlines are field which are required and the note specifies that the device frame is an example of the initial entry for that device. The convenience of the frame knowledge-base is that subsequent changes to the device will create new instances of the same frame with only the changes to the slots. This models exactly the behaviour of interest in the system, wherein the mobile device configuration is a point in the state space of the system and as it moves through various configuration the original frame does not change but it attributes can change always maintaining an efficient history of its path through the state space. Although the entry form is convenient for prototyping, the production KB would be initially populated with a series of XML documents from manufacturers. It is clear that a number of slots can be specified from the "as-built" configuration of the devices particularly if those slots can be filled through an XML feed from the vendor, but also that many must be specified at either the set-up of the device on the provider network or through dynamic discovery during a service requested session. Several problems with the initial model implementation were uncovered through evaluation using instance checking, this led to several changes particularly in

the area of required slots, facets and default values. Other issues like invalid data entry masks, or insufficient entry lengths were all easily identified and corrected using the instance validation. However, this is a manual and inexact process dependent on the domain knowledge of engineer to uncover issues. It also takes a study of the attributes of each device and then entry of that device to successfully validate the contents. The fact that the structure can accommodate the new information is important though and is useful validation during the development iterations.

4.6.1.2 Validation through Visualisation

An informal method of validation previously mentioned, is the usage of techniques found in the area of information visualisation. The technique provides yet another aspect of the validation procedure. Complex, unstructured or large volumes of information can be difficult to validate using text-based means. New methods of interactive visualization are emerging which have many applications from exploration of vast information spaces, to new forms of interactions with clusters of information within a search space. There is also the opportunity to use the visualization of large information spaces as a more efficient means to validate the relationships and contents found within the information in the knowledge-base. Research in the area of information visualisation methods and techniques has increased steadily over the last decade, particularly in response to the opportunity of knowledge systems based on the semantic web [164]. The key aspect of interest here is the ability to use the visualization techniques as a way to understand the patterns of information and then to reflect back on the domain itself, validating those patterns by the human expert. This process is essentially the "high level" validation whereas the consistency checking provides the more granular low level validation.

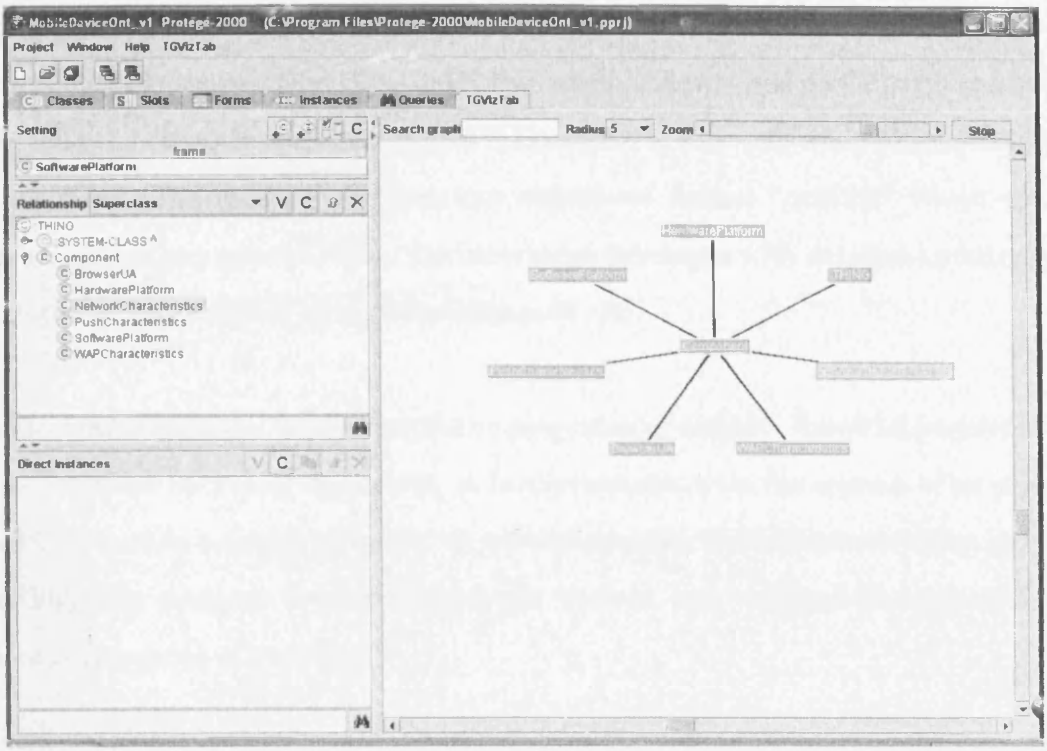

Figure 4-17: Modified View of the Ontology after First Validation

Figure 4-18 shows a simple visualisation of the lightweight ontology after some correction of structure through instance analysis. This visualisation shows the central position of the device with the various OMA user agent categories emanating from the device at the component level. This view confirms the relationship as "hub and spoke" instead of hierarchical which matches the real world model more closely. This view also acts to confirm to the designer that the proper relationship between the backbone ontology exists and that the instances of the representation of real devices can be modelled appropriately within in the structure.

4.6.1.3 Validation through Ontological Analysis

Yet another useful validation method which is based on more formal foundations is that of ontological analysis. Although much work is done on the appropriate design of the knowledge-base before implementation, to improve the quality of the system a validation of the post-implementation structure is also necessary. The ontological analysis needed in this case, is one in which the framework of the knowledge-base can be analysed to assess whether the design decisions made during the modelling phase are

appropriate [61]. The OntoClean methodology is one such formal mechanism for general ontology assessment. The OntoClean method developed by Guarino and Welty has been used on other ontologies and is the basis for at least one commercial product as of this date. The method specifies four significant formal "notions" which can be applied against any general KB by the knowledge developer with detailed knowledge of the domain. These formal assessment constructs are:

Essence and Rigidity – Two principles or properties of objects. Essential properties are those that must be true of that object. A further constraint on the essence of an object is what is termed as a "rigid" property or one that must be in all instances of the object in question. The designer should analyse the domain and segment the concepts into essential properties of the object.

Identity and Unity – These two properties are slightly more esoteric but no less important to each object in the knowledge structure. The identity pertains to that set of characteristics which make an object recognisable as an object. Unity on the other hand is the ability to indicate all the components of an object as necessary and part of that object itself. It is clear that the confusion in these two terms comes about from the common methods around object identification, in that many objects are recognised as an object by its component parts which will blend the two properties of Identity and Unity in every day experience.

Backbone Taxonomy Identification – through the analysis of the knowledge structure, the knowledge engineer will begin to identify a connected set of concepts which are all deemed "rigid" by the definition provided above. The connected structure will represent the key properties within the knowledge-base itself. The benefit of identifying the backbone is that this will identify the structural support for the hierarchy itself and the engineer can ensure that these concepts are well-designed and tested against the real world system that it intends to represent. Although it is difficult to prove, it is logical to assert the notion that the standards development process results in a structure that has a

prominent backbone "taxonomy" and is central to the thesis of this chapter. Validation of the backbone taxonomy is difficult as there is more than one possible backbone for any given domain. An approximation of this validation is that every device must be capable of being modelled by a subset of the taxonomy and that it must be expressive enough to provide these details in support of the most basic adaptation required in the system.

Subsumption Analysis – A critical design decision when developing an ontology is that of the subsumption relationships between properties. The use of subsumption to denote the fundamental taxonomic relationship is powerful, but often difficult to model correctly in the real world. The principle in the OntoClean methodology regarding proper subsumption is that every instance of a subclass is also a necessary instance of the superclass above it [61]. Validation of the subsumption relationship in MD-KB was performed both through the OntoClean process and through the evaluation of the instances of the mobile devices as specified through the Protégé 2000 interface. The different methods of validation revealed different issues and inconsistencies with the structure and the ease of use of the entry of the instances. The validation with a standardized process such as OntoClean provides a good framework for the ontological assessment. However, there were no material changes to ontology that resulted from the application of the OntoClean process. It is possible that this resulted because either the tools was not applied correctly as it did not uncover errors or because the ontology is so straightforward and simple that in fact the model is essentially correct. The other observation is that it is also possible that the use of the standards development process provided an ontology which was essentially compiled using the tenets of good ontology engineering which are embodied within the OntoClean methodology. Logically, if this is true, then this observation makes some sense and argues for the use of standards-based structures and outcomes more often in engineering development. From an engineering perspective, the use of the backbone taxonomy identification is essential in ontological engineering as it creates a "critical path" structure out of the ontology and allows the designer/engineer to ensure that this critical path matches the hierarchy of the

domain in the real world. This is a valuable engineering technique and one worth building into the engineering methodology.

4.7 Maintenance and Operation

The process of creating and maintaining an accurate mobile device feature and contextual information store is critical to the success of the new architectural model. Without automation of this task, the knowledge-base will become unusable in a short time. It should be clear that the ability to adapt against the static component of context derived from the database will be dependent on the accuracy and relevancy of the data contained with the KB. It is therefore important to understand the possible engineering solutions to new knowledge acquisition for the system as well as the challenges that will be faced even after the appropriate solution is implemented. There is considerable research and many general methods of automated knowledge acquisition which are in the literature [165],[166],[167]. Specifically for this knowledge engineering effort, the appropriate methods for population of the knowledge-base can be classed as

- Manual entry through some online interface
- Manufacturer automated input - using XML product specifications
- Information extraction from structured informational websites

One of the significant challenges with the use of a centralized knowledge-base for static device context is the maintenance of such a system. However, when it comes to accommodating entirely new concepts which cannot be inferred, it is necessary to provide a simple means for the repository to be kept current. There are a tremendous number of existing devices which are potential platforms for service provision as well as a vast number of combinations of features for contextual adaptation. The system must be constructed to support both a flexible process and an integrated set of functions that allow the acquisition of new concepts. The best process for large scale information handling is often to get the information from a source as close to the information originators as possible. This helps to reduce the possible errors in the information which

would then negatively affect adaptation and its subsequent usage. Manufacturers of these devices can provide specifications of their devices directly to the knowledge-base through a web service. They can also have updated specifications fed routinely into the KB via open application programming interface (APIs) supplied with the open knowledge-base. The set of accurate attributes straight from the manufacturer is available to all context-based application developers. At a minimum, a single set of static context variables can then be assured to provide a minimal yet level playing field for service adaptation. The question is what might motivate device manufacturers to provide such specifications and routine input information into this third party open knowledge-base? Primarily the potential for a greater number of far richer, quality services is the motivation. By doing so, the manufacturers of devices will simplify the acquisition of the basic, static device attributes for all service developers and through increased accuracy of information on which to adapt the context, they will improve user experience with their devices.

4.7.1 Machine Learning for Knowledge Extraction

By themselves, knowledge-bases are merely holders of an evolving set of facts and interrelationships as stated by some knowledge expert. While this can useful, it is far from providing the ability to learn in order to make knowledgeable decisions about the appropriate and possible states of a problem within some computerized system. The reasoning engine provides a way to create knowledge through inference on existing facts using the Algernon subsystem. The MD-KB can grow through manual input, but as yet, it cannot grow larger or more accurate without additional system components to provide the tools necessary to acquire new knowledge. As a specific analogy and one that was discussed briefly earlier in this chapter, the human thought cycle presents a well-used model for knowledge systems. In this cycle, there are several necessary components, e.g. memory, reason or logic and action. These three components are essential to how thought is performed at the highest level. The knowledge-base at the centre of the MD-KB roughly represents the memory component of the human thought process. To populate our artificial memory, the MD-KB, it is necessary to find mechanisms or design subsystems that will take existing information sources and feed

that information in an automated way into the MD-KB, our knowledge-base. Finding and retrieving the relevant information from the vast resources across Internet is a significant challenge, generally referred to as knowledge extraction or data mining. Knowledge extraction from web-based sources is an active area of research among the artificial intelligence community. The two areas of information extraction of most interest to this work is the ability to extract device specifications from manufacturer's web pages and the ability to take a structure data file perhaps in XML from a manufacturer as input to the knowledge-base [165]. These two automated processes could populate the knowledge structure as it pertains to the basic elements of the mobile devices entering the market. However, the reasoning component which is a set of axioms and rules about the basic features is not maintained by the concept loading of the KB itself. A second procedure would be necessary in order to create the context adaptation rules to utilise in reasoning. In this second procedure, one method would be to open the knowledge-base up to the development community in a participatory manner where developers would contribute their examples of services using context and these would automatically mapped on to the concepts in the knowledge-base of devices. Conceptually, the maintenance process would function as outlined in Figure 4-19 below:

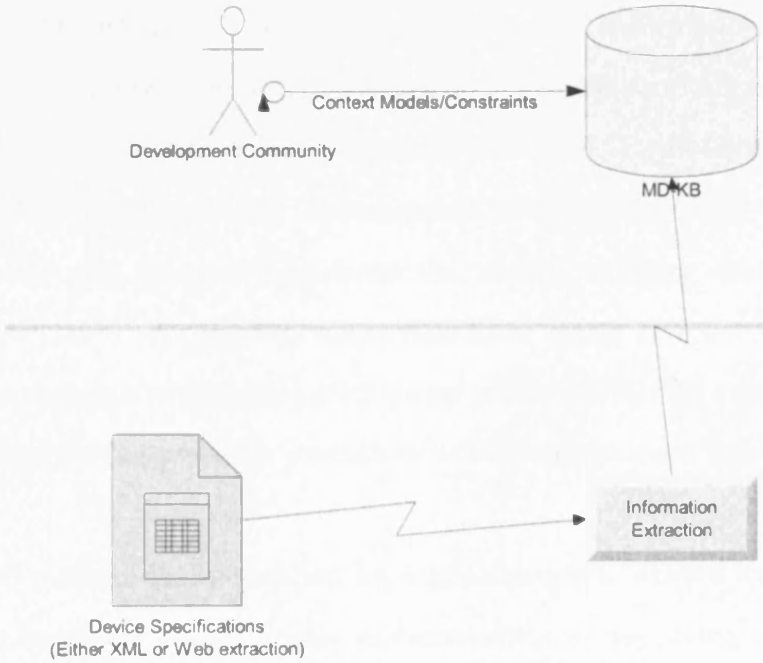

Figure 4-18: Maintenance and Operation Process

The automated extraction of information from the unstructured hypertext is a difficult task which is itself an active field of study. There is considerable research in the area particularly as it relates to maintaining information repositories and retrieving relevant information for the purpose of decision support. The two methods of maintenance, fully automated or community augmented each have their own benefits and disadvantages. In the first and technically preferred method, a series of page locators, classifiers and extraction agents would collaborate to find, interpret and extract the relevant new information about each device using the Internet as its primary data source. This would then be fed to the knowledge-base via the standard API for importing new instances and potentially even new classes. However, this vision is far from feasible today with acceptable accuracy and it would require significant research and development effort to get the state of the art to a commercially useable point. In the case of the WEB->KB project as outlined by [165] they were able to achieve slightly better than 77% accuracy using a customized algorithm called Sequence Rules with Validation for extracting text from within a hypertext page in a reasonably well defined domain of university web pages. Given the nature of the instance and slot values required for the ongoing operation of the MD-KB, a text extraction would suffice to search and find the specific features and specifications relating to the device. Promising work by [167] from the University of Massachusetts, Amherst combines the information extraction from unstructured text like the hypertext of the Internet with data mining techniques used in searching for patterns of data within a database structure. Combining these techniques means that using data about the unstructured text coupled with the extraction of important words and phrases improves the ability to infer and learn from the unstructured text [167]. For the MD-KB system to be viable as a service for commercial purposes, more research is necessary in this area and in particular, experimentation with the automation of the maintenance procedure would be important future work.

The second and more probable method for maintaining the MD-KB for online operation would be to engage the major device manufacturers in supplying XML files of new devices. This method is technically simple, but where the technology may be simplified the difficulties of building consensus and the business case for such a concerted effort

The automated extraction of information from the unstructured hypertext is a difficult task which is itself an active field of study. There is considerable research in the area particularly as it relates to maintaining information repositories and retrieving relevant information for the purpose of decision support. The two methods of maintenance, fully automated or community augmented each have their own benefits and disadvantages. In the first and technically preferred method, a series of page locators, classifiers and extraction agents would collaborate to find, interpret and extract the relevant new information about each device using the Internet as its primary data source. This would then be fed to the knowledge-base via the standard API for importing new instances and potentially even new classes. However, this vision is far from feasible today with acceptable accuracy and it would require significant research and development effort to get the state of the art to a commercially useable point. In the case of the WEB->KB project as outlined by [165] they were able to achieve slightly better than 77% accuracy using a customized algorithm called Sequence Rules with Validation for extracting text from within a hypertext page in a reasonably well defined domain of university web pages. Given the nature of the instance and slot values required for the ongoing operation of the MD-KB, a text extraction would suffice to search and find the specific features and specifications relating to the device. Promising work by [167] from the University of Massachusetts, Amherst combines the information extraction from unstructured text like the hypertext of the Internet with data mining techniques used in searching for patterns of data within a database structure. Combining these techniques means that using data about the unstructured text coupled with the extraction of important words and phrases improves the ability to infer and learn from the unstructured text [167]. For the MD-KB system to be viable as a service for commercial purposes, more research is necessary in this area and in particular, experimentation with the automation of the maintenance procedure would be important future work.

The second and more probable method for maintaining the MD-KB for online operation would be to engage the major device manufacturers in supplying XML files of new devices. This method is technically simple, but where the technology may be simplified the difficulties of building consensus and the business case for such a concerted effort

would require lobbying and influence. This style of maintenance operation would require a standards based approach using device and service providers to build necessary support to create and maintain the core ontology and KB. The precedence for this is seen in the development of the Dublin Core and other large ontologies where a technically knowledgeable community comes together in order to supply fundamental and integrated knowledge to the broader general community. The nature of the manufacturer feed exhibits the principle of network externality, wherein the benefit of belonging to the "contextual knowledge-base" standards consortium would only increase in proportion with the number of active members and contributors. The benefits of higher value added services to specific devices of those manufacturers who were participants, could be quantified and would surely prove a powerful incentive to participate. There are other examples of precedent in these areas, and in particular of the business-to-business industrial consortiums which are formed to promote information sharing and integration. These examples provide further evidence to indicate that this approach would be a viable method of maintenance for such a system. In many cases, these industrial consortiums act as human-powered mediators for the purpose of integrating and resolving knowledge between entities much like the automated mediator found in knowledge engineering between disparate knowledge systems [58]. At this point, the focus of the maintenance has been on the somewhat easy engineering problem of adding, modifying or deleting device components, features and their relations, i.e. subsumption between them. The more difficult issue and perhaps the greatest opportunity, if properly implemented, is finding the appropriate method for maintaining a library of possible contextual potentials that are afforded by the configuration of components. Unless the creation of these contextual applications is somehow controlled, standardized or generated via code, the development is subject to the creative aspirations of the developers of the service or application. A library of models of context adaptation which utilise the KB components is what is needed. This library would be open source and available to the development community through portals like SourceForge. The library would store the models which were enabled by the specific version of the MD-KB, as the classes, relations and instances in the MD-KB would directly support the implementation of the set of context adaptation models in the

specific version of the library. The architecture of Protégé 2000 lends itself very nicely to this community approach to modelling context as the model can be stored in RDF/RDFS compatible formats and validated against the MD-KB through a custom plug-in module. This approach to management of models using Protégé is eluded to by the developers for the purpose of creating new semantic web content, but it is equally possible to re-engineer the solution they propose to implement a community-based maintenance operation for the context adaptation as described above [168].

The final component on the maintenance and operation processes of the system pertains to the need for continued knowledge validation. It is clear that as new component instances, instance slots and whole new classes representing new capability are introduced to the market place, validation of the structure as well as the relations and other constraints will be critical. This is incrementally more difficult given that the contextual opportunities will be represented as feasible models by the development community or perhaps as a part of the launch by the device manufacturers themselves. This two component issue requires that formal assessment of the connection between the knowledge-base instances and the context adaptation models is done on a regular basis. At this point, it is difficult to see where this part of the knowledge system validation can be automated in any elegant way although concepts like specific test questions with identified patterns for answers are feasible but nonetheless incomplete. A knowledge engineer familiar with the domain in question would be able to quickly review the current version of the system using the two visualisati[illegible] to assess the nature th quality. Through the application of an increasing and evolving set of competency questions which are automated into a series of queries, the process of on-going validation can be assisted, but it is not possible to see this how this knowledgeable creation of CQs might be fully automated.

4.8 Summary

In this chapter, the design, development, investigations and results of the new architecture and model for context adaptation is described in detail. It is shown that the issues with dynamic context adaptation can have an alternate architectural solution which involves the use of a flexible and evolvable knowledge-base system. By segmenting the context 4-tuple into dynamic and static components, the context collection phases are shortened and a single source of static context is designed. The nature of this single source of static context was thoroughly analysed and a knowledge-base approach to the storage of the static context was selected and found to meet the requirements for dynamism of mobile device knowledge. The advantages of the knowledge-base architecture include the ability to model more accurately the mobile device context through the use of frames. The use of a frame-based approach allows an extensible model for the situational context of the device which includes both the features and capabilities, as well as the contextual model of further associated information. The tools necessary to make the mobile device knowledge-base a reality were assembled – a frame based ontology editor, an inference engine for reasoning about constraints, a set of visualization tools to assess the nature of the knowledge and a methodology for constructing and validating the end product. In addition to acquiring expertise in these new tools, the process of knowledge engineering was necessary to model the domain. By combining a standards derived ontology for mobile device feature specification with conceptual modelling of context, the knowledge-base was constructed and validated through a three step process. It was very clear that ontology building without validation was a misdirected task as much was learned from the process of validation which led to a better, more robust knowledge system. This was proven through a series of device level validations that were compared against the set of original competency questions that were then accurately answered. As the methodology of refining the ontology clearly showed, it took four full iterations of concept definition, model, relationship specification and knowledge-base implementation followed by the three phase validation to get the knowledge to properly answer the competency questions.

Through the design and analysis phase of the KB development it is shown that the need for inference or reasoning functionality is driven by the need for interpretation of the context parameters themselves. The introduction of an inference engine to establish the contextual potential of the device versus the service is reviewed as it was implemented and demonstrated through the use of an access limited logic language (ALL). The concept of deriving basic device related rules from requirements for the service was presented and an automated mapping was demonstrated and proven feasible although scaling this operation and QNM analysis is left to future work. While this architectural alternative supports the resolution of the original problem statement, a solution for context adaptation within a QoS-regulated environment, there remains several questions that are interesting and unanswered regarding such a networked environment. The scalability of KB-based context adaptation in a distributed system and that of a multi-device context network with fixed context servers is an important future question. The knowledge-base approach provide context to various types of devices which in turn use that context to provide higher valued services to consumer thereby increasing the satisfaction with those devices and providing more revenue for those service providers. This in turn, will provide more context-based applications which may favour certain types of devices. The operational and performance advantage of the new model is not presently demonstrated and it remains for the subsequent chapters to discuss how this approach provides benefit over the message passing approach reviewed in Chapter 3.

Leaving performance implications aside for the moment, the notion of both public and private contextual adaptation rules is observed and presented in this chapter. The concept of the use of a shared namespace derived from the descriptive ontology forming the backbone of the KB is discussed. This shared namespace would further enable the consistent use of the KB for the purposes of services such as context adaptation. One of the issues discovered and left to be addressed in subsequent work is the degree to which dynamic context and device features can be supported in such an architecture. The KB-centric architecture outlined in this chapter relies on the existence of basic static device information as the anchor for the adaptation and it was highlighted that the future of

mobile devices is towards reconfigurability, where the device is composed of a set of general purpose, field programmable processors. This future generation devices receive low level software, perhaps in real-time, that may implement new and necessary functions for the service. In this future environment, the nature of dynamicity will change and with it, the tools and architectures supporting the use of the knowledge of the device may also need to change. As increased system dynamicity is the future of computing, perhaps even within the next five to eight years, any distributed service architecture will need to support the full temporal scale of device configuration where fundamental changes to basic functionality may occur at random, short time intervals.

Chapter 5 C2Adapt2 Architecture, Prototype and Evaluation

Context collection and adaptation as an online, revenue generating set of services presents interesting and complex challenges to the software as service engineer. As previously discussed, the current thinking on request and response capability exchange for context adaptation is limited in its utility within the mobile device services industry. The latency and inaccuracy of the contextual information message exchange creates a serious negative impact on the delivery side of services. To add yet another degree of complexity, engineering these systems to perform efficiently and reliably in an environment governed by a service level agreement requires new methods and techniques. A new architectural model, outlined in Chapter 4, is the result of a search for new methods, tools and techniques that can be used in design and engineering for context adaptation. Machine intelligence is introduced into the service-oriented architecture to facilitate delivery and to reduce the roadblocks to achieving the agreed service level. Service-oriented architectures are the emerging information technology paradigm for business services where the emphasis is away for individual components of functionality and towards the functions as individual, well-defined and accessible services. These services can be bundled as a greater service or can be utilised individually. A concrete example of technology which directly supports the common notion of service-oriented architecture is web services. It is perhaps important to note that from a practitioner's perspective, the notion of web service preceded the further expansive architectural digression into a service-oriented architecture. The fundamental objective of being able to offer a single software service for a fee, online to the Internet customer is shared by both architect and the implementer. Web services are not the only technology that can be used to provide such access to software services and in fact, the history of Internet portals and electronic commerce sites is rich with examples of software services that were constructed to allow a customer or in some cases, a developer to provide a service. The famous "shopping cart" is a good example of a software notion which evolved from a custom individually unique set of developed

components integrated to provide a service. Ultimately, the shopping cart became a ubiquitous service component, available for any website with standard parameters, interfaces and a standard operational workflow. The benefit of using web services, as will be discussed in greater detail in this chapter, is its inherent ability to interoperate in a predictable way. Proprietary services restrict cost-effective access and access is the key to increasing revenue generation through services. The concept of software as service brings with it, a deeper interconnection between business and technology. The ability to select a third party component to plug into an architecture is not new and has been a part of the software development community's processes since component-based software and reusability became the vision of computing. Service-oriented architectures, and in particular web services, extends this metaphor slightly to open up the closed nature of the individual software component into a series of standard protocols and interfaces with XML as the basic language of data exchange and context description at all levels.

This chapter will take the resultant knowledge-base mobile device context architecture derived in Chapter 4, the principles and practice of service-oriented architectures and the requirements of the context adaptation system together to construct a prototype implementation of the knowledge-base context adaptation service. The chapter will also outline additional technology necessary to complete the workflow prototype and will investigate the use of business process management integrated with service workflows as a basic architectural design pattern for the C2Adapt2 system. As previously outlined, C2Adapt2 is the abbreviated name for the KB-based context collection and adaptation service system, an implementation of the abstract C2Adapt model already studied. The C2Adapt2 system is a set of orchestrated and monitored software implemented as web services which is the integrated results from the work presented in the previous chapters. Its primary purpose is to confirm the operation and feasibility of the knowledge-based web services architecture with the repository, MD-KB operating within a service level agreement environment. As will be discussed, the culmination of the research for the purpose of engineering work is commonly a prototype as it provides a standard means for assessing the elements of the programme through analysis of the

implemented technology. With the tremendous interest and increase in development of web services, other forms of validation and verification are now emerging for service-oriented architectures and web services technology. An excellent example of such a verification process is found in [169]. In this method, the verification works in two phases, one in design and one after implementation. The two "viewpoints" which are generated as a result of the phases are then compared to provide a formal model-based verification of the service workflow. In this chapter, the C2Adapt2 architecture will be described through design specifications. A discussion of proper notation for such specifications will also appear in subsequent sections. The notation for describing SOA systems is evolving and there are competing viewpoints as to how to formally represent the systems development process. On one hand, there is a new standards sub-initiative called Business Process Management Notation which comes from the BPMI.org group which describes the services workflow, primarily originating from the business perspective. A second, technically-based notation is provided by [170] who advocates the use of the well-known UML standard for describing systems and providing a modification of the UML profile to accommodate the unique needs of service-oriented architectures. Section 5.1 will look at these two technologies and provide some insight into the comparative advantages of using one or the other.

Before continuing with this next element of the programme it is beneficial to help the reader put this investigation into the scope of the overall engineering programme itself. The implementation of C2Adapt2 is designed to be another step in the programme of work which represents a cumulative validation of the results of the all prior investigations within the programme. This concept of using the C2Adapt2 as a validation of all foregoing work is depicted in Figure 5-1 below:

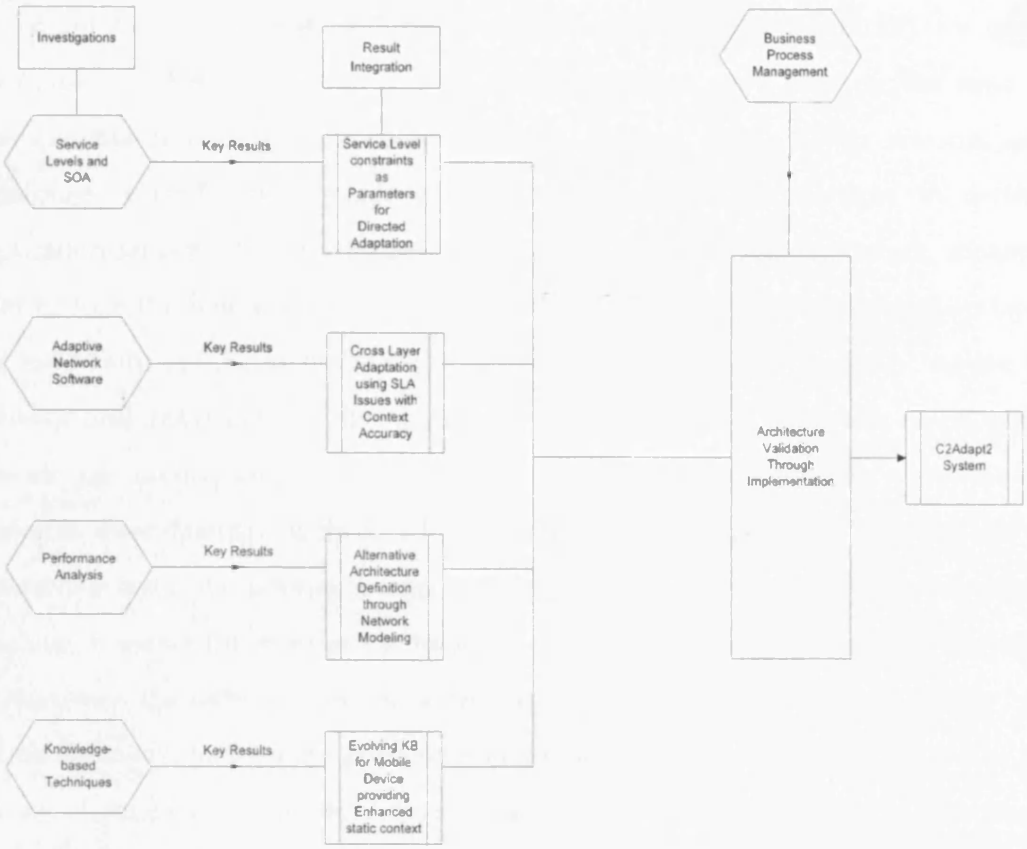

Figure 5-1: C2Adapt2 System – Cumulative Validation through Prototyping

A final point to the introduction in this chapter, is the justification for the use of web services technology as the basis for the architectural implementation. Throughout this programme of research, the central notion is of revenue generating services to the online customer. Inherent in this fundamental principle of the work described, is the notion of a ***requestor***, a ***service provider*** and from the work in Chapter 4, a ***centralised repository*** providing discovery and adaptation services that enable the delivery of enhanced applications and content. In future contextual adaptation, the vision is of machine-to-machine adaptation on behalf of the user of the service. Although this is not currently achievable as was demonstrated in Chapter 3 under the constraints of a service level agreement, the expectation is that through proper engineering the distributed network of intelligent computational resources will be able to perform complex adaptation and delivery of optimized content. Given both the roles outlined above and this future vision, it is logical to select web services (WS) as the technology foundation for implementation as it closely follows the tenets of service-oriented architecture or SOA,

its variant of semantic web services which is being designed specifically for use in machine-to-machine interaction and its WS foundations were built on the need for interoperable technology for online business services [171]. Web services as a technology extends the concept of network communication services to network application services through the creation of a new enhanced network stack, sometime referred to as the web services stack [172]. In this stack, the network and transport layers are essentially optimised for the purpose of carrying the web services. Above the network and transport, are the packaging, description and discovery layers which provide the unique web services facilities. Description and discovery are the two protocols most familiar to those who have read or developed web services. In the descriptive layer, the developer uses a descriptive language to specify for the generic machine, audience the features and functions of the specific web service being deployed. In discovery, the web services are categorised and a directory of the web service basic details are maintained for the purpose of generalised lookup. There are both public and private directories of online services to which a company or developer can subscribe to in order to market their deployed services. This represents the new technology-based marketing channel for a service-oriented business model [89].

Also in this chapter, the C2Adapt2 overall requirements will be restated and modified to include the semantic nature of the system. From this starting point, the details of the model-based design process will be presented along with a discussion on the concept of automatic composition of the workflow. The architectural design and BPEL4WS process model will then be presented along with analysis of the resulting service-oriented architecture using the MSC-LTSA tools [169]. A discussion of the tools and methods for engineering the resulting system will follow and from there, a detailed description of the implementation of the C2Adapt2 architecture will be presented. In the conclusion, the operational nature of the mobile case study will be the focus of the discussion using the information provided by the verification process.

5.1 System Architecture and Process Design for C2Adapt2

The operational proof of concept system architecture is depicted below in Figure 5-2. The system uses the web services distributed computing model for a service-oriented architecture as was previously mentioned. The mobile client initiates a request for rich imaging service, called "Photo Album" which is registered with the context aggregation or proxy server. The context aggregator plays a dual role as the entry point for the user service request and later as the component responsible for bringing the static and dynamic elements together and presenting these to the evaluation engine for the purpose of devising the rules for the adaptation engine. This is an extension of the context aggregator's role is to make choices about the extent to which elements of context are available for evaluation. The first step is to decompose the higher service request into the individual context components required and to pass these to the dispatch service. This dispatch service effectively distributes the retrieval of context between the dynamic and static elements. The dispatch service is responsible for the remote method invocation of the various get methods for the context. For example, the dispatch service will issue the *getStaticContext* method against the MDKB knowledge-base passing in the device identifier and issuing the appropriate Algernon query. As depicted in Figure 5-2, the context aggregator will also be responsible for dynamic context elements which will be requested with a time to live style parameterisation. At a minimum therefore, in this architecture the extent of adaptation will be that provided by the static elements found already in the knowledge-base. If for whatever reason, the dynamic context is not acquired in the required time, the evaluation will proceed based on static elements alone. In this way, the service can still meet the specified service level in terms of delivery; however the quality of the delivery will be compromised to some degree.

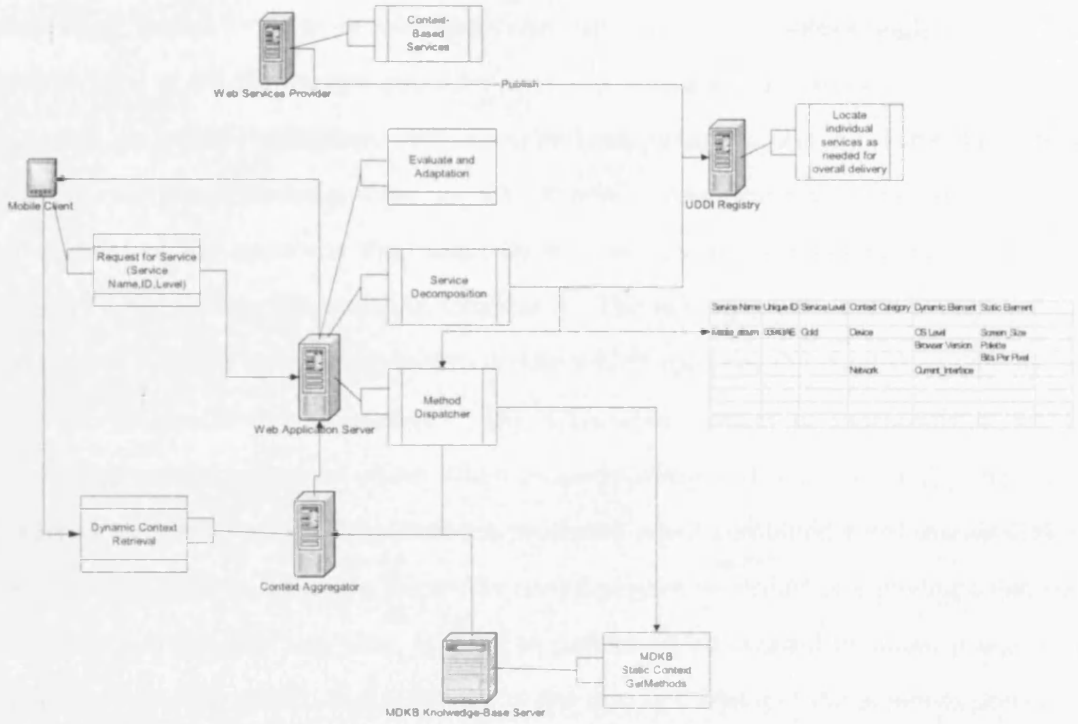

Figure 5-2: Context Collection and Adaptation System Architecture

The continued improvement in device attribute reporting, particularly that which can be envisioned in the reconfigurable device of the future, will mean that this architecture will scale well to the state where the device reports its own capability accurately and in a timely manner relative to the service level sojourn time.

As with the context collection phase of the system, the evaluation phase is broken into a series of web services for convenience and openness of the system. The application server, in possession of some variable degree of contextual elements, will call the evaluation service to determine the possible and most beneficial adaptation. A set of rules are then passed to the adaptation engine with the pointer to the variables and the adaptation is performed. Once complete, the resulting service at the user request level is available and a pointer can be dispatched to the client to initiate the receipt of the entire service with the content, navigation and interaction mapped to the device, network and preferences as designed by the service provider. The openness of the web services model means that as new services become available for such things as user preference retrieval from a private repository, these new services can be selected through the

dispatcher based on the service provider updating the context matrix. In this architectural style, the service provider uses the available web service components to compose the context collection, evaluation and adaptation system by using the context matrix and the knowledge-base as its principle specification. The service level component of the matrix is then used by the underlying QoS system to operate the network layer as was discussed in Chapter 3. The management of the overall service then is provided in one management console which specifies the context as well as the network service level parameters. The C2Adapt2 system is essentially a set of interacting service processes which when properly composed will instantiate one online business service offering. These service processes when combined form one workflow which is designed to achieve a hopefully useful service on behalf of a paying customer. The term workflow in this case, is used to denote an automated business process. It should be carefully noted, that although in the case of C2Adapt2 the business process is composed of a set of interacting services, there can be a one-to-one mapping between process and service. A key difficulty in service engineering is the proper decomposition of an overall service into well defined, efficient sub services. This is a familiar issue found in other software engineering areas, such as object orientation where the "magic" is in the skilled definition of the reusable objects extracted from the requirements specification. By defining C2Adapt2 as a workflow system, several conceptual benefits result during the design phase. First, each constituent service is an independent service which can be upgraded individually without affecting the overall service. Second, most workflow system environments provide a separate service management abstraction layer which allows each business service to be monitored. Specifically, this means that a separate service manager is provided as a part of the workflow in order to monitor and manage the operation of the C2Adapt2 service. This provides the necessary structure to implement a QoS environment. By providing the service level rules to the service manager, the operation of C2Adapt2 service can be managed to meet the service level which is changed via a message to the service manager and will not require changes to the individual functional services such as context collection and adaptation. This is clearly beneficial for an environment in which the user can modify their service level dynamically based on the needs of the moment. Lastly, the combination of web services

technology with an automated workflow system provides several standardized tools and methods for design, development and verification which are a critical component of successful engineering of production systems.

5.1.1 Process Design

To achieve the proposed service-centric design, it is worth briefly reviewing the nature of service engineering and establishing a terminology for the technical concepts in this area. Services, processes and workflow are variants of the principle of achieving work through a coordinated set of activities. No discussion of these principles in the context of a computing model is complete without some reference to the notion of process calculus or π-calculus as the formal basis and proof for the conversion of designed process into computing and communicating systems. There are numerous references to the definition and later implementation of π-calculus throughout computing theory literature. The use of a formal foundation for web services languages is a subject of some controversy due to the initial commercial interest in web services. Many WS tool companies are in fact using such formal computing theories such as process algebras to indicate that their tools or method which are supposedly based on formal methods are sound and perhaps better than others [173]. While this is an interesting and lively debate, the importance of a formal foundation for web services systems has yet to be proven worth the expense in commercial systems. The degree to which process algebras specifically and exclusively provide a means to create software systems which perform better, meet specifications to a greater degree or improve overall documented quality is still largely undocumented. However, a key component of service engineering should be some definition of the process for verification and validation between the design specification and the resultant implementation. In this chapter, the π-calculus is used as a reference point in the design of the process via the tools selected. It provides some level of assurance that the computing theory behind the process model language and representation is adequate to describe the patterns found within the C2Adapt2 system. The choice of π-calculus is sufficient given it was designed and is currently used to design modern computing system as it provides the means to ensure the logical operation of both the computational components and the communication channels in a

local or distributed environment. Using this process model approach to the C2Adapt2 system also requires some attention to the nature of the actual composition of the service components. There is a tremendous effort in academic and industrial research organizations to attempt to automate the composition of web services from some initial user request [171], [91], [68], [92], and [174]. While this automated composition of services is an important area of future research for context adaptation, it is beyond the scope of this work although in the subsequent section there is a discussion of what semantic requirements and process modifications are required to move to this new state. The C2Adapt2 system is a pre-defined service bundle composed of a set of atomic services, each engineered to perform a set function. There is nothing though that would stop new versions of these services from being dynamically discovered and inserted into the overall process. This is a part of the overall requirements for C2Adapt2 from the non-functional requirement around maintainability. The key difference though is that the composition of C2Adapt2 is pre-formulated by a designer and implemented by an engineer to run in a specific sequence with a priori knowledge of the operation desired.

With the basic process design background presented above, the next step is to describe the process design and modelling activities. After a review of the emerging methodologies in the area and standards support, it is clear that the diversity of standards promotes further complexity in engineering when it comes to the modelling and design stage. The decision to work within a process workflow environment directed the initial evaluation of several modelling methodologies. From a traditional software engineering approach adapted for service-oriented architectures as described in [170] to the business process modelling for early workflow systems as found in the Workflow Management Consortium's XML for Process Definition Language (XPDL), there are a multitude of approaches to creating models of processes. Some of these approaches are business oriented and some are variations of more technical software engineering processes modified to improve the readability and accessibility of the information to the business analyst. There is considerable overlap as well between languages and representations for process modelling and those for execution of the processes. The complexity increases further when these modelling languages each have a specific

documented "framework" or paper instantiation of the language which is then different from the development of the tool which allows the developer to actually utilise the language in the modelling process. Industrial competing interest abounds in this area which causes a contortion of the process towards one particular set of standards that support the representational environment conducive to the most powerful and influential entity involved with the standards process. This further complicates the choice of a path for modelling and designing such systems as it takes considerably more time to assess whether the claims for each language, representation and tool are as they are represented to the development community. After a review of BPMN, UML (adapted for workflow), BPEL designer tools, and more traditional object oriented modelling approaches, it was clear that no single modelling construct would suffice for this work. This was due to the nature of the modelling processes themselves. BPMN looks at the business process at the highest level within the business and successively drills down into the lower levels of the process providing the ability to translate to the technical level. UML, on the other hand, due to its origin and extensive use as a technical tool, starts with a fine-grained technical approach which can be generalized to provide a business-oriented process view [170]. The two leading modelling approaches, BPMN and UML, when compared against the normal benchmark of process patterns [175], were amazing equivalent [176]. There were very few significant differences and none of the differences noted were relevant to the domain of the C2Adapt2 system. Both methods provided roughly equal expressiveness for the problem of modelling the system and both had relatively equal tool support. Given this observation, the idea of using a modelling approach inspired by [169] was developed and the resulting concept for the C2Adapt2 modelling process is depicted in Figure 5-3:

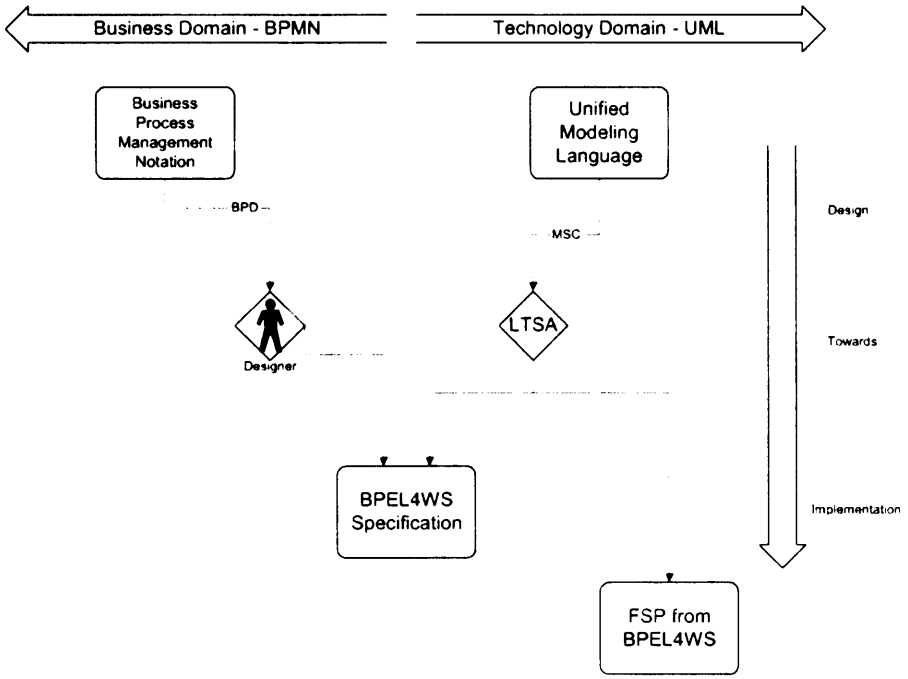

Figure 5-3: Integrating verification through process modelling validation

The result of the process outlined in Figure 5-3 is the development of two versions of the BPEL4WS specification for direct comparison, one taken from the business perspective and one taken from the more technical UML modelling process. At this stage in its development, the BPMN to BPEL4WS process mapping is not an automated one, but requires the application of a set of mapping rules to the BPD [177]. The BPMI.org consortium has publicly stated though that it will be producing such an automated mapping tool in the near future so that it allows a consistent method of translation from a BPD into the execution space of BPEL4WS. As a minor point, it is noted that strictly speaking in UML, it is a sequence diagram which appears as a part of notational framework and not necessarily the MSC. It is generally agreed however, that the sequence diagram within the UML specification is a modification of the more traditional MSC and therefore is functionally equivalent for the purposes described here. The results of the development of the C2Adapt2 MSC is depicted in the LTSA tool in Figure 5-4, this is the first step in the design of the prototype.

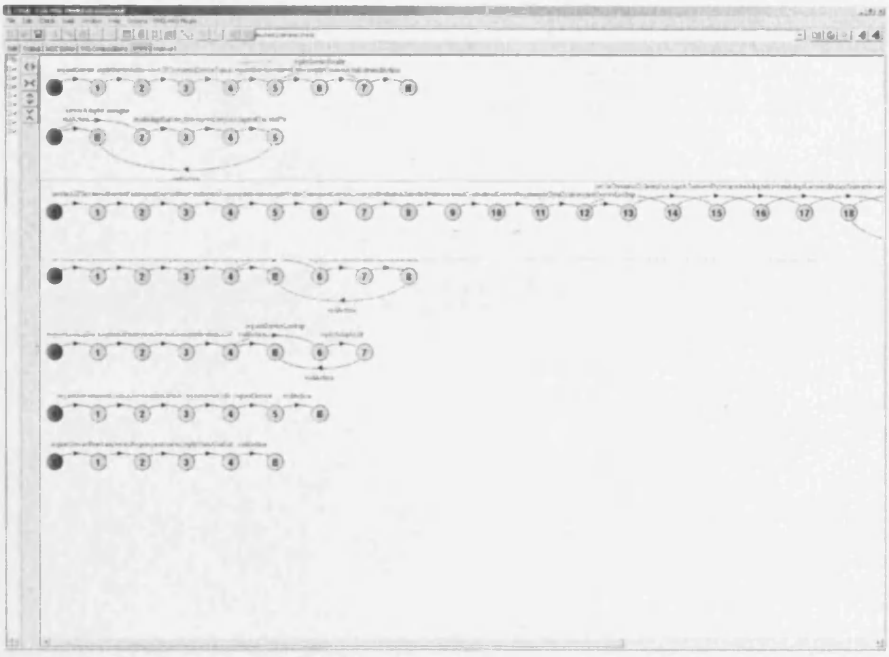

Figure 5-4: C2Adapt2 LTSA FSP Diagram of Compiled Model

The C2Adapt2 MSC model is then compiled in LTSA which provides a summary of the possible states for each service in the workflow. The resulting LTSA model compilation for the C2Adapt2 system can be found in the Appendix B of this document. The next step in the investigation is to take the same service specification which was resolved into the MSC and create a BPEL specification. The BPEL specification is then input into the LTSA BPEL4WS plug-in and another FSP is then generated. This second FSP is the compared to the first and differences are noted. Given the differences are not deemed significant by the systems engineer responsible for the design, the resulting BPEL service can then be categorized as having a formal foundation that maps directly to the design specification.

5.2 Service-Oriented Architectural Analysis and Design

The creation of the proof of concept system that demonstrates the feasibility of the knowledge-based context adaptation architecture requires that potential future architectural models be blended together efficiently. Although the entire prototype can be constructed in a highly efficient, closed system approach for example as a custom client-server, C++ application, this would defeat in many ways the overriding

engineering objective for the system. The list of summary level requirements is given below:

- Provide open interfaces that are accessible through the public Internet and which cover both the definition of how to access the functions as well as what the function will actually perform
- Ability to perform parallel queries for capabilities and requirements in a distributed, asynchronous manner
- Independent components which can be invoked without precursors
- Demonstrated openness and extensibility into mobile platforms and various platforms even to include future anticipated platforms of interest such as the GRID
- Transactional mechanics and characteristics to provide services which are interdependent to execute in a predictable way with knowledge of state
- Error handling integrated into the execution of the services both sequential and parallel, these aspects would also need to be interoperate with the transaction mechanics
- Implement service composition between service endpoints without the need to independently call each module; must implement a standard workflow based on the most likely scenario

The concept of using semantic web services as the basic architectural style for the prototype system allows the investigation into how a mobile client might request the imaging service and have the sub-services selected based on the semantic decomposition of the service request into service capability. This style fits precisely the needs of the demonstrator as outlined in the scenario. The semantic web services extend the nature of web services to include the ability to define what the actual web service achieves as well as describing it implementation via the SOAP and WSDL standard specifications [69]. This extension of web services means that services can be located based on their ability

to deliver specific elements of information back to the requestor, for example the invocation of the dynamic context service based on the specific elements which are needed in order to complete the higher order service request.

As the basis for the workflow prototype, the web services model provides many advantages. The openness of the web services technology as provided by the standard web service technologies – SOAP, WSDL, running over HTTP all essentially XML structured description languages. Web services as a set of technologies also allows the integration of the Protégé 2000 MDKB implementation into the services architecture though either direct query using Java with WSDL service descriptions and implementations or also through the use of a DAML extract from Protégé which can be used by a specific web service whose implementation would be a DAML analyser. Modelling each of the context subsystem components into the web services model is straightforward although there is an additional step of adding the necessary supporting components for the context subsystems, items such as the service request analyser which would determine the sub-services that would then go acquire the context. The use of web services through the proper implementation of composition which is outlined further below also supports the execution of services in serial or parallel and asynchronously as stated as a requirement. Using additional and auxiliary specifications to the core web services standards, such as WS-transaction and WS-Coordination, the requirements for atomicity of transactions and the integrated error handling can be achieved. These additional technologies are not as yet standards as are the core, SOAP and WSDL but they are specifications developed by Microsoft, IBM, and BEA Systems who each have strong software portfolios in the web services area.

It is not enough, however to simply locate the necessary context services but the next step is to orchestrate them in order to meet the overall service request. This is service composition. The service composition in the case of our imaging context example, will be the vehicle that brings the individual web services into the right order and ensure that the response to the services are available to those services which need them. This goes beyond the notion of sequential ordering of components with flow control into what is

now termed "orchestration". Traditional or non-semantic web services research has provided industry with several new terms for the autonomous hierarchical ordering of services into a meaningful delivery service. The term orchestration has to do with the ability for these individual services to be brought together in many cases to deliver on a complete business process [90]. Additionally, there are web services which are said to be "choreographed" as opposed to orchestrated. In web service choreography, the concept is to bring together services as they might appear to one of the key actors in the overall integrated process. Effectively using the analogy of the choreographed dance where each dancer knows the specific routine and then they come together to form the overall ballet for example. These terms while emotive and useful from a business perspective, are not indicative of the nature of how the web service might come together in order to achieve a specific service delivery. In reality, there are two styles of service composition, static which is the pre-programmed integration of web services in order to achieve a defined outcome. The second and more interesting style although study of it is beyond the scope of this work, is the dynamic services composition which uses the combination of semantic description, service definition and artificial intelligence to bring together the set of services which will **most likely** deliver the optimal service based on the service request and the request context.

The mechanisms and flexibility of the composition of services is critically important to the context workflow demonstration and to context-based systems in general. With the ever growing and dynamic nature of mobile applications as has been discussed, the context collection will change radically over time so in the near future, service composition must be as dynamic as possible in order to extract the high quality contextual elements for the adaptation. The ability to bring together service components from the best suppliers will improve the quality of the overall service delivered or in the case of an information acquisition service, it will improve the quality of the enabling information. From Chapter 4 and the results of the QNM investigation on the performance of such systems, it is clear that slight improvements in the accuracy and efficiency of the context subsystem components, now being modelled as services, will result in better adherence to the service level agreements and a better overall quality

of service. Therefore, the selection and composition methodologies for these services which will result in their enhanced operation are important to the success in the final prototype. Selection and composition are not distinct but are tightly interrelated. Selecting specific component services to compose a higher level end-user service will affect the way in which the overall service is composed [91]. Methods for composition extend from formal methods as in the work of [91] to the more industrialised approach as proposed by IBM in [[178],[179]] and to the standards bodies such as Business Process Execution Language for Web Services (BPEL4WS) [76], BPML [180]. The decision on how to implement the composition of the context adaptation service is made more difficult by the multitude of confusing "standards" that are actively promoted in this area. In light of this variety of competing views, it is essential to focus on the central thesis for the prototype itself. The purpose of compositional mechanisms for web services is principally to add state and organisational constructs to a set of stateless technologies. There are two requirements imposed on the composition, first the composition method needs to express the operation of all of the aspects of the context collection, evaluation and adaptation system. Second, the compositional method needs to have an accessible and available set of tools that could be used to build the prototype. The standards of BPEL4WS, XPDL, and BPML were compared through the use of design patterns which express different service behaviours and by evaluating the available software tools for development of compositional web services [181]. BPEL4WS provides sufficient expressiveness as determined by the extent of design patterns supported and appears to have a strong following of industrial companies and developer support. This strong support manifests itself in a number of software development tools and toolkits from commercial and academic sources. It is important to note that the implementation of the composition is not in BPEL4WS as such, but is in fact, modelled by the BPEL4WS language which has been developed within the standards body. In several respects, the BPEL4WS language follows on from Web Services Flow Language (WSFL) from IBM and XLang from Microsoft, but incorporates a set of second recommendations and updates from the developer community [182]. For the prototype work in this chapter, the challenge is set out as an engineering problem so the use of available tools is an essential part of the solution. In the subsequent section on tools this will be discussed in

detail, but it was observed that regardless of whether one believe in the formal underpinnings of the process model, the BPEL4WS approach maintained a strong lead in the array of and maturity of the tools which were available to the developer.

The context adaptation web service system, and its prototype implementation C2Adapt2, were modelled using the business process execution language using the BPEL4WS. The specifics of this design model are discussed in detail in this section using the semantics of the web services technology domain. It was shown above that through the use of message sequence charts; a designer can indicate the general operational flow of the context adaptation system including the important sequencing exchanges that enable the appropriate adaptation to take place. This sequencing and the time constraints of the completion, impose the state and flow requirements onto what would otherwise be a system which could in theory be stateless. The orchestration of the services is then required to achieve the sequencing and also to provide the facility for allowing the service to partially complete in order to preserve the service level agreement terms.

5.3 Tools and Methods

There are an overwhelming number of tools that a developer might choose from when developing a web service-based system. The software behind all these different tools though is fundamentally XML as the base framework of structured data and simple object access protocol, or SOAP as the protocol for service message exchange. The full range of integrated development environments (IDEs) exist as they have done before in support of every previous programming paradigm. There is the lightweight, more experienced programmer's workbench style of tool all the way through to the graphically oriented fully integrated development environment which often has extensive code generation capabilities. The original MANIAC investigation which was the subject of Chapter 3 was demonstrated using an early version of Microsoft's .NET framework which is an example of the latter style of an IDE. This style of programming environment is beneficial as a starting position for many who are experimenting with the coding and the basic understanding of the technical operation of a new technology.

However, there can be difficulty in understanding the actual nature of the integration of the various components which hinders debugging and performance analysis in complex systems. In light of the fact that the initial work was done using .NET, it was worthwhile to consider other alternatives from other vendors so that a comparison and additional learning regarding the engineering of such system could be made. The following development tools were reviewed through evaluation kits provided by the vendors:

Microsoft .NET – as mentioned above a fully integrated suite of tools including new variants on existing programming languages. As mentioned above, the version 1.0 tool set which was released in 2002 was used to build the first context adaptation engine system within the MANIAC research program. The tool has improved considerably since that time and now supports over 20 different programming languages including Java according to Microsoft. The development environment integrates well with Microsoft's Biztalk process server and the XLANG process specification language. It does in theory support the later integration of XLANG and WSFL, namely BPELWS4J although it is not clear the level of commitment to the process composition tool from the evaluation. There was limited demonstration or sample code available on this topic from Microsoft and it was unclear how extensive this support is and how the Biztalk server 2004 which is just now being released actually supports the standard. While the Biztalk 2004 server supports the BPEL standard, the process specification becomes a .NET assembly rather than a .bpel file which means it is tied to the Visual Studio environment and does not meet the requirement for openness as specified. The orchestration engine in Biztalk is however, impressive with its visual ability to design assemblies of processes and compose these assemblies using sequential or parallel component services.

IBM developerWorks/AlphaWorks – The IBM Emerging Technologies tool set, or ETTK is used in the design, creation and deployment of web services and includes several independent software applications built on Java, although C++ conversion to web services is also supported [183]. The tool set is primarily based on the following:

- Eclipse integrated development environment
- Web Services Development Kit
- WebSphere Application Server 5.0 [184]
- IBM Emerging Technology Toolkit
- BPWS4J for support of Business Process Execution Language (BPEL) for the purpose of orchestration

The Eclipse environment provides the integration point for each of the IBM individual web service tool components. As with the other environments, the programming is done in a Java IDE, the web services components are then created or generated from the implemented logic. The programming can be in various forms, EJB, a Java class, and existing web service either locally or on the Internet. The generation of the interface WSDL is done with a plug-in and the Axis SOAP server is embedded within the environment as well. All tools were downloadable under a full evaluation license or open source license from alphaworks.ibm.com as pre-production code. The IBM toolkit uses the Eclipse plug-in architecture to allow the addition of the business process and workflow management software such as the plug-in for the BPELWS4J compliant tool, BPWS4J. Although one of the most independent and time consuming environments to build, the IBM tool set allows the developer to move through the web services environment to design, create and deploy services with separate tools, all within the one Eclipse environment which is a benefit. The complexity of the tool can be reduced with this approach of a single development workbench. The tools are well-documented through the developerworks.ibm.com site and there is some limited technical support through the forum and developer community. The IBM approach is very much towards "on demand services", composed in an orchestrated manner, the company has considerable investment into making their software portfolio both compatible with and enabling of the service oriented architecture provided by web services.

Collaxa – BPEL Server and BPEL Designer – In this suite of tools from the independent software manufacturer Collaxa, their BPEL Designer acts as yet another Eclipse plug-in integrated nicely in the Eclipse IDE and provides an excellent range of functionality. The BPEL designer provides the ability to model graphically the business process, although the build process on a Windows platform requires the use of command environment programs which is an inefficient process for the developer. Once the process is designed in a manner which ensures it adheres to the BPEL standard language constructs, the process results in a .bpel file and a set of WSDL files each associated with the services within the overall process. This can be directly deployed through the BPEL console into a BPEL server for testing and deployment. The designer tool runs is straightforward to install and use. The server console operates as a browser application and manages all the dependent components necessary to deploy the web service.

Systinet WASP Server/Developer – Systinet provides a web services server implementation integrated with its developer front-end suite of tools in much the same design pattern as other available tools already mentioned. This suite of tools provides both a Java and a C++ application server for the deployment end of the web services which is somewhat unique in the marketplace. The process development tool is a slightly more traditional integrated development environment relying less on a visual process design and more on the developer understanding the components of the web services that are required. The Systinet Java developer is also an ECLIPSE plug-in, again a familiar pattern for the tools in this technology domain. As with Collaxa, Systinet provides tools or components for each layer of the web services technology stack and allows the deployment of services directly to the server with publication into a UDDI directory [185]. The WASP server and developer (now renamed to the Systinet Server and Developer) is somewhat unique in its more industrial support for transaction processing

The toolkit which fit the needs of the case study and project was Collaxa BPEL Designer and Server based its ability to implement the standard, provide an integrated suite of

applications and on its support for the developer. This is a topic which will be covered further in the next section. Beyond the toolkits and software needed for construction, there are the best practice methods that can be associated with web services development. While these methods are still emerging, some general approaches are common. There are two slightly different approaches to the methodologies that are necessary to implement the workflow prototype. First, from the business perspective, the methodology for business process development using BPMN and a BPD to describe the overall business process and the subsidiary component processes, is one way to create an integrated and orchestrated application out of the individual web services within the high level mobile imaging service scenario [176]. The second method is the more technical fine-grained method by which the individual web services are developed starting at the individual services level and then the workflow is composed to create the higher level service offering. In the former method, the higher level process is developed and modelled using a standard business process language, often based on the earlier mentioned process algebras. From there, each composite service is then technically designed and modelled using tools more suited to software and systems specifications such as UML. There are variants of these methods designed as a compromise depending on the technical needs, environment and developer preferences, this leads to the four approaches outlined below:

Green Field – this approach assumes that there is no existing service or component from which the design and/or development can be derived. The green field method starts with no a priori knowledge, proceeds to the development of the component, and to definition of the service interface. Once this step is complete, then the publication of the service through the service interface and the deployment of the components can be done. Deployment of the web service is far more complex than a simple publication of a class on a SOAP server but requires that the SOAP description is generated and deployed as well as the components of the service itself. Once the code is deployed, then the service implementation which defines how to access the instance of the service itself is deployed. This includes the WDSL description which defines the mechanism for

binding to the service. This definition is usually generated on behalf of the developer and is published through the UDDI registry either locally or on the Internet.

Top-Down – the top down approach starts with a previously describes service interface and then generates a skeleton or proxy from this interface. The developer can then "flesh" out the details of the new service on the basic framework of the existing interface and be assured that it will operate to specification. The top-down approach will then proceed as the green field through the phases of implementation definition, deployment and publication.

Bottom-Up – in the bottom-up methods, an existing class or component such as a Enterprise Java Bean (EJB) can be used as the basis for the generation of the service interface itself. Tools such as Java2WSDL can take the details of the class or component and create a service interface in WSDL. The process can then run as before in the top-down, where the service interface is used to create a proxy of service and the details of the program are left for the developer to implement.

Meet-in-the-Middle – a hybrid approach as the name implies, the meet-in-the-middle uses an existing service interface as with the top-down approach to generate a template for the new component. The developer will then use a mapping technique to adjust the methods or functions to the new desired state rather than re-developing the code as in the bottom-up method. The deployment and publication phases are identical to all previous methods however.

Given the context adaptation workflow requirements as specified, and although there are existing components of a similar nature available from the initial investigation, the new architecture for the subsystem is substantially different, enough to force the use of the green field approach. The previous components were created with the specific business process in mind and reflected some of the process logic within the code. This can be alleviated by using the two phase approach of redesigning the components within the framework of the new BPEL compliant process model.

5.4 Mobile Device Case Study

In order to assess the newly devised context adaptation architecture and a demonstration of how all the pieces might fit together in a web services environment, a simplified case study was designed involving mobile devices such as mobile phone requesting a photo album service application using the user's own image assets. The difference in this scenario from the one mentioned earlier in Chapter 2, is that this system is now a workflow of web services in an open services market. The open services market allows the possibility of a future automated third party service selection and incorporation into the workflow. This third party integration through discovery in the scenario is essential to the future service oriented marketplace and its revenue potential. It is also important for the quality of service monitoring implementation, as the use of a third party monitoring service will provide the consumer with the necessary SLA assurance.

5.4.1 Use Case Scenario

The use case outlined in Figure 5-5 depicts a single request for a context adaptive premium service from a mobile device through the public Internet. In this scenario, the service agreement is an actor whose function is to monitor and record key checkpoints in the workflow of the delivery of the service.

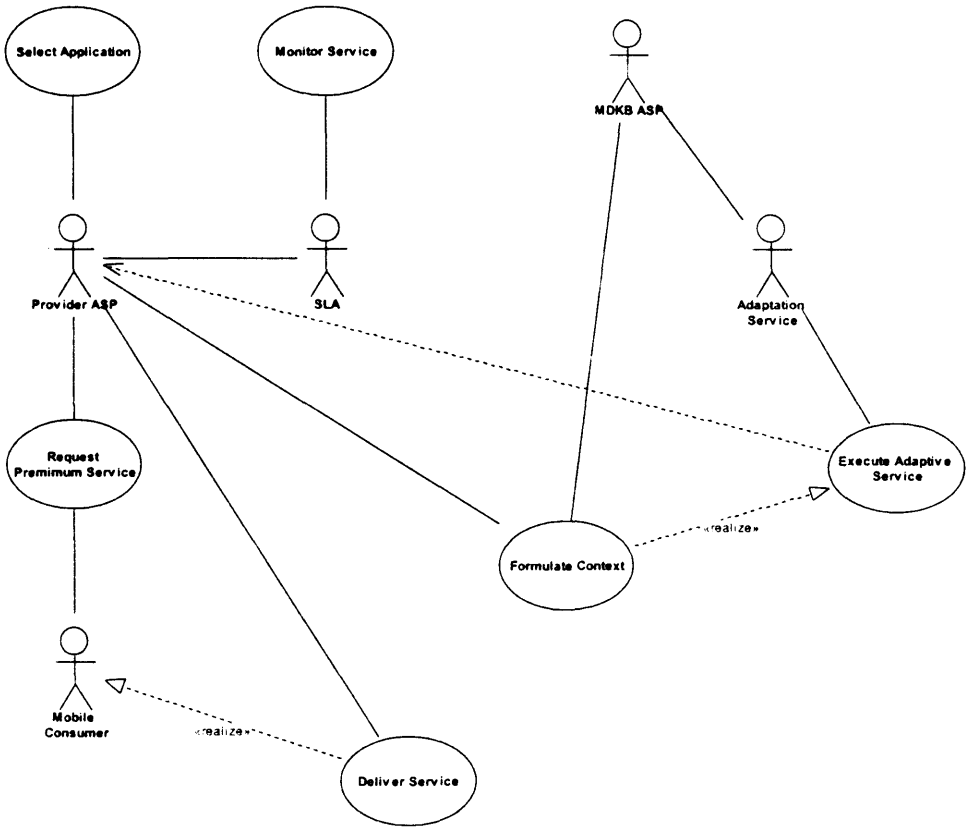

Figure 5-5: C2Adapt2 Mobile Device Use Case

This SLA service can be a separate trusted third party, an outsourced service of the ASP/ISP associated with the user or, in some instances, it is feasible that the application service provider could provide its own service level monitoring which would simplify the network message exchange.

5.4.2 Sequence Diagram

Earlier in this chapter, the use of labelled transition states was discussed as a method to verify the operation of complex web services workflow. It is now appropriate to review the use of sequence charts or diagrams as the primary starting point in the service design process. The use of message sequence charts or diagrams provides a convenient visual mechanism for specifying web services, which are fundamentally technology based around stateful message transfer [186]. The C2Adapt2 system is modelled using both a basic and a high level message sequence chart in the LTSA tool as the first phase of the prototype. In Figure 5-6, the high level chart is shown. The init process marks the

beginning of the overall process. Service initiation is the first execution step which bundles the process of the user service request through the application service provider and from the application service provider into the directory. In this flow, the next step is to collect the static context from the MDKB as well as collecting the known service requirements. These results are passed to an evaluator service which specifies the degree of dynamic context to collect. The dynamic collection is initiated through the application service provider and all results are joined for evaluation at the evaluator. The evaluator process builds the instructions against the context and provides that through the ASP to the adaptation engine service. The engine then performs the necessary steps of altering image assets, deploying new software or reformulating layouts and stores these components of the adaptation. The adapted application and content is then delivered to the client again through the application provider as either a component or a pointer to the appropriate assets. The flow itself is different than the original MANIAC system flow as the web services orchestration capabilities allows a more flexible set of operations to occur in a distributed fashion.

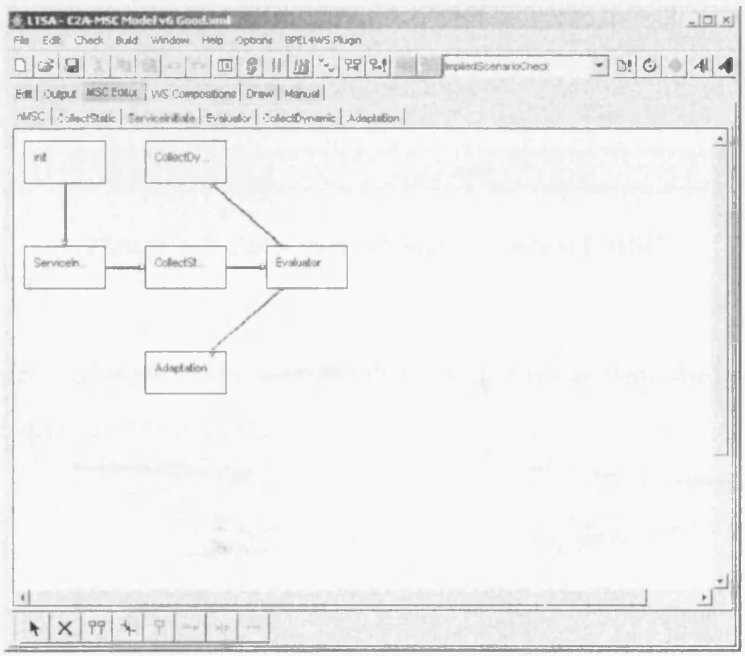

Figure 5-6: High Level Message Sequence Chart from LTSA tool

Each high level message sequence chart is further described at the lower level by using the bMSC tool found in LTSA. In this tool as shown in Figure 5-7, the actual message passing is seen between the actors. In the bMSC, the directory is explicitly depicted for purposes of clarity. In the workflow service prototype, the use of the directory services to lookup available web service components is implied as a basic function of the web service architecture. This means a separate directory actor is inherent in the architecture and is not needed in the process.

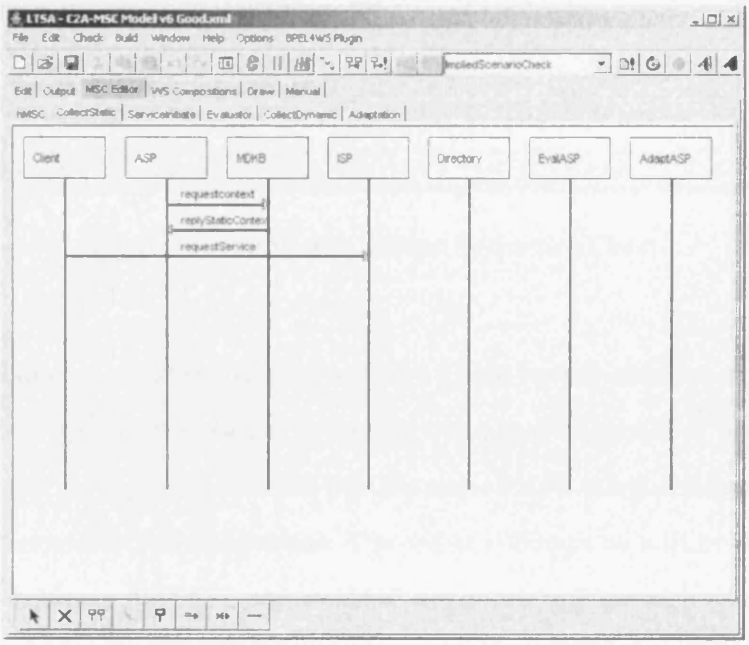

Figure 5-7: Collection of Static Context bMSC

The more detailed diagram of the service initiation process as described by the message sequence chart appears below in Figure 5-8.

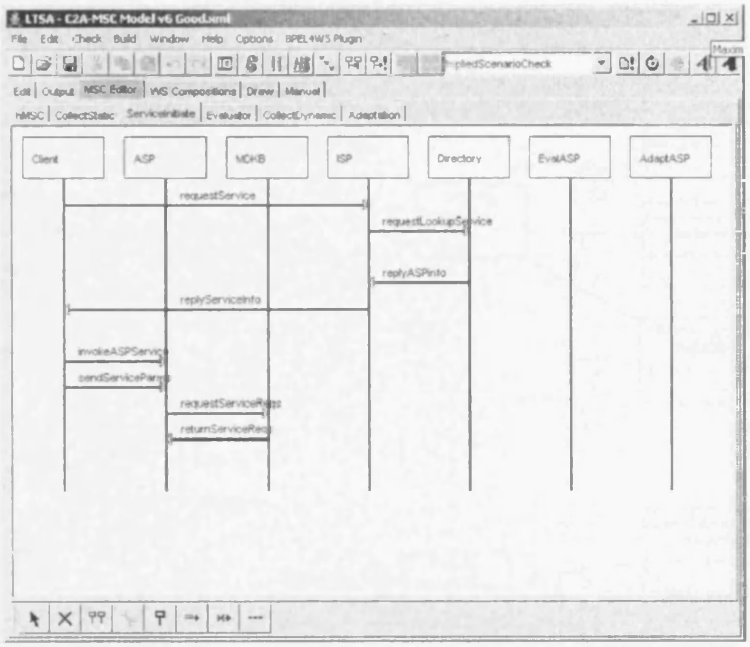

Figure 5-8: Service Initiation Sequence Chart

The message sequence chart is useful as a design guide for the interface definition of the prototype. Even without the timing or control components specified, it acts as a good starting point. By packaging the classes into the same bMSC categorisation as appears in Figure 5-8, a framework for comparison is possible although as will be discussed later the differences between the BPEL and the MSC are quite significant. In parallel with this process, the design of the components of the system was also done. This ultimately led to the development of the individual services and in some cases; these individual services resulted in additional classes or in a few cases, additional sub-services.

5.4.3 Class Diagram

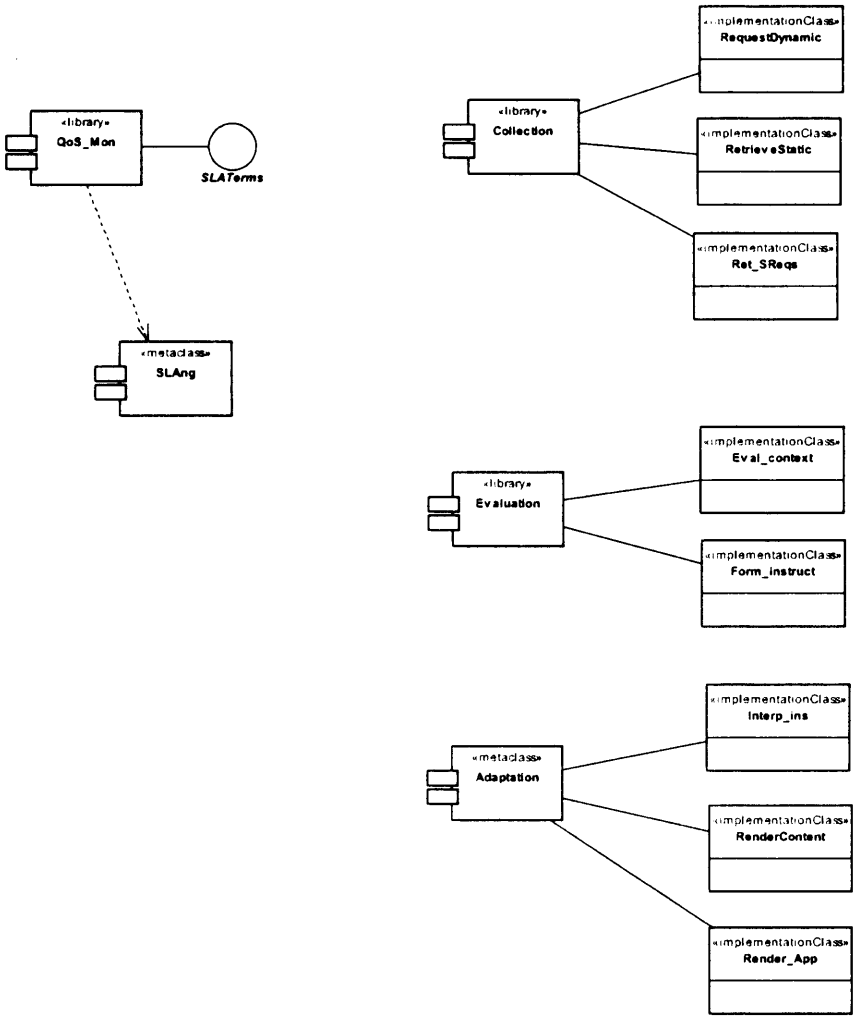

Figure 5-9: C2Adapt2 UML Static Structure Diagram

Each major subsystem of the overall process is bundled into a Java package at the source development time. For conversion to the web services architecture, each Java class is then exposed through the creation of WSDL. The exception to this is in the area of the dynamic discovery class which essentially implements a "reverse" Jini™-based dynamic discovery. Instead of the normal Jini™ process flow of the client requesting a service through a server proxy, for the case of the dynamic reporting of context the process will be reversed. The server will then invoke a secure remote method on the client through a stub object. This stub object will then execute a local service which reports back to the server specific elements of context necessary to complete the adaptation. As mentioned

above the Java classes are converted to WSDL through the use of the JAVA2WSDL automated tools in the BPEL Designer toolkit. The WSDL is then used as the base for the BPEL process definition at each process step. Service partner links are created using the actors as specified in the use case scenario as the endpoints and a validation of the individual role in the workflow is made through the use of the message passing outlined in the lower level MSC. The BPEL Designer then is used to organise the flow of the messages between the converted classes and this results in a .bpel file. The resulting .bpel file is then used as input in the LTSA BPEL4WS plug-in which automates the decomposition and subsequent conversion into states. This conversion results in a finite state process diagram which is compared to the original FSP diagram generated through the use of message sequence charts. The key difference should be the temporal nature of the system. The BPEL integration provides significantly more detail then the MSC description of the C2Adapt2 and requires the use of a task manager which is essentially the quality of service manager in the prototype.

5.4.4 BPEL Diagram and Source

The Collaxa, now Oracle BPEL Process Manager™ and Process Designer™, tools were used in both the design phase and in the management of the deployment of the web services. The process manager is responsible for the deployment and management of the web services, there were two orchestrated services deployed. The C2Adapt2 service was composed of several sequences, each matching the original stage of the queuing network. Additionally, a second web services flow was deployed to represent the functions of the quality of service monitoring. This second web service was the QoS monitor and it was based on the Worklist manager web service, which provide workflow constructs to the BPEL product suite. Using the basic design specification of the C2Adapt2 system, it was constructed also referencing the MSC developed within the LTSA tool. The current version of the LTSA tool provides a graphical file in the PICT file format. The BPEL designer does not provide a graphical output function at this point. The size of the processes is quite large and difficult to see so the following screenshots will be used to depict the process as modelled.

Figure 5-10: BPEL Process Overview

Figure 5-10 shows the high level overview of the C2Adapt2v2 with the integrated QoS management component. The partners involved in the service are shown on the right hand side of the main window. The initiating partner is shown on the left. At this level, the tool provides the basic information necessary about the process from the design view. The design and experimentation was performed with several versions of the Java SDK, BPEL Server and Designer and the Eclipse environment itself. The design and experimentation occurred over a 10 month period in 2003 and 2004. The final version of the environment was Java 1.4.2_05, BPEL Process Manager version 2.0.10 and Designer version 0.0.95 from Oracle and Eclipse environment version 3.0. The BPEL source can be seen by toggling the view which is found in the lower left of the main panel window. The source view can be seen in Figure 5-11.

Figure 5-11: BPEL Source View

The lower level detail of the process can be found by select the edit process map function from the overview window. Most of the design work actually occurs at this level or in direct manipulation of the resulting BPEL compliant XML code in the source view. Unfortunately, the graphical output options for the tool as of this date are quite limited to only a view as taken in screenshots. The more detailed process can be found in both Figure 5-12 and Figure 5-13. Several simplifying assumptions had to be made in order to define the BPEL model using the message sequence charts. For example, BPEL implements a strong sequencing of process steps and timings, those elements are not depicted in the bMSC while sequencing at a high level is apparent in the hMSC. These are two quite different granularities. As a specific example, the flow construct in BPEL permits the parallel execution of two processes with a delay until both process results are returned. This is not implemented in the MSC due to its lack of expressiveness in this area. In Figure 5-12, the flow is named *context 4-tuple*, which reflects the objective of its execution in the overall process.

Figure 5-12: C2Adapt2 Process Detail – Start

The design environment in the BPEL Process Designer works very well, although one issue of note is the largely unconstrained use of variables, giving the designer the dangerous ability to create local or even global variables on the fly. The knowledge of the messages to be exchanged was very useful as it is a pre-requisite to the definition of the BPEL process, and is particularly useful in validation of the variables that were created to support the process. However, a one-to-one mapping was not possible between those messages exchanges in the LTSA bMSC and the BPEL processes. The two systems although cooperative and complementary from a system design validation point of view, clearly express system operation differently. The most advantageous component of using the two for verification comes in the implementation of the messages and message exchanges. Using the message sequence chart to drive the uniform and consistent specification of the messages to be exchanged between orchestrated web services does provide a measure of formality to the web services development process. This would be particularly useful in distributed development environments where one team is building the core web services and other team is building supporting services. The teams can use the hMSC and the detailed MSCs as a shared basis for messages between partners and in the payload specification within the message exchange itself.

Figure 5-13: C2Adapt2 Process Detail – Mid – continued from 5-12

Figure 5-14: C2Adapt2 Process Detail – continued from 5-13

Figure 5-15: Worklist Manager Detailed Process

The Worklist manager, as originally provided by Collaxa and formerly known as the TaskManager, provides a template service for controlled user interaction. In this case, however, it also provides all the appropriate functionality to act as a monitoring service with the role of the user being filled by the C2Adapt2 process itself. The TaskManager provides service interfaces for task tracking from initiation to execution and then completion with reporting [187]. All of these tasks are relevant for the QoS monitoring which is actually providing a task tracking function to the "user" which is the C2Adapt2 web service. It is important to note that there are significant design details that are not represented by the MSC or the BPEL process flow. These details are at a much lower level, the level of the actual module or component function. For example, the design of the dynamic context collection subsystem is one which was given much thought and research as it is recognised as key contributor to delay in the system. This subsystem itself is composed of several "services" and is modelled on a reversed style Jini™ discovery pattern. The standard Jini™ discovery process has the client requesting a lookup for service and receiving a response to that request of the location of a lookup proxy service [188]. The proxy service is then available to the client through a stub. The client requests a service proxy for each service which it then receives as a jar file of the service. For dynamic context collection, this process is the same except for the roles of the client and the server are reversed and in some respects are closed. In this case, the

server requests a lookup on the client using the dynamic discovery proxy. Given the client has installed the dynamic discovery proxy object through the distribution from the service provider, this stub object is returned to the server as permission to receive device and client information as supplied by the specific stub object. The server can then securely query the list of available dynamic context attributes which are available to the server. The benefit of this remote method invocation approach is that the appropriate dynamic discovery stub can be provided for each specific service and specified device at runtime [189]. As the final step, the selection of dynamic context attributes is made and the array of values is required to the server as a message file. This specialised architecture for this one subsystem allows the use of trusted and secure techniques within the process.

To deploy the web services and test their operation, the BPEL Process Manager console is used to create the deployment environment. The designer tool also depends on this deployment environment when it is designing orchestrated web services. The local BPEL directory server is used to provide additional services that may be involved in the design. It also provides the ability to incorporate remote published services as necessary though this may require additional configuration steps within the Process Manager's administration environment. For the C2Adapt2 service, only local services were used and all services were deployed behind a firewall.

Figure 5-16: Oracle BPEL Process Manager™ Management Console

The BPEL designer tool creates a .bpel suitcase of each process once the compilation is clean. The BPEL design environment does not permit the validation of the compilation through a run stage as do most integrated development environment. The service is deployed to the appropriate domain and the execution is checked through the console or through a separate browser interface. From the deployed processes, the .bpel files are used as input to the LTSA BPEL4WS plug-in from the Web Service Engineering project at Imperial College [190]. The previously developed FSP and their labelled transitions from the MSC derived process are then added to the LTSA-BPEL subsystem and a manual map is attempted between the two processes.

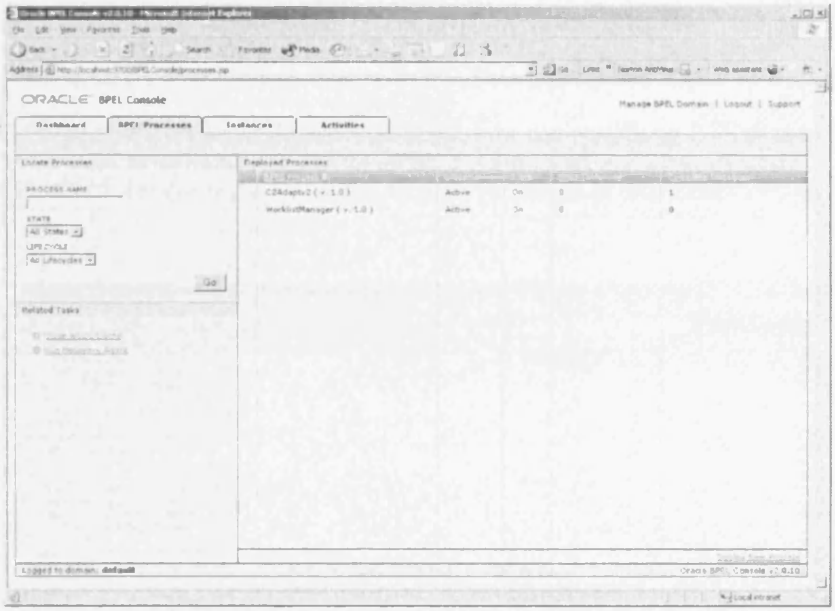

Figure 5-17: BPEL Process Manager Console Functions

The C2Adapt2 BPEL process is translated to a FSP through the execution of the plug-in. The mapping process is outlined in [191]. The result of the execution of this process is shown in Figure 5-18.

Figure 5-18: LTSA BPEL Plug-in Translation

The translation rules are applied as developed in [191] and the FSP for the overall process is generated as depicted in Figure 5-19. This allows the input to the composition of the LTS as before with the MSC derived FSP. It is the resulting LTS that is used as the basis for analysis of the correctness of the BPEL process as is demonstrated in [169].

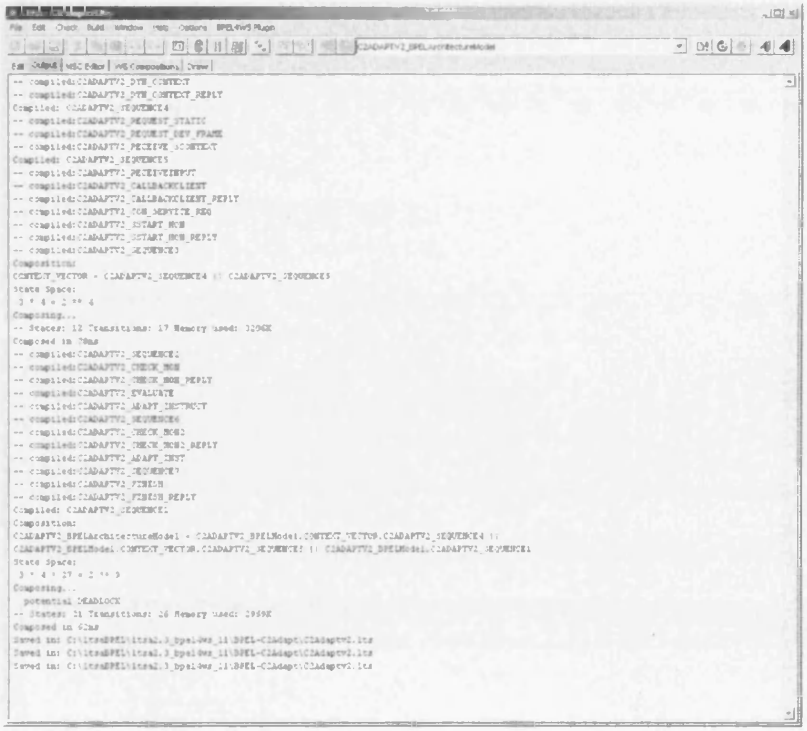

Figure 5-19: LTSA BPEL FSP Composition

The comparison between the MSC derived model and the BPEL designed model show very little correlation in the number of states and in the number of transitions. Removing the Directory partner from the MSC-based model as it is implied in the BPEL process design brings the two closer, but given the small size of the process, the difference is still significant. In the MSC, the number of states calculated without the directory partner contribution was 35 with 48 transitions. The BPEL process resulted in just 21 states with 26 transitions. While the number of states and transitions is not a verification indicator, it is interesting and a subject for further research, the nature of the difference. Both the MSC and the BPEL were flagged with a potential deadlock condition, a valuable observation for the developer. Once the state and the transitions differ, it is logical to assume the resulting labelled transition states will also differ, but it

is instructive for the designer or developer to see where the .lts differ and to understand the difference in terms of message flow and execution.

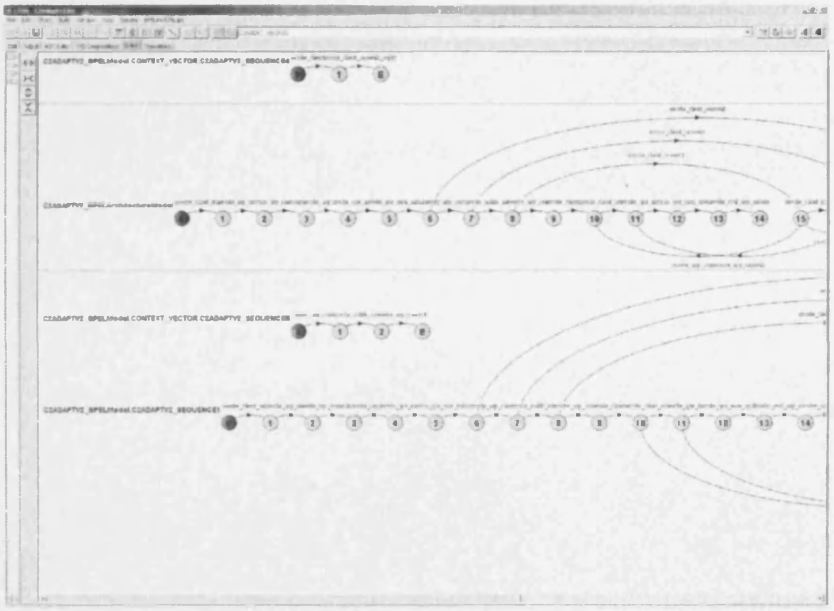

Figure 5-20: LTSA Labelled Transition State Diagram – LHS

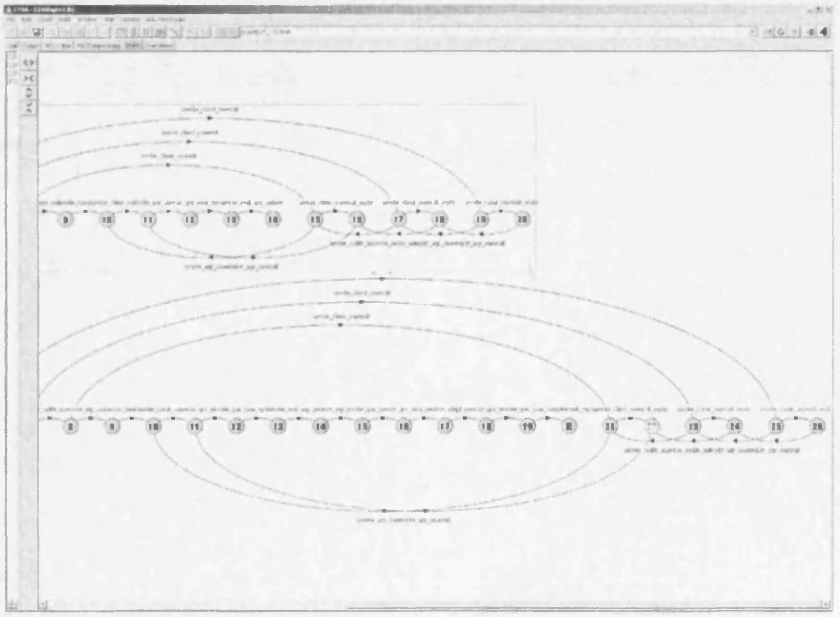

Figure 5-21: LTSA Labelled Transition State Diagram – RHS

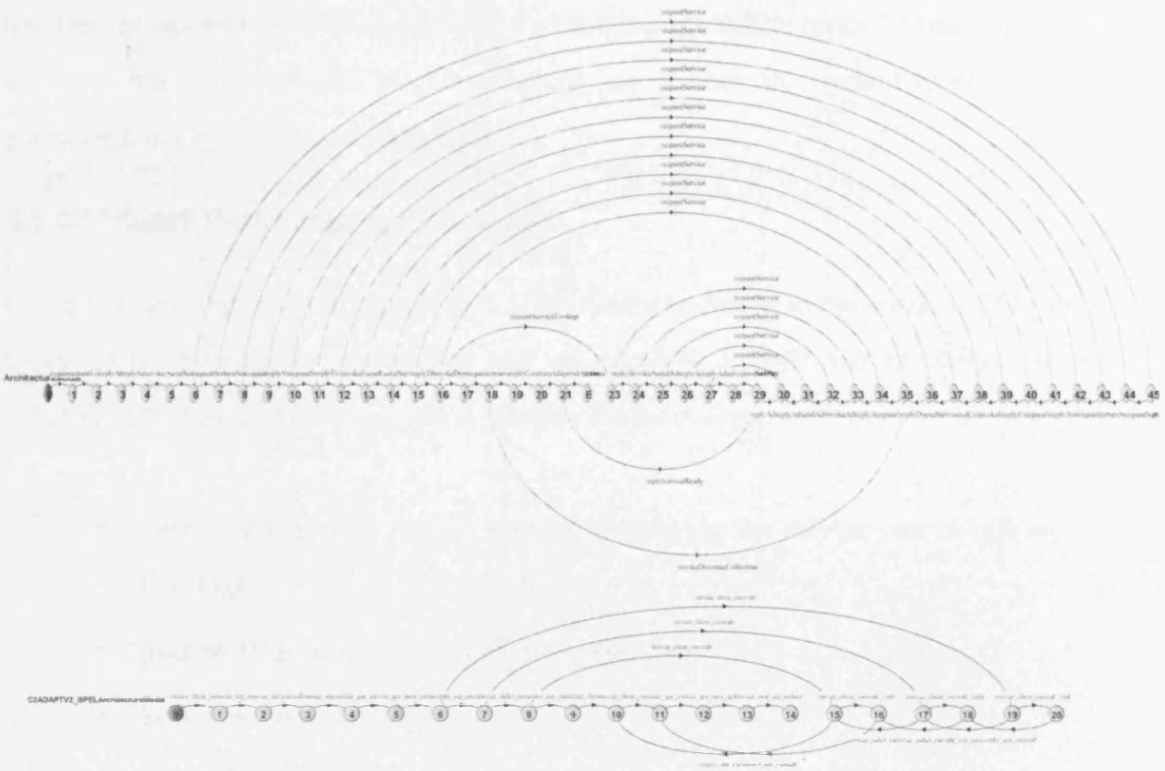

Figure 5-22: Direct LTS comparison

Analysis of the BPEL and MSC-derived LTS diagrams clearly shows the difference in the two processes state spaces. As mentioned above the BPEL designer process lacks approximately 13 states that the MSC LTS specified. This result is interesting, the MSC LTS also shows clearly the over-specification of message passing between specific states which is a function of the designer specification, whereas the BPEL LTS has a reduction of this issue due to the constraints on flow and sequencing which are a part of the BPEL standard. While the comparison of the two is not close enough to call it a verification step, it is extremely useful in the development phase to isolate the issues and causes for the difference, some of which are clearly tool-based, originating from the different starting point in the design process itself. The other component of this difference arises from the different design approach taken. The designer in the formal MSC to FSP to LTSA process is focussed on messages and actors. The designer in the BPEL commercial development process using the BPEL design tool is focussed on sequencing the operations properly and ensuring variables are properly handled between the partners. Empirically, the BPEL designer approach is to either be given or assumes the existence of

the basic component-based software and the design process is to properly compose those elements into a meaningful workflow where are all the appropriate information is populated, manipulated and exchanged.

5.5 C2Adapt2 Performance Evaluation

Using the new architectural approach it is important to return to the work in Chapter 3 to assess whether the improvements and objectives have been met as stated. From Chapter 3, the new architecture was to provide the following:

- Reduce the overall sojourn time by improving the service rate in one of the stages – by a minimum of 10%
- Improve the accuracy of contextual reporting
- Improve the scalability

Through the C2Adapt2 system, the second objective is achieved through specification into the knowledge-base of the static elements of context populated from the manufacturer. The third objective is to improve scalability through reducing the collection service time by at least 10%. The C2Adapt2 web service performance analysis is done through using the component analysis with the contextual variable acquisition. The C2Adapt system required thirty-four (34) dynamically acquired variables, the C2Adapt2 requires just twenty (20) variables to be acquired from the mobile device and fourteen (14) to be queried from the MDKB through the use of the web service. This represents a reduction 47% dynamically acquired variables at a mean time to acquire of 1.133 seconds per variable which implies that the collection phases could be reduced to just 22.66 seconds. However, this reduction does equate to a 47% reduction in the time necessary to complete the context collection as the initiation, query and response time necessary to extract the static elements from KB. Although ideally this will occur in parallel as indicated, this parallelism incurs the expense of doubling of the number of threads as each request will then require a thread for static acquisition and a thread for dynamic acquisition. Maintaining a single threaded approach, one that delivers the

service with one thread utilised, then requires that the operations are serialisable. In the table below, the result of calculating the performance in the model that the system operating in serial means using published web services response time and empirically observed times . The result of the performance for the alternative architecture is presented in Table 5-1.

Single Thread – serialised operation

Static Component –	
	Initiate request – 1.52 seconds
	KB query – 14*.016 =.228s on average (assumes mean time to acquire for each variable)
	Round trip time - request and response using web services [192]- .412
Total:	2.156s

Dynamic Component –	
	Initiate, collect and respond – 22.66s (assumes a mean time to dynamically acquire = 0.0133s/variable)
Total KB-centric context collection	24.816s

Table 5-1: C2Adapt2 Performance (serialised)

Taking this result and comparing the original determined and measured collection service time results in the following:

Model	Performance
NEU (C2Adapt)	30.2 seconds – with equivalent service rate of 0.0333
C2Adapt2	24.816 seconds – with equivalent of 0.0402
Percent Difference	30-24.816 = 5.184s / 30s = 17.28% improvement

Table 5-2: NEU-C2Adapt2 Performance Comparison

This improvement significantly exceeds the 10% requirement proving that the new model will indeed improve the ability of the service to meet its required service response time as previously proposed.

5.6 Summary

The most obvious observation in this component of the project work is that the parallel path web services verification process using state transitions and the resolution of those transitions is a feasible but difficult one. In theory, it provides a very sound and formal basis for the design of complex, orchestrated web services. In practice, the complexity of the context adaptation workflow and how it is modelled in two different tools, one a formal method-based tool and the other a commercial development tool was time consuming and resulted in two very different views of the same overall service. While the parallel design efforts provided input that was instructive and did point out potential deadlock conditions, the expectation of a dual path leading to one set of like states and transitions was not achievable in the time frame permitted for the experiment. The usual time constraints on systems engineering projects might prevent the use of the parallel path methodology given that the issues that occur and the manual reconciliation will reduce the benefits of the formal verification. The safety check and the state decomposition are two important components of verification; these elements should be added to the BPEL designer methodology. This verification does not necessarily though indicate that the BPEL process is the proper implementation of the specification. The ability to combine a set of individual processes and assess that set using the LTSA BPEL plug-in is valuable and a worthwhile additional step given that the designer and

developer both understand the results and meaning behind the LTS that is generated. The only hindrance to this step was the manual recoding required from the Oracle BPEL designer generated code to that which would successfully compile and compose in the LTSA tool. There were particular issues with the structure of the partner statements in designer tool which caused the LTSA BPEL plug-in to fail to compile and to not return any indication of the offending statements. Using the sample code provided, this was easily addressed. This fact also points to the broader observation about engineering systems in an open, orchestrated web services model. The technology standards and tools are highly dynamic. The number of different toolkits and individual components necessary to engineer a fairly simple set of web services was approximately seven different tools or toolkits. In the timeframe of this programme work, each of the different tools had significant new releases, in some cases, three separate times and in one case, a significant change to the complete product with the change of ownership. The change in the BPEL standard or in the tool vendors interpretation of the standard, for example, the change in the use of serviceLinktype to partnerLinktype although quite straightforward, means that the different rate of adoption between the several different tools results in underlying incompatibilities which are not evident at the development environment level. In addition to the process of building the C2Adapt2 service prototype, the changes of the environment and individual tools meant approximately twenty-four (24) new installations, configurations and tool validations were required to keep the development tools operating while the design and development process took place. This, more than any other component of the experiment, represented the largest time spent in the process. Issues with firewalls and networking inconsistencies within the process management console also contributed to a non-productive set of time delays in the overall development process.

Analysis of the workflow prototype itself provided evidence of the feasibility of the creation of the service-oriented context adaptation system using the knowledge-base as the repository for service requirements and device static context. Devices can evolve and requirements can change over time and the services which extract and use that information are required to understand its use through published interfaces. The issue

of retrieval of static context is straightforward in this approach using the programmatic method of a PAL query from the Protégé library, a fact that means a suitable Java component can be constructed to extract specific frames from the MDKB and in the future the constraints can also be retrieved. The most significant issue which remains is in the evaluation of the context matching, a well-published issue in the semantic web research area. The evaluation process involves the ability to determine the degree of the fit between the requirements and the capability, given that the information is represented by a set of parameterised values, this is a cumbersome process and takes far too long and may not even finish if there is insufficient determinism in the parameter values. The dynamic collection of context is designed using Jini™ services. It is recognised that many mobile devices do not or cannot support the Jini protocols and therefore cannot directly participate in a Jini network. This is changing however, with the availability of the Jini Surrogate Architecture which allows interconnects to non-Jini devices from Sun [193]. In this model, the client for the Jini interaction is actually the ASP server. The dynamic context server is the mobile device which provides the listing through a separate Jini service. It is also possible that the mobile device client component that interacts with the user, although not required in the prototype implementation, can be a J2ME client application written as a MIDlet, which is supported on many mobile micro platforms. The key interaction with the user is the display of progress in the context adaptation and in the permission for collection of and change to the dynamic attributes in order for the adaptation to be optimal. This, however, is not the primary focus of this chapter but is a future implementation detail if and when the prototype is used for possible advanced development activity.

In evaluating the C2Adapt2 system, the design of the orchestrated services allows for static context retrieval, dynamic discovery through a specialised subsystem, it permits the selection of evaluation services that can determine the necessary adaptation instructions and it allows the selection of adaptation services that fulfil the requirements of the overall value added service. While this may fit the original architecture proposed, there is still considerable message passing required to complete one such adaptation service. In a mobile world, message passing requires connection or endpoints and

mobility incurs disconnection as device and their owners pass between network access facilities. The performance gain was evaluated given the published access and response times for web services, the worst case scenario of one variable per KB query and a .412 second response time to the context aggregation server. Given this, the time improvement experience coupled with the accuracy improvement demonstrated that the C2Adapt2 KB-centric approach was empirically feasible and superior over the full dynamic discovery process for the context collection phase of the overall adaptation system. The new architectural model functions for a single request quickly and efficiently but becomes significantly more complex when the architecture needs to support multiple, simultaneous requests. The task of engineering such a future system where the device and the service itself are changing rapidly is made more difficult in the open service-oriented business model where third party services may be configuring the destination environment and the requested service execution for their own needs and convenience. It is worth considering such a possible future state given the emerging reconfiguration capabilities that are becoming a reality in hardware device engineering.

Looking forward towards the future of engineering context adaptation services using knowledge-base web services reveals a potential issue arising from the increasing dynamics of the mobile device itself. The C2Adapt2 system as a prototype implementation is not capable of handling the real-time reconfiguration potential since the implementation assumes a client which can report its dynamic context as a list of current values for relevant contextual elements of the service requirements. This occurs at a level of abstraction higher than that in reconfigurable systems which have the ability to change at more fundamental levels than at the application layer. This concept will be further outlined and discussed in Chapter 6 where the notion of reconfigurability and its impact on mobile context adaptation will be investigated. The C2Adapt2 prototype will be envisioned within the new reconfigurable device environments to determine what if any modifications to the services will be necessary. Finally, concepts concerning how to design services for these reconfigurable devices will be presented.

Chapter 6 Mobile Reconfiguration

6.1 Reconfiguration in Mobile, Service-Oriented Architectures

Context adaptive services promise to provide new opportunities for revenue-generating online services by using the capabilities and situation of the device, the network, the user and the desired service. In Chapters 4 and 5, it is shown that context adaptation services require special design and engineering in order to deliver such optimised services in the agreed upon time and within the contractually designated quality. This statement applies to the engineering of context-based services in the current technology environment, but future computing architectures need to be investigated to determine the implications of predicted changes in architectures against the operation of the context adaptive architecture and design presented in Chapter 5. The focus of this chapter is to analyze the nature of key future trends in mobile architectures, namely mobile reconfiguration, determine its impact on service delivery of the future and determine how the C2Adapt2 architecture as designed will behave in the presence of this potential future technology landscape. Reconfiguration is a concept which has recently emerged as the newest architectural paradigm for electronic systems design. As with terms and concepts like context, services, and architectures, the term "reconfiguration" suffers from several definitions and various interpretations. Reconfiguration, for the purposes of this work, is defined as the ability to change a device's lower level configuration through software after its original manufacture. Generally, the term reconfiguration has been applied to functions of the processing hardware, where reconfigurable processors provide the means to allow programmers the ability to change low level hardware algorithms in order to fundamentally change the operation of the device [87]. Equally, researchers working at the higher system layers have used the term to express the ability to change the service composition of devices such that a new set of functions or services can be offered [194]. In this project, the higher architecture layer reconfiguration is termed "adaptive services" or simply adaptation in order to

distinguish it from the lower level hardware reconfiguration concept. This paradigm of lower level reconfiguration which is initiated and implemented through software controlled systems is a significant change to the architectures found in computing. Software engineers and developers have historically come to depend on some static capabilities within their deployment environments. The possibility of a mobile, service-oriented architecture where the client device can change fundamentally from service to service adds yet another and still greater engineering challenge for the service developer. The implications which will be investigated in greater detail in subsequent sections of this chapter are that services themselves can facilitate the reconfiguration of the client device in order to prepare the device for a service to be delivered. As outlined in Chapter 5, this workflow of services which includes preparing the device and then delivering the service will be called context pre-processing. The purpose of this chapter is to present the results of analysis and investigation of how a knowledge services-based architecture, such as the one designed and implemented in this programme, would accommodate the future reconfigurable device environments. Also in this chapter, a review of several mobile reconfigurable architectures will be presented and the design implications of reconfiguration from an engineer's perspective will be discussed. An evaluation of the knowledge-base services within a possible reconfigurable architecture will then be presented with specific attention to QoS mechanisms and issues. In the final sections of the chapter, new concepts and design techniques will be presented for the evaluation of possible service workflows of the future which support reconfiguration. The advantages of a "pre-processing" architectural style for context will be explained and discussed as the knowledge-base service architecture evolves through the additional requirements of reconfiguration into an "assemble-to-order" service delivery architecture.

One of the challenges in engineering services and systems which implement reconfiguration support is defining a shared and concrete degree of reconfiguration. A system can be reconfigurable in many ways and at several levels with the systems architecture. Reconfiguration can also occur in varied ways throughout the function of a single layer as well as in varying time scales of execution [195]. On the processor level,

reconfiguration often refers to the use of domain specific field programmable gate arrays as component on a board which will allow some low level functions to be altered. This reconfiguration event occurs after the processor's original manufacture and is in contrast to the use of ASIC or application specific integrated circuits whose functionality is fixed once the chip is manufactured to perform a specific function [98]. As described in [195], "...reconfigurable systems can obtain specialization at run-time" [195]. This is a powerful computing concept, to essentially create a processing platform to a service specification at run-time rather than being entirely dependent on the higher level software and applications to deliver the services. Although combining high level software and applications that are designed to leverage the features of the device is the traditional service engineering method, the emergence of reconfigurable devices will present new hurdles and require new methods to provide networked services that leverage the new computing architecture properly. Hardware reconfigurable systems and reconfigurable subsystems within larger systems are slowly emerging as a result of many years of research. The need for such systems is being driven by the increasing complexity of personal devices and the need to accommodate changing features and functions while maintaining a minimum number of networked computing devices. Two of the key disadvantages often cited in reconfigurable processing platforms are speed and circuit density [81]. These generic processor architectures are characterized by comparatively large clock frequencies and low densities of circuits on the chip. This means that a single reconfiguration or change of function must ripple through the system at both the higher level of abstraction at a higher clock frequency and than at the hardware control level which operates a significantly lower cycle frequency [81]. This operational issue presents an entirely new world of "multiple phase" adaptation where new timing aware tools and methods for the designer and the developer will certainly be required.

As mentioned, adaptation to context in systems with reconfigurable endpoint devices will necessitate understanding of not only what is changed, but on what timescale the change occurs. Reconfiguration is possible at the instruction set level where new operations can be inserted in order to support the extension of capabilities or run-time

operation. The concept of reconfiguration across the various layers of the classic systems architecture stack is covered well in [94]. In mobile architectures particularly, two dominant scenarios drive the implementation of the reconfigurable platform design, power management as outlined in [94] and MAC layer reconfiguration as found in wireless networking topologies. In both these scenarios, a significant issue with engineering applications for mobile devices is motivating the move to a reconfigurable platform. In the MAC layer reconfiguration case, the reconfigurable processor blocks take up the specialized function of encryption and decryption [196]. By partitioning the encryption tasks for example, into the configurable blocks, then new perhaps more secure algorithms can be deployed quickly into the field through electronic software distribution processes. As well as cycle frequency issues mentioned already, the nature of reconfiguration at different levels occurring at different points in time creates a possibly conflicting processor environment where the task cannot be executed because the state of the system is never ready to run it. The timescale of importance in this case is the overall interval and frequency that is required to prepare the system across all the architectural levels with no loss of service. This is done in parallel with other executing tasks which do not require the reconfiguration so the overall system must stay in a safe computational state. From a hardware reconfigurability perspective, this is sometimes known as the "binding time" [94]. For general purposes of service reconfiguration, the service binding time will be the total time for reconfiguration across the system layers and the time necessary to create an operational state of the device for the next function.

Service binding time = micro-architecture binding + object level binding +

service interface binding

Equation 6-1: Service binding time calculation

An example of a service binding time in a reconfiguration operation is the previously mentioned encryption algorithm run-time reconfiguration in the MAC layer. When a mobility change event is initiated which dictates the movement from one wireless access interface to another, the MAC layer adapts to the change and to the new requirement for encryption which is specific to the new communication interface. The application

service will need to re-bind with a local UDDI server to the nearest application server which can support the necessary hardware control level reconfiguration, in this case, the change to underlying transport and in some cases physical layers. The concept of such reconfiguration service as a third party API extension is a topic of current research and is documented within the literature [197]. It is clear from this simple example that the resulting interplay between reconfiguration at varying levels over time is a complex engineering landscape of which application contextual adaptation is a subcomponent. This complexity increases still further when the situational context enters into the design environment. For example, the potential of a low battery condition occurring while the reconfiguration is in progress presents yet another set of reconfiguration operations which may pre-empt the service initiated configuration. Given the dynamic nature of mobile device operations, any reconfiguration requires pre-emptive architectures with a control plane on which real-time decisions can be made. This means that future service engineering which leverages reconfiguration will require a new approach, a top to bottom integration approach as well as the traditional software which requires more lateral component integration. In a sense, reconfiguration encourages a "3 dimensional" engineering approach which includes not only the service application software engineering, but also the reconfiguration service engineering required to deliver the service within the QoS parameters governing the services specified. The inclusion of a new set of non-functional requirements such as total binding time control and management support will be necessary if the future computing paradigms are to include multilevel reconfiguration.

6.2 Knowledge Services Support for Reconfiguration

With the possible emergence of reconfigurability in next generation mobile devices, online service developers will be faced with these new design and implementation challenges. Previously, variability of the destination device was a severe limiting factor in the quality of the deployed service and was also a key factor in constraining the possible service marketplace. As was previously mentioned, this variability forced the service designer to make significant assumptions about how a service will deploy to a heterogeneous market segment of device owners. Conversely, the service designer could decide that due to limited resources, their application service could only be deployed to specific sub segments of the market, that which could guarantee a certain level of functionality present for deployment. Yet even in this environment, the variability was largely borne out of changes in the application layer or limited changes to the device that were allowed in an operational setting, e.g. contrast, volume, etc. Reconfigurability extends the variability of the device deeper and into a more impacting level as it enters into the hardware abstraction layer. With this in mind, it is important to understand how such a new computing metaphor would impact the context collection and adaptation knowledge-base architecture which has been built in this research. The key question with any systems research is the extent to which it can support future paradigms such as reconfiguration and what implications can reasonably be foreseen that can be modified now in preparation for this future.

A first impact of reconfiguration is on the use of the KB which provides an evolving repository of field-based capability that can be used by the service developer to represent the known static context state of a device or device class. Static context, however, will most likely continue to decrease in significance if device reconfiguration becomes the dominant design metaphor in web services or future new architectures. Essentially the premise is that as devices become "more reconfigurable" there will be less static contextual information about them. At the limit, the static context elements become the small set of tangible, physical attributes of the device, only those that cannot be changed either through software distribution or reconfiguration. There are a series of

publications and reports from recent studies on reconfigurable architectures and their specific advantages and disadvantages to be found in the literature [198], [195], [85], and [95]. Most research work in this are focuses on component-based software architectures, not on service-oriented approaches. The concept of open, exposed interfaces to orchestrated services in relation to reconfigurability architecture has not appeared in the literature as of the date of this work. For example in [198], takes a more traditional component-based architecture approach of "objects and layers" when considering implementing an application out to a reference reconfigurable device [198]. This reference reconfigurable device architecture is sufficient for the purpose of understanding the future architecture and will be used in this chapter to understand the possible implications of using the C2Adapt2 with reconfigurability. The resulting architecture adapted reconfiguration reference model based on Georganopoulos, et al is depicted in the figure below:

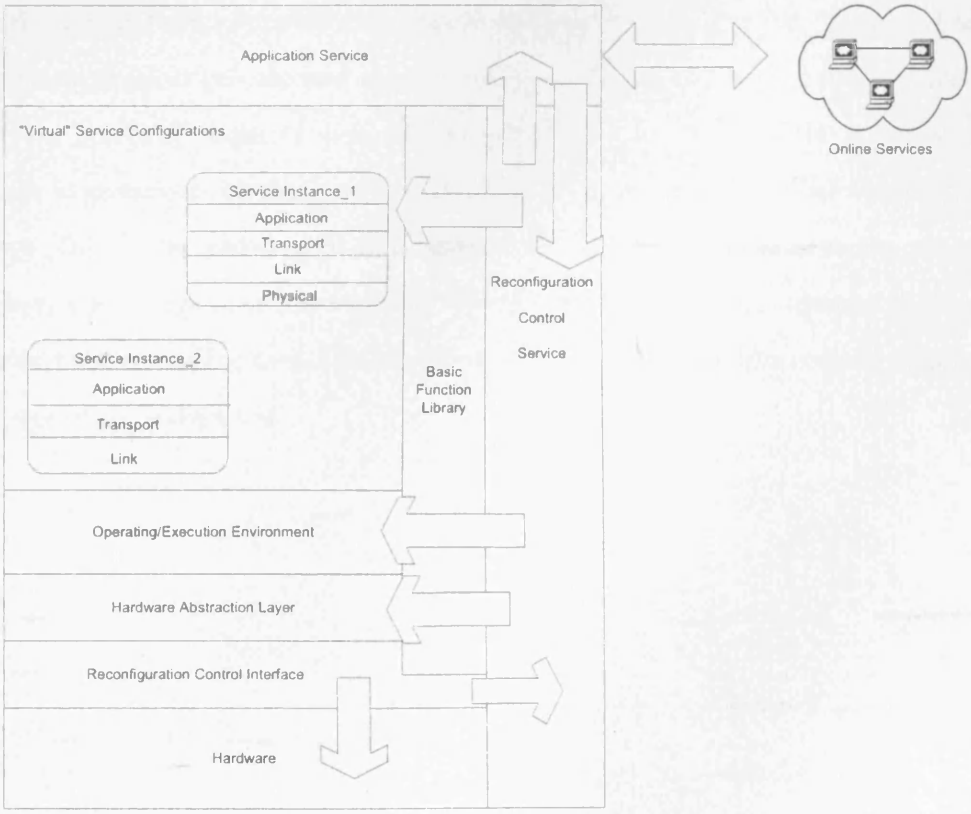

Figure 6-1: Adapted Reconfigurable Device Reference Model

While the device reference model outlined in [198] suits their purposes well, it can be modified to meet the needs of a service-oriented architecture. The modified model in Figure 6-1 is adapted to fit within the service-oriented architecture in a more generalised services market environment. In such an architecture, the reconfiguration support on the mobile device needs to transcend application and execution environments and needs to sit above these constructs in order to orchestrate the reconfiguration at all layers and to guarantee that there is no chance that the system will deadlock waiting for the optimal reconfiguration conditions to occur. The purpose of the knowledge-base in a reconfiguration environment would be to support the potential states and the last known good configuration. The KB would still take in the manufacturer specifications as before, but then with each adaptation service request, would maintain an expanded instance on the reconfiguration state designed to support the delivery of the service in question. The run-time binding would then be the vehicle which permits real-time reconfiguration through either component addition or through the linkage of service description to allow private and secure access to remote service invocation. Binding of this type, however, requires very careful engineering to ensure there is the ability to manage expectations of performance through a cross-layer, real-time reconfiguration process [88]. The addition of new facets, frames and sub-frames to the previously designed knowledge-base are required in order for the MD-KB to operate in the mode discussed above. In Figure 6-2 below, the use of the MD-KB as the core reconfiguration state repository is depicted.

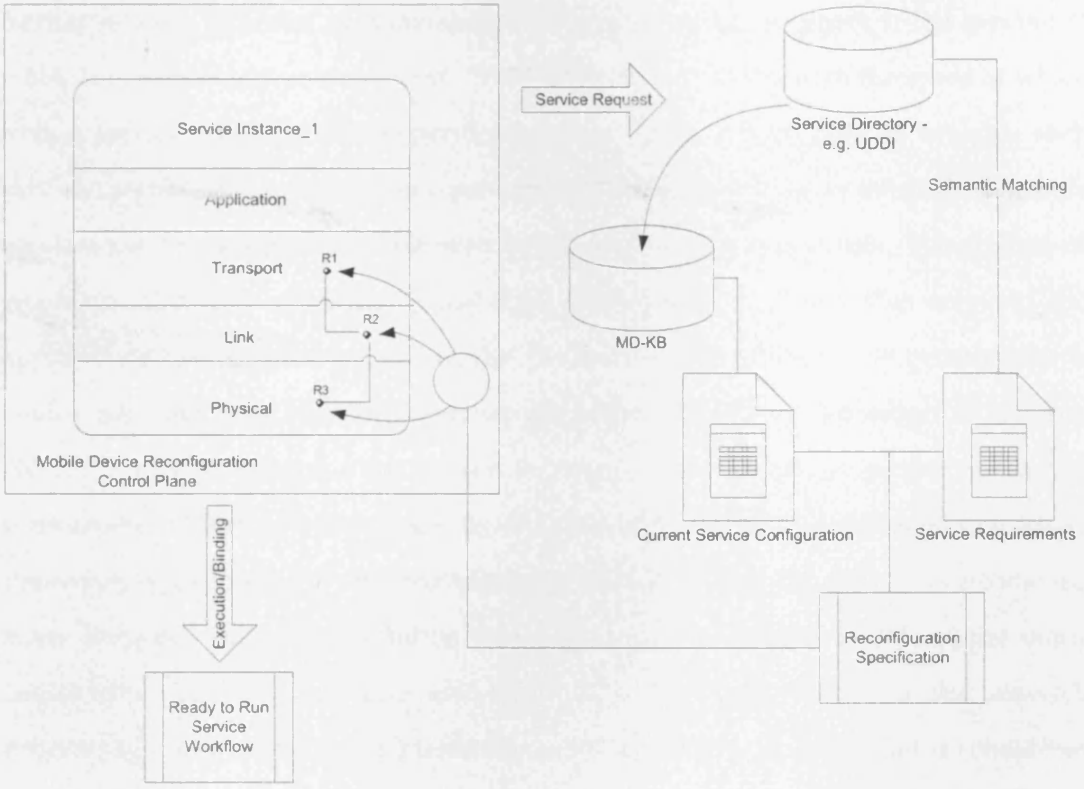

Figure 6-2: Knowledge-based Reconfiguration Support for Adaptation Services

One key concern with supporting real-time reconfiguration is management and control of the various service requests that require a response involving reconfiguration tasks. This is particularly worrisome when one considers that the future online services industry requires that many third party designers and developers deploy innovative new services. Many of these services will be designed and built by engineers who are unaware of each other's activities. It is clear that just as in the case of web services, reconfiguration in service-oriented adaptation architectures will require uniform standards and reference models in order to prevent chaos on the device level and a complete loss of service integrity for the customer. The modifications to the MD-KB are designed to support the creation of reconfiguration specifications which are in turn designed to support the whole service workflow, not each individual service component. An issue that will remain for future research is how to ensure that the separate service workflows will interact properly and optimally. This new knowledge-base service architecture includes reconfiguration support as an initial concept only and significant

further work is required to understand the detailed nature to prove if the concept is viable in a commercial environment. There are perceived issues with the speed at which such a service reconfiguration specification may be developed, how to evaluate each service specification against others queued or operating and how to integrate disparate services that are designed and engineered from several different parties. The promise of reconfiguration and specifically, real-time reconfiguration means that services and applications can be delivered essentially "to spec" on one flexible, yet generic mobile device allowing true low level context adaptation without proliferation of devices. However, this potential can be realised in many ways through an infinite number of mechanisms, technology and processes, the ideas in this section are intended to uncover the technological potential and provide some insights on ways to start to overcome the many hurdles. An overall solution and engineering environment will require many researchers and far more time and effort than can result from this one research programme. One of the more interesting issues in this area, and one that is considered in this chapter, is the need for new methods of design engineering to support reconfigurability within the service-oriented architectures. As mentioned above, the cross-layering aspects of adaptation coupled with multi-level reconfiguration increase the dimensionality of the systems engineering challenge several fold. The design of services in the multi-dimensional space will require more than component-based software engineering approaches which are in common practice today. One way to design and model these new services and include the unique requirements of reconfiguration is to utilize the new multidimensionality of these systems with techniques of software visualisation. In Chapter 4, software visualisation was used to depict the inherent nature and structure of the mobile device ontology and how well it supported the instantiation of the mobile device in the field. In this chapter, visualisation techniques can again be used, not for the purpose of demonstrating the model suitably but instead to create a multi-dimensional depiction of the service design.

6.2.1 Visualising Reconfigurable Services

The notion of using a KB-based context adaptation service with reconfigurable devices opens up many new questions in both research and engineering. To understand the area

and the problems that may be encountered in this future state, it is instructive to envision the device state space. The initial investigation into the nature of reconfigurable context adaptation operating on mobile devices revealed that new design engineering methods will be necessary in order to provide the means for system to be constructed within the constraints of both the spatial reconfiguration and the temporal adaptation [98]. Given the complexity of the system design and operation, techniques from software visualisation appear to be appropriate as they can represent both the spatial aspect of where the functions are implemented and the timescales of the functions simultaneously so that the engineer can visually inspect the operation over the time of the service. The visualisation would provide the ability to depict the conflicts of the reconfiguration that may be caused by disparate service instantiations. Three dimensional (3D) representations of software are common in several areas of systems engineering and are particularly common when dealing with very large scale software systems such as those found in the telecommunications industry [199]. The use of 3D visual representations allows complex system operations to be depicted with large volumes of data in a manner that the engineer can understand through the cognitive association of trends with the execution of the code. Software visualisation is often used to depict information about areas of code which are frequently called or those segments which are frequently maintained or altered by the programming staff [200]. The ability to visually inspect a system's operational data in real-time provides a significantly more effective way for engineers and designers to understand the complexity and interconnections of the code or in this case, the services [201]. The inadequacy of expressing the operation of these systems is well documented in [202], the notion that in order to properly understand the relationship between computational elements, in this case the service workflows, the physical nature and the timescales should be visually presented simultaneously which cannot be done in a 2D only design model.

A new 3D model of the service can be derived from the combination of the unique spatial attributes of reconfigurable hardware and the use of the software execution visualisation. The physical aspects of the reconfigurable processor are represented by the location of the circuitry in the x and z directions. This represents the deployed

location of the function in terms of a simplified grid. The impact of the module reconfiguration at the location specified in terms of the systems architecture level is then indicated in the y direction. As the service reconfiguration changes to accommodate the context adaptation over time, it will then create snapshots of surfaces in three dimensions. Each service workflow, a composite of individual services, will then have a series of surfaces representing its reconfiguration and operation on the platform. The set of surfaces which represent the service workflow is called the "service instantiation surface". Even in a hybrid reconfigurable environment, a 3D model will be useful in support of reconfiguration where those reconfigurable regions and the static processor sections can both be mapped depicting where code can be deployed and changed versus areas where it cannot. A central issue for reconfigurability is clearly seen in representations such as these, the issue of representation of the different timescales across the system architectures. Computationally, the higher levels of a reconfigurable architecture operate with threads, counters and registers while at the lower levels, execution is at the bit level with different cycle times dictating the step or task execution. Sections of the reconfigurable hardware processors must communicate with the sections that are implementing functions operating at the lower timescale levels [84]. This timescale issue can be demonstrated clearly as the engineer starts to create a 3D visualization without time scale information and finds that the representation soon loses its ability to simultaneously depict the events at all the levels. There has been a considerable amount of research already done in developing the foundation of configurable processor architectures, much of which has been done in support of the system-on-a-chip (SoC) architecture and technology area. This chapter cannot describe the vast amount of effort in this area, but proposes to highlight the concepts and implications of reconfigurable architectures that should be of interest to software service designers and engineers. One of the greatest implications of reconfigurability on services-oriented architectures is the return to the importance of understanding the nature and details of the hardware processing for software and systems engineers. The early days of programming for the computers required a detailed and thorough understanding of the hardware and the equivalent hardware abstraction layer in order to properly program the device. Programming was a hardware manipulation function

involving wire wrapping and machine level register manipulation. In direct parallel, these early days of reconfigurability brings with it a need for the software and service engineers to understand the nature of the hardware architecture and its possibilities. To this end, the next section will present ideas on how to achieve further understanding.

A particularly successful example of a hybrid reconfigurable architecture is that of the Gecko from IMEC in Belgium. This architecture provides an environment where two Xilinx© FPGAs are used in conjunction with fixed application-specific integrated circuits (ASICs) to demonstrate how tasks are executed in and moved between the different system architecture layers [84]. From a service-oriented architectural perspective, this early stage prototype demonstrates the flexibility of reconfigurable systems and architectures such as these, will take context awareness and adaptation to a new level of operation and value. The spatial efficiency and placement of reconfigurable code objects on the hybrid processor means that in context adaptation, it will be important to not only get the new function to adapt the service to the context, but to implement it in the right location of the hybrid processor architecture. It also demonstrates that new invention is required in software techniques to deal with the maintenance, performance tuning and integration of the systems. Creating a 3D abstract model of the device and service interoperating in real-time will depict runtime configuration and operation of not only the software but the aspect of the reconfigurable hardware as well. The diagram below, Figure 6-3, provides a high level depiction of this basic concept:

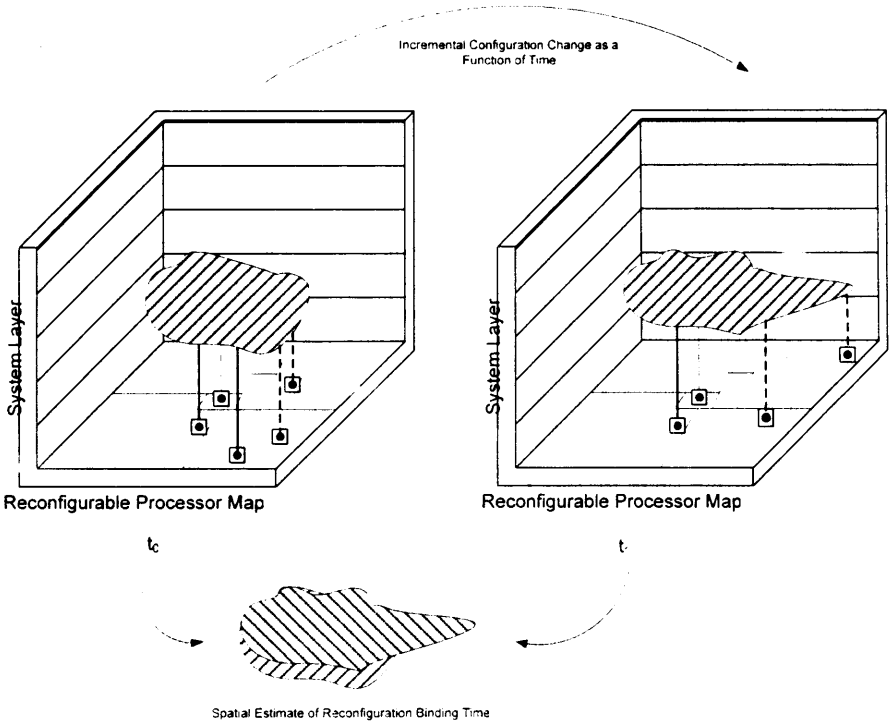

Figure 6-3: Three Spatial Dimensions and Time for Reconfigurable Systems

The service instantiation surface or SIS, is a mythical set of surfaces which visually represents the functionality of the service workflow and the reconfiguration in a specific configuration on the mobile device. The service definition indicates the required elements in either a networked directory or even in an XML DTD. The matrix location is formed by the processor configurable block address and the affected layer. The surface is instantiated and the execution environment manages the execution of the instance itself. The MD-KB stores and allows the evolution of the baseline service configuration and can be used as input to the creation of the non-runtime 3D model. The components of the current MD-KB are extended through the addition of the system layer frame. The system layer frame would then contain all the reconfiguration impacts for the service mapped on to the device for the execution of the service. The actual development work to implement the extension of the MD-KB to support reconfiguration in context adaptive services is beyond the scope of this programme and will be left for future work. The difficulty of understanding the complexity of implementation and operation of reconfigurable adaptation services is in some part to do to the increase in the number of informational dimensions. Visualising the operation and interoperation of services will

require a representation that allows the systems engineer to map the service operation and function location into one diagram and to link that diagram back to the set of service contained within the workflow itself [199].

As previously discussed, software visualisation provides techniques for design of complex reconfiguration services in a mobile environment. Other tools such as those for indicating direct service conflicts for resources, for analyzing possible deadlock conditions from the service description and reconfiguration instructions themselves will be important to provide the means to create robust, production services. Tools for performance analysis and optimisation of multi-party service workflows will also be necessary in order for engineers to properly create services that are manageable, efficient and perform to expectation in a heterogeneous environment. As noted above, reconfiguration introduces a spatially important dimension into the engineering problem for all reconfigurable services, not just context adaptation. The spatial layout provides one way to design an efficient set of tasks, pointers or objects across a SoC [80]. This efficiency will change over time as the context changes and the device is forced to respond to changes from the outside environment as well as the requests of the user. In earlier chapters, this is presented and discussed as a series of state changes over the system in time varying, t. It was mentioned that for each time, t there is a set of surfaces by service describing the changes to the reconfigurable sections at the system layer of the change. This virtual surface set can be used to understand the runtime and design time nature of the service workflow itself.

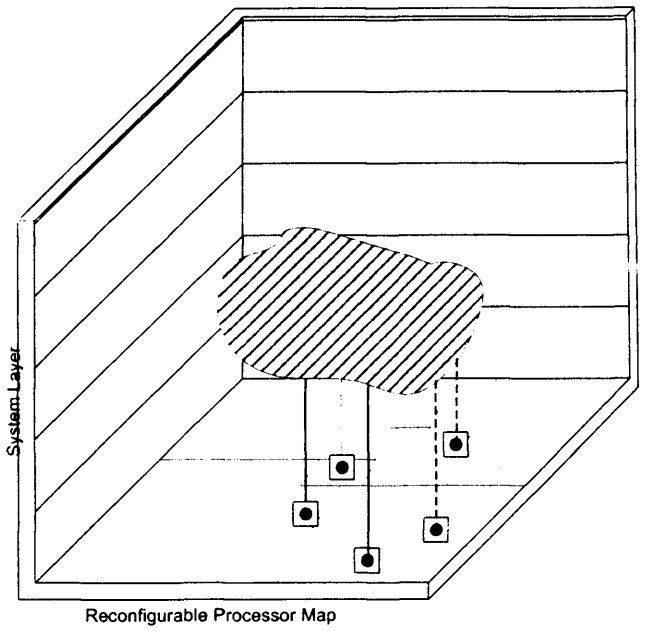

Figure 6-4: One Service Instantiation Surface

In Figure 6-4, the service is represented and depicts the regional change and the layer change in the vertical direction. From this, a designer or engineer can visually understand the extent of reconfiguration by the total surface area. It can depict the impact of changes in the layers of the system, thus a service impacting lower levels of the system architecture will have a greater impact on the performance of the overall device when it comes to interoperability. Visually inspecting the service and its implications will assist the engineer in perhaps prioritizing and planning for testing, e.g. for those service which impact a greater number of system layers, these may need further integration testing [201]. A second advantage of introducing a 3D service design space is that the current configuration of the device can also be represented by a set of surfaces. The use of spatial analysis between the current configuration space and the required service configuration space can be an alternate way of determining the extent of device level adaptation required for a service delivery. This will, of course, not outline the environmental context adaptation requirements or other non-device specific alterations that are necessary.

In the knowledge-base database, hysteresis data from the instance surfaces of a service for the devices are maintained in much the same manner as the individual device instance records were created and provide a historical record of the changing device state over time. A simple log of the most recent services and the surfaces can be used in order to provide a representation of a possible improved configuration baseline for each device. For example, as the user purchases services from various third-parties, the extent and nature of the reconfiguration and adaptation will be contained within the knowledge-base. Over time then, each baseline configuration of the device can be tailored directly to the user's habits and behaviours by analyzing the commonality found across the service instance surfaces. The benefits of this new approach and representation are the following:

- Provides a clear representation of a complete service request at all layers of the system architecture, across the reconfigurable components of the hardware and provides for the depiction of the components as needs for the service adaptation
- Provides a new design method which unifies the cross-layering design with the 2D representation of the operating functions in the reconfigurable hardware
- Provides a visual tool for analysis and allows the engineer to observe the operation and integration of services much as in the case of the Building Box system from A.T.&T allows the service developers in the telecommunications industry to design new services and visualise how they will interoperate before deploying them [201]

A major disadvantage of this approach is that a surface representation requires a set of consistent parameters in order to have meaning. In order for surfaces to have meaning for comparative analysis the nature of the elements within those surfaces must represent the same functional component or it must be categorized at a coarse-grain to allow for variations in the detailed nature of the representation [200]. This implies that as the device is reconfigured, perhaps dynamically in some cases, the extent of the reconfiguration must be summarized at a higher level in order to be useable and

meaningful. The nature of the 3D surface representation must give the appropriate reconfiguration details without sacrificing usability by depicting too fine-grained information. Another important question is the nature of the visualisation itself. Many visualisation tools and execution profilers use previously recorded and collected information to provide the representation on code operation rather than operating in real-time. Context adaptation services cannot be assessed in this way completely as the operation of the context adaptation is based on a previously unknown set of circumstances so profiling or collecting information may provide some clues on performance but it will insufficient. The question is how to understand the location of the functions as they are dynamically deployed without incurring crippling overhead of tracking and reporting all the miniscule reconfiguration events. This limitation is counter to the desire to have a device in which whole blocks are reconfigured at the control of the processor without having to be constrained by the current physical layout. Further work on these implications is required and it may, in fact, not be a significant issue as board interconnects and logic blocks may dictate the configured components one and only location.

6.3 Implications of Reconfigurability on Context Adaptation

As discussed above, the advantage of reconfiguration is increased flexibility and functionality deployed in real-time to users based on need. The disadvantage is sub-optimised performance. The hybrid reconfigurable architectures attempt to afford some degree of flexibility while maintaining optimised circuits for resource intensive and repetitive operations. Designing context adaptation systems, either service-oriented or component-based, in reconfigurable device architectures means a greater degree of understanding is required of the reconfigurable hardware itself. This is quite obvious and the tie between context adaptation and reconfiguration is through the detailed operation of the device in the field. Current engineering of context adaptation (CA) systems requires some, albeit limited understanding of the deployed hardware. It is also requires the use of some capability reporting mechanisms for the majority of the information about the nature of the destination device. Without the use of the knowledge-base approach which is the focus of this work, current context adaptation

engines will rely heavily on the device reporting its own attributes to the service originator as a pre-processing step for adaptation. For example, in CC/PP, the client initiates a service with the requirements for the service in the header, the details of the client capability are not dynamically derived in fact they are specific to implementation of the client CC/PP implementation and version. While the quasi-static nature of the RDF and URI description is a step forward, it will clearly depend on the version of CC/PP implemented and the specific device. Reconfiguration of the device in real-time based on the service requirements will require the ability to add new tags to the semantic description of those capabilities. It is difficult to see how the CC/PP standard will be able to support this in a uniform manner. New descriptive tags and RDF triplets will be required based on the nature of the context required for the service and the level of support. Like many integrating standards, or at least those that have a significant purpose to provide integration capabilities between technologies, the key requirement is flexibility to evolve on the same timescales as the technologies on which the standard is based. CC/PP as written does not include the provision for reconfigurability nor does it handle context capabilities aggregated at the service level, both of which will be required for next generation context adaptation services.

The combination of concepts and opportunities of reconfiguration with flexibility found in context adaptive web services provides a new style of distributed computing. Although as outlined above the process of context adaptation needs some measure of control for the orderly evaluation of context, the nature of the service itself within an environment of reconfigurable hardware is as a distributable service. Reconfiguration extends adaptation still further into the intrinsic functions of the mobile device and this increases the challenge of engineering such services under the constraint of a quality of service agreement. This topic will be discussed further in the subsequent section but once again it provides a performance limit on the complexity that a system can afford. It is the significant cost factor in the overall optimisation equation for these adaptive systems. Device changes are initiated by instructions by the adaptation manager service which is the service that monitors and manages the execution of a specific context

service. In the first instance, an evolving KB is linked into this process to assist with input to the collection and evaluation phases of the adaptation workflow. In future, devices which are fully reconfigurable will start to nullify the advantages of this type of architecture and service workflow. Devices will have shifting functionality at all levels of the traditional system architectural stack, not just at the higher application layers. The notion of dynamic device discovery becomes significant as many of the elements of the context 4-tuple that were previously assumed to be static will be dynamic and bound to the hardware at runtime.

Adaptation in the future of reconfiguration warrants a new approach, one that can mirror the flexibility of the system itself and one that can attempt to reduce the increased complexity. A new spatially oriented design and deployment model based on 3D surface analysis is proposed to motivate an improved methodology for engineering integrated reconfigurable services in online systems. To support this transition, the KB of the MD-KB architecture can be a distributed, replicated repository for the service 3D surface by maintaining the reconfiguration frame for each service workflow constituent. It is straightforward to modify the existing MD-KB to accommodate this new model as well as to add the instantiation of the service in the time domain. When a service request is made, its associated service instantiation surfaces can be created and these can be mapped on the surface which represents the current configuration of the device. A simplified version can utilise the baseline configuration of the device rather than the real-time version for enhanced performance.

6.4 Reconfiguration versus End-to-End QoS

The appropriate partitioning of adaptation execution for a commercial mobile service is a difficult design decision and this decision is made significantly more difficult when placed within the constraints of QoS regulation. The nature of the adaptation is an important determining factor in where it might need to run to meet timelines. However, the degree of the adaptation is not necessarily known until the context evaluation provides the instructions for the degree of the adaptation. There are several papers that

deal with the issues and nature of this partitioning, but from an engineering perspective there are essentially three choices – server-side, proxy and client-side [5]. A truly distributed, middleware solution would allow real-time distribution of adaptation to an available platform which would be able to satisfy the request while maintaining the service level. Currently in practice, this is not feasible as the nature of the adaptation even in a simple degradation of content quality, or for example image decimation, is too time consuming and require specific functions to allow real time selection and dispatch to the appropriate available node for adaptation. Given this scenario, the concept of reconfigurable service within a quality of service constrained environment seems even more unlikely to be viable. It is important to consider carefully options in this area to determine if, in fact, reconfigurable service architectures and QoS agreements are mutually exclusive constructs. It is equally as important to evaluate the engineering implications and issues that arise when these two technologies coexist. In Chapter 5, it was shown that a service workflow for context adaptation using a knowledge-base repository provided the requested service being respectful of the service level agreement. The orchestration of the service is done with the help of a service manager which is capable of maintaining a view of the adherence to the service level agreement and making certain decisions when necessary to abbreviate the service to maintain the agreed level. This management level is necessary and sufficient to provide the knowledge-base adaptation service within the QoS agreed, but does not necessarily ensure that reconfiguration of hardware is possible.

A fundamental question arises when the notion of reconfiguration is added to the architecture, the question is the point at which the requested the service is no longer able to deliver its primary objective due to reconfiguration. For example, if a service request for a specific rendering of image is managed under a specific service agreement and that service requires extensive reconfiguration in order to deliver it with the desired quality, will the service agreement apply in the same way as if the service requires little or no reconfiguration. It becomes clear then that based on fundamental service-oriented systems architectural patterns, that when introduced within a QoS constrained environment, reconfiguration services should operate as separate, callable services[203].

Bundling of reconfiguration services as a combined set of process steps within the body of a higher level adaptation service, for example within an image rendering engine, will reduce the ability of the adaptation middleware to make decisions about the execution of the service within the desired service level. The nature of reconfiguration as outlined in the earlier sections of this chapter are not simple and there are several "subservices" that when combined make a reconfiguration workflow event. The ability of the service manager to oversee not only the functional service workflow but the non-functional reconfiguration services of which context adaptation can be a subset will be important in an SLA-constrained environment. This migration to a separate reconfiguration and adaptation service workflow is consistent with the principles of separation of concerns prevalent in software architecture design. With the reconfiguration service separated into an invokable service workflow, the implications on adaptation will be as follows:

Service Time – the reconfiguration service workflow will need to be managed under estimated time to complete. In order for the service level to be maintained, the reconfiguration must take place in a time period governed by thresholds so as to permit the adaptation to take place to the extent necessary. Evidence from research using a realistic hybrid reconfigurable device environment for a wearable computing scenario, show that a configurable data size of 1-4MB results in a possible reconfiguration time of from 12.5 ms to up to 50 ms [204]. This argues for a mechanism to bundle the discrete reconfigurations and necessary modifications into one reconfiguration event to avoid time consuming multiple reconfiguration events as pre-processing steps to context adaptation.

Service Quality – the implications of reconfiguration on context adaptation service quality is seen in the fact that reconfiguration is a non-functional, pre-processing step intended to improve the quality of the consumer requested service. If the time to reconfigure is overly long or if in fact, this setup time does not permit the completion of the context adaptation under the constraints of the service level then the positive quality effects of reconfiguration are lost and the overall planned quality of the service delivered will suffer.

It is proposed that reconfiguration services will operate as a separate service workflow and will act as a context pre-processing step to the context adaptation workflow. In this pre-processing, the reconfiguration workflow will operate under the guidance of the service manager, and will be responsible for establishing a platform on which the service will operate and about which the context adaptation will execute. The engineering requirements for this QoS-aware reconfiguration and adaptation are:

- The time "budget" of the reconfiguration must be established carefully to ensure that the necessary reconfiguration action can be complete and the platform left in a stable, useable state. The reconfiguration actions must not be wasted meaning that the context adaptation service must be able to run in the time remaining otherwise the device has undergone reconfiguration and the adaptation will not be able to take advantage of it. An estimate of the end to end service delivery time is important to acquire during early service prototyping and into beta testing phases. This process should be further defined and include in future service engineering methodologies.
- The service workflow manager who oversees the request execution and overall service delivery will utilize estimates associated with the service maintained in the KB and information collected in testing. The service manager can use this information compared with the service level agreement thresholds to determine alerts and decision points for altering the execution of the service. The operation and decision-making process for service workflow manager or middleware is an area for further research and exploration.

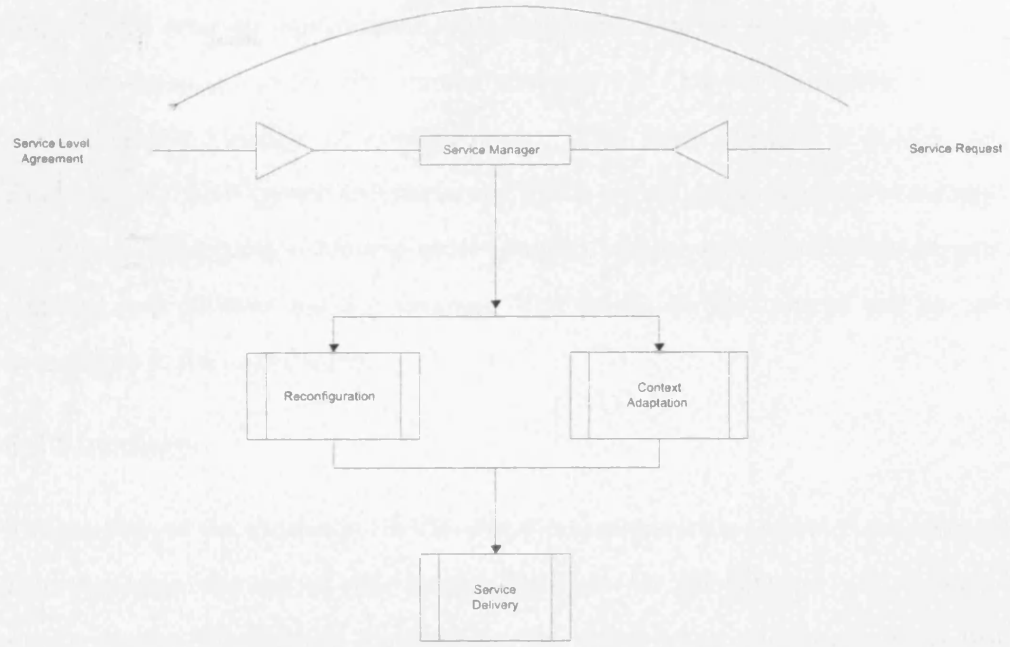

Figure 6-5: Reference Schematic for Service-Oriented Reconfiguration

In Figure 6-5 is a simple schematic of the overall new model showing the separation of the reconfiguration service and the need for the service manager. This service manager utilises both the service level agreement details which are usually provided in XML format [49] and the service request response from whichever service directory is most suitable. By comparison, the most similar work to this found in recent literature, is that found in [205] whose research involves the use of agents and component-based systems that utilise reconfiguration and context-adaptation in a distributed systems environment [205]. Like the work described here, Brandt and Reiser use a central repository, although not a knowledge-base, to store classes and profiles which are then used to provide an object-oriented contextual adaptation system. Finally, it is important to consider the outstanding issues and research questions in engineering combined systems involving context adaptation and reconfiguration with service level agreement constraints. First, the extent of reconfiguration depends on current device configuration and the nature of the service requested so estimating the extent and therefore, the time for reconfiguration will be inexact, so adhering to SLA's that are written on a fine-grained level with any certainly will be difficult. Safe and variable thresholds, perhaps "fuzzy SLAs", will need to be established by each service through testing of prototypes and

beta testing prior to deployment. A second issue to be investigated is how to best minimize reconfiguration with context adaptation in QoS environments by maintaining some evolving baseline of configuration. This topic requires a greater degree of computational intelligence and reasoning in the overall service workflow manager as the conditions of integration become more complex and the set of constraints require a more sophisticated decision making process. The details of this concept will be presented thoroughly in the next chapter.

6.5 Summary

The purpose of the discussion in Chapter 6 is to present the notion of the implications of reconfigurable devices as the future platform for all Internet and mobile service deployments. Throughout the chapter, the major issues and implications this future platform approach may have on the knowledge-based context adaptation architecture and prototype is outlined. Reconfiguration represents an emerging technical approach which promises to provide user's with truly extensible and flexible computing devices, alleviating the proliferation of platforms and devices by supporting fundamental configuration changes in the field by the user after the manufacture. It was found that reconfiguration as a technical concept is broadly defined and suffers from the same overloading of terminology as context adaptation. In fact, context adaptation is found to be a subset of the highest level of device reconfiguration. By extending software adaptation into the lower levels of the device architecture, it transitions into the research area currently known as reconfigurable systems, or historically systems on a chip (SoC). It is clear that hardware-software reconfiguration at a deeper level supports and extend the value proposition of context adaptation by allowing systems to be composed in real time in order to optimally execute the desired service on behalf of the user. This concept of assemble to order platforms for service deployment incurs design, development and performance penalties as the complexity of the creation and maintenance of such systems, particularly in a mobile environment, increases significantly. Fundamental changes in real-time device configurations require careful management and control to ensure that the device is maintained in an operational state and that the fine-grain reconfiguration and adaptation events combine efficiently to form a coarse-grain

reconfiguration that does not conflict with the service objectives or place the device in some deadlock state.

The issues with the design of reconfigurable services within a service-oriented architecture are many. Yet while reconfiguration and service-oriented architecture are active areas of research in the academic communities, the combination of the two are not often mentioned. The future interaction between the two technology areas impose an even greater need for new knowledge, techniques and methodologies for hardware and software co-design as the reconfiguration capability now extends into the previously static hardware function layers of the system architecture having vast impact on how the overall platform will respond to new functions and operations. Of particular interest and note in the design space, is the ability to visualise how these reconfiguration events will occur based on service requests, the multiple system layer impact of reconfiguration and the spatial distribution of functions in hybrid reconfigurable processors means a higher order visualization is necessary. Mapping the designed application service using surfaces of required reconfiguration allows the designer to quickly infer the nature and extent of reconfiguration imposed by the service workflow and this in turn can provide insights into the design time behaviour of the individual adaptive services. Combining the required reconfiguration surfaces for several simultaneous service workflows is the next step in the analysis. A method is presented for establishing the baseline configuration of the mobile device establishes the starting point for the combined analysis of service reconfiguration requirements. Each service then can be requested and the resulting reconfiguration requirements are used to generate an instantaneous surface representation. All service surfaces can then be combined through spatial analysis and animation techniques to visualise the instantaneous reconfiguration set for all services requested above the baseline. The use of an optimal baseline configuration, the set of all consistent functions, is necessary as the extent of the configuration information will exceed the engineer's ability to understand the volume of information depicted. This baseline configuration is recognized as a compromise that enables this first method to provide a meaningful representation. Further research is necessary to develop the visualisation technology to the point where the multiservice mode of operation can be

presented properly without a baseline configuration. The need for integrated tools for reconfigurable software and hardware co-design as well as overall systems analysis tools and performance tools for the reconfigurable services has been widely recognized. The need to reduce the complexity of the information resulting from these tools and associated analysis tasks will be important to enable the growth of the service-oriented, reconfigurable revenue generating systems. New design visualisation techniques using a 3D surface which represents required reconfigurations to a baseline configuration of a device can provide engineers and designers with an early assessment as to how their service will perform against a set of test case configurations in the field. This method and analytic data can then be used to set and tune reconfiguration service workflow operational parameters so that the core service, perhaps content adaptation, can be delivered within the time constraints of the user service level. The use of service instantiation surfaces and the shapes that would result from a multidimensional visualisation would then allow service characterization based on the required reconfiguration shape. The ability to use representative shapes to characterize reconfiguration requirements for specific requested services will enhance the engineering effort by providing the means to visually detect service reconfiguration conflicts by inspecting the surface interactions.

Important in this chapter are the discovery of architectural implications and the definition of the modifications necessary to the MD-KB and the C2Adapt2 service which result from analyzing the impact of reconfiguration on these designs and prototypes. The operation of reconfiguration within a QoS constrained environment requires a level abstraction for control and management of the reconfiguration and adaptation. This level of abstraction requires intelligence and some degree of reasoning so that decisions can be made within the constraints of the environment. Bundling reconfiguration as a sub service of the adaptation workflow was analysed and found to be inconsistent with the future needs of the service adaptation system. If reconfiguration is the paradigm of choice in future mobile platforms, then reconfiguration and adaptation must operate and interoperate as separate workflows with checkpoints so that partial reconfiguration and partial adaptation can be supported. This partial operation is necessary once QoS time

and quality constraints are implemented and the operation of the service is regulated by them. The knowledge of the service level agreement for the service at the time of request means that the service level information must be accessible in real-time to the service provider. This service level information also should include threshold timings on reconfiguration and adaptation service components which are acquired in the engineering evaluation and performance testing phases of the service engineering process. The concept of including upper and lower bounds on the execution elapsed time of sub-services of a reconfiguration adaptation workflow is well supported in the emerging process execution languages and web service orchestration languages and can efficiently be included in the future engineering of reconfigurable services as well.

Engineering service-oriented architectures to withstand the technological evolution of the future means looking out at the currently emerging technical innovations, evaluating the customer demand and need, as well as investigating other of the most likely market drivers to determine the most likely significant reconfigurable hardware with adaptive software for mobile devices is one paradigm to consider when architecting and prototyping an application service. There are of course, many other possibilities and permutations to consider as well. The finding of this work is that reconfiguration as an emerging technology will be the most impacting new technical approach to affect the mobile device service engineer and therefore, including a chapter on its nature and implications was most appropriate. The next chapter will take the question of mobile device baseline service configuration a bit further to explore how to limit reconfiguration time and minimise events which impact the ability to deliver within the service level agreement. Additionally, the chapter will outline problem solving techniques for how best to configure a fully reconfigurable device for the evolving needs of the user and how, by adding device configuration history, the overall system will be able to compose the optimal baseline configuration which acts to minimize the number of reconfiguration events and the length of reconfiguration time.

Chapter 7 Co-evolutionary Computing for Knowledge-based Context Adaptation

As discussed in Chapter 6, reconfiguration will act as a force to increase the dynamic nature of capability in mobile devices. By increasing the ability to change the basic configuration of a device through real-time distribution of software, dynamic collection of contextual capabilities will increase. This is, of course countered by the needs of performance, and specifically counterproductive to the performance of a system that is tightly regulated by a service level agreement. The problem is that the service provider will need to be constantly collecting and assessing the current configuration of the device in order to provide the best quality service. Software defined radio and device reconfigurability provide a new metaphor for context adaptation asserting that the very nature of the attributes and connection technology of a device can be varied over time and specifically changed to suit the required service interaction. As John Holland describes when he details that nature of genetic algorithms which are computational models of biological evolution that implements "perpetual novelty" [206]. This term clearly describes the continuing variation of devices, services, systems and interactions that the environment of mobile networks embodies today. Given this, in what condition will the device be left and how can this baseline condition be optimized to reduce the extent of dynamic context collection and future reconfiguration that is necessary? Minimising the amount of reconfiguration necessary for a user means that the amount of dynamic context will also be reduced. The problem is then to determine the optimal configuration of the mobile device called the baseline state that will act to reduce the possible future reconfigurations that are necessary. There are several methods known in the computational science literature that are used to determine optimal solutions to complex problems. One such method which is often used is that of evolutionary computation or the process of using a mathematical interpretation of biological evolution as a tool for solving computer problems. Evolution is an unsupervised and uncoordinated search through a space that extends over normally vast amounts of time. The power of evolution can also be its major disadvantage. This power is in the subtle

accumulation of adaptations or changes that take place in a system through many tiny almost imperceptible steps. The disadvantage in this is that the accumulation is not necessarily guaranteed to result in the most efficient or pragmatic outcomes. However, the mechanics of evolution are particularly well suited to the domain of problems where there are multiple independent variables all changing even slightly in relation to a dynamic environment. This is highly relevant in this research programme as a means to reduce the second most impacting operation in the system, dynamic context collection. Evolution and the simulation of evolutionary processes can provide some insight into the nature of a dynamical and complex ecosystem. In this chapter, the concept of using evolutionary search as the means to find the optimal configuration for the reconfigurable mobile device will be presented.

To understand why evolution provides the appropriate approach to this optimisation problem of service – device capability mapping, it is worth looking at the nature of evolution itself. Taking the postulates from the famous "The Origin of Species" completed by Charles Darwin in 1859, we have firstly, that "individuals within the species are variable", commonly referred to as variation. The second postulate is "some of the variations are passed onto offspring", called inheritance. The third postulate, "In every generation, more offspring are produced than can survive", also known as abundance. The final of Darwin's postulates is "individuals with the most favourable variations will reproduce or reproduce the most" which is at the heart of the evolutionary theory. The evolutionary process strives over time to reach an optimal solution given a set of environmental conditions. This statement encapsulates the desire in this last phase of the context adaptation work, to seek a method that provides an optimal configuration for a given environmental conditions. Searching a solution space using an algorithm such as a genetic algorithm or co-evolutionary algorithm, is not automating the evaluation of all solutions but is an effort to add efficiency into the process of matching device configurations to multi-objective service requirements. This is a feature of evolutionary algorithms, the ability to find an optimal set of solutions via a limited search of the possible space [122]. One of the difficulties in creating the appropriate algorithm is in determining the most likely environment condition that a

given service or device will encounter, this is the solution space. To help overcome some of this difficulty is that due to the manner in which the C2Adapt2 system is designed, the service provider will maintain information about the configuration of a successful service experience, this will become the historical pool of successful solutions. Each instance of a mobile device configuration is matched to the service delivered which essential creates over time, a pool of successful individuals representing configurations. Using evolutionary principles, the reconfiguration service can then successively "select", the more advantageous configuration based on the service history of the user, in the environment with a given device. The purpose of this last phase of the engineering research programme is to explore and investigate an evolutionary method for providing an optimally matched mobile device configuration. A key challenge when designing and deploying such systems is the ability to understand and predict the nature of the device landscape that will be faced by the application. It is possible then to share the configuration baseline with the service provider through the KB with the goal to help provide better services. Returning to the central hypothesis of this work, these enhanced services will drive increased value extraction from the market and that context adaptation provides a clear means to enhance services as long as it remains within the agreed service level. These basic premises are consistent with the motivation for a reduction in the extent of dynamic adaptation required. Establishing an optimal baseline state for a device given the current and historical service usage profile is one concept to provide this reduction in dynamic adaptation. In many ways, dynamic context is equivalent to reconfiguration requirements. In dynamic context collection the device is reporting its current virtual state and in reconfiguration the service provider will essentially be establishing the virtual state necessary to advantage the service to be delivered. Given this hypothesis, if one can routinely reconfigure the device to an optimal state, then the extent of service specific reconfiguration will minimised. Logically, this implies that the extent of dynamic context collection will also be minimised as there will be a minimal amount of new information on which to report.

There are a number of methods to improve dynamic context handling and in particular caching is often used to help with the performance of dynamic systems. While caching

of dynamic context is a worthwhile design pattern for today, caching in the future world of reconfigurable systems-on-a-chip is unrealistic. In reconfigurable devices, far too much of the fundamental functionality of the device will need to be in the cache to make it feasible. Caching of large records is an expensive operation, and there is additional time needed to transmit these cached records, this will again reduce the roundtrip performance. At a high level this does not serve to increase the overall performance of adaptive system. Reducing the amount of dynamic contextual information that is required for the service to be properly adapted is a complex and difficult computational problem. It is directly at odds with the notion of context adaptation and its benefits. In many ways, dynamic context is unavoidable, e.g. the environment in which the service is being used will change with the whims and needs of the user. However, the elements of dynamic context that are implemented in software; those changes that occur through the interoperation of a mixed portfolio of executing services are theoretically avoidable, or more realistically, are capable of being significantly reduced. In *software as service*-based reconfigurable system which supports multi-level reconfiguration, it is feasible to construct a device configuration profile which represents the union of the most common components utilised in a stable state. This configuration concept is called the reconfiguration baseline and will vary depending on the usage profile of the user. The most interesting objective behind the concept though, is the possibility to increase overall performance by reducing the need for extensive reconfiguration by creating an optimal baseline configuration for the device given a historical view of the deployed services.

The main question addressed in this work is how to construct such a configuration profile with more intelligence and adaptability than the user agent profiles found in common computing platforms today. As stated, this problem can be seen as one of intelligent optimisation within time constraints. The focus of this chapter is on using optimisation techniques to develop a feasible reconfiguration baseline given the dynamic nature of changing devices and service requirements. In the following sections, the optimisation problem will be analysed and an algorithm will be proposed to meet the needs of the key aspects of the baseline configuration, that of matching a configuration

with the requirements of the service. From there, a small example of such an algorithm will be constructed and a brief assessment of the result will be provided.

7.1 Optimisation for Context Adaptation in Reconfigurable Environments

In the earlier chapters, the nature of the device capability and service interaction has been presented. This interaction is governed by the consumer request, either by a human user or a machine, which results in evaluation and adaptation for maximal value given a specific set of environmental constraints. In chapter 6, reconfiguration at several functional layers of one aspect of the environment, the device, was added thereby increasing further adaptive complexity of the entire system. In such complex systems, it is rarely if ever the case that one set of parameters can be determined that optimised all aspects of the operation simultaneously. Determining an optimal set of parameters for the adaptation service objective falls in the category of a global optimization problem where there are many variables and many potential but sub optimal solutions [110;207]. The context adaptation service set in a future of low level reconfigurability presents a problem space where there are significant populations of mobile devices and services evolving independent of each other. These populations when brought together to form an adaptive system instance will have potential for large numbers of local minima solutions which tend to make the solution of these problems difficult. There are many methods available for use in hard global optimisation problems, most of which attempt to determine the optimal set of solutions based on a priori knowledge or smoothing functions. This global optimisation exercise is then one of searching through a large complex space for the set of composite device and service states which minimises the costs of adaptation. For such problems there are essentially three commonly used techniques, simulated annealing, genetic algorithms, and clustering methods [208]. In reviewing the literature on these three methods it was concluded that evolutionary methods were well suited to the unique nature of the problem at hand. Simulated annealing is shown to have less success in a diverse solution space such as envisioned here and clustering, an older method, is known to have problems with inefficiency as local optima can be repeatedly selected [209]. There was a second rationale for the use of a genetic or evolutionary algorithm approach. As mentioned in the chapter

introduction, there is a strong similarity in operational dynamics between the mobile device market place and context adaptation services as an analogue to co-evolutionary interdependence of species. Each component of the problem needs to evolve functionality and feature independent of a specific request or response service. When a request is made for service however, they must come together and adapt such that they can meet the needs of the user in an efficient and cost-effective manner. This intuitively supported the use of evolutionary optimisation techniques from the perspective of a qualitatively accurate model in addition to a well-founded mathematical premise. Given this, an overall optimisation approach and model was devised and represented in the language of evolutionary computation. The central matching algorithm was designed and developed as one point of proof for the system operation. The basic concept of the optimisation subsystem designed for this component of the project is depicted in Figure 7-1 below.

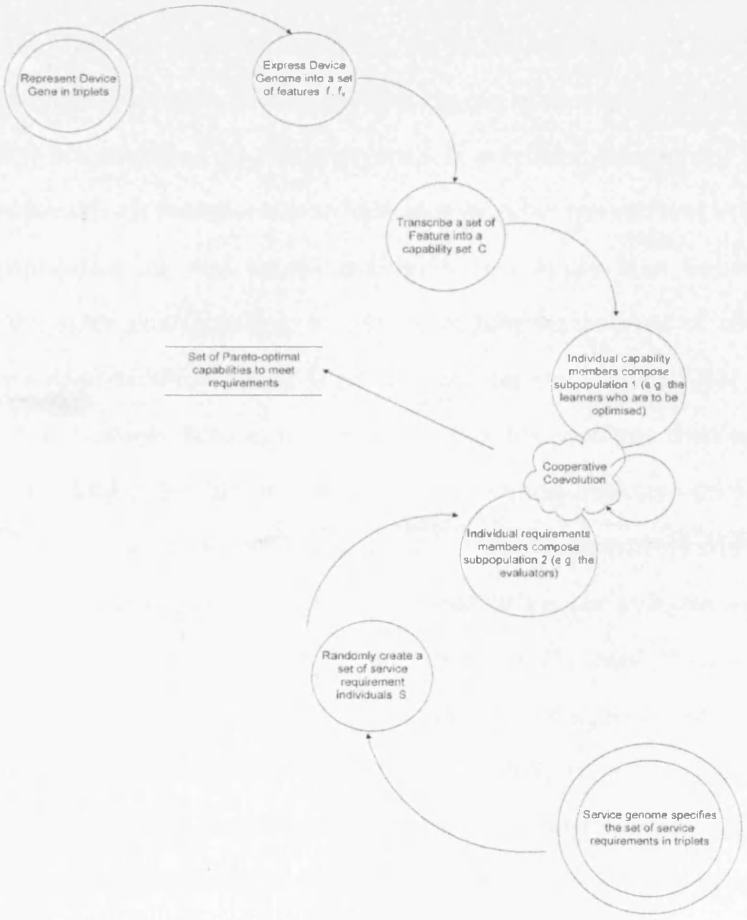

Figure 7-1: Co-evolutionary Context Adaptation Subsystem (CoCA)

The figure shows clearly the two population approach, one based on the service needs starting from the bottom right and the other around the device capabilities starting from the top left. This subpopulation structure provides an additional benefit in the fact that it allows the nature of the population to evolve over time away from the context adaptation environment. Not all device and service interactions will require the contextual adaptation services at all times, or equivalently changes will occur to these population outside of subsystem operation. This design allows the population to remain independent and evolving while permitting them to join for the purpose of formulating an optimal service environment.

7.2 Co-evolutionary Engineering for Reconfiguration Baselines

In this section, the details of the subsystem and algorithm are presented as well as the foundation from the supporting literature for a co-evolutionary approach to the optimisation question. In evolutionary computing, an understanding of the population dynamics which are involved in the ecosystem is a critical component of the overall optimisation approach. It is important to look at how other researchers in this field have addressed optimisation of real world problems and apply that knowledge to the optimisation of device configuration to service requirements goal of this work. For example, a two sub-population model is used in deJong and Pollack [243] which fits the context adaptation problem description. If applied to this problem, then one population representation would be the set of possible device configurations which are feasible; these are called the device baselines. The set of service requirements forms the second population. The device configurations are allowed to evolve independent of services based on evaluation and selection of those devices with most favourable common feature sets. The highest ranking devices in several sub-species categories are then evaluated by the set of all service requirements to determine the optimal candidate configurations. These candidate configurations are then post-processed for commonality and the components and functions in common are then selected as the principle baseline although the other candidates are maintained in the population as the configuration

"elites". Using the learner/evaluation co-evolution approach allows the service to essentially reward those device configurations which are most promising to the enable the service's functions [119]. Partial acceptance of a device configuration does not imply that no reconfiguration will be required just that current state of the device represents a position where the reconfiguration will be minimal therefore the fitness for the service will be maximal. The use of co-evolutionary algorithms in place of the more traditional genetic algorithms was the result of research into the current state of the art and a comparison with the problem domain. Initially the concept was to evolve the device capability independent and have it optimised against the static set of context requirements. The issue with this approach is that the service context requirements will change and in fact, may change dramatically based on the real-time environment. Over several months, other mechanisms for applying evolutionary theory to real-world problem solving techniques were evaluated and several methods such as evolutionary strategies, and multiobjective optimisation genetic algorithms (MOOGAs) were analysed for applicability to the problem [210]. In all cases, some aspects of the approach could be used but they were not effective in handling the independent yet interconnected population variation that exists between services needs and device offerings. One of the key advantages of the co-evolutionary approach is the ability to use the attributes of one subpopulation as means to determine the fitness of the other interconnected population [117]. The primary concern with the single population evolutionary models is the subjective nature involved in the mathematical formulation of the fitness function.

Another benefit of using a co-evolutionary approach exists here and addresses the concern over mathematically sound and representative fitness functions. Co-evolution fitness functions can be designed to reward one population for either the comparison to, or the interaction with, another po .. where the interaction can be deemed either a negative or competitive one, or as a positive or cooperative interaction [116]. The ability to evaluate one population as it relates to the other is an important aspect of this work; in fact, this concept can be programmatically applied in the automated matching of devices with service needs. Once the decision to pursue a co-evolutionary approach was made, the question was whether a positive or negative interaction was to be modelled. In

nature, the evolutionary process is primarily thought to be competitive, hence the phrase "survival of the fittest" which implies that those most able to compete for certain resources are those that are selected for reproduction. While true, there are many instances in biological systems of cooperative or positive interaction which may affect the longer term viability of one species over the other. The system in question for context adaptation uses the two subpopulations of devices and their configurations evolving relative to or as a result of, service needs and requirements. This is a symbiotic relationship although there is clearly a competitive element within the subpopulations particularly if one considers the dynamics of the marketplace. Services, though, require devices in order to be used and devices require something of value to run in order to be useful. This argues for a cooperative relationship where one population enables the other and vice versa. Another important aspect of the device – service interaction also affects the algorithm creation. There is not a one-to-one correlation between a single set of service requirements, or objectives and a device configuration in most cases, except for highly specialised devices. The set of services required for one overall user service may be many and may operate across many layers of the systems architecture. The difficulty is how to evolve an overall configuration in response to the many requirements and sub requirements that may be imposed on it. A co-evolutionary approach offers some clear advantages in the selection of the most fit configuration for a large, disparate set of service needs. The important issue here is the ability to address the underlying service needs across the multitude of services that make up a service offering. This topic is central to co-evolutionary research as there are many real-world problems which are a composition of sub problems and underlying objectives which cannot necessarily be known or simultaneously evaluated by a CGA. However recent formal methods work in the area of evaluation in co-evolution has demonstrated that an ideal set of evaluations can be formulated which meets this need [115]. The inclusion of this proof and the exploration of how to include underlying objectives which are embedded within the user service requirements are left to future work.

7.3 The Genetic Plan

All evolutionary optimisation problems are best formulated using a concrete and parameterised approach. The genetic plan represents those parameters, functions and general setup that are used to create an evolutionary model of a real world problem. It is known that using the elements of a genetic plan is essentially the process of generating and evaluating structures, in this case, a structure chosen to be equivalent to a string representing a physical configuration [211]. The plan was designed as input to the automated heuristic searching for the optimal solution. It is important that all aspects of the plan are designed and implemented such that what results is a competent, converging set of genetic operations [100]. One of the most researched areas within the genetic plan is the issue and nature of how best to represent the unique nature of the individual solutions within the problem space. The oft cited references for a clear and concise statement around genetic representation can be found in [111] and [121]], where the issues around the hand engineering of appropriate representations are discussed. For the purposes of this chapter, a preliminary investigation by the author was completed in the use of genetic algorithms to describe a genetic plan and representation which could be used in experiments around evolving a device configuration [212]. The device ontology described in Chapter 4 is used to produce a basic genetic representation of the genome is one step towards automating the representation development. Researchers have recently reported on using a genetic algorithm to evolve a representation, there are still other methods other than evolving or hand coding the representation [120]. The method described here relies on the ontology generated from the device and service marketplace itself to provide the basic genome framework. This genome is then encoded as described in section 7.3.1. The implications of this ontology-based representation are several and in many ways understanding the performance of such an approach is a complex research question; one which will require significant additional research and experimentation. Nonetheless, the first steps in this area are to design an evolutionary strategy and approach which can be used for experimentation and subsequent enhancement.

7.3.1 Representation

The act of representing the unique individuals in the search space is often called enumerating the search space [103], p.51. There are an infinite number of ways to represent an individual solution in the real world, the task of conversion mapping the important aspects of the potential solution from the physical, real world into the computational work where abstract elements such a bits or 4-tuples can take the place of tangible entities is both art and science. This conversion to the abstract world must occur in both the problem domain and the possible solution domain with some relative interdependency between the two domains so that there is consistency in the translation. The representation approach taken here is to map the ontology which already encodes the physical world into a virtual representation and to again translate that into a numeric form that is suitable to standard computational evolutionary processes.

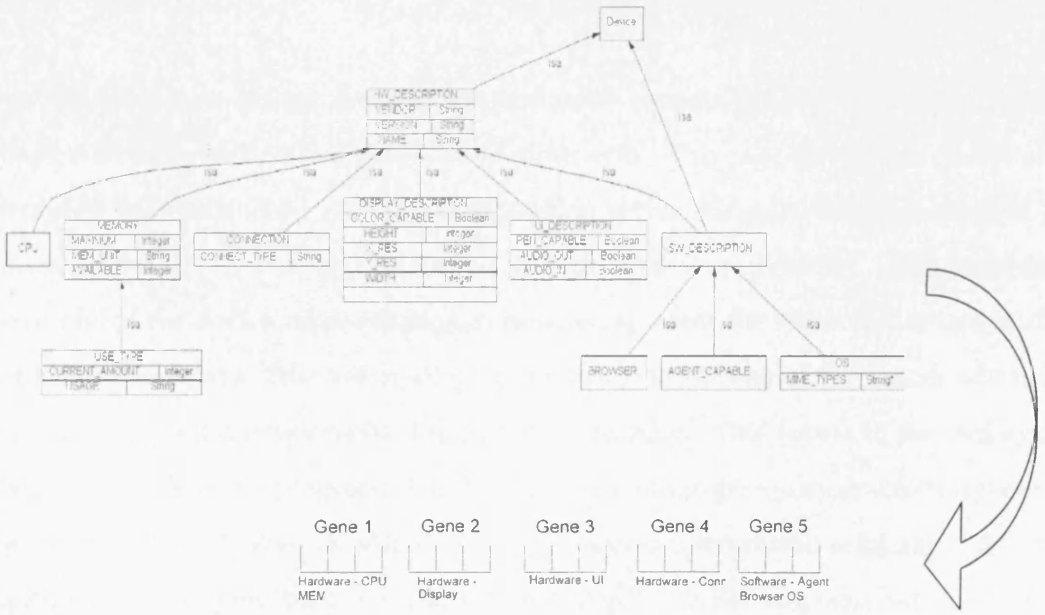

Figure 7-2: Ontology to Genotype Representation

This is followed by the genotype to phenotype conversion which results in a physical description of the configuration of the device. It should be clear that this physical description of the device is a configuration, not a physical appearance or operational description of a particular device. In this way, the configuration can then be evolved to

represent the details of how the reconfigurable device can be instantiated regardless of the nature of its physical or real-world appearance. The way in which the configuration manifests itself based on the information encoded within the genome is known as the resultant phenotype. In this component of the work, the objective is to search for the set of most fit genomes that will map to the service requirements. This will result in a configuration which should be most optimal for the execution of the service. This optimal execution environment is the phenotypic representation which results from the most fit genotype. Once again, the analogous process of going from genotype to phenotype in a biological system is far more complex than can be represented mathematically and in many ways, the true nature of this process is largely unknown for higher order species. Many simplifications to the evolutionary process are required here in order to use it in the search for solutions, so the extent to which the genetic plan mirrors the natural world is not truly known nor is it the basis for the utility of the algorithm.

The representation chosen for these experiments consists initially of a fixed length genome for both the device and service requirements. This genome consists of five sets of triplets with each allele, an individual position within the gene, having a possible 16 values in the 0-9, A-F or the traditional hexadecimal representation. One troubling question for the device representation is how to represent the structural nature of the attributes themselves. This occurs due the positional sensitivity of the values where in the genome specific positions having semantic meaning. This occurs in the biological genomes as well as it is proposed that there are start and stop sequences which represent the beginning and end of encoding for a specific structural attribute. In the representation designed here, the use of the full stop, ".", is the structural delimiter. This representation is consistent with and inspired by the genome composition derived in [213], where genes are defined as the triplet values maintained between the full stops, an allele is an individual element within the gene and the genome is the total structure inclusive of the position of the full stop itself. While the interpretation is simplified by using this genomic punctuation, the implication of these non-coding regions will be discussed in the section on experimental results. One immediate implication of

embedding structural information through position is that the representation immediately starts to deviate from its biological analogue. For example, in the biological system of genomic representation, the genomes are independent of position [214]. Although there are genetic algorithms that implement genome position independence, the problem domain for device configuration requires this structural information. Another method to encode this is to have a set of start and stop indicators which are unique to the structural element being encoded. This may be closer to the biological model but results in a very large genome and one which may be computational infeasible given the resource available to this project. An example of the encoded device and service requirement genome is given by:

servicegenome=encodeSentence["8DF.4CE.AA3.010.4EF."]

Figure 7-3: Genome representation

As detailed above, this string is the objective of the device evolutionary search process which is formulated from the service requirements. The process is approached as a structured string matching algorithm which is both computationally efficient and representative of the problem when approached with this style of encoding. The use of the string matching algorithm was a concept formulated based on the cumulative selection work outlined by Jacob in [101]. When designing a software service, requirements are often modularised and this module design can be reflected in the coding sections of the triplets of the genome in a manner similar to the way in which a human genome is composed.

7.3.2 Selection and Mutation

The next component of the genetic plan is the model of how selection and introduction of random variation will occur in the subsystem. The characteristics of an evolutionary theory based search system are straightforward and actually quite simple. Any evolutionary computation (EC) system must have essentially three modules, a facility to store the solutions of interest, a mechanism to evaluate and select potential solutions and the means to vary the solutions in a controlled manner repetitively over successive

generations [215]. The simplistic nature of these algorithms is deceiving however as the meaningful results depends on many complex factors and in order to be useful, it must closely represent the mapping between the search space and the evaluation space. The difficulty lies in the fact that in order for the algorithm to provide a truly Pareto optimal set of solutions, e.g. baseline configurations, it must not only work right and converge to a set of solution but the operation must model the important and affective aspects within the problem domain itself. A key component of the modelling of the problem domain is in designing mechanisms by which the algorithm selects members of the population that are promising solutions and then how the algorithm will vary those solutions in order to explore all possible solutions in a sometimes large search space. For the co-evolutionary context adaptation algorithm, the selection method is not fitness proportional, but is in fact elitism. In fitness proportional selection, the most fit individual is discarded to improve the exploration capability of the algorithm [100]. The algorithm outlined implements true elitism as the most fit genome is used as the basis of random mutation, but is not subject to the mutation itself. Every generation individuals are evaluated against the service requirements and the Hamming distance between the two genomes is calculated. The Hamming distance is a well-known and well used algorithm which compares two strings or more generally, any two sequences of bits and calculates to what extent the two sequences disagree [216]. In this system, the objective is to evolve a device configuration set which has a minimal Hamming distance from the specific service requirements set. The algorithm then adds this best performing configuration to a set of new configurations which are generated using the mutation operator with the specified parameters. This new mutated population then represents one known fit solution and the variations based on that one fit individual.

It is well established that searching for an optimal solution requires that the exploration of a large solution space is implemented through the use of mutation and in fact, the more expressive nature of the mutation operator the greater the variation across the length of the genome. However, the use of extensively large range mutation in the individual alleles appears to also result in a failure to converge. This fact will be seen and

discussed in further detail within the experimental results section of this chapter. The dominant role of the mutation operator in this algorithm design is as the primary mechanism for variation which means that the parameters of mutation need to be tuned carefully or the algorithm will converge to a local optima, or in many cases will fail to converge at all. The mutation operator here is different then in many of the advanced evolutionary algorithms as it takes the place of the reproduction operator, which gives it a very prominent position in the overall process. This is a feasible design choice for the case of this proof of concept experiment, it is anticipated that in subsequent efforts a standard crossover operator could be used although as will be discussed in the results section, this approach appears to function efficiently and is suitably robust. The mutation operator as implemented contains two parameters that affect how the subsequent populations are generated. It maintains both a probability of mutation and a neighbourhood or "radius" of choices for the mutation process. It controls both the chance of the mutation and the range of values which the mutation can span. This is an important deciding factor in the choice of this style of algorithm for this work as the range of values for mutation has specific meaning in device configuration. It must be contained with the representation or else the random generation of new possible solutions could result in a configuration which is largely infeasible. The precedent for carefully determining the complete range of possible solutions to allow within the subpopulation is well established, and in effect, this design parameter is equivalent to determining a subset of all possible mating combinations so as to prevent possible suboptimal combinations to randomly form [125]. The danger in this approach is that by excluding some combinations to form, the search space will be unnecessarily constrained. In the evolutionary subsystem outlined in this chapter, there are two areas where this may have impact, first in the subpopulation evolution where each subpopulation is evolved independently in an "island" approach and second, in the range of constraints on the mutation possible in each device configuration solution.

7.3.3 Evolution Process

With the basic plan outlined and several of the initial parameters defined, the central algorithm was then developed. The model of the context adaptation co-evolutionary

process, or CoCA, is broken into two phases, the evolution and selection of the most common genome in the service requirements occurs first generating the objective genome. The second phase is to evolve the set of device configurations which will most closely match the service requirements. The evolutionary process for each population was chosen as a standard approach which first creates a random initial population which represents the first generation of solutions. In the experiment, 50 individual solutions were created in the subpopulation and evaluation is then performed on the initial population to select the solution with minimal number of alleles that disagree with the service objective genome. As discussed above, the solution with the minimal index or alternatively the highest fitness is maintained and used to create 49 more individual solutions which are again evaluated. This loop continues until the evaluation reaches 0 alleles which disagree or until the number of generations becomes excessive indicating the inability to converge to an optimal set. The algorithm implements elitism by maintaining the most fit solution through one generation although it evaluates a solution with a higher level of fitness, then the previous elite member is discarded in favour of a new generation based on the new most fit genome. The important point about this evolutionary process is the termination condition can be adjusted to reflect the rigidity required for the matching of the service to device. In many evolutionary algorithm applications, the objective is to arrive at the set of most optimal solutions in the minimal number of generations to conserve resources. While this is certainly true for the CoCA, it is also true that the objective is to reduce the extent of dynamic reconfiguration necessary but to still deliver the adaptation within the service level agreement. The implication here is that the termination condition is adjusted so that the algorithm continues until the best solution is found or until the solution nearest to the best is found within the service time allocated to the matching operation itself. In the CoCA-based evolutionary process, there is no concept of a non-converging solution as the assumption is that it will return either the optimal solution or a better solution than the random configuration within the specified time. The evaluation as to whether the solution is a better configuration is done in the tuning of the algorithm rather than through an online, real-time check process. Due to the calculation time constraints envisioned on the system, it was necessary to do offline tuning rather than online validation.

A series of experiments were run using Mathematica 5 and modifications to the notebook for Chapter 1 of Illustrating Evolutionary Algorithms by Christian Jacob. The purpose of the experiment is to understand how the device configurations might evolve relative to the service requirements. This understanding can be used in future work to develop a device to service matching web service based on this evolutionary string matching concept. The issue with current online service to device matching services is the inability to make automated trade-offs between the individual objectives as to their relative importance. The current practice of service negotiation uses XML profiles and priorities within the service requirement specification to provide the means to decide if a device is capable of fulfilling requested service to the level of service required by the user. This structured method has two major drawbacks, it lacks dynamic extensibility as discussed in Chapter 2 and it lacks the ability to easily accommodate conflicting underlying objectives without some programmatic intervention by the developer. To test the viability of using a co-evolutionary process to evolve a device configuration baseline for service, several existing methods and tools were studied. There are a multitude of resources for experimentation in traditional and multiobjective evolutionary algorithms in particular [106;217;218]. After a review of many of the tools found in these pages, a subset of systems was formulated which appeared to match the resources and skills available. The tools which were investigated in further detail for this experiment were:

- Distributed Resource Evolutionary Algorithm Machine (DREAM) - provides a set of tools and infrastructure support for the development and tools of complex evolutionary in a truly distributed networked environment to industrial platforms[108]
- EOS from BTExact Technologies - a framework developed in Java that allows research and implementation of evolutionary algorithms. EOS also supports the simulation of ecosystems and the visualisation of the results of such simulations through the use of a spatial environment [104].

- ECJ from Sean Luke at George Mason University – a flexible, platform independent development toolkit for a design and implementation of a variety of EC solutions. ECJ is also written in Java [109].
- Evolvica from Christian Jacob – a set of Mathematica notebooks and tutorials which demonstrate the principles of various evolutionary algorithms and optimisation techniques [107].

Several other tools and genetic algorithm libraries were reviewed for use in this proof of concept, tools such as GALib from MIT, SPEA2 for multiobjective optimisation from Zitzler, et al at the Swiss Federal Institute of Technology (ETH) and Generic Evolutionary Algorithms (GEA) from Kokai at the University of Erlangen-Nurnberg. All the tools reviewed had specific applicability in certain areas of the problem domain for dynamic context collection reduction, but in some cases the environments and their associated toolkits were designed for more basic research rather than industrial application. An iterative approach of evaluation and test was repeated for each tool. All software with the exception of EOS was installed on both a Toshiba Tecra 9100 Pentium IV 2.4 GHz and Hypersonic Aviator GX 6 Pentium IV 3.2 Ghz machines. EOS was reviewed with the development team at BTExact Technologies at their facility in Martlesham Heath in the UK. An informal evaluation was made as to the ease of use for the purpose of the experiment between DREAM, ECJ and Evolvica. The DREAM system was more than capable of running such an experiment but given the goal of the experiment was not necessary at this point to test the distribution of the algorithm in the DREAM framework which has more power and network capability than was needed. The remaining two tools, ECJ and Evolvica were roughly equivalent in their ability to handle the experiment as designed. The ECJ toolkit is more versatile and provides an excellent development environment for designing evolutionary and co-evolutionary algorithms. The use of the parameter files as the primary method of interaction and setup makes for efficient tuning of algorithms. The advantage of the Evolvica notebook system is in the presentation layer where the notebook and the graphing functions are easily accessed and it provides an excellent, quick prototyping environment. Equally

important in industrial research is the ability to create a comprehensive view of the experiment in terms of future production engineering and to present that view in a manner that a non-technical audience will understand. Evolvica notebooks provided this capability without additional coding and the specific example of the Hamming distance as a method of fitness evaluation also fit well, with minor modifications, the needs of the experiment. Future work in creating a code optimised web service of the co-evolutionary device configuration service would be best implemented in ECJ however.

The length of the genome was fixed at 20, based on key matching attributes, with three triplet encoding the function and 5 non-coding characters indicating the structural aspects. The Protégé 2000 MD-KB provides the values for the triplets on the basic device configuration and the service requirements, a PAL query is made against the repository and the results are then mapped via a "transcription" table into the encoding. This table can be eliminated in subsequent versions and a frame could be added to the repository which includes the encoding triplet values. The next step in the further development of system will to add such an encoding service to the repository. The encoding service is not simply an additional frame to store the triplet values, but it would need to include the ability to interpret the value of the relevant of the device or service frame and create the codes in the encoding frame.

7.4 Experimental Results and Discussion

Using the CoCA algorithm setup, several tuning experiment were run to achieve convergence. By manipulating the mutation parameters, a set of four experiments were then run with a starting population of 50 individuals and the fixed length genome as discussed above. The results of the experiments are summarized in the following table:

Experiment #	Population	Mutation Probability	Mutation Radius	Fitness Attained	Generations Required

1	50	.1	1	1	56
2	50	.1	5	0	42
3	50	.2	2	0	35
4	50	.5	5	NC	4000+

Table 7-1: Device configuration co-evolution results

Of the best performing algorithms, the experiment 3 resulted in the generation of the matching configuration in the shortest number of generations. The un-optimised execution time was less than 2 seconds. Two plots shown in Figure 7-4 at different levels of granularity show the convergence behaviour of the experiment.

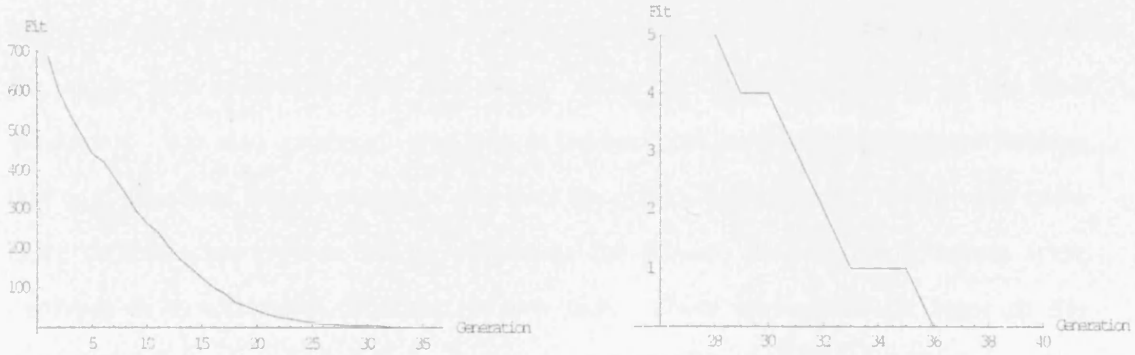

Figure 7-4: Best Performing Configuration or "Learner" Plot

On the left of Figure 7-4, the convergence behaviour appears as a steeply sloped but gradual descent. Looking more closely on the right hand side, the detailed view shows that the algorithm maintained certain fitness levels for a few generations before moving towards a better solution. As this is the fixed length genome example, the terminal solution is at a position in generation 35, when all components of the service requirements and device configuration genomes are identical. This gives the optimal configuration on which to deploy the service with the specific requirements. The assumption here is that all requirements are equally important and that allows the problem to be tractable in this first proof of concept phase. Analysis of the performance

of the other parameters can help to suggest the nature of the underlying objectives and can provide some idea about the problem domain itself. The poorest performer was in experiment 4 which failed to converge after 4078 generations with a fitness value at generation 4078 of 2. The most interesting observation on this experiment is that it very quickly moved to a position of relatively good fitness so after just 300 generations it acquired a fitness of 7, although this is considerably worse than the results of experiments 3, it did not reveal that it would take a further 2329 generations to acquire 5 more points in fitness. The interpretation of this is empirical and it would seem that having large changes in the values of the genome triplet elements does not necessarily mean that one gets a higher more beneficial exploration of the search space. This is perhaps the case initially but it seems to cause a greater inability to then return to find the global optima. It is quite possible that this can be corrected by retaining more of the higher fitness solutions and using some form of crossover rather than relying entirely on the single best individual and the use of mutation for the formation of the next generation. It is also constructive to look at the two best performing parameter settings and to understand the advantage of one over the other. Experiments 2 and 3 were quite close in their convergence but in reviewing the process of how the genomes were evolved, it is noticeably different in approach. There appears to be areas in the experiment 2 genome that recur or "stick" for a generation or two more often than that which occur in experiment 3.

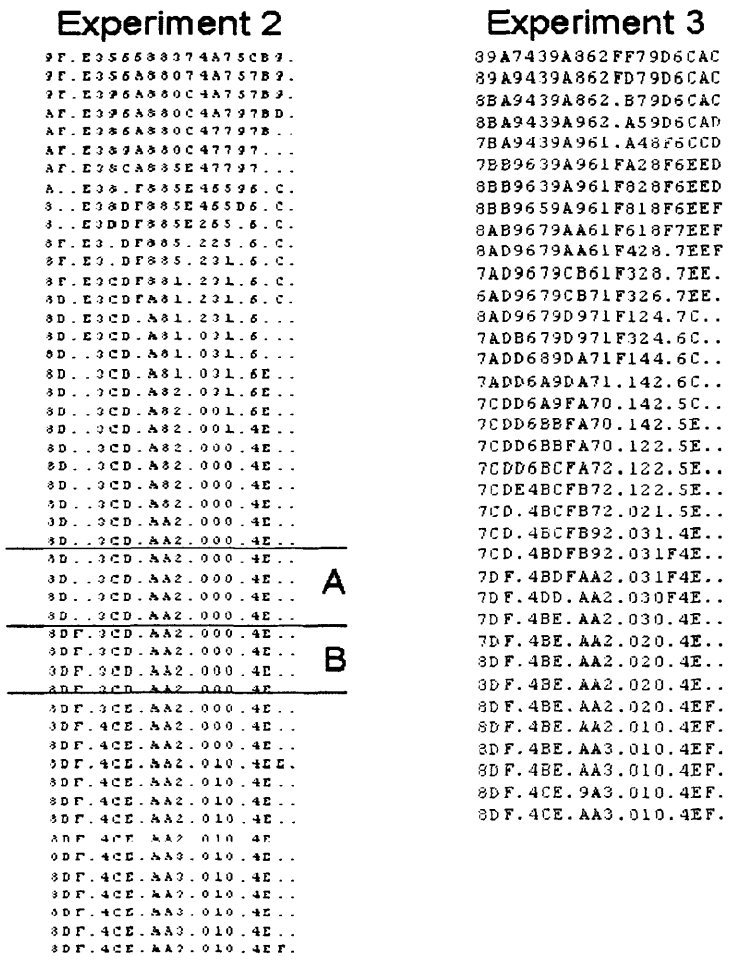

Figure 7-5: Comparative Analysis of Genome Derivation

Experiment 3 parameters provide for a more adaptive result as shown in Figure 7-4 of a more adaptive algorithm which is searching and adapting at a rate higher than 2. This is phenomenon is noticeable in the two regions marked A and B where there is a stasis of genome for several generations, which are ultimately selected out of the popu... This appears to provide evidence for the increased mutation rate which in fact, results in experiment 3 and a better performing algorithm. Further increasing the mutation radius however, as done in experiment 4 does not increase the performance but results in a set of solutions with a lower fitness index. This offline tuning appears to have found a reasonably good set of evolutionary parameters in Experiment 3.

While the work on evolutionary computation as a means for establishing a service optimised device configuration appears to have some promise, there are several important and significant issues that still remain. A key issue that was addressed is the need for a variable length genome that can represent the changing nature of the service and device. By accommodating a variable length genome in the design of the algorithm, the configuration of the device and the requirements of the service are then free to evolve as the market demands. The disadvantage of this approach however is the increases difficulty of the algorithm when a variable length genome is used. This is an active area of research within the evolutionary algorithm (EA) community. There exist several difficulties with variable genome representations, for example, the question of how many genes fully express the solution domain? How to add genes to the genome and maintain the positional meaning required for appropriate decoding? How to evolve the genome so that it expresses the unknown diversity of the possible phenotypic solution space instead of the known variation in the genotypic space? These questions are complex and address some of the deeper and more fundamental mysteries inherent in the evolutionary process. The issue that will need to be addressed for this work is not necessarily how the genome evolves as that aspect will be dictated by the evolving contents of the MD-KB, but rather how to evaluate the fitness of mismatched and variable genomes between the service and device configuration. In the case of the variable length genome approach to the problem, either population may have a slightly different length genome so that the problem becomes "open-ended" in the sense that learner population will seek to continually optimise itself relative to the evaluator but that there can be no specific terminal solution due to the probable length differences in the genome which result in a probabilistic rather than deterministic string matching fitness function [219]. A second major issue with this approach is the need for the hand tuning of the parameters in order to achieve convergence. Clearly, if this approach is to be used in an automated system then hand tuning for convergence is not possible and given the need for variable length genomic representation there is some evidence that suggests the parameters would be highly specific to each individual service requirement and device capability match event although this remains to be proven in subsequent work.

7.5 Summary

The concept of finding an optimal set of device configurations for a given set of contextually adapted services leads the engineering effort towards the techniques found in optimisation and operations research. The underlying requirements of online services present a set of objectives to which the device configurations must match in order to be perceived as "value added services". Multiple objective optimisation problems are common in engineering and operations researchers have recently applied evolutionary algorithms to this area with some success [220]. These facts provide suitable motivation for the investigation of the use of evolutionary algorithms for evaluation and matching in context adaptation services. The current thinking on performing such a matching involves structured statements such XML descriptions and the resolution of the statements against configuration specifications. While this is successful in some cases of prioritised service descriptions that are clearly articulated, it becomes very difficult and requires considerable software engineering to deal with all cases of service objectives and the possible device configurations. An alternative to this semantic matching process, is to evolve service requirements and device configurations as subpopulations in a cooperative ecosystem with evolutionary theory providing the operations necessary for matching. The co-evolutionary algorithm approach allows the two populations to change over time independent of each other, but when it comes to completing the request for a context service, they come together in a cooperative evolutionary process to formulate the optimal device configuration to complete the service. The proof of concept system, CoCA provides an example of how this might be done in practice by using the service requirements encoded from the MD-KB as an evaluator population against which the device configurations, also from the MD-KB, must optimise or "learn" better behaviour.

Although the CoCA subsystem concept is a simplification, it provides the operational information on how such a set of solutions might be formulated; the manner in which the parameters must be tuned to provide convergent behaviour and it also provides some valuable insight on the underlying problem domain. The use of hexadecimal

representation for the values of the features provides a visual indicator to the configurations although it is recognised that the length of the genome is not representative of the problem domain and many additional feature triplets would be necessary to provide the expressiveness needed in a production environment. It is also clear that the CoCA system would require the use of a variable length genome to achieve the original goal of the system which is to directly support the extensibility and dynamic nature of the mobile device environment when deploying context adaptive services. Constraining the genome to a fixed length essentially implies that all possible features and functions are known in advance and are fixed; this is in direct contrast to the observed operation of such systems in practice as outlined in Chapter 2. The support for a variable length genome would best be implemented through extending the knowledge-based repository which is described in Chapter 4. Through the addition of an "encoding frame" which uses an encoding service to evaluate the addition of new features and provides systematic inclusion of new these features into the genome itself. This functionality could be added as a separate module to the system, or even as a callable web service which enables the retention of useful intermediate information, a beneficial function with multiobjective problems in and of itself [112]. The benefit of such a new module would be in the ability to have new features added through the public maintenance of the knowledge-base and then the automated encoding of the new features done offline on some regular maintenance cycle. This would achieve the original goal of the overall context adaptation architecture, which was the ability to enable context adaptation with minimal developer intervention. The generalised system architecture derived from the project work including the evolutionary algorithm effort is depicted in Figure 7-7.

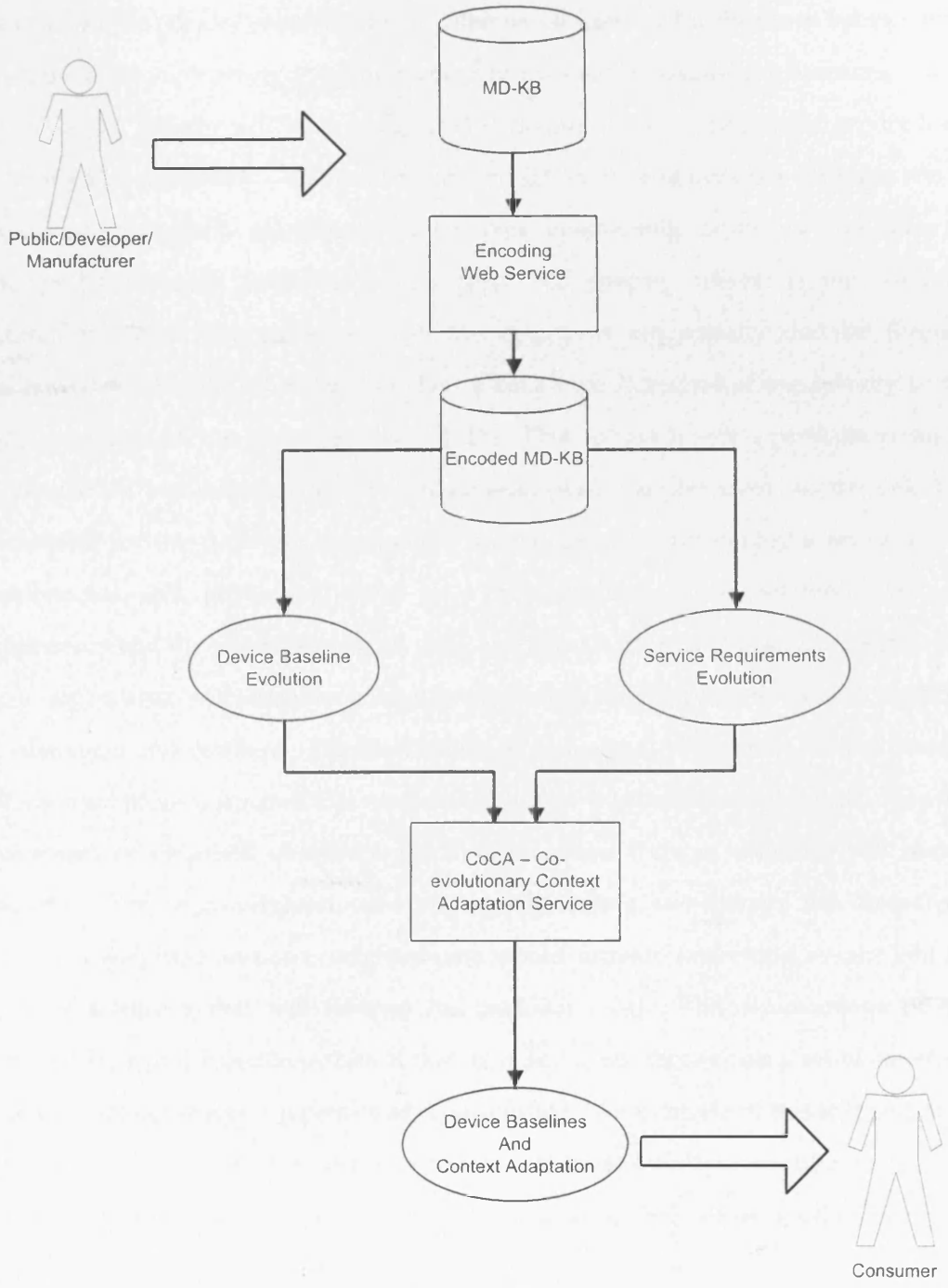

Figure 7-6: Knowledge-Based Web Services for Context Adaptation

Another potential issue with the CoCA design is that it specifies as a simplifying assumption that all objectives in the multiobjective problem domain are assumed equal in priority. Experience in service engineering and software engineering indicates that this concept is rarely the case. Service requirements will have priority and in current

systems, this priority of requirements is often relied upon as the means to balance time constraints. Requirements, or configurations in the case of establishing baselines, which are of a lower priority will often be ignored in favour of meeting a specific service level for response to a request. The need for prioritisation of objectives is a common one in applying evolutionary algorithms to complex engineering problems and there is considerable research available in this area. Of specific interest is the work on preferences where the preferences on the objectives are actually derived through evaluation of possible solutions and then a backward inference of the priority of the objectives based on the chosen solutions [221]. This approach seems particularly useful in the CoCA architecture as the marketplace place can be used as the selection mechanism for the preferred solutions. For example, by prototyping a set of service requirements and presenting those to a beta audience, the preference for each requirement can then be inferred and used as the basis for preferential weighting. The same can be done with baseline configurations using buying patterns directly specified by customers and retailers. The implication of using such preferences on the possible optimum solutions is an area that would need further evaluation and research. By using preferences or weighted objectives, the Pareto optimal front of solutions will change from that of the non-weighted case. As an engineering tool though, the comparison between a weighted and non-weighted case would provide interesting insight into the range of solutions that will address the problem [124]. The disadvantage of this preferential or prioritised approach is that, in many cases, there exists a set of objectives which cannot be properly represented or quantified. An example of this is found often in the non-functional requirements of services and systems, such as ease of use which are critically important to the value of the service delivered but are extremely difficult to specify.

A final point of discussion on the use of co-evolutionary context adaptation for service and device matching is the validity of starting with simplified genomes and non-prioritised objectives for use in fitness evaluation. The mobile device and online service market is a complex and extremely dynamic environment, one that requires both complex features and interactions. At first, it might appear that evolutionary

computation is poorly suited to the task of handling such complexity. The CoCA algorithm and its associated system represent a first proof of concept in this area, one that sacrifices complexity in favour of establishing a process and a method. Besides being a reasonable way to start by establishing such a proof of concept, there is additional precedence for the use of simplification, particularly as it relates to the genome itself. The process of evolution in biological systems is often surmised to have started with the simplest of life forms and on vast amounts of time, evolutionary processes have moved these simple individuals to a great level of component and system level complexity through a process of "complexification" [219]. This term is now being used in the evolutionary computation arena to mean the addition of new genes into the genomes, thus creating a more complex individual solution and that solution being translated into some physical manifestation. The difficulty with the biological analogue to this process is that the increase of complexity in biological systems is known to be accompanied with the addition of redundant or useless genes as the means to create new points in the search space to explore [222]. While the process of creating more complex individuals is a necessary next step, the process of doing so through the addition of redundant or unused genes is one that will be counterproductive to the CoCA design objectives. The use of redundant genes in the computational world of evolution would seem to incur a performance penalty for the solution search which easily outweighs the gain in creativity. The use of redundant genes is generally theorised to be for the purpose of experimentation in adaptation, which will be relevant in the proactive prediction of just what set of configurations and requirements are most beneficial in a specific market environment. The research question on complexity enhancement for the CoCA system can then be split between how to increase the complexity of the genome in a manageable way while keeping the algorithm tractable, and a second question about the inclusion in the more complex genome of redundant genes from which to experiment on new features and functions that could possibly evolve to support specific service requirements. In the first case, the current implementation is clearly not sufficient without a variable length genome to represent the changing features of both populations. Previous work in this area can be used for this additional feature including

the addition of a more generic approach to determining the appropriate representative variable length which properly expresses the problem domain [123].

The CoCA system and the experiments described here demonstrate that co-evolution can be used to create a fast and computational efficient matching algorithm for service requirements onto optimal device configurations given the simplifying assumptions made. It also fits neatly into the overall knowledge-base context adaptation architecture as the matching algorithm web service which provides the context evaluation necessary for the adaptation phase. The use of cooperative co-evolutionary algorithm embedded within the production environment of a commercially available service is a relatively new, but not unheard of, technique for optimisation. The technique is particularly useful in combination with distributed agent technology and is seen in several industrial application domains such as Internet search and retrieval, inventory planning and network traffic management [118]. The novelty with the approach outlined in this chapter is in its use as a set of interacting web services enabled through the use of a publicly populated knowledge-base.

Chapter 8 Programme Conclusions and Future Work

The purpose of this chapter is to present a summary of the contributions provided by the programme, to review and unify the conclusions developed in each of the technology areas and to present beneficial areas for future work. A brief critical assessment of the work will be done relative to the contributions made in order to provide the reader with a clear picture of the issues still outstanding with the approaches and results described. From this critical assessment and further analysis of interesting technical activities from each chapter, ideas for future work will be suggested and briefly outlined.

8.1 Summary of Contributions

Simply stated, the goal of the programme was to provide deeper understanding and technology to support the creation of higher valued context adaptation services to the mobile consumer. In Figure 8-1 is a high level diagrammatic representation of the topics, investigations and programme contributions divided into three phases. Within each phase are topics indicated as ovals and investigations indicated in rectangular boxes. The role of the author is indicated as annotations to each box and the connections between the topics and the investigations are shown through the series of connected arrows. All investigations were based on concepts developed and proposed by the author although in some projects, the author was assisted by others in the implementation and architecture phases. In the first discovery phase, the investigations concerned gaining a clearer understanding of active networks, cross layer context adaptation and the opportunities provided by these topics in the delivery of image-based adaptive mobile services. In the second phase, develop and demonstrate, the specific issues and opportunities from the discovery phase are combined with development activity in an effort to provide novel approaches and technology to the corporate research department. As outlined in the previous chapters, the development activity results in either prototype code and systems or new methods for specific product development activities. The final phase is to investigate the ability of the new

technology to withstand the potential future trends that may have significant impact on its performance.

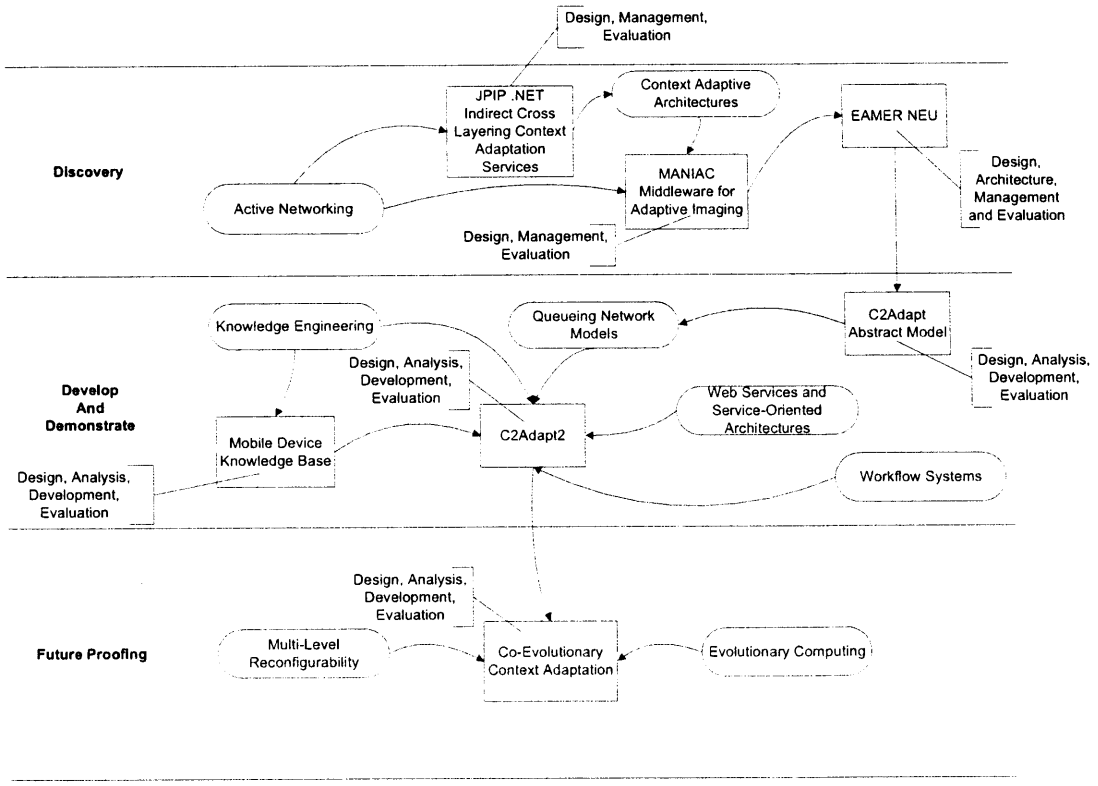

Figure 8-1: Programme Contribution Summary

From the **discovery phase**, the specific contributions are:

- A cross layered context adaptation system linked through a well-formed service level agreement demonstrating the viability of a Kodak image adaptive network solution
- A web services-based context adaptation system demonstrating multiple service levels and service types for use in mobile networks utilising a DiffServ architecture

Several interesting engineering issues arose when setting about to design and construct a commercially viable context adaptation solution. This discovery phase was largely motivated by the desire to build a unique Kodak service offering using state of the art networking technology. The first issue was basic; how to construct a simple adaptation service using the meta-information from the network and application layer in a

coordinated way. It was found that the use of a unifying element, that of the service level agreement, provided a solid foundation for a network adaptive service. This new unifying structure was not without its technical implications, however, as it became clear that while it provided a straightforward way to combine adaptation directives, it also meant that the service delivery was now QoS regulated and a new approach to meeting that regulation was required. The subsequent work then focussed on expanding the prototype concept to real-time, server directed adaptation in the NEU platform. Through the use of the NEU context adaptation architecture, issues of performance and scalability were observed. From the NEU service, it was demonstrated that cross layering, or sharing relevant information across the system protocol stack, provides new mechanisms for active adaptation. The penalty for this adaptation are issues with performance and adherence to service guarantees as was documented in prototyping of context sensitive image album applications in programmable network environments.

From the **develop and demonstrate** phase, the specific contributions are:

- An abstract model of context adaptation which provides an operational schematic for evaluation of performance issues
- A evaluation of performance using QNM and scalability analysis
- A design, architecture and prototype of a new solution which utilises knowledge-base systems and web services to improve the overall performance and accuracy of the context adaptation services

In this next phase, experimentation into these new approaches showed the clear need to incorporate the management of time-sensitive decision making into the underlying architecture of the services themselves. In building context adaptation services for mobile computing, it was found that quality of service depends significantly on fast access to accurate knowledge of device capabilities. Collecting these details in real-time or at a time nearest to the user's need of the service, provides the best relevancy of information on which to base evaluation, but at the direct expense of performance. From this conclusion, it was important to investigate further the important aspects and architecture required for the collection of context and how that contextual information

could be utilised efficiently in delivering the mobile user the specified and agreed service. Isolating, analysing and focusing on the significant components of the overall adaptation service led to the formulation on an abstract model. This model was evaluated using empirical and analytical data from the real world application.

Through the QNM modelling exercise, it is became clear that two areas of the system operation were of particular interest, namely dynamic collection of context and the extent of the evaluation necessary to deliver the adaptation properly. This method of analysis provided a high level guidance on the nature of the system's operation and helped to focus the engineering study on the most important aspects. Careful attention was paid to understanding the issues of simplification that can occur with a model-based approach. Using the conceptual engineering approach mentioned in Chapter 3, the complexity of adaptation was decomposed into three constituent parts; collection, evaluation and adaptation. The modelling work which utilised queuing network theory demonstrated that for practical purposes it was the collection process that had the greatest impact on the overall ability to deploy a context adaptation service that would definitively met a service level objective, and within this the degree of dynamic collection of context was the single greatest contributor to whether or not the service was able to meet a time parameter of a service agreement. Equally these elements also contributed directly to the ability of such a system to scale in terms of the number of computing resources needed to service requests. The process of function decomposition as an analysis step in the network of queues modelling effort revealed fundamental issues at each stage. This is an important engineering observation, that forcing the overall adaptation service into a network of queues model creates a set of component services which are very straightforward to implement as a set of individual web services. The analytic modelling resulted in further insight into the performance and scalability issues, but it also provided an excellent foundation for the design of the new, enhanced system, C2Adapt2. Analysis of the model provided one architectural solution for efficient context adaptation, one in which the services are components of a highly segregated functional nature. Through analysis of alternative architectural models and based on corporate strategic technology directions, the concept of using a persistent

repository for service requirements and static device capability was a key component within the programme.

Creating a persistent repository for storage and caching of context provided a robust and convenient way to improve performance by reducing the amount that is collected dynamically and it served to increase the potential for accuracy of the data as well. However, a simple relational database was found to not provide the flexibility that is necessary to effectively capture the evolving capabilities of the mobile device market. To react properly to missing or incomplete information as well as the ability to learn and add new features to both device capability and service requirements requires a repository with knowledge and a reasoning facility. From these results, a new model of context adaptation was formulated from a deeper understanding of these two somewhat complex operations, a model that uses a knowledge-base and a lightweight reasoning engine at the centre of a set of QoS monitored web services. The approach of embedding an evolvable knowledge-base repository into the design of the future context services is one that relies on a centralised style of architecture, not one that is distributed. This architecture improves the accuracy of the context information for those able to quickly access it, and it may also then improve the quality of the delivered service. However, this approach is counter to the current approach of truly distributed systems where information is distributed closest to the point of use. This could mean that individual user performance would be reduced as the services offered became more widely distributed and would then require another higher layer of context repository networks to provide the information on context in a timely manner. While populating such a knowledge-base structure with static context is straightforward and perhaps useful, the degree of dynamically acquired context that is necessary will act as an upper limit to the overall performance of the improved architecture. However, the conclusion of this programme is that an open, accessible, and ontologically sound knowledge-base of mobile device contextual information as specified by the manufacturer provides an enabling platform for all service providers. The concept of providing such an open knowledge-base for enhanced service information would require effort and investment by the key stakeholders in the process, and specifically device manufacturers who would

have to supply details of their products at a much more specific level. But the resultant improvement in service rate, in accuracy of adaptation and thus, in the overall quality of service is clearly an advantage for the industry. Methods of populating the repository range from the fully automated information extraction using manufacturer provided specifications to EDI-like XML feed provided when new mobile device products are launched. The motivating factor for manufacturers of mobile computing devices to take part in this effort is that it serves as an enabling source of accurate information for the development of new novel services.

Through the process of isolating the systematic issues with engineering context adaptation in a QoS environment, and then designing an alternative approach and implementing this approach as a set of orchestrated, it is clear that context adaptation inherently requires a form of automated reasoning. Even it is simplest form of image rendering to a specific format for a type of display, as in the case of an imaging service, reasoning about whether this can be achieved in the time allocated by the service level agreement must be done. In the world of best effort services, the use of a reasoning facility is an option. However, in the world of regulated or constrained service levels with penalties, reasoning and decision-making becomes a requirement. From this result is that service engineering in a QoS environment will require some form of computational intelligence embedded in the service workflow to make valid decisions on behalf of the user. The primary contribution of this work is an architecture and set of prototypes demonstrating such a knowledge-based web services solution for context adaptation. Using business process and business process execution language, this architecture is straightforward to implement and reflects the goals of the overall system quite well. The use of a two parallel process design means that the context adaptation service runs monitored by a second web service with greater authority on behalf of the provider. The critical design requirement is that each context adaptation stage is "atomic" in that it is capable of executing a complete component function without reliance on the other stages of the overall process so that in this way, the monitoring task can break the process at a decision point given the guidance of the user or a pre-determined profile. This is an important conclusion as there are issues with how to

inject decision points into such real-time, online systems so that the user can be queried as to their desire, e.g. are they willing to forego the service level guarantee to receive a higher quality result? Viewing context adaptation as a set of individual, yet orchestrated services with a separate trusted QoS service providing the business "conscience" of the transaction is a commercially feasible approach.

A conclusion developed in the develop and demonstrate phase of the programme contributes is in the area of systems engineering tools, techniques and methods for contextually-based service engineering. Through the experience found in the different technology focus areas, it was clear that a single toolkit for development of a context adaptation service or solution was simply not available. New tools had to be acquired and learned at each phase. In traditional systems engineering projects, the transition from architectural design to tools and development methods is an important step and it is often one that slows a project's progress significantly due to steep learning curves for the project and/or the maturity of the tools themselves. This was particularly true in this project. There exists a vast array of disconnected software tools and development toolkits in various states of production readiness. This condition is indicative of the new direction the industry is taking towards a service-oriented architecture and business process orientation. One area which is lacking currently for tool support in that of validation and verification of the web services, particularly those that are intended to be composed dynamically. While the tools for business process management (BPM) has some relevance in this domain and a longer history, emerging service-oriented technology promises to go well beyond business processes particularly in the area of reconfigurable and embedded systems. The BPM tools, techniques and best practices are complementary but in many cases, they conflict or at the least, utilise a substantially different vocabulary and approach to than that found in the more technical SOA community. There is, however, some promising activity to converge the business process technology approach with that of the service-oriented architecture which will benefit service designers, developers and providers by allowing an integration of the language that describes the business with that of the technology. Another interesting contribution from an engineering perspective follows from this lack of tool support for

the verification of composable web services. There appears to be no common method to analyse the design and specification of these services. This is quite a large and important issue for service-oriented system engineers who need to be able to ensure that the system will interoperate in a consistent and robust manner. The use of finite state process analysis as a common point of verification for the design and development provides one consistent mechanism to compare dissimilar technological domain such as business process and UML message sequence. This approach, much like finding a "least common denominator" in mathematics, means that both the business process and the technical design should resolve to a fundamental set of possible states. This common representation also provides a formal foundation for these systems which will be necessary in order to move towards service-oriented computing as a part of mission critical systems. Through the comparative analysis between a BPEL design tool focussed on process and the UML message sequence chart focussed on messages, the software service engineer can gain very valuable information about the performance and feasibility of the system prior to its complete code implementation. Model checking the workflow states is highly recommended as the complexity of the interaction between some number of automated services and the impact this has on the consistency of the overall system will be critical to building reliable context services. However, this expectation that the engineer might arrive at the same model starting from two separate and quite different points proved unrealistic and the time necessary to resolve the potentially disparate models would be difficult to justify in a commercial development project. It is clear from this work that not only does a service engineering project need a process designer which provides for a standardised design, but it will also require the use of either a visualisation or simulation tool to depict the interconnected behaviours found in more complex systems. It is important to note that due to the resources available in this programme the tool evaluation and analysis was not exhaustive and given the dynamic nature of the industry the technologies available are almost certain to change almost frequently.

From the **future proofing phase**, the specific contributions are:

- The identification of future issues of context adaptation services that may conflict when operating in reconfigurable hardware architectures
- A novel method for determining the possible reconfiguration baseline state that a device could be set to in order to minimize the service specific changes required
- A novel design and experiment into the co-evolutionary matching of the device capability with the service requirement using the minimisation of string distances

In this phase, the investigation focused on looking deeper into the possible trends that might affect context adaptation service engineering. This process revealed some interesting challenges and opportunities. The nature of context adaptation will become far more complex as the analysis into the impact of reconfigurable architecture has shown. As the hardware industry continues to move forward with fully reconfigurable devices, the software and services engineering industry will need to be able to address the challenge that such ultimate adaptability will place on their current and new systems. In a reconfigurable device environment, the fixed or dependable device state attributes will reduce over time as more functionality is deployed into the reconfigurable tiles of the processors. This requires that service engineering for new context adaptive services will need to implement runtime configuration management logic and embed control into the services workflow that will expose configuration details to the necessary layers of the service itself. This will ensure that the adaptation can occur against an accurate set of destination device attributes. There are several possible ways to build such a system, one conclusion is that the historical record of the device's runtime configuration over time will be extremely useful as a means to formulate an optimal baseline configuration. From this, it is possible that a secure repository of such information will enable the optimised context adaptation services of the future by guaranteeing a statistically minimal service to device mapping gap, given a statistically relevant amount of device configuration history. With services intended for mobile devices, this historical information is most likely not stored on the device itself but rather on a node with more computational resource. A research question derived from the

future potential of context adaptation in reconfigurable device environments is who and how will the baseline configuration of the devices be maintained at the lower architectural levels? Runtime configuration changes need to be carefully controlled particularly as they will have fundamental impact on the device operation; security, consistency and reliability are critical factors in the success of such future architectures. New service level agreements and policies will be needed that specify the levels to which reconfiguration is permitted and which actor is authorised to alter functionality at which layer of the systems architecture in preparation for the execution of the service. It is proposed that it will be necessary to have a guaranteed end state of the overall device reconfiguration which must be agreed and known if third party adaptation services are to be viable. In today's service engineering world, hardware specifications provide a certain level of surety as to the state a service can expect to encounter. This assumption will erode as more functionality becomes software reconfigurable. Reasoning about the adaptive potential of a device given a specific baseline or expected configuration is a possible approach for the future. Reconfiguration will also necessitate new ways of designing context adaptation as well as new techniques for implementing it into production environments. It is clear that reconfigurable processors impose both spatial and temporal requirements on system design and current design methods and tools do not necessarily take these elements into account. Development processes at the higher abstraction level will require model checking and simulation of device state to reflect the spatial and temporal changes that are a result of the service execution so that the changing state of the device can be validated for safety and reliability. Coupling device reconfiguration with dynamic service composition means that the number of possible unknowns in the system may exceed a safe level and formal constraints should be in place to ensure any mobile device system can continue operation after the execution of the complex workflows. The notion of surface analysis is contributed in this work as a design tool, but there are other methods in use in the hardware co-design technology area.

In the last phase of the programme, the investigation into the use of co-evolutionary computation for optimising the evaluation phase of the context adaptation system. The

technique contributed here is specifically applicable to the problem of optimisation where one is interested in the best possible set of non-linear solutions given a time dependent set of constraints. Through the use of cooperative co-evolutionary techniques with a string-based genome, a prototype of the capability and requirement matching algorithm for context evaluation is demonstrated and it is concluded that in this case, the addition of the time constraints means that an upper limit on the number of evolving generations must be set. Although a basic functionality is demonstrated for optimal capability to requirements matching, a significant problem still remains to be solved and will be included as a part of the subsequent section on future work. The transformation and translation of the basic configuration of device as a phenotype should be converted programmatically into a genotypic representation. An automated process, as a pre-processing step, could be done offline, triggered on each modification of the frame instance in the MDKB. A separate translation service is needed to update the genotype, or strings as shown in the experiment, based on the new specification of features. This historical record of the genotype transformations can then be used to provide an optimal baseline configuration. In the experiment, a manual transformation of phenotype to genotype was done in a process that involved creating the device capabilities and service requirements ontology and translating this into the knowledge-base itself. Supporting contributions for this method were found in defining the relationships between the concepts and determining the constraints necessary to compose a large set of physical device representations results in the genotype. It was found that using the FIPA ontology as the backbone for such a device and service genotype provided an efficient and effective method. This concept of using a standards-based technical specification as the "backbone" ontology for an application domain is known practice in KB development, but not formally specified in the service engineering. It is also proposed here that this shared ontological view is essential when developing web service components that will ultimately interact or come together dynamically to achieve a specific goal.

In summary, through the various prototypes and demonstrators in this programme, contributions are provided in a broad range of technologies involved in con... ...g

context adaptation services. Each subsequent investigation within the programme targets a significant problem found in the previous investigation and the results are then brought together in the C2Adapt2 prototype, incrementally improving the system so that it can then be used as a reference model for context adaptation service engineering. In the future proofing phase, the C2Adapt2 system was then further analysed and modifications were proposed that speculatively seek to maintain its viability even in future mobile device computing environments.

8.2 Future Work

There are several threads of beneficial investigative and engineering work that would advance the elements of this work further. The primary goal of future work from an industrial research perspective would be to further integrate and automate the connections between the prototypes and systems. As mentioned above, the first and the most challenging research effort is in the area of "semantic conversion" or automating the phenotype to genotype conversion as an alternative to the manual process. Converting from a published set of services to their underlying genotype could possibly miss the unexpressed potential in the service as is found in the biological analogy. Investigating the feature loss by converting to the genotype and then experimenting re-expressing the genotype, perhaps even into a virtual 3D model, to the phenotype of the device would provide a useful validation of the overall process. Study and experimentation in this area will expose fundamental processes in capability to requirement mapping through modelling and provide new insight into the process itself. This further investigation could use new and emerging techniques found in the other biologically-based industries as a starting point to creating a computational workflow for semantic matching of capability to service requirements. With an automated conversion process for both device and service, the research work could then turn towards tuning the matching evolutionary process directly to the desired service level. As mentioned previously, the benefit of using the co-evolutionary approach is that one can evolve an optimal solution or alternatively stop the evolutionary process at some sub-optimal generation and still be assured a level fitness between the two processes. This partial matching methodology is difficult and time-consuming process when using

hierarchical structured information paths such XML DTD matching, but is very straightforward when using the co-evolutionary technique. The question is how to regulate the matching algorithm based on the quality guarantees that are in force for the requested service. A future project in this area would involve the integration of the task monitoring and the service level with the co-evolutionary algorithm for the matching operation. The expectation of this project would be to provide a unified and automated solution with the necessary interfaces as depicted in Figure 7-7. A further beneficial area of future engineering study and experimentation is that of the proper architecture for moving MKDB to a distributed knowledge-base architecture. It was recognised and presented that the architecture as described fails to implement a truly distributed systems model. In subsequent work, it would be important to rectify this situation. The research question is how best to create a distributed knowledge-base of evolving device capability and service requirements accessible to the service provider in order to provide adaptation to their mobile consumer.

Further work on the issues of systems engineering for reconfigurable device systems in service-oriented architectures, specifically the notion of adding spatial and temporal simulation or visualisation into the design phase of dynamic service composition, will be important. There is a significant new paradigm to explore in this area that goes beyond hardware and software co-design as it is today. Integrating the trend in service-oriented architectures with reconfiguration provides a wealth of research questions across the layers of the system architecture stack. Investigating the nature of an integrated design process for this new environment, including topics such as how to handle requirements across the system, how to deal with temporal constraints which occur at different levels of granularity, and how best to approach the architectural design either from bottom-up or top-down are all important questions that will need to be answered to support this future direction. One specific area of planned work is to explore the notion of an optimal baseline specifically evolved for a generic reconfigurable device based on evaluation of its own historical usage, capability and service request genotypes. The ability to evolve a set of configuration states specified by cross-layer parameters for a reconfigurable device and then to test those on an actual device is the next step in this

engineering programme. Understanding the issues of power, consistency and a key concern around multi-service configurations are all topics that would will be addressed. Specifically, the core issue with the complexity of running many services each with their own specific configuration requirements is a challenging and fascinating avenue of research to pursue. Given that context adaptation is not just one service, but a way in which to provide high value online software services, it is quite possible that many of the requested services would each require a varying degree of context adaptation and the management of this scenario will require both research and engineering validation.

In addition to the topics previously mentioned, there is considerable advanced development and commercialisation work still to be done in the integration of the technology and concepts developed thus far to meet a standard of robustness. As presented, the systems architecture is still at its earliest phase, although individual components are developed and tested to the point of confirming functionality, the integration work that creates an engineered solution remains to be completed and is an important future work.