UNIVERSITY OF LONDON THESIS

Degree PhD Year 2007 Name of Author SMITH, Richard Bartlett

Design and Integrity of Deterministic System Architectures

Richard Bartlett Smith

Centre for Systems Engineering

Mullard Space Science Laboratory

University College London

This thesis is submitted in accordance with the Regulations for the degree of Doctor of Philosophy in the University of London

October 2007

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

Declaration

This thesis has been written and compiled by the author and is submitted in accordance with the Regulations for the degree of Doctor of Philosophy in the University of London.

Richard B Smith: Date: 25th Oct 2007

ABSTRACT

Architectures represented by system construction 'building block' components and interrelationships provide the structural form. This thesis addresses processes, procedures and methods that support system design synthesis and specifically the determination of the integrity of candidate architectural structures. Particular emphasis is given to the structural representation of system architectures, their consistency and functional quantification. It is a design imperative that a hierarchically decomposed structure maintains compatibility and consistency between the functional and realisation solutions.

Complex systems are normally simplified by the use of hierarchical decomposition so that lower level components are precisely defined and simpler than higher-level components. To enable such systems to be reconstructed from their components, the hierarchical construction must provide vertical intra-relationship consistency, horizontal interrelationship consistency, and inter-component functional consistency.

Firstly, a modified process design model is proposed that incorporates the generic structural representation of system architectures.

Secondly, a system architecture design knowledge domain is proposed that enables viewpoint evaluations to be aggregated into a coherent set of domains that are both necessary and sufficient to determine the integrity of system architectures.

Thirdly, four methods of structural analysis are proposed to assure the integrity of the architecture. The first enables the structural compatibility between the 'building blocks' that provide the emergent functional properties and implementation solution properties to be determined. The second enables the compatibility of the functional causality structure and the implementation causality structure to be determined. The third method provides a graphical representation of architectural structures. The fourth method uses the graphical form of structural representation to provide a technique that enables quantitative estimation of performance estimates of emergent properties for large scale or complex architectural structures.

These methods have been combined into a procedure of formal design. This is a design process that, if rigorously executed, meets the requirements for reconstructability.

Keywords. *Systems Engineering; System Design; Design synthesis; Architectural structures: Architectural integrity.*

Dedication

'To those who wish to work smarter, not harder.'

Acknowledgements

To the supervisors Professors Kenneth Hambleton, David Kirkpatrick and Alan Smith, and Dr. Michael Emes at University College London, whose support and advice has been essential to enable me to carry out this programme.

To Professor Ludwig Finklestein at City University for providing the introduction to the analysis of measurement system designs. To Professor Robin Whitty at South Bank University for providing the introduction to graph theory. With respect to the use of Mathematica, to Professor P. McIver, School of Mathematical Science at Loughborough University for advice on the program construction of rule based mathematical procedures, and to general advice provided by the support helpline at Wolfram Research Inc.

To the Directors and colleagues of the Avionic Systems Division, Platform Solutions, BAE SYSTEMS plc for their sponsorship and support during the preparation of this thesis. Of special note is the support from Gordon Belcher for his encouragement to write this thesis, to Fred Mackley and Bob Wilkinson for making it financially possible, to Stephen Taylor for his clarity of review and coaching in abstract algebra, mathematics and modelling, and to Martin Field for his encouragement and guidance to enable it to be completed. Also to Debbie Kemp, Christine Hobbs and Stuart Earl for assistance with typographical presentation.

To my wife, Maggie, whose constant support and encouragement has enabled me to complete this programme.

The author.

Contact email address is seymr.smith@btinternet.com.

Table of Contents

List of Appendices

List of Figures

List of Tables

List of Programs in Main Body of the Thesis

Mathematica Notebooks

Name	Description
ThesisM1	Determination of Number of Paths between each Node.
ThesisM2	Matrices showing number of paths between each node
ThesisM3	Computation of node-to-node functionality with two, three and four step functionality
ThesisM4	Computation of path-to-path functionality with two, three and four step functionality
ThesisM5	Graph with arbitrary edge capacity values and with highlight of capacity capability between A and C
ThesisM6	Example of analysis of series function path structure with insertion of 'across' variable functions
ThesisM7	Example showing Functionality of Interconnectivity Path Structures – with feed forward Interconnections only combined with insertion of attenuation functions.
ThesisM8a	Example showing Functionality of Path Structure combined with insertion of attenuation functions.
ThesisM8b	Example showing Functionality of Path Structure with insertion of combined series/parallel impedance network
ThesisM9	Example of Decomposition of a Direct Product expression with insertion of a polynomial function
ThesisM10	Example of Decomposition of a Direct Sum expression with insertion of a matrix function
ThesisM11	Example of Decomposition of Direct Product and Sum combination expression with insertion of a matrix function

List of Programs in Appendix 1 of the Thesis

Excel spreadsheets

Name	Description
Helo N2 rsm	Top down Management structure interrelationships
Helo N2 rs	Overall system structure interrelationships
Helo N2 r1	Composition domain interrelationships
Helo N2 r2w	Flow Capacity domain interrelationships
Helo N2 r2e	Load capacity domain interrelationships
Helo N2 r3	Interconnectivity domain interrelationships
Helo N2 r4a	Functional domain interrelationships

Mathematica Notebooks

Name	Description
Example1	Demonstration of functional decomposition of path structure
Example2	Demonstration of flow capacity analysis
Helo Mrsm	Top down Management structure
Helo Mrs	Overall system structure
Helo Mr1a	Composition domain analysis
Helo Mr2w	Flow Capacity domain analysis
Helo Mr2e	Electrical distribution capacity domain analysis
Helo Mr3	Interconnectivity domain analysis
Helo Mr4a	Functional domain analysis

Executive Summary

The discipline of Systems Engineering is concerned with entities that involve interaction and cooperation between constituent components. The discipline gained initial recognition in the World War 2 era, and since then it has attracted considerable support primarily as the means of tackling large-scale problems.

The 'V' Model is the preferred structure of that part of the Systems Engineering Process model that addresses the design and integration phases of an overall system lifecycle. One form of its representation. see Buede, 2000, [65], is shown schematically in Figure 1. The left hand side is concerned with the creation of the design solution and its capture in the form of engineering drawings; the right hand side is concerned with the realisation of the solution from component manufacture, through integration and to system validation.

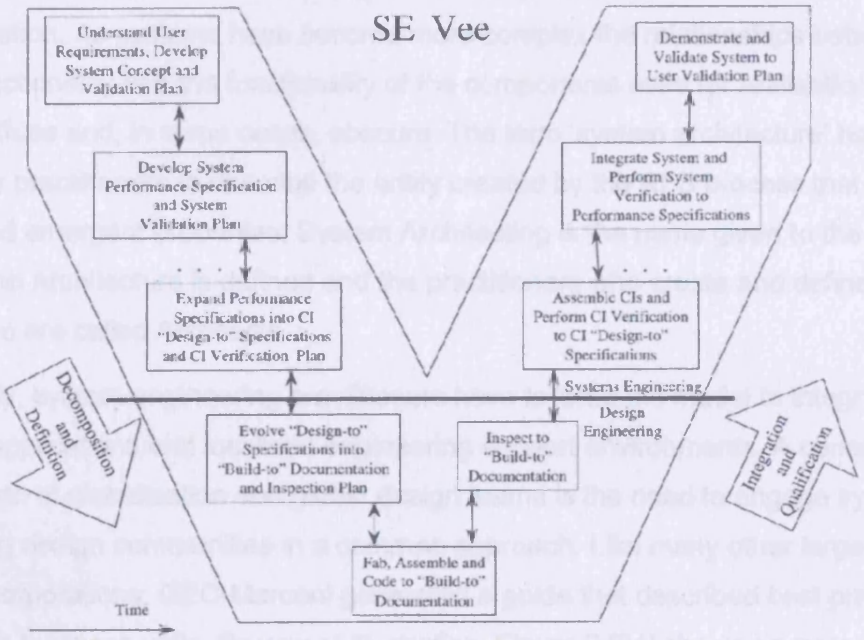


Figure 1 - System Engineering V Diagram Process Model (P1)

Since the mid 90s there has been a proliferation of models that use the 'V' Model as the generic construction to generate variations that suit particular communities or applications. This can readily be seen by a casual review of the articles published in the journal of 'Systems Engineering' published jointly by INCOSE and Wiley. Further, tremendous strides have been made by the CALS/STEP/AP 233 programmes to provide a neutral data exchange design environment for engineering drawings and multi-media data products. Currently a framework that enables a wide range of specialist design tools to cooperate via neutral data exchange is at an advanced stage of development.

The core of the basic systems engineering process model is defined by the tripartite relationships between a system's Requirements, its Functional definition and allocation, and the Synthesis of its implementation that provides the stated needs and functionality; this has been termed the 'RFS Model' by the author. The model was originally generated for MIL-STD 499 and has been perpetuated within the ISO 15288 specification. The model works well when there is an intimate relationship between the functions that form the required system and the functionality of the components used for their implementation. As systems have become more complex the relationships between the system functionality and the functionality of the components used for realisation has become diffuse and, in some cases, obscure. The term 'system architecture' has been adopted by practitioners to describe the entity created by the RFS process that provides the required emergent properties; System Architecting is the name given to the process by which the Architecture is defined and the practitioners who create and define the Architecture are called Architects.

Traditionally, system engineering practitioners have tailored the model to integrate with particular applications and localised engineering support environments. A consequence of the growth of globalisation and global design teams is the need to engage system engineering design communities in a common approach. Like many other large-scale industrial corporations, GEC-Marconi generated a guide that described best practice for use in all its business units. By way of illustration, Figure 2 [94] shows an instantiation of the process model. As the size of problems tackled by systems engineering design teams had grown the authors realised that the process by which the 'Functional Concept' and the 'Implementation Concept' become an 'Architecture' was substantially undefined. The term Design Synthesis has been incorporated into the diagram to represent the activity that determines the architecture. It is called an activity so that its means of delivery can include processes, procedures and methods.

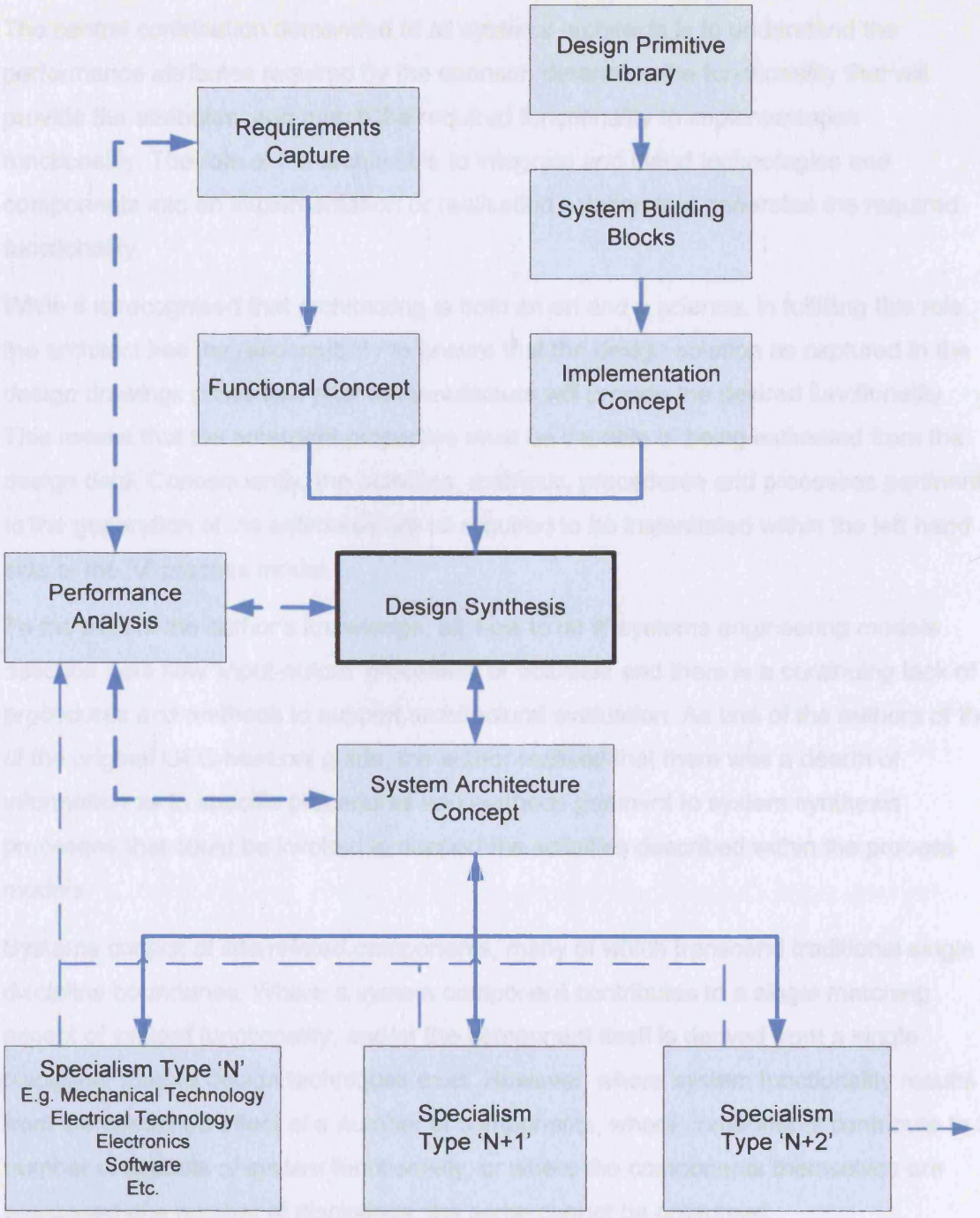


Figure 2 - Schematic of the MIL-STD 499 Architecting Process Model depicting the role of Design Synthesis (D22)

The central contribution demanded of all systems architects is to understand the performance attributes required by the sponsor, determine the functionality that will provide the attributes, and match the required functionality to implementation functionality. The role of the architect is to integrate and blend technologies and components into an implementation or realisation solution that generates the required functionality.

While it is recognised that architecting is both an art and a science, in fulfilling this role the architect has the responsibility to ensure that the design solution as captured in the design drawings presented prior to manufacture will provide the desired functionality. This means that the emergent properties must be capable of being estimated from the design data. Consequently, the activities, methods, procedures and processes pertinent to the generation of the estimates are all required to be instantiated within the left hand side of the 'V' process model.

To the best of the author's knowledge, all 'how to do it' systems engineering models describe data flow 'input-output' processes or activities and there is a continuing lack of procedures and methods to support architectural evaluation. As one of the authors of the of the original GEC-Marconi guide, the author realised that there was a dearth of information as to specific procedures and methods pertinent to system synthesis processes that could be invoked to support the activities described within the process models.

Systems consist of interrelated components, many of which transcend traditional single discipline boundaries. Where a system component contributes to a single matching aspect of system functionality, and/or the component itself is derived from a single discipline, mature design techniques exist. However, where system functionality results from the combined effect of a number of components, where components contribute to a number of aspects of system functionality, or where the components themselves are composed of a number of disciplines, the same cannot be presumed.

System integrity is concerned with the unity and wholeness of the design in relation to its intended purpose. Clearly, the integrity of the architecture is of prime concern to all stakeholders and, for all non-trivial systems, it cannot be taken for granted. Its determination is complicated because of the interconnectivity between components and the multi-functional nature of these interrelationships. Therefore, a means of formal evaluation is required that encompasses the multi-functional and multi-disciplinary nature of modern designs.

Despite the maturity of systems design processes, the author asserts that the methods associated with architectural analysis that are typically available to the system architect have not been developed to support the complexity of modern systems. The consequence is that the emergent properties of non-trivial systems are, in general, only partially/selectively predicted during the design process. Therefore, many significant emergent properties, in terms of acceptance compliance, only become apparent during the later stages of integration and commissioning. Such properties put unwanted risk into the acceptance process, have the potential to result in programme schedule extensions and substantial additional costs if corrective action is required, and cause user dissatisfaction.

The motivation for this thesis is to establish and describe some methods that mitigate the likelihood of such undesirable properties not being identified during the design process by:

- Asserting that structural integrity evaluation should be instantiated within the system design process model, and
- Generating analytical methods that provide quantitative estimates of the system's emergent properties during the design process and, by implication, identification of undesirable emergent properties.

Within this thesis the author justifies the assertion that the processes, procedures and methods described herein, enhance the knowledge pertinent to the structural integrity of system architectures.

The key concepts are introduced in five stages: -

- Machine system design process – an overall process is proposed that is centred around use of structural integrity evaluation.
- System evaluation coherence – a necessary and sufficient set of viewpoint domains of the structural integrity of an architecture is proposed to provide confidence in the completeness and coherence of the architecture.
- Traceability methods – methods of analysis are proposed that allow assessment of the impact on architectural structural integrity arising from realisation constraints.
- Architectural structure representation – a means of representation of architectural structures is proposed to further support functional analysis.
- Functionality expression - a method of functional population is proposed enabling node-to-node functional analysis that is inclusive of functionality arising from the disparate disciplines and technology bases that modern systems typically employ.

These techniques enhance the ability of the system architect to estimate emergent properties throughout the design process. Examples are used to illustrate the viability of the approach.

Firstly, a description of a Machine System Design Process is proposed. It extends the widely accepted Requirements, Functional Decomposition and Solution Synthesis model of the engineering design process to ensure that full structural analysis of the architecture is instantiated within the design process. This assures that the structural composition of a system and its evaluation, held in tacit knowledge form, is formalised in the design definition.

Secondly, the coherency of system viewpoint evaluations is addressed. Viewpoint Analysis is a technique in common use that enables system architects to evaluate a system design from a specific perspective. It is a means of simplification by taking a single subject focus for evaluation; that facilitates quantification by identifying consistent component function models and metrication. Usually, viewpoints are selected for analysis according to perceived need, or interest or subject knowledge, and the number of viewpoints analysed for any particular application will vary according to, for example, perceived risks as well as budgetary constraints. The result is that while a qualitative appreciation of the totality and coherence of the number of viewpoints for a particular application may exist, the adequacy of the scope and consistency of quantitative coverage is rarely considered.

To provide a coherent basis for viewpoint population, it is proposed that the integrity of a system design can be determined by thorough consideration of the architecture in four

domains of interest, viz. composition, capacity, messenger and functional behaviour, each of which has special significance to system architects. This is shown in Figure 3.

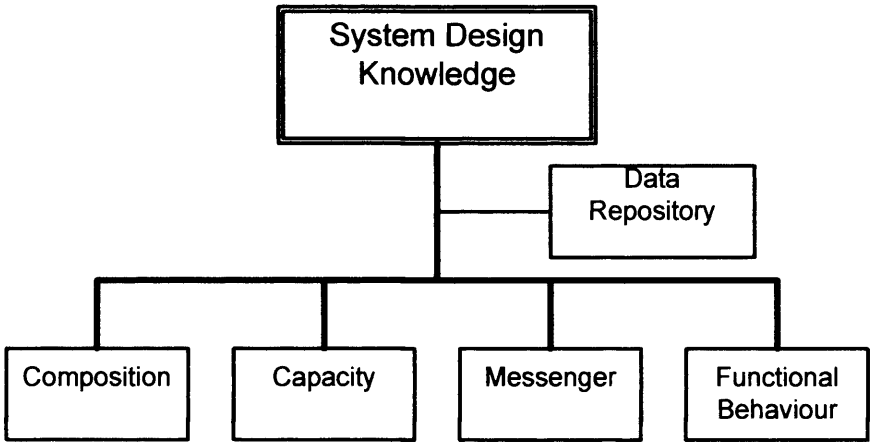


Figure 3 - Architecture Design Domain Decomposition (D1)

Note. The system structure and attributes of each building block are stored in the data repository.

The method determines that a system design is complete when its:

- structural arrangement is consistent: Composition.
- capability to meet the boundary condition performance requirements is ascertained: Capacity.
- interfaces have been constructed to provide the required connectivity and information transfer: Messenger.
- combined behavioural functionality between all nodes in all the architectural connections has been determined: Functional Behaviour.

Therefore, the hypothesis is that the scope of the proposed methods meets the sufficiency criteria for system architectural constructions.

While this decomposition instantiates coherency into viewpoint selection and aggregation, the problem of complexity remains. This is addressed during the design phase by further decomposition. Throughout this process the architect is obliged to verify the design by matching the desired functionality of each building block to the functionality provided by its realisation solution. This is further complicated by the fact that the current scope of technology capability is such that the range of realisation options that occupy the viable design space for the system of interest is substantial.

Thirdly, therefore, to enable the traceability between functional building blocks and realisation components to be determined, two methods of analysis are described: hierarchical traceability and causal consistency. The first method uses Boolean matrices to evaluate hierarchical decomposition consistency, and the second uses a causality network to match functional dependency and realisation component compatibility. Both methods identify inconsistencies between the functional and realisation architectures.

Fourthly, in order to provide the architect with a generic means of representing architectural structures, it is proposed that these be created and held in the form of graphs. Directed Graphs consist of nodes and edges, wherein each edge is defined as being unidirectional. It is proposed that architectural structures be modelled using Directed Graphs consisting of nodes and paths, wherein each node is a building block and each edge a labelled path.

Directed Graphs may be analysed using Linear Algebra. Matrix representation and analysis techniques enable the paths from any 'source' to 'sink' node to be determined.

The Directed Graph form of an arbitrary system architecture is shown in Figure 4.

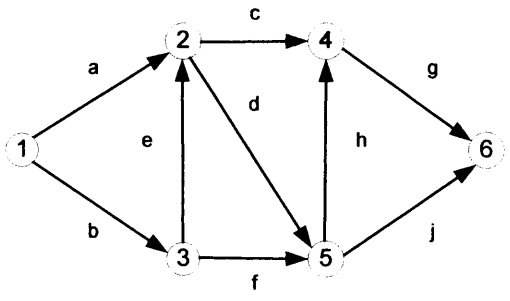


Figure 4 - A System Architecture in the form of a Directed Graph (D2)

The labelled path constructs are in the form of algebraic Direct Products/Sums; i.e. of the form a ∘ b or a + b, where a, b are functional components and '∘, + ' are types of binary association.

This enables a means of source-to-sink functional analysis to be carried out based on the decomposition of the labelled path structures into functional components and association relationships.

Fifthly, a method of functional population is proposed that enables the functionality of a source-to-sink path structure to be evaluated. To do this, models of each building block and path component need to be constructed. In addition, their rules of association need to be defined.

To demonstrate the capability of the method, various forms of model representation are used in the examples. Of particular interest is the use of network-based models as they provide a comprehensive and readily accessible technique of multi-disciplinary analysis. For example, Two-port models are based on linear algebra constructions that provide equivalence between electrical and mechanical component forms.

While the purpose of decomposition is simplification, the clear intention is that the collection of components so created can be integrated to form an entity that is indeed the desired system. Reconstruction is the complement of decomposition. Reconstructability is the term that describes the attributes of the system components relevant to their integration into a coherent entity.

The methods of traceability, and functional population combined together meet the requirements for reconstructability. Therefore, the thesis includes a proposal for a formal design method. This is shown as a process diagram in Figure 5 and the thesis includes a proposal for an algebra of system design.

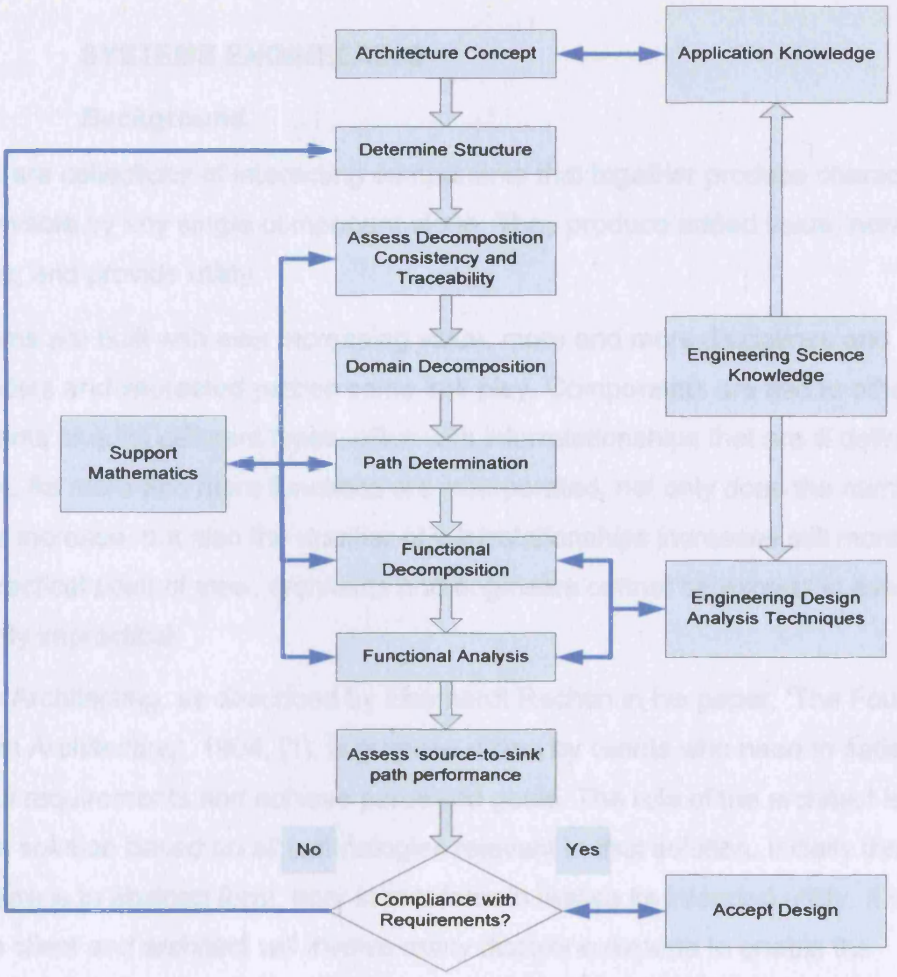


Figure 5 - Schematic of Architecture Structure Analysis (D3)

In summary, this thesis is concerned with procedures and methods required to support Design Synthesis. It proposes a structured method and analytical procedures pertinent to the justification of architectural concepts. Consequently, the systems architect is provided with both a means of decomposition that encompasses the requirements for completeness and the means of full functional analysis.

1 SYSTEMS ENGINEERING

1.1 Background

Systems are collections of interacting components that together produce characteristics not achievable by any single component alone. They produce added value, normally economic, and provide utility.

As systems are built with ever increasing value, more and more disciplines and stakeholders and interested parties come into play. Components are tied to other components of quite different types, often with interrelationships that are ill defined, and uncertain. As more and more functions are incorporated, not only does the number of elements increase, but also the number of interrelationships increases still more. From a purely practical point of view, architects and engineers cannot be experts in everything; it is humanly impractical.

Systems Architecting, as described by Eberhardt Rechtin in his paper, 'The Foundations of System Architecting', 1994, [1], is purpose driven by clients who need to satisfy stated functional requirements and achieve perceived goals. The role of the architect is to provide a solution based on all technologies relevant to that solution. Initially the architecture is in abstract form, later in real form to realise its intended utility. It is likely that both client and architect will involve many discipline experts to enable the requirements and means of mechanisation to be rationalised. The architect controls the architectural definition and facilitates the integration of the disciplines and stakeholders to provide a rounded and mature solution.

Architecting is an iterative process that results in the creation of an architecture that both client and provider jointly agree, in realistic terms, satisfies both the purpose and the solution mechanism; the 'what' is satisfied by the 'how'. An architecture is an object in abstract form. Its formulation provides the basis of the agreement between the client and the design authority as to the structure of the solution mechanism. Heuristics, familiarity of similar applications and knowledge of science and technology all play a significant role in the formulation of the architecture. Once the architecture in its abstract form is maturely acceptable to both client and architect, the solution and use of the solution are refined in detail, so that the implementation solution can be provided to the client for integration with operations and support, to obtain the utility benefits of the asset.

While heuristics play a very important role in the initial creation of the architecture, modelling is a capability used by both client and architect to quantitatively estimate the properties of the solution. Models are derived from the architecture and support its refinement.

These models provide estimates of quantitative performance and support the articulation of the various disciplines used to create the real solution; Computer Aided Systems and Software Engineering (CASE), Computer Aided Engineering (CAE) and Computer Aided Design and Manufacture (CAD/CAM) all fulfil a role to capture and effect the design, and estimate the attributes, behaviour and performance of the solution. Clients expect solutions to provide high economic availability, combined with minimum costs of support. Modelling provides the architect with the means of assessing the impact of variation both for the solution itself and for its operational context.

The process of certification and acceptance is designed to ensure that the real solution provides the required emergent properties. Clients are satisfied when their evaluation of the real solution confirms that their requirements are met. In practice the complexity of many systems means that the evaluation process identifies many emergent properties that were not anticipated by either the client or the architect. In such cases it is imperative that such attributes are evaluated in the context of the original architecture; if not, the value of such attributes is indeterminate, with the consequence that certification and acceptance are jeopardised. Further, post acceptance modifications that cannot be evaluated in the context of the original architectural concept, put the system re-certification at risk.

Robert Frosch, in the following extract from his famous address in 1964, [2a], just prior to his appointment as NASA Administrator, expressed the central concern about the need to match art and science.

> 'The fundamental difficulty is that we have all become so entranced with technique that we think entirely in terms of procedures, systems, milestone charts, PERT diagrams, reliability systems, configuration management, maintainability groups and their minor paper tools of the "system engineer" and manager. We have forgotten that someone must be in control and must exercise personal management, knowledge and understanding to create a system. As a result, we have developments that follow all the rules, but fail.
>
> As we are now behaving, we are using up our best people filling out documentation for their superiors to read, and most of the time nobody is running the store.
>
> We have lost sight of the fact that engineering is an art, not a technique; a technique is a tool. From time to time, I am briefed on the results of a systems analysis or systems engineering job in a way that prompts me to ask the questions: "That's fine but is it a good system? Do you like it? Is it harmonious? Is it an elegant solution to the real problem? For an answer I usually get a blank stare and a facial suggestion that I have just said something really obscene.'

Questions of the form 'Is the solution good?', 'Is the solution elegant?', 'Is the solution harmonious?' are subjective. Consequently, the architect has great difficulty in formulating relevant responses.

A more objective approach has been provided by Churchman in 'The Design of Inquiring Systems' ,1971, [3] by addressing the concepts of 'wholeness, soundness and virtue'. More explicitly he provides a set of nine necessary conditions, stated as follows, that enable something that is called a system, S, to be conceived as an entity.

1. S is teleological.
2. S has a measure of performance.
3. There exists a client whose interests (values) are served by S in such a manner that the higher the measure of performance, the better the interests are served, and more generally, the client is the standard of the measure of performance.
4. S has teleological components that produce the measure of performance of S.
5. S has an environment (defined either teleologically or ateleologically), which also co-produces the measure of performance of S.
6. There exists a decision maker who – via his resources – can produce changes in the measures of performance of S's components and hence changes in the measure of performance of S.
7. There exists a designer, who conceptualises the nature of S in such a manner that the designer's concepts potentially produce actions in the decision maker, and hence changes in the measures of S's components, and hence changes in the measure of performance of S.
8. The designer's intention is to change S so as to maximise S's value to the client.
9. S is stable with respect to the designer, in the sense that there is a built-in guarantee that the designer's intentions are realisable.

The author interprets the use of the word 'teleological' to mean that S is complex, purposeful and created by a designer. These conditions imply that the client and designer join together to reconcile an acceptable measure of performance with a realisable concept solution; the role of the decision maker is to provide the resources to enable S to fulfil its intended purpose.

Since these words were written the language has changed but the fundamental message is as clear today as it was then.

Now systems are considered to be complex and by implication more difficult to design and performance models more difficult to construct. Nevertheless, we have the benefit that most, if not all, disciplines have developed their capabilities. Further, performance models do not need to be confined to the quantitative domain as the use of language can

be no less disciplined than numeric information. For example, in recent years process capability has developed a language of attributes to characterise maturity. These attributes can be compared with an internationally agreed set of best practices scaled as Capability Levels. The technique was developed by Carnegie Mellon University Software Institute and is known as CMM/CMMI, [4].

The term System Architect has replaced that of Designer and measures of performance are as much about specific capabilities expressed with specialist vocabulary as well as measures confined to numeric expression. Churchman placed the responsibility on the shoulders of the client for utility values, the decision maker for resources and the designer for the solution. The duty of care to debate explicit performance in relation to concept solutions and resources has not changed.

To explain the design problem, the term architecture may be likened to a building wherein the space is delineated for use as a home that provides security, physical comfort, personal sustainability, a place of social engagement and a place for personal fulfilment. The space is provided by the layout and its structure. The structure depends on a wide range of engineering and science disciplines both to create the space and ensure that it is safe to use throughout its lifetime, including its tolerance to environmental and other long-term risk factors. Behaviour may be likened to how people interface with the space, layout and services, so that it becomes a home in which each area may be associated with some aspect of domestic activity e.g. working, eating, sleeping, playing, and performance may be associated with, for example, the roles fulfilled by specific functions (e.g. living rooms together with sustainability rooms, kitchen, bathroom, utility etc, and environmental management systems). The quality of living may be likened to process attributes, for example, for interpersonal behaviour or self actualisation activities, or to the 'ilities' in terms of, for example, the reliability, availability, and maintainability, and the aesthetic characteristics of the home.

Similarly, systems engineers recognise that architecting is both an art and a science; the art of the architect is to apply imaginative skill and knowledge to create the form of a mechanism that will provide the clients functionality requirements; the science of architecting is to quantify the composition, behaviour and performance of the solution.

The architectural concept is the basis of the solution mechanism and its use. It has an all-pervasive influence on the engineering, operational and environmental infrastructure that supports the solution throughout its lifetime.

Consequently the System Architect needs to be able to show the appropriateness of the solution in terms of its architectural attributes; in a way similar to how an architect of a building demonstrates its aesthetic, functional and structural qualities. In addition, the

system architect needs to be able to justify the quality of the design solution especially in relation to its integrity and robustness.

Designers describe the holistic characteristics or 'wholeness' properties of a design solution by reference to its 'integrity'. There are many manifestations of holistic properties, for example, functional compatibility, interconnectivity, data flow, information flow, strength, capacity, 'ility' compatibility, tolerance to variation, compliance.

Engineers use the term 'robustness' to describe the strength characteristics of a design solution. In particular, it refers to the ability of a design to sustain its performance whilst being subjected to variation; these variations include both internal and external parameters. Typically, the robustness of each design function is evaluated under conditions of normal operations, abnormal operations and failure mode conditions.

Both terms have a lot in common. For example, the strength of a pin joined frame depends on both the arrangement of forces as well as the properties of each strut. What is important is the deflection of the assembly under load; it is not possible to differentiate between the integrity of the arrangement and the robustness of the arrangement without a great deal of refinement. Therefore, to avoid any ambiguity within this thesis the word integrity will be used to describe the collective meaning of both integrity and robustness.

To determine the integrity of the solution the Architect needs to have a description of the structure of the system architecture that will support its population with the design definition that provides the capability values of the solution, namely:-

- How the solution works as a machine to provide the required attributes.
- Quantitative evidence that describes the qualities of the architectural solution.
- Quantitative evidence that justifies the integrity and robustness of the solution.

This thesis is an attempt to propose and elucidate a generic means of determining the integrity of candidate architectural solutions.

1.2 Systems Architecting

Systems architecting is defined by Maier and Rechtin, 2002, [5] as: -

> 'The art and science of designing and *building systems* (erecting buildings)'

It is a slight adaption of the definition in Webster II, [6]. This compares with the definition of architecture and architect found in English sources.

> 'The art and science of designing and superintending the *construction of systems* (erection of buildings) and similar structures.'
>
> 'A person qualified to design *systems* (buildings) and to supervise their *construction* (erection).'

(Note. Words in italics have been substituted by the author to make them generic objects.)

Appendix C of Maier and Rechtin, [5], is entitled 'On defining architecture and other terms'. It contains definitions from five different sources. Key words include: -

> art, science, concept, style, process, environment, evolution, value, specification, designing, construction, building, cost, risk, components, connections, structure, interrelationships, organisation, execution.

The English view implies that both 'architecture' and 'architect' have the responsibility as the controlling mind for what is produced; the original Greek definition defined architect as the 'Director of Works'. The process of architecting uses both qualitative and quantitative approaches to design decision-making. It acknowledges that it is both an art and a science.

Specific forms of architecture are recognised; buildings, marine vehicles, computers, communication networks. Other forms of definition are used by different industries; e.g. Chief Designer for aircraft, Engineer for civil construction works. Designer, a person who devises and executes designs, as for works of art, clothes, machines etc. where design is defined as 'to work out the structure or form of something; a coherent or purposeful pattern, as opposed to chaos'. In abstract form, a design may be described mathematically as a multi-optimum inverse problem.

It is acknowledged that many product types involve the formal identification of the architect's role with specifically allocated tasks. As a management structure, it solves many coordination problems, as there is 'someone' who will use their best skill and judgement to set critical parameters. Once set, these enable progressive refinement of most other implementation and characterisation parameters.

The role of the system architect is to create a solution that provides the required functionality and performance, that is robust, that meets 'ility' (e.g. produce-ability, reliability, maintainability, availability) expectations, and is cost effective. The architect has to have a concept of the machine that will meet all these expectations.

With respect to the role of systems architect we must recognise that the modern day architect is quite different from that during the post war period. Then the role was concerned with the ability of technology to provide the functionality and performance, and the feasibility of specification compliance was central to business and commercial considerations. Architects from that era had the benefit that functional design was intimately related to the functionality of the components used for implementation. Now, the functionality of applications is described in abstract terms that bear little relationship to the functionality of the technology being used for their implementation. In addition, the ability of technology to provide the requisite performance is taken for granted. The current trend is to define needs in the form of a capability requirement. The emphasis is to describe needs and functionality with considerable precision, and the comprehensiveness and stability of the Statement of Requirements become factors central to business and commercial risk.

Therefore, the modern architect carries the somewhat daunting responsibility of ensuring that this 'top down' mentality is fully reconciled with 'bottom up' reality. The role of the architect or the architecting process is to create a collection of 'things' that, when put together, provide useful properties that are uniquely generated by the collection in unison. Therefore, it is implicit within all these definitions that:

- the system must work as intended,
- it shall provide the emergent properties required by the sponsor, and
- it shall not provide emergent properties that are undesirable and not wanted by the sponsor.

In general, a system is defined as a structure that embodies lower level components in a relationship.

The structure has emergent properties that provide performance characteristics based on the operational characteristics of the sub-systems in the context of their environments. However, this research programme is intended to support engineers responsible for and involved in the design and supply of complex products. It is concerned with the class of systems that are generally called 'hard' systems. The term 'hard' is a colloquial term used to differentiate between 'soft' systems (i.e. human activity centred systems, as defined by Checkland et al, 1999, [7], and traditional rule-based (unconscious) applied technology systems.

Even though these terms are in common use by practitioners and academics, many system engineers (including the author) do not agree with the implied separation of type as many applications have both 'hard' and 'soft' attributes. From a design perspective, the more important issue is to be able to create a formal structure of the system of interest, whether it has human centred activities or not. Therefore, the author proposes the following definition to clarify the focus of interest for this thesis, as follows.

> 'A system is defined as a structure, which embodies sub-systems in a formal relationship, which has emergent properties that provide the user with only the required and tolerable set of functional performance characteristics.'

Consequently, the principal assumption is that all the properties of the system and its constituent parts are formally definable in a structured relationship. Therefore, it is entirely reasonable to expect that the architect is able to predict the emergent properties of the system and assess their conformance with the intended requirements (formal or informal).

System architectures are formed from a union of requirements, functionality, technology and implementation. Throughout this thesis, the term Design Synthesis is used to define the activity that provides the union.

1.3 Design Integrity

The terms integrity and robustness are used by engineers to address that part of the design process domain that is concerned with the dependability of the solution in relation to both internal and external parameter variation. The words integrity and robustness are often used interchangeably, even though integrity generally refers to the 'joined upness' characteristics of the solution, and robustness refers to the capacity characteristics of the solution, particularly with respect to its strength when subject to load variation.

Its application takes many forms, for example:

- basic strength or capacity to fulfil the role
- inter-connection compatibility
- functional consistency
- measurement and metrication consistency
- variational tolerance (to external or internal characteristics or stimuli)
- static and dynamic stability
- failure tolerance

Most integrity evaluations are associated with particular functional viewpoints wherein each viewpoint must be defined with consistent functional components and metrication.

System architectures add complexity because of multi-functional interconnectivity.

This study is particularly concerned with design synthesis activities that enable the content and integrity of the emergent properties from the architectural solution to be estimated. These estimates include the capability and functionality of its constituent components in the context of their interrelationships in the system structure, and their operational characteristics in the intended environment.

The rationale for this focus is that experience demonstrates that many systems fail acceptance compliance at initial offering. The common reason is that many 'unwanted' emergent properties that require correction are identified during the evaluation phases.

The Pugh Model described in 'Total Design' 1990, [8], states that "there are no alternatives to meticulous and thorough approach to Product Design Specification (PDS) preparation in a competitive world".

It seems as though there is a universal assumption that the architectural structure is held in someone's head or it is knowledge held in common by a select group and that progressive refinement of its component parts will ensure that the integrity of the structure is maintained. So, either 'it' is simple - for example a child's scooter – or 'it' is so

complex – for example a car - that the integrity of the design emerges from a multiplicity of lower level decisions.

The implication is that someone, somewhere, somehow, is able to comprehend the totality of knowledge to compile a competent PDS for each target application. The reality is that the sheer size of many systems makes this impractical.

2 SYSTEMS MOVEMENT – CAPABILITY REVIEW

2.1 Introduction

During the last century, a consequence of our increasingly complex organisational and technological capabilities has been the need to provide coherent means to address multidisciplinary problems. Many problems cannot be addressed in the context of a single discipline and now many multidisciplinary combinations have become specific specialisms in their own right; exemplars include biophysics, chemical engineering and informatics.

It is generally recognised that specific attention to the solution of large-scale engineering developments in a structured manner emanated from the World War II period. The primary examples are those associated with defence systems associated with sea/ground-to-air and air-to-air operations. Such solutions involved a mixture of capabilities and the task of the system engineering team was to understand the operational needs and integrate technologies, skills and resources towards a common goal.

The growth in scale of product programmes has been paralleled by the growth in the size and capability of both public and private institutions. These have come to be regarded as systems and it is no coincidence that the definition of Systems Engineering, in the 'Guide to the Practice of Systems Engineering', INCOSE Handbook, 2006 v3, [9], encompasses both, as follows: -

> "Systems Engineering is an interdisciplinary approach and means to enable the realisation of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem. Systems Engineering considers both the business and technical needs of all customers with the goal of providing a quality product that meets the user needs."

The English Dictionary [10] defines system engineering as:-

> "The branch of engineering based on systems analysis and information theory concerned with the design of systems.",
>
> where a system is: -
>
> "A group or combination of interrelated, interdependent, or interacting elements forming a collective entity; a methodical or coordinated assemblage of parts, facts, concepts etc."

While many advocates promote systems engineering as a distinct discipline, practitioners as a collective body are highly eclectic in terms of picking the best from many disciplines and cloaking them with the 'systems engineering' attribute. The result is that the definition of the domain has supported inclusiveness rather than specialisation. Many attempts have been made to delineate core disciplines that are regarded as being essential components of the discipline. Nevertheless, to date, this has eluded the systems engineering professional community. The best that has been achieved is a delineation of the skills required by practitioners who describe themselves as systems engineers.

Nevertheless, the body of knowledge that practitioners are able to draw from is considerable. The sources of knowledge include operations research, industrial engineering, systems analysis, systems dynamics, systems thinking/soft systems methodology, project management, control theory, and many aspects of specialist engineering disciplines; see e.g. 'Confronting an Identity Crisis - How to Brand Systems Engineering', Emes, Smith and Cowper, 2005, [11].

This analysis does not specifically identify the contribution made by applications specialists although the breadth of the domain is exemplified by the fact that the article identifies seventeen professional societies that could identify systems engineering as a core competence.

The sources that now form the body of knowledge available to the systems engineer stem from, at least, the following domains.

- Applications engineering
    - Defence and aerospace engineering
    - Automotive engineering
    - Chemical processes
    - Transportation logistics
    - Utilities
    - Mass Transportation
    - Telecommunications and Informatics
    - Computer based systems engineering
    - Organisational and management systems
- Design environments
- Modelling and Simulation

Sciences

- Physical, chemical, biological
- Human
- Mathematics
- Computing
- Operations research
- Systems theory

Each domain develops a holistic approach that is rationalised and supported by reductionism processes, methods and procedures; see Klir, 1991, [12a].

Domain	**Development Perspectives**	**50s**	**60s**	**70s**	**80s**	**90s**	**00s**
Defence and Aerospace	Technology integration	*	*				
	Total cost of ownership			*	*		
	Project/programme process standardisation				*	*	
	Interoperability (international)						*
Automotive	Quality/RAM; (reliability/availability/maintainability)			*	*		
	Simultaneous engineering			*	*		
	Automatic assembly				*	*	
	Concurrent engineering					*	
Petro-chemical processes	Measurement and instrumentation. Automation systems.	*	*	*	*	*	
	Multi-variable control analysis.		*	*	*	*	
	Mass and heat balance optimisation. Plant component modelling.			*	*		
Utilities	Energy and supply optimisation			*	*	*	

Domain	Development Perspectives	50s	60s	70s	80s	90s	00s
Transportation Logistics	Route optimisation. Markov chains and queuing theory.			*	*	*	*
	Supply chain optimisation					*	*
Mass Transportation (esp. rail)	Signalling systems			*	*	*	
	Operational management					*	*
Telecommunications	Networks			*	*	*	*
	Data packaging and routing			*	*	*	*
	Messenging protocols				*	*	*
Informatics	Video conferencing					*	*
	Virtual office						*
	Virtual workplace interactions						*
Computer based systems	Software engineering (including configuration control and change management)				*	*	
	Computing structure				*	*	
	Networks					*	*
	Informatics						*
Organisational and Management	Industrial research; work study	*	*	*			
	Operations research	*	*	*	*		
	System analysis		*	*	*		
	Soft systems methodology				*	*	
	Project/process/organisational management				*	*	*

Domain	Development Perspectives	50s	60s	70s	80s	90s	00s
Design environments	Speciality analysis			*	*	*	
	ILS; CALS, LSA			*	*	*	
	CAD/CAM			*	*	*	
	CASE				*	*	
	Design discipline automation					*	*
	Integrated design environments. STEP; ISO 10303; AP 233					*	*
Modelling and simulation	Technology speciality analysis	*	*				
	Integrated process and realisation Analysis		*	*	*		
	Computing simulation			*	*		
	Behavioural analysis				*	*	*
	System simulation					*	*
General System Theory	Cybernetics	*	*	*	*		
	Control and automata theory	*	*	*	*		
	Optimisation		*	*	*		
	Self regulating (genetic) systems				*	*	
	AI/Neural networks				*	*	*
	Fuzzy logic				*	*	*
	Interconnected structures			*	*	*	

Table 1 - Chronology of Development of Systems Engineering Capability – Applications Domains

Table 1 has been compiled by the author to show the chronology of the major subject areas that have dominated the development of each domain since the middle of the last century.

The author has sought to highlight those aspects that have contributed to the wider development of system engineering capability. In addition, it illustrates the eclectic nature and provenance of the knowledge base that is available to current practitioners. However, the reader should appreciate that it does not purport to provide a definitive historical record of the existence of the domain or its specific contribution to the knowledge base.

2.2 Aerospace and Defence

The general-purpose engineering process structure was defined by Goode and Machol, 1957 [13], in the 1950s as consisting of the following main stages: -

Initiation

Organisation

Preliminary Design

Principal Design

Prototype Construction

Test, Training and Evaluation

The process was dominated by design techniques associated with the use and analysis of technology, particularly those associated with control engineering.

At a Cal Tech conference, see Miles 1971, [14], the structure had evolved to the following.

Goal definition or problem statement

Objectives and criteria development

Systems synthesis

Systems analysis

Systems selection

Systems implementation

The MIL-STD-499 'Systems Engineering Management' [15] was first published in 1969 (MIL-STD-499 A/B followed in 1974) and was adopted for general use by most defence acquisition programmes and became the standard used by most contractors.

The emphasis then progressively changed technology application to operational application. In 1981 the term 'life cycle engineering' was introduced into systems thinking (See e.g. Blanchard, 1992, [16]) and the emphasis moved on to the issues surrounding the deployment and in-service logistical support of large-scale complex systems. Whole life cost became an important focus for comparison of competing solutions.

Various new publications were produced by the DOD, including the 'Systems Engineering Management Guide', 1986, [17], and MIL-STD-499 version C was published in 2005. However, it was recognised that considerable benefits could be derived by using the generic principles to major non-defence orientated programmes. So work began to compile documents that were of generic application.

Consequently, these were followed by the EIA/ANSI Interim Standard 632, the IEEE 1220, the Systems Engineering Handbook by NASA, 1995, [18], and the INCOSE Systems Engineering Handbook, 2002, [9]. Finally, international accord was achieved with the publication of 'Systems Engineering – Systems Lifecycle Processes' (ISO 15288) in November 2002 [19].

The most recent thrust has been dominated, as usual, by the USA Defence Agenda, by interoperability issues and the need for collaborative force defence capabilities. The result is the 'Architecture Framework', [20] formally issued in 2002.

In summary, the emphasis of the discipline has moved forward in stages that reflected the complexity of the design task as perceived at that time. In chronological order, the focus of interest may be summarised as follows, approximately in decades since the 1950s.

50s	Applied Technology
60s	Appropriateness of solution concept
70s	In-service affordability
80s	Network systems and Lifecycle management
90s	Interoperability

It can be seen that the historical record has reflected the challenges imposed by the growth in size and complexity of the design tasks tackled by the aerospace and defence communities.

2.3 Automotive

This industry concentrated on automatic assembly and in-service reliability. Many techniques were developed in Japan. These particularly addressed the poor quality of manufactured products in the immediate post war period. Quality techniques included the tolerance compatibility methods developed by Taguchi, 1987, [21], and the need to formalise the design of human-centred components. This came to be known as Quality Function Deployment, e.g. Cohen, 1995, [22], a structured approach to support the design of components to meet driver and passenger preferences; in effect applying a soft systems approach to requirements definition.

The term simultaneous engineering was coined to describe the use of multi-disciplinary design teams. This complemented the traditional 'over the fence' design process management organisations, whereby each part of the design was completed in stage sequence with little reference to the preceding or subsequent stages. Although the result from each stage was correct at a detailed level, the overall result from the integration of all stages of design was the emergence of incompatible attributes, e.g. Hartley, 1991 [23].

The Simultaneous Engineering process, more correctly, renamed as Concurrent Engineering, ensured that at each stage of design all stakeholders were involved in the decision making process so that their different needs were reconciled at the point of decision-making. This ensured that all stakeholders were satisfied with the emergent properties of the finished item. These techniques were instantiated into all stages of the lifecycle, activity planning, product design, process design, production and post purchase support. The result was an enviable reputation for reliability that transformed the image of the 'Made in Japan' label. Now the technique has been applied to many other fields to refine aspects of utility in a structured manner. Nevertheless, the reader should appreciate that the technique becomes increasingly difficult to apply when the available solution space diminishes in relation to growth in variety of different objectives, viewpoints or disciplines that must be considered.

2.4 Petro-chemical and allied Process Industries

These industries concentrated on the development of measurement instruments and control components. They have a long pedigree of commercially focused developments. Control systems have been based on single input/single output schemes. However, as plant components became more integrated e.g. heat exchangers and distillation columns, many problems of non-linear and multi-variable control needed to be solved. The state space concepts provided the means of analysis. Computer control replaced single loop control with consequential enhancement of overall plant control and management capabilities.

2.5 Utilities

The developments in these industries, including that of nuclear power, have been closely aligned with those associated with petro-chemicals. Particular emphasis has been placed on optimisation techniques, and plant and infrastructure status data management systems.

2.6 Transportation Logistics

Although transport routing problems have been long-standing (e.g. shipping logistics), the development of retail supermarkets, global product sourcing and fast moving consumer markets provided the impetus for major development during the last two decades. Developments in the use of Markov chains and queuing systems have been particularly relevant.

Communications technology combined with computer-based systems enabled very comprehensive logistical control systems to be developed.

2.7 Mass Transportation

These industries have made extensive use of the capabilities in telecommunications and informatics to develop data collection, signalling and traffic management systems. These parallel the characteristics of Command, Control, Communications and Intelligence (C^3I) systems developed in the defence industries.

2.8 Telecommunications

The expansion of telecommunications capability has paralleled the developments in computer-based technologies. This enabled the development of state transition methods for data package switching techniques, and messenging protocols were developed for use in, for example, automatic teller machines and mobile communicators.

2.9 Informatics

The extensive developments in image processing enabled basic video conferencing facilities to be developed in the 80s. The World Wide Web (WWW) has enabled virtual networking to be accessible to both home based and professional users. Originally confined to text based message transfers, the capability now includes single and video imagery, and audio products. The virtual office concept, originally confined to single person transactions, is being developed to include various forms of work place combined with multiple person interaction. The development of virtual systems is arguably one of the most demanding challenges for the current generation of system architects.

2.10 Computer based systems

Over the last 30 years, the development of digital systems has resulted in the dominant need for both industry and academe to invest in computer science and software engineering processes, methods, procedures and tools. By the beginning of the 80s it had become clear that informally structured software packages had inadequate reliability and unwanted emergent characteristics. Corrective action processes had become unsustainable and uneconomic.

Structured methods were developed, e.g. Yourdon, 1979, [24], and these are now accepted as the norm. Since then it has been possible to undertake large-scale programmes without undue risk, when supported by sensitive management and semi-automated design environments.

These have been developed to include, for example, module specification and construction, algorithm evaluation, object code generation, behaviour simulation, target code performance evaluation using emulators combined with configuration control and change management systems.

Research into software robustness provided various design constructions that provide reliability at module level. See e.g. Dijkstra et al. 1972, [25], and Hoare, 1985, [26]. Further analysis methods, namely data flow analysis, static and dynamic analysis and theorem proving techniques, enabled reliability to be built into the development processes, see e.g. Le Carre,1979, [27].

Recent developments have been associated with the need to define the requirements in detail. To ensure operational compatibility between software products, precise definitions of interface and functionality are required. To ensure that the dependability of 'systems of systems' for safety related applications is achieved, precise definitions of interface and functionality are required.

2.11 Organisational and Management systems

The foundations of scientific management can be traced back to what came to be known as industrial engineering. The specialisation of manufacturing activities into narrowly defined tasks became known as 'Fordism'. The repetitive nature of the work resulted in loss of morale and personal esteem and consequentially loss of productivity. The social impact of specialisation and efficiency was investigated by F. W. Taylor (see 'The Principles of Scientific Management, 1911 and 1947, [28]). He devised methods that maximised the efficiency of manual tasks in steel-works and bricklaying.

Such methods became known as time and motion studies. The application of these and similar scientific principles to the management of complex systems became known as Operations Research.

Many theories of the 'firm' were invented. However, these concentrated on the social aspects of team working until Stafford Beer et al, 1979 and 1983, [29a, 29b] attempted to apply the scientific method to operations management. They applied the ideas of cybernetics to the need to manage change throughout an organization in an uncertain environment and formulated neurocybernetic organisational and operational models of the enterprise.

The advent of the Cold War and the need for nuclear weapons superiority resulted in the initiation of the nuclear submarine programme. The programme management team, sponsored by the US Navy, developed the Programme Evaluation and Review Technique (PERT), 1957, [30], and the programme became an iconic model for the management of complex programmes.

Since then the technique has been the foundation of programme planning techniques throughout the world. Many enhancements have been incorporated and flexibility of use has been enhanced alongside the growth in availability of computer power and corporate IT environments. Now fully integrated planning, organisation, resource, logistics and budgetary control systems are available to most corporations.

All these approaches and methods were goal oriented in some way. However, the fundamental review of systems carried out by Peter Checkland et al at Lancaster came to the conclusion that the performance criteria for many systems was obscure. Studies of such loosely defined systems became known as Systems Thinking. Systems thinking, as defined by Checkland et al, 1999, [7], is an epistemology which, when applied to human activity, is based upon the four basic ideas: emergence, hierarchy, communication and control. When applied to natural or designed systems the crucial characteristic is of emergent properties of the whole.

Checkland also defines the hard systems methodology for tackling real world problems in which the objective or end goal is taken for granted. Whereas soft systems methodology is used for tackling real world problems in which the 'known to be desirable' end goals cannot be taken for granted. Viewpoint analysis of such systems has become known as Soft Systems Engineering.

These definitions cannot be complete. Firstly, it excludes abstract system constructions associated with generic behaviours and secondly, one only has to review, for example, an air traffic control system, to see that the difference between hard and soft subsystems is blurred, if not ambiguous. Clearly, the rules to prevent collisions and such like are well defined. However, there are many human centred aspects associated with the comfort and stress of traffic controllers and their organisation and management structures will be viewpoint dependent and, in some cases, only loosely defined.

Nevertheless, the techniques have been shown to be useful and the current definition of systems engineering enables an inclusive approach to be taken so that the operational and human aspects of any system can be addressed in a manner that enables full and appropriate integration with real (i.e. physical, chemical and biological sciences) world aspects.

2.12 Design Environments

Designers have always needed to be able express their ideas in a form that is understandable by others. Many forms of construction aid have been developed especially in the form of documents, drawings, models, prototypes and templates.

In engineering the design definition is captured in what is known as, for example, the 'Material Record Inventory' (MRI). It provides the definitive statement of the design in relation to its intended application. It is used by all who have a need to know about the system, how it works, how made, how to maintain, repair and dispose. Ultimately, it is the knowledge base that underpins any defect review proceedings.

The growth in scale and complexity of systems has resulted in a corresponding growth in the assets that form the substance of the MRI. Paper based and physical item supported MRIs reached such complexity that these too resulted in asset systems with their own weaknesses of ambiguity, sustainability and cost effectiveness.

The DODs Continuous Acquisition and Logistics Support (CALS) initiative, 1993, [31a, 31b], was an important enabler in terms of electronic drawing standardisation and documentation production for both production and logistics support activities. Its initial focus was to transfer support related engineering and operational instructions from paper-based environments to electronic media based environments. Interleaf became the dominant package for documentation production and change management, now overtaken by the use of Microsoft Office products.

Engineering drawings were transferred to electronic form using STEP (the Standard for Product Data Exchange, ISO 10303, [32]) and the development of the Express language. Substantial investment into CAD, CAM and Integrated Manufacturing systems using wire frame model techniques has led to seamless transfer of design to machine centre. The treatise on the design of mechanical components by Pahl and Beitz, 1977, [33], provided a detailed exposition of standard design.

Considerable output by researchers like Ishii and Kusiak have resulted in improved front end Man Machine Interface (MMI) for mechanical part design. Design For Manufacture (DFM) capability has developed for automated production and assembly. However, the work is limited in that it only addresses physical properties.

The processes and procedures of change management and configuration control are intrinsic to a reliable MRI. The development of software engineering support processes has been an important enabler of providing semi-automatic environments for library tools that record design definition problems, causality, impact, correction and embodiment.

The complexity of information now carried in large-scale systems and the impact of globalisation and virtual team working has been supported by the 'Systems Engineering Data Repository and Exchange System' (SEDRES, [32]), programmes to develop ISO 10303; AP 233. This standardises the definition of the information object to facilitate multi-media electronic data exchange.

A typical construction of a neutral system-engineering repository based on AP 233 integrates specific tools that encompass the following domains: -

- Requirements Analysis
- Requirements Baseline Validation
- Functional Analysis
- Functional Verification
- Synthesis
- Physical Verification

(Courtesy of JFE Johnson, SEIC, BAE Systems plc.)

2.13 General Systems Theory

The term system is defined as "a group or combination of interrelated, interdependent, or interconnecting elements forming a collective entity; a methodical or coordinated assemblage of parts, facts, concepts etc. (See Collins Millennium Dictionary, [10]).

The mathematical representation of this definition is as follows.

$$S = (T, R) \qquad \text{(Equation 2.1}$$

Where ... S is the system

T is a set of components (things)

R is a set of relationships on T (system hood; connectivity)

Its simplicity however belies the richness of the field. There have been many attempts to develop generic system models. Exemplars include contributions from researchers like L Von Bertalanffy, 'General Systems Theory; Foundations, Developments, Applications', 1952, [34], W R Ashby, 'General Systems Theory as a new discipline', (General Systems Yearbook, 1964, [35]), R C Conant 'Laws of Information which govern systems', 1976, [36]), J W Forrester 'Principles of Systems', 1968, [37]), and L A Zadeh 'Outline of a new approach to the analysis for complex systems and decision processes', (IEE Trans. On Systems, Management and Cybernetics, SMC-1(1); [38]). G J Klir addressed the problem of structure and component connectivity in 'An Approach to General Systems Theory', 1969, [39]. These are just a few examples of the large numbers of contributions that have been made to the field.

Later G J Klir brought together the main ideas that now form the basis of current systems science in his book 'Facets of System Science', 1991, [12a]).

Von Bertalanffy addressed the problem of natural systems wherein the quantities associated with some parts are related to other parts also with associated quantities. He established principles of progressive differentiation, output/input dependency and prediction.

Ashby addressed the problem of state determined systems (especially for continuous systems) and 'black box' identification. Zadeh developed, with others, the state space approach.

Mesarovic formulated a formal mathematical construction of systems which he described in the 'Mathematical Theory of General Systems' 1972, [40].

He provided an algebraic foundation theory that addressed structure, closure, associativity, cardinality, sets, functionality, state objects and state space, as well as goal-seeking systems and multi-level systems. Wymore addressed the issues of time-based systems, both continuous and discrete. (See 'A Mathematical Theory of Systems engineering: The Elements', A. W. Wymore, Wiley 1967, [41]).

The invention of graph theory was attributed to Euler in 1736, See Biggs et al, 1986, [42a]. Then, in the twentieth century, the interest in connectivity relationships provided the motivation to develop graph theory further. Harary, Norman and Cartwright consolidated the mathematics associated with 'structure' into what is known as the theory of directed graphs [42b]. Initial applications were used to structure sociological, psychological and genetic problems. Current applications include physical science and engineering, communication systems, artificial intelligence, and international finance; see Beineke, 1997, [43].

Klir reviewed various approaches for modelling systems, especially those concerned with reconstructability, and developed the General Systems Problem Solver (GSPS), see e.g. 1969, [39], The GSPS differentiated between types with universal structure and coupling (UC), and state transition (ST) structures. He then used these concepts to address discrete, combined discrete and continuous, probabalistic and time varying systems. Although it was generically a single layered approach, no guidance was provided on its application to multi-layered structures.

Mathematical system theory developed in a similar time frame mainly in automata theory and graph theory.

In the immediate post World War 2 (WW2) period, the main focus of attention was component integration driven by the control and automation community.

The process of reductionism enabled the development of techniques for component representation and system networks. Block diagrams and signal flow theory became the main stay of the control discipline; see e.g. 'Control Systems Analysis', Truxall, McGraw Hill; [44a, 44b]. Simulation techniques were based on the need to solve differential equations and component functionality was modelled in terms of their physical science properties; (See e.g. 'Engineering Systems Analysis', A. G. J. MacFarlane, 1964, [45]). In 1961 Paynter proposed that component models be based on energy transfer relationships; (See 'Analysis and Design of Engineering Systems', H.M. Paynter, 1961, [46]). These became known as Bond Graphs; see e.g. 'Analysis and Simulation of multi-port systems', D.C. Karnopp and R.C Rosenberg, 1968, [47]).

Studies that investigated the relationships between computational systems and human interaction were initiated by Norbert Wiener and von Neumann in 1944. These studies developed the discipline of Cybernetics; see Wiener, 1948, [48] and W R Ashby, 1956, [48a].

Measurement and Instrumentation science was developed by R.D. Watts, P. H. Sydenham, and L. Finklestein; see 'The Elements of Design; The Design method', 1966, [49], and 'Measurement and Instrumentation Science - an analytical review', L Finklestein, 1994, [50]. They addressed the highly interactive issues associated with measurement and the design of measurement instruments.

Automata theory and the concept of the state space, state transition transformed our view of control engineering from a SISO (single input single output) capability to a full multivariable MIMO (multiple input multiple output) capability. See e.g. 'Topics in Mathematical System Theory', Kalman, Falb, and Arbib, 1969, [51]. These were followed by the development of mathematical optimisation methods; for the use of variational calculus, see M R Hestenes, 1966, [52]; for the use of the Wiener-Hopf Equation, see e.g. 1960, [53]; for the use of Pontryagin's Maximum Principle, see 1962, [54]; for the use of Linear Programming, see 1962, [55a,55b]; and for the use of Bellman's Dynamic Programming, see 1962, [56].

2.14 Modelling and Simulation

It is almost impossible to be a practitioner in the systems field without reference to some form of model. Traditionally, modelling has been used to produce some form of prototype of the product or component that enables some aspect of performance to be evaluated. When the model is defined mathematically, the form and functionality can be simulated. Such models enable the form, behaviour and performance of the final product to be evaluated. In addition, models that represent the functional structure pertinent to a specific viewpoint can be constructed.

What counts in all cases is the validity of the predictions produced by the model for the purpose intended, consequently, all models need to be subject to both verification and validation processes.

2.14.1 Models of Technology Components

Models are used to describe the components and interactions within a system of interest. Science, technology and component based models have been developed to support integration and standardisation.

The development of servo-mechanisms and control theory was accompanied by advancements in the use of graphic techniques to construct models of servo components. Macfarlane developed generic component models that met the need for analytical consistency in dynamic simulation; see e.g. 'Engineering System Analysis', A. G. J. MacFarlane, 1964, [45], and 'Dynamical System Models', A. G. J. MacFarlane, 1970, [57].

Interest in process automation created interest in the development of models for measurement systems. For example, Finklestein and Watts commented about measurement, as follows, (See 'Measurement as a systematic study', 1969 [58]).

> ...Measurement involves the establishment or maintenance of functional relations among physical variables...Measuring instruments are generally built up by interconnecting simpler components...Measurement is concerned with systems and the functional relations maintained by them...It follows that the language and methodology of system science can be usefully employed in the study of measuring methods and techniques...
>
> ....Measurement is concerned with systems and the functional relations maintained by them...Systems and their elements are described by equations relating their inputs and outputs. For linear systems the conventional language of transfer functions and transfer function matrices is thus available to express system relationships, while for more general systems it is possible to use the concepts of state and state space. Block diagrams and signal flow graphs offer a powerful way of visualising relationships....

In the electrical/electronic field, considerable advancement has taken place in digital device design, production, test and verification. Particular emphasis has been given to VLSI, the development of VHDL, tools for 'systems on a chip' and integrated workstations (e.g. Mentor Graphics) for the electronic engineering discipline.

2.14.2 Models of particular phenomena

Many problem specific models have been developed to facilitate performance analysis of interacting components. The growth in availability of high computational power has enabled large scale simulations to be produced. However, as the complexity of both hard and soft systems has grown, the need to integrate many simulations into a consistent single entity has arisen. Current methodologies are concerned with the validation and verification of large scale simulations produced from a 'patchwork' of simulation components.

2.14.3 Models of computing arrangements

Models based on representations of the input-output functionality and control of the system components enable overall behaviour and performance to be estimated. Models that represent the precise internal construction of its system components in functional terms enable the design to be verified. In each case the algorithmic representation of each component and its interrelationships is crucial to the utility of the model.

2.14.4 Models of Processes

Systems that are formed from the interaction of many components include transportation, production and automation processes. Such models have a wide range of theoretical techniques to exploit, including, for example, conservation of mass and/or energy, queuing (Markov), control (control), stochastic (decision theory), games (von Neumann), genetic, neural network, etc. constructions.

2.14.5 Models of Soft Systems

There are many forms of human interaction input and response interface, both by individuals and groups. These soft system models are usually associated with a particular viewpoint. They range from simple input/display evaluation, to communication and control centre operations, institutional behaviours, combat effectiveness including gaming to evaluate tactical effectiveness.

2.14.6 Models for System Architecting

In the systems engineering field, the search for an effective architectural modelling technique and language is only partially complete. The concepts from automata theory and computer science have been used to formulate various potential solutions.

During the decade that spanned 1990 Wayne Wymore and others attempted to formalise the DOD design process model into an executable form using constructions based on algebraic set theoretic and state transition methods: See 'Model Based Systems engineering', A Wayne Wymore, 1993, [59]. The problem facing the system designer was extremely well defined in terms of the need to understand the operational need and

provide a technologically viable, verifiably compliant and cost effective solution. However, the syntax that embodied the DOD process model in abstract algebra form has resulted in a language form that obscures the definition of the actual design problem. Consequently, it is difficult to comprehend.

The development of model based specifications continued and Cohen, Harwood and Jackson developed a usable form of Pascal like pseudo code that is described in 'The Specification of Complex Systems', by B. Cohen, WT Harwood, and MI Jackson, 1986, [60]. This followed industry practice whereby the problem of ambiguity inherent within the general use of English language for specifications was overcome by the use of algorithmic constructions to define requirements in a precise manner. For example, the Vienna Development Method (VDM) is based on formal specification and syntax, 1990, [61]. It has very strict typing and data structures, and is useful for model-based systems.

In parallel, the computer science community has progressively developed methods that describe the interactions and behaviour of computer based systems. All such solutions in general use have adopted a graphical means of problem description.

Languages, e.g. MASCOT, see Jackson and Simpson, 1975, [62], were developed for applications that involve a wide range of operator options. These involve the definition, management and validity of data pools.

Computing system design environments that incorporate state transition methods of description and evaluation have been developed; see e.g. 'State Mate',1987, [63]. These enable precise evaluation of computing structures to be made.

In the systems engineering field, work associated with behavioural modelling of system architectures, sponsored by TRW, resulted in a method called Requirements Driven Design (RDD). This was supported by a tool called 'RDD-100', from 'Ascent Logic'; see 1990, [64]. It was developed by Mack Alford and based on the simple concept that any design could be represented by a two dimensional network of parallel or serially dependent functional design elements. The design elements could be event or resource dependent, making it suitable for rapid prototyping of computer based machine systems. Despite the initial enthusiasm by the systems engineering community, sponsorship waned and development momentum was lost.

One functional modelling technique that has gained wide acceptance is IDEF0. Its semantics enable functional definition, feed forward, feed back and control. This emanated from the Integrated Computer-Aided Manufacturing (ICAM) programme sponsored by the U.S. Air Force. The acronym represents ICAM Definition and '0'

represents the first technique generated by the programme. The full complement of IDEF models is shown in Table 2.

IDEF Family	Capability Description
IDEF0	Provides functional or process model
IDEF1	Provides information needed to support functions in a model.
IDEF1X	Data model using relational theory with an entity-relationship technique.
IDEF2	Provides a dynamic model.
IDEF3	Provides both process and object state transition model.

Table 2 - The Capability of the IDEF Family of Design Modelling Languages [65]

During the 70s the telecommunications industry, and Ericsson AB particularly, needed to address the design of software for networks of computers.

The emergence of systems consisting of networks of computers came to be known as large scale 'software intensive programmes'. Ericsson tackled the problem of monolithic complexity by basing their designs on decomposing large blocks into smaller, more understandable, blocks. Ericsson devised a means of using these blocks by defining a series of 'traffic cases'. This gave them the means of describing all the blocks and how they fitted together. Also, many of these programs relied on event-response programs that involved human interactions and sequential functional processing. This resulted in the development of the 'actor' and the message sequencing syntax now defined in the Specification and Description Language (SDL).

SDL is a design language that is specifically for reactive, discrete systems. There are many forms of such systems, including, for example, vending machines. Reactive means that behaviour is characterised by responses to external stimuli pertinent to its operating environment. It is not designed for continuous systems, the implication being that the response is immediate and simple e.g. delivery of a bottle of drink or food package. SDL is not suitable for systems with responses that result in complex functions, e.g. an automatic motion control system, that responds to continuous perturbation.

SDL-92, see e.g. 1994, [66], has been adopted by the International Telecommunications Union (ITU), a specialist agency of the United Nations, and is maintained by them. It is widely used in the telecommunications industry for the design of public switching systems and telecommunication systems. It supports the generation of specifications,

designs, implementations in the form of automatic code generation, and documentation. Also, it is used to support other specifications for signalling and network functions.

Although the idea of OO (Object Orientation) for software design had been promoted for some years, it had received little support from the embedded software design community as it failed to address the structural implications of real time execution.

However, in 1994 Grady Booch, Ivar Jacobson and James Rumbaugh combined the OO viewpoint with the Rational Unified Process to create UML, the Unified Modelling Language.
This was primarily developed as a software modelling and development environment that had clear semantics and syntax. It provides a means of both static and dynamic design and evaluation in visual diagrammatic form.

The principal diagrams are shown in Table 3.

Static Model	Dynamic Model
Class diagram	Object diagram
Component diagram	Use case diagram
Deployment diagram	Sequence diagram
	Statechart diagram
	Activity diagram

Table 3 - System Design Model Diagrams supported by the UML and the unified process.

(See J Arlow and I Neustadt, 2002, [67])

These activities have been hosted by the OOG (the Object Orientation Group) and a large measure of success is due to their ethos and commitment to create a truly international standard. In 1989 the OOG was reformed into what is now the Object Management Group (OMG), [68]. In 1997 the OMG accepted the UML as its preferred OO modelling language and UML now stands as the preferred industry standard. The specification for UML 2 achieved certification in October 2004.

Although the systems engineering community identified the potential that both UML1 and UML2 provided a means of modelling a limited range of design applications, practitioners appreciated that the specification did not include constructions for structural decomposition. Therefore, a range of extensions has been developed to support, amongst other matters, the use of the idea of a class of objects in conjunction with a group of objects that relate to a partitioned group in the context of a decomposition arrangement, as shown in Table 4.

The full specification for the SysML was adopted by the OMG in July 2006.

SysML Diagram Types		
Structure Model	Requirements Diagram*	Behaviour Diagram
Structure Diagram		Behaviour Diagram
Block Definition Diagram*		Activity Diagram*
Internal Block Diagram*		Sequence Diagram
Package Diagram		State Machine Diagram
Parametric Diagram*		Use Case Diagram

Table 4 - SysML Diagram Types with Extensions to UML 2 highlighted with an * [68]

(Courtesy the OMG SysML Official Web Site; http://www.omgsysml.org)

Initially IDEF0 and UML were adopted by the manufacturing and software communities respectively. These enabled formal structures of interactive, time based processes to be constructed and evaluated. So systems architects now have a design language specifically developed to support systems engineering as a discipline in its own right.

The development of a fully integrated design support environment for systems engineering is still some way off. Nevertheless the systems engineering community now has the three core items, a design language, a process and a means of data exchange defined as internationally recognised standards. These will facilitate the integration of current best practice discipline specific support tools, thereby creating a component necessary to support the trend for the globalisation of design capability.

Systems Design Environment Core Specifications	
Designation	Description
SysML	Specification language for systems modelling
DODAF/MODAF	Defines views of specific system engineering processes
AP233 (STEP ISO 10303)	Defines a neutral file format for the exchange of complex engineering structures.

Table 5 Integrated Systems Design Environment Core Specifications

(Courtesy Eurostep Group AB. 2006, [69].)

3 SYSTEMS ENGINEERING PROCESS MODELS

3.1 Review of Current Standards

3.1.1 Introduction

There are numerous documents that describe the systems engineering design process, see e.g. Kossiakoff and Sweet, 2003, [70] and Stevens et al, 1998, [71]. Many companies, together with industry, professional and national organisations have sponsored work with the primary objective of creating an internationally recognised code of best practice. Some documents have had a somewhat shorter life than would be expected from the quality of the material. Nevertheless the need to obtain a process description with international recognition has been the predominant motivation. Consequently the individual works have been absorbed by the working parties empowered to achieve specifications with international credibility.

In summary, just three specifications dominate current operations. The workhorse of the industry has been MIL-STD 499 A/B. It has dominated the aerospace industry for over twenty years and has had a profound influence on the formation of all interim and current specifications. Even though formal support for its maintenance has been withdrawn, its utility continues through its support for many legacy programmes. The efforts of many specification generation groups were eventually recognised by the International Standards Organisation. Now the ISO 15288 standard [19], formally issued in 2002, is being established as the international benchmark for both commercial and defence undertakings, and the Architecture Framework generated for the US Department of Defence [20] is becoming established as the process that underpins the largest multi-national defence programmes.

3.1.2 DOD MIL-STD-499, A and B.

Arguably, MIL STD 499 A/B is the most well known specification as it has dominated the aerospace industry since 1970. It describes a process that supports the design activity from a top-down perspective. It assumes a 'V' Diagram, for example see Buede, 2000, [65], form of product development process and describes a staged process that leads from concept generation to atomic level product definition.

The concept that forms the design model is based on intimate relationships between the three core components of need, functional solution and realisation solution; formally these are called Requirements, Functional Decomposition and Solution Synthesis. A schematic of this 'RFS' design process model extracted from the Specification is shown in Figure 6.

The arrowed lines that link the 'boxes' together show that the basic model of 'R' to 'F' to 'S' is supported by reconciliation processes between 'F' and 'R', and 'S' and 'R'. The intention is to ensure that both the functional solution and the realisation solution are intimately related to the defined requirements. Many practitioners also assume that the model should include a reconciliation process between the realisation solution and functional solution; clearly, it is imperative that the functionality of the realisation solution should match the functional decomposition as closely as practical.

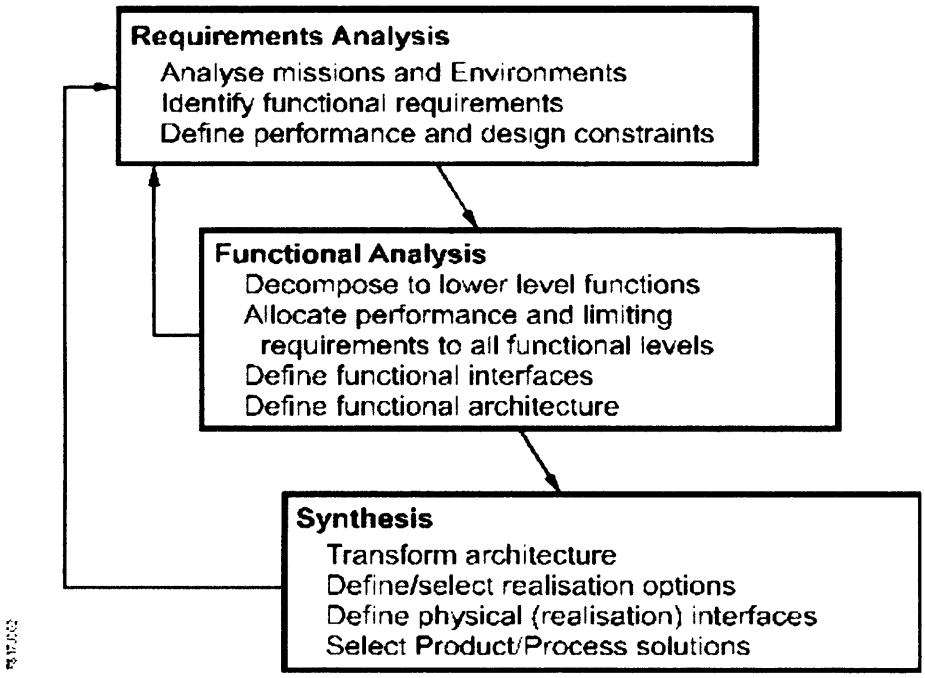


Figure 6 - Schematic of System Design (P2)

The Specification describes in detail what is required to be achieved, in the form of output objectives. However, there is little guidance as to how these objectives are to be achieved. In fact, the literature is substantially deficient in terms of providing procedures and methods that show how the processes are to be supported. In making this assertion the author is well aware that there are rich knowledge bases concerning a number of techniques used by system and software engineers. These include, for example, requirements management, viewpoint analysis, behavioural modelling, object-orientated design, configuration control, to name but a few.

3.1.3 ISO 15288

The international standard was finally published in November 2002. It consolidated the work of many organisations including NASA, the IEEE and INCOSE. Similarly, the processes are described in detail however there is no guidance as to how the objectives are to be achieved.

The route to publication of the International Standard was complex. In summary, the P1220 model was developed in the mid 1990's. It attempted to refine the processes identified by the MIL-499 Model. The model made good sense in that it highlighted some of the useful techniques e.g. FMECA, to support the justification of the proposed solution. However, its prescriptive tone and lack of generality meant that it did not have the widespread support of the system engineering community. The European Space Agency (ESA) published its Standard [72] in 1991; the revised edition was published in 2004. Then INCOSE took on the task of generating a Handbook and offered it as a Standard for general use. For a brief period, the Society of Automotive Engineers (SAE) took responsibility for its publication. Finally, the ISO agreed to sponsor the generation and publication of the international standard. Now it is a document that embodies multi-national best practice.

However, in the opinion of the author, the simplicity of use embodied in the style of the early standards has been obscured by the descriptive comprehensiveness of the current version. Consequently, modern practitioners need to have substantial knowledge of practical engineering processes to be able to interpret its advice for use in real design environments.

3.1.4 DOD C4ISR Architecture Framework.

The C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, Reconnaissance) model enhanced the basic MIL-499 Model to encompass the very large scale systems associated with integrated defence capabilities, and particularly those associated with C4ISR systems of systems concepts. It replaces the generic concepts of Requirements, Functions and Synthesis with Operations, Systems and Technical Standards; these are combined in the context of a detailed lifecycle process model.

The two models are compatible and can coexist. The rationale for the DODAF construct is that it has been developed to facilitate international cooperation during development and operational use.

Recent deployments of multi-national peacekeeping forces have highlighted the problem of systems interoperability. In general, equipment from each participating nation is not compatible, thereby creating many deployment and operational constraints. Therefore, future equipment procurement planning includes interoperability as a major goal. This standard supports this goal by incorporating a detailed exposition of the work items to be completed and delivered at each phase of the lifecycle.

The mapping is shown in Table 6.

MIL 499B System Engineering Model	DOD Architecture Framework
Requirements; detailed expressions of specific needs.	Operational viewpoint; identifies and structures all stakeholder preferences.
Functional Decomposition and Assessment including quantitative estimates of behaviour and performance.	System viewpoint; through the use of components and touching points.
Solution Synthesis and Reconciliation, in terms of the specific implementation.	Technology viewpoint; especially via Standards.

Table 6 Mapping MIL 499B to DOD Architecture Framework

3.2 Integration of Process Models with System Design Models

All these models have been concerned with lifecycle process issues. The advantage of this is that they provide the system engineering community with a common approach to all phases of a system/product lifecycle. Convergence to a common approach, especially by global design teams, will facilitate, over time, cooperation between islands of expertise/resource/capability to enable them to work in harmony towards a common objective. Nevertheless, we have to recognise that the growth in the knowledge base means that 'one size does not fit all' and that the process solution for any particular programme must be appropriate.

Current industry practice is to interpret each of these process models as complete replacements for what has gone before with scant attention to prior knowledge. However, that is erroneous thinking. Capability has been built over many years and the historical record is that these models were developed in response to enable each new scope of complexity to be addressed, each one building on the capability of what went before. For

example, the best way to use the Architecture Framework is in combination with MIL-STD-499B, or another specification that is pertinent to lower level design.

Nevertheless, the author asserts that the issue of a model of system design is outstanding. The issues that are largely missing from the knowledge base stem from management difficulties associated with the complexities in product design and the associated structural evaluation of the architecture.

3.3 System Modelling as a Process

To estimate the emergent properties of a system, systems architects, whether as individuals or as a team, have a mental model of the system that is used as the reference for the design definition and its evaluation. Engineering modelling is primarily used to predict the performance of equipment and set design parameters. The modelling process is outlined in Figure 7.

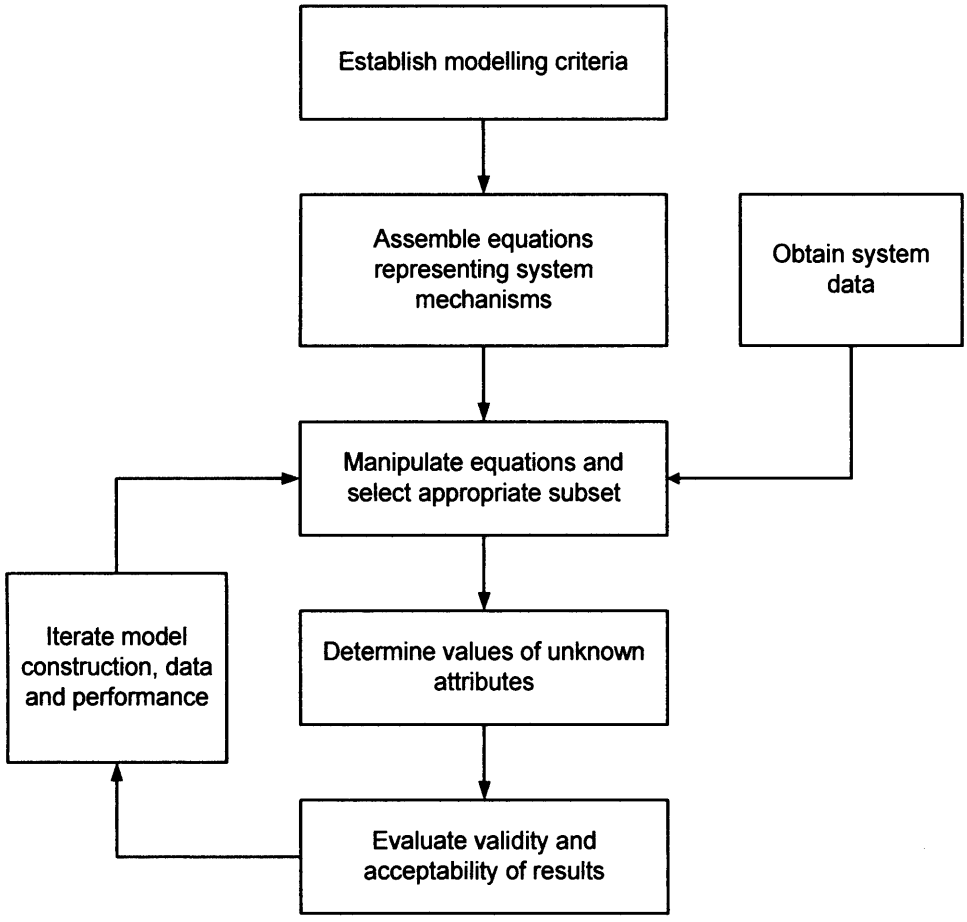


Figure 7 - Schematic of Modelling Process (D7)
(Courtesy of Electronic Associates Ltd. See 'Modelling of Dynamical Systems', Vol. 1. ed H. Nicholson, 1980, [73])

Type	Linear			Non-linear		
Number of independent variables	One	Several	Many	One	Several	Many
Form						
Algebraic	Trivial	Easy	Impractical	Very difficult	Very difficult	Impossible
Ordinary differential equations	Easy	Difficult	Impractical	Very difficult	Impossible	Impossible
Partial differential equations	Difficult	Impractical	Impossible	Impossible	Impossible	Impossible

Table 7 - Classification of Mathematical Representation and Ease of Analytical Solution
(see Franks [73a])

However, mathematics has a limited capability to solve equations, as summarised in Table 7. Further, for equations to be soluble they must be constructed to address the issues of linearity, boundary values, whether the functions are implicit or explicit, and the issue of sufficiency and redundancy; e.g. number of equations = number of independent variables.

To enable these constraints to be met, many techniques are used to assemble the equations for the model. Typically, model components are constructed from an understanding of the generic science, technology, process, from interpretation of experimental data, or by attribute representation. Then the construction of the model is evaluated to determine solubility. Principal amongst these is the use of linearised equations, often combined with the use of high-speed/high-resolution iteration, so that a particular part of the operational space can be evaluated. The full operational space is typically evaluated by progressing the data constants and variable coefficients of the model across the full operational space of the application.

3.4 Structural Representation of System Designs

3.4.1 Causality Representation

There are many forms of functional relationship description. As a result of their simplicity of use, the predominant form of representation is the causality network. There are many forms of causality diagram.

The most common in use by systems engineers is the block diagram, reliability engineers use reliability network diagrams, and programme managers use the PERT diagram to name but a few. From a practical point of view it is straightforward to reduce such networks to a common form and analyse their characteristics.

3.4.2 Family Trees

This is the simplest form of decomposition. It shows the relationships as a hierarchy in the sense of progressive containment. Their construction is usually based on the idea of space occupancy. However, it has a very wide range of use to show groups of components in various contexts, including those that illustrate the contents of commercial undertakings.

For many applications, the issue of component interaction is not relevant. However, the lack of component interaction information means that the form of construction should not qualify as an acceptable form of system representation. Nevertheless, its widespread use means that users need to be aware of the implications associated with the simplicity of the representation.

3.4.3 Block Diagrams

The 'Block Diagram' is in common use by systems engineers and in particular those with their roots in control engineering; early references include Systems Engineering by Goode and Machol [13]. It was used for both functional and architectural representations. For example, the advice provided by Goode and Machol is that the Equipment Diagram must complement the Functional Block Diagram so that the allocation of functions to the proposed physical solution is defined unambiguously.

These describe the system as functional blocks that are linked by information flow connections. The relationships are shown by the use of the concept of causation. Normally they appear in the form of input-output relationships. They complement the 'Family Tree' construction in that the functional interconnectivity is stated independently of hierarchical position. From a practical point of view, their pictorial form of presentation makes them easy to assimilate and they are accompanied by various means of algebraic manipulation whereby the decomposition may be varied, while the end-to-end functionality is retained.

Originally, the connections were unidirectional and corresponded to input-output relationships. In more modern versions, and particularly associated with digital systems, these connections may be unidirectional or bidirectional.

3.4.4 Signal flow Diagrams

These diagrams, which were originated by Mason, 1953, [74], came into the inventory in the mid 1950s, and used to complement and enhance the basic Block Diagram technique by describing the signal flows between functions in a structured manner. Formally, they could be represented as graphs and an algebra was developed that enabled full quantitative treatment.

3.4.5 Network Analysis

In addition to the notion of function and signal, equivalence relationships between mechanical, hydraulic and electric systems were established. The study of dynamic systems enabled different forms of relationship diagrams to be developed.

Generic forms of components, e.g. resistance, capacitance, etc. appear in many contexts. In electrical engineering a consistent form of component representation, based on Kirchoff's Laws, enabled network analysis techniques to be developed, enabling, for example, complex filters to be modelled.

Further, diagrams that used energy as the controlling variable provide a direct means of having a unified approach to the analysis of mixed technology solutions. For example, a graphical method based on energy and information flow has been developed. Such models are called Bond Graphs as the technique enables models of individual components to be 'bonded' together to form complex system networks that enable performance and dynamics of, for example, mechatronic systems to be modelled.

3.4.6 Analogue Computing Diagrams

This is a specific form of representation to solve differential equations associated with the computation of dynamic behaviour and performance. These diagrams are in the form of block diagrams in which the function blocks are connected by summing junctions and the function blocks refer to specific mathematical operations, usually addition, subtraction and integration. Originally associated with analogue computing, the range of specified diagrammatic functions has been extended as the capability of both analogue computing and digital computing based simulation languages has expanded.

3.4.7 Reliability Networks

These describe the system in terms of functional dependency, and are used to estimate the reliability and availability of complex systems, see e.g. Billinton and Allan, 1983, [75]. They characterise complex systems failure mode dependency in series and parallel network constructions with uni-directional connections. Such networks have the considerable benefit that they are capable of graphical representation and analytical treatment.

3.4.8 Software Program Flow Diagrams

Many are in the form of process and data flow diagrams. The simplest programme flow chart uses process function blocks with unidirectional connections showing the computational flow combined with True/False decision blocks to control the direction of the computational flow. Various techniques for structured programming were developed.

Yourdan enhanced the methods to address issues of complexity [24], Jackson [62] developed MASCOT to address the use of data pools, and Hatley and Pirbhai proposed the schema for real-time systems that enhances the basic DFD with a control structure; (See 'Strategies for Real-Time System Specification, 1988, [76]).

3.4.9 Behaviour Diagrams

There are many forms or representation that enable the time dimension to be incorporated. Many problems involve systems that depend on descriptions of existential states and their delineation. Some form of finite state machine description is used to describe behaviour or flow and various forms of state transition and message sequence diagrams were created. Typical examples include models of Turing machines, state transition diagrams for logical processes, and computing structure diagrams.

3.4.10 Transportation and Production Flow Process diagrams

There are many similarities with problems that involve the flow of materials or goods. Transportation systems are mainly concerned with routing optimisation associated with the logistical movement of goods and commodities. Production engineers are concerned with the processing of materials, components and assemblies. Just-in-time systems (*kanban*) are concerned with the minimisation of stock holdings.

Production engineers have used matrix methods to represent material flows through machining centre systems. More recently, techniques have been developed to support the use of the Design Structure Matrix for various types of industrial design problem.

3.4.11 Object Oriented Design

Object orientated design is (currently) the most advanced method of system model representation available to system engineers. The Universal Meta Language (UML) provides a syntax for describing both the causal structure of a system and its behaviour. However, it is not prescriptive in the sense that system structures or architectures need to be defined in a particular manner. Syntax checks are limited to input-output consistency. However checks between, for example, partitioning and functionality continue to be illusive.

3.4.12 The N Squared (N^2) Chart

Systems engineers have used the N^2 Chart method, proposed by Lano, 1979, [77], to describe relationships between system components or state transition behaviour. This form of system decomposition enhances the understanding of the relationships between system components. It is used in conjunction with a hierarchical ('Family Tree') decomposition. The components are placed on the diagonal of a square matrix and the interrelationships are described in the off-diagonal elements, whereby the rows represent outputs and the columns inputs.

Various types of construction are shown in Figure 8, where sub-systems are designated A, B, C, D.

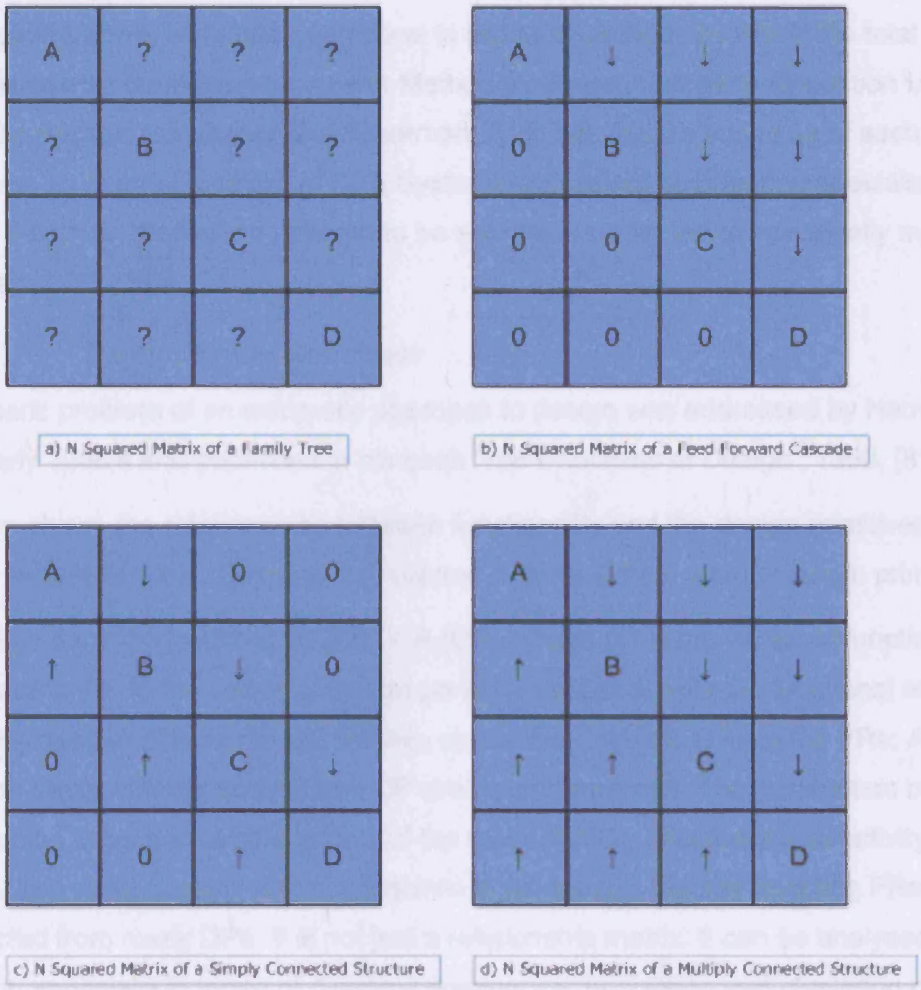


Figure 8 - Various Types of N Squared Matrix System Constructs (D23)

a) Family Tree with no interrelationship data.

b) System with only Feed-forward Causality Structure.

c) System with only adjacent interrelationships.

d) System with multiply-interconnected Sub-systems.

It can be seen that the construction enables all types of system arrangement to be described in formal manner. Further, the construction has the considerable merit that it enables all the relations at any level of the 'Family Tree' decomposition to be shown. This property means that the construction is of special relevance to this thesis.

Various methods of analysis have been described, for example see 'A Practical Introduction to Formal methods' by J C Boarder, 1994, [78].

More recently, some work has been done to produce methods by which the total number of interfaces may be minimised; see 'A Method for System Interface Reduction Using N^2 Charts' by Becker, Ben-Asher and Ackerman, [79]. Yet despite the value of such techniques an internal meeting of BAE Systems plc system engineering specialists showed that their application appears to be sporadic and limited to specifically minded individuals, 2004, [80].

3.4.13 Design Matrix Construct

The generic problem of an axiomatic approach to design was addressed by Nam P Suh in the early 1980's and published in his book 'The Principles of Design', 1990, [81].

This form shows the relationships between functionality and the design primitives actually selected by the designer; e.g. volume is derived from a set of length primitives.

The design equation is stated as {FR} = **A**.{DP}, where FR is the vector of functional requirements, DP is the vector of design parameters that enable the functional attributes to be generated and '**A**' is the relationship matrix that links the DRs to the FRs; **A** is called the Design Matrix and FR and DP are column matrices. The parameters of the **A** are expected to be partial differentials of the type ($\delta F/\delta D$), effectively a sensitivity function. The design matrix **A** is not required to be square, thereby enabling FRs to be constructed from many DPs. It is not just a relationship matrix. It can be analysed in its own right, especially in terms of diagonal dominance, population and clustering of the relationships.

In addition to providing the basic construction of the Design Equation, Suh proposed the fundamental axioms of design. In 'The Principles of Design' it is postulated that there are just two axioms, the Independence Axiom and the Information Axiom.

In summary, Suh et al, stated that design consisted of two axioms and seven corollaries as follows.

Axioms.

- The independence axiom: Optimal design always maintains independence of FRs.
- The information axiom: The best design is a functionally uncoupled design that has minimum information content.

Corollaries.

- Corollary 1: Decouple coupled FRs or provide independent FRs by using separate parts.
- Corollary 2: Minimise the number of FRs and Constraints.
- Corollary 3: Integrate design features into a single part if multiple FRs can be independently satisfied.
- Corollary 4: Use standardised/interchangeable parts.
- Corollary 5: Use symmetrical shapes/arrangements where possible.
- Corollary 6: Specify the largest tolerance in stating FRs.
- Corollary 7: Seek an uncoupled design that requires less information than coupled designs need to satisfy FRs.

Somewhat controversially, they proposed that the optimal system is one that maintains independence of the functions; alternatively, the best design is a functionally uncoupled design that has the minimum information content.

3.4.14 Design Structure Matrix

The production-engineering specialists at MIT created the Design Structure Matrix to represent product production process planning. This enabled cluster analysis to enable the best process schedule to be determined, see Pimmler and Eppinger, 1994, [82a].

Of particular interest to this thesis is the potential utility of the design equation and the design matrix [**A**] with respect the determination of implementation traceability though hierarchical decomposition, see Guenov and Barker, 2005, [83].

4 QUANTITATIVE DETERMINATION OF SYSTEM FUNCTIONALITY

4.1 Relationship to Laws of Physical, Chemical and Informatic Sciences

The SI is made up of three classes of units; base units, derived units and supplementary units. The base units are the metre, the kilogram, the second, the ampere, the Kelvin, the candela and the mole. All Derived units are formed from the base units and the Supplementary units are the radian and steradian, [84].

To enable systems architects to compare and contrast the performance of solution options a common variable and unit of measurement is required. Many scientists have addressed the problem of comparability and energy has become the unit of choice. Even so, there are many different units in which it is measured. However, the Quantity of Heat equation, the Electron Volt (eV), Einstein's equation ($E = mc^2$), Planck's equation ($E = hv$) and Boltzmann's equation ($E=kT$) provide a means of enumerating equivalence.

4.2 Networks, Causality Diagrams, and Matrices

Quantitative systems engineering has many historical associations with the methods developed for dynamical systems. Typical analyses cover the real and complex domains and propositional calculus underpins computer systems analysis.

Causality diagrams provide a means of representation to show the system functions and the flow of information between each function. Diagrammatic representation is both easy and intuitively understandable. Many partially automated design environments make extensive use of tools based on functional causality. A key feature of all forms of instantiation is that they all have a common property; that is they are unidirectional. This means that the associated connectivity matrix forms have upper triangular form. Such matrices have special properties, for example, their eigen values are zero.

Although many block diagram representations incorporate both forward and reverse flow interfaces, analysis methods that support bi-directional structures must incorporate specific means to determine behaviour; e.g. state transition.

For the system architect the uni-directional constraint has profound implications. To describe the principle those familiar with PERT diagrams know that partial completion of an event cannot be addressed by 'feedback' to an earlier event. For example, many activities are part of a process and managers need to view maturity of completion in relation to the performance consequences of many lower level activities.

When a performance deficiency is identified in some activities and corrective action demands that they are repeated, the model should enable recursive action, but PERT cannot model such activities. System Function Flow diagrams also have this feature,

although it is recognised that many workarounds are employed; e.g. state transition or feedback control system analysis.

While the value of function flow diagrams is not in doubt, the architect must appreciate that these represent one viewpoint from the many that the architect must take into account.

To enable the structural description to support recursive relationships, the matrix based N Squared model shows the 'feed-forward' relationships in the upper triangular part of a square matrix and the 'feed-back' relationships in the lower triangular part of the square matrix.

There are many applications that require recursion effects; these include, for example,

- Mechanical structures e.g. leaf springs, trusses, heat engines.
- Chemical Plant components e.g. reflux processes, distillation columns, heat exchange systems.
- Power electrical components, e.g. generators/motors, power regulators, distribution systems.
- Communications networks, e.g. message-response information exchange protocols.

Formally, a network is a directed graph and the duality between graphs and linear algebra has a very important impact on system descriptions. It is a straightforward matter to represent the connectivity of a network in the form of a square matrix, whereby the system components are allocated to the diagonal cells and the off-diagonal cells describe the interface or relationship between the components. This form has the distinct advantage that it enables all the 'building block'-to-'building block' component relationships to be identified, allowing the properties of the network to be evaluated by analysis of its matrix form.

Further, it should be noted that the representation has been used to decompose both functional relationships and state relationships.

4.3 Network Modelling

To enable the systems architect to estimate quantitatively end-to-end performance, system modelling must provide a multi-disciplinary consistent set of component models and a set of algebraic rules for model construction.

Networks are a more generic form of causality constructions. Formally, a network is a directed graph and the relationship between graph theory and linear algebra is both

precise and comprehensive. It is a straightforward matter to represent the connectivity of a network using unidirectional links in the form of a matrix and its properties can be evaluated by analysis of its matrix form.

The designer annotates each connection with the intended functionality, usually in algebraic form. It has the advantage over the Block Diagram form that the internal relationships in a node may be represented. Another advantage is that the pictorial representation is more compact than that of the Block Diagram form. This facilitates the algebraic manipulation of more complex functionality.

Evans and van Dixhoorn summarised the historical development of network modelling techniques in 'Physical Structure in Systems Theory – Network Approaches to Engineering and Economics', edited by J.J van Dixhoorn and F.J. Evans, 1974, [85].

The search for a generic form of cross discipline model can be traced back to Maxwell, when he postulated that force and voltage could be treated as being analogous. The philosophical approach to a unified system was based on two basic types of variable, through and across; see 'A New Analogy between Mechanical and Electrical Systems', F.A. Firestone, Journal of Acoustics Society A, 4, pages 249-267, 1933, [86a]. Through variables can all be measured at a single point e.g. force, current, fluid flow, heat flow, mass flow, whereas across variables must be measured by a difference e.g. velocity, voltage, pressure, temperature and concentration.

4.4 Two-port Analysis

In electrical engineering the problem of Two (2)-Port (four terminal input-output component models) network synthesis was tackled, for example, by Vaulot, 1927, [86a], Strecker and Feldtkeller, 1929, [87], and Pipes, 1940, [88].

Electrical circuit analysis is carried out by the application of Kirchoff's Laws. The methods used include direct use of equations, lumped component and signal flow theory.

Representations of more complex networks need to be constructed by synthesis techniques based on the use of 2-Port representation for each component whereby generic network components can be represented in matrix form.

S R Deards provided the following description of 2-Port Network analysis; see 1966, [89]. In electrical network synthesis a typical 2-Port construction is shown in Figure 9, as follows.

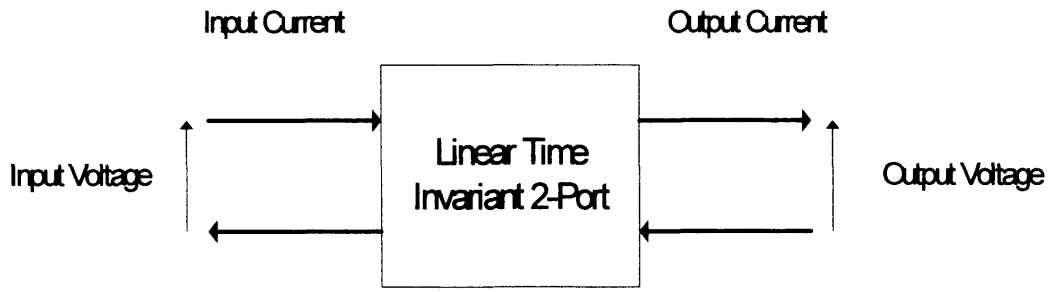


Figure 9 - Schematic of Two-Port Network Component (D6)

Each component is represented by a matrix. For example, an impedance model is of the form:-

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}$$

Where Z_{11} is the import impedance (outport open), Z_{21} is the forward transfer impedance (outport open), Z_{12} is the (minus) reverse transfer impedance (inport open), and Z_{22} is the (minus) outport impedance (inport open).

There are six forms of 2-Port network component. To enable models of complex networks to be synthesized, the impedance form is not the most convenient. Models of networks constructed in the form of series and parallel components can be synthesised by using the A type, the transfer function matrix form, for series components whereby the component matrices are multiplied together (i.e. $A_1 \times A_2 \times \ldots$) and the Y type, the admittance function matrix form, for parallel components whereby the component matrices are summed together (i.e. $Y_1 + Y_2 + \ldots$).

Standard forms have been created for generic components. For example series impedance, shunt admittance, transformer, gyrator, amplifier etc. Importantly the A form of representation is fully defined for generic components. Further, these generic components have consistent analogues across the range of electrical and mechanical components.

4.5 Analysis of Composite Dynamical Systems

The automatic control discipline provided considerable momentum to the creation of models that provide cross discipline consistency. Integrated constructions based on the conservation of work, energy, and power have been developed; see MacFarlane, [45] and Paynter, [46], to provide a unified approach for composite systems involving mechanical, fluid, thermal and electrical components.

Many fundamental physical properties (e.g. force, momentum, charge, current, entropy) are interpreted as spatially intensive variables since their measurement at any given point in space only involves that single point in space. These are often termed 'through variables' (pervariables) as they are propagated through interconnected components.

Also, variables (e.g. displacement, temperature, voltage) are interpreted as spatial extensive variables since their measurement involves two points in space. These are termed 'across variables' (transvariables) as they require two points to effect their measurement.

Each component can be considered as representing a relationship between pairs of measurement points. In all cases one measurement is of a transvariable and the other is of a pervariable.

Models of connected components can be derived from generic components; stores, couplers, converters and dissipators; perstorage elements that store are represented in mechanical engineering and electrical engineering as inertia or inductance respectively, and transtorage elements as springs or capacitance respectively. Dissipators are represented as restrictors and resistors. Converters and couplers are represented as transformers or gyrators.

The following table demonstrates the discipline equivalence for various 2-Port models.

Discipline	Across variable	Through variable	Resistance	Inertial	Capacitance
Generalised component	A	B	Resistance b/a	Inertia, J= $a/(db/dt)$	Capacitance J= $\int(1/a).b.dt$
Mechanical (linear)	Force	Velocity	Friction	Mass	Spring constant
Mechanical (rotational)	Torque	Angular velocity	Friction	Moment of Inertia	Spring constant
Hydraulic	Pressure	Flow	Restriction	Mass	Compressibility
Pneumatic	Pressure	Flow	Restriction	Mass	Compressibility
Thermal	Temperature	Heat flow	Thermal conductivity	N/A	Thermal capacitance
Electrical	Voltage	Current	Impedance	Inductance	Capacitance

Table 8 - Generic Components with Electrical and Mechanical Equivalence.

(See 'Modelling of Dynamical Systems', Vol. 1, ed H. Nicholson, 1980, [73])

4.6 Energy as a Common Unit for Engineering Sciences

Most systems engineers work with technologies that encompass (at least) the following domains:

> Properties of matter, mechanics, gravitation, relativity, hydromechanics, thermodynamics, acoustics, electro-statics, electro-magnetics, optics, quantum mechanics, radiation, radioactivity, information processing.

MacFarlane expressed the view that 'the spectacular success of the concept of energy in unifying the description of physical phenomena has made it one of the most fundamental in scientific work'; see [45]. Nevertheless many systems engineers can (and do) work in environments dominated by the other sciences including chemical, biological, human and economics.

4.7 Multi-port Systems Analysis

Paynter adopted the notion of the energy port previously introduced by Wheeler in 'Measuring the Efficiency of a Superheterodyne Converter by the input impedance circle diagram', H. A. Wheeler and D. Dettinger, Wheeler Monograph No. 9, 1949, [90]. It was postulated that energy transfer was not restricted to two wire circuits and that consistent representation could be established by multi-port reticulation in a given region of space. Components are bonded together to preserve energy flow.

A Bond graph portrays the system in terms of power bonds, connecting the elements of the physical system to power exchange junction structures. It provides a more general capability as, for example, three-port gate valves may be properly modelled; something that is not possible with 2-Port representation, see [47].

Symbolic representation of each bond is provided by a line with a half arrow to show positive power flow and annotated with effort and flow (e.g. pressure and flow rate), corresponding to the ideas of across and through. An example of the representation is shown in Figure 10.

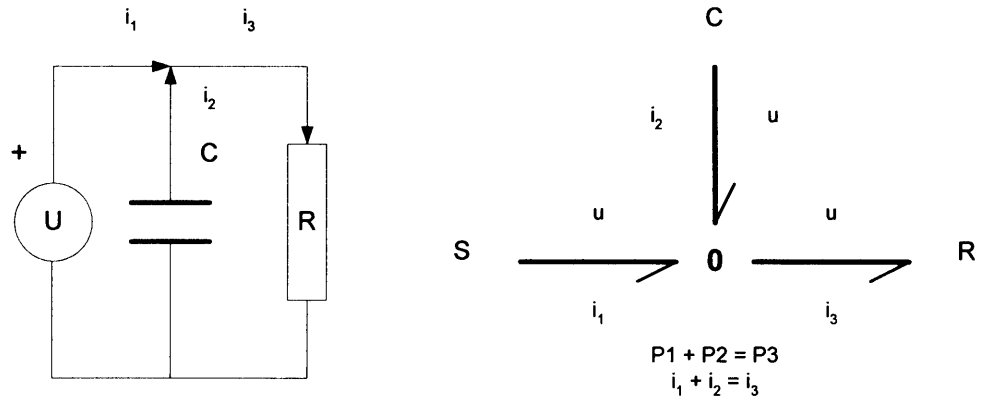


Figure 10 - Schematic of Bond Graph Representation (D7)

In 1968 Karnopp formulated the concept of the multi-port power conservation transform; 'Power Conserving Transformations: Physical Interpretations and applications using Bond Graphs', D. Karnopp, Journal Franklin Institute No. 288, 1969, [91].
Each component is modelled in terms of effort and flow and

Table 9 shows the equivalence of physical system variables for a range of disciplines.

Discipline	Effort	Flow	State Variable
Mechanical	Force	Velocity	Displacement
	Torque	Angular velocity	Angular displacement
Electrical	Voltage	Current	Charge
Magnetic	Magneto motive force	Magnetic flux	Magnetic flux
Hydraulic/ acoustic	Pressure	Volumetric rate of change	Volume
Thermal	Temperature	Entropy change rate	Entropy
Material	Chemical potential	Mole flow rate	Mole number
Chemical	Enthalpy	Mass flow rate	Mole number

Table 9 - Table of Bond Graph Components for various Disciplines [92].
(See 'About Bond Graphs'. www.bondgraphs.com. A. K. Sumantaray, 2001, High Tech Consultants.)

As the functionality of components in many engineering disciplines can be represented by state variable descriptions, the table also provides the normal physical variable associated with each discipline.

All Bond Graphs can be constructed from three generic types of elements; 1-port, 2-port and 3-port. There are four 1-port elements; Resistance, Inertance, Capacitance and Source; all dissipate energy.

There are two 2-Port elements; Transformer (effort to effort) and Gyrator (effort to flow); both conserve energy. There are two 3-Port elements; Junction Type 0 and junction Type 1. Junction Type 0 is common effort and Type 1 is common flow; both conserve energy: (See 'System Dynamics. Modelling and Simulation of Mechatronic Systems', 4th Edition, D.C Karnopp, D.L. Margolis, R.C. Rosenberg, 2006, [93]).

4.8 Bond Graphs with Discipline Capability Extensions

Formal relationships for Bond Graph elements have been established for the:

- Translational and Rotational Mechanical Systems Domain
- Hydraulic and Pneumatic Systems Domain
- Electrical and Magnetic Systems Domain
- Thermodynamic Systems Domain
- Substance and Chemical Systems Domain.

To enable the Bond Graph methodology to be fully inclusive the term pseudo Bond Graph enables its methods to be applied to technology domains that have pseudo energy variables.

Further, for the technique to be fully useful to Systems Design, component relationships need to be established for, at least the:

- *Optical Systems Domain*
- *Radiation Domain*
- *Radioactivity Domain*
- *Informatic Systems Science Domain.*
- *Transportation Logistics Domain*
- *Institutional Activity Domain.*

The author generated a set of additional generic components and state variables to enable the utility of pseudo Bond Graph constructions to be extended to multi-disciplinary systems. As these examples require validation they have been included only for information at Appendix 3.

5 SYSTEM DESIGN

5.1 Introduction to Systems Architecting

A useful starting point is to state in terms that are meaningful to those responsible for systems design what is meant by systems architecting. The generally accepted definitions have already been stated in Section 1.2, System Architecting. They all imply or state that systems engineering and architecting involve components, of whatever type, that cooperate to achieve some purpose.

To ensure that the reader appreciates that this thesis is concerned with the exploitation of science-based technologies, the author's 'best estimate' is to define the role that systems architects fulfil as follows.

> 'System architecting is the application of the science associated with the study of collections of co-operating objects as mechanisms and their instantiation within the science and technology disciplines.'

Systems architects need to have a means of instantiating these 'collections of objects' into structures. The word collection implies some form of boundary within which collections may be confined. Of course interaction with other collections may occur by having relationships between many boundaries. Within each boundary the set of components may be fully or partially connected.

To achieve point performance compliance the author asserts that most system designers regard the system of interest as being entirely quantifiable. Normally, this means that the systems are deterministic constructions. That is not to say that probabilistic or stochastic behaviours are excluded; clearly, components that interact to generate outcomes that are described in statistical terms are within the context of this assessment.

To enable the behaviour and performance of a collection of objects to be determined, systems engineers impose a structure onto the collections pertinent to the application. The best word to describe such structures is 'arrangement'. Such arrangements may be addressed as architectures, however not all architectures imply 'arrangement' as, for example, a state transition diagram or a functional sequence block diagram are often described as 'architectural' diagrams, or more strictly as 'behaviour' diagrams.

The task of the system designer is to ensure that the design provides only those characteristics valued by the customer.

To achieve this objective in an economic manner the designer must strive to establish that the proposed design complies with this objective prior to build. In another way, the

designer must understand the architectural arrangement, its behaviour, its technology instantiation and performance of the proposed solution.

5.2 The Complexity Problem

The fundamental difficulty faced by system architects can readily be appreciated from Reconstructability Analysis. A seminal paper published by Klir in 1986, [94], showed that a system may be simplified by being decomposed into sub-systems. Klir states that "A system with n variables has 2^n sub-systems and there are $2E2^n$ system structures that can be formed from these sub-systems." He goes on to say that that even when redundant structures are excluded the number of potential sub-systems grows rapidly e.g. for n = 6, nearly eight million potential structures can be formed, amply demonstrating the difficulty of the problem faced by system architects.

By way of practical illustration, the size of a system can be characterised in terms of the number of information items held to support its configuration control. A configuration control system is likely to address, at least, the items shown in Table 10, as follows, independently of size or scale of the application.

Needs definition	Build Standard
Science/technology solution basis	Embodiment Standard
Functional solution	Manufacture and supply
Implementation solution	Supportability
Design Standard	Disposability

Table 10 - Typical Information items held in a Configuration Control System

The size of the knowledge base required to support such a configuration control system is primarily related to the number of Requirement Objects that are needed to define the overall system of interest at Acceptance. When applied to a hierarchy of systems the growth profile appears to be logarithmic.

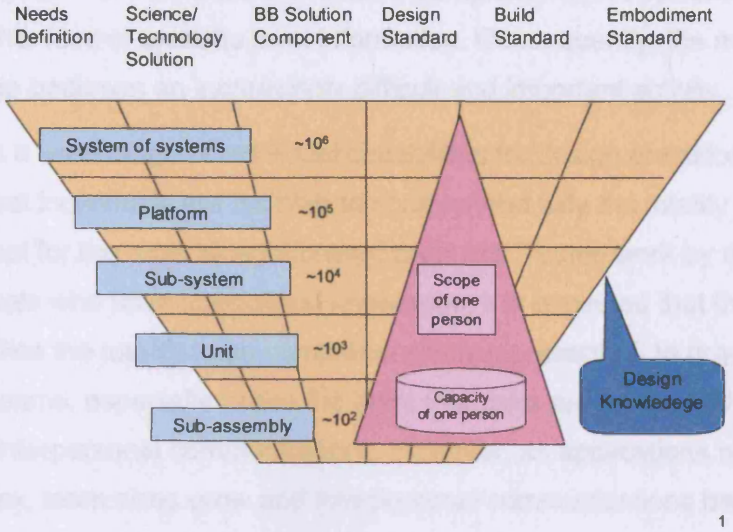


Figure 11 - Complexity illustrated by number of Requirements Objects needed to define a System (P15)

As the complexity increases each domain of configuration control must be individually recorded and managed. As a rough estimate, each level adds an order of magnitude (i.e. decade) of information objects; say from one hundred at the lowest level to say over one million objects for very large systems. This is illustrated in Figure 11.

It shows that at the lowest level of complexity a single person is able to hold all aspects of configuration control in his head (perhaps supported by a simple record of needs, build and deployed standards). As the size/complexity/scope of the system increases, the scope of knowledge-based activities that an individual can address competently diminishes. As the number of contributors increases, the interpersonal activity required for design coordination grows with a consequential reduction in the scope that each person is able to address.

Consequently, system complexity is at the root of all systems problems, and the multiplicity of system engineering process models and analytical techniques are associated with simplification. For example, a practical guide to the implementation of a process model that supports the 'V Model' structure has been provided by Stevens, Brooks et al [71], and Flood and Carson, 1988, [95] have summarised various approaches to complexity management techniques.

A sound simplification strategy should minimise the loss of relevant information. At some stage, each designer has to balance the residual complexity of the system components in relation to further loss of systems level information. Consequently, the management of design knowledge becomes an increasingly difficult and important activity.

Although there is a wide range of individual capabilities for design practitioners, it is very apparent that most individuals are not able to comprehend fully the totality of design knowledge, except for the most straightforward products. Teams work by networking between individuals who have specialised knowledge; it is expected that the team constitution enables the totality to be comprehensively represented. In practice this works for small teams, especially where the team members are co-located and have the benefit of direct interpersonal communications. However, as applications become larger and more complex, team sizes grow and interpersonal communications become diffuse and sporadic. Consequently, design decisions become increasingly based on incomplete or ambiguous information, with an inevitable growth in error rate.

5.3 The Context of System Design as an Activity

In the current business environment, management groups make increasing demands for greater understanding of all stakeholders' needs. The result is that specifications now include more detailed definitions or descriptions of needs in terms of point performance objects. The hypothesis is that the enhanced detail removes ambiguity from the customer interface and will thereby lead to point-by-point compliance; the commercial stance is that if it is not specified, then any such emergent property is acceptable. While the need is to compile unambiguous and reconciled definitions of user needs, the extensive use of vernacular language places heavy demands on those writing such statements in terms of their lexical and grammatical capabilities. The result is that Requirements Management is emerging as a discipline in its own right; however, significant difficulties remain as a consequence of the poor lexical skills of most technical authors combined with a lack of knowledge of most initiating authors as to the availability of cost effective realisable solutions.

Customers rely on suppliers' expertise, as Design Authorities, to ensure that solutions exist that are indeed 'fit for purpose' and have the full weight of law to support such positions. So the system architect has to estimate the emergent properties of the proposed solution; in the end he/she will be judged not only on their ability to produce 'compliance' with the stated requirements, but must also ensure that the solution is indeed 'fit for purpose'.

Each generation of engineers has focused on the problems of the day. Each has made its contribution to capability enhancement, with each enhancement built upon the knowledge contributed by previous generations.

Designers should be able to rely on the wealth of knowledge that exists to provide well-proven means of determination. In summary, today's practitioners have the benefit of what is now a comprehensive and refined knowledge base.

In the late 1980s the lack of a common approach to systems engineering was recognised in the USA by the major aerospace companies. A joint meeting, sponsored by Boeing, in 1990 in Seattle led to the founding of the NCOSE (National Council of Systems Engineers) organisation. The vision was to create a forum to facilitate the determination and communication of best practice and it is organised by territory into Chapters. Practitioners in the UK shared the enthusiasm, and the first international Chapter was founded in the UK in September 1994. Since then NCOSE has become the INCOSE (International Council of Systems Engineers) with a worldwide following.

In addition to the international efforts to define a best practice common approach, many companies developed their own systems engineering guides. These tailored best international practice into the practical context of each company. Despite the proliferation of process models the process of design is not well understood. A common characteristic is that the models refer to or imply a process or indicate responsibility. The role of discipline specialists is recognised. However, the level of recognition rarely goes beyond an organisational chart that acknowledges a relationship between the system engineering community and the communities of discipline specialists. They do not refer to the sciences or technologies that underpin these relationships. This omission means that, in the author's experience, there is a tendency amongst managers and practitioners to focus on the process issues, without proper regard for the enabling sciences and technologies.

These models all assume that the design is created somehow, presumably by someone. For large-scale systems or complex products it is recognised that no individual has the personal mental capacity to 'know' how all the components work together to fulfil the desired characteristics. Team members simply cooperate in a pragmatic way to produce useful entities. In doing so, they develop a collective knowledge of what the system of interest is supposed to do and how it does it. The term 'concurrent engineering' is used to describe a process whereby design decisions are taken collectively by a team of relevant stakeholders and discipline experts. The motivation is to balance both the breadth and depth of knowledge required for competent decision-making.

However, a great deal of it is in the form of tacit knowledge, with the consequence that customer desires for evolutionary enhancement are incompatible with current business preferences for personal contracting employment regimes. The competence of the method relies on the means by which team members identify and communicate with each other. Therefore, the utility of the method is limited to projects that can be tackled by interactive communications between members.

In 1994 the author compiled a technical memorandum [96], as a private submission to GEC-Marconi management, that outlined the problem and provided a potential solution framework. At that time practical considerations meant that these or other potential solution routes could not be refined and, since then, the author's contention is that little work has been done to address the deficiency.

The author is not alone in the assertion that the issue of system design needs to be addressed by the systems engineering community. For example, the authors of 'System Design is an NP-Complete Problem', Chapman, Rozenblit and Bahill, 2001 [97], concluded that "There is a need for theory in the field of System Design". Also Korn observed in 'A Problem of Identity of Systems Engineering', (See INCOSE (UK) Symposium, 1997, pp 73-83, [98]), that:-

> "Currently systems science appears to be directed towards problem solving in organisations but without reference to general problem solving methods. It operates mostly in terms of descriptive, rather vaguely defined theoretical constructs and models which are difficult to relate to observations. As such, it operates at a meta-physical level, fragmented and remote from well established branches of knowledge."

Further, Tony Shell, in 'System Implementation and Behavioural Modelling; A Systems Theoretic Approach' Systems Engineering, Vol. 4, No. 1, 2001, [99], pointed to the existence of a substantial knowledge base of mathematical systems theory that has the potential to support quantitative system design.

Consequently, it would be natural to expect that the analytical techniques that underpin design integrity and robustness would have been developed alongside the growth in process capability. However, the situation is that, in relation to the growth of system size, little work has been done to develop methods and procedures for the analysis of large-scale system arrangements. Therefore, considerable reliance continues to be placed on the personal capabilities of the designer to 'integrate' various representations to produce a holistic view of the system design.

The result is that there is a substantial gap between the process definition and the procedures and methods required to achieve competent definition of the overall design of collections of components that form a system entity. All too often, the phrase 'design' is left to the imagination of the supplier as to what is demanded and the evidence that should be provided to achieve compliance.

5.4 Design Integrity

Systems architects are required to ensure that the: -

1. Solution solves the problem as perceived by the customer.
2. The problem as specified has been solved in the correct way.

Design integrity is a general term used by engineers to describe the collection of information that provides the justification to confirm that the architect has provided the right solution in the right way. It includes evidence gained by inspection, analysis, demonstration and test.

Although the principle of the scientific method is grounded in observation of the facts, many fully engineered applications do not have the luxury of certification through the provision of performance observation. The general public has both an ethical and legal right (c.f. The Health and Safety at Work etc. Act, 1974, and associated legislation) to expect that any item that they use will not have an adverse impact on their lives.

All designers involved in safety related applications e.g. utilities and services, transportation, built environment, pharmaceuticals, medical techniques, places of work, etc. have to address fitness for purpose; for an exposition of product safety matters see Abbott 1987, [100]. Many designs cannot be tested or evaluated as advocated by the scientific method; for example the extreme performance and behaviour attributes of aircraft 'fly-by-wire' systems cannot be evaluated by test, as the risk to people and property is unacceptable. In such circumstances great reliance must be attributed to evidence that is based on analysis.

To enable the systems architect to present such information the design process must support a rigorous process of analysis throughout the design phases. The 'V' Diagram, shown in Figure 12 is an extract from a GEC-Marconi engineering guide, 1996, [101], and depicts the need for a formal relationship between each process component that both verifies and validates the design.

The quality of workmanship for the design verification processes throughout the 'left hand side' of the 'V' Diagram has a substantial impact on the subsequent outcome of the programme.

With the current availability of state-of-the-art technological knowledge and means of calculation, supported by computational power where required, there is little room for excuse that some undesirable emergent property is in some sense unpredictable.

Personal management experience by the author supports the view that all troubleshooting activities during reconstruction (the right hand side of the 'V' Diagram) occur from a deficiency in the scope and quality of analysis that had been carried out during the design phases.

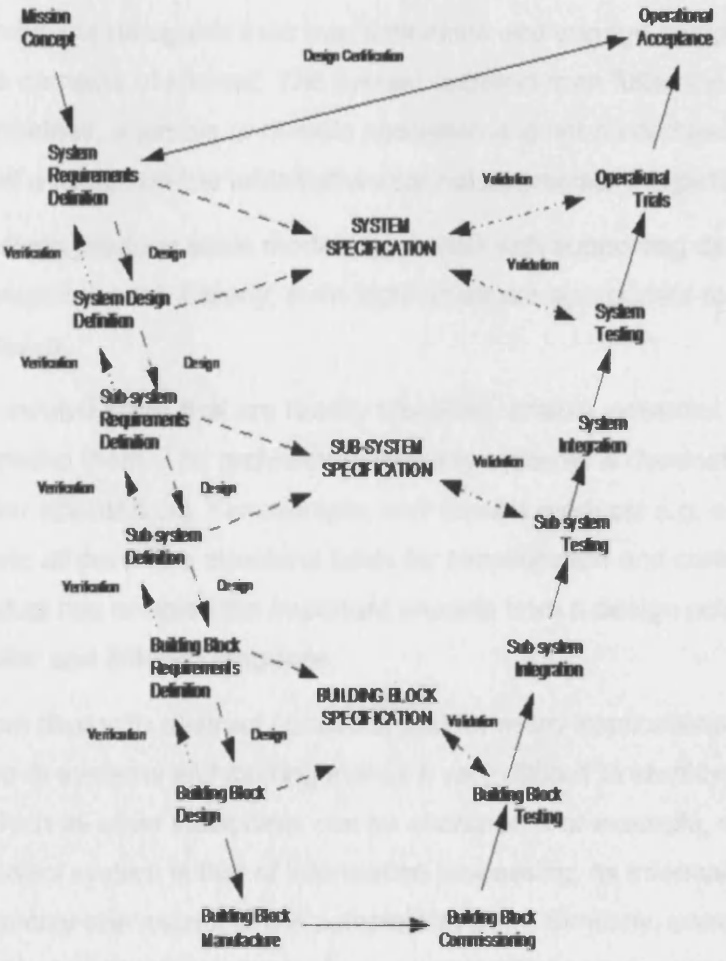


Figure 12 - The 'V' Diagram showing step-wise verification and validation (P22)

Engineers are trained to carry out functional analysis of design attributes. The scope of analytical capability is primarily organised by specialism, initially by type, e.g. civil, mechanical, electrical, electronic, acoustic, optical, informatic, communications, etc.

Each specialism has a full range of analytical techniques to address problems of common interest; many are refined to enable specific in-depth analysis of particular aspects.

The essential question is why does this requirement for rigorous analysis not apply to system architects? Of course, most system architects would refute such a view, although, when pushed, most would state that they have taken a particular viewpoint. Inevitably, that is likely to be strongly correlated with their personal knowledge of particular technology domains. Ignorance is bliss, but this will not do for an architect.

Conscientious architects recognise their own limitations and engage competent specialists for the domains of interest. The system architect then fulfils the role of Chairman. Nevertheless, a jumble of domain specialisms is not an architecture; we may have the pieces of a jigsaw on the table but we cannot appreciate the picture.

Architects of buildings produce scale models combined with supporting data to describe and support a design concept. Clearly, such techniques are appropriate to describe the use of physical space.

Applications that involve items that are readily identified, enable viewpoint structures to be constructed around them. The architecture of many systems is dominated by familiarity of similar applications. For example, well formed products e.g. a ship, train, aircraft, bicycle, etc all provide a structural basis for simplification and consolidation. Each type of product has enabled the important aspects from a design point of view to be described in familiar and efficient language.

Systems engineers deal with abstract concepts, and for many applications, the multi-disciplinary nature of systems architecting makes it very difficult to identify a dominant viewpoint from which all other viewpoints can be anchored. For example, while the role, of an air traffic control system is that of information processing, its informatic structure of information flow is only one aspect of the complete system. Similarly, unmanned aircraft systems need to be described in many ways; e.g. informatic, aerodynamic, communications, command and control, and the operational demands of deployment that include launch and recovery. No one viewpoint can be said to be the architecture. All are required to provide a comprehensive description of such a system.

5.5 Estimation of Integrity during Design Synthesis

An essential part of GST, termed systems methodology, is concerned with problem solving. Since the mid 90s many models have been generated to provide process constructions for particular applications. This can readily be seen by a casual review of the articles published in the Journal of 'Systems Engineering' published jointly by INCOSE and Wiley.

Nevertheless, to the best of the author's knowledge, all 'how to do it' systems engineering models describe data flow 'input-output' processes or activities and there is a continuing lack of procedures and methods to support architectural evaluation.

The developing capability of systems engineering is exemplified by the process structure shown in Figure 13.

The baseline structure is the MIL-STD 499B Model (See Figure 13) that was generated in the operational context that architects were dealing with designs in which there was very close association between the functional definition and the functionality of the implementation components (e.g. analogue computing technology). However, this paradigm was changed fundamentally by the development of computer based systems technology. This technology enables the separation of functional and implementation solutions.

The emphasis of the 90s to establish an internationally acceptable process definition resulted in the compilation of bespoke engineering process definitions that tailored international best practice to models pertinent to the engineering infrastructures of private corporations. The commitment to adopt 'best practice' was supported by the introduction of Capability Maturity Assessment and accreditation.

In common with many corporations, GEC-Marconi sponsored a group of specialist systems engineers, including the author, to compile a best practice 'Guide to Systems Engineering' to tailor best practice system engineering to the context of the GEC-Marconi engineering infrastructure, see [101]. Its authors realised that there was a dearth of information as to specific procedures and methods that could be invoked to support the activities described within the process models. The problem is that somehow the 'Functional Concept' and the 'Implementation Concept' become an 'Architecture', in which their embodiment is assumed to be deliverable by the architect.

The developing capability of systems engineering is exemplified by the process structure shown in Figure 13. The drawing is an extract from [101], and the reader should note that it includes a process component entitled 'Design Synthesis'.

Therefore, the MIL-STD 499B Model was enhanced, by the authors, to show that an activity designated 'Design Synthesis' needed to be included to link the concepts of functional solution and implementation solution to a common architectural definition that enabled the full range of technology specialisms to be integrated into a common solution.

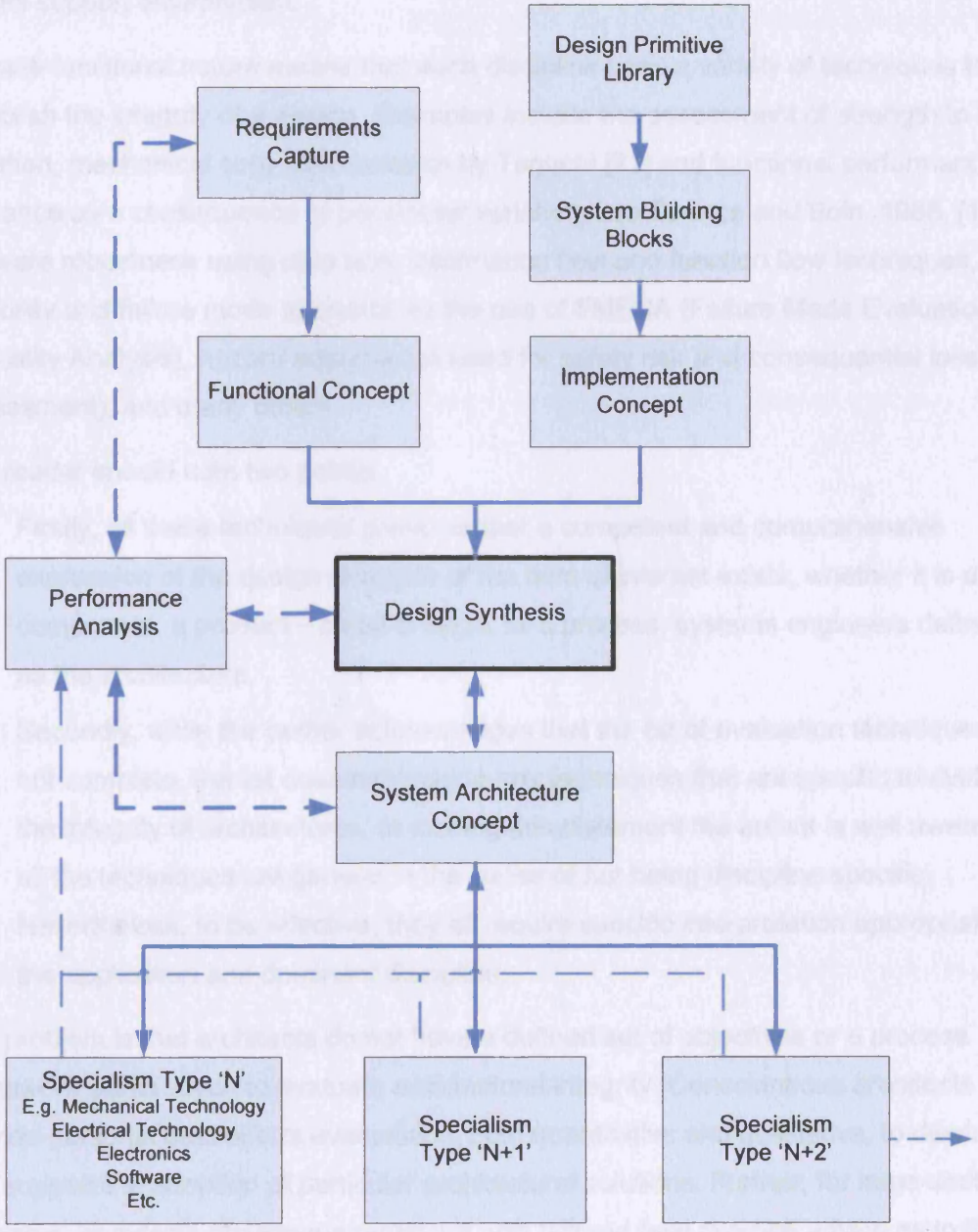


Figure 13 - Schematic of the MIL-STD 499 Process Model enhanced to show relationship with the System Architecture and technology Specialisms (D22)

Note. Because of its pivotal role, many traditionally trained system designers would normally expect the System Architecture Concept to be described within the System Specification or the Systems Engineering Management Plan.

The role of the Design Synthesis activity is to address the integrity of the union of the requirements, the functionality, its implementation and technologies that together provide the system architecture.Integrity is concerned with the ability of a design to work successfully (acceptably) in the context of its implementation, its operational environment and its support environment.

Its multi-functional nature means that each discipline uses a variety of techniques to establish the integrity of a design. Examples include the assessment of strength to load variation, mechanical copy size variation by Taguchi [21] and functional performance tolerance as a consequence of parameter variation, see Spence and Soin, 1988, [102], software robustness using data flow, information flow and function flow techniques, reliability and failure mode tolerance by the use of FMECA (Failure Mode Evaluation and Criticality Analysis), hazard analysis (as used for safety risk and consequential loss assessment), and many others.

The reader should note two points.

1. Firstly, all these techniques presume that a competent and comprehensive expression of the design structure of the item of interest exists, whether it is a component, a product – small or large, or a process; systems engineers define this as the architecture.
2. Secondly, while the author acknowledges that the list of evaluation techniques is not complete, the list does not include any techniques that are specific to evaluate the integrity of architectures. In making this statement the author is well aware that all the techniques are generic in the sense of not being discipline specific. Nevertheless, to be effective, they all require specific interpretation appropriate to the application and dominant discipline.

The problem is that architects do not have a defined set of objectives or a process framework within which to evaluate architectural integrity. Conscientious architects provide personal best efforts evaluations, both quantitative and qualitative, to describe and support the adoption of particular architectural solutions. Further, for large-scale problems, architects provide management with tailored best practice advice as to the scope of determination that should be undertaken for specific applications.

Sadly, too many budget holders constrain the scope and quality of such determinations with the potential result that important emergent properties are not predicted during design formation. This frustrates the presentation of the qualities of the solution, and increases the scale of corrective action activities during the latter phases of integration and evaluation. Usually, the result is loss of customer confidence, schedule delays and increased programme cost.

Consequently, it is the contention of the author that a proper understanding of the 'Design Synthesis' activity has not been developed. Further, there continues to be a dearth of procedures and methods pertinent to structural appraisal of system architectures.

5.6 System Complexity and Simplification Strategies

The complexity of systems has occupied the best exponents of systems theorists, analysts, engineers and managers over many years.

Studies are characterised by views of: -

- System knowledge.
- Complexity theory.
- Simplification strategies.

As a consequence of the all-encompassing view of systems engineering that it includes objective systems with conscious behaviour and unconscious rule-based automata, and organisational systems that include systems management structures and processes, the size of the knowledge domain is dauntingly large. Even when the sub-set of the domain is limited to just that of rule-based objective systems, the size of its knowledge base is substantial.

Systems engineers concerned with 'unconscious' rule-based systems define complexity as the number of objects of information that are required to define the system of interest. This is misleading as the complexity of a rule-based entity is concerned with both the number of objects and their interconnectivity; even more complexity is added when considerations of behaviour and timing are included.

All studies of system knowledge and complexity have concluded that complete determination of system characteristics is not practical except for systems with comparatively few variables.

Bremermann commented in 1962, [103], that problems involving large numbers of possibilities would not be solved by sheer processing capability; we must use our creativity and ingenuity to find tractable solutions to specific problems.

The principal simplification strategies available from GST, described by Klir in 'Facets of System Science', Chapter 9, [12c], employ elimination of variables, aggregation of variables into composite groups and decomposition into lower level (more understandable) components. Clearly, the intention of all reduction approaches is to maintain the required functionality; therefore, these processes must satisfy reconstructability criteria.

Systems architects make extensive use of all simplification strategies. Simplification either requires that the level of reduction produces specifications that are limited to a specific level of definition; alternatively specifications must be reduced to a level at which the uncertainty in the outcome is acceptable.

The additional variables reduce uncertainty; therefore, having the minimum number of variables required to achieve the required level of uncertainty optimises the system definition.

One instantiation of this process is to decompose the system into interconnected objects. The complexity is then determined by the number of variables consolidated into each object and the interconnectivity between objects for each variable. The system engineering community describe this as partitioning and decomposition; it is the primary means of simplification.

System designers use the discipline of partitioning to enable the requirements to be provided from a structured set of cooperating objects or 'building blocks'. The choice of the set of building blocks and their structural relationships is critical to the achievement of the required emergent properties without introducing a whole set of undesirable (or unacceptable) characteristics.

Partitioning is the principal contribution of the system architect; if it is done well the system is likely to meets its performance objectives, it is likely not to have any nasty or unacceptable characteristics, and the programme will be easy to manage. Poor partitioning leads to increased management difficulty (and cost), and an increase in the generation of unwanted or unacceptable behaviour characteristics.

Partitioning creates relationships between the set of component objects that form the overall system. As the number of objects is increased to 'simplify' the role of each design object the number of internal variables required to provide the required functionality also increases.

The task then faced by the architect is to understand the functionality of groups of objects. The emergent functionality then needs to be compared with the desired functionality, then any unwanted emergent properties need to be eliminated/mitigated.

It is surprising that, given the importance of the complexity issue, there is a dearth of work and results available to the practitioner as to systematic methods that can be applied to design problems to provide minimum complexity. Also the lack of openness by 'architects' to justify their design proposition hinders the compilation of design basis data, which would be useful to both reviewers and educators.

5.7 Hierarchical Decomposition and System Reconstruction

While hierarchical decomposition is a 'common sense' concept that is superficially easy to apply, the decomposition of a non-trivial structure requires rules of decomposition to be established.

Robert Rosen in 'Complexity as a system property', (Int. Journal of General Systems, Vol. 3, pp 227-232, 1977; [104]), identified two features that must be exhibited by a successful analysis of a complex system into sub-systems: -

1) The sub-systems must be simpler than the system from which they were extracted.
2) The sub-systems must allow the properties of interest in the original system to be reconstructed from the properties of the sub-systems.

There is a further rule, derived from empirical use by practitioners. The level of decomposition is determined by the ability of the design team to specify the role and functionality of a sub-system component in precise and unambiguous terms. The intention is to remove uncertainty from the definition of the component so that it becomes entirely a deterministic and a rule based entity.

There is a clear implication that the system of interest can be reconstructed from its hierarchical decomposition. The reconstructability aspects of hierarchical analysis was investigated by many including Scholz in 'The Architecture of Hierarchy', (Kybernetics, Vol. 11, pp 175-181, 1982; [105]).

Scholz defined three types of hierarchy; -

- Type 1 for intra-systemic relations.
- Type 2 for inter-systemic relations.
- Type 3 for Functional hierarchy.

These are shown schematically in Figure 14 and Figure 15.

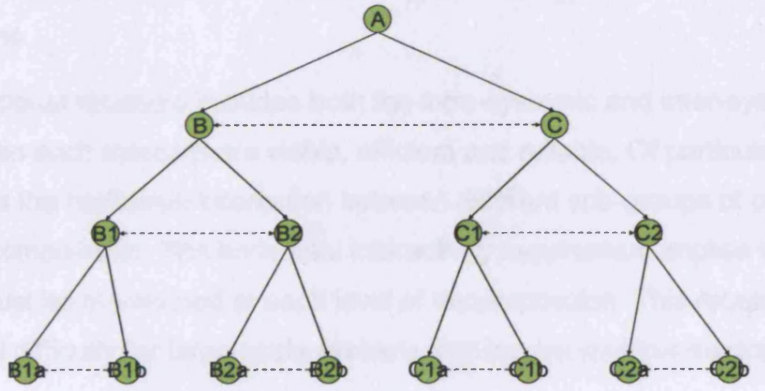


Figure 14 - Hierarchical Decomposition showing aggregation dependency relationships (P17)

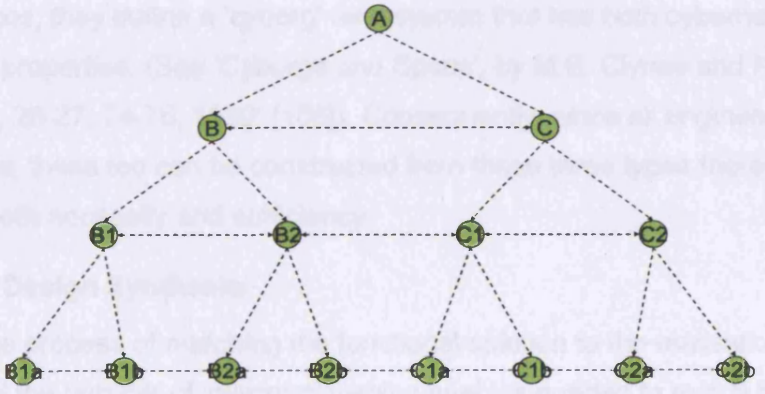


Figure 15 - Hierarchical Decomposition showing interrelationships (P18)

The interpretation of Type 1 is that the vertical lines mean subdivision in the sense that the attribute relationships between lower level components can be aggregated into higher-level components.

The interpretation of Type 2 is that the vertical lines mean interaction between lower-level and higher-level components. The horizontal lines mean interaction between components from the same hierarchical level.

The interpretation of Type 3 is that the functional composition of both horizontal and vertical interactions must be consistent; both Type 1 and Type 2 hierarchies can fulfil Type 3 functions.

When the functional structure includes both the intra-systemic and inter-systemic hierarchies, then such systems are viable, efficient and reliable. Of particular importance, and difficulty, is the horizontal interaction between different sub-groups of partially decomposed components. The horizontal interactivity requirement implies that analytical consistency must be maintained at each level of decomposition. This requirement creates special difficulty for large-scale systems that involve multiple disciplines.

The analysis techniques, described in Sections 8 and 9, that have been incorporated into the Machine System Design Process, correspond to the determination of hierarchical consistency, the determination of causality consistency between components and the determination of inter-nodal functionality respectively. Therefore, machine systems analysed in this way should also be viable, efficient and reliable.

Further, the GST community has shown that all living systems can be constructed from these three types; they define a 'cyborg' as a system that has both cybernetic and organisational properties: (See 'Cyborgs and Space', by M.E. Clynes and N.S. Kline, Astronautics 5, 26-27, 74-76, 1960; [106]). Consequently, since all engineered systems have a purpose, these too can be constructed from these three types thereby fulfilling the conditions of both necessity and sufficiency.

5.8 Design Synthesis

Synthesis is the process of matching the functional solution to the realisation solution. The problem of the number of interrelationship variables needed to match the functional model to the realisation model is illustrated in Figure 16. It shows both an ideal matched implementation structure and one in which the functional allocation is driven by realisation imperatives.

To introduce the reader to the use of matrices to represent system architectural structures, Figure 17 shows the Adjacency Matrix for each system construction.

An Adjacency Matrix is a form of the N^2 Matrix in which the system components are allocated to the diagonal elements and their interrelationships are represented by the {True(1)/False(0)} designation in the off-diagonal elements. The interrelationships are unidirectional; for example the relationship 'A -> B' is shown by '1' in element (1, 2), and the relationship 'C -> B' is shown by '1' in element (3, 2).

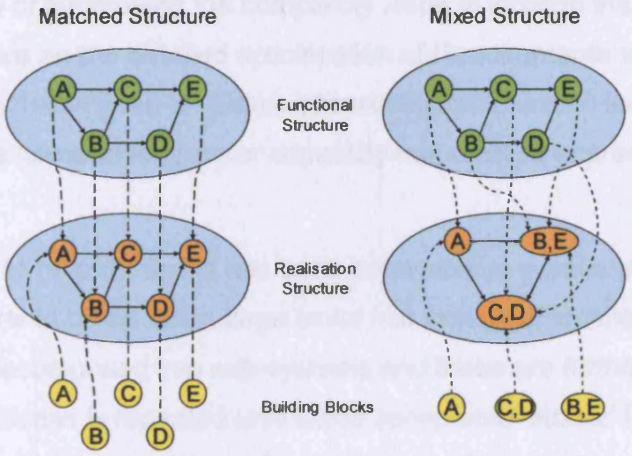


Figure 16 - Schematic showing structural impact of Matching Functional and Implementation Solutions (P16)

$$\begin{bmatrix} A & 1 & 1 & 0 & 0 \\ 0 & B & 1 & 1 & 0 \\ 0 & 0 & C & 0 & 1 \\ 0 & 0 & 0 & D & 1 \\ 0 & 0 & 0 & 0 & E \end{bmatrix} ; \quad \begin{bmatrix} A & 1 & 1 & 1 & 1 \\ 0 & B & 1 & 1 & 1 \\ 0 & 1 & C & 1 & 1 \\ 0 & 1 & 1 & D & 1 \\ 0 & 1 & 1 & 1 & E \end{bmatrix}$$

Figure 17 - Adjacency Matrix Form of System Implementation Construction

The construct enables a complete definition of the system to be described. In the example, the adjacency matrices show that the interface complexity has increased as a result of the use of building blocks with mismatched functionality.

Consequently, the simplification introduced through the use of fewer building blocks has resulted in a substantial increase in interconnectivity and complexity of the realisation structure.

The principal means of addressing the complexity issue is through the expansion of process models, focus on the detailed specification of Requirements and the extensive use of hierarchies to decompose problems into manageable units. However, the apparently insatiable demand for greater capability has created unprecedented levels of complexity.

To enable solutions to be provided, it has been imperative to pursue simplification strategies. It is natural to break down large tasks into groups of smaller tasks. Large-scale systems are decomposed into sub-systems and these are further decomposed to components. The process is repeated until some acceptable 'atomic' level of definition has been achieved, where acceptable means understandable and unambiguous. Such decompositions are carried out as a result of political and business influences that reflect ownership as well as pragmatic considerations based on engineering or technological factors.

Clearly, each technology and science discipline has many methods and techniques that enable functionality, behaviour and performance to be quantified. The performance of each component as a separate entity can be quantified. A system creates additional 'added value' functionality by linking components together. A functional requirement that needs components to interact, means that components must be linked together.

The simplest means to create functional integration is to add links on the principal of the shortest route. This leads to spaghetti like architectures that generate emergent properties that are neither anticipated nor desirable. Strict compliance with the specified needs becomes progressively unobtainable. Multi-layered architectures present great challenges and, to the best of the author's knowledge, there are no techniques that address the quantification of emergent properties of multilayered system constructions.

5.9 Problem Solving Context

The purpose of this study has been to develop methods for estimating emergent properties of system architectures, both desired and unwanted.

Systems engineers who tackle large-scale problems are forced by complexity considerations to use partitioning and decomposition strategies to make the problem understandable and tractable.

Consequently, the internal complexity of the solution expands in relation to both the number of layers used to provide hierarchical decomposition and the number of units that are partitioned at each layer to 'simplify' the specification of each unit. While most practitioners are comfortable with vertical decomposition, the problem of layering consistency potentially means that extra non-intuitive layers need to be included in one sub-group to provide analytical consistency with another sub-group. Further, the multi-functional constitution of a system means that many technology disciplines and therefore functional viewpoints must be evaluated.

With the exception of very simple systems, experience shows that systems architects cannot be presumed to be competent in all the disciplines involved in a system design. The idea of a universal 'all knowing' designer is untenable.

As has been stated previously, the process models require both functional and solution synthesis definitions to be created; however there is no overarching advice as to how this is to be achieved. So, the fundamental question is by what means does the architect understand the qualitative and quantitative construction of the 'machine system'?

The role of the individual system engineer or system engineering group responsible for the architecture of an application is that the individual or group in concert needs to have the ability to blend a variety of disciplines and technologies into a coherent and fully reconciled entity. This means that the conversion of the set of functions that populate the systems function space into functions in real space should be both inclusive of and un-biased in terms of the total science base. Therefore, any definition of a set of system functional structure vectors must be orthogonal to the set of science base disciplines.

The first problem for the systems engineer is to define the set of real spaces that are both necessary and sufficient to enable the design to have steady-state robustness. The second problem is to extend the set of real spaces into their functional context and their state space context to determine both static and dynamic robustness.

Most engineers specialise in one or just a few technologies. Each technology has its own 'short cut' means of analysis for frequently repeated calculations. While professional guidance directs that a systems engineer must be competent in more than one major discipline, it is a daunting task for any individual to achieve substantial multi-disciplinary capability. This is reflected in the uneasy alliance between systems engineering and specialist communities.

Nevertheless, such boundaries have no value when full quantitative analysis is required. Consequently, in order for any supporting analytical techniques to be useful to the architect, such techniques must be inclusively compatible with all major technology disciplines and each design domain must be analytic across the full range of science base disciplines.

It is the role of the unit architect/designer to state the vector space in which the relationships are defined. Usually each designer adopts a narrow field of view with the result that recognition of cross-functional relationships depends on his personal skill and knowledge.

This is very unsatisfactory for complex, interactive designs that are created by mixed ability teams, as the 'worst' designer determines the qualities of the product's emergent properties.

The key feature of system robustness is the integrity of the multi-functional structure. Further, robustness is as much concerned with the completeness of the interfaces as well as functional and tolerance compatibility. A link not identified during the design stage may do immense damage when its effects on integrated system performance become visible. Corrective action is likely to be costly and sometimes only partially practicable.

It is critically important to the quality of analysis to meet tests of completeness, sufficiency and necessity. The variables must match throughout the decomposed and partitioned structure, and the functional relationships must be consistent in terms of the technology/science base that is used for implementation.

5.10 Outline of Proposed Solution Structure

A schema of machine system design is proposed in this thesis that provides a more formal design synthesis process that facilitates evaluation of the architectural structure. In addition, this thesis describes how this schema can be integrated within the 'V' Model product development process model.

To address the issue of the multiplicity of viewpoints, it is proposed to structure the functionality of a system in terms of four generic constituents:

- its general arrangement,
- its ultimate performance,
- its interconnectivity, and
- its interconnected functionality.

Each constituent is referred to as a domain. This proposal does not limit the ultimate number of viewpoints that the architect may need to consider. However, it does provide a structure to control and manage the objective and utility of each viewpoint in a coherent framework.

Descriptions of large systems make use of hierarchical decomposition to atomise functional definition. The impact of such decomposition is to obscure the relationship between a function and the platform used for its delivery. Such obscuration means that sometimes functions will be delivered by multiple platform components and sometimes multiple functions will be delivered by single platform components.

Therefore, the designer needs a method, or methods, that track the relationships between the functional structure and the implementation structure. To address the issue of function to implementation matching, an extension of the 'Design Matrix' is proposed. To address the issue of causality an extension of the 'reliability network' is proposed.

These two methods together provide the designer with information that shows the match between the functionality structure and the implementation structure to establish both hierarchical and causality integrity. The designer will then accept the quality of match or modify it until it is acceptable.

To enable architects to have a structural determination of the design that encompasses the range of disciplines and functionality used by systems architects, it is proposed that system structures be represented by graphs. Specifically it is proposed that the architecture be represented as a functional structure that consists of functional objects and interrelationships. The implication is that the functional structure will take precedence over the implementation structure and, therefore, the implementation structure will be aligned to the functional structure. This proposed construction provides a generic means of architectural structure representation.

To enable the emergent properties to be quantitatively estimated from the system graph, each component of the graph is then provided with a label that is linked to an expression of its functionality. Transposing the labelled graph form into matrix form enables the

functionality structure of all source-to-sink paths to be determined in the form of direct product/sum expressions. Replacement of the function labels in the direct product/sum expressions by their associated functions enables the source-to-sink functions to be determined and quantitatively populated. Evaluation of these functional expressions provides the architect with estimates of the emergent properties.

In summary, it is proposed that the Design Synthesis activity be supported by five methods: -

- A design process.
- A set of domain viewpoints pertinent to the knowledge aggregated for an architectural design solution.
- Means of determination of traceability consistency through hierarchical decomposition.
- Structural representation of system architectures.
- Determination of functional construction and quantification.

6 DESIGN SYNTHESIS AS A STRUCTURED PROCESS

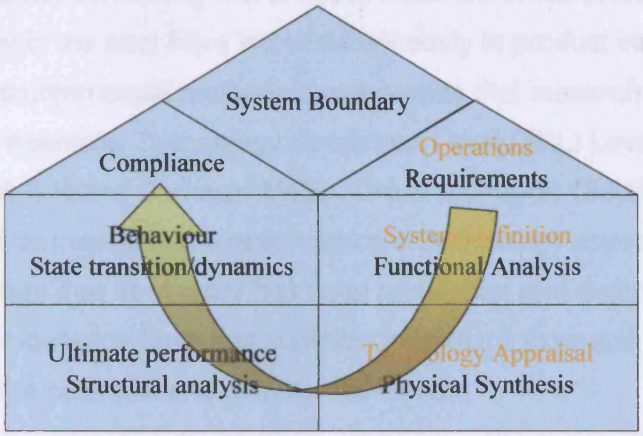
6.1 Process Schema Description

With respect to terminology, to remove the ambiguity invoked by the use of term 'system' the term 'machine system' is used by the author in this thesis to refer to the solution object when seen as a mechanism. Further, the reader should note that it is common practice for practitioners to use the term 'Building Blocks' (BBs) to refer to 'assemblies' of components or modules; the term has a connotation of completeness in the sense that its functionality and interfaces are well defined to facilitate integration with other BBs. Its use enables designers to use the word components in the context of lower level or atomic items that constitute 'assemblies'. Building block components may include human engineering roles provided the human centred functionality can be expressed in rational terms.

All 'hard' systems solutions are machine system concepts that are constructed from the technology components that the architect has at his disposal. That means that the architect has a duty of care to choose building blocks that have the attributes that are appropriate to all the stakeholders' skills, facilities and resources.

To address the requirement to instantiate a process component for structural analysis within the normal systems engineering process model, the author proposes a schema for Machine System Design. It is designed to address the need to overtly incorporate architectural structure at the heart of design. It is based on the MIL 499B model and is compatible with the Architecture Framework Model proposed by the DOD.

The author has devised a pictorial means of representing the schema, as shown in Figure 18. The author has named the concept 'The House of System Design ' as the schematic model looks like an outline of a simple two-up, two-down dwelling.



Note. Architecture Framework Definitions

1

Figure 18 - Schematic of 'House of System Design' (P3)

(Note. Figure 18 is presented with clockwise interpretation to align with the Spiral Model of sytem design).

System decomposition and reconstruction are usually portrayed on the vertical axis, so the north-pointing apex of the arrow shape represents the fully constructed system, and progressive decomposition expands north to south. The roof space consists of the boundary that delineates the system of interest. The outline in the form of an upper floor provides the 'rooms' for the Requirements definition and its complement that supports the Acceptance Criteria and the Compliance Matrix. The middle floor provides the 'rooms' for Functional Decomposition and its complement Behavioural and State transition/dynamics Performance Analysis. The lower floor provides the rooms for System Synthesis and its complement that is entitled 'Structural Analysis and Ultimate Performance Analysis'. The arrow depicts the process sequence of the model. The left hand side is concerned with design decomposition. It starts at Requirements, and is followed by Functional decomposition and Physical Synthesis. The right hand side is concerned with reconstruction and validation. It starts with Structural analysis, and the determination of Ultimate performance, it is followed by the determination of behaviour and then dynamic performance. These processes lead to and enable compliance assessment in the context of the system boundary.

6.2 Design Process Integration

The design activity is not something that exists in isolation. While some 'blue sky' research activity may in the past have led simultaneously to product instantiation the current business and commercial methodology assumes that research continues until a maturity level of, for example, Technology Readiness Level (TRL) Level 4: (See Document DoD 5000.2-R and Carnegie Mellon University report CMU/SEI-2002-SR-027, [107]) is achieved. This means that a new science or technology aspect has been developed to the extent that its viability has been prototyped and demonstrated. At that stage, the relevant knowledge base has sufficient substance to enable the exploitation risks to be profiled in a business and commercial context.

Mainstream design (i.e. TRL level 5 and subsequent levels) is then a series of activities that fit the knowledge base to the needs of the application in the context of an exploitation process; such processes are the means by which all those involved in the enterprise cooperate to achieve a common goal.

This integration of scientific and technological knowledge, the context of the design application and a process infrastructure to identify a viable solution space is shown schematically in Figure 19 as the intersection of Knowledge, Design and Process.

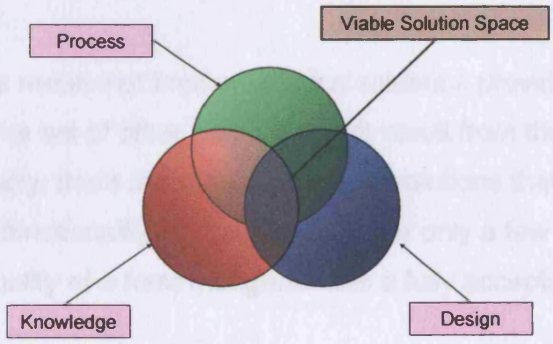


Figure 19 - Integration of system knowledge with design and process to achieve a viable solution space (P4)

Wayne Wymore, in his book 'Model Based Systems Engineering', (See Para graph 1.13, CRC Press LLC, 1993, [108]) described the search for solution space viability in the form of many cotyledons (i.e. a single leaf in a seed bearing plant) that form a solution structure.

The idea is that the design space of a particular aspect of the system design is represented by one leaf and that the ultimate design would eventually flower from the intimate overall structure formed by all these leaves together.

Independently of the reader's empathy with this description, all system architects know that there are many aspects that need to be considered to enable a fully reconciled solution space to be identified.

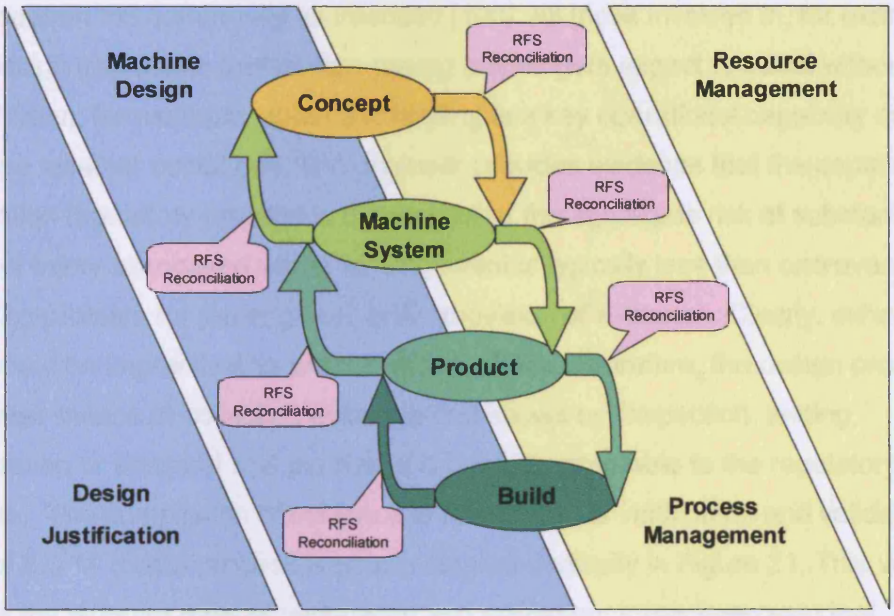
A dictionary definition [9] of a machine is that it is "an assembly of interconnected components arranged to transmit or modify a force in order to perform a useful function". Similarly, a mechanism is "a system or structure of moving parts that performs some (machine) function".

The design process links the user to the proposed machine solution. Although there may be a large number of individual requirements, usually, a few have a dominant influence on the choice of architecture and sub-system definition.

For example, a typical short list of top-level requirements for a weapon system associated with mission profile, deployment logistics, number of operator roles, number of workstations, sensors, fault tolerance, safety etc. set the basic arrangement in place. Then detailed elicitation and analysis of requirements leads into functional definition and allocation to the major sub-systems. The synthesis task is to generate a verified physical design that will fulfil the requirements. Ideally, this means that it provides only the demanded functionality.

Practical considerations mean that implementation solutions provide the demanded properties together with a set of other properties that result from the physical means of implementation. Invariably, there are many candidate solutions that have the potential to provide the demanded functionality. However, there are only a few physical solutions that have additional functionality of a form that generates a fully acceptable set of emergent properties.

The problem of complexity associated with new product design is addressed by instantiating a top-down design process with a process of progressive refinement. This latter process is delineated into a stepwise sequence to coincide with programme review milestones; viz. concept definition, machine system structure, product definition and 'atomic' build definition. Then the new product design process is integrated with the management and engineering infrastructure required to support the programme, as illustrated in generic form in Figure 20.



1

Figure 20 - Generic Structure of Machine System Design Process (P5)

The proposed process is a stepwise embodiment of the core process described in MIL-STD 499B wherein each stage involves a reconciliation of the Requirements (R), the Functional solution (F), and the Realisation solution (S). These core stages also provide the structure that addresses the definition and justification of the machine solution and that of the resources and logistics required to achieve the end goal.

The scientific method is based on the principle of theory validation by observation and testing, see e.g. Davis, 1965, [109]. Engineers employ the principle to ensure that designs support the community as intended [100]. All those involved in, for example, aircraft design appreciate that all fare-paying passengers expect to travel without harm, or fear of harm; for example, automatic landing is a key operational capability especially for adverse weather conditions. The engineer provides evidence that the capability is safe; aviation regulatory practice is to ensure that the aggregate risk of substantial damage or injury associated with a landing event is typically less than one event in one million. The problem for the engineer is the provision of evidence. Clearly, exhaustive testing would be impractical as well as unacceptable. Therefore, the design process incorporates means of collecting evidence that shows by 'inspection, testing, demonstration or analysis' that the risk of hazard is acceptable to the regulatory authorities. The compilation of evidence is referred to as verification and validation. A version of the 'V' model process is shown diagrammatically in Figure 21. This version highlights the different roles of verification and validation. It has been said that verification is to show that 'the defined problem has been solved correctly' and that 'validation shows that the correct problem has been solved'.

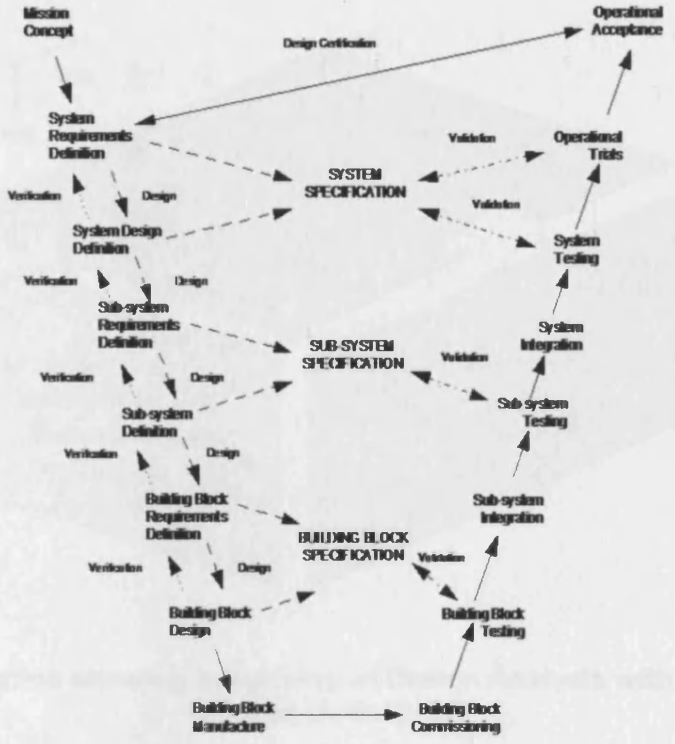


Figure 21 - The 'V' Diagram showing verification and validation (P22) (Courtesy BAE Systems; [101])

A fully verified solution is a physical solution that is robust and produces a fully acceptable set of emergent properties. The emergent properties are predicted from knowledge of the technology, the design of the candidate solutions, and tolerance to both normal and extreme values of the system stimuli and operating environment.

The ability to make cross discipline comparisons in quantitative terms is a fundamental pre-requisite for the systems engineer. It does not matter if it is part of a new system design, or a change to an existing design, or validation for performance declaration. The issue that has to be addressed is that of compatibility.

Therefore, the design synthesis activity is required to include those tasks that are necessary to predict the emergent properties of candidate machine solutions and assess them for acceptability in relation to the demanded functionality. Clearly, the better the knowledge of the technology, the design and the environment, the better is the integrity of the verification evidence. Here, integrity means that the scope of the evidence provided meets tests of coverage, breadth and depth, comprehensiveness of hazard review, and analytical competence.

The role of design analysis is to estimate the emergent properties of candidate solutions. To enable these estimates to be produced the analysis role must be integrated with the process model. This is illustrated in Figure 22.

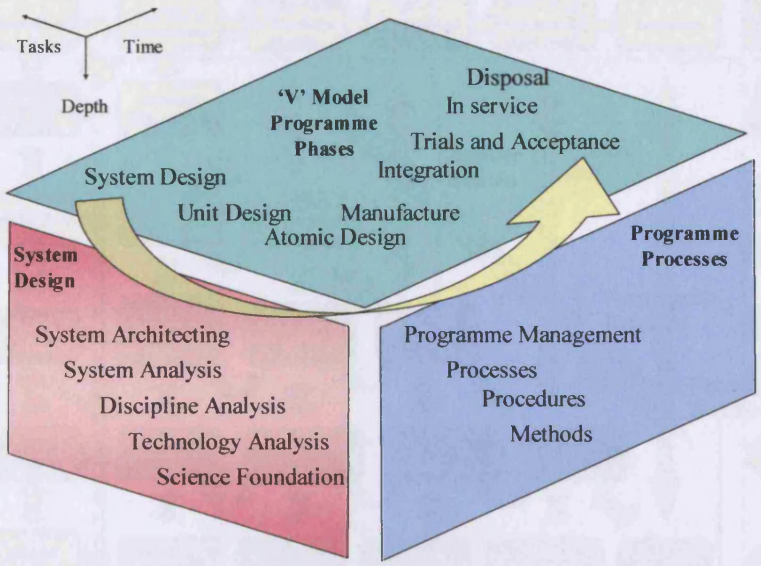


Figure 22 - Illustration showing Integration of Design Analysis with the 'V' Diagram Process Model (P21)

Figure 23 shows how engineers cooperate to design complex products by the application of progressive refinement. The design process ensures that the design definition is progressively refined through its structural, behavioural and quality (ility) attributes.

Functionality definition is accompanied by partitioning, thereby creating the structural arrangement. Since, in practice, the physical embodiment of the realised end product is substantially defined by the nature of the application, partitioning is usually a key input to functionality definition.

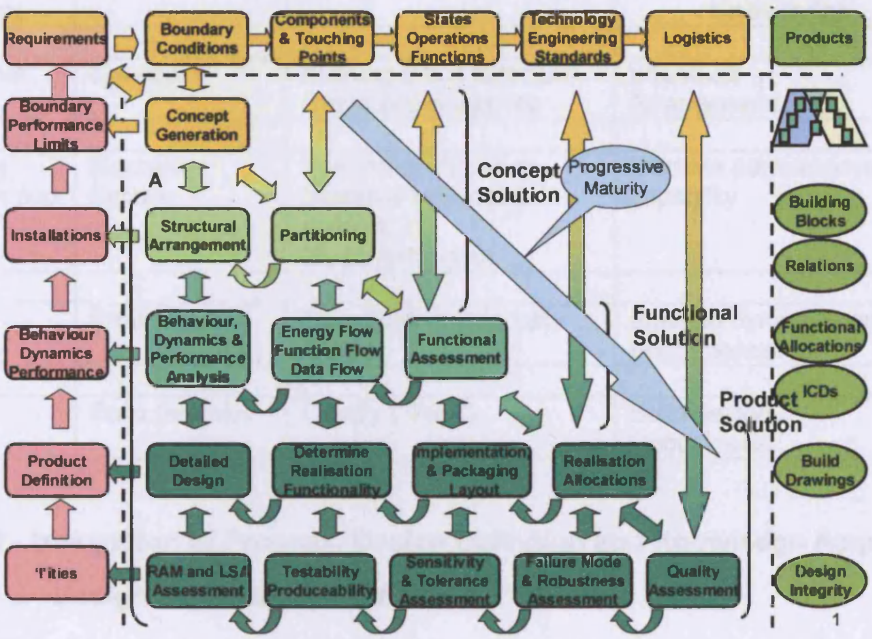


Figure 23 - Integration of Machine System Design Process with Generic Process Model (P7)

Table 11 summarises the structure of the design process. It shows how the lifecycle process is applied to generate a fully justified design. This process demands that, as the design is progressively refined, it is supported by a knowledge base that shows how/why design parameters have been chosen/determined.

The left hand column simply summarises the RFS model. The second column shows how progressive maturity is achieved. Most lifecycle models invoke the 'top down' approach from Concept through to the atomic level that is defined in the build drawings. In doing so, the focus of design is progressively refined in terms of its structure, its behavioural characteristics, and then its quality and sensitivity parameters to ensure that all stakeholder needs are complied with.

The centre right column shows how the design processes are supported by analysis. The right hand column shows the 'chilled' knowledge that is used by the design team to support further design activitiese; composition, capacity, messenger, function. These are justified in Section 7.

Generic processes	Design Phase	Analytical Focus	Consolidated Knowledge
Requirements	Concept	Building Block Structure and interconnectivity.	Structural Arrangement.
Functionality identification and allocation	Machine System	Functional allocation. Interface informatic content. Modal behaviour.	Ultimate performance capability.
Realisation Synthesis	Product	Quantified functionality.	Solution dynamics and performance.
	Build (atomic)	Quality ('ilities').	Build design. Robustness.

Table 11 - Integration of Process, Design Definition and Knowledge Acquisition

6.3 Design Process Structure

A prerequisite of any design methodology is that it provides a structure within which the components of analysis can be identified and their roles put in the context of an overall analytical framework. Therefore, the functional complexity of current applications still needs to be flowed back into the provision of a robust and cost effective integration of technology components into a machine solution.

Also, the reader should note that commercial practices, in general, require strict adherence to the principal of First Article Configuration Inspection (FACI) for compliance acceptance. Therefore, the methodology needs to provide a structure that enables the traditional goals of form, fit and function compliance to be determined.

Figure 24 shows how the definition of needs from an operational perspective is instantiated, via the machine system, into system components that are fully qualified for their intended role.

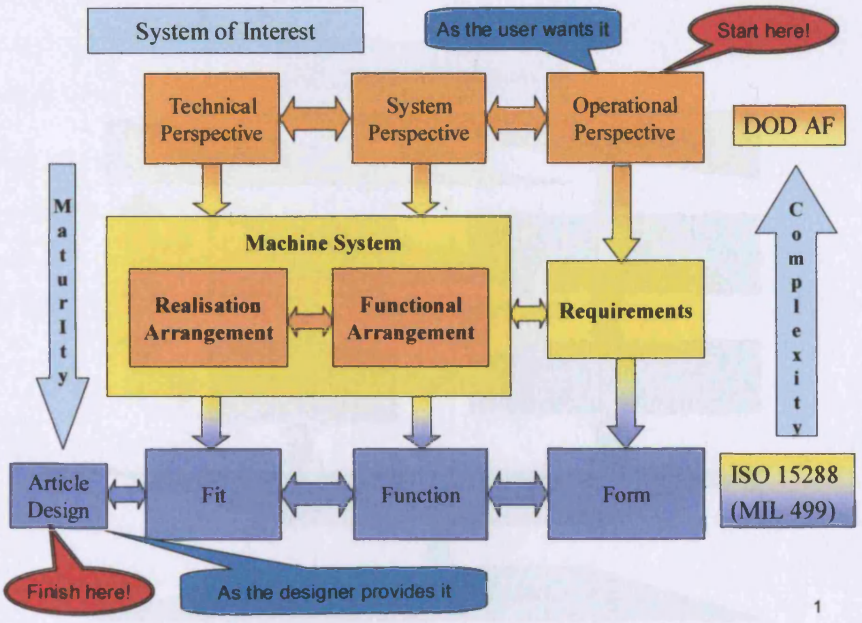


Figure 24 - Process Model Integration (P8)

The machine system definition is, of course, the goal of all the process activity. This goal is met by an object that is an arrangement of components, usually from a library of known assets. It is this object that has to be integrated with the process. Figure 25 is a schematic that depicts the machine concept in relation to the HSD process model, including its supporting asset library.

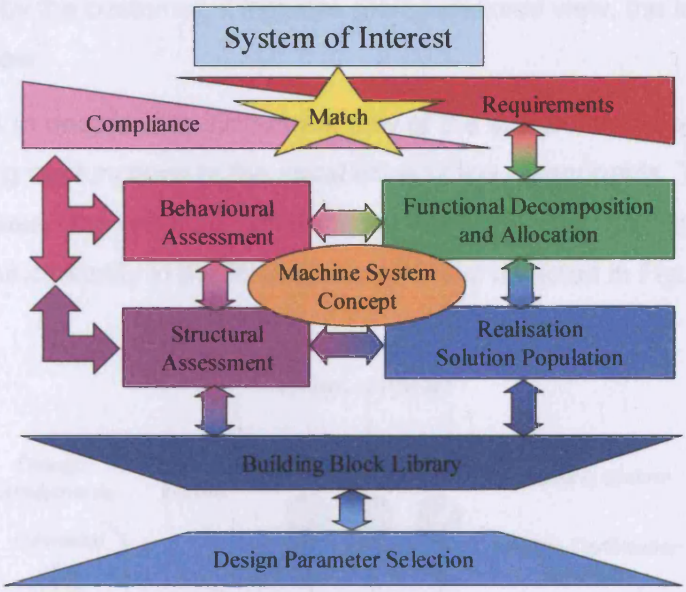


Figure 25 - Integration of the Machine System Object with the HSD Design Process (P9)

The design process for the machine system needs to integrate three perspectives; the operational viewpoint, the machine viewpoint and the components to be employed. Figure 26 provides a pictorial representation of the integration of these viewpoints.

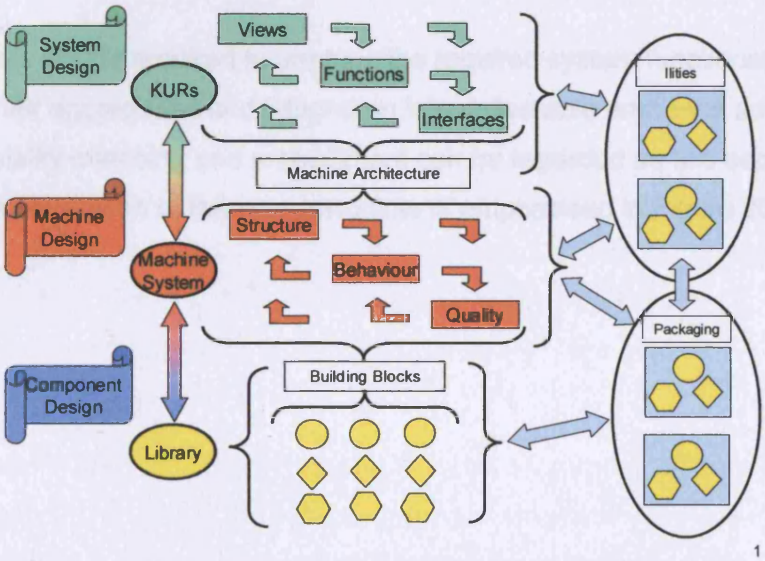


Figure 26 - Integration of User Solution Viewpoint, the Machine Solution and its Components (P10)

Note. This schematic uses the term System Design to describe the viewpoint of the solution as seen by the customer; it includes the operational view, the logistical view and the technology view.

The architect has to decompose the functionality of the system and reconstitute it in real terms by matching the functions to the capabilities of the components. The intimate relationships between the required system functionality and the Building Blocks (BBs) that realise that functionality in the form of a machine is depicted in Figure 27.

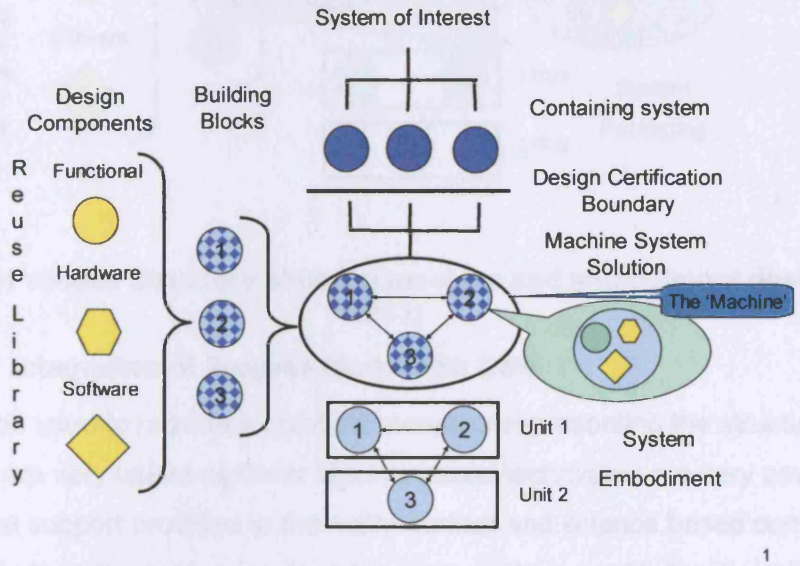


Figure 27 - Machine System Design Process Structure (P11)

When the group of BBs required to produce the required system functionality has been determined, their aggregation and integration into deliverable entities is addressed. In effect, functionality matching and embodiment can be regarded as two separate aspects of design. The separation of the two viewpoints is emphasised in Figure 28.

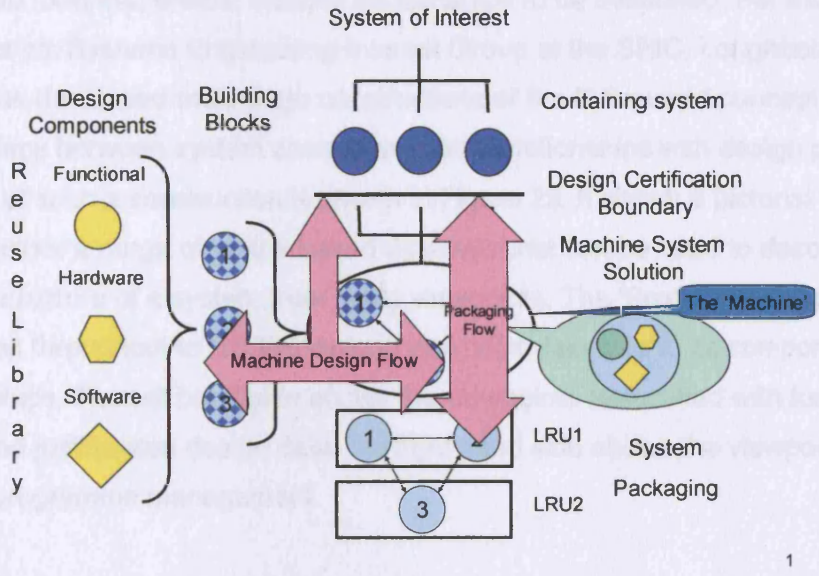


Figure 28 - Process Structure showing machine and embodiment design flows (P12)

6.4 Integration of Process Model with DSM

Matrices can be used to provide a compact means of representing the structure of a system. They are very useful as linear algebra based techniques are very powerful and the educational support provided to the mathematical and science based curricula is widespread. Further, the system engineering community is readily familiar with matrix-based techniques for system decomposition representation and analysis.

R J Lano proposed the use of matrix constructs in the form of 'N^2 Charts' to describe the functional decomposition of system structures [77]. More recently the Design Structure Matrix has been developed, by Pimmler and Eppinger at the Industrial Process Development Group at MIT [82a], or see the Design Structure Matrix Web Site, (www.dsmweb.org/index/) [82b] to provide an insight into the internal structure of systems. They have shown that four different types of data can be represented and analysed using DSM methods: these are component based, team based, activity based and parameter based forms, see 1994, [110]. Therefore, it provides a very useful adjunct to system process models.

While it has been shown that the N Squared Matrix is a compact description of an integrated system, it has limited usefulness, as the sub-system interrelationships are in fact sets of multidimensional relationships. Normally its use is limited to a simplistic pictorial way of describing functional relationships, but the notion of functionality is not described except in the most general terms.

To enable complex systems to be addressed, designers have used various extensions of the N Squared form that enable multiple relationships to be evaluated. For example, the BAE Systems plc Systems Engineering Interest Group at the SEIC, Loughborough University, has developed multi-page constructions of the N Squared concept to address the relationships between system components and relationships with design processes. A schematic of such a construction is shown in Figure 29. It shows a pictorial framework that links together a range of matrix-based data sets that can be used to describe and analyse the structure of a system from many viewpoints. The 'Product Composition Matrix' is used throughout to link the various viewpoint data sets to its components and interrelationships. The left hand side shows the viewpoints associated with functional, realisation and justification design data; the right hand side shows the viewpoints pertinent to programme management.

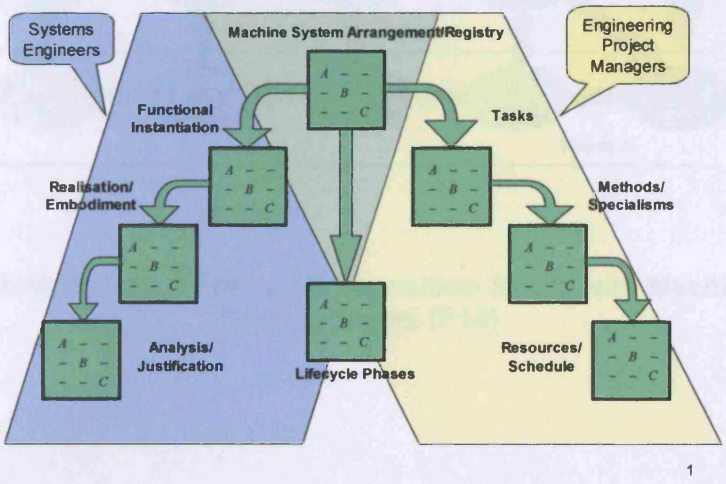


Figure 29 - Use of Matrices in Concert to support a Systems Engineering Process (P13)

The complete range of viewpoints for any application can be organised in terms of a System Framework. While the construction of each Framework is particular to the specific needs of the application, they have a common attribute in that the interfaces between each viewpoint provide a comprehensive channel of information transfer. Therefore, wherever possible, the format and content of such interfaces should be standardised. These constructions provide considerable economic benefits as they enable piecewise development of the range of viewpoints required for the target application.

The knowledge held in the Product Composition Matrix will change and be refined as the design progresses through each stage and each phase of the development lifecycle. The multi-matrix schematic from Figure 29 is used in Figure 30 as an icon to represent the design knowledge data set. Figure 30 shows how the Product Composition Matrix data set can be integrated with each stage of the Machine System Design Process so that the data held is progressively enhanced throughout the design process.

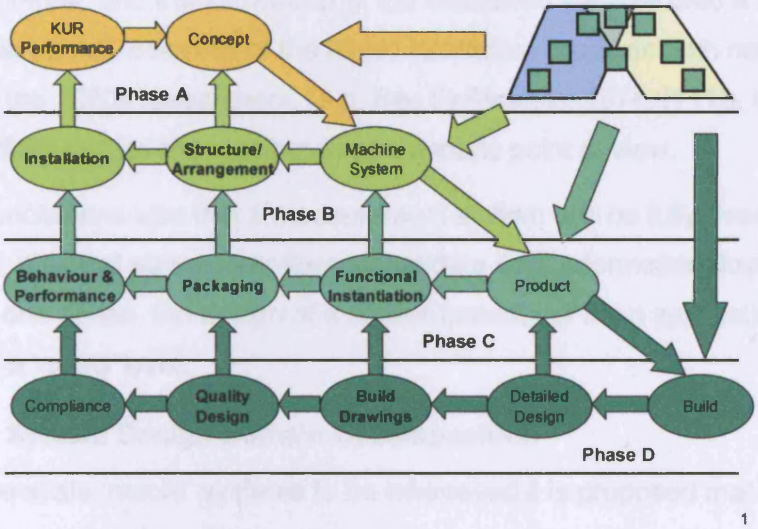


Figure 30 - Integration of Product Composition Matrix with Machine System Process Model (P14)

7 SYSTEM KNOWLEDGE AND ITS DECOMPOSITION

7.1 On Sensor Design and System Design

Sensor technology is an essential part of any application that interfaces with the physical world. Sensor design has become a discipline in its own right because of the complexities involved in creating a device that has characteristics that are sensitive to the parameter of interest, measurement without interference or modification of the parameter of interest, and transformation of the measured variable onto a low cost measured value carrier. Because of the highly interactive cross-domain nature of sensor design, during the 1970s researchers, (e.g. See Finklestein, 1974, [111]), in the measurement field tackled the problem from a generic point of view.

One of their conclusions was that a measurement system can be fully described by five characteristics; physical size, power flow, signal/data flow, information flow and overall knowledge. In one sense, the design of a sensor transducer is an application of system engineering at a 'micro' level.

7.2 System Design Domain Decomposition

To enable large scale 'macro' systems to be addressed it is proposed that these basic ideas be transposed into four System Domains with corresponding attributes of Composition, Capacity, Messenger, Functional Behaviour. These combine to define the overall Knowledge of the system design. To clarify what is meant by the phrase 'System Design Knowledge' its scope is intended to include the information required to analyse its attributes and produce at least one copy.

The hypothesis is illustrated in Figure 31.

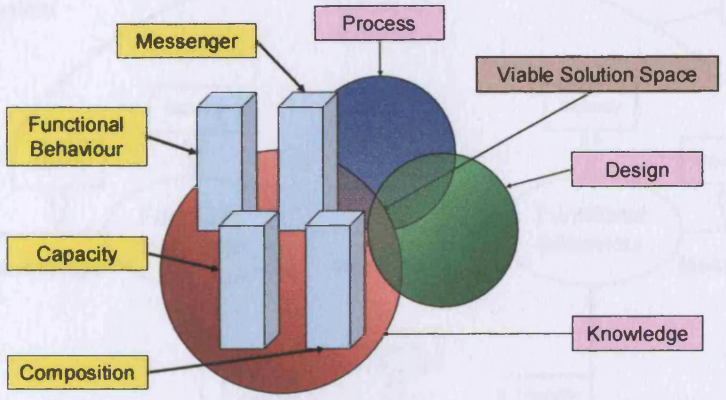


Figure 31 - Schematic of Design Domain Integration (P19)

The composition of system design knowledge is illustrated as follows.

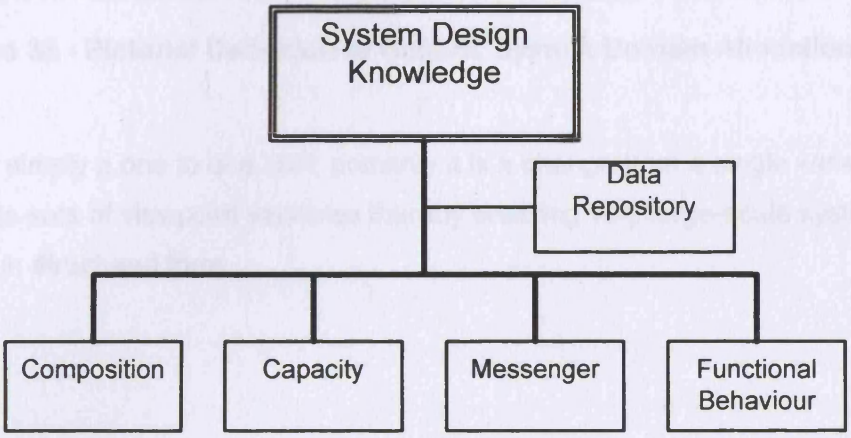


Figure 32 - Composition of System Design Knowledge (D1)

(Note. In order to facilitate analysis the system attributes the Data Repository is shown as a supporting component.)

When applied to a system construction the generic form of domain allocation is shown in Figure 33.

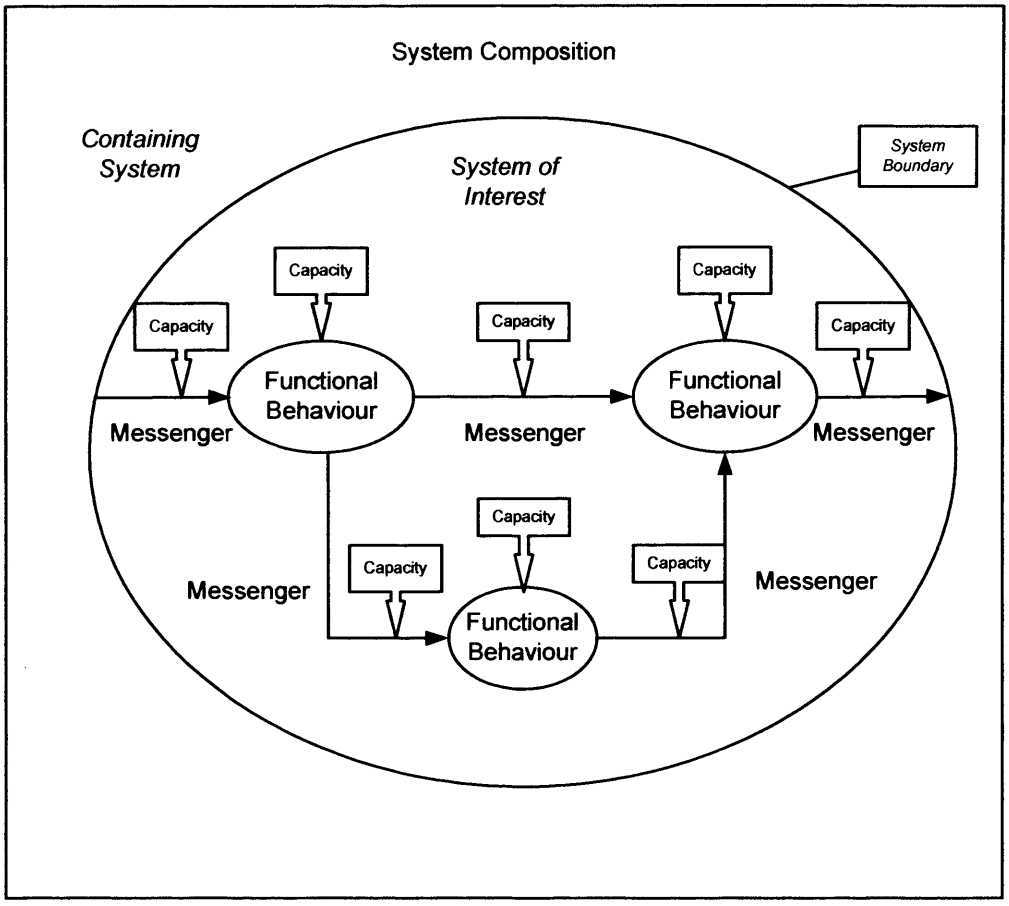


Figure 33 - Pictorial Definition of Generic System Domain Allocations (D4)

This is not simply a one to one shift; primarily it is a change from a single variable viewpoint to sets of viewpoint variables thereby enabling very large-scale systems to be described in structured form.

The hypothesis is then that each domain is analytic across the full range of science base disciplines.

Composition	This domain describes the arrangement, configuration, structure, physical size, and physical and material properties of the system building blocks.
Capacity	This domain describes the quantitative characteristics of the system building blocks in terms of their absolute limits to carry out their roles in the defined arrangement to enable boundary performance limits to be evaluated. It describes, for example, material, or strength, or energy properties, or a system's ability to transport, or the informatic capacity of an IT system, or the activity ability of an institutional organisation.
Messenger	This domain describes the information parameters that are exchanged between the building blocks of the system arrangement; they constitute the links between the components. They are the signals/data elements that carry information between sub-systems and other systems. Examples are force, temperature, pressure, audio intensity, luminance, electronic signal characteristics, voltage, variables etc. In an informatic system they are the message and data components transferred between functional blocks. In an institutional organisation system they are data or resources that are exchanged between activity centres.
Functional Behaviour	This domain describes the functionality and behaviour of the system arrangement in terms of its performance, state space, state transition and dynamical capabilities.
Design Knowledge	This domain incorporates the information concerned with the system's top-level purpose, performance, functionality and behaviour. It includes all the design information required for its justification and realisation; i.e. to make a copy.

A critical characteristic of each system domain is that it is associated with the science base including the set of fundamental constants and derived units; for example co-ordinates to measure occupancy, the capacity to do work, measurable parameters etc.

The intuitive conclusion is that the four domains of Composition, Capacity, Messenger and Functional Behaviour fulfil the requirements of both necessity and sufficiency to enable the Knowledge domain for a system of interest to be fully defined, analysed and justified.

(Note. This study is limited to the physical and chemical science domains as a result of the author's lack of knowledge of organic systems and the biological design domain. Nevertheless, there appears to be no reason why the proposed construction should not be applicable; the main areas of weakness are likely to arise because of the dynamic and inherent hierarchic nature of biological systems. However, the energy and information theoretic base should prevail).

8 TRACEABILITY ANALYSIS

8.1 System Partitioning and Decomposition

System Engineers design systems in terms of output/performance requirements, building blocks and their characteristics, and solution design space. When dealing with a large-scale design problem it is necessary, in general, to have a layered approach to design definition. A four level model in general use, see e.g. DOD 178B [112], is shown in Table 12 as follows.

Level	Description
1	Top level description
2	Functions, components, architecture; machine and state transition structures
3	Component definition; as functions and interfaces
4	Common component definitions

Table 12 - Decomposition Model for Large Scale Systems

The complexity of modern systems (i.e. the brain full problem) forces many systems engineers and most engineers from single disciplines to operate at a single level of design. Most design decisions are forced rapidly down through the user specification and requirements layers to levels, especially 3 and 4, where they know enough about the input, the functionality and the output characteristics to actually carry out the design. It is natural - it is their opportunity to make a personal contribution after all the training and hard experience they have obtained. What is difficult, is to have the perception and sensitivity to appreciate the constraints or performance objectives imposed or implied as a consequence of the decomposition and partitioning activities. In the 'good old analogue days' real estate partitioning and territorial ownership was a guiding design principle for the system engineer. It also had the considerable virtue in that it provided a natural frame of reference for management. Importantly, the functional group boundaries were closely matched to the physical boundaries. (Anecdotal comments show that the requirements specification for a bicycle encompasses some eight thousand objects if the term bicycle is not used; clearly, it is much easier to use the term 'bicycle' than the formal specification, as the arrangement is familiar and the specifications can focus on the overall performance and behaviour of the assembly, and of its constituent parts).

Products from a mainstream civil or mechanical engineering have used and still use physical space as a natural way of partitioning. Similarly, electrical engineering uses connection nodes to control partitioning naturally.

However, digital technology has provided the means for creating functionality that is not associated with its physical form. In effect, the benefit of digital technology has also created the need for a positively structured approach to partitioning.

Also, computer technology has enabled very large systems-of-systems to be constructed wherein their architectures are dominated by functionality and information transfer considerations.

It is easy for the novice to say that we have requirements, followed by functional analysis, followed by a synthesis process that turns functions into physical entities; see MIL STD 499B. However, multi-layered decomposition is required for complex systems.

As stated in the 'NASA Systems Engineering Handbook' [18], good economic practice requires that architectural components are aligned with sub-contract components. Nevertheless, many business and partnership relationships cause this good practice rule to be violated. One cause of such practice is that at the business proposal stage of a product lifecycle much of the information regarding requirements and functionality is simply not available. Nevertheless, good systems engineers are able to invent product/system architectures that are definable in implementation terms to a level that supports the proper compilation of cost data for competitive bidding combined with profitable business outcome. Such success is achieved not just from the notion of similarity with existing products with minor changes; radical and completely novel designs are created.

The proposed graphical construction is based on a two dimensional view of the architecture. Therefore, to enable the proposed graphical construction to be applied correctly, the hierarchical decomposition matrix needs to be evaluated. Firstly, the integrity of the decomposition needs to be analysed to ensure that the architecture construct at each layer of decomposition is compatible with the architecture construct at its next highest level of decomposition. This simple rule enables the integrity of the overall top to bottom architecture to be maintained.

Two methods of evaluation are proposed: The first checks the integrity of hierarchical decomposition and the second checks that the realisation construction matches the functional construction; (c.f. Scholtz criteria).

8.2 Evaluation of Multilayered Structures

8.2.1 Generation of Dependency Matrix Equation

As stated in Section 6.1, to enable systems engineers to design complex systems they work with the concept of 'Building Blocks'. Building blocks are conceptual, they are not necessarily just physical entities. They have real meaning in terms of the system architecture and they are precisely definable; usually they consist of groups of functions.

So the first step in applying the 'Principles of Design' [81] is to substitute 'Building Block' for 'Design Parameter' giving {FR} = [A]{BB}.

The next step is to provide the bridge from 'Building Blocks' to 'Codified Products'. Each codified product is in effect a sub-assembly of some sort, essential to the implementation solution; it may not be of much use in its own right, nevertheless it is properly defined. It is bounded and will consist of one or more realisable components, sub-assemblies or technology products. Applying the Design Equation yields {BB} = [B]{CP}.

While this gives us a mechanism for linking system performance to building blocks to codified embodiment product in structural terms, we still need to address the relationship between each technology product and the design levers that apply that technology. Performance requirements associated with environments or support services are generally specified in the form of compatibility statements rather than in absolute measures. For the system engineer these translate into design space occupancy concerns. Applying the Design Equation yields {CP} = [C]{EP}.

So now we have the set of three system equations relating system performance to building blocks, its technology products and to its environmental design space.

$$\{FR\} = [A]\{BB\}$$
$$\{FR\} = [A][B]\{CP\}$$
$$\{FR\} = [A][B][C]\{DS\} \qquad \text{Equation 8.1}$$

The general view is that the design matrices A, B, C are populated with sensitivity or partial differential coefficients. However, this is not the only useful information that can be used to populate them. In many circumstances, it is sufficient for the system engineer to appreciate whether a relationship exists or not between one element and another. These matrices can be viewed in the form of connectivity statements by simply populating them with a 1 or a 0 representing a link or no link. Clearly, for analysis they form Boolean Algebra expressions.

For the purposes of representation the Boolean form will be written as A',B',C' giving the connectivity set of equations as follows.

$$\{FR\} = [A']\{BB\}$$
$$\{FR\} = [A'][B']\{CP\}$$
$$\{FR\} = [A'][B'][C']\{DS\} \qquad \text{Equation 8.2}$$

For the system engineer concerned about completeness of design a construct showing the relationships that have been built up through the system architecture as a result of design decisions taken at technology product level is invaluable.

8.2.2 Example of Use of the Construction

To demonstrate the construction, a simplified form of a helicopter based Command and Control system is used. The architecture is reduced to a generic form, whereby data is collected from multiple sensors and is multiplexed via duplex mission processing centres to duplex workstation displays. The multiplex bus provides full electrical isolation between the sensors and the workstations. Clearly, the intention is to ensure full mechanical and electrical isolation between the individual sensors, the mission processors and the display systems. A block diagram of the system arrangement is shown in Figure 34.

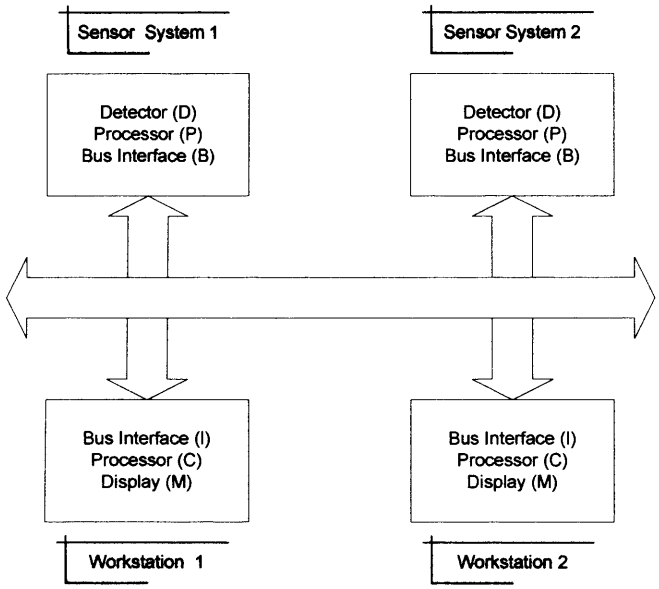


Figure 34 - Schematic of System Architecture (D8)

The dependency construction is shown by Equation 8.3. The functionality provided by each workstation display, F1 and F2, is generated by workstations, W1 and W2, with data input from sensors S1 and S2.

$$\begin{bmatrix} F1 \\ F2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W1 \\ W2 \\ S1 \\ S2 \end{bmatrix}; \qquad \text{Equation 8.3}$$

Figure 35 - Functionality Dependency Matrix Equation for generic Command and Control System

For the purposes of demonstrating the impact of changes to the implementation, the sensors are modified so that they are formed from two units, e.g. a detector followed by a pre-processor, i.e. D1, P1, D2, P2. The revised dependency construction is then as follows.

Firstly,

Figure 36 shows the matrix equation for the dependency construction between the workstations and the sensors.

$$\begin{bmatrix} W1 \\ W2 \\ S1 \\ S2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} W1 \\ W2 \\ P1 \\ P2 \\ D1 \\ D2 \end{bmatrix}; \qquad \text{Equation 8.4.}$$

Figure 36 - Workstation Dependency Matrix

Secondly, substitution of the right hand side of the revised Workstation Dependency equation into Equation 8.5 followed by Boolean multiplication yields the Functional Dependency Matrix Equation as follows.

$$\begin{bmatrix} F1 \\ F2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} W1 \\ W2 \\ P1 \\ P2 \\ D1 \\ D2 \end{bmatrix}; \qquad \text{Equation 8.5}$$

Figure 37 - Design Matrix Equation for Command and Control System with sensor pre-processors

It can be seen, by inspection, that the functionality of each workstation is generated by each associated workstation and by the joint combination of each sensor with its pre-processor.

8.2.3 Conclusion

This shows that a matrix construction that allocates the system's partitioning relationships associated with each layer of decomposition to its matrix elements, can be combined in the form of a matrix equation that enables the overall construction of the relationships to be tracked. When used in the context of a desired solution and an implemented solution the 'difference' matrix shows the changes to the desired solution that have been introduced by the specific construction of the implementation solution.

The 'Principles of Design' enunciated by the MIT Team can therefore be extended into the system design domain so that system building blocks and design space allocations can be defined and set into a formal structure. Thus, the technique may be used to enhance the use of the N Squared form of causality architecture by enabling the relationships between the functional and physical allocations to be explored.

8.3 Evaluation of Functional and Implementation Causality Consistency

8.3.1 Generation of Causality Network Model

A very important aspect of the RFS process model is the relationship between the functional decomposition of the system and the resources that are used to 'deliver' each functional component. The system designer needs to have an intimate understanding of these relationships and the means of determination and analysis. The analysis of complex system arrangements is based on having an understanding of topology and this section addresses some methods that are used to relate functional and physical decompositions.

Traditional system design methods have relied heavily on the use of 'block diagrams' constructed on the basis of causality. Typical constructs show the functionality of the system based (often loosely) on a 'left to right ' view of information flow. Frequently these diagrams are enhanced with multiple tiers of information. It is normal practice to associate an implementation layer with the functionality representation to provide visibility of the physical resource that will 'deliver' that functionality. A typical form of construction is shown in Figure 38.

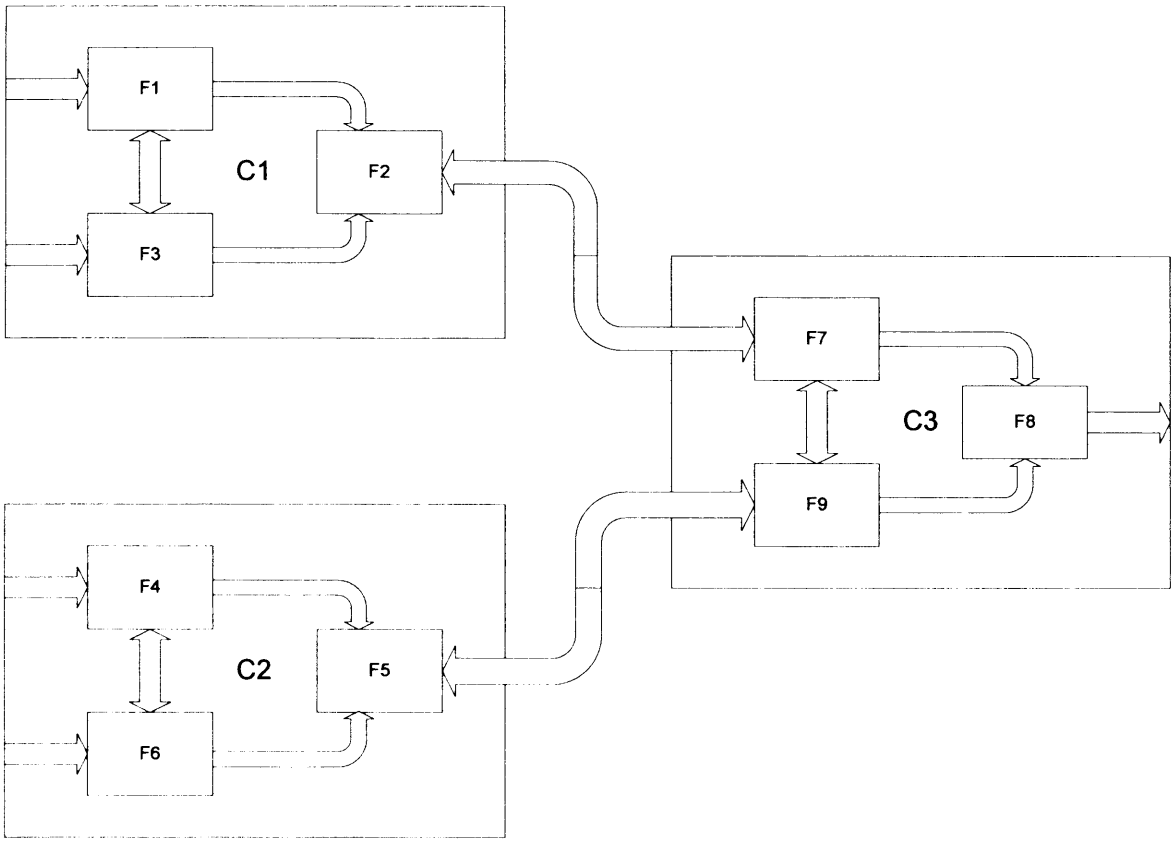


Figure 38 - Generic Causality Block Diagram Construction (D9)

The diagram shows a functional arrangement (F1,...,F9) allocated to a physical arrangement (C1,..,C3).

8.3.2 Description of Analysis Procedure Model

Reliability engineers have made extensive use of causality networks to describe system arrangements so that reliability and availability attributes can be determined. This section describes an extension to this form of analysis to enable complex functionality arrangements to be assessed in terms of the resources that are employed to realize each component of functionality.

The analysis procedure is as follows:

i) Describe the functional causality in the forms of a causality network: (F_m components).

ii) Annotate the network with the resources that are required to support the 'delivery' of each functional component: (D_n components).

iii) Form the Boolean expression from the functional components that provide the 'start' to finish' functional expression: ($F = f(F_m)$).

iv) Replace the functional components (F_m) with the set of resources (D_n) that are used to support the 'delivery' of each function: ($D = f(D_n)$).

v) Reduce and analyse the implementation expression as required to determine the dependency attributes.

8.3.3 Example of Use.

An arbitrary system network is shown in the following diagram.

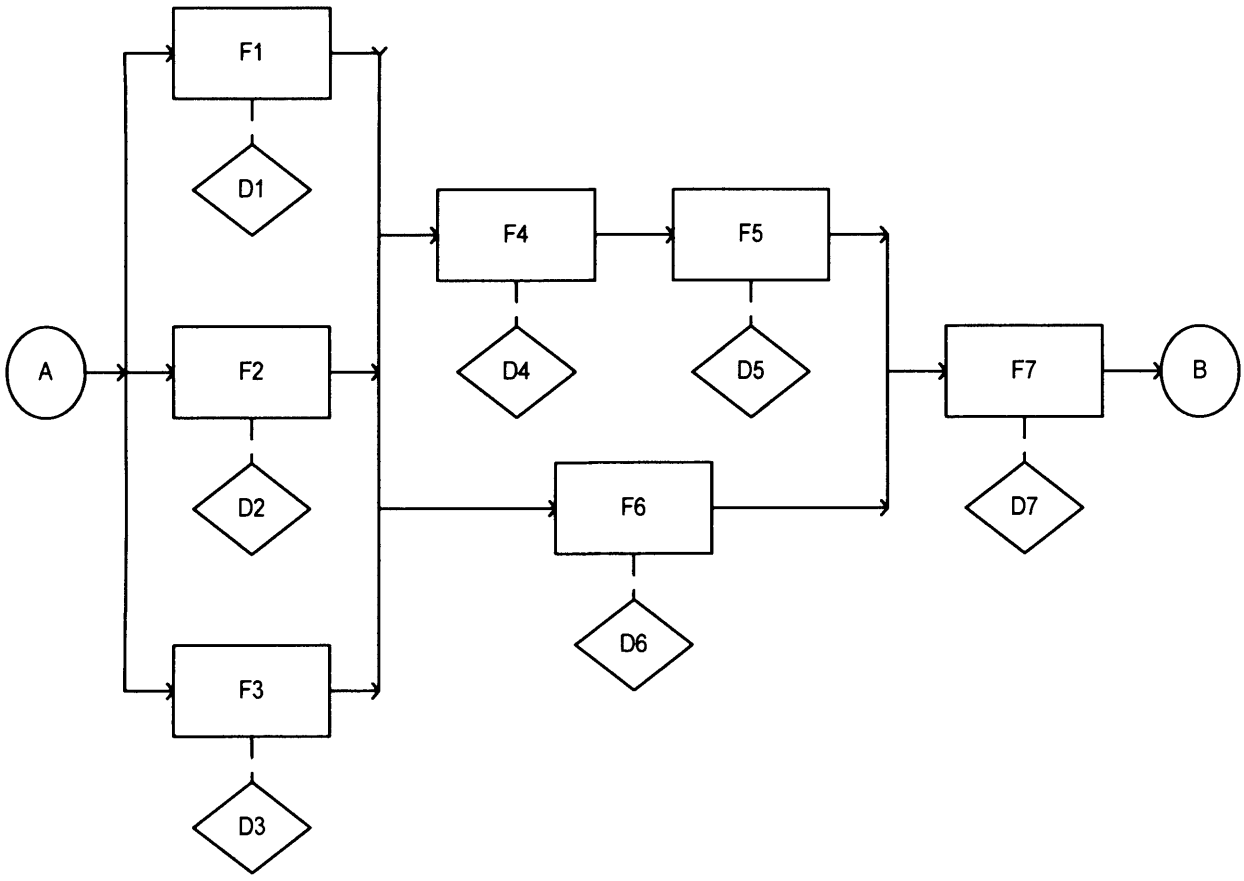


Figure 39 - Exemplar Functional Causality Network (D10)

The system functionality is described by seven components (F1,..., F7) each of which has a set of resources (D1,..., D7) that enable the function to be delivered.

The Functionality expression is:

$$F_{AB} = (F1+F2+F3)(F4.F5+F6)(F7) \qquad \text{Equation 8.6}$$

The resources required to deliver each function are shown in the following table.

Resource	Type
D1	R1
D2	R2
D3	R3
D4	R4
D5	R5
D6	R5
D7	R6

Table 13 - Resource Table 1

The realisation structure expression is determined by substitution of the Resources into their corresponding Functions, and is shown in Equation 8.7, as follows.

$$D_{AB} = (R1+R2+R3)(1+R4)(R5.R6) \qquad \text{Equation 8.7}$$

This simple example shows that the system has serial dependency on R5 as well as R6, a result that is easy to see by inspection.

To demonstrate the impact of a different realisation structure, the example is modified so that the same functional construction is implemented with a more complex arrangement of the resources required to deliver each function. The modified resource allocation is shown in the following table.

Resource	Type
D1	R1
D2	R2
D3	R3
D4	R3 and R4
D5	R3
D6	R3 and R4
D7	R3

Table 14 - Resource Table 2

The realisation structure is then shown as follows.

$$D_{AB} = (R1+R2+R3).(R3.R4.R3+R3.R4)(R3).$$

$$D_{AB} = (R1+R2).[(R3.R4)+(R3.R4)].$$

$$D_{AB} = (1+R1+R2).(R3.R4).$$

Equation 8.8

Clearly, the system decomposes to dependency on R3 and R4.

For complex systems where the dependency is not intuitive, the method enables formal evaluation and verification of the compatibility between the ideal and the implementation causality architectures.

9 ARCHITECTURAL STRUCTURES AND FUNCTIONAL ANALYSIS

9.1 Architectural Structures as Graphs

A Directed Graph (Digraph) consists of nodes and unidirectional interconnections (edges). A digraph is a net with no loops or parallel lines; it is an irreflexive relation. The order of the system is the number of nodes; the size of the system is the number of edges. From a system design perspective the main interest is in the relationships between the nodes of the system.

The duality between graphs and matrices has a very important impact on system descriptions. Formally, a network is a directed graph and it is a straightforward matter to represent the connectivity of a network in the form of a matrix. The properties of the network can be evaluated by analysis of its matrix form.

To enable the reachability within a system to be determined the adjacency matrix must be defined. For any digraph its adjacency matrix is a square matrix in Boolean form with one row and one column for each node. For the adjacency matrix A, and any pair of nodes at [i,i; j,j], the entry is $a_{ij} = 1$ if line v_1v_2 is in D, while $a_{ij} = 0$ if v_1v_2 is not in D.

A key theorem is that, for an adjacency matrix A and the non-negative integer k, the ij-entry a_{ij} of A^k is the number of ij-paths of length k in D. Also, the maximum number of unique paths is determined when the value of k equals the rank of the matrix D; that is the order of the graph. The adjacency matrix provides a means of determining many properties. For example, the 'true' entries may be labelled so that the construction of the path structure between source and sink can be observed. These constructions combine the labels in the form of Cartesian Sum and Cartesian Product expressions.

Graph theory includes the notion of colourisation. This enables each vertex to be annotated with a 'colour'. The graph can then be decomposed in terms of cut sets for each colour. Importantly a system graph with coloured cut sets can be evaluated by 'subtracting' the cut set of each colour to produce a residue graph. The use of different colours for different cut sets with repeated 'subtraction' enables the final residue, if any, to be identified. This has potential application for domain decomposition and viewpoint identification.

Further, the problem of determining the transportation flow or capacity of a network occurs in many contexts. Each edge is assigned with a flow or capacity and the network can be analysed to determine the overall flow or capacity of the network e.g. maximum, integer, multi-node supply and demand matching.

9.2 Functional Decomposition

The proposed labelled path structure method has its roots in abstract algebra that considers relations, functions, partial orders and induction. Graph theory enables the path structure between any source and sink to be determined with each path typically involving many steps. The technique of labelling each path enables the functional components involved in any path to be identified. Path analysis generates the functional combination of labels in any path in the form of a direct product.

As already introduced in Section 3.4.12, systems with bidirectional (feedback) relationships are conveniently described in the form of a matrix, whereby the system components are allocated to the diagonal cells and the off-diagonal cells describe the interface or relationship between the components. To ensure that all relationships are described in unidirectional form, the upper triangular part of the matrix is used to describe the feed-forward relationships and the lower triangular part the feed-back relationships.

To enable the functionality of each path to be uniquely defined the author determined that each path be identified by a label, wherein each label may be linked to a functional definition of the building block components in each sub-system. Then, using the labels as variables, the source-to-sink relationships between functional components can be generated as an expression.

Using arbitrary labels a, b, c, d, these expressions are in the form $((a \circ b) + (c \circ d))$. This means that the overall function f(a, b, c, d) is the intersection of the functions represented by the function containing the function represented by label 'a' in union with the function represented by label 'b', and the function containing the function represented by label 'c' in union with the function represented by label 'd'.

When the rules of union and intersection are not defined, the general form of this expression is known as a Cartesian Product/Sum. However, as pointed out by S G J Taylor [113], when the functions and their rules of binary combination are defined as a consistent coordinate system (c.f. Abelian), the expression is known as a Direct Product/Sum.

Therefore, when these expressions are Abelian, the functionality of the system for each defined coordinate system is shown by the expressions of ordered combinations of labels formed from each element of the A^n matrix.

Then an estimate of system performance between any source and sink node is obtained by substitution of the functions represented by the labels and their rules of binary association into each expression.

9.3 Computer Mechanisation of Analysis Methods

The complexity of designs that are of central interest to this thesis means that computer based computation is required to determine quantitative performance attributes. While Fortran and other similar scientific languages could be programmed to host the analysis, both 'Maple' and 'Mathematica' have the high level functionality required to compute the algebraic expressions, construct the graphs, as well as the normal methods of applied mathematics employed by engineers. The author* has constructed programs in both Maple and Mathematica to demonstrate the core methods of functional analysis. For the purposes of this thesis, the author has adopted Mathematica [114], as the preferred platform. Therefore, the demonstration of the analysis procedure is displayed using Mathematica Notebook extracts.

*Note. To enable the author to generate these programs advice has been obtained from the support help line from Wolfram Research Inc., and Prof. P McIver at Loughborough University. The specific contribution by Prof. McIver was to construct various Mathematica program rules associated with the manipulation of the direct product/sum expressions.

9.4 Method of Functional Quantification

The method consists of four steps; -

- Firstly, the structure of the system architecture is transposed into the form of a labelled adjacency matrix.
- Secondly, the stepwise path structure is determined in the form of a Direct Product/Sum expression.
- Thirdly, the labels are substituted with their functions and the logical rules of association are substituted with real rules of association.
- Fourthly, the functions are populated with the system data and evaluated.

The method is demonstrated using a system architecture of arbitrary structure.

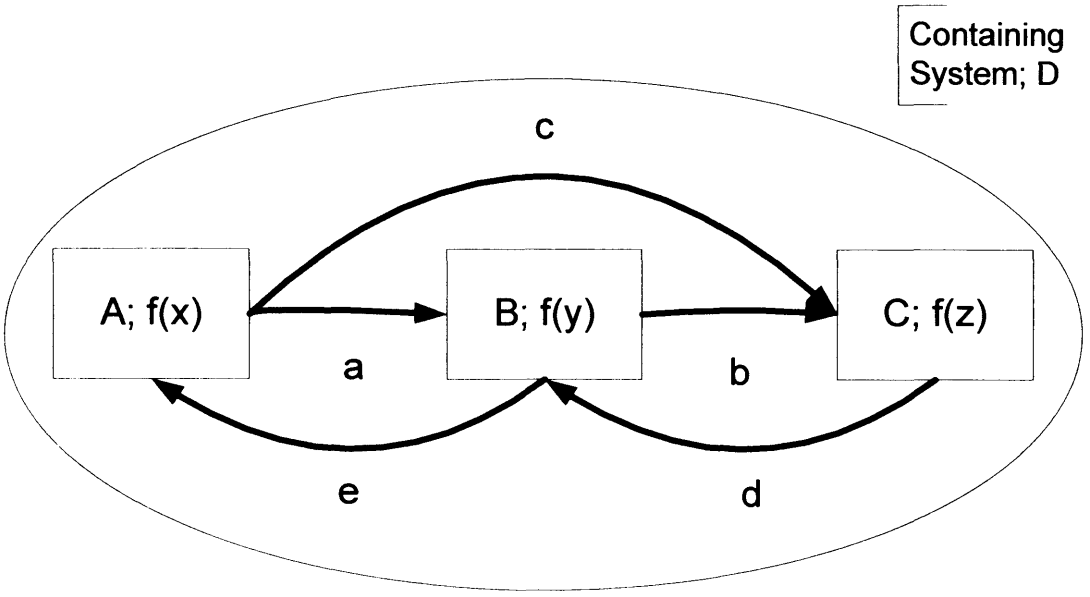


Figure 40 - Schematic Block Diagram of Exemplar System (D11)

The structure of the arbitrary exemplar system is shown in Figure 40. The components are represented by A, B and C with functionality labels x, y, z, and the labelled paths that link the components together are represented by a, b, c, d, e and g. The Containing System is designated D and 'g' is the interconnection path to the containing system.

The N^2 matrix for the exemplar system in **D** is shown in Equation 9.1, as follows.

$$S_D = \begin{bmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & g \\ 0 & 0 & 0 & D \end{bmatrix} \qquad \text{Equation 9.1}$$

9.5 Construction of the System Adjacency Matrix and Graph

The first step in the analysis procedure is to construct the system Graph.

The system matrix is shown in Equation 9.1 and is repeated in Figure 41. Also the corresponding System Adjacency Matrix and System Graph is shown in Figure 41.

The program, ThesisM1, modifies the System Matrix as follows.

To construct the adjacency matrix the labels have been substituted with '1' to represent 'True' relationship between the nodes; otherwise '0' is used to represent 'False', meaning that there is no relationship. The nodes, A, B, C have been substituted with '0' to show

that there are no internal relationships of the form A->A, B->B, C->C that are of relevance. The system graph is a directed graph and it can be seen that there is a unidirectional link between A, C and that the links between A, B and B, C are bidirectional, corresponding to the matrix.

Insertbitmap

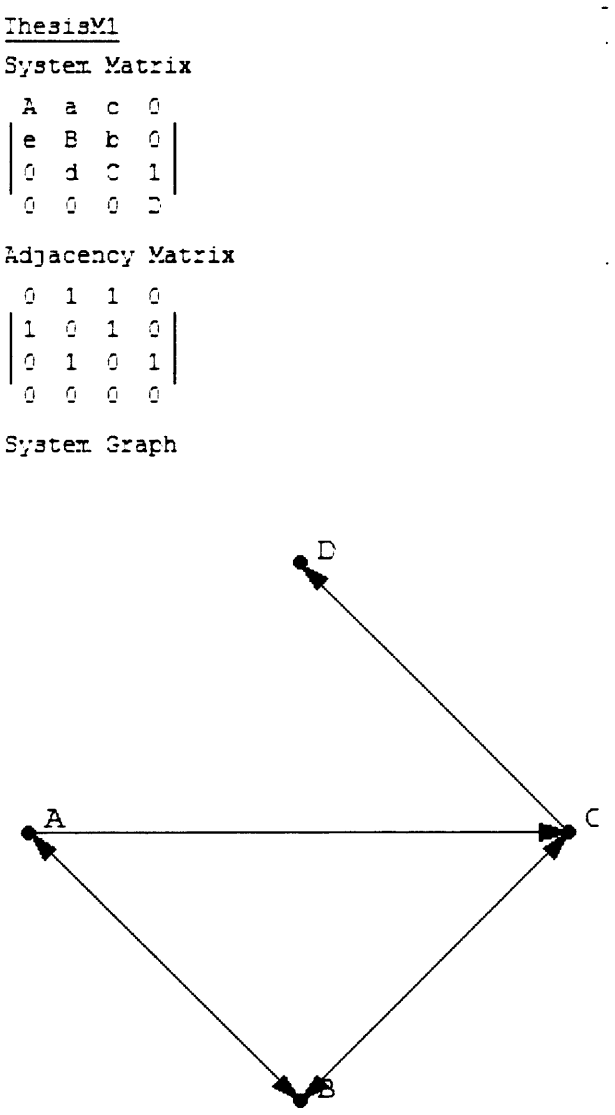


Figure 41 - Construction of System Adjacency Matrix and Graph (M1)

9.6 Determination of Number of Paths between each Node

Figure 42 shows the computation of the number of paths that link pairs of nodes. The two-step matrix shows the paths of length two, and the three-step matrix shows the paths of length three; for example, it can be seen by observation that there are two paths between nodes A and C, each of three steps. For a matrix of rank 4, the maximum number of steps between any source and sink nodes is four; therefore two, three and four step determination is sufficient to evaluate all potential source-to-sink paths.

ThesisM2

System Matrix

A	a	c	0
e	B	b	0
0	d	C	1
0	0	0	D

Adjacency matrix

0	1	1	0
1	0	1	0
0	1	0	1
0	0	0	0

Path Structure Matrix - 2 steps

1	1	1	1
0	2	1	1
1	0	1	0
0	0	0	0

Path Structure Matrix - 3 steps

1	2	2	1
2	1	2	1
0	2	1	1
0	0	0	0

Path Structure Matrix - 4 steps

2	3	3	2
1	4	3	2
2	1	2	1
0	0	0	0

Figure 42 - Matrices showing number of paths between each node (M2)

9.7 Determination of Direct Product/Sum Node-to-Node Expressions

The systems architect needs to be able to analyse the end-to-end functionality in two ways. Firstly, to analyse the source-to-sink functionality on the assumption that the interconnectivity relationships transfer information between functional blocks without modification or constraint. Secondly, to analyse the interconnectivity relationships on the assumption that each function block simply outputs all information received without modification, in effect, a transfer function of unity.

Therefore, two forms of analysis are required:

- Firstly, to derive the node-to-node relationship expressions on the assumption that the transmissivity between the nodes is unity (True) or null (False).
- Secondly, to derive the path-to-path relationship expressions for their interconnectivity on the assumption that the nodes have unity transmissivity.

To determine the node-to-node expressions of functional labels, the convention that has been assumed is that each function will be applied at the output of the function block. It is also assumed that the same function applies to all 'True' path links in each row. Therefore, the adjacency matrix for the node-to-node expressions is shown in Equation 9.2.

$$A_{Dn} = \begin{bmatrix} 0 & x & x & 0 \\ y & 0 & y & 0 \\ 0 & z & 0 & z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Equation 9.2}$$

The adjacency matrix for the path-to-path expressions of functional labels is shown in Equation 9.3.

$$A_{Dp} = \begin{bmatrix} 0 & a & c & 0 \\ e & 0 & b & 0 \\ 0 & d & 0 & g \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Equation 9.3}$$

With respect to node-to-node functionality, the two, three and four step functionality is shown in Figure 43. With respect to path-to-path functionality, the two, three and four step functionality is shown in Figure 44.

ThesisM3

System Matrix

$$\begin{vmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & 1 \\ 0 & 0 & 0 & 2 \end{vmatrix}$$

Labelled Function Matrix

$$\begin{vmatrix} x & 1 & 1 & 0 \\ 1 & y & 1 & 0 \\ 0 & 1 & z & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Substitution Matrix

$$\begin{vmatrix} 0 & x & x & 0 \\ y & 0 & y & 0 \\ 0 & z & 0 & z \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - 2 steps

$$\begin{vmatrix} xy & xz & xy & xz \\ 0 & xy-yz & xy & yz \\ yz & 0 & yz & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - 3 steps

$$\begin{vmatrix} xyz & x\ xy-yz & x^2y-xyz & xyz \\ xy^2-y^2z & xyz & xy^2-y^2z & xyz \\ 0 & z\ xy-yz & xyz & yz^2 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - All Steps

$$\begin{vmatrix} xy-x^2y^2-xyz-xy^2z & x-xz-x^2yz-x\ xy-yz\ -xz\ xy-yz & x-xy-x^2y-x^2y^2-xyz-x^2yz-xy^2z & xz-xyz-x^2yz-xyz^2 \\ y-xy^2-y^2z-xy^2z & xy-yz-xyz-\ xy-yz^2 & y-xy-xy^2-y^2z-xy^2z-xy\ xy-yz & yz-xyz-yz\ xy-yz \\ yz-xy^2z-y^2z^2 & z-xyz^2-z\ xy-yz & yz-xyz-xy^2z-y^2z^2 & z-yz^2-xyz^2 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function from A to C - All Steps

$$x-xy-x^2y-x^2y^2-xyz-x^2yz-xy^2z$$

Figure 43 - Computation of node-to-node functionality with two, three and four step functionality (M3)

ThesisM4

System Matrix

$$\begin{vmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & g \\ 0 & 0 & 0 & 2 \end{vmatrix}$$

Labelled Function Matrix

$$\begin{vmatrix} 0 & a & c & 0 \\ e & 0 & b & 0 \\ 0 & d & 0 & g \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - 2 steps

$$\begin{vmatrix} ae & cd & ab & cg \\ 0 & bd-ae & ce & bg \\ de & 0 & bd & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - 3 steps

$$\begin{vmatrix} cde & a\,bd-ae & bcd-ace & abg \\ bde-ae^2 & cde & b^2d-abe & ceg \\ 0 & d\,bd-ae & cde & bdg \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function Structure Matrix - All Steps

$$\begin{vmatrix} ae-abde-cde-a^2e^2 & a-cd-acde-a\,bd-ae\,-cd\,bd-ae & ab-c-ab^2d-bcd-a^2be-ace-c^2de & abg-cg-bcdg-aceg \\ e-bde-ae^2-cde^2 & bd-ae-cde-\,bd-ae^2 & b-b^2d-abe-ce-bcde-ce\,bd-ae & bg-ceg-b\,bd-ae\,g \\ de-bd^2e-ade^2 & d-cd^2e-d\,bd-ae & bd-b^2d^2-abde-cde & g-bdg-cdeg \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Function from A to C - All Steps

$ab-c-ab^2d-bcd-a^2be-ace-c^2de$

Figure 44 - Computation of path-to-path functionality with two, three and four step functionality (M4)

It is important to note that the expressions are in the form of a Boolean Product of the function labels. Therefore, the expression for the relationship between A and C shown as

$$x + xy + x^2y + x^2y^2 + xyz + x^2yz + xy^2z$$

should be understood as

$$x \,||\, x \,\&\, y \,||\, x \,\&\, y \,\&\, z.$$

(Note. Functions of the form x^2 & y^2 are x & y; x || y means x or y.)

9.8 Demonstration of Capacity Determination

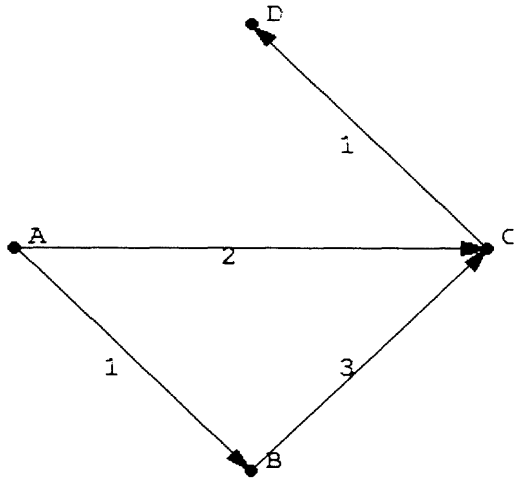
The graph with values of path transportation capacities in terms of arbitrary 'weight' values attached to each edge is shown in Figure 45. Also shown is a graph that highlights the cut set of capacity limited paths from A to C.

ThesisM5

System Matrix

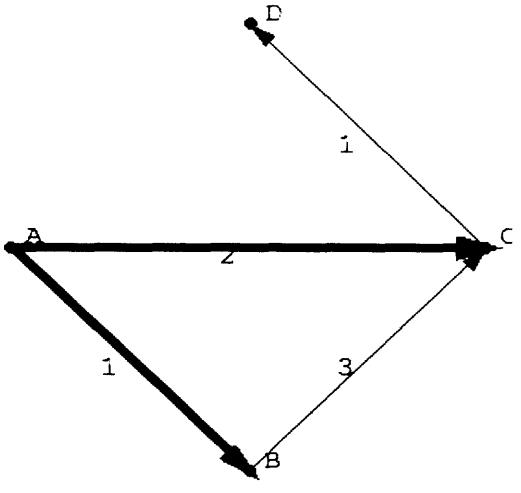
A	a	c	0
e	B	b	0
0	d	C	1
0	0	0	D

Graph with Edge Weights



- Graphics -

Graph with Cut Set



- Graphics -

Figure 45 - Graph with arbitrary edge capacity values and with highlight of capacity capability between A and C (M5)

9.9 Demonstration of Functional Determination

To demonstrate the functional composition of the system, each node is allocated a function, x, y, z, and the node-to-node interrelationships are shown as True/False. Then powers of the adjacency matrix generate the functional construction of each node-to-node path.

To demonstrate the functional substitution of expressions showing the functional composition, a simple example that assumes that the ANDed path functions are combined using the PLUS rule of association, and that there is a single through variable (e.g. current) so that the function labels, A, B, C, are substituted by three across variables (e.g. voltage), shown as V1, V2, V3. . The result is shown in Figure 46.

ThesisM6

System Matrix

A	a	c	0
e	B	b	0
0	d	C	1
0	0	0	D

Labelled Function Matrix

x	1	0	0
0	y	1	0
0	0	z	1
0	0	0	0

Labelled Function Substitution Matrix

0	x	0	0
0	0	y	0
0	0	0	z
0	0	0	0

Logical Domain Labelled Function Structure Matrix - All Paths

0	x	xy	xyz
0	0	y	yz
0	0	0	z
0	0	0	0

Real Domain Labelled Function Structure Matrix - All Paths

0	x	x-y	x-y-z
0	0	y	y-z
0	0	0	z
0	0	0	0

Real Function Substitution Matrix

0	V1	V1-V2	V1-V2-V3
0	0	V2	V2-V3
0	0	0	V3
0	0	0	0

Function for A to C Path

V1-V2-V3

Figure 46 - Example of analysis of series across variable 'voltage' functions (M6)

It can readily be seen that elements {1,2} and {2,3} and {3,4} show the across variable voltage function between nodes A, B and B, C and C, D as V1, V2, V3 respectively, and that element {1,4} is their sum from A to D.

Each step of the computation process is shown, as follows.

1. The 'Labelled Function Matrix' shows that there are three functions x, y, z in series.
2. The 'Labelled Function Substitution Matrix' shows the functions x, y, z in their corresponding adjacency elements.
3. The 'Logical Domain Labelled Function Matrix' shows the expressions for all the paths in logical domain form, meaning 'x & y & z' etc.
4. The 'Real Domain Labelled Function Matrix' shows the expressions for all paths with real domain rules of association; i.e. with the '&' rule replaced by the 'sum' rule of association.
5. The 'Real Function Substitution Matrix' shows the substitution of the function labels x, y, z with the across variable 'voltage' functions V1, V2 and V3.

The computation process shows that there are three distinct steps to quantify the functionality in real terms. These are: -

1. Compute the path structures in the Logical Domain.
2. Substitute the rules of association relevant to the Real Domain of interest.
3. Substitute the function labels with the actual functions for quantification.

9.10 Functional Determination of Interconnectivity Relationships

The method of analysis to determine the functionality of the inter-node interconnectivity is very similar to that used to determine the node-to-node functionality. To demonstrate the functional composition of the interrelationships in the system, each path is labelled as a, b, c, d, e, and f as before. For simplicity of demonstration, only the feed-forward paths a, b are enabled. Then, assuming that each path is a simple first order attenuation function, where 's' is the Laplace operator and k, l, m, p, q, r are arbitrary constants, the node-to-node attenuation functions are calculated. The result is shown in Figure 47.

ThesisM7

System Matrix

$$\begin{vmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & g \\ 0 & 0 & 0 & D \end{vmatrix}$$

Labelled Interconnectivity Matrix

$$\begin{vmatrix} 0 & a & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & 0 & 0 & g \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Two Steps

$$\begin{vmatrix} 0 & 0 & ab & 0 \\ 0 & 0 & 0 & bg \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Three Steps

$$\begin{vmatrix} 0 & 0 & 0 & abg \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Four Steps

$$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - All Steps

$$\begin{vmatrix} 0 & a & ab & abg \\ 0 & 0 & b & bg \\ 0 & 0 & 0 & g \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labels Substituted with Attenuation Function where 's' is the Laplace operator

$$\begin{vmatrix} A & \frac{k}{p+s} & \frac{kl}{(p+s)(q+s)} & \frac{klm}{(p+s)(q+s)(r+s)} \\ 0 & B & \frac{l}{q+s} & \frac{lm}{(q+s)(r+s)} \\ 0 & 0 & C & \frac{m}{r+s} \\ 0 & 0 & 0 & D \end{vmatrix}$$

Series Attenuation Function from A to D - All Steps

$$\frac{klm}{(p-s)(q-s)(r-s)}$$

Figure 47 - Example showing Functionality of Interconnectivity Path Structures – with Feed forward Interconnections only (M7)

It can be seen that the path structure between 'A' and the 'Containing System' is a&b&g.

Since the functionality is that of series attenuation functions, the rule of association for AND is 'Multiply'. Then, with the label substitution rule of a->k/(s+p), b->l/(s+q), and f->m/(s+r), the end-to-end attenuation function is: -

$$(klm)/[(s+p)(s+q)(s+r)]$$

A more complex example is shown in Figure 48a. Here all the interconnection paths are enabled. The attenuation functions are first order, 's' is the Laplace operator and j,k,l,m,n,p,q,r are arbitrary constants. The rules of association are that AND means Multiply and OR means Plus.

ThesisM8a

System Matrix

$$\begin{vmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & g \\ 0 & 0 & 0 & D \end{vmatrix}$$

Labelled Interconnectivity Matrix

$$\begin{vmatrix} 0 & a & 0 & 0 \\ e & 0 & b & 0 \\ 0 & d & 0 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Two Steps

$$\begin{vmatrix} ae & 0 & ab & 0 \\ 0 & bd-ae & 0 & b \\ de & 0 & bd & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Three Steps

$$\begin{vmatrix} 0 & abd-ae & 0 & ab \\ ae-bde & 0 & bd-abe & 0 \\ 0 & bd-ade & 0 & bd \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Four Steps

$$\begin{vmatrix} ae-abde & 0 & abd-abe & 0 \\ 0 & bd-ae-abde & 0 & bd-abe \\ ade-bde & 0 & bd-abde & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labels Substituted with First Order Attenuation Function - Three Steps

$$\begin{vmatrix} A & \frac{jln}{(k+s)(m+s)(p+s)} - \frac{jq}{(k+s)(r+s)} & 0 & \frac{jl}{(k+s)(m+s)} \\ \frac{jq}{(k+s)(r+s)} - \frac{lnq}{(m+s)(p+s)(r+s)} & B & \frac{ln}{(m+s)(p+s)} - \frac{jlq}{(k+s)(m+s)(r+s)} & 0 \\ 0 & \frac{ln}{(m+s)(p+s)} - \frac{jnq}{(k+s)(p+s)(r+s)} & C & \frac{ln}{(m+s)(p+s)} \\ 0 & 0 & 0 & D \end{vmatrix}$$

Series Attenuation Function from A to B - Three Steps

$$\frac{jln}{(k-s)(m-s)(p-s)} - \frac{jq}{(k-s)(r-s)}$$

Figure 48a - Example showing Functionality of Interconnectivity Path Structures – with All Interconnections enabled and insertion of Attenuation Function (M8a)

It can be seen that the attenuation function between nodes A, B combines the attenuation function of two routes. In summary, each element can be reviewed in relation to the example system and the node-to-node attenuation functions can be checked by inspection.

The use of the same path structure for different functionality is illustrated, as follows in Figure 48b, by replacement of the attenuation function with an impedance function. The reader is reminded that Kirchoff's Law states that impedances in series are combined by addition, and that impedances in parallel are combined by addition of their admittances.

ThesisM8b

System Matrix

$$\begin{vmatrix} A & a & c & 0 \\ e & B & b & 0 \\ 0 & d & C & g \\ 0 & 0 & 0 & D \end{vmatrix}$$

Labelled Interconnectivity Matrix

$$\begin{vmatrix} 0 & a & 0 & 0 \\ e & 0 & b & 0 \\ 0 & d & 0 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Two Steps

$$\begin{vmatrix} ae & 0 & ab & 0 \\ 0 & bd-ae & 0 & b \\ de & 0 & bd & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Three Steps

$$\begin{vmatrix} 0 & abd-ae & 0 & ab \\ ae-bde & 0 & bd-abe & 0 \\ 0 & bd-ade & 0 & bd \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labelled Interconnectivity Path Structure - Four Steps

$$\begin{vmatrix} ae-abde & 0 & abd-abe & 0 \\ 0 & bd-ae-abde & 0 & bd-abe \\ ade-bde & 0 & bd-abde & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Labels Substituted with Combined Series and Parallel Impedance Function - Three Steps

$$\begin{vmatrix} A & \frac{1}{Z1+Z2+Z3}-\frac{1}{Z1+Z4} & 0 & \frac{1}{Z1+Z2} \\ \frac{1}{Z1+Z4}-\frac{1}{Z2+Z3+Z4} & B & \frac{1}{Z2+Z3}-\frac{1}{Z1+Z2+Z4} & 0 \\ 0 & \frac{1}{Z2+Z3}-\frac{1}{Z1+Z3+Z4} & C & \frac{1}{Z2+Z3} \\ 0 & 0 & 0 & D \end{vmatrix}$$

Series Impedance from A to D - Three Steps

Z1 - Z2

Series Impedance from A to B - Three Steps

$$\frac{1}{\frac{1}{Z1+Z2+Z3}-\frac{1}{Z1+Z4}}$$

Figure 48b Example Showing Functionality of Interconnectivity Path Structures with all Interconnections enabled and Insertion of Series/Parallel Impedance Functions (M8b)

For example, it can be seen that the three step path impedance between nodes A,D is:

$$Z1 + Z2$$

and that between nodes A, B is:

$$[(Z1+Z2+Z3)(Z1+Z4)]/(2Z1+Z2+Z3+Z4).$$

9.11 Mechanisation of Functional Insertion

Each function label can be linked to a variety of functions and, to be analytic, the functional models of each component type must be consistent and the rules for combining functional models must be clear and unambiguous. Further, the types of functional models will typically include integers, quotients, real numbers, inequalities, complex numbers, polynomials, rational functions and matrices. The relationships will typically include the laws of association, distribution and commutation. Further, both metrication and variable consistency are required for quantitative analysis.

9.11.1 Demonstration of Insertion of a polynomial function

For this example a polynomial is inserted into each label. The polynomial example is a simple 'straight line' form for each function with 'multiply' as the rule of association. The result is shown in Figure 49.

ThesisM9

Matrix Showing Labelled Path Structure

$$\begin{pmatrix} 0 & abe & b & ac & abdf & 0 \\ 0 & 0 & 0 & cdh & d & cg \\ 0 & e & 0 & cefh & def & 0 \\ 0 & 0 & 0 & 0 & 0 & g \\ 0 & 0 & 0 & h & 0 & gh \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrix Showing Labelled Paths replaced by first order polynomial expression; Columns 1 and 2 only

$$\begin{pmatrix} 0 & (ak - bk\,ek - am\,x - bm\,ek\,x - bk\,em\,x - bm\,em\,x^2) \\ 0 & 0 \\ 0 & (ek - em\,x) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Matrix Showing Labelled Paths replaced by first order polynomial expression; Columns 3 and 4 only

$$\begin{pmatrix} (bk - bm\,x) & (ak\,ck - am\,ck\,x - ak\,cm\,x - am\,cm\,x^2) \\ 0 & (ck - dk\,hk - cm\,x - dm\,hk\,x - dk\,hm\,x - dm\,hm\,x^2) \\ 0 & (ck\,ek - fk\,hk - cm\,ek\,x - ck\,em\,x - fm\,hk\,x - fk\,hm\,x - cm\,em\,x^2 - fm\,hm\,x^2) \\ 0 & 0 \\ 0 & (hk - hm\,x) \\ 0 & 0 \end{pmatrix}$$

Matrix Showing Labelled Paths replaced by first order polynomial expression; Columns 5 and 6 only

$$\begin{pmatrix} (ak\,dk - bk\,fk - am\,dk\,x - ak\,dm\,x - bm\,fk\,x - bk\,fm\,x - am\,dm\,x^2 - bm\,fm\,x^2) & 0 \\ (dk - dm\,x) & (ck\,gk - cm\,gk\,x - ck\,gm\,x - cm\,gm\,x^2) \\ (dk\,ek - fk - dm\,ek\,x - dk\,em\,x - fm\,x - dm\,em\,x^2) & 0 \\ 0 & (gk - gm\,x) \\ 0 & (gk\,hk - gm\,hk\,x - gk\,hm\,x - gm\,hm\,x^2) \\ 0 & 0 \end{pmatrix}$$

Figure 49 - Example of Decomposition of a Direct Product with insertion of a polynomial function (M9)

ThesisM10

Matrix Showing Labelled Path Structure

0	a - b.e	b	a.c	a.d - b.f	0
0	0	0	c - d.h	d	c.g
0	e	0	c.e - f.h	f - d.e	0
0	0	0	0	0	g
0	0	0	h	0	g.h
0	0	0	0	0	0

Matrix of Labelled Path Structure with Add Rule of Association applied

0	a - b - e	b	a.c	a - b - d - f	0
0	0	0	c - d - h	d	c.g
0	e	0	c - e - f - h	d - e - f	0
0	0	0	0	0	g
0	0	0	h	0	g.h
0	0	0	0	0	0

Matrix showing Labelled Paths replaced by 2x2 generic matrices; Columns 1,2,3

0	{{a1 - b1 e1 - b2 e3, a2 - b1 e2 - b2 e4}, {a3 - b3 e1 - b4 e3, a4 - b3 e2 - b4 e4}}	{{b1, b2}, {b3, b4}}	
0	0	0	
0	{{e1, e2}, {e3, e4}}	0	
0	0	0	
0	0	0	
0	0	0	

Matrix showing Labelled Paths replaced by 2x2 generic matrices; Column 4

{{a1 c1 - a2 c3, a1 c2 - a2 c4}, {a3 c1 - a4 c3, a3 c2 - a4 c4}}
{{c1 - d1 h1 - d2 h3, c2 - d1 h2 - d2 h4}, {c3 - d3 h1 - d4 h3, c4 - d3 h2 - d4 h4}}
{{c1 e1 - c2 e3 - f1 h1 - f2 h3, c1 e2 - c2 e4 - f1 h2 - f2 h4}, {c3 e1 - c4 e3 - f3 h1 - f4 h3, c3 e2 - c4 e4 - f3 h2 - f4 h4}}
0
{{h1, h2}, {h3, h4}}
0

Matrix showing Labelled Paths replaced by 2x2 generic matrices; Columns 5,6

{{a1 d1 - a2 d3 - b1 f1 - b2 f3, a1 d2 - a2 d4 - b1 f2 - b2 f4}, {a3 d1 - a4 d3 - b3 f1 - b4 f3, a3 d2 - a4 d4 - b3 f2 - b4 f4}}	0
{{d1, d2}, {d3, d4}}	{{c1 g1 - c2 g3, c1 g2 - c2 g4}, {c3 g1 - c4 g3, c3 g2 - c4 g4}}
{{d1 e1 - d2 e3 - f1, d1 e2 - d2 e4 - f2}, {d3 e1 - d4 e3 - f3, d3 e2 - d4 e4 - f4}}	0
0	{{g1, g2}, {g3, g4}}
0	{{g1 h1 - g2 h3, g1 h2 - g2 h4}, {g3 h1 - g4 h3, g3 h2 - g4 h4}}
0	0

Matrix showing generic matrix elements populated with two port network admittance components; Columns 1,2 and 3 only

0	{{2, 2 R}, {2 s, 2 - 2 R s}}	{{1, 0}, {2 s, 1}}
0	0	0
0	{{1, R}, {0, 1}}	0
0	0	0
0	0	0
0	0	0

Matrix showing generic matrix elements populated with two port network admittance components; Columns 4,5 and 6 only

{{1, 2 R}, {0, 1}}	{{2 - 2 R s, R}, {3 2 s, 2}}	0	
{{2, R}, {2 2 s, 2}}	{{1, 0}, {2 s, 1}}	{{1, 2 R}, {0, 1}}	
{{2, 2 R}, {2 2 s, 2}}	{{2, R}, {2 2 s, 2 - 2 R s}}	0	
0	0	{{1, R}, {3, 1}}	
{{1, 0}, {2 s, 1}}	0	{{1 - 2 R s, R}, {2 s, 1}}	
0	0	0	

Figure 50 - Example of Decomposition of Direct Product with insertion of a Matrix Function with Plus rule of association (M10)

9.11.2 Demonstration using a matrix function with single rule of association

The matrix example is based on components each with a simple 2x2 form for each function with 'Plus' as the rule of association.

This example also demonstrates the use of the 2-port Network Model form to generate a source-to-sink transfer function. The network example is an RC network consisting of a mixture of series resistance and parallel capacitance components.

The matrix form of the Type A (Transfer Function) form of function as a 2-Port component is: -

$$\begin{bmatrix} Vin/Vout & -Iout/Vin \\ Iin/Vout & Iin/Iout \end{bmatrix}$$, where V is voltage and I is current.

Therefore, the A Form matrix of a series resistance is $\begin{bmatrix} 1 & R \\ 0 & 1 \end{bmatrix}$, and the A Form matrix of a parallel capacitance is $\begin{bmatrix} 1 & 0 \\ sC & 1 \end{bmatrix}$.

These are incorporated into the program by associating each function label with its associated Type A form of matrix. The path structure provides the actual construction of the network. The reciprocal of the transfer function of each path is determined by inspection of the {1,1} element of the source-to-sink matrix. The results are shown in Figure 50.

It can be seen by inspection that the output matrix at Row 3, Column 4, {3,4}, is $\begin{bmatrix} 2 & 2R \\ 2Cs & 2 \end{bmatrix}$, and that the transfer function (i.e. *Vout/Vin* as a voltage gain) between {3,3} and {4,4} is ½ = 0.5.

9.11.3 Demonstration using a matrix function with combined Multiply and Plus rules of association

It can be seen that the normal form of the path structure matrix is for the labelled path structure expression to be a combination of AND (i.e. multiply) and OR (i.e. add) rules of association.

The pictorial view of the demonstration is that the functionality of the system structure is similar to a network that consists of a mixture of series and parallel resistance

components. This is required as network theory requires that the A Form is required for series combination of components and that the admittance Form is required for parallel combinations. This means that the process of computation must be constructed as follows.

1. Replace the function labels with the A Form [{a11, a12}, {a21, a22}] of the resistance components.
2. For those parts of the labelled expressions that are Direct Products multiply the A Form of the components together.
3. Then transform the result into a 2x2 matrix in Type Y form (Admittance function) using the transform:

   [{a22/a12, -detA/a12}, {1/a12, -a11/a12}].

4. The result will be an expression that is in the form of a Direct Sum of 2x2 matrices.Then complete the computation by summing the admittance form matrices.
5. The admittance form(Y) can then be transformed to the A Form so that the outport open transfer function (i.e. 1/a11), is easily accessible using the transform:

   [{-y22/y21, 1/y21}, {-detY/y21, y11/y21}].

This calculation is demonstrated in Figure 51 and Figure 52. Figure 51 shows the admittance function of the element {1,2} and Figure 52 shows the transfer function for the pair of nodes A to F.

9.11.4 Summary

These program fragments demonstrate the method by which the functional population of the direct product expressions can be determined. This demonstrates that the method provides the system architect with the means of quantitatively determining the functionality between any source-to-sink of the defined architecture. The only constraint is the ability of the architectural team to generate and populate the model with quantified expressions of functionality. This is not meant to be a trite comment as the author appreciates that the task of component modelling is a complex task in its own right.

Note. To enable the result to be presented in an A3 format both programs show the calculation for only paths of length one and two steps. However, the programs can be used to calculate paths of higher step length by simple modification of the matrix power instruction.

ThesisM11

System Matrix

$$\begin{vmatrix} 0 & a & b & 0 & 0 & 0 \\ 0 & 0 & 0 & c & d & 0 \\ 0 & e & 0 & 0 & f & 0 \\ 0 & 0 & 0 & 0 & 0 & g \\ 0 & 0 & 0 & h & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Matrix Showing Labelled Path Structure

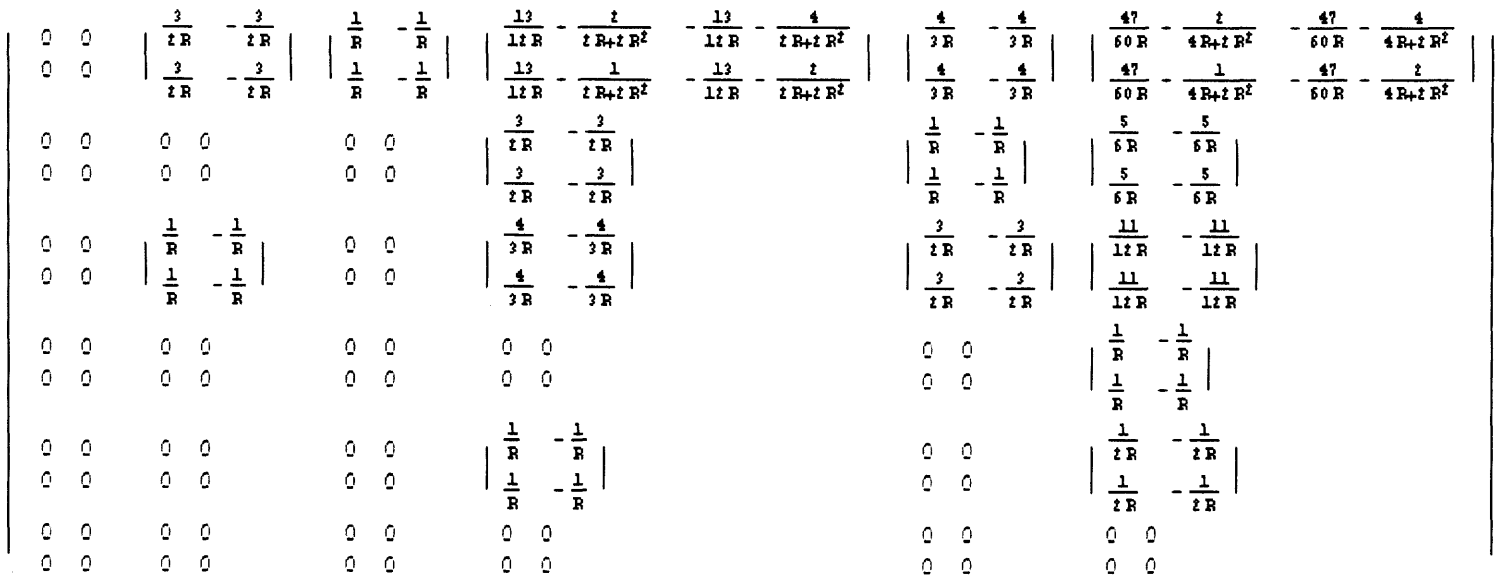
$$\begin{vmatrix} 0 & a-b.e & b & a.c-b.ce-fh-a.d.h-b.d.e.h & a.d-b.f-b.d.e & a.c.g-ad-bf.g.h-b.c.e.g-b.d.e.g.h \\ 0 & 0 & 0 & c-d.h & d & c.g-d.g.h \\ 0 & e & 0 & c.e-f.h-d.e.h & f-d.e & c.e.g-f.g.h-d.e.g.h \\ 0 & 0 & 0 & 0 & 0 & g \\ 0 & 0 & 0 & h & 0 & g.h \\ 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

2-Port Network Form showing Matrix Admittance Function of Element 1,2

$$\begin{vmatrix} \frac{a4}{a2} - \frac{b3\,e2+b4\,e4}{b1\,e2+b2\,e4} & \frac{a2\,a3-a1\,a4}{a2} - \frac{(b3\,e1+b4\,e3)(b1\,e2+b2\,e4)-(b1\,e1+b2\,e3)(b3\,e2+b4\,e4)}{b1\,e2+b2\,e4} \\ \frac{1}{a2} - \frac{1}{b1\,e2+b2\,e4} & -\frac{a1}{a2} - \frac{-b1\,e1-b2\,e3}{b1\,e2+b2\,e4} \end{vmatrix}$$

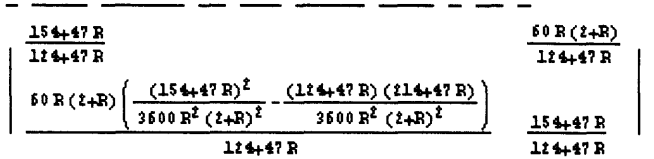
Figure 51 - Example of Decomposition of Direct Product with insertion of a Matrix Function with combined Times and Plus rules of association; Part a (M11)

2-Port Network Form showing Source to Sink Path Admittances in Matrix Form



Admittance Matrix of All Paths Structure from Node A to Node F

A Form Matrix of All Paths from A to F



Transfer Function of All Paths A to F

$$\frac{124 - 47R}{154 - 47R}$$

Figure 52 - Example of Decomposition of Direct Product with insertion of a Matrix Function with combined Times and Plus rules of association; Part b (M11)

10 EVALUATION OF ARCHITECTURAL STRUCTURES

10.1 Architectural Knowledge

The decomposition of the system viewpoints into the domains of Composition, Capacity, Messenger and Functional behaviour was introduced in Section 7. Further, the means of analysis for functional matching and source-to-sink functional evaluation was described in Section 9.

These domains have been chosen to focus the attention of the architect to the key review questions:

- What is its construction as a machine; how does it work?
- What is its ultimate performance?
- What is the role and impact of the system interconnections?
- What is its functional behaviour?

Clearly, each question invokes a range of viewpoints for consideration. Experience shows that a peer group review will identify many ways of looking at the solution; some will be simple while others will be complex. Further, a key aspect of the review process is to educate the review group so that they understand what the design solution offers.

A well-known example as to the importance of viewpoint selection is associated with the motion of our planetary system. While the Greek astronomer, Ptolemy, had correctly established that the earth was round, he taught that the earth was at the centre of the universe. While the scheme had some success, later measurements showed that the complications that had to be introduced to make the model work, invoked inconsistent behaviour. These were resolved by the work of Copernicus and Galileo over one thousand years later.

System architects face the same problem. If the architect is unable to ensure that the review group understands the machine structure, the review will become overly complex and the risk of misunderstanding increase unnecessarily. More importantly, decision-making groups that do not appreciate the basic structure of the machine will be prone to erroneous decision-making. The adverse impact of such ignorance is witnessed all too frequently!

Clearly, it should be understood that the architect should understand the machine system structure. However, modern engineering environments that focus on detail tend to obscure essential structural characteristics of machines. The volume growth of such detail information is at the heart of the comments by Robert Frosch [2a]. Further, as many major programmes take many years to achieve 'first of class' acceptance and may be in-service for some decades, it is characteristic that the intentions of the architectural team get lost in the mists of time. It has long been apparent that the obscuration of the tacit knowledge held in common by the primary design team as the team disperses after the development programme is complete, leads to substantial difficulties for change implementation and certification during the operational phase of the programme.

Consequently, a part of the motivation for this thesis is to propose methods that will enhance the information available to those engineers whose role it is to sustain or modify system performance during the operational phase of the programme; that is, in effect, to support evolutionary development.

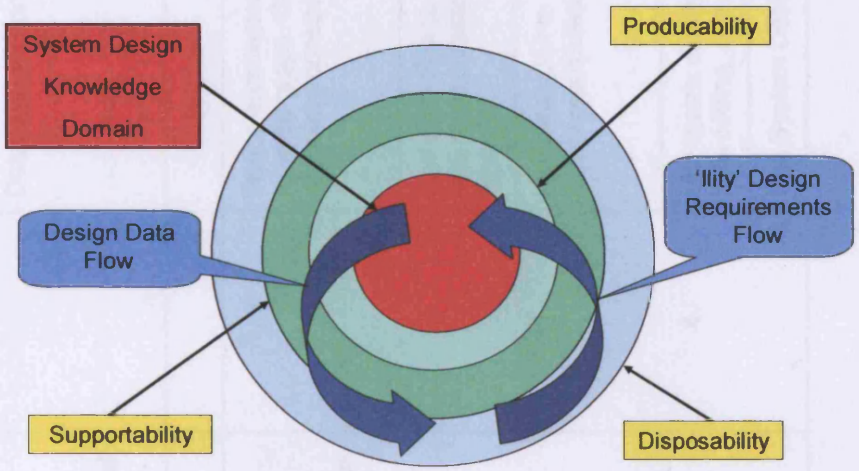
While no specific claims can be made as to the number of viewpoints that an architect may need to consider, it is imperative that the type and aggregation of the overall number of viewpoints that are generated are evaluated as a group in terms of the value added to the design knowledge of the overall properties of the machine system. This is in addition to the justification of their existence in relation to the evaluation of specific atomic requirements. A Framework component for the evaluation of viewpoints as a group is described in the next section.

10.2 Context of Domain Knowledge

The author has postulated (q.v Section 7) that the decomposition of system design knowledge into the four domains fulfils the conditions of both necessity and sufficiency to define the design of a system. Specifically it provides a structure to position and aggregate all viewpoint analyses; the intention is to prompt the architect into defining each viewpoint so that the point performance requirements for the set of key questions can be answered for each viewpoint and that sufficient viewpoints are chosen to determine all the point performance requirements. Further, it should be appreciated that the system Design Knowledge Domain is just part of the total information held in the System Repository.

It is important to understand that the System Design Knowledge Domain provides the data to support many viewpoint evaluations, both qualitative and quantitative. For example, Figure 53 is a pictorial representation of the interrelationships that link design data to 'ility' evaluations. It shows that System Design Knowledge data that is specific to

the design of the system as a machine 'flows' to the 'ility' domains including produceability, supportability and disposability. The results from 'ility' domain evaluations then feed back to the System Design Knowledge domain wherein the the design of the System as a mechanism may be adjusted to provide enhanced compliance with 'ility' performance objectives.



1

Figure 53 - Schematic Showing Relationship between System Design Knowledge Repository and the 'ility' Domains (P20)

The implication is that the integrity of the Design Knowledge Domain must be impeccable; if not, the results and interpretation of 'ility' and other viewpoint determinations will be erroneous. The information held in these four domains must then be both sufficient and necessary to define the core design.

In the context of the DODAF model the system Design Knowledge Domain is part of the Technology Sector, and it is this that supports the other domains including the Logistical and Operational Sectors. To facilitate the organisation of information for large projects, typically, the system information repository is held in a formal structure or Framework. This Design Knowledge Domain can form the Framework Component assigned for the architectural structure and its functional construction.

10.3 Domain Activity Population

To answer the key questions, posed at the beginning of this chapter, the author has compiled exemplar sets of technology information that must be determined in order to describe the machine system.

Architectural Domain Determination				
Domain	**Activity**	**Technique**	**Analysis Type**	**Analysis Support**
Design Data Repository	Hierarchical decomposition with consistent traceability of functional and realisation building blocks.	Define lists of building blocks and interconnection attributes including component, interface, design, composition, substance and physical properties definition.		Database support.
Composition	Define arrangement of architecture construction with building blocks and interrelation links.	Determine building blocks and interconnection attributes including component, interface, design, composition, substance and physical properties definition.	1, 5	Define architecture in the form of a labelled graph. Compile adjacency matrix of system construction.
	Implementation allocation analysis with functional and realisation building blocks.	Determine hierarchical allocation compatibility.	2	Use extension to 'Design Matrix', **F=AD** with matrix manipulation with Boolean logic.
		Functional and Implementation Causality Compatibility using networks with functional and realisation building blocks.	3	Use extension to network analysis using matrix manipulation with Boolean logic.
	Partition efficiency	N^2 form with type coded interfaces with matrix cluster analysis.	4	Use matrix cluster analysis including link type coding, see e.g. [74].
	Connectivity completeness	Data flow model		Use System Design Tool of choice.

Architectural Domain Determination				
Domain	**Activity**	**Technique**	**Analysis Type**	**Analysis Support**
Capacity	Analysis of capacity of arrangement by linking capacity of each building block 'source-to-sink' path structure of the system arrangement.	Determine 'source-to-sink' paths. Create source-to-sink capacity structure of each path from capacity of each building block and messenger link. Decompose path algebra to real functions. Then analyse function disciplines (See Table 20) as required.	6a	Populate function labels in system adjacency matrix with capacity functions. Use Graph Theory for transport flow capacity analysis, using 'units' of capacity Use graph theory with function labels populated with capacity functions; these may include polynomial and matrix forms for algebraic and calculus function forms.
Messenger	Evaluate links, as ICDs, between building blocks in terms of information, message and platform layer analysis, including technology typing.	Analysis of information, message coding and delivery technology. Deterministic and stationary/non-stationary parameter, message and signal processing.	6b	Populate function labels in system adjacency matrix with information transfer functions. Use graph theory with function labels populated with information transfer functions; these may include polynomial and matrix forms for algebraic and calculus function forms. Note. All functions must have consistent units and dimensions.

Architectural Domain Determination				
Domain	**Activity**	**Technique**	**Analysis Type**	**Analysis Support**
Functional Behaviour	Analyse behavioural functionality of arrangement by linking state space functionality of each building block 'source-to-sink' path structure of the system arrangement.	Determine source-to-sink paths. Create 'source-to-sink' functional structure of each path from behavioural functionality of each building block. Decompose path algebra to real functions. Then analyse function disciplines (See Table 20) as required.	6c	Populate function labels in system adjacency matrix with building block functional behaviour functions. Use graph theory with function labels populated with functional behaviour functions; these may include polynomial and matrix forms for algebraic and calculus function forms. Note. All functions must have consistent units and dimensions.

Table 15 - Table of Machine System Design Domain Analysis Requirements

Legend

Type Reference	Description
1	Hierarchical structure
2	Functional and Implementation resource compatibility analysis.
3	Functional and implementation resource causality compatibility analysis.
4	Architecture construction efficiency analysis, see e.g. cluster analysis [74].
5	Define labelled graph structure and form system structure adjacency matrix. Compute direct product/sum expressions for source-to-sink viewpoint of interest path structures.
6 a, b, c.	Populate expressions with rules of binary association. Populate function labels with actual functional expressions; a-> capacity functions, b-> information transfer functions, c-> functional behaviour functions.

Table 16 - Type Description of Methods of Analysis

Occupancy. Physical properties. Molar capacity.	Displacement capacity. Gravitational capacity. Hydraulic/pneumatic capacity.	Thermodynamic capacity. Electrical capacity. Magnetic capacity.	Radiation capacity. EM capacity. Optical capacity. Radioactivity capacity.	Informatic capacity. Transportation capacity. Institutional activity capacity.

Table 17 - List of Disciplines

Table 15 shows the four design domains each with its main design analysis techniques that are described in this thesis, together with the type of analysis required for quantification. The List of Disciplines, shown in Table 17, provides the cross reference by Type allocation of Physical Science Disciplines to the analysis methods described in Sections 8 and 9.

Population of the analysis requirements described in Table 15 places heavy reliance on modelling and simulation. Their capability was introduced in Section 2 and it can be seen that each discipline has evolved preferred methods and techniques of analysis. To support the means of functional population described in Section 9 it is imperative that each function type is accompanied by consistent rules of association. Two such methods, i.e. Linear Two-port Networks and Bond Graph constructions, that enable complex systems to be synthesised from its constituent components, were described in Section 4.

The investigation of Linear Two-port capability confirmed the completeness and maturity of the technique. To demonstrate the determination of source-to-sink functional behaviour, each label was allocated a generic 2x2 matrix form. Then the path functionality was determined by the series or parallel combination of these 2x2 matrices. A demonstration of this capability is show in Figure 51 and Figure 52.

The investigation into Bond Graph capability showed that the full range of technology domains that are pertinent to systems engineers, has not been established. The author has included suggested forms of pseudo Bond constructions that discipline experts could refine in Appendix 3. Taken together they provide a more comprehensive range of consistent component models across the full range of science disciplines than can be tackled by the linear Two-port methodology. Therefore, the Design Analysis Table shows the list of technologies for which functional viewpoints are in common use by systems architects, all of which have the potential to be matched with a corresponding Bond and Pseudo Bond component models.

While the author's investigations did not result in the identification and validation of a complete set of component models, the consistent use of the standard state space form with generic manipulations for series and parallel combinations will enable the end-to-end functional performance to be determined. The basic construction of such combinations is also shown in Appendix 3.

However, the reader should note that the author's investigations into sources of expertise that included the UK, Europe and the USA failed to identify any source of expertise to validate the proposed constructions and provide some guidance as to the methods that

are employed to combine component models. Consequently, the author concluded, with regret, that the maturity of information implied that it was not practical to pursue at this time or generate a demonstration of the construction for this thesis.

Consequently, the examples in this thesis have been confined to the use of Two-Port Network Analysis as it has a well defined set of consistent component models and means of association for both series and parallel combinations.

10.4 Summary of Procedure to Determine the Functional Structure of each Domain

The proposed method implies that there is a single unique architectural structure for any application; clearly, there cannot be competing architectures for a particular application. However, there can be subset architectures to represent specific parts of the system. The rationale for the four design domains (See Section 7.2) is to ensure complete coverage of the design while minimising their functional overlap. While it is necessary that each domain is competently evaluated, the overall depth profile of evaluation for each domain will be at the discretion of the architect's view of complexity; it is recognised that it will be necessary for some specific evaluations to be carried out with considerable depth.

The steps involved in the procedure are summarised as follows.

1) Define system components and interconnection paths.
2) Define architectural structural with Building Blocks and interconnections.
3) Create N^2 matrix for system architecture; allocate function labels. (Use Excel)
4) Determine implementation structure; define implementation components.
5) Analyse integrity of hierarchical decomposition; match and reconcile functional and implementation resource structures. (Use Design Matrix with True/False relationship coding).
6) Analyse integrity of causality structure; match and reconcile functional and implementation resource structures. (Use logical network form with functional and implementation resource components.)
7) Determine efficiency of architecture structure; (e.g. use binary coding to type code interface definitions; use cluster analysis to determine efficiency). Modify architectural structure definition as appropriate.
8) Create sub-set N^2 matrix for each domain from system architecture. Assess viewpoints required to be incorporated within by each domain.
9) Derive function labelled Adjacency Matrix for each domain.

10) Determine structural integrity of each domain. Use True/False Boolean representation of component-to-component links. Refine structural integrity so that it is consistent and complete.

11) For the Composition domain, allocate constitution type functionality to each component. Analyse source-to-sink path structure as direct product expressions of function labels. Decompose direct product/sum expressions by replacing labels and association with viewpoint functions and rules of association. Assess source-to-sink Structural functionality.

12) For the Capacity domain, allocate Capacity type functionality to each component. Analyse source-to-sink path structure as direct product expressions of function labels. Decompose direct product/sum expressions by replacing labels and association with viewpoint functions and rules of association. Assess source-to-sink capacity capability, for both normal range and maximum and minimum conditions.

13) For the Messenger domain, allocate Messenger type functionality to each interrelationship component. Analyse source-to-sink path structure as direct product expressions of function labels. Decompose direct product/sum expressions by replacing labels and association with viewpoint functions and rules of association. Assess source-to-sink functionality of information transfer structure.

14) For the Functional Behaviour domain, allocate behavioural functionality to each component. Analyse source-to-sink path structure as direct product/sum expressions of function labels. Decompose direct product expressions by replacing labels and association with viewpoint functions and rules of association. Assess source-to-sink performance, behaviour and dynamics.

15) Evaluate emergent properties of each domain by comparison with the emergent properties required by the specification.

16) Modify system design to ensure compatibility of estimated emergent properties with required emergent properties.

10.5 Proposed Procedure for Structured System Design

The proposed labelled path structure method has its roots in abstract algebra that considers relations, functions, partial orders and induction. Graph theory enables the path structure between any source and sink nodes to be determined with each path typically involving many steps. The technique of labelling each path enables the functional components involved in any path to be identified. Path analysis generates the functional combination of labels in any path in the form of a direct product.

As explained by S J G Taylor [113], Cartesian Products produce ordered n-tuples in which there are not necessarily any implied functions relating the entities thereof. A Direct Product/Sum is of the form a ∘ b / a + b where each part of the product/sum is required to have the algebraic terms and the binary rules of composition e.g. association, distribution and commutation, defined as part of a coordinate system. Further, the coordinate system for a Direct Product/Sum encompasses the notion that operations can be applied independently to the 'coordinates' applicable to each individual term.

In general, the 'coordinates' can themselves be any well-defined algebraic structure. They do not have to be linear, numeric, commutable, well-ordered, or anything else; they could be, for example, lists, logical expressions, polynomials, inequalities, real or complex functions, matrices, or graphs.

Nevertheless, to be useful, the interrelationships between the entities need to preserve the structure in some sense; they need to be isomorphic. Therefore, they need to be specifically defined as part of the space structure spanned by the set of 'coordinates' – for example, as inner (dot) products are for metric spaces.

This means that the scope of analysis encompassed by the methodology includes all architectural viewpoints for which a consistent set of coordinates and algebraic rules of composition can be defined. For example, these include soft systems, management structures, and process structures. Consequently, the potential capability of the method is substantial.

Clearly, the range of applications that will be able to be addressed is not fully perceivable at this stage of its development. Therefore, the author does not consider it practical to provide a full description of the total capability provided by the proposed method in this thesis.

Nevertheless, to elucidate the capability encompassed by the method three types of analytical capability are described as follows.

i) Data sets associated with non-analytic functions.

Typical members of this category are character strings, lists and labelled objects. Relationships are simply formed from data manipulation operations including, for example, merge, partition and rank. It enables data or knowledge objects to be organised in some way so that an observer is able to identify patterns, or segregated types of knowledge of interest.

ii) Functions that are formalised in terms of logical, propositional calculus or Boolean algebra.

Typical members of this category include sets of numeric or character strings with defined value of meaning that can be manipulated according to defined logical rules. These rules are normally associated with binary relations for rules of union and intersection, and true and false propositions. These can be applied to sets that form groups, fields or rings. The functionality includes classical logic, propositional logic, first order languages including those associated with computational structures and time based sequential logical constructions.

iii) Functions that are formalised with integers, quotients, real and complex numbers.

The members of this category are defined by these number systems, the metrication of value and their associated algebraic rules, typically those of identity, association, distribution and commutation. The most common relationships are those of addition and multiplication. The functions are analytic and their capability includes those of equivalence, inequality, linear algebra and calculus.

The following table shows various types of expression together with data population options and means of association. Each type of expression has an associated coordinate system (or set of algebraic rules) that can be adopted to enable the quantitative implications of the binary combination of variables to be determined.

Type Ref.	Type of Expression	Data Population	Type of Association, Distribution and Commutation
1	Sets/Lists	Characters, integers, quotients, real numbers, complex numbers	Count, merge, union, combine, partition, rank.
2	Logical structures	Characters, integers.	True, false, union, intersection, Boolean algebra.
3	Propositional logic	Sets, characters, integers, quotients, real numbers, complex numbers.	True, false, union, intersection, Boolean algebra, first order predicate calculus.
4	Algebraic – inequality	Variables with integers, quotients, real numbers, complex numbers.	+,-,x, ÷,<,>,=
5	Polynomials	Variables with integers, quotients, real numbers, complex numbers.	+,-,x, ÷,<,>,=
6	Matrices	Variables with integers, quotients, real numbers, complex numbers.	+,-,x, ÷,<,>,=
7	Rational Functions – linear and non-linear	Variables with integers, quotients, real numbers, complex numbers.	+,-,x, ÷ ,<,>,=
8	Differential calculus	Variables with integers, quotients, real numbers, complex numbers.	+,-,x, ÷,<,>,=

Table 18 - Types of expression, data population and binary combination

The previous sections of this thesis show that a procedure for Structured System Design can be established that is based on the following rules.

- The system architecture consists of Building Blocks and Relationships.
- The architectural structure is defined in the form of a labelled directed graph, from an Adjacency Matrix.
- The functional structure and the implementation structure are reconciled to ensure compatibility.
- The architectural structure is evaluated to determine its effectiveness and efficiency.

- The set of viewpoints for each domain of the architectural structure to be analysed is defined.
- The set of functions for each label is defined, together with their through variables, coefficients, metrication standard and data sets.
- The rules of binary association for each functional viewpoint are defined.
- For each domain, the step wise through variable connectivity is determined in the form of direct product expressions of labels: 2-steps, 3-steps, 4-steps,... -> True/False.
- For each domain viewpoint, the labels are replaced with the viewpoint functions and the expression reformed using their rules of association.
- For each domain viewpoint, the expressions for 2-step, 3-step, 4-step, ... etc. paths are formed, the coefficients are populated with data and the expressions are solved for the defined variables. The variable values so determined constitute the emergent properties of the architecture for that domain viewpoint.
- The estimated emergent properties so determined are compared with the desired properties so that out of range divergences will be highlighted.

In summary, the construction enables complex implementation structures to be intimately related to functionality so that, for any functionality network, the attributes imposed by the implementation can be determined. Once the integrity of the system structure has been confirmed, the overall performance and robustness of the design can be estimated.

10.6 Demonstration of Architectural Analysis Techniques

To ensure that the proposed analytical techniques are practical to use for the assessment of a real life-sized application, the author devised an evaluation model based on an outline of a command and control system installed in an attack helicopter. The description and evaluation of this model is shown in APPENDIX 1.

The description of the system provides the reader with an insight as to the issues that must be taken into account by the architect of such a system. In particular, the reader should appreciate that requirements to ensure adequate flight safety have a substantial impact on the form and capabilities demanded of the implementation solution.

The evaluation demonstrates the functional analysis use of each of the methods of analysis; hierarchical implementation traceability, causality dependency and source-to-sink node functional performance.

For the purposes of this thesis, functional insertion is limited to various types of generic functional expressions. The primary reason is that the results of specific analysis are not

relevant to this thesis. In addition, very substantial systems engineering resources, probably involving many person-years of work by design specialists, are required to provide analyses that contribute value to engineering design, and are well beyond the singular personal capacity of the author.

The programs generated by the author for this thesis demonstrate the following properties:

- The generation of the architectural structure of each viewpoint and presentation in graphical form.
- The determination of the source-to-sink node labelled path expressions for functionally consistent viewpoints.
- The insertion of the rules of association for various functional viewpoints.
- The insertion of generic types of algebraic functional expression.
- The computation of source-to-sink functionality.

The rank of the N Squared matrix for this system exemplar is 34. As the size of this exemplar is similar to that produced by the author professionally for various aerospace systems, the reader is able to have confidence that the techniques and programs can be used for real sized practical applications.

With respect to the computation of source-to-sink functional performance, Excel programs were designed to support the generation of the N-Squared definition of the system, and the formation of the labelled adjacency matrix. Then Mathematica program notebooks were created to generate the directed graphs that describe the structure of the overall system and its decomposition into the four domains. The structure of each architectural viewpoint is shown diagrammatically, and the source-to-sink path structures using the functional labels are computed in the form of logical expressions. The programs also transpose these direct product/sum expressions by substitution of the rules of association pertinent to the type of function being processed together with the function associated with each label, and compute the source-to-sink functionality between each pair of nodes. Some data constants are inserted into selected functions to demonstrate that full quantification results can be obtained.

The form of each Notebook program is shown in Table 19.

Viewpoint Analysis Program Description
Data file input from Excel spreadsheet.
Construction of the system structure graph.
Generation of source-to-sink path structures.
Transpose of rules of association.
Insertion of functions.
Substitution of values for coefficients and variables.
Computation of functional expressions and performance estimates.

Table 19 - Construction of Mathematica Notebooks for Viewpoint Analysis

These programs have been designed to present the computing structure in a readable form; they have not been optimised in terms of programming expertise or efficiency. This has been done in order that potential users can use these programs as the starting point for tools that will support other applications of interest.

The author has constructed the system by the use of engineering judgement based on personal design experience of similar applications. The implication is that the design viewpoints proposed for formal analysis are based, in general, on the personal expertise of the architect.

The system matrix includes both 'hard' connected and 'function' connected relationships. The distinction is not necessarily obvious as can be appreciated by, for example, whether a radio link or a relationship with the atmosphere are seen as a 'hard' or 'function' connected relationships.

Human centred systems are sometimes classified as soft systems, however cockpit arrangements are designed to ensure that aircrew can fulfil their roles. Consequently human interaction interfaces also need to be included; often these are quantified in functional terms by the use of human engineering techniques. The architect must also take these into account. Consequently, as can be seen from Figure 65 [Helo Msr], the overall system matrix is very complex.

To enable the system to be analysed it was decomposed into its four domains: composition, capacity, interconnectivity and functionality. Each domain was represented by exemplar viewpoints based on the author's experience of real applications. Then programs were created for each viewpoint that enable its structure to be shown in graphical form, and pair-wise node to node quantitative analysis to be completed.

Each program has the generic structure shown in Table 8. Each program can be modified to reflect a particular viewpoint and functionality, and host the values that

enable quantification (See Note 1). The demonstrations include examples of the use of particular rules of association and algebraic forms of functional representation.

The results shown in the Appendix demonstrate the computation of the source-to-sink functionality for a variety of path structures with simple, polynomial and matrix forms of functional expressions.

The author found that it was sensible to control the number of steps taken for each path structure computation. The author also found that, from experience, paths of two and three steps, even and odd, provided substantial insight into the functional structure of each viewpoint. Paths of higher step length were evaluated. However, many of these 'long path' evaluations make use of 'short path' relationships; in effect a path structure first identified with 'm' steps then reoccurs with 'm + n' steps, where 'm + n' < rank of the adjacency matrix. Consequently, the author concluded that short path length evaluations provide valuable insight into the attributes of the architectural structure.

Nevertheless, as a key objective of this thesis was to provide a method that identified obscure pair wise relationships, there is a continuing need to find long path relationships. While the method enables this to be done, as the calculation involves taking powers of matrices $\mathbf{A}^n$, where n may be any integer between one and the rank of the system matrix, each matrix element may be a very complex function. In such cases, the analyst will need to refine the form of the display and analysis of such functions to ensure comprehension.

The demonstration confirmed that the analytical methods proposed in this thesis can be applied to real system applications, particularly those of substantial complexity.

Readers may be interested to note that the Mathematica kernel computed the results of all demonstrations within a few seconds on the author's 1.7 GHz PC using Mathematica Version 5.2.

(Note 1. The evaluation of the test model has been directed at demonstration of the use of design techniques for the purposes of this thesis. Therefore, all performance results are arbitrary, and are not representative of real life performance attributes.)

11 CONCLUSIONS

11.1 Review of Proposed Analytical Method

The objective of this study programme has been to identify means of representation and analysis of system architectures that have the comprehensiveness to substantiate a claim of design robustness.

The mathematical representation of a system definition is as follows.

$$S = (T, R) \qquad \text{Equation 11.1}$$

Where ... S is the system of interest

T is a set of components (things)

R is a set of relationships on T (system hood; connectivity)

Systems are structures consisting of building block components and their interrelationships. To enable system architects to obtain a comprehensive understanding of the system properties, the representation must enable all relations to be captured, identified and evaluated in a structured manner.

The primary method of complex system simplification is by the use of hierarchical decomposition, whereby each lower layer of decomposition is simpler than its immediately higher layer. This results in a commensurate increase in the number of relationships required to describe the system connectivity. While decomposition enables relevant stakeholders to focus on individual components and individual relationships, the problem of reconstruction becomes increasingly complex. The GST community showed that reconstruction is achieved when attribute aggregation, and functional interrelationship integrity has been established. Consequently, system architects need to have practical methods that enable systems of interest to be analysed accordingly.

As an aid to complexity reduction, it is proposed to represent complex architectures in four domains pertinent to system architects, viz, structure, capacity, informatic relationships ('messenger'), and interconnected functionality. The boundary assumptions for each domain ensure that each domain is distinct from each other and can be analysed independently; ideal cross boundary links are used by each domain to enable the functionality of the domain of interest to be evaluated without interference from the functionality of its associated domains.

These four domains together are sufficient to provide complete coverage of the design space required to be analysed to establish robustness of the complete system.

The author proposes that the N Squared form of construction provides a means of comprehensive description of system architectures, as it enables components and their interrelationships to be described in a structured form. When this form of decomposition is combined with abstract and linear algebra analysis, the architect has a formidable capability to evaluate the robustness of the design. Therefore, to achieve the primary objectives of this study the author investigated the use of abstract and linear algebra techniques to estimate the emergent properties of complex system structures.

To address the reconstructability requirements, three methods of analysis have been developed:

1. To describe and analyse the relationships between emergent properties and design structure, with specific consideration of the relationships between the structural allocation of functional building blocks and the implementation building blocks. Firstly, the method enables the construction of the emergent functionality from the design components to be determined. Secondly, the method enables the compatibility of the ideal and actual realisation allocations to be determined.
2. To describe and analyse the causal relationship between ideal causal flow structures and implementation building block causal flow structures. The method enables the functional and realisation structures to be compared.
3. i) To represent system architectures in graphical form as directed graphs, and demonstrate that graphical analysis can be used to support various kinds of quantitative analysis.

   ii) The use of labelled paths has enabled the interconnectivity to be described in the form of direct product expressions. These expressions have been decomposed and populated with quantifiable expressions that represent meaningful viewpoints of capability. The decomposition of these expressions to show quantified variable analysis has been demonstrated.

These three methods support the reconstructability requirements of intra-component compatibility, inter-component compatibility and functional integrity. Further, this thesis describes a process that enables structural analysis to be incorporated into the well-established 'V' Model of development.

11.2 Review of Proposed Embodiment Methodology

The traditional system engineering process model is based on the interaction of Requirements, Functionality and Realisation. While the need for Realisation to match Functionality and the Requirements has been a fundamental tenet of the design process, the complexity of scope, solution feasibility, technological scope and management structures involved in modern large scale applications have resulted in obscuration of the relationships particularly between functional construction and realisation.

This thesis commences with a review of process methods. It postulates that despite the development of enhanced process models, the means of matching functionality to realisation has not been developed to cater for the loss of 'one to one' matching. Inadequacies in solutions to match needs have prompted substantial interest in operational considerations. Modern process definitions have emphasised the behavioural aspects both from an operational perspective and from solution state transition characteristics. The consequence is that prediction of behaviour and ultimate performance estimation as provided by the realisation design, has become obscured.

The 'House of System Design' model has been proposed by the author to reassert the need for architectural structure and ultimate performance competence.

Personal experience has highlighted the need for a design team to achieve a mature architectural construction and confidence in the estimated ultimate performance capability for both the ideal and implementation solution concepts, prior to further development and refinement of the system solution building blocks. For example, as a design reviewer, the author would expect substantial confidence in these aspects before the level of achievement at the System Design Review milestone would be acceptable.

Therefore, it is expected that these analytical methods would be incorporated into the design process from an early stage in the overall design process. The methods of assessment for decomposition, causality determination and functionality are all designed to populate the knowledge base for the specific applications so that the architect is able to support with confidence design statements that assert that the proposed solution is fit for purpose.

11.3 Implications for the Integrity of System Designs

System architects make extensive use of causality diagrams. Many instantiations have associated techniques that facilitate quantitative analysis of behaviour and performance. The author appreciates that causality diagrams provide great insight into the performance and behaviour of many applications, and that many practising architects have the knowledge and capability to add sufficient value to ensure integrity. In addition, many tools exist that populate semi-automated design environments that enable many aspects associated with feedback and recursive properties to be determined. Nevertheless, their intrinsic inability to cater for feedback (recursive) relationships is a profound limitation to their value in terms of the determination of structural and emergent properties of the system in its entirety. Therefore, the author contends that such methods fall well short of providing a comprehensive means of property determination.

The methods described herein provide the architect with information that has a substantive impact on the architect's understanding of the system structure. Complexity is a primary determinant of design cost and the role of any architect is to ensure that there is proper balance between the complexity of each building block and the complexity generated by creating more interconnectivity as a consequence of further building block decomposition into simpler functional entities.

While formal quantitative optimisation of architectural constructs is not part of this study, all system architects have a duty of care to optimise the structure in qualitative terms. Although the analytical methods enable precise determination of the architectural interconnectivity and functionality, it is intuitively obvious to any system architect that unfettered inclusion of building-block-to-building-block links is not to be encouraged. Therefore, at a practical level, system architects will assess the Functional Specification of each building block with respect to its complexity and consistency, together with its ICD (the Interface Connectivity Definition) document with respect to the number of interconnections, the complexity of each connection and its relevance to the architecture.

The role of the architect is to ensure that each separate connection has a proper role and that its required attributes can be defined in straightforward terms.

During the 1980s the software engineering community learned that spaghetti-like module interconnectivity led to unreliable designs; that is designs of indefinable or indiscernible emergent properties. Rules for module size and interconnectivity were defined. Dijkstra [25] provided formal methods that showed that robust constructions could be produced if each module had a single information flow entry point. Later, formal rules for termination, data flow and information flow led to robust implementation of properly defined requirements.

It is proposed that system architects need to create similar rules and clearly the work of the software engineering community provides them with a very good start. Perhaps the use of building blocks with single data and information entry points would help reduce complexity. Complexity will also be reduced by the use of building blocks that have been subjected to a process of precise functional checking and functional termination determination; both will help to ensure that 'rubbish' is not transmitted to other building blocks. There are many such rules of engineering design discipline that will improve the reliability of architectures.

11.4 Level of Achievement

The methods described in this thesis have been derived by the author to provide a structured means of implementing intuitive design methods that he used during his career as a system designer. Based on personal experience, the author believes that all successful architects work with a mental structure of how the system works as a deterministic machine system. These methods enable architects to elucidate architectural structures in a formal setting. The graphical form facilitates peer and specialist review processes. The functional analysis capability is limited only by the capability of the architect to represent functions in a form supported by algebraic rules of combination, (e.g. intersection, symmetry and association). Therefore it is postulated that an architect has the means of demonstrating the completeness of a design.

Each of the methods can be used independently of the others. Each provides valuable insight into the architectural structure. Nevertheless the architect requires a methodology that satisfies both necessity and sufficiency criteria to establish design robustness. Taken together the methods address the relationships between the ideal and implementation solutions in terms of decomposition, causality and functionality.

The author is unable to prove formally that the three methods together provide design sufficiency. Nevertheless, the three methods address the sufficiency requirements for the reconstruction of a system that has been subject to hierarchical decomposition, specifically:

- The extensions to the Design Matrix provide both a formal description of the decomposition and a means to ensure that the functional and implementation solutions are compatible.
- The causality network provides a formal description of the inter-functional relationships and a means to ensure that the functional and implementation solutions are compatible.
- The labelled direct product provides a complete description of all internal linkages and each label can be provided with functionality expressions for any viewpoint that is required to be addressed by the architect.

Research into viewpoint analysis is ongoing. However, many specialists have considered the quantitative question as to 'How many viewpoints need to be addressed?' without conclusion. With respect to a qualitative approach, a good architect will be able to determine a set of viewpoints that should be analysed; every architect has to exercise judgement as to the number and type of viewpoints that constitute a sufficient set for justification of any particular design application.

With respect to a quantitative approach to be provided by an architect, the assessment of Bond Graph and Pseudo Bond Graph capability has shown that it is possible to provide consistent component models for a wide range of disciplines. The sixteen disciplines, shown in Table 17, identified by the author encompass the field of applications that the author considers to be sufficient for the systems based on physical, chemical and informatic technologies.

The study has shown that there is considerable untapped potential for gaining quantitative insight into system architectures by combining abstract algebra, graph theory, linear algebra and functional analysis into a coherent methodology. It is envisaged that an algebra for architectural analysis can be developed. This study proposes a generic set of rules, written in English, which can be transposed into propositional calculus form using algebraic notation. The rules can then be refined as required.

The specific forms demonstrated include that of polynomial expression, 2-Port network analysis and the use of the Laplace transform for dynamic system analysis.

State variable methods using the standard form of expression can also be used to populate the function expressions. Investigations identified two constructions that enable state variable functions expressed in standard form to be combined for series and parallel functional constructions. However, although the principle of such state space based computations is already well established, the investigations showed that there is a

lack of generic Pseudo-Bond Graph component constructions. As these generic constructions need to be algebraically consistent to enable the end-to-end functionality to be constructed, and the consisted form was not identifiable, the author did not demonstrate the use of these constructions.

With respect to data capture, the thesis shows that the construction of the system N Squared Matrix in Excel is straightforward. The hierarchical decomposition was 'flattened' into a 2x2 matrix with rank equal to the sum of the ranks of the matrices that represent each layer. Associated spreadsheets were then programmed to generate the Adjacency Form required for path connectivity analysis. For the purposes of the demonstration herein, all path connections through each building block were enabled. For real systems however, not all connections will pass through each building block. The method has the potential to block functional interconnection. This can be mechanised by using an on/off data element that is associated with the function labels that modifies the adjacency matrix accordingly.

With respect to mechanisation of the analytical methods the author has used both Maple and Mathematica to effect the construction of the directed graphs and provide functional decomposition of end-to-end labelled paths. The use of rule based programming in Mathematica enables the manipulation of the terms of the direct product, particularly the interpretation of the association rule. These instructions in the form of rules enable the function expressions to be transformed into further, and generally more complex, functionality. Therefore, non-linear and multivariable analysis can be accomplished for identified path structures.

It is recognised that further difficulties of functional expression generation are associated with real time and timeline based systems. The computer-based systems specialists have developed many design support environments that simulate time line based applications and product solutions. There is no specific constraint to the incorporation of such results into direct product expressions; all that is required is specific cross-reference to the path combinations and routes that have been analysed.

Clearly, it may not be practical or economic to compile analytic expressions for all functions of interest. Nevertheless, the engineering management processes of identifying and honing down the number of expressions that require qualitative review provides added value as to the reduction of the risk of non-performance or non-compliance.

11.5 Limitations to Use

With respect to the utility of the proposals, it must be acknowledged that for centuries large-scale problems have been addressed by the use of hierarchical decomposition and reconstruction. To date, no limitations of the generic methods has been identified. However, the author recognises that the generation of architectural node and interrelationship functions in mathematical form may be beyond the capability of many systems engineers. Consequently, it is likely that practical considerations associated with functional representation and analysis will limit full explotation of the generic method.

Sadly, there are many examples of projects, from mainly anecdotal reports, that have used hierarchical decomposition techniques and run into trouble. Invariably, these troublesome applications have been the result of incompetent use of the generic method, particularly where practical considerations have imposed limitations on the quality of the formation of the decomposition hierarchy and content. The author is of the opinion that these limitations occur from lack of knowledge of the application, superficial consideration of the decomposition items, inadequate consideration of decomposition items in relation to reconstructability, and, sadly, ignorance.

All experience points to the fact that deterministic mechanisms are fully determinable. In making these assertions the author is aware that the determination of relevant component functionality is neither easy nor sometimes practicable. Component models tend to be of the 'black box' form; that is they provide representative behaviour and performance. While most component models are based on the underlying science or on experimental data, they are not necessarily complete. Therefore, during system integration, unpredicted system properties become apparent. Some architects argue that all human centred activities should be excluded from system design, simply because of the perceived difficulty of modelling human centric behaviour.

The author is of the opinion that this is erroneous thinking as there are many successful models of human behaviour and response. These include, for example, aircraft pilots, ships helmsmen, air traffic controllers, and many human centred transactional information response work places.

A part of the responsibilities to be discharged by the architect is to understand the limitations of the component models used to predict emergent properties. Then, either further data is sought to remove ambiguity or uncertainty, or the integration process will be designed to support specific investigations to determine the actual outcome. For example, flight test or ship sea trials provide a regime to establish actual behaviour and performance.

The scope of the approach presented in this thesis is necessarily limited by the ability of the architect to generate component models in the form suitable for functional analysis. Sometimes functionality may be described in English as a role or activity, or perhaps in abstract algebra form using set theory. This detracts from the ability of the method to provide source-to-sink quantitative estimates of performance. Nevertheless, the author has found that useful information may be obtained about the structure of the system, particularly when the functionality is described using sets.

With respect to the computational aspects of the methods developed herein, the impact of the scale of architecture structure descriptions induced by hierarchical decomposition of large scale systems has been addressed by creating 'flat' matrices with rank equal to the sum of the rank of the matrix of the structure at each level of decomposition. While this method results in matrices of large rank, the analysis tools (Excel and Mathematica) and computational power used to demonstrate the techniques developed herein, demonstrate that the methods are of practical use for large-scale system structures. The Excel column limit of 250 will restrict the size of flattened data entry matrices. However, this is unlikely to constitute a constraint for systems of practical size. Mathematica has sparse matrix functions that enable it to handle very large matrix based constructions. Also, since the author's programs using version 5.2 have an execution runtime of a few seconds, it is unlikely that the computational capacity of Mathematica will present a practical constraint.

12 RECOMMENDATIONS FOR FUTURE WORK

12.1 Data Capture

Data input to Excel spreadsheets to create the N Squared form and associated adjacency matrices has been presented in a straightforward form. However, the method will facilitate more complex analysis of path structures by modifying the adjacency matrix to include/exclude interrelationships. This can be mechanised by having an on/off key data input element for each building block that switches the building block relationships appropriately. Then the spreadsheet can be programmed to modify the adjacency matrix accordingly.

12.2 Use of Graph Theory and associated Mathematical Fields

With respect to graph theory, the thesis has shown that robust identification of path structures is practical, that unique labels can be attached to each path and that the functional structure associated with any path structure can be determined. Nevertheless, the thesis has not explored the scope of graph theory and its associated mathematical fields that could be used by systems architects to gain further insight into the characteristics of the structure. With respect to graph theory itself, for example, the use of cut-sets and colouration will aid evaluation of the construction, compatibility and consistency of viewpoints. In addition, there are substantial bodies of knowledge related to graph theory [43], for example, logic, linear algebra, geometry topology, networks and computing, all of which have potential for use in the context of architectural analysis. While the author would not expect architects, in general, to be able to exploit directly the capability, various analysis techniques could be incorporated into evaluation tools that could be applied directly by architects.

12.3 Use of Standard models of Functional Combination

With respect to functional population the thesis has shown that logical, polynomial and matrix forms can be used to characterise and quantify the functionality of path structures.

Network analysis is based on the generation of flow/impedance component models in conjunction with the general application of Kirchoff's Law. Two-Port analysis has been included as the method shows that functionalities of individual building blocks can be combined to provide path structure quantification. This demonstrates that standard forms of functional representation with specific rules of association and combination enable interconnectivity functionality determination.

State space representation was included in the expectation that its standard form and means of combination for series and parallel constructions could provide a means of

calculating end-to-end dynamic behaviour and performance. The thesis shows that the algebraic construction of such combinations is well understood. Nevertheless, the author concluded that, while the inclusion of such program fragments would add to the range of methods available to system architects, the development and inclusions of such constructions did not add to the essential level of achievement that is the main objective of this study. Also, the author concluded that the generation of Mathematica program fragments for such constructions was not trivial and that such constructions should be generated by experienced state variable method practitioners.

Bond Graph representation was included in the expectation that it would provide a ready means of standard component modelling and the means of algebraic association and combination to enable end-to-end determination of capacity and performance.

However, investigations to date show that the development of across and through variables is incomplete for many functionalities that the systems architect is likely to encounter, e.g. computer based systems components. In addition, component models in matrix form to facilitate model building need to be developed.

Furthermore, investigations to date show that while state variable constructions for Bond Graph models is mature, the specific problem of combining models of models, especially in matrix form does not appear to have been addressed by the research community.

12.4 Extensions to the Type Definition and Population of the Direct Product

The functional domain of the Direct Product is by definition not specific. All that is required is that the rules of type standardisation for each domain of interest enable functional combination. This implies that there is substantial scope for the development of domain components for technological and management domains. Such domains might include, for example, project management based tasks, support pipeline constructions, or operational scenarios. It is envisaged that many forms of component library components could be developed in a similar way that cell libraries have been developed for CAE/CASE/CAD/CAM systems.

12.5 Generation of Algebra of System Design

It is proposed that a specific algebra for the decomposition and quantitative determination of the direct product be developed.

12.6 Integration with Design Structure Matrix Methods and Procedures

Current interest into the use and development of DSM capability has focused on the development of product and work breakdown structures and schedules, especially for multi-layered complex systems and programmes, e.g. see www.adeptmanagement.com. In addition, for example, a construction, that combines the DSM with the Design Matrix has been proposed by Guenov and Barker, [115].

However, all these methods have yet to address the issue of quantitative population, and it is suggested that the DSM has the capability to host the method of functional analysis described in this thesis.

12.7 Optimisation Extensions

It is recognised that, as the method formalises the structure of system architectures, the constructions have the potential to support formal optimisation determination. For example, structural optimisation by interface type cluster analysis has been shown to be practical; see Ref. 2. This implies that there is substantial scope for optimisation methods to be developed. The author advises that while such studies will produce interesting results, the reality of architecting is that considerations of interactivity involve many domains including programme management and business capability factors. The consequence is that optimisation is likely to remain a human centred activity.

APPENDIX 1 – DESIGN PROCEDURE EVALUATION EXAMPLE

A1.1 Description of Evaluation Example (Attack Helicopter)

The exemplar for this research programme is that of an avionic systems design for a tandem cockpit configuration attack helicopter. The tandem configuration is that normally used for attack helicopters. Civil helicopters normally use a side-by-side configuration and this has the considerable benefit that both pilots can see and reach many of the controls and displays in the central section of the overall cockpit layout; it also facilitates natural communication and interaction. The tandem arrangement enables improved aerodynamic performance and provides a spatial allocation for fitting auxiliary wings (or lifting surfaces) and the carriage of external stores. The disadvantage is that both pilots operate in self-contained environments with the implication that all interactive tasks must be addressed and facilitated within the design of the avionic system.

Such helicopters are required to operate in both day and night all weather conditions. The two forms of flight mode are Visual Meteorological Conditions (VMC) and Instrument Meteorological Conditions (IMC) and the corresponding piloting modes are Visual Flight Rules (VFR) and Instrument Flight Rules (IFR).

When visibility from the cockpit is greater than 1000 feet altitude and one nautical mile, it is presumed that the aircraft can be flown without the aid of flight or navigation instruments. When visibility from the cockpit is less than either 1000 feet altitude or less than one nautical mile it is then presumed that the aircraft cannot be flown without the aid of flight or navigation instruments.

It is a principal of air-worthiness that in any situation the aircraft may continue to be flown safely following any single fault. Therefore, the cockpit control and display systems are configured to ensure that the pilot is able to maintain safe flight in the event of failure.

To achieve this the set of Reversionary Instruments provides the minimum information required to enable the aircraft to be flown and operated safely in the following conditions. These assume that basic navigation aids are available, and that the minimum visibility requirements enable a safe approach and landing to a recovery location to be effected; i.e. minimum cloud base of 150 feet and visibility of greater than 500 metres.

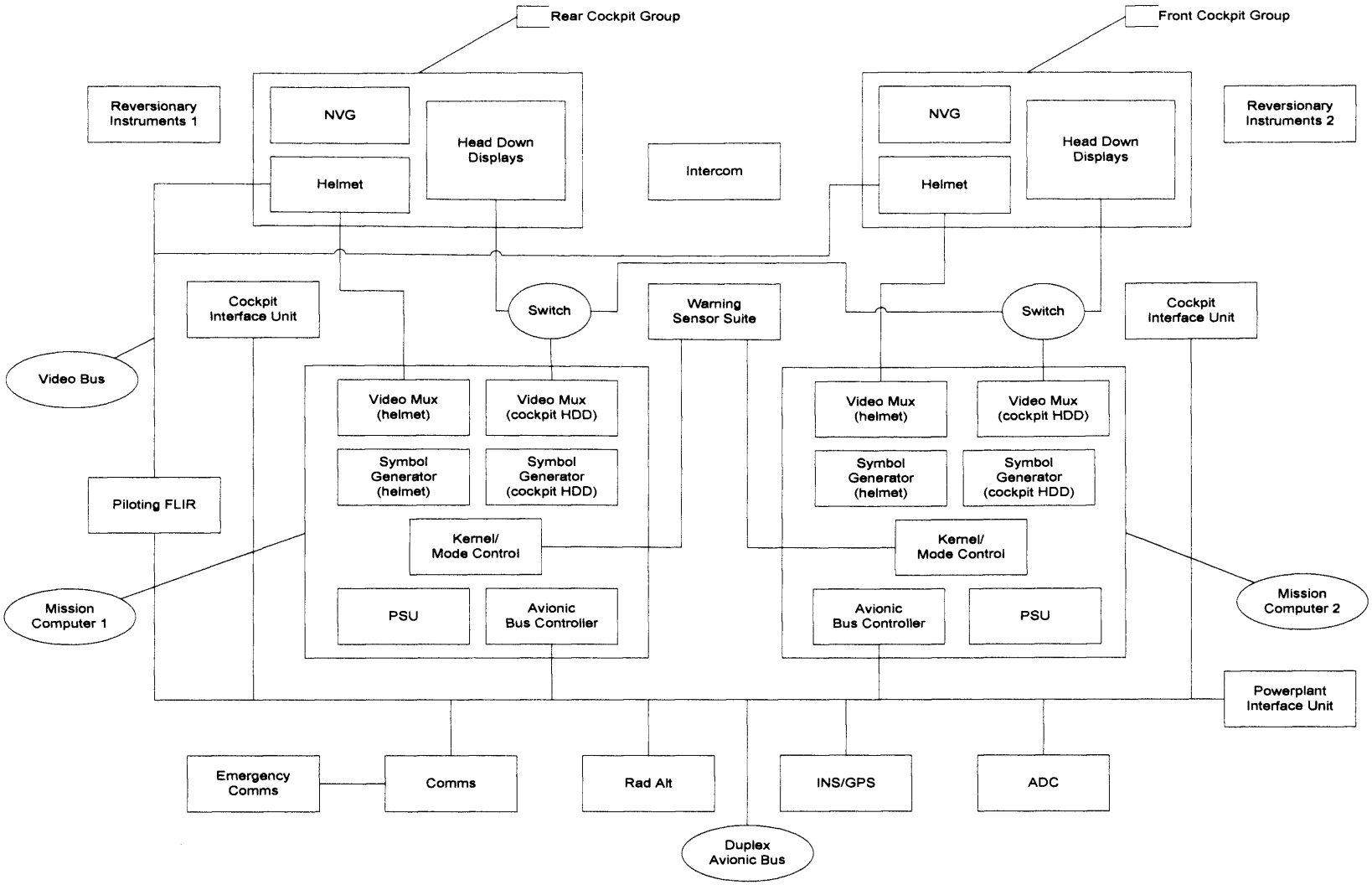


Figure 54 - Schematic of Attack Helicopter Avionic System (D20)

The Primary Instrument System is the set of integrated multi-function displays and avionic systems normally available to each pilot to enable the aircraft to meet its mission objectives including return/recovery to its designated operational base in IMC.

A schematic of the avionic system suite is shown in Figure 54.

A1.2 Operational Requirements – Failure Management Perspective

The single fault survival principle of airworthiness applies to all aspects of operations and the equipment required to 'move, fly and fight' the aircraft. For a day/night all weather capability the normal configuration for an attack helicopter is a twin cockpit two-crew tandem configuration. The two cockpits correspond to the roles of mission management and platform control. Normally the commander carries out the mission tasks – the 'fight' role, while the pilot flies the aircraft - the 'move and flight' role. However, the mission commander must be able to take full control of the aircraft from an un-attentive hands-off situation (normally specified as 5 seconds after the fault before intervention is allowed). Alternatively, the mission commander may view his own displays and use the intercom to provide the pilot with the flying cue information needed by the pilot to control the aircraft.

The Visual Piloting System is the group of sensor systems that provide information to each pilot to provide enhanced visual images of the cockpits field of view. This normally includes the following sensors.

Type 1 - Monocular piloting Forward Looking Infra-Red (FLIR) Camera system.

Type 1 - Monocular Low level Light TV (LLTV) (visual band).

Type 2 - Monocular targeting FLIR.

Type 2 - Stereoscopic Night Vision Goggles (NVG).

Type 1 sensors are those normally used to support piloting in limited visibility conditions; Type 2 sensors provide additional information particularly in the event of failure.

To enable these sensors to aid the pilot, the functional structure required to present the imagery in a form suitable to support piloting tasks is shown in Figure 55.

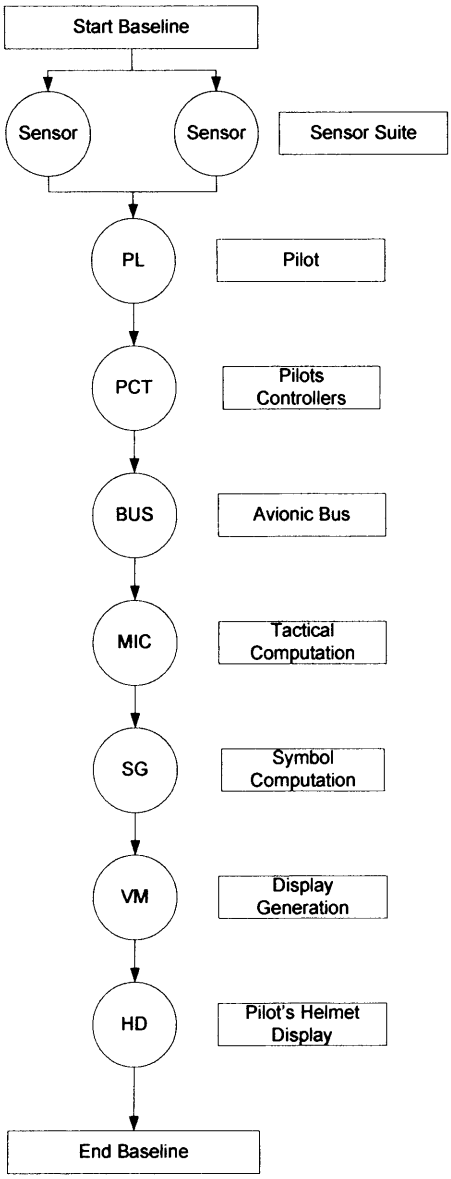


Figure 55 - Functionality required to present imagery to Pilot (D21)

The principle method of operation is for the pilot to scan continuously the piloting information displays and determine the validity of the information by consistency checking. In the event that erroneous data cues are suspected the pilot will give preference to the values shown on the reversionary instruments and make a request to the commander via the intercom for crosscheck data from the rear cockpit displays. On receiving the crosscheck data from the commander the pilot will determine which of the data displays is incorrect. This process is supported by the Continuous Built In Test (C-BIT) facilities incorporated into the individual displays and their associated sensors and processors. It is expected that the C-BIT coverage capability in each sub-system exceeds 90% of all potential output faults.

In order to maintain basic control of the aircraft under any conditions, it is necessary that the pilot has immediate visibility of attitude and heading displays. For power plant related data however, it is considered that it is sufficient for either crewmember to be able to see the displays and use the intercom to provide the pilot with the required parameter information.

For the attitude cues the low level (i.e. Height < minimum safety altitude) piloting requirements in IMC conditions require that the pilot has normal instrument data (attitudes, height, speed, torque) together with the FLIR generated vision system. Therefore in the event of failure of either the primary flight instrument data or primary PVIS information sets, then the flight mode will have to be aborted and the aircraft recovered to a safe operation condition. Further in the event of complete failure (i.e. both primary and secondary instrument or vision systems) then a safety critical condition will arise. Even in this condition though, as a result of the correlation between pitch attitude and speed with collective lever, and roll attitude and slip ball with compass heading, it is expected that, unless major un-commanded aircraft manoeuvres occur, the pilot will be able to maintain adequate control for a 'land as soon as practical' decision.

At night, the night vision goggles provide sufficient VMC equivalent performance to enable the aircraft to be flown safely in the event of failure of the main piloting vision system. However, the night vision goggles do not provide an effective reversionary capability during daylight as a result of luminous saturation of the photo tubes.

Therefore, it is a condition for safe flight that during either daylight or at night using the night vision goggles, the aircraft is clear of cloud and the pilot or commander is clear of ground with sufficient range to enable a recovery manoeuvre to be executed in accordance with minimum obstacle clearance reference region requirements.

For the navigation cues, the primary method of recovery from a navigation system failure in IMC is to use air traffic control to provide sufficient steering data so that a safe recovery may be effected by using only the basic heading, height, speed and elapsed time data from the reversionary instruments. Therefore, it is a condition of safe operation that the external communication system and the IFF/SSR Transponder have sufficient capability to interface with the air traffic control systems emergency position finding facilities.

A1.3 Decomposition of the Attack Helicopter System

In order to ensure that the analysis is in the desired context the system is decomposed into a three level hierarchy as follows.

Operational Context

Aircraft Platform

Avionic System

The family tree of the system is shown in Figure 56.

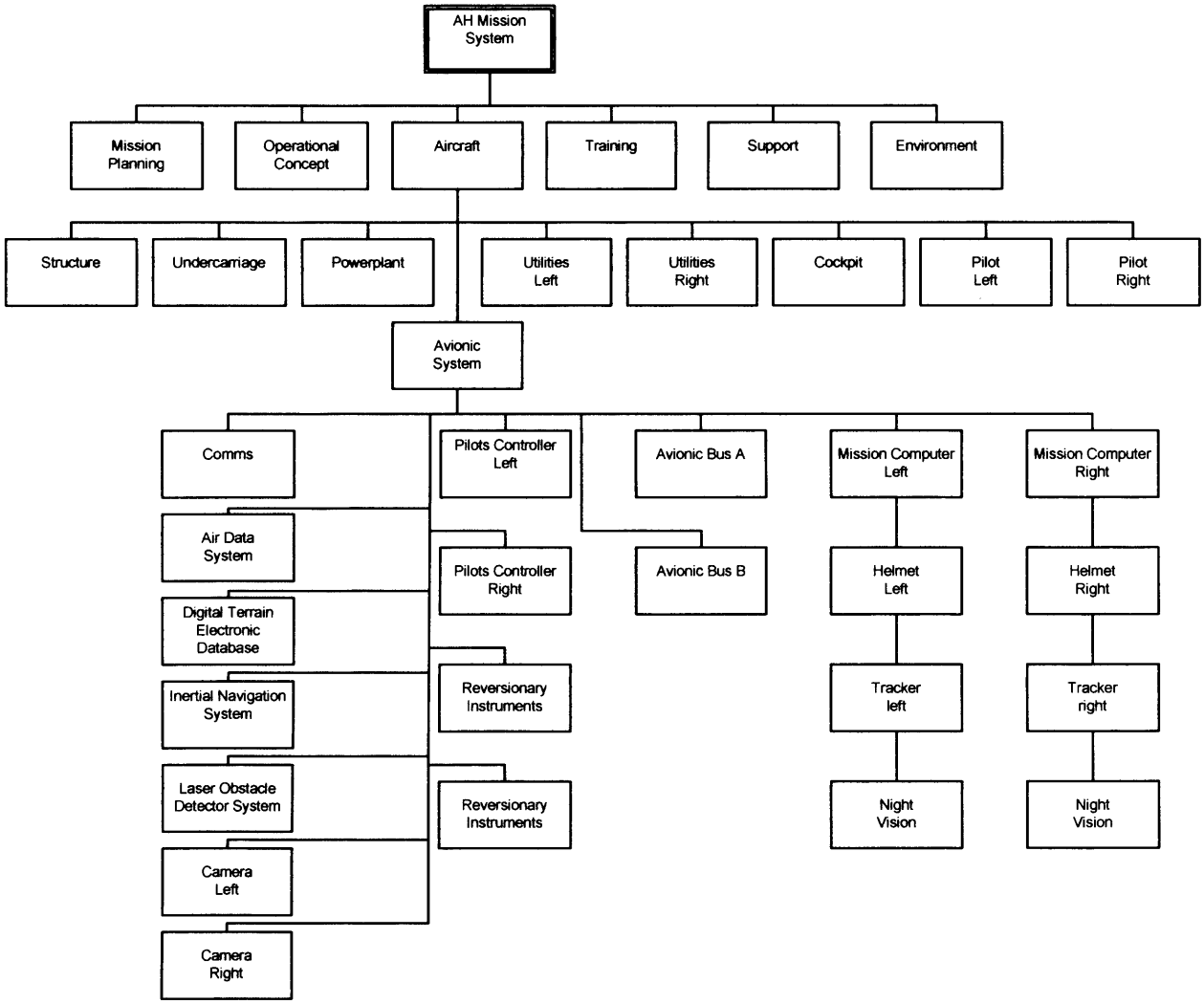


Figure 56 - Hierarchical Decomposition of System into Components (D12)

Nomenclature Table -

System Level	
Code	**Description**
M	Mission Planning
C	Operational Concept/Aircraft
T	Training
S	Support
E	Environment

Aircraft Level	
Code	**Description**
ST	Structure
UC	Undercarriage
PW	Powerplant
UTL	Utilities left
UTR	Utilities right
CP	Cockpit
PLL	Pilot left
PLR	Pilot right
AV	Avionic system

Avionic system	
Code	**Description**
COMS	Communications System
ADS	Air data System
DTED	Digital terrain database
INS	Inertial navigation
LOD	Laser obstacle detector
CAML	Camera left

Avionic system	
Code	**Description**
CAMR	Camera right
PCTL	Pilots controller left
PCTR	Pilots controller right
BUSA	Avionic Bus A
BUSB	Avionic Bus B
MICL	Mission Computer left
MICR	Mission Computer right
HELL	Helmet left
HELR	Helmet right
TRKL	Tracker left
TRKR	Tracker right
NVGL	Night Vision Goggles left
NVGR	Night Vision Goggles right
RISL	Reversionary instruments left
RISR	Reversionary instruments right

Table 20 - System Nomenclature

A1.4 Evaluation of System Installation Implementation

A1.4.1 Physical Installation

The normal arrangement is as follows: -

The sensors are positioned to obtain the best measurement accuracy for the specific aircraft. For example the FLIR may be adjacent to the cockpit windshields (without encroaching on visual geometry), although modern machine rotor heads now incorporate a sensor platform.

The mission computers and supporting electronic units are housed in avionic equipment racks. At least two racks are provided so that duplicate installation redundancy is obtained wherever required.

Units that are specific to the human interface with the crew are installed in the cockpit(s). The configuration selected for this study is that of a two-crew tandem arrangement; therefore there are two separate cockpit installations.

A1.4.2 Electrical Installation

The normal arrangement is for power to be provided as 28 Volt dc, as follows:-

The power electrical system is normally based on two separate bus bar systems each with its own combined generator-rectifier unit. In addition, an Essential Equipment bus system is provided that is connected to both generators together with a battery-charger unit that will provide electrical power for up to about 20 minutes in the event of dual generator failure.

Distribution segregation is maintained for the two racks and sets of avionic equipment. However, essential units will be powered from the essential bus as appropriate.

A1.4.3 Analysis of Alignment of Functional and Installation Architectures

The generic form of the system arrangement is shown in Figure 57.

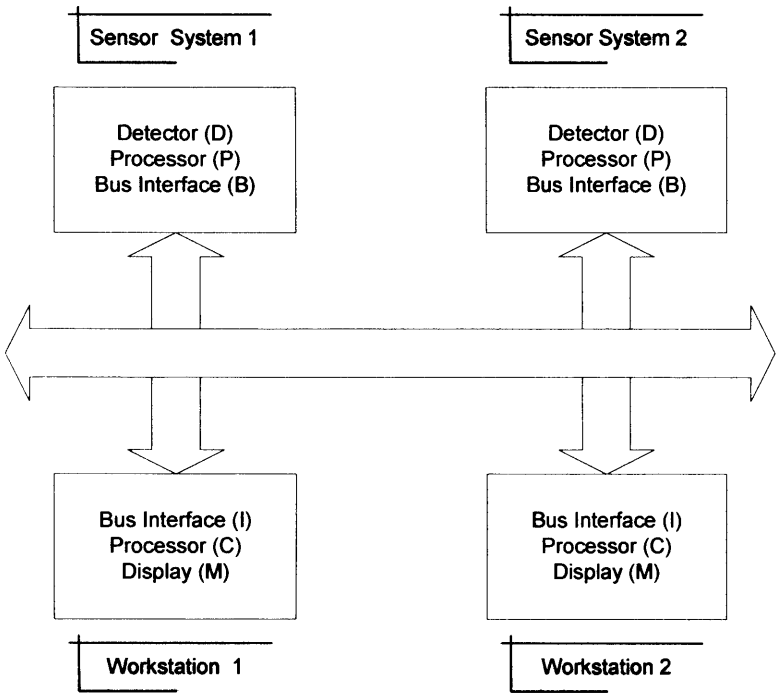


Figure 57 - Schematic of System Architecture (D8)

To demonstrate the use of the Dependency Matrix construction, a simplified form of the system installation in the helicopter is used.

To obtain the highest availability from the installation arrangement, the requirement is to ensure full mechanical and electrical isolation between the individual sensors, the mission processors and the display systems.

However, in the real arrangement, even though the sensors are 'smart', some signal processing associated with each sensor has been physically located within the mission computers. Then electrical power is supplied to the sensor units from individually isolated star points, and separate star points are used to support each avionic equipment rack. The ideal and realistic mechanical and electrical installations are shown in Figure 58 and Figure 59 respectively.

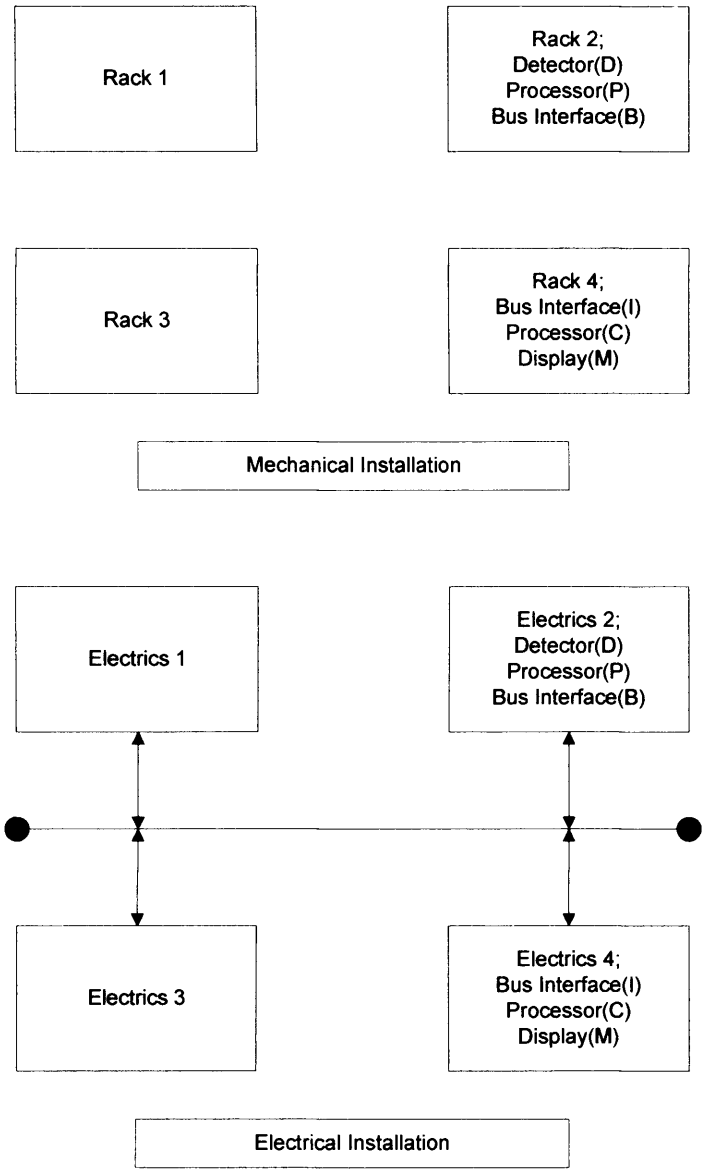


Figure 58 - Schematic of Ideal Mechanical and Electrical Installation Arrangement (D13)

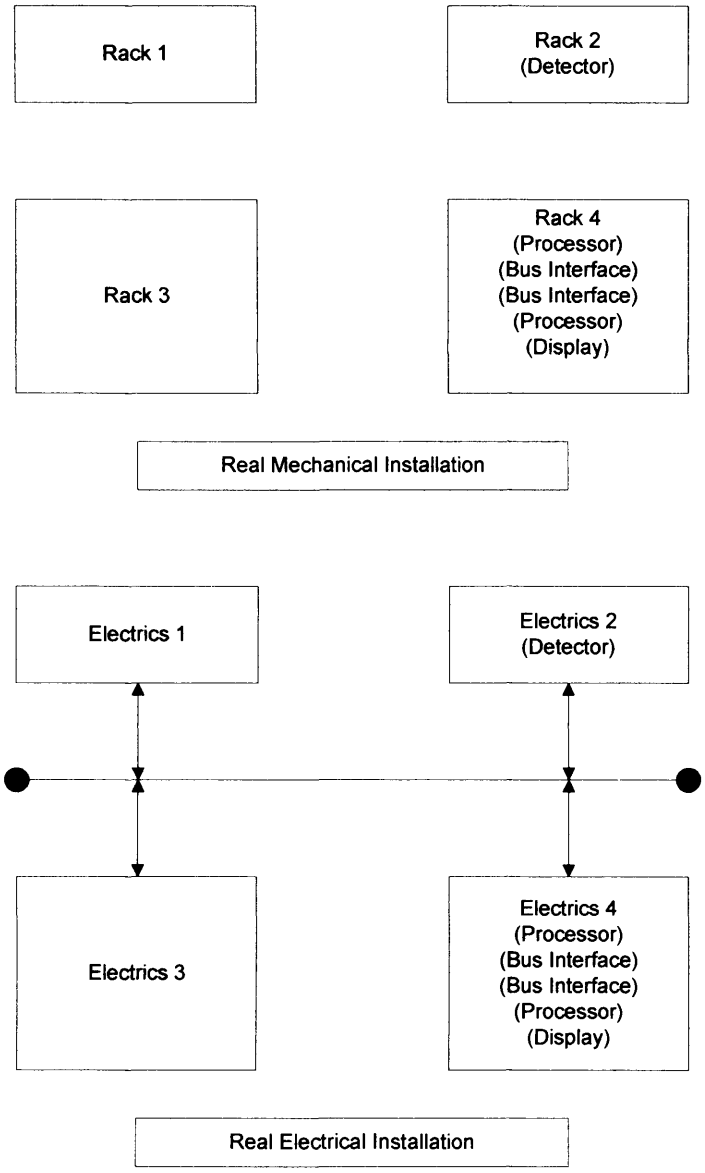


Figure 59 - Schematic of Real Mechanical and Electrical Installation Arrangement (D14)

The intuitive engineering implication is that the signal processing building blocks associated with each sensor are intrinsically connected to the mission computer in which they are installed. The potential practical impact is that electrical noise and additional earth loops will impact the quality of both analogue and digital signals, effectively reducing the signal to noise ratio (and the detection sensitivity) of the system.

The objective of the construction is that the dependency analysis should show up the additional (unwanted) functionality that is inherent within the real arrangement. Building block and design space connectivity is the aspect of interest. Therefore, the matrices have been constructed on the basis of connectivity; 1 means a connection and 0 means no connection.

Using the Boolean form of the Design Matrix construct the A' and B' matrices are constructed as follows.

(Note. For the general case of two matrices **A** (m rows, p columns), **B** (p rows, n columns) each element Cij of the matrix product **AB** (m rows, n columns) is the sum of k=1 to p of the Boolean product $\mathbf{A}_{ik} \times \mathbf{B}_{kj}$ for i = 1 to m and j = 1 to n.)

$$\begin{bmatrix} F1 \\ F2 \\ 0 \\ 0 \end{bmatrix} = A' = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} W1 \\ W2 \\ S1 \\ S2 \end{bmatrix};$$

$$B' = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} D1 \\ P1 \\ C1 \\ M1 \\ D2 \\ P2 \\ C2 \\ M2 \end{bmatrix}; \text{ Equation A1.1}$$

The Building Block/Design Space characteristic matrix C_I' shows the ideal relationships as follows.

W1 depends on R1, R2, R3, E1, E2, E3

W2 depends on R1, R2, R4, E1, E2, E4

S1 depends on R1, E1

S2 depends on R2, E2

Therefore

$$C_I' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \\ E1 \\ E2 \\ E3 \\ E4 \end{bmatrix}; \quad \text{Equation A1.2}$$

Therefore

$$\{FR_I\} = [A'][B'][CI'] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \\ E1 \\ E2 \\ E3 \\ E4 \end{bmatrix}; \quad \text{Equation A1.3}$$

Similarly C_R' shows that the real relationships are:-

W1 depends on R1, R2, R3, R4, E1, E2, E3, E4

W2 depends on R1, R2, R3, R4, E1, E2, E3, E4

S1 depends on R1, R3, E1, E3

S2 depends on R2, R4, E2, E4

Therefore

$$C_R' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \\ E1 \\ E2 \\ E3 \\ E4 \end{bmatrix}; \quad \text{Equation A1.4}$$

Therefore

$$\{FR_R\} = [A'][B'][CR'] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \\ E1 \\ E2 \\ E3 \\ E4 \end{bmatrix}; \text{ Equation A1.5}$$

Therefore

$$\{FR_I\} - \{FR_R\} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R1 \\ R2 \\ R3 \\ R4 \\ E1 \\ E2 \\ E3 \\ E4 \end{bmatrix}; \quad \text{Equation A1.6}$$

Subtraction of the ideal and real matrix products identifies the additional or missing relationships generated by the real design. The construction clearly shows that by re-locating the sensor co-processors P1, P2 into racks R3, R4 the characteristics of the workstations now have dependencies which reflect the change in racking and their connection to the source of electrical power. The construction also shows the change in the dependencies on the sensors themselves.

A1.5 Evaluation of Compatibility of Functional and Implementation Causality Architectures

A1.5.1 Functional Causality Determination

To demonstrate the use of the resource enhanced causality network to determine the compatibility of the functional and implementation architectures, a generic form of the computing structure in the Command and Control system has been derived.

To achieve all weather flight capability with low altitude combat effectiveness, the information must be provided to the pilot in an 'eyes out' mode. The generic form is for sensor data to be refined, transported, combined with other data and portrayed on each pilots helmet display.

The functional construction of the computing structure is shown in Figure 60.

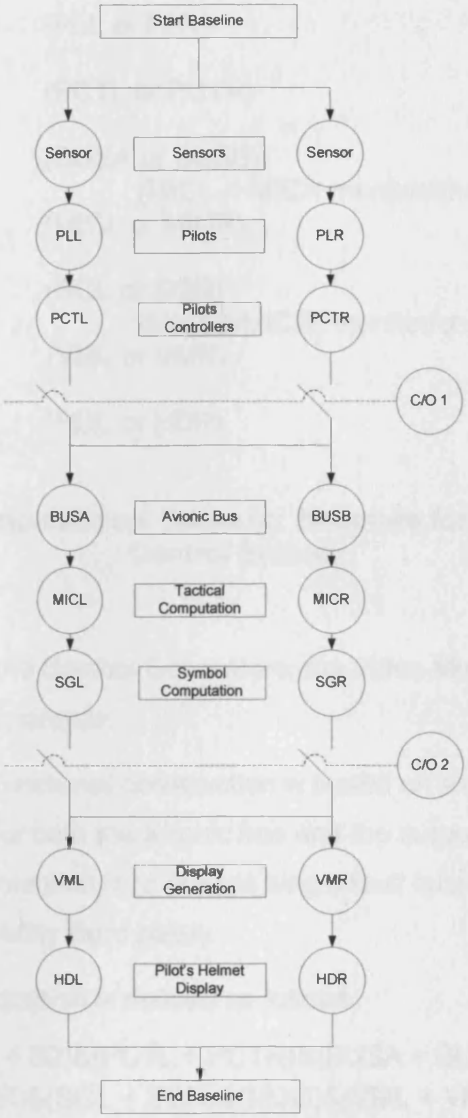


Figure 60 - Functional Structure of Avionic System Computing Structure (D15)

For the Command and Control System information set to be healthy the availability logic for a typical sensor to display causality structure is shown in Figure 61.

(Note. A detailed description of the information required for safe flight is provided in the Addendum to Appendix 1.)

Sensor:	(Generic)
AND	
Command:	(PLL or PLR)
AND	
Pilots Controller	(PCTL or PCTR)
AND	
Avionic Bus:	(BUSA or BUSB)
AND	(MICL ->MICR monitor/changeover healthy)
Tactical Analysis	(MICL or MICR)
AND	
Sym Gen:	(SGL or SGR)
AND	(MICL->MICR) monitor/changeover healthy)
Video mix:	(VML or VMR)
AND	
Display:	(HDL or HDR)

Figure 61 - Typical Computational Causality Structure for Avionic Command and Control System

Note. In this construction the Symbol Generators, the Video Multiplexor and the Display are generic functional components.

This logic shows that the functional construction is based on a duplex arrangement with lane changeover options for both the avionic bus and the output to the video multiplexers. The design intention is to provide single fault failure survival capability to facilitate low-level, low visibility flight safety.

The functional causality equation is derived as follows;

$$H = (S1 + S2)\&(PCTL + PCTR)\&(BUSA + BUSB)\&(BCO)\&(MICL + MICR)\&(SGL + SGR)\&(MCO)\&(VML + VMR)\&(HDL + HDR)$$

Equation A1.7

where H defines system health, and BCO and MCO refer to the Avionic Bus and Mission Computer monitor/changeover logic switch, respectively.

A1.5.2 Description of System Implementation

In order to minimise the number of installed items in the vehicle the current approach to implementation is to integrate as much functionality into single units as is possible; a philosophy that is very well supported by the progressive miniaturisation of digital electronics.

Therefore, the design of modern mission computers has many features that are shared by general-purpose computers; these include multiplexed input/output, a common digital highway that links all computation processors, non-specific processors and separate graphics processing. A schematic of the main implementation features of a modern mission computer is shown in Figure 62.

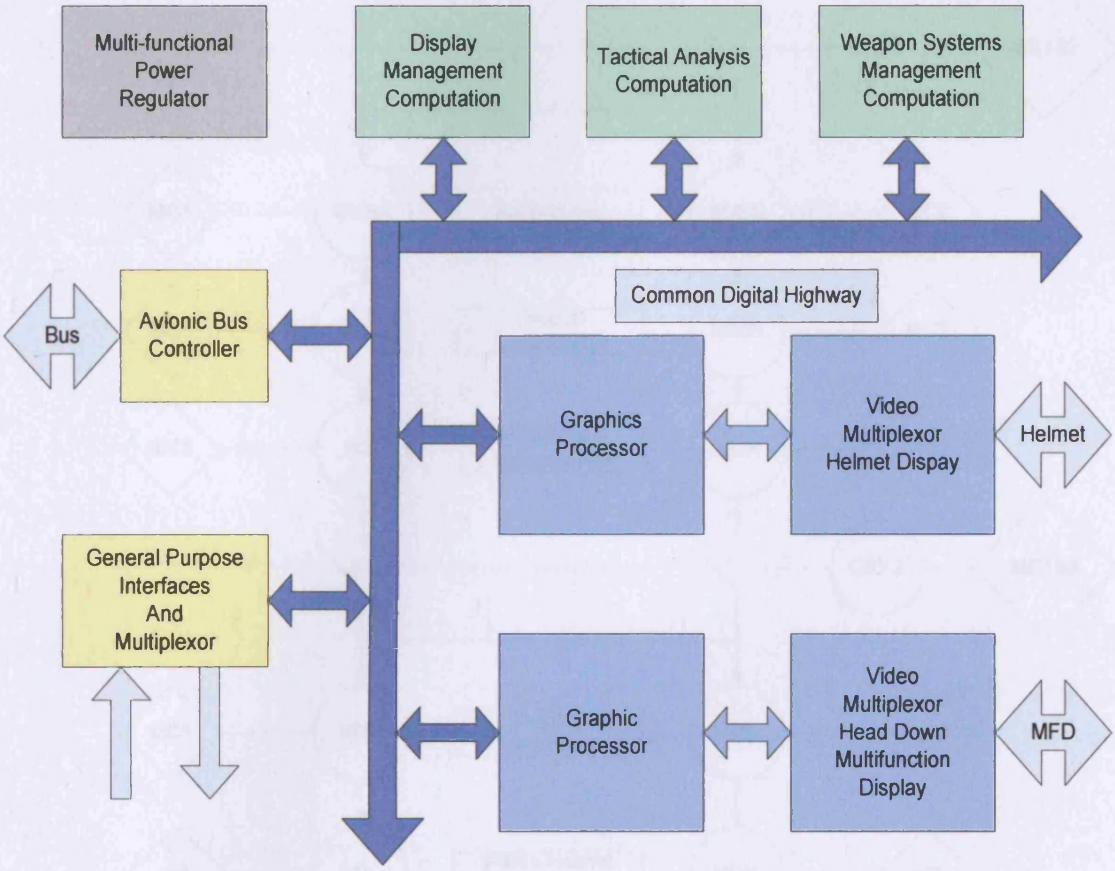


Figure 62 - Implementation Arrangement of Mission Computer (D16)

It is apparent that the internal arrangement implies substantial interdependency between all of its internal sub-units.

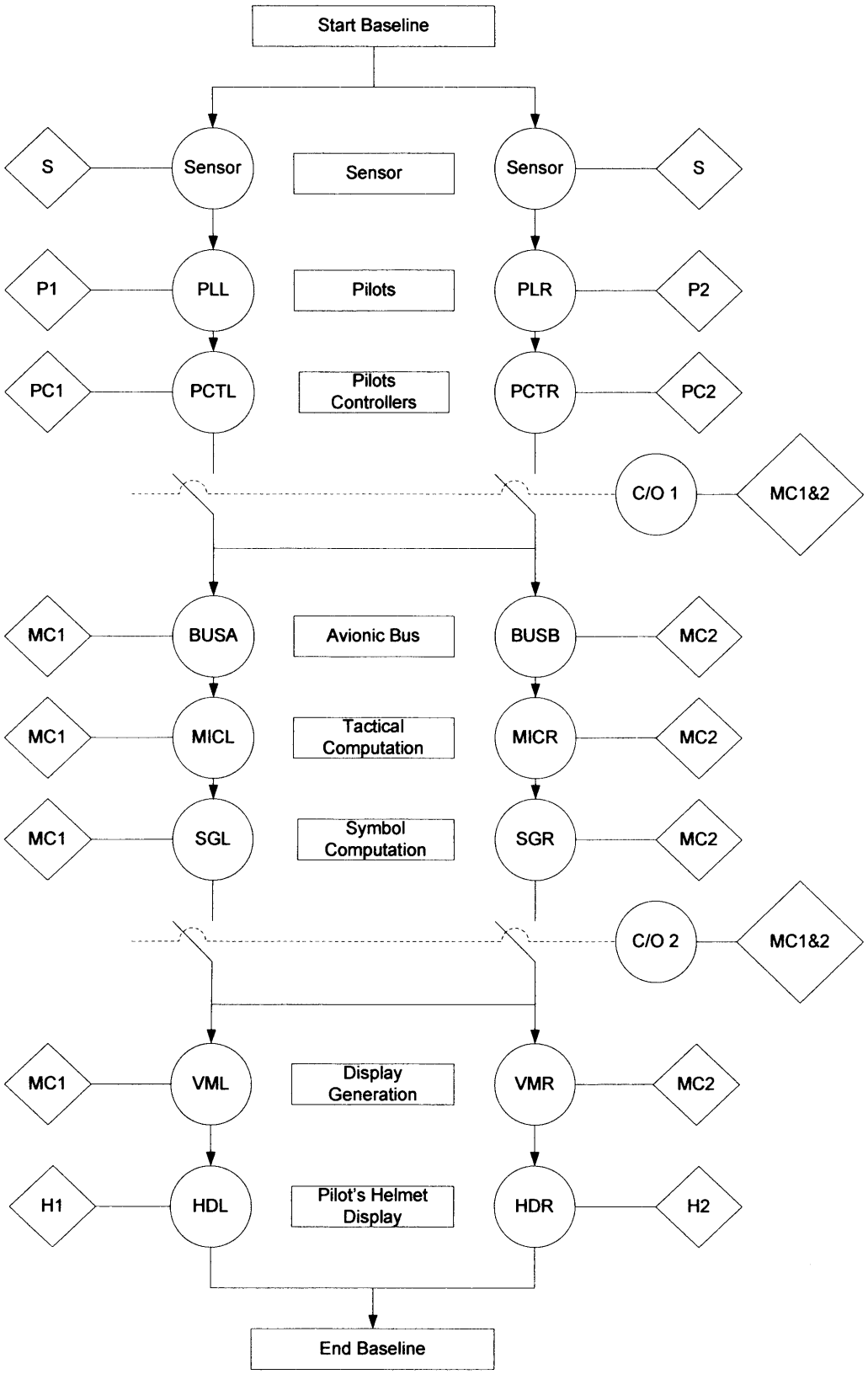


Figure 63 - Functional Causality Structure with Implementation Resources (D17)

A1.5.3 Compatibility of Ideal System and Implementation System Arrangement

To demonstrate the impact of the implementation arrangement of the functional causality structure, Figure 63 shows each functional component linked to its implementation resource.

The causality equation, Equation A1.7, is then modified by substituting each function with its matching implementation resource to yield Equation A1.8, as follows.

$$H = (S + S)\&(PCTL + PCTR)\&(MC1 + MC2)\&(BCO)\&$$
$$(MC1 + MC2)\&(MC1 + MC2)\&(MCO)\&(MC1 + MC2)\&(H1 + H2)$$

Equation A1.8

(Note. The pilots are presumed to be logically True in this expression.)

The equation is in logical form and is simplified to yield Equation A1.9.

$$H = S\&(PCTL + PCTR)\&(MC1 + MC2)\&(BCO)\&(MCO)\&(H1 + H2)$$

Equation A1.9

It can be seen that the implementation has modified the causality in terms of

- A single sensor.
- Separate pilot's inceptors.
- Joint mission computers.
- Separate pilot's helmet displays.
- Common components to provide avionic bus and mission computer automatic health monitoring and changeover to improve system availability in the event of failure.

The overall system availability can be determined by substituting probability values for the 'health' of each component.

A1.6 Functional Analysis

A1.6.1 Generation of the N Squared Form

The overall system has a three-tiered hierarchy, so the construction should consist of three matrices, one for each layer. The matrix for each layer contains its sub-system components laid out on the diagonal.

The order in which the sub-systems for each layer are allocated in each diagonal is discretionary and, in this case, the order has been chosen from the perspective of causality, left to right.

M	0	0	0	0
0	T	0	0	0
0	0	C	0	0
0	0	0	S	0
0	0	0	0	E

Table 21 - Decomposition of Top-level System of Family Tree

ST	0	0	0	0	0	0	0
0	UC	0	0	0	0	0	0
0	0	PW	0	0	0	0	0
0	0	0	UTL	0	0	0	0
0	0	0	0	UTR	0	0	0
0	0	0	0	0	CP	0	0
0	0	0	0	0	0	PLL	0
0	0	0	0	0	0	0	PLR

Table 22 - Decomposition of Aircraft Level of Family Tree

COMS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	ADS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	DTED	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	INS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	LOD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	CAML	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	CAMR	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	PCTL	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	PCTR	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	BUSA	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	BUSB	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	MICL	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	MICR	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	HELL	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	HELR	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRKL	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TRKR	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NVGL	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NVGR	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RISL	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RISR

Table 23 - Decomposition of Avionic System Family Tree

Clearly, there are relationships between sub-system components at different levels; for example, that of physical environment affects system components at all three levels. So the three matrices are related so that they form a tensor of Rank 3.

This is not easy to analyse so a matrix of Rank 2, as shown in Figure 64, is used constructed from the nine matrices needed to ensure that all potential relationships are represented.

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

Figure 64 Layout of N Squared Matrix for system with three hierarchical tiers

Domain	Excel Reference	Mathematica Reference
System	Helo N2 Msr	Helo Msr
Composition	Helo N2 Mr1,	Helo Mr1a
Capacity	Helo N2 Mr2e, Mr2w	Helo Mr2e, Helo Mr2w
Message	Helo N2 Mr3	Helo Mr3
Behaviour	Helo N2 Mr4a	Helo Mr4a

Table 24 - List of Excel and Mathematica programs to support analysis of Attack Helicopter Exemplar

For each domain, its architectural structure in matrix form is created in Excel Sheet 1. The sub-systems are shown on the diagonal and the off-diagonal elements are annotated with a True (1) to show that a structural relationship exists between two sub-systems. The associate matrix, shown in Excel Sheet 2, shows the adjacency matrix with the diagonal elements replaced with the functional label in each sub-system that defines its structural functionality. To enable the labelled adjacency matrix to be input to Mathematica for analysis it is stored in Tab Delineated form. Then Mathematica is programmed to input this form for analysis.

The analysis program for each domain has been constructed to show:

1. The links within the structural arrangement.
2. The functionality of the paths up to 3 steps.

The directed graphs show the edges that link the sub-systems; two are shown, the second rotated by 90 degrees to show the sub-systems obscured by the graph drawing program.

The graphs clearly show the structural connectivity through the arrangement. It can be seen that, by inspection of the graph, up to five steps of connectivity may be involved in any path.

The functionality of the system is shown by the substitution of the f 'n' functions that describe the structural functionality of each sub-system component. Therefore, the overall functionality of the interconnected system is demonstrated by taking the sum of the functionality of one to five steps to show the composite 'end-to-end' transfer function.

Numerical substitution of functional expressions and arbitrary values completes the demonstration of enumeration for each domain.

A1.6.2 Architecture Overview

The relationships between the components have been determined from experience of such systems and been captured into an N Squared Matrix held in an Excel spreadsheet (Helo N2 Msr). The spreadsheet has been programmed so that it automatically compiles the system functional matrix for input to Mathematica for analysis.

The system interconnectivity is shown in the form of a directed graph in Figure 65.

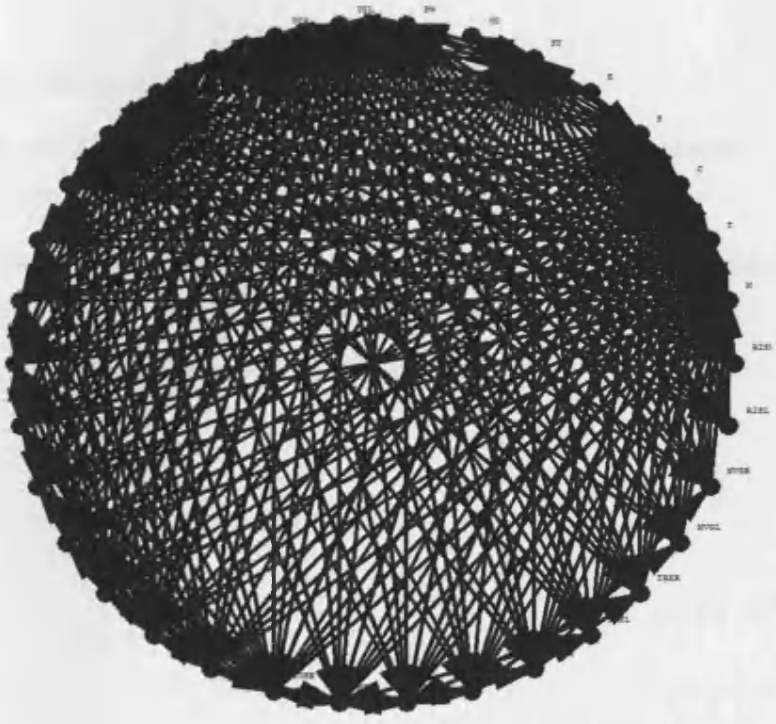


Figure 65 - System Architecture Connectivity (Helo Msr)

The Mathematica programs have been written so that the system connectivity is presented as directed graphs. These connectivity diagrams have been constructed so

that they present the system components (nodes) equally spaced around the rim of a circle. Then each arrowed line depicts the existence of a functional relationship between the designated nodes; each line represents a unidirectional 'from-to' information transfer relationship with the arrow head drawn at the 'to' end of the line.

This diagram shows the combined connectivity of all the viewpoints based on functional relationships that must be addressed by the design team; these include hardwired building block interconnections, functional relationships with the physical environment and relationships with the operational and support infrastructure.

The complexity of the system is immediately apparent. Further, the actual complexity is even greater as many of the connections are multivariable.

Clearly, the architect is able to review these interconnections in a qualitative manner. This has the benefit of providing a structured description and means of review of the interconnections that need to be evaluated by the design team. However, the complexity that is apparent implies that substantial partitioning is required in order to identify the connectivity associated with different viewpoints. Therefore, the next step is to address the system structure in terms of the attributes of composition, capacity, messenger and functionality.

A1.6.3 System Composition

A sub-set of the system has been compiled by the author, that represents interrelationships that are dependent on materials or substances.

It has been captured into Excel 'Helo N2 Mr1' and imported into Mathematica Program 'Helo Mr1a' from which the following assessment has been generated.

The composition structure is shown in Figure 66.

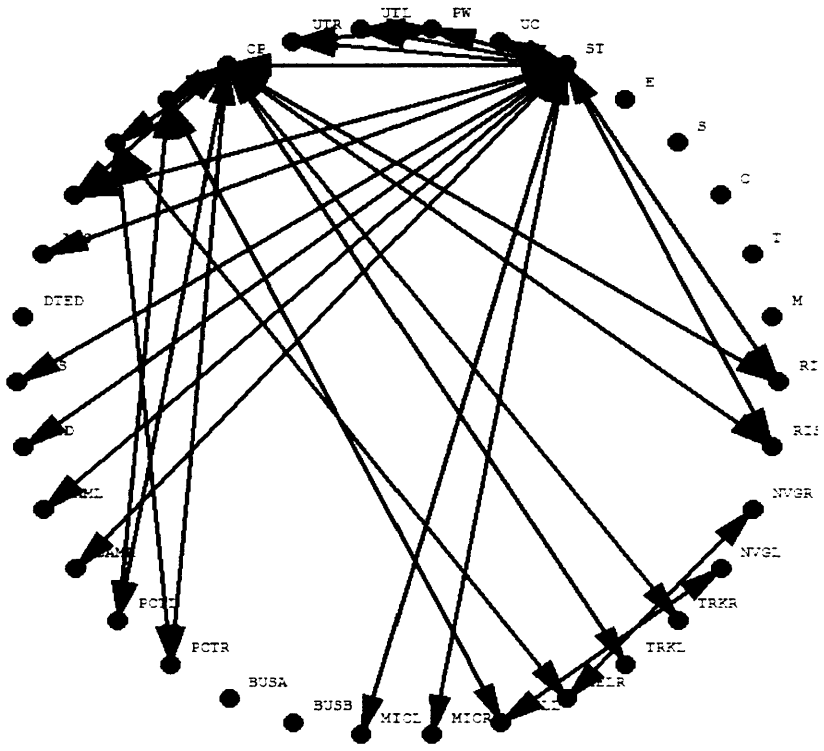


Figure 66- Graphical Representation of System Composition Viewpoint (Helo Mr1a)

This representation provides the architect with an uncluttered picture of the systems material interconnectivity. The interconnectivity relationships show where each pair of nodes are joined by a hard mechanical link. This partition makes it easy to assimilate. However, it belies the actual complexity of the construction.

The complexity is revealed by consideration of the number of distinct graphs that may be derived from the viewpoint.

Number of Graphs
3
218
1540944
1793359192848
34126043195297258035 2

Figure 67 - Extract of Number of Graphs for System Composition Viewpoint (Helo Mr1a)

The extract shows the number of 2, 4, 6, 8, 10 step graphs that can be obtained. The growth in the perceived complexity is apparent.

Number of Paths from Helo Structure Vertex - 1 step

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}

Number of Paths from Helo Structure Vertex - 2 steps

{1,1,1,1,1,1,1,1,1,1,1,2,3,15}

Number of Paths from Helo Structure Vertex - 3 steps

{1,1,3,3,4,4,4,4,10,15,15,15,15,15,15,15,15,17,17,17,18,18,18,24}

Number of Paths from Helo Structure Vertex - 4 steps

{1,1,4,4,10,10,10,10,10,10,10,10,24,24,27,27,28,28,29,29,34,34,34,44,86,249}

Number of Paths from Helo Structure Vertex - 5 steps

{4,4,30,30,86,86,115,115,118,118,249,249,249,249,249,249,249,249,293,293,303,335,335,335,
366,513}

Number of Paths from Helo Structure Vertex - 6 steps

{30,30,122,122,366,366,366,366,366,366,366,366,513,513,631,631,658,658,669,669,879,879,87
9,952,2009,4399}

Figure 68- Number of n-step paths that link the Structure Vertex to other vertices, shown in ascending order (Helo Mr1a)

Columns 1-13

M	0	0	0	0	0	0	0	0	0	0	0	0	0
0	T	0	0	0	0	0	0	0	0	0	0	0	0
0	0	C	0	0	0	0	0	0	0	0	0	0	0
0	0	0	S	0	0	0	0	0	0	0	0	0	0
0	0	0	0	E	0	0	0	0	0	0	0	0	0
0	0	0	0	0	{ST - 30 x}	0	{4 x}	{2 x}	{2 x}	{6 x}	{2 x}	{2 x}	
0	0	0	0	0	0	{UC - 2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	{4 x}	{2 x}	{PW - 6 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	{2 x}	{2 x}	{2 x}	{UTL - 4 x}	{4 x}	{2 x}	0	0	
0	0	0	0	0	{2 x}	{2 x}	{2 x}	{4 x}	{UTR - 4 x}	{2 x}	0	0	
0	0	0	0	0	{6 x}	{2 x}	{2 x}	{2 x}	{2 x}	{CP - 20 x}	{2 x}	{2 x}	
0	0	0	0	0	{2 x}	0	0	0	0	{2 x}	{PLL - 6 x}	{2 x}	
0	0	0	0	0	{2 x}	0	0	0	0	{2 x}	{2 x}	{PLR - 6 x}	
0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	{2 x}	0	0	0	0	{2 x}	{2 x}	{2 x}	
0	0	0	0	0	{2 x}	0	0	0	0	{2 x}	{2 x}	{2 x}	
0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0	
0	0	0	0	0	0	0	0	0	0	{2 x}	0	0	
0	0	0	0	0	0	0	0	0	0	{2 x}	0	0	
0	0	0	0	0	{2 x}	0	0	0	0	0	{2 x}	{2 x}	
0	0	0	0	0	{2 x}	0	0	0	0	0	{2 x}	{2 x}	
0	0	0	0	0	0	0	0	0	0	0	{2 x}	0	
0	0	0	0	0	0	0	0	0	0	0	0	{2 x}	
0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	
0	0	0	0	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	

Figure 69 - System with Composition Functionality using Variable x with Plus Rule of Association; 2 steps. Columns 1-13 (Helo Mr1a)

Columns 14-24

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{2 x}	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{2 x}	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{2 x}	0	0
{COMS - 4 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0
{2 x}	{ADS - 2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
0	0	DTED	0	0	0	0	0	0	0	0
{2 x}	{2 x}	0	{INS - 2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{LCD - 2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{CAML - 2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{CAMR - 2 x}	0	0	0	0
{2 x}	0	0	0	0	0	0	{PCTL - 4 x}	{2 x}	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{PCTR - 4 x}	0	0
0	0	0	0	0	0	0	0	0	BUSA	0
0	0	0	0	0	0	0	0	0	0	BUSB
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0
0	0	0	0	0	0	0	{2 x}	0	0	0
0	0	0	0	0	0	0	0	{2 x}	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{2 x}	0	0
{2 x}	0	0	0	0	0	0	{2 x}	{2 x}	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
{4 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0
{4 x}	{2 x}	0	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	{2 x}	0	0

Figure 70 - System with Composition Functionality using Variable x with Plus Rule of Association; 2 steps. Columns 14-24 (Helo Mr1a)

Columns 25-34

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	{2 x}	{2 x}	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	{2 x}	{2 x}	0	0	0	0	{2 x}	{2 x}
0	0	0	0	{2 x}	{2 x}	{2 x}	0	{2 x}	{2 x}
0	0	0	0	{2 x}	{2 x}	0	{2 x}	{2 x}	{2 x}
{2 x}	{2 x}	0	0	{2 x}	{2 x}	0	0	{4 x}	{4 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
0	0	0	0	0	0	0	0	0	0
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
0	0	{2 x}	0	{2 x}	{2 x}	0	0	{2 x}	{2 x}
0	0	0	{2 x}	{2 x}	{2 x}	0	0	{2 x}	{2 x}
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
{MICL - 2 x}	{2 x}	0	0	0	0	0	0	{2 x}	{2 x}
{2 x}	{MICR - 2 x}	0	0	0	0	0	0	{2 x}	{2 x}
0	0	{HELL - 4 x}	0	0	0	0	0	0	0
0	0	0	{HELR - 4 x}	0	0	0	0	0	0
0	0	0	0	{TRKL - 2 x}	{2 x}	0	0	{2 x}	{2 x}
0	0	0	0	{2 x}	{TRKR - 2 x}	0	0	{2 x}	{2 x}
0	0	0	0	0	0	{NVGL - 2 x}	0	0	0
0	0	0	0	0	0	0	{NVGR - 2 x}	0	0
{2 x}	{2 x}	0	0	{2 x}	{2 x}	0	0	{RISL - 4 x}	{4 x}
{2 x}	{2 x}	0	0	{2 x}	{2 x}	0	0	{4 x}	{RISR - 4 x}

Figure 71 - System with Composition Functionality using Variable x with Plus Rule of Association; 2 steps. Columns 25-34 (Helo Mr1a)

A different viewpoint is generated by consideration of the number of graphs that emanate from a single vertex. For the purposes of demonstration the Vertex entitled "Structure" is used as the source vertex. Figure 68 shows the number of n-step graphs, up to 6 steps, that exist between the Structure node and other vertices in the system. Again, the growth in complexity is apparent.

To determine the functionality between any source and sink node, recall that each vertex has been provided with a function label. Also assume that each function has a common variable x. Then, assuming that the rule of functional association is **Plus**, Figure 69, Figure 70 and Figure 71 show, in matrix form, the internode functionality for all two step relationships in the composition domain.

For example, the function that links nodes 6 (Structure) and 8 (Powerplant) is 4x, where x is the variable of composition. Also of interest is the fact that most nodes have a path structure that feeds back to themselves.

A1.6.4 Capacity Analysis

Graph algebra provides the system architect with the capability to determine the overall capacity between two components. This type of analysis is normally described as the transportation or flow problem.

The electrical power capacity structure for the helicopter system has been determined, captured into the N Squared matrix and populated with arbitrary power capacity 'flow capability weights' to demonstrate the method.

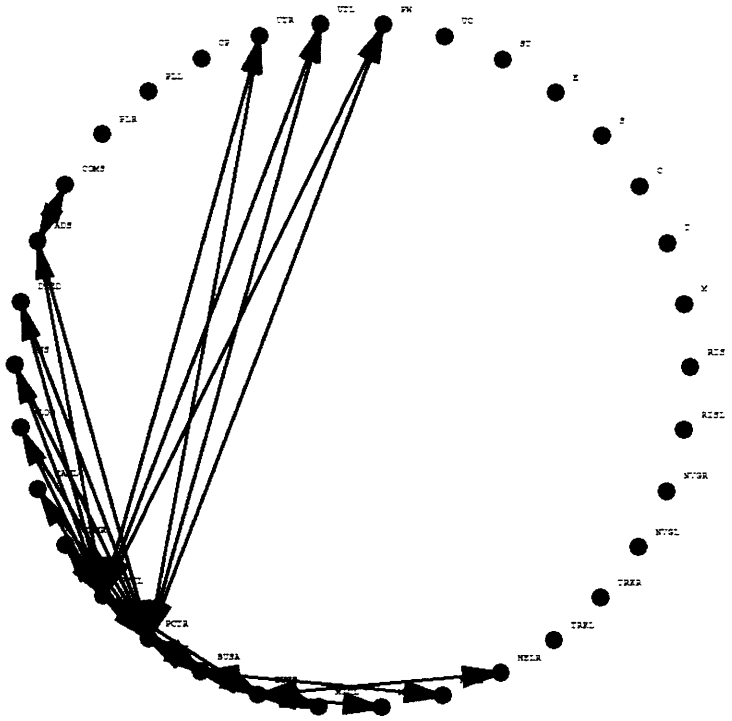


Figure 72 - Graph showing capacity structure (Helo Mr2w)

The capacities of the component links are shown as weights in the following matrix.

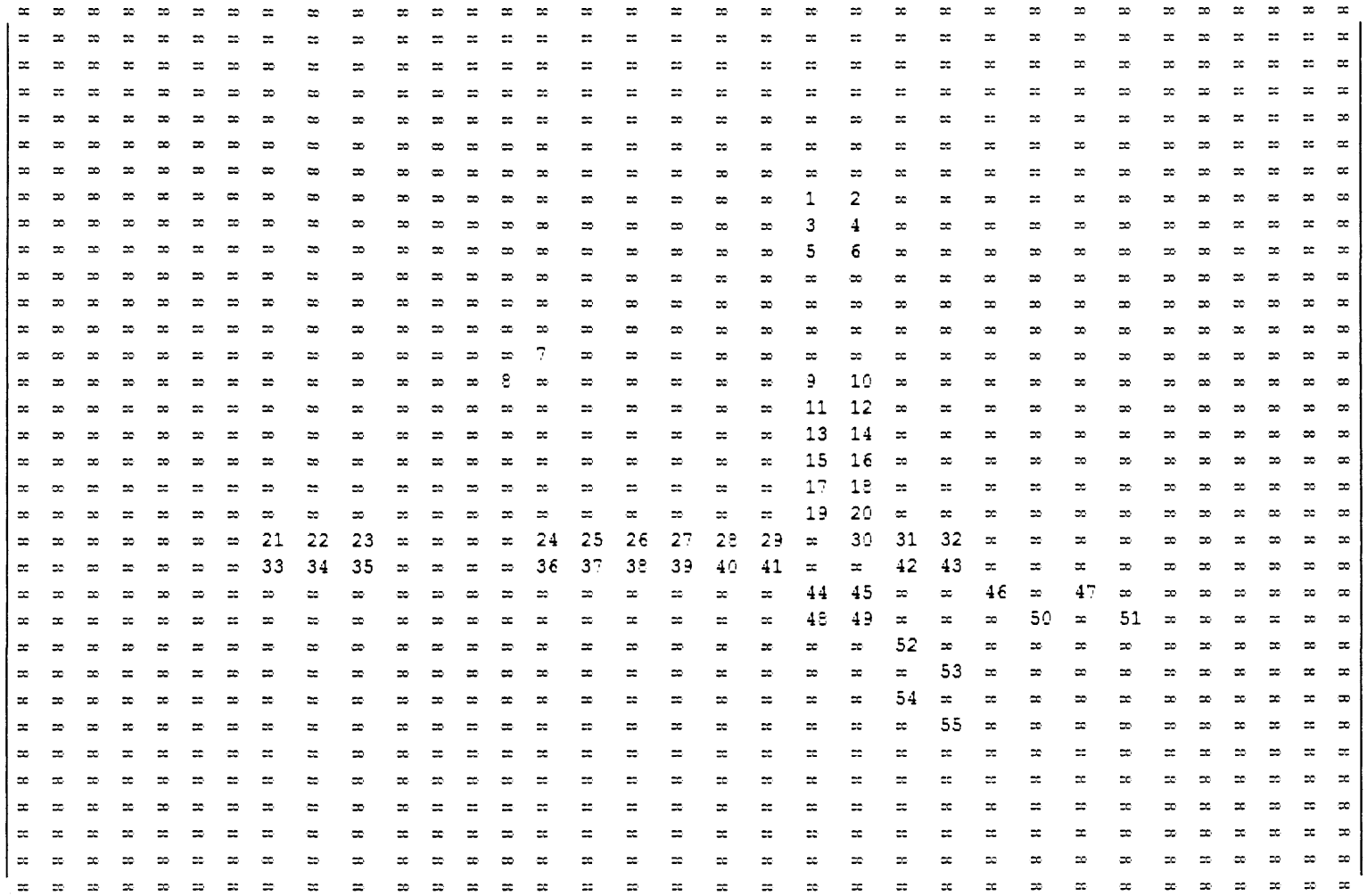


Figure 73 - Adjacency matrix with arbitrary capacity capabilities between each component (Helo Mr2w)

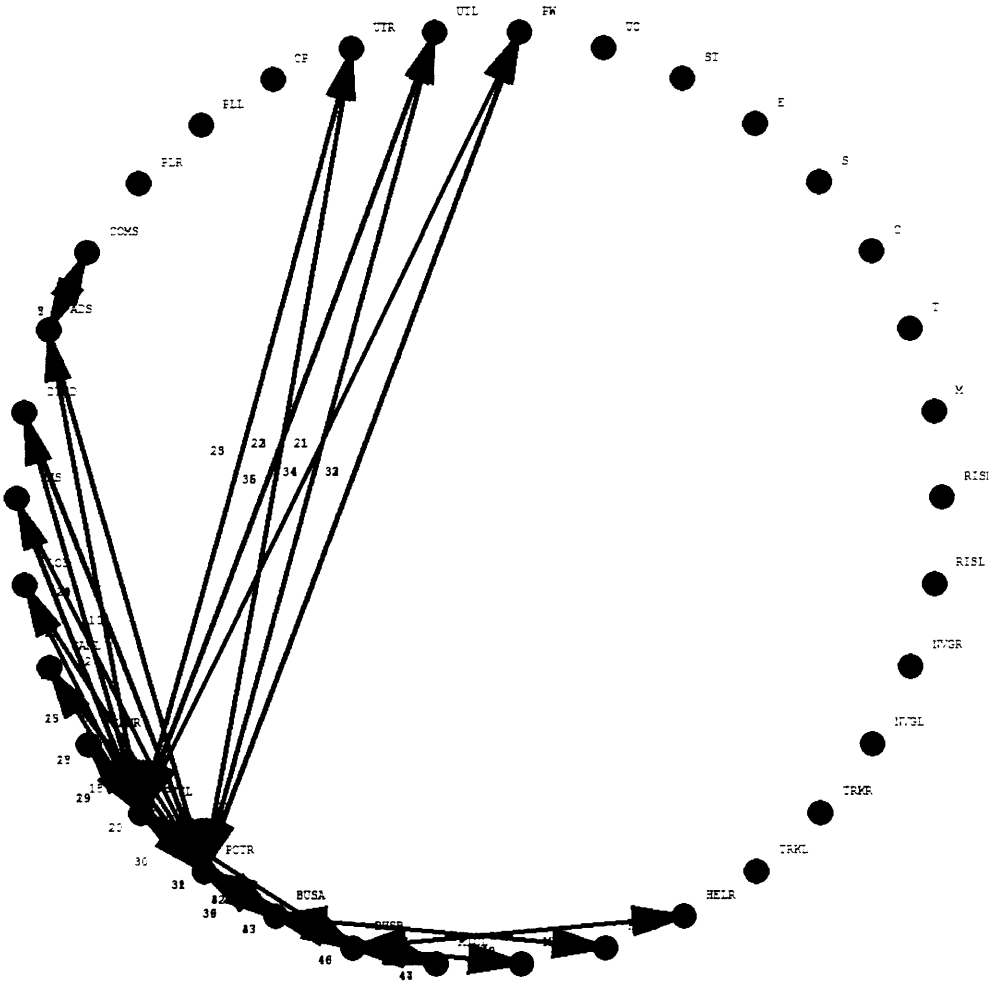


Figure 74 - Graph diagram of System Structure shown with capacity (arbitrary values) of each link (Helo Mr2w)

The analysis shows the paths that link the Helmet (28) to the Power System (8) are as follows.

{{{8,21}, 1},{{8,22}, 2},{{21,24}, 1},{{22,24}, 2},{{24,28}, 3}}

Each link has its associated power capacity and it can be seen that the limiting capacity is determined by link (8,21) with capacity 1 unit and link (8,22) with capacity 2 units.

Capacity is, of course, not just concerned with 'transportation' like constants. Many forms require each component to be expressed in functional multivariable form.

To demonstrate this viewpoint capacity is expressed in the form of a single polynomial variable and, for the purposes of demonstration, a simple linear form is used.

The structure of the electrical power distribution system shown in Figure 75.

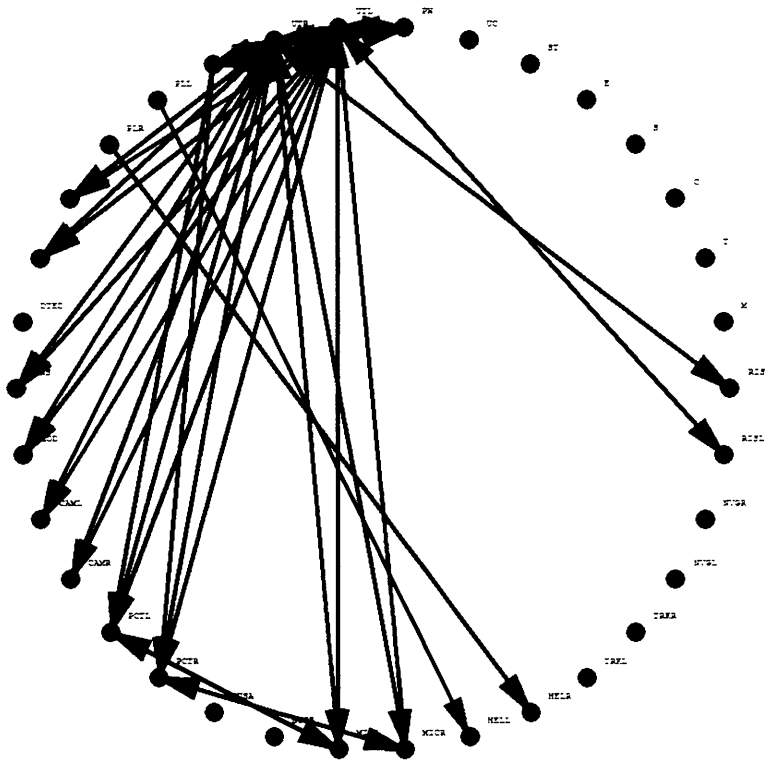


Figure 75 - Electrical power distribution system (Helo Mr2e)

Each system component has been allocated a unique arbitrary function that describes its power function. Figure 76 demonstrates the analysis of the N Squared Matrix for the system.

Mr2e, Columns 9-12

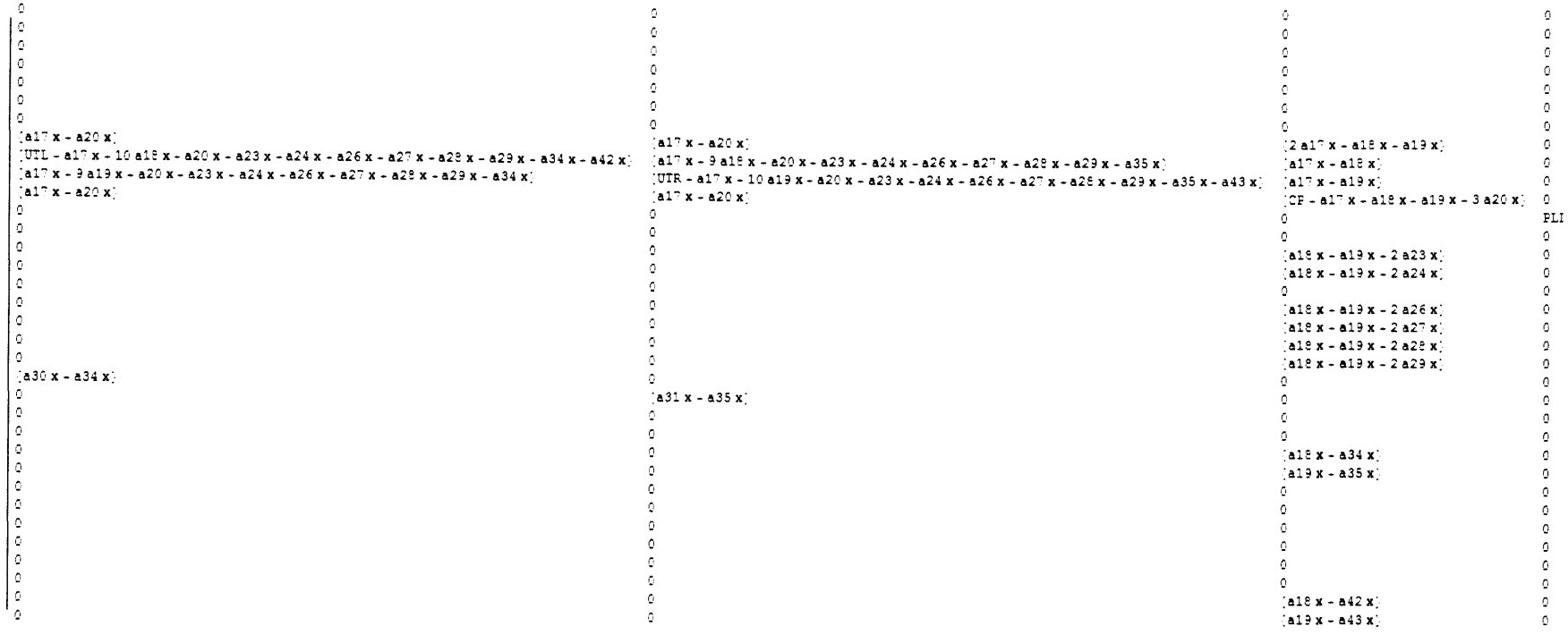


Figure 76 - Power Distribution System Function matrix showing 2 step dependency paths; Columns 9-12 only of 34 (Helo Mr2e)

The relationship between the Utility Bus Systems left and right are in element (10,1). The functional decomposition shows the summation of the 2 step paths that link the two bus systems is as follows.

"Expression between Utility (LHS) Bus and Utlity (RHS) Bus; Element 10,9 (10,1)"

{a17 x+9 a19 x+a20 x+a23 x+a24 x+

a26 x+a27 x+a28 x+a29 x+a34 x}

i.e. (a17 +9 a19 +a20 +a23+a24 +a26 +a27 +a28 +a29 +a34)x

It shows that the impedance function coefficient 'a19' combines with nine other impedance coefficients to enable the voltage distribution to be calculated for any current load (x).

A1.6.5 Interconnectivity Analysis

The data-bus interconnectivity structure has been determined and captured in the form of an N Squared Matrix held in Excel spreadsheet 'Helo N2 Mr3'.

The graphical form is shown in Figure 77.

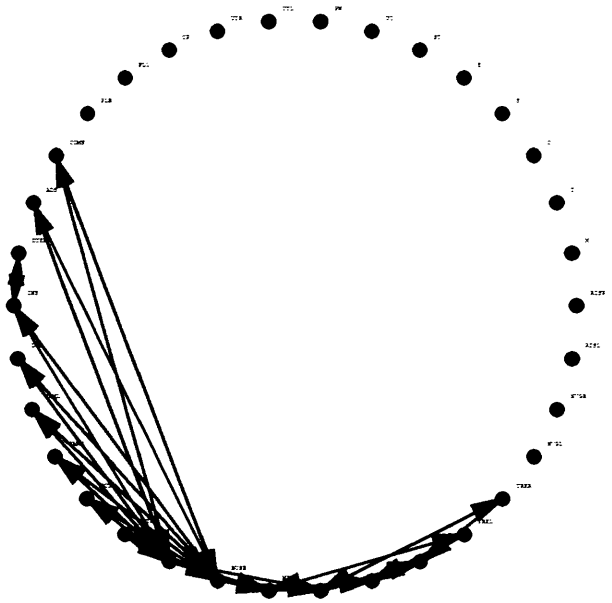


Figure 77 - Avionic Bus System Interconnectivity Structure (Helo Mr3)

It is assumed that the functionality of interconnectivity is not related to the functionality of each element; in effect each component has transfer function of 1.

To demonstrate the functionality relationships a part of the two step path matrix is shown in Figure 78.

To demonstrate the integrated functionality of the interconnected system the functionality of each link is described by an arbitrary (a.x + b) type polynomial where x is a system variable. Each link has unique a, b coefficients and the connectivity between the Laser Obstacle Detector and the Bus System for 3-step path combinations are shown in Figure 79.

The functionality analysis also shows that only odd lengths of path steps produce any relationships.

This example demonstrates the capability of the method to provide the overall functionality between pairs of nodes for a function that can be represented by a polynomial. The example does not attribute any engineering value to the polynomial. However, in this context the author suggests that it might represent bus traffic.

Interconnectivity Matrix with three steps and polynomial functional insertion; lower left quadrant

0	0	0	0	0	0	0	0	0	0	0	0	0	f327 f332 - f327 f333	f327 f332 - f327 f333	0	f327 f332 - f327 f333
0	0	0	0	0	0	0	0	0	0	0	0	0	f328 f332 - f328 f333	f328 f332 - f328 f333	0	f328 f332 - f328 f333
0	0	0	0	0	0	0	0	0	0	0	0	0	f329 f332 - f329 f333	f329 f332 - f329 f333	0	f329 f332 - f329 f333
0	0	0	0	0	0	0	0	0	0	0	0	0	f330 f332 - f330 f333	f330 f332 - f330 f333	0	f330 f332 - f330 f333
0	0	0	0	0	0	0	0	0	0	0	0	0	f331 f332 - f331 f333	f331 f332 - f331 f333	0	f331 f332 - f331 f333
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f326 f332	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f326 f333	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78 - Two step functionality relationships; Part Matrix only - lower left quadrant (18-34, and 1-17 of 34,34) (Helo Mr3)

Single Element Interconnectivity Function with three steps and polynomial functional insertion; Nodes BUS A to LCD => Element [23,18] only

$$[b23\,b32^2 - b24\,b32^2 - b26\,b32^2 - b27\,b32^2 - b28\,b32^2 - b29\,b32^2 - b30\,b32^2 - b23\,b32\,b33 - b24\,b32\,b33 - b26\,b32\,b33 - b27\,b32\,b33 - b28\,b32\,b33 - b29\,b32\,b33 - b30\,b32\,b33 - 2\,a32\,b23\,b32\,x - a33\,b23\,b32\,x - 2\,a32\,b24\,b32\,x - a33\,b24\,b32\,x - 2\,a32\,b26\,b32$$
$$a33\,b26\,b32\,x - 2\,a32\,b27\,b32\,x - a33\,b27\,b32\,x - 2\,a32\,b28\,b32\,x - a33\,b28\,b32\,x - 2\,a32\,b29\,b32\,x - a33\,b29\,b32\,x - 2\,a32\,b30\,b32\,x - a33\,b30\,b32\,x - a23\,b32^2\,x - a24\,b32^2\,x - a26\,b32^2\,x - a27\,b32^2\,x - a28\,b32^2\,x - a29\,b32^2\,x - a30\,b32^2\,x - a32\,b23\,b33\,x -$$
$$a32\,b24\,b33\,x - a32\,b26\,b33\,x - a32\,b27\,b33\,x - a32\,b28\,b33\,x - a32\,b29\,b33\,x - a32\,b30\,b33\,x - a23\,b32\,b33\,x - a24\,b32\,b33\,x - a26\,b32\,b33\,x - a27\,b32\,b33\,x - a28\,b32\,b33\,x - a29\,b32\,b33\,x - a30\,b32\,b33\,x - a32^2\,b23\,x^2 - a32\,a33\,b23\,x^2 - a32^2\,b24\,x^2 -$$
$$a32\,a33\,b24\,x^2 - a32^2\,b26\,x^2 - a32\,a33\,b26\,x^2 - a32^2\,b27\,x^2 - a32\,a33\,b27\,x^2 - a32^2\,b28\,x^2 - a32\,a33\,b28\,x^2 - a32^2\,b29\,x^2 - a32\,a33\,b29\,x^2 - a32^2\,b30\,x^2 - a32\,a33\,b30\,x^2 - 2\,a23\,a32\,b32\,x^2 - 2\,a24\,a32\,b32\,x^2 - 2\,a26\,a32\,b32\,x^2 - 2\,a27\,a32\,b32\,x^2 -$$
$$2\,a28\,a32\,b32\,x^2 - 2\,a29\,a32\,b32\,x^2 - 2\,a30\,a32\,b32\,x^2 - a23\,a33\,b32\,x^2 - a24\,a33\,b32\,x^2 - a26\,a33\,b32\,x^2 - a27\,a33\,b32\,x^2 - a28\,a33\,b32\,x^2 - a29\,a33\,b32\,x^2 - a30\,a33\,b32\,x^2 - a23\,a32\,b33\,x^2 - a24\,a32\,b33\,x^2 - a26\,a32\,b33\,x^2 - a27\,a32\,b33\,x^2 - a28\,a32\,b33\,x$$
$$a29\,a32\,b33\,x^2 - a30\,a32\,b33\,x^2 - a23\,a32^2\,x^3 - a24\,a32^2\,x^3 - a26\,a32^2\,x^3 - a27\,a32^2\,x^3 - a28\,a32^2\,x^3 - a29\,a32^2\,x^3 - a30\,a32^2\,x^3 - a23\,a32\,a33\,x^3 - a24\,a32\,a33\,x^3 - a26\,a32\,a33\,x^3 - a27\,a32\,a33\,x^3 - a28\,a32\,a33\,x^3 - a29\,a32\,a33\,x^3 - a30\,a32\,a33\,x^3]$$

Figure 79 - Functionality of Avionic Bus to Communications system for 3-step path combinations (Helo Mr3)

A1.6.6 Functional Analysis

Having established the system structure, its capacity and its interconnectivity, the next and final task is to assess its functionality.

The functional structure has been determined n the form of an N Squared Matrix held in an Excel spreadsheet (Ref. Helo N2 Mr4a), and imported into Mathematica Notebook Helo Mr4a. This is shown in adjacency matrix form in Figure 80. Its graphical description is shown in Figure 81.

One of the most useful means of system analysis is the use of matrix based network analysis, and Two-port analysis is just one such technique.

The technique describes each component as a 2x2 matrix each with unique and application specific coefficients. There is a variety of generic component forms, including, for example, series impedance, shunt impedance, transformer, coupled inductance. To enable each type of component to be combined in series (tandem), in parallel or with hybrid connectivity, three different forms of representation for each component, or groups of components, are required to maintain analytical consistency. These types are called transfer function, admittance and hybrid. Each type has a form of transform that enables its type to be transformed into the other types.

To enable complex networks to be analysed the components need to be grouped according to whether they are connected in series, in parallel or in a hybrid form.

In the example that follows, each function is a 2-port network component that is either a series or shunt impedance; in fact a series resistance or a shunt capacitance. Therefore, the labelled direct product/sum expression in the form ((a ∘ b) + (c ∘ d)) means:

(('a' in series with 'b') in parallel with ('c' in series with 'd')).

This means that, firstly the components need to be in A Type, then grouped as

a.b = m and c.d = n terms of products, where m,n are arbitrary matrices,

then m,n need to be transformed to Y form m',n' and then summed as (m'+n') to form the Y Form of the overall network (q.v. section 9.11.3).

To demonstrate the technique, the resistance and capacitance components are assumed to be identical and represented by the symbols R and C, and s is the Laplace operator.

The relationships for a one and two path construction has been calculated and Figure 82 shows columns 12,13,and 14 of the 34 column matrix.

Each function is represented by a 2x2 matrix using the designation a1,a2,a3,a4; b1,b2,b3,b4,... etc. Figure 82 shows Column 13 of the matrix. Then Figure 83 shows the Element (1,13) that demonstrates the construction of the 2x2 matrix of the one and two path structure between the Cockpit Subsystem and the Operational Mission.

Then the symbols R, C and s are inserted into the 2x2 matrix elements and the transfer function calculated, as shown in Figure 84. With arbitrary values of R = 1, and C = 1, the transfer function reduces to 2/(2 + s).

Note. Practical considerations of this A4 based compilation constrain the functionality that can be shown; please see referenced programs for complete analysis.

f410a	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	f411a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	f412a	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	f413a	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	f414a	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	f415a	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	f416a	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	f417a	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	f418a	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	f419a	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	f420a	1	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	f421a	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	1	f422a	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	1	f423a	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	f424a	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f425a	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	f42
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 80 - Structure of

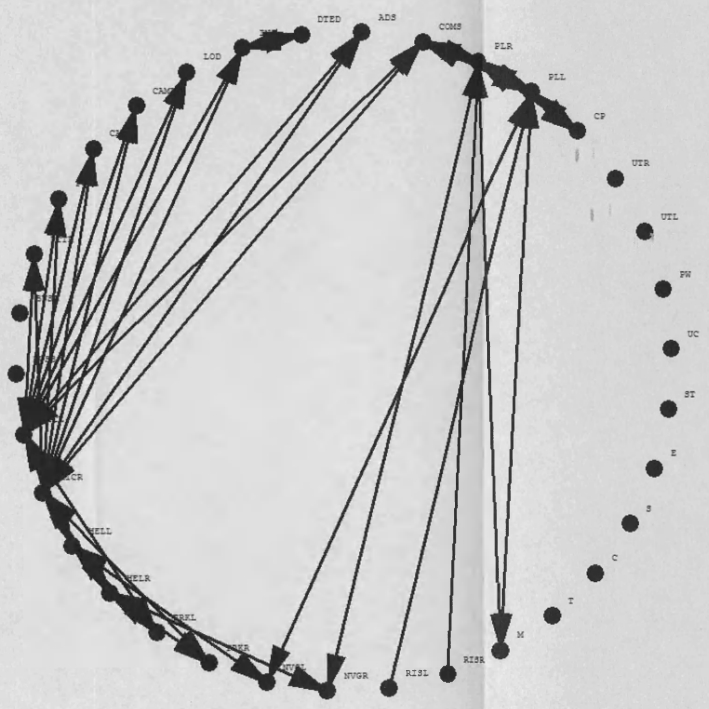


Figure 81 - Graph of Functional Domain

Path Structures; One and Two steps only; Column 12,13 and 14 only

f410a - f410a f422a	f410a - f410a f421a
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
f420a - f420a f422a	f420a - f420a f421a
f410a f421a - f420a f421a - f421a f422a - f421a f423a - f421a f440a	f421a - f410a f421a - f420a f4
f422a - f410a f422a - f420a f422a - f422a f423a	f410a f422a - f420a f422a - f4
f423a - f422a f423a	f423a - f421a f423a
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
f423a f434a	f423a f434a
f423a f435a	f423a f435a
f436a f440a	0
0	f437a f441a
0	0
0	0
f440a	f421a f440a
f422a f441a	f441a
f442a	f421a f442a
f422a f443a	f443a

Figure 82 - Path Structures for one and two step

Mission to Pilot Function shown in Matrix Element mm{1,13} in Admittance Form

$$\begin{vmatrix} \frac{f410a4}{f410a2} - \frac{f410a3\ f421a2+f410a4\ f421a4}{f410a1\ f421a2+f410a2\ f421a4} & \frac{f410a2\ f410a3-f410a1\ f410a4}{f410a2} - \frac{(f410a3\ f421a1+f410a4\ f421a3)\ (f410a1\ f421} \\ \frac{1}{f410a2} - \frac{1}{f410a1\ f421a2+f410a2\ f421a4} & -\frac{f410a1}{f410a2} - \frac{-f410a1\ f421a1-f410a2\ f421a3}{f410a1\ f421a2+f410a2\ f421a4} \end{vmatrix}$$

Mission to Pilot Function shown in Matrix Element mm{1,13} in A Form

$$\begin{vmatrix} \frac{f410a1^2\ f421a2+f410a2^2\ f421a3+f410a1\ f410a2\ (f421a1+f421a4)}{f410a2+f410a1\ f421a2+f410a2\ f421a4} \\ \frac{f410a1\ (2\ f410a3\ f421a2+f410a4\ (-1+f421a1+f421a2\ f421a3+f421a4-f421a1\ f421a4))+f410a2\ (2\ f410a4\ f421a3+f410a3\ (1+f4}{f410a2+f410a1\ f421a2+f410a2\ f421a4} \end{vmatrix}$$

Figure 83 - Mission to Right Hand Pilot Function shown in Matrix Element mm{1,13} in

Mission to Pilot Function shown in Matrix Element mm[1,13]\n shown with coefficients of the 2-Port Form\n for series resitance and shunt capacitance components, R and C

$$\begin{vmatrix} \frac{tR+CR^2s}{tR} & \frac{R}{t} \\ Cs & 1 \end{vmatrix}$$

Mission to Pilot Function shown in Matrix Element mm[1,11] with R=1 and C=1

$$\begin{vmatrix} \frac{t+s}{t} & \frac{1}{t} \\ s & 1 \end{vmatrix}$$

Mission to Pilot Function shown by Transfer Function between nodes Mission and Cockpit

$$\frac{2}{2-s}$$

Figure 84 - Insertion of 2-Port Component Symbols and Values. Generation of Transfer Function From Mission to Right Hand Side Pilot

ADDENDUM TO APPENDIX 1 - INTRODUCTION TO HELICOPTER FLIGHT SAFETY

A helicopter is a type of aircraft that is capable of hovering at zero ground speed as well as normal translational flight over ground. This is achieved by the use of vertical thrust with a power that exceeds the aircraft's weight combined with normally powered wing borne flight, when the power required to sustain flight is a fraction of the aircraft's weight.

To ensure safe flight many types of hazard that have the potential to lead to a severe safety related condition need to be addressed. Therefore, to understand the rationale for the system configurations and the appraisal of the analysis results, it is useful for the reader to understand certain issues that pertain to safe flight of a helicopter. The major flight regimes are described as follows.

AA1.1 Avoid Curve

This is the boundary of the region of the operational flight envelope within which in the event of loss of power, insufficient altitude clearance exists below the aircraft to enable the rotorcraft either to make a safe autorotation landing or to attain sustainable flying speed in level flight.

AA1.2 Minimum Power Speed

This is the speed, in knots, at which the rotorcraft flies in sustainable level flight at sea level ISA conditions with minimum engine power.

AA1.3 Minimum Safety Height

This is defined as 1000 feet above the highest terrain obstacle within a radius of 5 nautical miles of the aircraft.

AA1.4 Minimum Autorotation Height

This is the minimum height above a proposed landing site for the pilot to establish and control the aircraft to a safe landing from the decision point to make a safe a autorotation landing.

AA1.5 Minimum Obstacle Visual Reference Region

This is the region within which it is necessary for the pilot to see any terrain features or obstacles in order to execute avoidance manoeuvres without the aid of the NVS This is typically a minimum of 400 feet altitude and 500 metres range.

AA1.6 Operation Flight Envelope Segmentation

As a consequence of the versatility of a helicopter for both powered and aerodynamic flight in relation to the techniques for hazard survival, helicopter flight envelopes are segmented as shown in the following table.

Band	Range	Description
Altitude	*(feet)*	
1	0-150	This is the band for minimum visibility required for taxiing, take-off, low level flight and landing (including autorotation).
2	150-400	This band covers the altitude range in which it is highly undesirable to combine with flight speeds that are lower than that required to sustain equivalent wing borne flight.
3	400 - Ceiling	This is for all standard mission flight profiles and segmented by minimum safety altitudes required to comply with ATC flight rules.
Speed	*(knots)*	
1	0 - 40	This covers the speed below the transition speed, the speeds within the avoid curve and hovering in and out of ground effect.
2	40 - 70	This covers the range between minimum flight speeds for sustained equivalent wing borne flight above the avoid curve and the minimum power speed.
3	70 to VNE	This covers the range of speeds for normal flight operations.

Table 25 - Typical Segmentation of a Helicopter Flight Envelope

AA1.7 Piloting Modes

Assuming that the minimum visibility for authorised flight is based on obstacle clearance with 400 feet minimum cloud base and 500 metres range, then three piloting modes are required to provide day night all weather operations.

1. VMC
2. IMC assisted with external visual reference information to provide minimum obstacle clearance.
3. IMC without any external visual reference information.

Therefore, the operational modes are as shown in Table 26.

Mode	Operational Description	Piloting Mode
1A	Standard Flight in Controlled Airspace	a)VMC b) IMC without VMC equivalent external visual references.
1B	Standard Flight not in Controlled Airspace	a)VMC b) IMC without VMC equivalent external visual references.
2	Hover and Low speed/altitude manoeuvring.	a)VMC b) IMC with minimum VMC equivalent external visual references.
3	Nap of the earth.	a)VMC b) IMC with minimum VMC equivalent external visual references.
4A	Low speed out of ground effect manoeuvring in Controlled Airspace.	a)VMC b) IMC without VMC equivalent external visual references.
4B	Low speed out of ground effect manoeuvring not in Controlled Airspace.	a)VMC b) IMC without VMC equivalent external visual references.

Table 26 - Operational and Piloting Modes for a Typical Helicopter

Note 1. The effects of day and night operations are included within the definitions of piloting modes.

Note 2. IMC manoeuvring operations without the use of visual references of the ground below for the hover and NOE modes are not safe.

Note 3. The effects of environmental conditions need to be included in the selection of the piloting mode.

Note 4. For flying at altitudes above 400 feet and below the minimum safety altitude in IMC it is a condition that either the pilot or commander must be able to see

the ground with VMC equivalent performance assisted as required by NVG or other piloting vision system.

The minimum information requirements for the pilot to deal satisfactorily with various hazards are shown in the following table.

Ref.	Hazard	Relevant Piloting Parameter; Measured	Relevant Piloting Parameter; Inferred
A	Loss of attitude control.	Pitch attitude Roll attitude Heading Height Speed (IAS) Torque Vertical Speed	Yaw rate
B	Loss of spatial position. Loss of geographical position.	Pitch attitude Roll attitude Heading Height Speed (IAS) Torque Position Height (Baro) Speed (IAS) Heading Track Map (hand held)	Ground Speed Height error Heading error Track error Ground speed
C	Loss of power.	Height Speed (IAS) Torque Engine rpm Engine PTIT Rotor rpm	Yaw rate Roll rate Height rate Pitch attitude Roll attitude
D	Loss of terrain or Obstruction clearance.	Pitch attitude Roll attitude Heading (Height) Speed (IAS) Map (hand held)	Tip path plane Ground slope Inceptor positions Terrain / obstruction data
E	Collisions: - - other aircraft - foreign objects - birds - obstacles	 Ext & Int Comms Int Comms Ext & Int Comms Internal Comms	 Manoeuvre limits Engine response Safe escape route
F	Disorientation.	Artificial Horizon Heading Height Speed (IAS) Vertical Speed	State recognition

Table 27 - Hazard Control Piloting Information Requirements

APPENDIX 2 BACKGROUND MATHEMATICS

A2.1 Abstract Algebra

The following analysis techniques are based on some basic principles of abstract algebra. Many textbooks elucidate the required mathematical principles and the following axiomatic definitions are taken from 'Algebra', J.W. Archibald, 1970, [115].

A Set is defined by the following axioms.

Axiom 1.

i) Reflexive property of inclusion; $S \subset S$ for all S.

ii) Transitive property; If $S_1 \subset S_2$ and $S_2 \subset S_3$ then $S_1 \subset S_3$.

Axiom 2.

i) Reflexive property of equality of sets; $S = S$ for all S.

ii) Symmetric property; If $S_1 = S_2$ then $S_2 = S_1$.

iii) Transitive property; If $S_1 = S_2$ and $S_2 = S_3$ then $S_1 = S_3$.

Axiom 3.

i) Idempotent property of intersection; $S \cap S = S$ for all S.

ii) Symmetric property; $S_1 \cap S_2 = S_2 \cap S_1$ for all S_1, S_2.

iii) Associative property; $S_1 \cap (S_2 \cap S_3) = (S_1 \cap S_2) \cap S_3$ for all S_1, S_2, S_3.

Axiom 4.

i) Idempotent property of union; $S \cup S = S$ for all S.

ii) Symmetric property; $S_1 \cup S_2 = S_2 \cup S_1$ for all S_1, S_2.

iii) Associative property; $S_1 \cup (S_2 \cup S_3) = (S_1 \cup S_2) \cup S_3$ for all S_1, S_2, S_3.

Axiom 5.

Let S be a non-empty set. A binary relation R from S to S i.e. a subset of $S \times S$ is an equivalence relation on S when:

i) Reflexive property; xRx for every x in S.

ii) Symmetric property; whenever xRy then yRx.

iii) Transitive property; whenever xRy and yRz then xRz.

A Group is defined as follows.

A system (G, ∘) consisting of a non-empty set G and a law of binary composition ∘ is a Group when

i) G is closed under ∘

ii) ∘ is associative

iii) ∘ has a left identity e

iv) Every a in G has a left inverse a^{l} with respect to e.

When two sets A, B are defined as {a, b} and {c, d}, the Cartesian Product, $A \times B = \{(x, y): x \varepsilon A \text{ and } y \varepsilon B\}$ is the set of all ordered pairs {(a, c), (a, d), (b, c), (b, d)}. Each component represents a function. So each ordered pair represents the functions that must be combined to determine the functionality of the pair; see 'Introduction to Set Theory and Topology', K. Kuratowski, 1972, [116]. The component functions in each set may be combined by the operations of Union, Intersection or Difference.

Two elements commute when:

1. (a + b) + c = a + (b + c),
2. a + b = b + a,
3. a + 0 = a, and
4. a + (-a) = 0.

The system is Abelian when its binary operation is commutative.

A Group G is called a Direct Product of proper sub-groups $H_1,...H_n$ when:

i) Every x in G has a representation $x = h_1,...,h_n$ with h_1 in $H_1,...$, h_n in H_n.

ii) This representation is unique.

iii) Every element in Hi commutes with every element in Hj when i≠j.

Then $G = H_1 \times H_2 \times ... \times H_n$, the order of the factors being immaterial.

This also applies to the direct sum G=H1 + H2 + ...+ Hn.

A2.2 Graph Theory

The origins of graph theory are typically traced to a paper written by Leonhard Euler in 1736 for the Imperial Academy of Science at St. Petersburg; see 'Graph Theory 1736-1936', N. L. Biggs, E. K. Lloyd, R. J. Wilson, Oxford, 1976, [42a].

The problem concerned the city of Koenigsberg. The river Pregel flows through the city and at one point it separates into two branches that surround an island. To enable the

population to have easy access to all four parts of the city the river is spanned by seven bridges. The problem was to devise a route whereby a traveller would reach each part of the city by crossing each bridge only once.

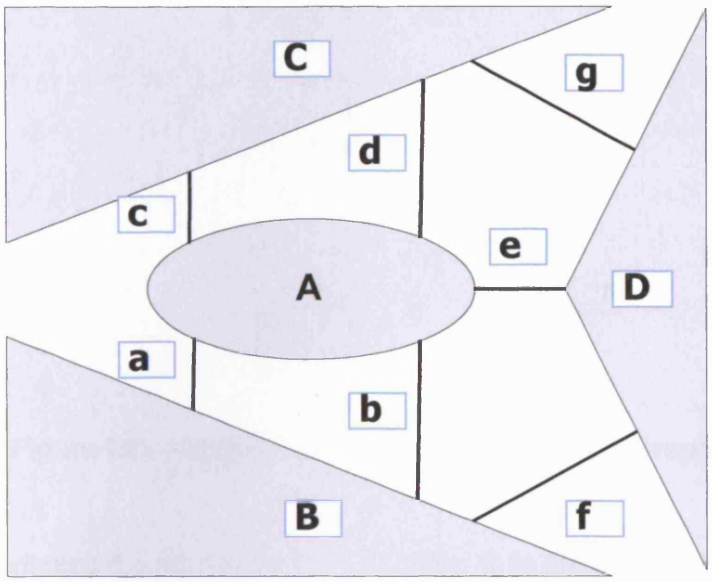


Figure 85a - Schematic of City of Koenigsberg and Bridges over the River Pregel (D18)

A schematic of Koenigsberg with the island of Kneiphof is shown in Figure 85a. Euler's paper considered the problem by allocating four letters, A, B, C, D, that represented the four parts of the city and represented the bridges by the letters a,...., g.

Firstly, Euler abstracted the problem into a form of graph as shown in Figure 85b.

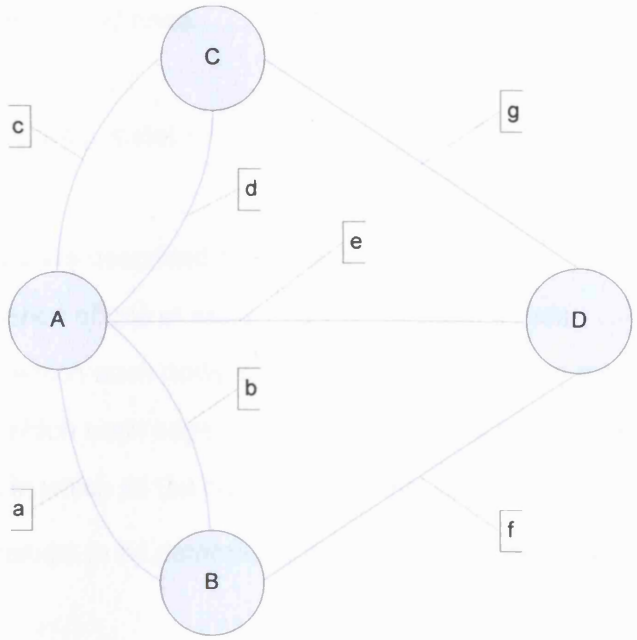


Figure 85b – Map of Koenigsberg shown as a Graph

Secondly, he represented a movement from A to B to C by the sequence ABC. The number of potential routes could then be determined by consideration of sequences of letters. To be successful a sequence of one plus the number of bridges is required. Therefore, a sequence of eight letters is required to consist of pairings equal to the number of bridges used. So AB is required twice, AC is required twice and AD, BD, CD must each occur once. Therefore, 'A' must occur three times and 'B', 'C' and 'D' must occur twice; this is clearly impossible in an eight-letter sequence.

Therefore, it is not possible to find such a route. He derived a rule that says that for an odd number of bridges the number of times a territory can be accessed must equal 'half of one plus the number of bridges'.

For an even number of bridges the number of times a territory can be accessed is equal to half the number of bridges if not starting from A, and one plus half the number of bridges if starting from A.

The theory of directed graphs (digraphs) is based on four primitives and four axioms, as outlined by Harary et al, [42b].

The primitives are:

P1: A set V of elements called 'vertices or nodes'.

P2: A set X of elements called 'directed lines'.

P3: A function f whose domain is X and whose range is contained within V.

P4: A function s whose domain is X and whose range is contained within V.

A1: The set V is non-empty and finite.

A2: The set X is finite.

A3: No two distinct lines are parallel.

A4: There are no loops.

The types of relationships are described as follows:

A walk is the sequence of one or more and zero or more edges.

A path is a walk in which each node is distinct; i.e. there are no repeated nodes.

A trail is a walk in which each edge is distinct. A circuit is a closed trail.

A cycle is a circuit in which all the nodes are distinct.

To enable these relationships to be determined the idea of connectedness is used. Each pair of nodes is:

Weakly connected if there is a semi-path between them.

Unilaterally connected if there is a path between them.

Strongly connected if there is a path in both directions.

The structure of the graph can be evaluated in terms of the trees and distance, connectivity and colourings, see 'Introduction to Graph Theory', D. B. West, 2001, [42c].

A2.3 Dynamical Systems

Since the underlying assumption that a machine system is some form of automata, the foundations of modern systems dynamics are summarised by the following axioms:

(See 'Topics in Mathematical System Theory', R.E Kalman, P.L Falb, M.A.Arbib, 1969, McGraw Hill. [51].

1) There is a given time set T, a state set X, a set of input values U, a set of acceptable input functions $\Omega = \{\omega : T \rightarrow U\}$, a set of output values Y, and a set of output functions $\Gamma = \{\gamma : T \rightarrow Y\}$.

2) T is an ordered subset of reals.

3) The input space Ω satisfies the following conditions:

a) Ω is not empty.

b) An input segment $\omega(t1, t2)$ is $\omega\varepsilon\Omega$ restricted to $(t1, t2) \cap T$. If $\omega, \omega'\varepsilon\Omega$ and $t1<t2<t3$ there is an $\omega''\varepsilon\Omega$ such that $\omega''(t_1, t_2) = \omega(t_1, t_2)$. and $\omega''(t_2, t_3). = \omega'(t_2, t_3)$.

4) There is a state transition function $\varphi : T \times T \times X \times \Omega \rightarrow X$ whose value is the state $x(t) = \varphi(t; \tau,x,\omega)\varepsilon X$ resulting at time $t\varepsilon T$ from the initial state $x = x(\tau)\varepsilon X$ at initial time $\tau\varepsilon T$ under the action of the input $\omega\varepsilon\Omega$ where φ has the following properties:

   a) φ is defined for all $t>\tau$ but not necessarily for all $t<\tau$.

   b) $\varphi(t; \tau,x,\omega) = X$ for all $t\varepsilon T$, $x\varepsilon X$, $\omega\varepsilon\Omega$.

   c) For any $t1<t2<t3$ we have $\varphi(t3; \tau1,x,\omega) = \varphi(t3; \tau2,x\ \varphi(t2; \tau1,x,\omega),\omega)$ for all $x\varepsilon X$, $\omega\varepsilon\Omega$.

   d) If $\omega, \omega'\varepsilon\Omega$ and $\omega(\tau,t). = \omega'(\tau,t)$ then $\varphi(t; \tau,x,\omega) = \varphi(t; \tau,x,\omega')$.

5) There is a readout map $\eta : T \times X \rightarrow Y$ which defines the output $y(t) = \eta(t, x(t))$ where the map $(\tau,t) \rightarrow Y$ given by $\sigma \rightarrow \eta(\sigma,\varphi(\sigma; \tau,x,\omega))$, $\sigma\omega(\tau,t)$ is an output segment, that is the restriction $\gamma(\tau,t)$ of some $\gamma\varepsilon\Gamma$ to (τ,t).

A system is constant (time invariant) if its response to a given input segment (in a given state) is independent of the time interval in which the trial takes place.

This is defined as follows:

A dynamical system Σ is constant iff:

1) T is a additive group (usually of reals)

2) Ω is closed under the shift operator $Z^\tau : \omega \rightarrow \omega'$ defined by $\omega'(t) = \omega(t+\tau)$ for all τ, $t \varepsilon T$

3) $\varphi(t; \tau,x,\omega) = \varphi(t+s; \tau+s,x,Z^s\omega)$ for all $s \varepsilon T$

4) The map $\eta(t, .) : X \rightarrow Y$ is independent of t.

A dynamical system Σ is continuous iff T = reals; Σ is discrete time iff T = integers.

A dynamical system Σ is finite dimensional iff X is a finite dimensional linear space; Σ is finite state iff X is a finite set. Σ is finite iff X, U and Y are finite sets and, in addition, Σ is constant and discrete time. Thus dim Σ Δ dim X.

A dynamical system Σ is linear iff:

1) X, U, Ω, Y and Γ are vector spaces (over a given arbitrary field K).

2) The map $\varphi(t; \tau,.,.): X \times \Omega \rightarrow X$ is K linear for all t and τ.

3) The map $\eta(t, .) : X \rightarrow Y$ is K linear for all t.

A dynamical system is smooth iff:

1) T = R the real numbers.

2) X and Ω are topological spaces.

3) The transition map has the property that defines a continuous map.

A dynamical system Σ in the input/output sense is a composite mathematical concept defined as follows:

There are given sets T, U, Ω, Y and Γ satisfying all the properties required by Definition 1

There is given a set indexing a family of functions

$$\Im = \{f_\alpha : T \times \Omega \rightarrow Y, \alpha \varepsilon A\}:$$

wherein each member of $\Im$ is written explicitly as $f_\alpha (t, \omega) = y(t)$ which is the output resulting at time t from the input ω under the experiment α.

Each f_α is called an input/output function and has the following properties:

1) Direction of time. There is a map $\iota : A \rightarrow T$ such that f_α (t, ω) is defined for all $t>\iota(\alpha)$.

2) Causality. Let τ, $t = \in T$ and $\tau<t$. If ω, $\omega' \varepsilon \Omega$ and $\omega_{(r,t)} = \omega'_{(r,t)}$ then f_α (t, ω) = f_α (t, ω') for all α such that $\tau = \iota(\alpha)$.

If Σ is a smooth dynamical system then assume that;

1) T = **R**, X and U are normed spaces.

2) Ω is the normed space of continuous functions $T \rightarrow U$ with $||\omega|| = \sup(t = \in T) ||u(t)||$.

3) $\varphi(.; \tau,x,\omega) \in C^1(T \rightarrow X)$ for each τ,x, and ω and the map T x X x $\Omega \rightarrow$ X given by $(\tau,x,\omega) \rightarrow \varphi(dot)(t; \tau,x,\omega)$ is continuous for each t with respect to the product topology.

Then the transition function φ of Σ is a solution of the differential equation dx/dt = $f(t,x,\pi_t\omega)$ where the operator π_t is a map $\Omega \rightarrow U$ given by $\omega \rightarrow u(t) = \omega(t)$.

For dynamical systems that are finite dimensional, linear and smooth and as the state space must be a vector space with $X = \mathbf{R}^n$. Then the transition function must be linear on X x Ω then

$$\begin{aligned} \varphi(t; \tau,x,\omega) &= \varphi(t; \tau,x,0) + \varphi(t; \tau,0,\omega) \\ &= \Phi(t, \tau)\, x + \Theta(t, \tau)\, \omega \end{aligned}$$

The linear map $x \rightarrow \Phi(t, \tau)\, x$ defined by the first term is the transition map of Σ.

As Σ is smooth then T = R and φ satisfies the differential equation

$$(d/dt)\, \varphi(t; \tau,x,\omega) = f(t, \varphi((t; \tau,x,\omega), u(t))$$

Then the functions F : T $\rightarrow$ {n x n matrices} and G : T $\rightarrow$ {n x m matrices} can be introduced as follows:

$$f(t,x,u(t)) = F(t)x + G(t)u(t)$$
$$Y = f(t,x,u(t))$$

where φ satisfies the differential equation dx/dt = F(t)x + G(t)u(t) and the transition matrix Φ satisfies the differential equation $(d/dt)\, \Phi(t, \tau) = F(t)\, \Phi(t, \tau)$.

Further, since η is linear on X then $y(t) = \eta(t,x(t),u(t)) = H(t)x(t) + J(t)u(t)$

Therefore, every continuous time finite dimensional linear smooth dynamical system Σ obeys these relationships.

A2.4 State Space representation of Combinations of Dynamic Systems

Block diagrams of dynamical systems are usually represented in transfer function form. Sub-systems can be connected together in series or parallel as shown in Figure 86.

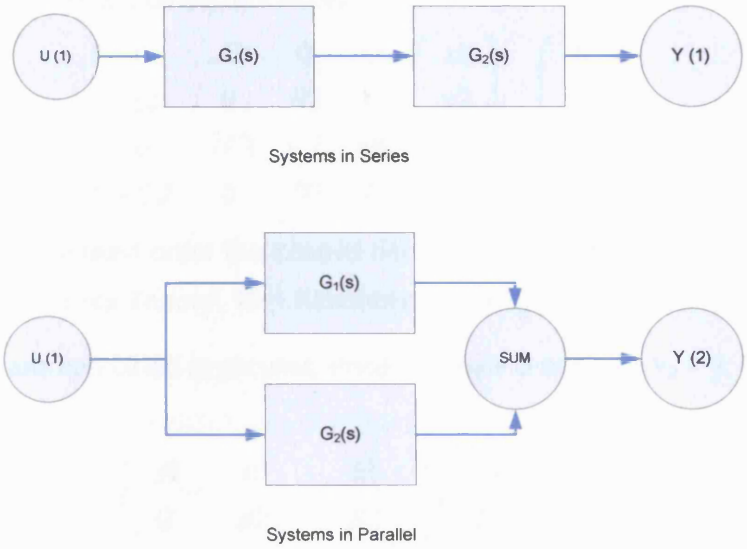


Figure 86 - Transfer Function Block Diagram of Systems Connected in Series and Parallel (D19)

If the two systems have the same dimensions and inputs so that

$$y_i(s) = G_i(s).u_i(s), \text{ where } i = 1,2,$$

the transfer function of systems in series and parallel are respectively

$$y_2(s) = G_i(s).\ G_2(s).\ u_i(s), \text{ and}$$
$$y_2(s) = [G_i(s) + G_2(s)].\ u_i(s).$$

The normal form of the state equations are as follows:

$$s\underline{x} = \mathbf{A}x + \mathbf{B}u$$
$$y = \mathbf{C}x + \mathbf{D}u$$

where s is the Laplace differential operator, **A, B, C, D** are the system matrices, u is the input matrix, x is the system state matrix and y is the output matrix.

The standard form of the system equation is:

$$\begin{pmatrix} sI - A & B \\ -C & D \end{pmatrix} \begin{pmatrix} x \\ -u \end{pmatrix} = \begin{pmatrix} 0 \\ -y \end{pmatrix}$$

The transfer function G(s) can be represented in state variable form as follows:

$$G(s) = C\,(sI\text{-}A)^{-1}\,B + D$$

Two systems are designated as $P_1 = \begin{pmatrix} sI - A1 & B1 \\ -C1 & D1 \end{pmatrix}$ and $P_2 = \begin{pmatrix} sI - A2 & B2 \\ -C2 & D2 \end{pmatrix}$

Systems in series are combined as follows:

$$\begin{pmatrix} 0 & A1 & 0 & B1 \\ A2 & 0 & B2 & 0 \\ 0 & -C1 & -I & D1 \\ -C2 & 0 & D2 & 0 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ -u2 \\ -u1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -y2 \end{pmatrix}$$

To reduce this to the least order the internal decoupling zeroes must be found; see 'State Space and Multivariable Theory', H H Rosenbrock, 1970, [117].

When systems are combined in parallel, since $u_1 = u_2 = u$ and $y_1 + y_2 = y$, the system matrix is as follows:

$$\begin{pmatrix} A1 & 0 & B1 \\ 0 & A2 & B2 \\ -C1 & -C2 & D1 + D2 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ y \end{pmatrix}$$

A2.5 Conclusion

The behaviour of subsystems can be represented in state variable form and can be combined as a union (x) or intersection (+).

APPENDIX 3 BOND GRAPHS

A3.1 Component Models for Bond Graphs

To enable the Bond Graph methodology to be applied to all systems engineering domains of interest, the following extension to generic parameter definition is proposed.

Relationship with Optical systems Domain

It is proposed that the across and through variables correspond to solid angle and luminous flux.

Relationships with Electro-magnetic Radiation Domain

It is proposed that the across and through variables correspond to electric field and current.

Relationship with Radioactivity Domain

The unit of activity is based on the number of disintegrations per second; Curie or Rutherford.

The unit of radiation is the Roentgen and the Rad is the unit of absorbed radiation.

It is proposed that the across and through variables correspond to half life and radiation intensity.

Relationship with Information Science Domain

As part of his work on communication systems, Norbert Wiener, in 'Cybernetics' [48] deduced the fact that information could be represented by a binary code and Shannon, 1948, [118], defined the number of bits required to represent information in terms of its uncertainty as:

$I = \log_2(1/p)$ where p is the probability of the information event.

The energetic capacity of an information system is known as its information entropy. Suh addressed the information content of design, [81]. He defined the probability of success in a measurement as the Acceptable Range/Actual Range; tolerance/range.

Relationship with Transportation Logistics Domain

It is proposed that the across and through variables correspond to volume flow and unit type.

Ref.	Discipline	Across variable	Through variable	Resistance	Inertance	Capacitance
1	Occupancy	Length	Metric space unit	N/A	N/A	N/A
2a	Mechanical (linear)	Displacement	Force	Friction	Mass	Spring constant
2b	Mechanical (rotational)	Angular displacement	Torque	Friction	Moment of Inertia	Spring constant
3a	Mechanical dynamic (linear)	Force	Velocity	Friction	Mass	Spring constant
3b	Mechanical dynamic (rotational)	Torque	Angular velocity	Friction	Moment of Inertia	Spring constant
4	Hydraulic	Pressure	Volumetric rate of change	Restriction	Mass	Compressibility
5	Pneumatic/ Acoustic	Pressure	Volumetric rate of change	Restriction	Mass	Compressibility
6	Electrical	Voltage	Current	Impedance	Inductance	Capacitance
7	Magnetic	Magneto motive force	Magnetic flux	Impedance	Inductance	Capacitance
8	Optical	Field	Current	Absorber/ spectrum filter	Path delay	N/A (pump/ Intensifier)
9	EM	Solid angle	Luminous flux	Impedance	Inductance	Capacitance
10	Radioactivity	Radiation intensity	Dose rate	Absorber	<>	Half life
11	Thermal	Temperature	Entropy change rate	Thermal conductivity	N/A	Thermal capacitance
12	Material	Chemical potential	Mole flow rate	Absorber	<>	<>
13	Chemical	Enthalpy	Mass flow rate	Absorber	<>	<>
14	Computing	Information rate (baud rate)	Memory	Rate restriction	Delay	Storage/ memory
15	Transportation Logistics	Item/asset flow	Unit type	Transaction rate restriction	Delay	Storage
16	Institutional Activity	Activity/Task flow	Unit type	Transaction rate restriction	Delay	Storage

Table 28 - Proposed Bond and Pseudo Bond Graph Generic and Discipline Equivalence Components.

Relationship with Institutional Activity Domain

It is proposed that the across and through variables correspond to volume flow and unit type.

A3.2 Extension of Applicability of Bond Graph and pseudo Bond Graph Capability

The proposed additional generic components and state variables, that enable the utility of pseudo Bond Graph constructions to be extended to multi-disciplinary systems, are shown in Table 28.

A3.3 Component Representation for Intra-Domain Analysis

To enable sub-system representations to be combined using the terms of the Direct Product in the sense of union or intersection, the representation needs to assess series and parallel component combinations.

The following Table has been generated to consolidate sub-system representation using across and through variables for both capacity and behavioural analysis.

Ref.	Discipline	Across Variable	Through Variable	Capacity			Behaviour		
		A	b	Parameter	Series	Parallel	State variable	Series	Parallel
1	Occupancy	Length	Metric space unit						
2a	Mechanical Strength	Displacement	Force	Work $W = a \times b$	$W_1 \geq W_2 \ldots$	$W_1 +- W_2 +- \ldots$	Displacement	$D = \sum d_n$ $n = 1,..,N$	$D = d_1 \leq d_2 \ldots$
2b		Angular displacement	Torque	Work $W = a \times b$	$W_1 \geq W_2 \ldots$	$W_1 +- W_2 +- \ldots$	Angular displacement	$D = \sum d_n$ $n = 1,..,N$	$D = d_1 \leq d_2 \ldots$
3a	Mechanical motion	Force	Velocity	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Displacement	$D = \sum d_n$ $n = 1,..,N$	$D = d_1 \leq d_2 \ldots$
3b		Torque	Angular velocity	Displacement	$D = \sum d_n$ $n = 1,..,N$	$D = d_1 \leq d_2 \ldots$	Angular displacement	$D = \sum d_n$ $n = 1,..,N$	$D = d_1 \leq d_2 \ldots$
4	Hydraulic	Pressure	Volumetric rate of change	Work $W = a \times b$	$W_1 \geq W_2 \ldots$	$W_1 +- W_2 +- \ldots$	Pressure	$P = p_1 \leq p_2 \ldots$	$P = \sum p_n$, $n = 1,..,N$
5	Pneumatic/Acoustic	Pressure	Volumetric rate of change	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Intensity	$S = s_1 \leq s_2 \ldots$	$S = \sum s_n$, $n = 1,..,N$
6	Electrical	Voltage	Current	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Charge	$Q = q_1 \leq q_2 \ldots$	$Q = \sum q_n$, $n = 1,..,N$
7	Magnetic	Magneto motive force	Magnetic flux	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Magnetic flux	$M = m_1 \leq m_2 \ldots$	$M = \sum m_n$, $n = 1,..,N$
8	EM/RF	Field	Current	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Field strength	$M = m_1 \leq m_2 \ldots$	$M = \sum m_n$, $n = 1,..,N$
9	Optical	Solid angle	Luminous flux	Energy $E = a \times b$	$E_1 \geq E_2 \ldots$	$E_1 +- E_2 +- \ldots$	Intensity	$I = i_1 \leq i_2 \ldots$	$I = \sum i_n$, $n = 1,..,N$
10	Radioactivity	Half life	Radiation intensity	Dose $R = a \times b$	$R_1 \geq R_2 \ldots$	$R_1 +- R_2 +- \ldots$	Intensity	$R = r_1 \leq r_2 \ldots$	$R = \sum r_n$, $n = 1,..,N$

Ref.	Discipline	Across Variable	Through Variable	Capacity			Behaviour		
		A	b	Parameter	Series	Parallel	State variable	Series	Parallel
11	Thermal	Temperature	Entropy change rate	Heat H = a x b	$H_1 \geq H_2 \ldots$	H_1+- H_2 +- ...	Entropy	$R = r_1 \leq r_2 \ldots$	$R = \sum r_n$, n = 1,..,N
12	Material	Chemical potential	Mole flow rate	Energy E = a x b	$E_1 \geq E_2 \ldots$	E_1+- E_2 +- ...	Mole number	$R = r_1 \leq r_2 \ldots$	$R = \sum r_n$, n = 1,..,N
13	Chemical	Enthalpy	Mass flow rate	Energy E = a x b	$E_1 \geq E_2 \ldots$	E_1+- E_2 +- ...	Mole number	$R = r_1 \leq r_2 \ldots$	$R = \sum r_n$, n = 1,..,N
14	Informatic	Information rate (baud rate)	Memory	C = a x b	Inequality	$\sum C$ max/min	C = a x b	Transfer function (dC/dt)	$\sum C$
15	Transportation logistics	Item/asset flow (units/sec)	Item/asset type	T = a x b	Inequality	$\sum T$ max/min	T = a x b	Transfer function (dT/dt)	$\sum T$
16	Activity/information	Activity/Task flow (Units/sec)	Activity/Task type	A = a x b	Inequality	$\sum A$ max/min	A = a x b	Transfer function (dA/dt)	$\sum A$

Table 29 - Component Representation for Intra-domain Analysis

APPENDIX 4 -

Ref.	Author	Title	Section/P
1	E Rechtin	The Foundations of Systems Architecting	
2a	RA Frosch	A Classic Look at Systems Engineering	Introduction
2b	FT Hoban and WM Lawbaugh Editors	Readings in Systems Engineering	
3	C W Churchman	The Design of Enquiring Systems	
4		Capability Maturity Model Integration	
5	M. W. Maier and E. Rechtin	The Art of Systems Architecting; 2nd Edition	
6		The Riverside University Dictionary,	
7	P Checkland et al	Systems Thinking, Systems Practice	
8	S Pugh	Total Design	
9		Systems Engineering Handbook	Versions 2 and 3
10		English Dictionary - Millenium Edition	
11	M Emes, A Smith and D Cowper	Confronting an Identity Crisis - How to Brand Systems Engineering	
12a	G.J.Klir	Facets of Systems Science	Chapter 2
12b	G.J.Klir	Facets of Systems Science	Chapter 4
12c	G.J.Klir	Facets of Systems Science	Chapter 9
13	H.H.Goode and R.E Machol	Systems Engineering: An Introduction to the Design of Large Scale Systems	
14	R F Miles	Systems Concepts; Lectures on Contemporary Approaches to Systems	
15		Systems Engineering Management	
16	BS Blanchard	Logistics Engineering and Management	Continuous Acquis Supprt
17		Systems Engineering Management Guide	
18		NASA Systems Engineering Handbook	
19	Joint Technical Committee, JTC 1	Systems Engineering - System Lifecycle Processes	

Ref.	Author	Title	Section/P
20		Architecture Framework	
21	G Taguchi	1. System of Experimental Design Edited by D Clausing 2. Introduction to Quality Engineering	
22	L Cohen	Quality Function Deployment	
23	J Hartley	Simultaneous Engineering	
24	E Yourdon	Structured Design Fundamentals of a Discipline of Computer Program and Systems Design	
25	O-J Dahl, EW Dijkstra and CAR Hoare	Structured Programming	Section 1
26	C A R Hoare	Communicating Sequential Processes	
27	B Carre	Graphs and Networks	
28	F W Taylor	The Principles of Scientific Management	On the Art of Cuttir
29a	S Beer	The Heart of the Enterprise	
29b	H G Dallenbach, J A George and D C McNickle	Introduction to Operations Research Techniques	
30		Programme Evaluation and Review Technique	
31a		Automated Interchange of Technical Information	
31b		CALS Implementation Guide	
32		Systems Engineering Data Exchange System	
33	G Pahl and W Beitz	Engineering Design: a Systematic Approach	
34	L von Bertalanffy	General Systems Theory, Foundations, Developments, Applications	
35	W R Ashby	General Systems Theory as a New Discipline	3, 1-6
36	R C Conant	Laws of Information which Govern Systems	On Systems, Mana Cybernetics pp 24(
37	J W Forrester	Principles of Systems	
38	L A Zadeh	Outline of a new Approach to the Analysis for Complex Systems and Decision Processes	On Systems, Mana Cybernetics
39	G J Klir	An Approach to General Systems Theory	
40	M D Mesarovic	Mathematical Theory of General Systems	Pp35-40

Ref.	Author	Title	Section/F
41	A W Wymore	A Mathematical Theory of Systems Engineering	
42a	N L Biggs, E K Lloyd, R J Wilson	Graph Theory, 1736-1936	
42b	F Harary, RZ Norman, D Cartright	Structural Models; An Introduction to the Theory of Directed Graphs	
42c	D B West	Introduction to Graph Theory	2nd Edition
43	L W Beineke, R J Wilson	Graph Connections	
44a	WR Evans	Graphical Analysis of Control Systems	Control Systems S Truxal
44b	J G Truxal	Control System Synthesis	
45	A G J MacFarlane	Engineering Systems Analysis	
46	H M Paynter	Analysis and Design of Engineering Systems	
47	D C Karnopp and R C Rosenberg	Analysis and Simulation of Multi-port systems	
48	N Wiener	Cybernetics	
48a	W R Ashby	An Introduction to Cybernetics	
49	R D Watts	The Elements of Design; The Design Method	
50	L Finklestein	Measurement and Instrumentation Science: an analytical review	Vol 14 Pp 3-14
51	R E Kalman, P L Falb and M A Arbib	Topics in Mathematical System Theory	
52	M R Hestenes	Calculus of Variations and Optimal Control Theory	
53	N Wiener - H Hopf	Wiener-Hopf Equation	See The Statistica Communication by
54	L S Pontryagin, V G Boltyanskii, R V Gamkreligze, E F Mishenko	The Mathematical Theory of Optimal Processes	
55a	G Hadley	Linear Programming	
55b	B S Gottfried, J Weisman	Introduction to Optimisation Theory	
56	R Bellman	The Theory of Dynamic Programming	Dynamic Program the Travelling Sale
57	A G J MacFarlane	Dynamical System Models	

Ref.	Author	Title	Section/P
58	L Finklestein and R D Watts	Measurement as a Systematic Study	Pp 101-105
59	A W Wymore	Model based Systems Engineering	
60	B Cohen, W T Harwood and M I Jackson	The Specification of Complex Systems	
61	C Jones	Systematic Software Development using VDM	
62	K Jackson, H Simpson	MASCOT; A Modular Approach to Software Construction, Operation and Test	See Flow based P Chapter 17.
63	D Harel, A Pnueli, JP Schmidt, R Sherman	On the Formal Semantics of State Charts	State Mate
64	M Alford	Requirements Driven Design	RDD-100
65	D M Buede	The Engineering Design of Systems; Models and Methods	Chapter 1.
66	A Olsen, B Moeller-Pederson, R Reed, JRW Smith	Systems Engineering Using SDL-92	
67	J Arlow and I Neustadt	UML and the Unified Process	
68		The Object Management Group	SysML
69	DODAF/AP233	Integration of DODAF with STEP	
70	A Kossiakoff, W N Sweet	Systems Engineering Principles and Practice	
71	R Stevens, P Brook, K Jackson, S Arnold	Systems engineering; Coping with Complexity	
72		Systems Engineering	
73	H Nicholson (Editor)	Modelling of Dynamic Systems	Vol 1
73a	R G E Franks	Modelling and Simulation in Chemical Engineering	
74	S J Mason	Feedback Theory; Some properties of Signal Flow Graphs	
75	R Billinton, R N Allan	Reliability Evaluation of Engineering Systems	
76	D J Hatley, I A Pirbhai	Strategies for Real-time System Specification	
77	R J Lano	N2 charts and Interface Analysis	TRW Series on Sc
78	J C Boarder	A Practical Introduction to Informal Methods	Software Engineer Education. Editors Ross, G Staples, C

Ref.	Author	Title	Section/F
79	O Becker, J Ben-Asher, I Ackerman	A Method for System Interface Reduction Using N2 Charts	
80		DSM Interest Group internal meeting notes.	
81	N P Suh	The Principles of Design	
82a	T U Pimmler, S D Eppinger	Design Structure Matrix	
82b		Design Structure Matrix	
83	M D Guenov, S G Barker	Application of Axiomatic Design Structure Matrix to the Decomposition of Engineering Systems	pp. 29-40
84	G W C Kaye and T H Laby	Tables of Physical and Chemical Constants	16th Edition
85	F J Evans and J J van Dixhorn (Editors)	Physical Structure in Systems Theory - Network Approaches to Engineering and Economics	Towards More Phy Systems Theory
86	F A Firestone	A New Analogy between Electrical and Mechanical Systems	A 4 pp 249-267
86a	M Vaulot	Sur les Constantes d'un Quadripole Passif	Vol 22. p 493
87	F Strecker and R Feldtkeller	Grundlagen der Theorie des Allgemeinen Vierpols	Vol 6 p. 93
88	L A Pipes	The Matrix Theory of Four-terminal Networks	p. 370
89	S R Deards	The Matrix Theory of Linear two-port Networks	ES451 SRD MJW
90	H A Wheeler and D Dettinger	Measuring the Efficiency of a Superheterodyne Converter by the Input impedance circle diagram	
91	D Karnopp	Power Conserving Transformations: Physical Interpretations and Applications using Bond Graphs	Vol 288 Pp 175-20
92	AK Sumantaray Editor	About Bond Graphs - The System Modelling World	
93	D C Karnopp, D L Margolis and R C Rosenberg	System Dynamics. Modelling and Simulation of Mechatronic Systems	
94	G J Klir	Reconstructability Analysis: An offspring of Ashby's Constraint Analysis	
95	R L Flood, E R Carson	Dealing with Complexity	
96	RB Smith	System Design Environments	
97	W L Chapman, J Rozenblit and A T Bahill	System Design is an NP-Complete Problem	

Ref.	Author	Title	Section/I
98	Korn	Problem of Identity of Systems Engineering	pp 73-83
99	A D Shell	System Implementation and Behavioural Modelling: A Systems Theoretic Approach	Vol 4 No 1
100	H Abbott	Safer by Design	
101		Lifecycle Engineering Management; A Guide to Systems Engineering.	
102	R Spence and R S Soin	Tolerance Design of Electronic Circuits	
103	H J Bremermann	Facets of Systems Science. Edited by GJ Klir	Chapter 8
104	R Rosen	Complexity as a System Property	Vol 3, pp. 227-232
105	C Scholz	The Architecture of Hierarchy	Vol 11, pp 175-18
106	M E Clynes and N S Kline	Cyborgs and Space	Pp 26-27, 7-76
107		Technology Readiness Level	
108	A Wayne Wymore	Model Based Systems Engineering	Paragraph 1.13
109	J T Davis	The Scientific Method	
110	T U Pimmler, S D Eppinger	Design Theory and Methodology	Integration Analys Decompositions
111	L Finklestein	Fundamental Concepts of Measurement	IMEKO VI, Vol.1
112		Software Considerations in Airborne Systems and Equipment Certification	
113	S G J Taylor	Technical Memorandum	
114	S Wolfram	Mathematica	Version 5
115	J W Archibald	Algebra	4th Edition
116	K Kuratowski	Introduction to Set Theory and Topology	
117	H H Rosenbrock	State-space and Multivariable Theory	
118	C E Shannon	A Mathematical Theory of Communication	

APPENDIX 5 - ABBREVIATIONS

Abbreviation	Description
AP 233	STEP Application Protocol for the neutral exchange of systems engineering data
BBs	Building Blocks
BCO/MCO	Avionic Bus and Mission Computer Changeover
C3I	Command, Control, Communication and Intelligence
C4ISR	Command, Control, Communication, Computer, Intelligence, Surveillance, and Reconnaissance
CAD/CAM	Computer Aided Design and Manufacture
Cal Tech	California Institute of Technology
CALS	Continuous Acquisition and Logistics Support
CALS/STEP-AP233	Computer Aided Acquisition and Life-cycle Support/ Standard for Exchange of Electronic Product Data
CAE	Computer Aided Engineering
CASE	Computer Aided Systems and Software Engineering
C-BIT	Continuous built-in Test
CMM/CMMI	Capability Maturity Model/ Capability Maturity Model Integration
CMU	Carnegie Melon University
CP	Codified Products
CRC	Cyclic Redundancy Check
DFD	Data Flow Diagram
DOD	United States Department of Defense
DODAF	Department of Defense Architecture Framework
DSM	Design Structure Matrix
EIA/ANSI	Equipment Industries Association/ American National Standards Institute
ESA	European Space Agency
FACI	First Article Configuration Inspection
FLIR	Forward Looking Infrared
FMECA	Failure Mode Evaluation and Criticality Analysis
FR	Functional Requirement
GST	General systems Theory
HSD	High Speed Data
IAS	Indicated Air speed
ICAM	Integrated Computer Aided Manufacturing
ICD	Interface Connectivity Defintion
IDEF	ICAM Definition Languages
IDEF0	ICAM Definition Language Type 0
IEEE	Institute of Electrical and Electronic Engineers
IFF/SSR	Identification Friend or Foe/ Secondary Surveillance Radar
IFR	Instrument Flight Rules
ILS	Integrated Logistics Support/ Instrument Landing System
IMC	Instrument Meteorological Conditions

INCOSE	International Council of Systems Engineers
ISO	International Standards Organisation
ITU	International Telecommunications Union
LLTV	Low Level light TV
LSA	Lifecycle Systems Analysis
MASCOT	Modular Approach to Software Construction, Operation and Test
MIMO	Multiple Input Multiple Output
MIT	Massachusetts Institute of Technology
MMI	Man Machine Interface
MODAF	Ministry of Defense Architecture Framework
MRI	Material Record Inventory
NCOSE	National Council of Systems engineers
NOE	Nap of the Earth
NVG	Night vision Goggles
OMG	Object Management Group
OO	Object Orientation
OOG	Object Orientation Group
PDS	Post Design Services
PERT	Programme Evaluation and Review technique
PTIT	Engine power turbine inlet temperature
PVIS	Pilot's visual instrumentation system
RDD	Requirement Driven Design, by Ascent Logic
RFS	Requirements, Functional definition and allocation, and Synthesis of implementation
SAE	Society of Automotive Engineers
SDL	Systems Design Language
SE V	Systems Engineering V Process Model Diagram
SEDRES	Systems Engineering Data Repository and Exchange
SEIC	Systems Engineering Innovation Centre, Loughborough University
SI	Systeme Internationale; International System of Units
SISO	Single Input Single Output
SMC	Systems Management and Cybernetics
ST	State Transition structure
STEP	ISO Standard for Product data Exchange
SysML	Systems Engineering Meta language
TRL	Technology Readiness Level
TRW	TRW Automotive Inc. (Thompson Products Inc. with Ramo-Wooldrige)
UC	Universal structure and Coupling
UML	Universal Meta Language
VFR	Visual Flight Rules
VHDL	VSIC Hardware Description Language
VLSI	Very Large scale integration
VMC	Visual Meteorological Conditions
VNE	Speed of sustainable normal wing borne flight
WWW	World Wide Web

APPENDIX 6 - GLOSSARY

Systems Engineering has developed its specialist terminology and many publications include comprehensive glossaries: for example Kossiakoff and Sweet [70], Maier and Rechtin [5], Buede [65], Stevens et al [71], and the INCOSE Systems Engineering Handbook [8].

Therefore, this glossary is confined to those terms that are specific to system design.

Vocabulary of System Design	
Systems Engineering Framework	A Framework is a decomposition of the products, disciplines and processes pertinent to the the total lifecycle of the system application. While the constituents of a Framework will be tailored to suit each particular application, all such Frameworks must ensure that the information and data flows between its members must be comprehensive, sufficient and consistent.
Design structure	A system/sub-system is a collection of components with defined form, fit and functionality that are organised into a structure which, when embodied into its system/sub-system application environment, has emergent properties that provide the user with only the required and tolerable set of functional performance characteristics.
Design space	Each system, sub-system or component shall have its design space defined. Design space may consist of one or more design domains. It defines the fit boundary of Building Blocks.
	Design space is the fit boundary of the objective function(s) within the design space.
	The form of a system, sub-system or component is the design domain which best characterises its principal dimension.
	The fit of a sub-system or components defines all the relationships (interfaces) it has with other sub-systems or components or design spaces or domains.

Components	A functional component is a quantitatively defined entity together with its rules of association, which may be mutable or immutable. Functions shall be defined with single mode dimensions. If more than one dimension is required then a separate function shall be defined for each dimensional entity.
	A Building Block is an implementation component with mature and stable design, and shall be described and assessed in terms of its form, fit and function. Note, mechanical systems use the equivalent concepts of space (relationship), energy and signal (interface). A Building Block may consist of one or more Building Blocks.
Architectures	The structure of a system is called its architecture. The architecture expresses the interconnections of the arrangement of the systems functional components (building bocks) functions and their dependencies.
	A functional architecture is a coherent structure of scientifically quantified entities that together provide the required operational characteristics and performance.
	An implementation architecture is a coherent structure of realisable machine entities which together generate the mutable functional entities of the functional architecture to enable its operational characteristics and performance requirements to be achieved.
	A connectivity network describes the relationship between the components (building blocks) of the system in terms of information flow.
Libraries	A Function Block library is a reference collection of defined functional elements. Each element is in the form of a quantitative expression describing its function and its input - output relationships.
	An Implementation Block/Component library is a reference collection of the building blocks used to generate the mutable functional entities.
	The Primitive library is a reference collection of unique and indivisible elements each of which provides a unique technology or expertise as a building block component.