Literate Modelling — Capturing Business Knowledge with the UML

Jim Arlow[1], Wolfgang Emmerich[2], and John Quinn[1]

[1] British Airways Plc, TBE (E124), Viscount Way, Houslow, UK
Jim.Arlow@btinternet.com
[2] Dept. of Computer Science, University College London, London WC1E 6BT, UK
w.emmerich@cs.ucl.ac.uk

Abstract. At British Airways, we have found during several large OO projects documented using the UML that non-technical end-users, managers and business domain experts find it difficult to understand UML visual models. This leads to problems in requirement capture and review. To solve this problem, we have developed the technique of Literate Modelling. Literate Models are UML diagrams that are embedded in texts explaining the models. In that way end-users, managers and domain experts gain useful understanding of the models, whilst object-oriented analysts see exactly and precisely how the models define business requirements and imperatives. We discuss some early experiences with Literate Modelling at British Airways where it was used extensively in their Enterprise Object Modelling initiative. We explain why Literate Modelling is viewed as one of the critical success factors for this significant project. Finally, we propose that Literate Modelling may be a valuable extension to many other object-oriented and non object-oriented visual modelling languages.

1 Introduction

In working with several large companies such as British Airways, we have become aware of significant deficiencies in visual modelling as it is currently practised. OO was introduced into BA in 1991 [1], and since then there have been about ten significant projects that used Object Technology. CASE tools, such as Rational Rose and System Architect, support easy construction of models which capture business processes and rationale. Once embedded in the model, however, this information becomes somewhat inaccessible. In order to be able to extract the information from the model one generally needs a working knowledge of the visual modelling languages and some knowledge of the operation of the CASE tool. The visual modelling languages that we have used at British Airways include Boochs Design Notation [3] and OMT [9]. We have also used the UML [7] in several large projects at British Airways and since last year, UML has become the standard notation for Object-Oriented Analysis at British Airways.

We gained the experience reported about in this paper when we developed the enterprise object model for the airline. This enterprise object model has been developed during the past two years. The work was started using Booch diagrams and we switched using to the UML subset supported by Rational Rose 4.0 as in Summer 1997. The enterprise object model captures the semantics of concepts, such as flights, segments, aircraft and crew at such a level of abstraction that they can be re-used across different projects of the airline. The enterprise object model is, thus an analysis object model that fully abstracts from implementation details.

Even with UML knowledge, it is often impossible to uncover the important business requirements and imperatives underlying the model, as these become invisible when taken out of business context and expressed in a visual notation. It is quite common for an Analyst or Designer to look at a UML model constructed just a few months ago, and be unable to explain the forces and business requirements that shaped that model. We call this the trivialisation of business requirements and it is not unique to the UML – other visual languages suffer precisely the same defect. The reason for this is that one of the strengths of visual modelling, its conciseness and terseness is also a weakness. Important business requirements may be expressed as a class, relationship, method, constraint or multiplicity, and so become lost amidst other similar modelling elements, which have less business impact. In this paper, we explore this problem from the perspective of the UML and using the successful Enterprise Object Modelling work recently carried out at British Airways (BA) to provide examples. We then go on to describe Literate Modelling, our extension to the UML developed during our work at BA, which can alleviate these problems. This paper is structured as follows. In the next section, we assess how well different groups of people involved in a system development can understand UML. This assessment is based on anecdotal evidence that we gained from using the UML in various substantial and mission critical projects at British Airways. We then argue why the UML tends to trivialise important business requirements and give an example for that. We introduce the concept of Literate Modelling that we have successfully employed to avoid such trivialisation. Literate Modelling is used to produce Business Context documents that are hybrid documents in which different types of UML diagrams are interleaved with carefully produced natural English explaining them. We conclude by indicating requirements for tool support for Literate Modelling.

2 Accessibility and Comprehension of the UML Models

The UML provides a variety of powerful and useful models, and each of these models targets and is accessible and comprehensible to a limited audience. In this context accessibility means “ability to understand the syntax and work the CASE tool”. From our perspective comprehension is the more important issue, and it is often contingent upon accessibility. By comprehension we mean “understanding the business semantics of the model”. Table 1 illustrates comprehension for a logical Analysis model. The table rates comprehensibility of diagrams in analysis models by different stakeholder groups on a scale between 0 and 6. We obtained the figures when we observed comprehension of different stakeholders during structured walkthroughs and presentations of the enterprise object model. Zero indicates the lowest and 6 indicates the highest comprehensibility. At this point, we do not consider the physical models (Component diagrams and Deployment diagrams). We also defer discussion of design models to a later date. We agree with [8] that Activity diagrams would be useful to specify workflow. Because our most commonly used CASE tool does not support them, we have however insufficient experience and data to include Activity diagrams in this discussion. However, we expect that their comprehensibility and accessibility will be quite high – similar to Sequence Diagrams.

	Manager	User	Domain Expert	Analyst	Designer	Programmer
Use Case Descriptions	4	6	6	6	3	2
Use Case Diagrams	3	5	5	6	4	3
Sequence Diagrams	2	2	4	6	5	4
Collaboration Diagrams	1	1	3	6	6	5
Class Diagrams	1	1	2	6	6	5
State Diagrams	0	0	0	4	6	5

Table 1. Comprehensibility of UML Diagrams in Analysis Models

We have considered six common classes of customer for UML models, and rated their comprehension on a scale of zero to six. On this scale zero means no comprehension at all and six denotes full comprehension of a UML model. These estimates are averages based on our private communications with many individuals performing roughly these roles over the course of many different UML projects in many businesses. Clearly, the chart is subjective, but it serves to illustrate several important observations.

2.1 Use Case Descriptions

What is immediately apparent from Table 1 is that Use Case descriptions have the highest comprehensibility. This is because:

1. They are usually written in plain English and so there is no accessibility problem
2. They are often already familiar to non-OO practitioners, as they are just descriptions of business processes from the point of view of the user. It is quite easy to step inside the Use Case Description and role play in order to enhance comprehension

However, one of the problems with Use Cases in general is that they are descriptions of specific business processes from the perspective of a particular Actor. As such they do not give a clear picture of the overall business context and imperatives that actually generate the requirements for these business processes. This means that they can be quite incomprehensible to non-domain experts. We can cite many examples of this from our work at British Airways and other companies where Use Cases, taken individually, make little sense to the uninitiated. This is because there is a business context, such as rules of the air, operational imperatives, interoperation with partners, interoperation with Actors which are legacy systems and standard jargon. This context is not well captured or explained by the Use Case Description or any other UML construct.

More technical Designer and Programmer roles can have problems with comprehension: Understanding a Use Case often requires domain knowledge that the designer or programmer simply may not have. What is disturbing is that the UML provides no formal mechanism (except for the catch-alls of the Note and free-text annotations to diagrams) to capture and present this important information. Embedding this information in the Use Cases themselves is not really an option, as this business context is in many ways orthogonal to any particular business processes.

2.2 Use Case Diagrams

Generally we find these to have similar comprehensibility to Use Case Descriptions, and, although diagrammatic in nature, their accessibility is very high because of the simplicity of the notation. Compared to Use Case Descriptions the Use Case Diagram is semantically weak and we have found that they are not comprehensible without explanation or reference to the Use Case Description. We have therefore given them a lower comprehensibility than Use Case Descriptions for the non-technical roles, Non-Technical Manager, User and Domain Expert. We expect that the Analyst will have sufficient mastery of the CASE tool to navigate to the Use Case Descriptions from the Use Case Diagram, and so there is no loss of comprehensibility. Again, more technical roles, such as Designer, Programmer, may not have sufficient business domain knowledge and knowledge of the business context to appreciate the Use Cases fully.

2.3 Sequence Diagrams

We are now in the realm of object-orientation, and comprehensibility falls sharply for non-OO literate participants. We have found that Non Technical Managers and Users find raw sequence very difficult to follow and they dont really understand the details of object interaction. Comprehension is slightly higher for Domain Experts, as these roles often have some exposure to object-orientation through working with Analysts. Adorning the Sequence Diagrams with scripts increases comprehensibility markedly for this group, but comprehensibility is now of the script rather than the visual model that remains largely obscure.

Because Sequence Diagrams express interaction between objects, Designers and Programmers naturally understand the interactions even though they might not be so sure about the underlying business processes that drives the interaction.

2.4 Collaboration Diagrams

Non-technical roles such as Non Technical Manager, User and Domain Expert typically find these confusing, and there is no reasonable opportunity for adornment with a script to increase comprehensibility. We give these a very low comprehensibility for this audience although again, comprehension is higher for the Domain Expert. However, Analysts, Designers and Programmers find these diagrams useful and comprehensible.

2.5 Class Diagrams

For comprehensibility these require:

1. Some basic OO training

2. Knowledge of UML syntax
3. Ability to use the CASE tool to uncover class and relationship semantics

We have found that comprehensibility of these diagrams is typically very low for Non Technical Managers and Users. It is slightly higher for Domain Experts, as these roles often have some exposure to object-orientation through working with Analysts. Naturally, for the technical group, Analysts, Designers and Programmers, comprehensibility is very high, although we have noticed that many programmers do not have sufficient understanding of UML syntax and object-oriented analysis to fully appreciate them. Hence, we argue that the Programmers comprehension is lower than that of Analysts and Designers.

Analysts will tend to understand the class diagram from the business perspective, Designers will often know less about the business, and their comprehension will be more in terms of object-oriented design issues such as patterns and idioms. Programmers are in many organisations more junior than Analysts and Designers. They will tend to know little about the business and little about good object-oriented design principles. This leads to a lack of comprehension of all aspects of the model, and is particularly dangerous.

2.6 State Diagrams

State diagrams are quite specialised and have a particularly elegant yet terse syntax, which is rarely understood by the non- technical group of Non Technical Managers, Users and Domain Experts. Comprehensibility is effectively zero on our scale for this group. In fact, we have found that it is only really Designers, and not all Designers at that, who have a clear grasp of State Diagrams. The essence of this problem is that we are trying to capture a dynamic system in a static notation. It is our suspicion that State Diagrams will only increase in comprehensibility when they can be executed and animated in the CASE tool.

2.7 The Problem

Several important issues arise from the above discussion:

1. As we move from Use Cases to State Diagrams, the non technical group gradually loses comprehension of the models
2. As we move from Use Cases to State Diagrams the emphasis shifts from a focus on business requirements and imperatives to a focus on the intricacies of object modelling
3. There is a traceability issue. As the non-technical group who understand the business requirements best lose comprehension of the UML Sequence, Collaboration, Class and State diagrams, the traceability of high level requirements to class diagrams becomes more and more problematic.
4. We have found that the technical group Designers and Programmers often have little understanding of the actual business and its needs, and so we cannot rely on them to capture business requirements correctly in their models.

5. Important business requirements become trivialised to classes, relationships, constraints or multiplicity, which may be hidden amongst others of their kind.

We call this process trivialisation, because important requirements are translated into a context in which their importance is no longer apparent. We discuss this issue in the next section.

3 The Trivialisation of Business Requirements by Visual Modelling

Some business requirements are more important than others. Often, there is no way to tell from a blunt statement of the requirement how important it is to the overall operation of the business. We need to see the requirement in its business context to correctly gauge its importance. It is precisely the business context that is lacking in all UML meta models.

As well, in the real world, important business requirements are often highlighted by a certain amount of ceremony – there may be papers, working groups investigating the requirement and discussion at managerial level. This activity is typically how we know that something is perceived to be important to the business. All of this valuable contextual information is absent from the UML model. Although we may have a statement of a particular requirement as part of a UML Use Case, we have no formal mechanism to highlight the importance of this requirement or set it in its true business context. As well, when the requirement is expressed in a class diagram, it becomes a cluster of modelling elements much like any other, and so essential requirements, rather than being highlighted in the visual model, may fade into the background.

We have a specific example from our work at British Airways, which nicely illustrates this point. The 90s have been the age of the global airline, and there has been a great deal of activity forming various alliances so that one partner may sell seating capacity on another partners flight. This is known as codeshare, and is good for passengers, as they can complete a complex journey using a set of co-operating companies. It also improves customer service and can generate new business worth millions of pounds sterling. There is a clear and important business requirement at BA and other alliance partners to support codeshare in its systems. How is codeshare represented in Alliance systems? The key to this is that each flight must have the capability to have many flight numbers. Otherwise codeshare is not supported by the system, and millions of pounds may be lost.

How is the business requirement for codeshare represented in a UML model? Clearly, there will be a Use Case where a passenger travels on a flight which has a BA flight number but which is operated by an Alliance partner and vice versa. Already, we lose sight of the requirement in the sea of Use Cases. In the class diagram codeshare is represented as shown in Fig. 1 below:

So a multimillion-pound business requirement, affecting an alliance of companies together worth billions, is represented as an asterisk on a UML Analysis class diagram. This is exactly and precisely what we mean by the trivialisation of business requirements by visual modelling.

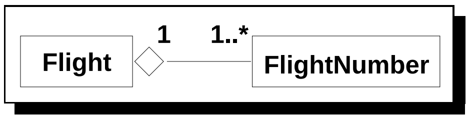

Fig. 1. Example of Trivialisation

4 Literate Modelling

We aim to solve the dilemma of precision and conciseness versus comprehensibility by applying a technique that we refer to as Literate Modelling. Literate Modelling is the application of Knuths idea of Literate Programming [6] to object-oriented analysis models. Similar to Knuths approach we aim at interleaving models with text that explains the model to both the author and externals so that the models can be better appraised and changed. Literate Modelling attempts to address both of the issues we have raised: the accessibility and comprehensibility of the UML models and the trivialisation of business requirements by visual modelling. It does this by providing the missing business context for the models.

We extend the UML by adding new documents, Business Context documents, which explain the model in light of the business context that has generated the forces that have shaped it. Important business requirements are highlighted and unambiguously mapped to parts of the model. As well, important features of the model are discussed in these documents and it is explained why particular modelling choices were made, again always in terms of the business requirements and context.

It is our experience that this provides the missing information that makes a UML model accessible and comprehensible to a wide audience whilst simultaneously resolving the trivialisation issue.

4.1 The Business Context Document

When we had completed the first iteration of the Enterprise Object Model at British Airways, we then had to present these results to a wide audience. This audience ranged from non-technical Senior Managers to Programmers. We even presented it to Alliance Partners, who did not belong to the Airline. We found that we were having great difficulty presenting the standard UML models.

Often our audience had no understanding of UML syntax or of object-orientation at all. We performed model walkthroughs that have been suggested for other Analysis techniques [4]. We took a business perspective highlighting key features of the model that supported specific, key business requirements, introducing the bare minimum of UML syntax to create understanding as we went along. In many cases, we also had to explain the business context and forces that generated these requirements and our realisation of them in UML models. We found that even skilled UML practitioners were not always able to consider a UML modelling element and relate that back to a specific business requirement such as codeshare. The level of detail they were confronted with in relatively short periods overwhelmed the non-technical audience. They regularly missed the important points we tried to make during a walkthrough. We concluded that walkthroughs through UML diagrams are not the right way to communicate with a non-technical audience about essential business requirements. We found that a more permanent representation was needed to explain the business requirements that led to UML models. We decided to extend the UML with Business Context documents that capture this information. We used the following approach:

1. Any UML class model is partitioned into packages, which mapg onto essential, recognised business areas. For example, we might have Products, Accounts and Orders.
2. Within each of these packages, UML class diagrams are created with the criterion that each diagram should "tell a story". By this, we mean that the diagrams should illustrate important business processes and requirements.
3. Business context documents are created which discuss in general terms each of these business areas. They discuss background, general principles and concepts, essential requirements and the forces that shape this part of the business.
4. The Business Context documents are always worded in terms of the classes in one of the class diagrams. For example, if we are writing a Business Context about the Products area, then it will contain phrases such as "Each Product has a PriceList which contains zero or more Prices". It will then go on to explain exactly why this has to be the case, often presenting simple examples. The words in Italics are the names of specific classes on one of the diagrams in the Products package. In a similar way, all attributes, operations and relationships that are discussed in the Business Context are explicitly named in the model, and are always referenced by name. In this way, we tie a high-level description of the business requirements and context directly to the class diagram. This gives a high degree of requirements traceability without introducing formal requirements engineering techniques.
5. UML diagrams are embedded in the Business Context documents as figures.
6. When a UML diagram is discussed, a brief explanation of the relevant UML syntax is included in a footnote. The discussions are carefully constructed so that:
   (a) The non-technical reader can follow them without referring to the diagram.
   (b) The readers with limited object-orientation, database or UML knowledge can glean some understanding of the diagram by referring to the footnotes.
7. Any description of any part of the model is always from a business perspective.
8. Interesting/subtle use of UML and use of design patterns is referenced in footnotes for the technical reader.
9. We have found that all our readers like real, yet simple, examples to illustrate specific points. Again, the example will be couched, wherever possible, in terms introduced in the UML model. If we find we have to use business terms which do not exist in our model, this indicates either
   (a) The example exceeds the scope of the model in some way.
   (b) The scope of our model needs to be expanded.
10. We find that the Class Diagram, Use Case Diagram and Sequence Diagram are quoted most often in the Business Context document. However, in some cases, we have found it expedient to include references to State Diagrams for more technical readers.
11. Informal diagrams can be used liberally wherever they enhance the text.

A good Business Context is actually quite difficult to write as the author must have a very sound and broad overview of the business, excellent UML modelling and communication skills and be highly articulate and literate. One of us is a licensed practitioner of Neuro-Linguistic Programming [5], and was able to bring these skills directly to bear in the creation of the Business Contexts. Neuro-Linguistic Programming is a model of communication, and a set of specific techniques, which facilitate and improve the quality of communication. We find that we get the best results when the Business Context document is lively, involving, direct, provocative, precise, concise and, if possible, humorous. It is, however difficult to incorporate humour well and it should be avoided if in doubt. The passive voice should never be used as it has hypnotic qualities [2], disassociates the reader from the story line of the document and leads to boredom and low comprehension. The purpose of the Business Context is communication, and this requires capturing and involving the attention of the reader. In particular, we have found in many different companies that high-level managers often have quite short attention spans.

4.2 Business Processes, Business Context and Packages

It is well known that a business workflow might cut across the Package and Use Case structure of a UML model. For example, selling a Product will require collaboration between classes in the Products, Orders and Accounts packages as a minimum and will involve many Use Cases. UML Activity Diagrams cut across Packages, Use Cases, and model workflow, in which several Use Cases will participate. Similarly, our Business Context documents cut across Packages and Use Cases. A Business Context document associated with Orders is, primarily, associated with the Orders package, but it actually quotes from diagrams that belong to the Products, Accounts and Orders packages.

4.3 Making Enterprise Object Modelling succeed with Literate Modelling

We believe that the issues of accessibility and comprehensibility are two critical reasons why Enterprise Object Modelling has largely failed so far. Our introduction of Literate Modelling has been shown to successfully address these issues. Literate modelling renders object-oriented notations, such as the UML, useful for Enterprise Object Modelling. It is particularly important in that application domain that the ideas and concepts expressed in a model can be explained and discussed with a large body of non-technical stakeholders. The textual descriptions provided as part of a literate model renders formal UML diagrams understandable and makes them accessible for a review by these non-technical people. The transient descriptions provided during a walkthrough are largely inappropriate for that purpose.

There is a second advantage to using Literate Modelling for Enterprise Objects: the textual explanations of business semantics that result from literate models are extremely precise. When applying Literate Modelling in the Enterprise Object Model of British Airways, we found that the textual descriptions that describe a UML diagram explain business semantics more precisely than those that can be elicited from stakeholders do. A marketing manager at British Airways greeted one of our literate models as the best description of codeshare he had seen ever seen. We attribute this to the fact that analysts develop a very precise and unambiguous understanding of concepts during the development of UML models. This enables them to write very clear and well-structured natural language descriptions, which are largely free of ambiguity.

4.4 Literate Modelling - The Future

We see a bright future for Literate Modelling, and will certainly apply it in future projects. We hope to develop the technique in the following ways:

1. CASE support. At present, the Business Context is maintained in a word processor, and ideally, we would like it managed by the CASE tool
2. Even high level models can rapidly become quite complex. We feel the need to introduce a Business Roadmap document that provides a high-level overview of the Business Context documents, and, as its name suggests, provides a way for interested parties to rapidly find the information they need.
3. We would like to publish the Literate Model on the Internet/Intranet:
    - (a) We would like to hyperlink key words and phrases in the Business Context document directly to the appropriate UML diagrams.
    - (b) We would like to hyperlink the Business Contexts and model to other important documents that may already exist.
    - (c) A search engine would be a useful addition. Then users could look up key words and phrases in the Business Contexts.
    - (d) We would like discussion forums where readers can discuss the model.
    - (e) FAQs (frequently asked questions) have proven to be a very powerful way to lower support costs, so we would like to develop FAQs for our model. We have found that the same questions come up repeatedly. We expect that we will need Business FAQs for each Business Context document and a Technical FAQs for UML modelling in general.
4. Attempt to define a more formal structure for the Business Context document and generate guidelines for writing Business Contexts.

5 Summary

Our initial application of Literate Modelling has been very successful and we believe that the technique addresses accessibility and comprehensibility issues in the UML. The embedding of UML diagrams into explanatory text renders the diagrams comprehensible, even for audiences who are not literate in object-oriented concepts and notation. UML analysts are able to describe business semantics more precisely, once they have formalised it from different perspectives.

Literate modelling is highly advantageous when UML models have to be used as a basis for communication between those with a very detailed understanding of object-oriented concepts and those who are not object-oriented literate. This is particularly the case when object-oriented analysts need to interact with stakeholders that have a non-technical understanding. This type of communication is required during object-oriented analysis, enterprise object modelling and domain modelling.

We conclude by pointing out that we believe that Literate Modelling is also beneficial for visual notations other than the UML. All visual object-oriented analysis notations that we are aware of suffer from precisely the same problems as the UML when they need to be used for communication with a non object-oriented literate. Yet they are all amenable to Literate Modelling as they all produce diagrams of different forms that can be embedded into explanatory text. We suspect that the technique is just as applicable to non object-oriented formalisms, though we suspect that many of these lack some of the expressiveness of the UML and other object-oriented notations. This makes them less suitable for enterprise and domain modelling.

References

1. J. Arlow and M. Phoenix. Introducing Object Technology into British Airways. In *Proc. of the Object Expo Europe 1995*. SIGS Conferences Ltd, 1995.
2. R. Bandler and J. Grinder. *Patterns of the Hypnotic Techniques of Milton H Erickson*. Meta Publications, 1977.
3. G. Booch. *Object Oriented Analysis and Design with Applications*. Benjamin Cummings, 1994.
4. T. de Marco. *Structured Analysis and System Specification*. Yourdan, 1978.
5. R. Dilts. *Applications of Neuro-linguistic Programming*. Meta Publications, 1983. ISBN 0-916990-13-0.
6. D. Knuth. Literate Programming. *The Computer Journal*, pages 97–111, 1984.
7. Object Management Group, 492 Old Connecticut Path, Framingham, Mass. *UML Notation Guide*, ad/97-08-05 edition, NOV 1997.
8. B. Paech. On the Role of Activity Diagrams in UML. In J. Bezivin and P.-A. Muller, editors, *UML 98 – Beyond the Notation, Proc. of the Int. Workshop*, Lecture Notes in Computer Science. Springer, 1999.
9. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.