

original | copy No:

Testing the Predictive Ability of a Requirements Pattern Language

PETER MERRICK peter.merrick@uea.ac.uk

School of Computer Science, University of East Anglia, Norwich, NR4 7TJ

tel: +44 (0)1603 592847

PATRICK BARROW pbmb@cmp.uea.ac.uk

School of Computer Science, University of East Anglia, Norwich, NR4 7TJ

September 18th, 2003

Revised February 4th, 2004

Abstract

This paper looks at a case study in commercial procurement of an IT system to support learners on short educational courses. It compares the Use Case model created before the system was built with the Use Case model after the system was delivered. The original Use Case model was created through the application of a requirements pattern language designed to be employed during the procurement phase of an IT system. The final Use Case model was reverse engineered from the working application. The objective was to discover how accurately the original model represented the final application to provide a measure of the potential usefulness of the pattern language during procurement.

Keywords: procurement, use case modeling, requirements patterns

Introduction

Use Case modelling is a popular technique for representing requirements and is normally employed in the analysis phase of the software engineering lifecycle. UML [1] specifies Use Case modelling as the basis of a development approach where Use Cases act as the input to design and as the basis of verification, validation and testing. Use Cases were first introduced by Jacobson [2] and have been accepted as part of the UML standard. A Use Case is comprised of two elements, a graphical element and a textual element that serves as a natural language elaboration. In this paper, where modelling is undertaken for the purpose of procurement, only the graphical representation is presented. This sub-set of a complete Use Case is what Kulak [3] refers to as a 'façade' which can be thought of as a graphical *table of contents* [4]. A complete description of Use Case Modelling is beyond the scope of this paper; the interested reader is directed towards [3, 5, 6]. In addition, consideration is limited to functional requirements, the 'what does it do' question of system specification that must invariably be answered before considering the question of 'how it is done'. The scope of the enquiry is limited to a comparison of functional requirements before and after the system is built.

Use Case modelling is not the only possible approach that might have been employed to construct a more formal requirements representation to address the perceived shortcomings of natural language [7]. For some practitioners the objective of Requirements Engineering (RE) is to translate informal observations of the real world

into mathematical models [8] through the use of a specialised calculus such as that employed in the specification language Z [9] or Kaos [10]. This approach may lead to models that can be proven but which many stakeholders find difficult to verify. It can be argued that the effectiveness and value of a notation is determined by how well its users are able to work with it. Use Cases are easy to understand [4] where Z may inhibit requirements understanding [11] as it is acknowledged that a formal approach to requirements specification requires appropriate training [12] which customers generally do not have. Use Cases have become a popular choice for requirements expression because they allow wider stakeholder participation in the analysis, primarily because they are expressed in familiar terms [13].

The first step in the process of specification is the identification of high-level goals which are refined until a set of requirements can be identified likely to satisfy those original goals [12]. The UML specification foresees Use Cases as an artefact of the *analysis* stage within a 'Use Case driven' [14] approach to software engineering. This paper seeks to suggest that Use Cases may also have a role to play in procurement [15], a stage of the developmental lifecycle to which they have not typically been put. Although there is some precedence for patterns that describe well-formed Use Cases [16], and patterns of fundamental Use Cases [17, 18], the work presented here, to the authors' knowledge, represents a new perspective on the formation of Use Cases based on the application of requirements patterns.

This case study was undertaken during procurement or 'pre-analysis'; the informal software engineering lifecycle phase for which a specific requirements pattern language (RPL) has been developed. The burden on commercial organisations responding to tenders is an act of weighing the likelihood of success against the cost of producing a comprehensive proposal. The objective of employing patterns to create Use Cases can be characterised as a 'light weight' approach that recognises the constraints inherent in commercial tendering.

The principal question under investigation is whether an accurate Use Case representation can be constructed from a loosely defined customer requirements statement. To this end the customer requirements statement was transformed into a set of pre-implementation Use Case models through the application of the RPL. After the system was built and delivered, the completed application was captured again in a set of post-implementation Use Case models. The models are compared in this paper to test the efficacy of the pattern language in creating models that accurately predict the

form of the final delivered system. It is argued that the degree to which the final models correspond to the initial models will provide some evidence as to the usefulness of the pattern language. The research began with an analysis of the high level requirements contained in the supplier's tender document.

The remainder of this paper is divided into a description of the project being investigated followed by an overview and application of the pattern language including all resulting diagrams both before and after the application was built. Results follow, along with a section on lessons that were learned and suggestions for further work.

A Description of the Project Under Examination

The project under consideration in this case study was undertaken in Norfolk, England, where the local Learning and Skills Council (LSC) went out to tender for the supply of a web-enabled database with end-user and call centre operator interfaces. The Request for Proposal (RFP) was advertised widely in the regional press. There were three separate tender documents in total for the supply of an IT system, the provision of a call centre facility and the undertaking of a marketing exercise. The customer adapted an earlier document which had previously served to secure funding from the national body to produce their requirements statement. Potential bidders were invited to respond to the tender within the deadline with proposals and costs for undertaking the work. One potential supplier, Accent Design, a web-design and programming company approached the authors to enquire whether a response could be written employing UML and Use Cases without entering into detailed analysis. Historically, when invited to tender for IT projects, Accent had relied on building what they termed a 'site skeleton' which is a map of the screens with which the customer would interface to the system. Although this had been successful for smaller projects, the limitations of this approach were identified as being time consuming to produce and difficult to verify as representing a functionally coherent system. To represent the customer's requirements without detailed analysis it was suggested that the requirements pattern language (RPL), already under development, could be employed. The customer's original requirements, as described in their RFP, were taken as inputs to the pattern language. The customer received thirty-eight responses

offering to construct the web-site and database. The Accent submission, featuring the application of the RPL won the tender and they were appointed the supplier.

This paper presents the Use Case models created both before the construction of the system and after the system was delivered.

The Customer Requirements

According to the RFP the proposed system should allow learners to search for and book short courses being run in their area. The LSC sponsors one day 'taster' courses in a wide range of subjects that are open to the general public. The objective of the LSC is to attract as many attendees as possible with a view of supporting the UK Government's stated aim of encouraging the goal of 'life long learning' within the wider community. Courses are run by a range of providers who may themselves have a vested interest in 'showcasing' the educational opportunities they provide. The objective of the course providers is primarily to encourage students who attend the one day event to sign-up to a longer period of instruction which may lead to the achievement of a recognised qualification. Special attention is paid to fostering the involvement of individuals who may not have achieved widely in the past, and who may therefore have a negative association with learning.

Accordingly, the RFP specified that the system would hold course information to include the content of the course, the level of instruction and equipment necessary. A course would be described by the maximum and minimum numbers that could attend along with the current number booked on the course to derive the number of available places. A course is described by the time and date it will run and the location at which it will be held. Course provider details will be represented in the system. The system will be delivered over the internet and will be connected to an email and text messaging engine. Letters will be produced giving course details, for those without internet access, confirming attendance.

The system must provide a comprehensive range of reporting. Reporting should include information on targeting and attendance by attributes such as postcode, course provider, course type, gender and previous learning experience. Reports should be available that represent bookings over different periods of time (day, week, month). Current course numbers should be reported. Information on student personal details should be held for use either by the individual themselves or via a call centre operator.

The above description of the customer's requirements is a précis of the original documentation contained in the RFP and has been written in summary with the intention of preserving its essential meaning.

An Application of the Pattern Language

Patterns capture best practice and facilitate reuse. The history of the pattern movement in software can be traced from Alexander [1] where it was applied originally in the architecture of buildings. According to Alexander, a pattern describes a reoccurring problem in an environment to which a core solution has been proposed which may be applied repeatedly without resulting in the same outcome.

Although the association between non-functional requirements [19] and design patterns [20, 21] is established, that between functional requirements and patterns is less so. The distinction has been made between *software* patterns that are used for implementation and *business* patterns employed in specification. Applying business patterns capable of capturing requirements in a notation that is accessible to users yet rich enough to be interpreted by developers offers a potential way forward [22]. As Use Cases have found favour to express specific requirements, there is reason to believe they will be effective in expressing generalised requirements such as those captured by the pattern language employed in this case study.

A pattern language (or catalogue) is a collection of related patterns that can be used in a non-prescriptive way to achieve some given process.

Use Cases created through the application of the pattern language are constrained to represent user goals at different levels of hierarchical abstraction. A hierarchical approach to the requirements modelling process is found elsewhere in the literature such as in [23]. The process of moving from goal to requirement is described in [10] through the recognition that there exists a meta level of domain specific higher goals that are decomposed into lower level goals which themselves are decomposed into sub-goals. By this approach, a goal that cannot be sub-divided is a candidate requirement (or an assumption) that can be assigned as the responsibility of an individual agent. Equally, the accurate application of the RPL depends on the complexities of Use Case abstraction being understood as they are described according to different axes of both detail and goal [3, 6, 24].

This requirements pattern language (RPL) is comprised of the following patterns: **Minimal Representation**, the **Usual Suspects**, **NewSearchModify (NSM)**, **Informed Management**, **Role Based Assignment**, and **Coherent Whole** to produce an initial Use Case model. The constituent patterns of the RPL are not presented in their entirety due to space restrictions, however an overview is given that introduces the principles governing the construction of each model. A prerequisite to using this pattern language is that a *sufficient* requirements specification exists. This must state the intended purpose of the system, although how well requirements are captured, written and understood will vary from document to document. The quality of the original specification statement will have an impact on the quality of the results, an issue to which the pattern **Coherent Whole** is devoted. The most important prerequisite for success is that the original specification should include a description that includes the nouns that will form the basis of an analysis that will lead to the identification of entities as described by Abbott [25]. The first pattern is **Minimal Representation** which requires the construction of a logical minimal model sufficient to support the business. The process of applying the pattern language relies on the expertise of the practitioner in being able to construct such a minimal data model. To accomplish this task, noun analysis was applied, with the intent of identifying candidate nouns as entities (potential database tables). Care was taken to select only nouns that would represent entities as opposed to nouns that represented attributes of entities [26]. From this analysis the following nouns were identified as candidates for persistent representation; course provider, course, student and booking. One additional entity, course_instance, was included in the representation. This was done to accommodate the fact that a course may run on many occasions and on each occasion it has a unique time, date and location.

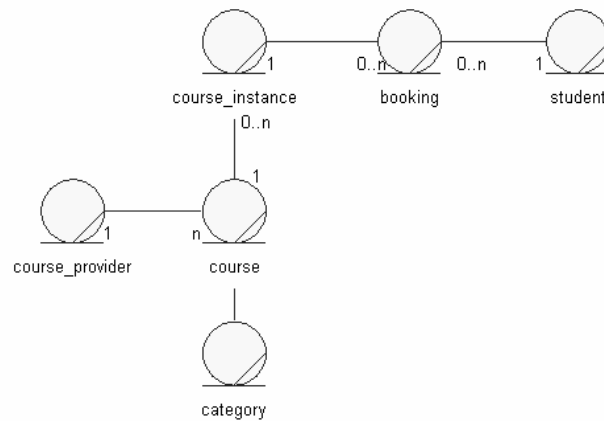


Fig. 1: The output from the **Minimal Representation** pattern is a set of logical entities for the storage of data sufficient to support the business.

Once a candidate data model had been constructed, the next pattern of the RPL can be applied, known as the **Usual Suspects**. The objective of this pattern is to allow for quick identification of roles (known as actors [1, 2]) that have a high likelihood of occurring in a 'transactionally-driven' software application that offers a service to an end-user whose needs are supported by a call centre. The customer's tender conformed to this prerequisite. After the application of the **Usual Suspects**, it is appropriate to apply the **Informed Management** pattern which generates reporting Use Cases, often thought of as providing the functionality associated with a Management Information System (MIS). The pattern **Role Based Assignment** provides a way of grouping Use Cases and representing them in a manner that is logically coherent from the perspective of presentation. The final pattern, **Coherent Whole** provides a check to ensure that the system model generated represents a logical unit whose sum provides recognisable value to its cast of users (actors).

Figures 2 through 11 represent a *before* and *after* model of the system that was developed. All the *before* models were generated at the same time according to the RPL prior to the system being built. All the *after* models were generated through an inspection of the resulting system.

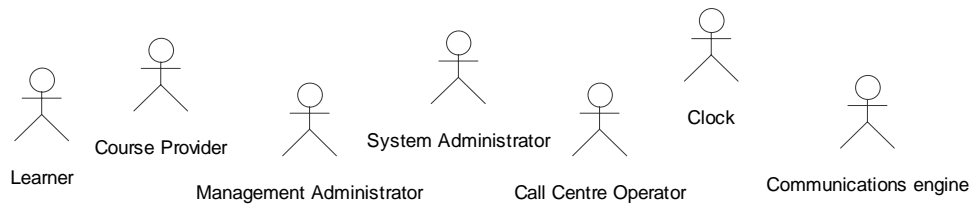


Fig. 2: Diagram illustrating an application of **The Usual Suspects** pattern and the actors that resulted.

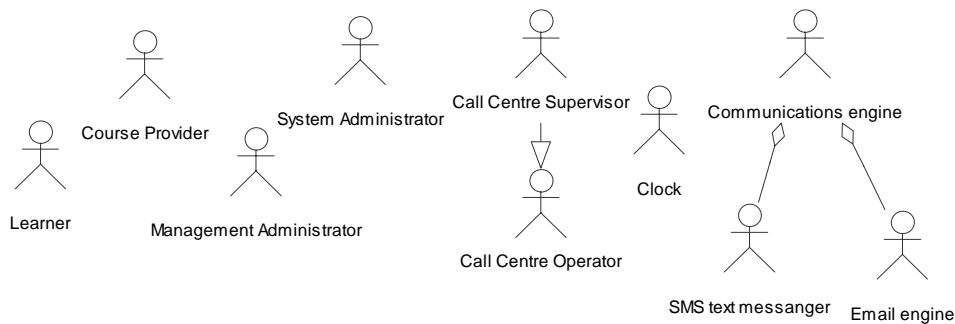


Fig.3: Diagram of actors who appeared in the final system shows strong similarity to those predicted. All the actors identified in Fig. 2 also appear in Fig. 3. In Fig. 3 the primary actor *Call Centre Supervisor* has been added, and more detail is available with respect to the *Communications Engine*. Fig. 3 shows the *Communications Engine* as an abstract actor implemented by a Small Messaging Service (SMS) and email engine. The *Call Centre Supervisor* is an actor that appears in the **Usual Suspects** pattern, and should have been accounted for in Fig. 2 but was omitted due to uncertainty in the original specification as to the sophistication required in the call centre handling functionality.

The RPL specifies that after the application of the **Usual Suspects** pattern, the **New, Search, Modify** (NSM) requirements pattern is applied. This pattern is based on the Create, Read, Update, Delete functionality more commonly known as CRUD [17]. In **NSM**, the concept of ‘Delete’ functionality is subsumed within the concept of modification. According to this pattern, NSM functionality is defined over every logical data entity specified in Fig 1. The complete available set of Use Cases generated by this pattern is given in Table 1.

Data Entity	New <<entity>>	Search <<entity>>	Modify <<entity>>
course_provider	new course_provider	search course_provider	modify course_provider
course	new course	search course	modify course
course_instance	new course_instance	search course_instance	modify course_instance
booking	new booking	search booking	modify booking
customer	new customer	search customer	modify customer

Table 1: Application of the **NewSearchModify** (NSM) pattern over the identified candidate tables generated from the prior application of the **Minimal Representation** pattern.

The fourth pattern, **Informed Management**, is designed to generate management information Use Cases that allow for reporting to take place. According to this pattern, reporting will be based on the entities identified by **Minimal Representation** (see Fig. 1). The bulk of the reports will centre around entities that resolve *many:many* [26] relationships, in this instance, bookings.

With the application of the first four constituent patterns specified by the RPL, the fifth pattern can be utilised to draw relationships between the identified actors and the identified Use Cases. This pattern is termed **Role Based Assignment**. It defines Use Cases on the basis of identified roles. To facilitate this, an abstract actor, *User* is introduced to aid the representation. The pattern is completed by assigning the Use Cases identified by **NSM** and **Informed Management** to the actors identified and specialised by the **Usual Suspects**. The complete results of applying the **Role Based Assignment** pattern is presented over the following pages in Fig. 4 – 11.

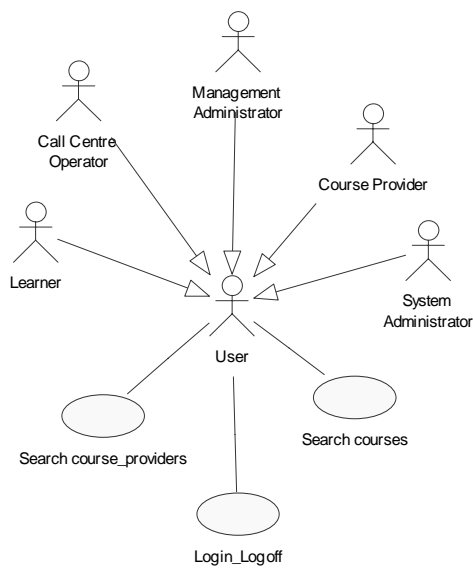


Fig. 4: **General functionality** is one of the categories for representing Use Cases according to the **Role Based Assignment** pattern.

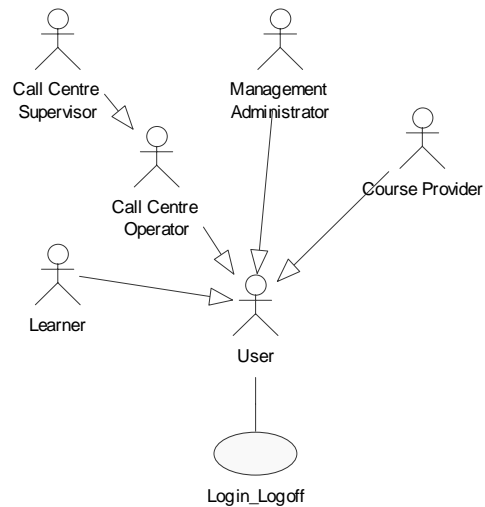


Fig. 5: General functionality after the system was built. No *System Administrator* was required as this functionality was performed without an end user interface.

The differences between Fig. 4 and Fig. 5 are, apart from the representation of the Call Centre Supervisor (accounted for in Fig. 3), the loss of the Use Cases *Search course_providers* and *Search courses*. These Use Cases are still available in the implemented system confined to *Learners* and *Call Centre Operators* (see Fig. 7 and Fig. 9). All the functionality identified in Fig. 4 was implemented in the final system, however the functionality to *Login_Logoff* was the only functionality shared by all users.

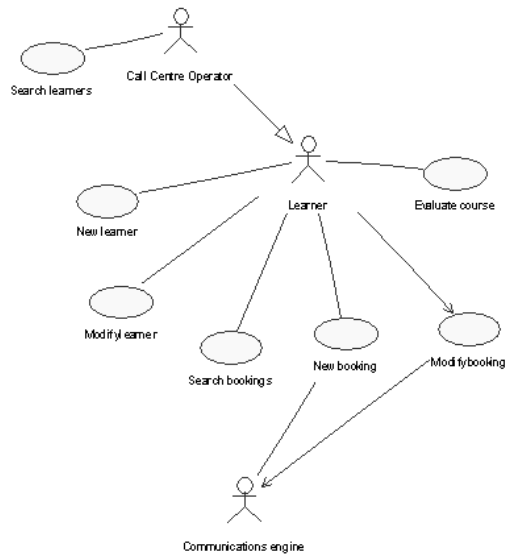


Fig. 6: **Learner functionality** is one of the categories for collecting Use Cases according to the **Role Based Assignment** pattern.

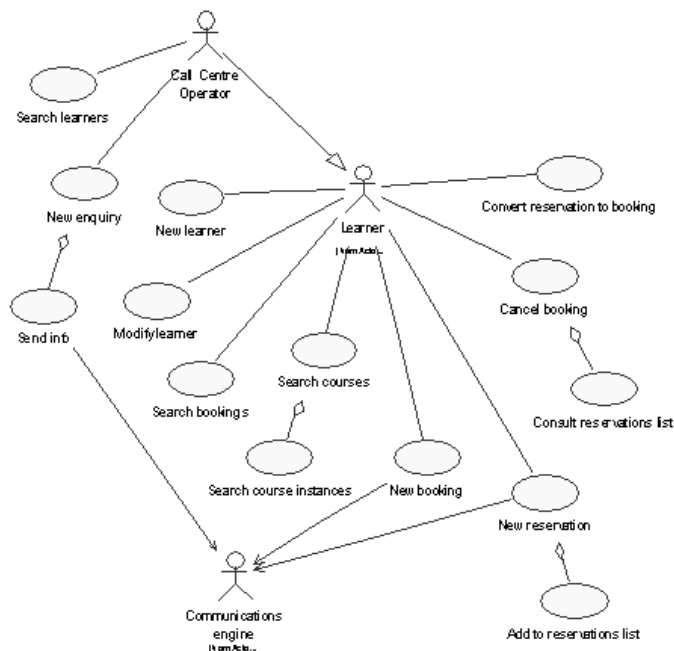


Fig. 7: Learner functionality after the system was built. The diagram features three new user-goal Use Cases [*New enquiry*, *New reservation*, *Convert reservation to booking*]

All the Use Cases defined in Fig. 6 appear in Fig. 7 apart from the ability of a *Learner* to *Evaluate course*, which now appears in Fig. 11. The delivered functionality of a Learner came to include the provision of being able to make reservations which necessitated the addition of *New reservation* and *Convert reservation to booking*. Additionally, *Call Centre Operator* gained the ability to respond to enquiries.

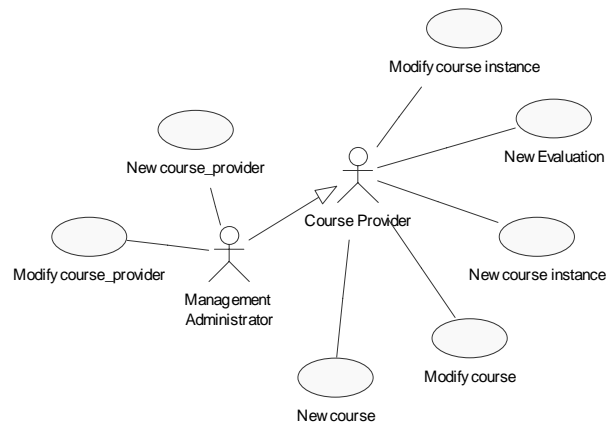


Fig. 8: **Course Provider** functionality is one of the categories for representing Use Cases according to the **Role Based Assignment** pattern.

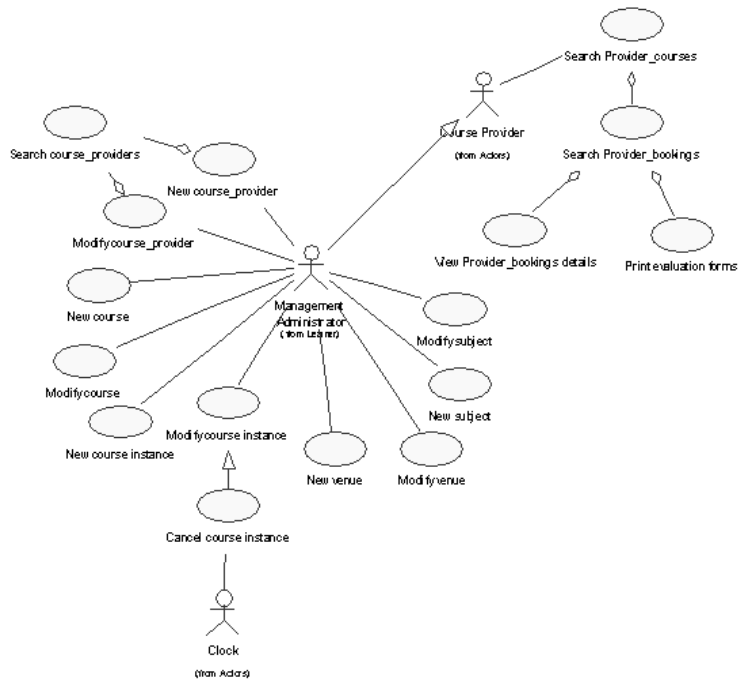


Fig. 9: Course Provider functionality after the system was built. The diagram features five new user-goal Use Cases [*New venue*, *Modify venue*, *New subject*, *Modify subject*, *Search Provider_courses*] alongside some additional sub-goal Use Cases that add detail.

All of the Use Cases in Fig. 8 appear in Fig. 9 however they are not necessarily triggered by the same actor. Assumptions were made as to who would trigger certain Use Cases that were later shown to be incorrect. The functionality itself is the same, but the ‘owner’ of the functionality differs. Much more responsibility is associated with the *Management Administrator* than originally thought. Fig. 9 is a more detailed diagram that also introduces new user-goal Use Cases. Where originally it was assumed that either *Learners* would input their course evaluations directly into the system, or *Course Providers* would input details on the *Learner’s* behalf, in the event both these assumptions were proved false. In the implemented system *Course Providers* are required to send completed paper evaluation forms back to the Management administrator who enters the details onto the system instead. This was a business rule required to ensure attendance and calculate funding.

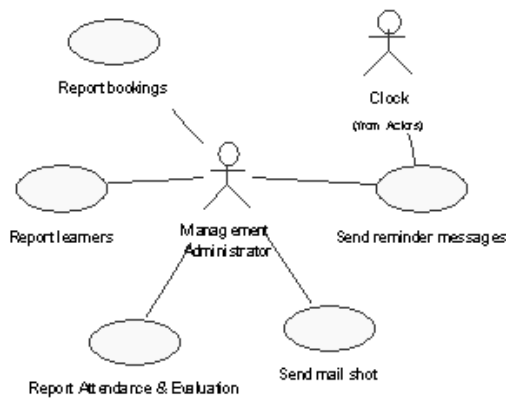


Fig. 10: The Use Cases that appear in this diagram were generated through the application of the **Informed Management** pattern. **Management Administration** is one of the categories for collecting Use Cases according to the **Role Based Assignment** pattern. The Use Case *Send mail shot* was not implemented.

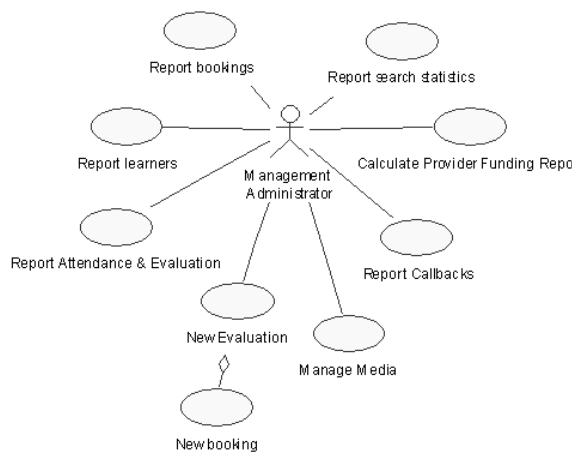


Fig. 11: Management administration functionality after the system was built. The diagram features four new user-goal Use Cases [*Manage Media*, *Report Callbacks*, *Calculate Provider Funding Report*, *Report Search Statistics*]. *New Evaluation* was originally thought to be an action associated with a different actor (*Learner* or *Management Administrator*)

The Use Cases identified in Fig. 10 are all contained in Fig. 11 apart from *Send Mail Shot* which was not implemented after further consideration by the client. Fig. 11 demonstrates that the task of performing the input of course evaluation forms falls to the *Management Administrator*. Additionally, Fig. 11 introduces Use Cases that the pattern language did not foresee; *Manage Media*, *Report Callbacks*, *Calculate Provider Funding Report*, *Report Search Statistics*. The Use Case *Manage Media* is

an aggregate that represents the collective functionality of new, search and modify over the *Media* entity. *Media* became a table in the physical database over which the *Administrator* required direct control (as opposed to an essentially static list). *Report Callbacks* hints at enhanced call centre functionality that escaped representation by the pattern language. Unlike any of the other Use Case models, Fig. 11 was further decomposed to reveal sub-goal Use Cases aggregated by *Report Bookings* and *Report Attendance and Evaluation*. These Use Cases, at the level of user-goals, decomposed to represent available reports that conform to the general description but which offer many specific variations, such as reporting bookings by date range, or by course provider etc.

Results

The results of this case study are based on a comparison of the Use Case models produced before and after the building of the application. It is reasonable to assume that as a project progresses more user requirements will be discovered and those that have been identified will gain more detail. On first inspection of the diagrams it is evident that the ‘after’ diagrams contain a larger number of Use Cases which might lead the observer to a conclusion that the patterns employed to generate the initial diagrams were of only marginal benefit. Instead, before any conclusion is drawn, it is necessary to define the basis of the comparison more rigorously.

Use Cases can be represented at different levels of abstracted goals, therefore it is necessary to agree on the levels being employed to enable the comparison of like with like. According to Cockburn [6], Use Cases exist on a goal centred scale, ranging from ‘very high level’ to ‘too low’ (very high summary, summary, user-goal, sub-function, too low). The choice of *sub-function* as a possible classification option might better be termed *sub-goal* [24]. All of the Use Cases defined in the first modelling iteration feature Use Cases at the level of *user-goal*. In the second modelling iteration, where there are differences, Use Cases have been introduced at both the *user-goal* and the *sub-goal* level. The comparison of the models between the first iteration and the second confines itself to a comparison of user-goal Use Cases and omits sub-goal Use Cases which are treated as an example of greater detail being added to a parent Use Case, as evidenced by their association via an aggregation relationship.

According to Jacobson [2], Use Cases are expressed in the language of the user. One way of interpreting this premise is in the sense that there is one dominant level, and that level is the *user goal* level. Other levels are greater or lesser abstractions of the same Use Case which should contain the degree of detail consistent with the current level of abstraction. Therefore the comparison presented in this case study is made on two accounts. The first comparison is on the basis of whether Use Cases defined in the first iteration remain in the second iteration. The second comparison looks at the models from the perspective of the built system, and compares the number of new user-goal Use Cases that were not predicted in the initial model.

The Use Case models generated before the system was built are a good reflection of the final Use Case models as measured by their ability to predict functionality that would be implemented in the final system. A comparison of Use Cases at the user-goal level of abstraction indicates that only one Use Case predicted from the initial diagrams was not implemented in the final system (Send Mail Shot – Fig. 10).

The final system contained more Use Cases than those predicted by the output from the pattern language. In the final system, there were 30 Use Cases delivered, of which 21 were predicted, or 70%.

Assumptions lead to the mis-assignment of Use Cases to particular actors in the first modelling iteration. This was due to ignorance of business rules that were later clarified. The assignment of functionality to the wrong actors was a relatively minor problem to rectify because of the flexibility inherent in the software architecture which kept permissions isolated from functionality. Of all identified Use Cases, fully 23% were assigned to the wrong actor in the first modelling iteration.

Lessons Learned

The application of the RPL is an effective procedure for generating a model quickly, at an early stage in the process, from a non-structured requirements statement. In this case study the resulting models were useful in clarifying assumptions made in the procurement process with respect to business rules, being the subject of conversation and informal negotiation between the supplier and the customer prior to the award of the contract. There was nothing extraordinary about this case study to suggest the method is not more generally applicable.

Although the authors applied the patterns to the original requirements specification on behalf of the prospective supplier, he reported finding the process clear and understandable and expressed the intention of using it again should the occasion arise. When interviewed, the individual responsible for the selection of the successful supplier volunteered that she believed the models contained in the bid demonstrated the supplier had a firm grasp of the problem to hand, had done a large amount of work in preparation and would therefore make faster progress if chosen to do the work. The winning bid was not selected on the basis of offering the lowest price and was, in fact, the second most expensive bid from a field of thirty-eight who competed for the contract. The reasons the customer cited for selecting the supplier demonstrated an application of value for money principles such as those recommended by the Office of Government Commerce (OGC) for the evaluation of competitive bids [27, 28]. Certain Use Cases, such as those that represent reporting functionality, may be too coarsely grained in their current representation at the abstraction level of *user-goal* and may be capable of prediction at the lower level of *sub user-goal*.

Although Use Cases are strongly associated with the Rational Unified Method (RUP) [29] and object oriented methods, this project suggests Use Cases are useful without adopting these technologies as the project was built by the supplier without any prior experience of UML .

The Use Cases that were not predicted by the pattern language point to the opportunity for the constituent patterns to be improved, although how and where that improvement should be made will not become clear until they have been used more often.

Threats to Validity

This case study can do no more than suggest the method presented has merit. The size of the project undertaken is small compared with many. The programming team was also small, made up of two programmers, which made the need for extensive modelling beyond façade Use Cases unnecessary. The method is dependent on the initial construction of a minimal logical data representation which is dependent on the skill of the practitioner. Also, the entire method has been applied by the authors, and it remains to be seen if other practitioners will find it so successful. The modelling depends on Use Cases being captured at the same level of abstraction. If a Use Case is

decomposed to a level that is closer to implementation, and then included in a comparison, the results will be misleading.

At present too much ambiguity exists with respect to the management of *requirements change*. The distinction between adding a new Use Case that describes additional functionality or simply adding detail to an existing Use Case is captured in the hierarchy of Use Case abstraction but is capable of misinterpretation.

Directions for Future Work

Encouraged by the results to date, work has been undertaken to employ the RPL in a reverse engineering exercise for the Health and Safety Executive (HSE) whereby a model is created from a specification that refers to a recently built application. The model was constructed from patterns in isolation from the existing software and then compared with the final product. The results of this research were similar to those presented here. It would be desirable to assemble a body of evidence where the RPL had been applied to qualify the results that have been reported so far. As many opportunities as possible for the application of the RPL should be pursued to add to the body of evidence. To facilitate this it is necessary that the RPL be sufficiently well articulated to allow other practitioners the opportunity of applying it. In this way the usefulness of the RPL to a wider user base can be tested. Work is ongoing with suppliers to the HSE, on a new project's specification employing the RPL which will go some way to satisfy the objective of testing the method when applied by others. It is generally agreed that requirements change as a project progresses [30]. It is instructive to study the role of Use Cases in wider project management, such as the reporting of progress, and the management of requirements that change.

An interesting enquiry will be the exploration of the relationship between the ability to predict the model of a finished application and the estimation of effort. This work will follow on from that explored in applications of the Use Case Points Method [31-34], an emerging effort estimation algorithm that takes Use Cases as an input. This is an area of top-down effort estimation that could be very useful as existing algorithms for effort estimation [35, 36] are not widely employed [30, 37].

An instructive application of the approach would be to employ the RPL to express functional requirements in a request for proposal that was sent out to a range of prospective suppliers, to understand how supplier organisations respond to a

specification expressed through an application of the RPL, with its attendant effect on assumptions, ambiguity and cost over-runs. The question of whether requirements can be expressed finely enough to avoid ambiguity, yet coarsely enough to allow for evolving detail to be added as the development process proceeds is intriguing. The purpose of attempting to put in place a complete requirements 'table of contents' is to address the problem of scope creep used to justify cost increases and schedule delays. Berry [38] describes this phenomenon as being commonplace in other industries and alludes to the practice taking place in software development whereby a contractor depends on requirements changes to make a profit against an otherwise unrealistic bid.

Bibliography

- [1.] Fowler, M. and K. Scott, *UML Distilled - A Brief Guide to the Standard Object Modeling Language*. 1999, Upper Saddle River, N.J.: Addison Wesley Longman
- [2.] Jacobson, I., Jonsson, P., Christerson, M., Overgaard, G., *Object-Oriented Software Engineering - A Use Case Driven Approach* ACM Press. 1992, Upper Saddle River, N.J.: Addison Wesley Longman
- [3.] Kulak, D. and E. Guiney, *Use Cases - Requirements in Context*. 2000, Upper Saddle River, N.J.: Addison Wesley Longman
- [4.] Fantechi, A., S. Gnesi, G. Lami, and A. Maccari, *Applications of Linguistic Techniques for Use Case Analysis*. Requirements Engineering, 2003. **8**(3): p. 161-170, Springer-Verlag.
- [5.] Pooley, R. and P. Stevens, *Using UML - Software Engineering with Objects and Components* Object Technology Series, G.J. Booch, I; Rumbaugh, J.; 1999, Harlow: Addison Wesley Longman
- [6.] Cockburn, A., *Writing Effective Use Cases*. 2001, Upper Saddle River, N.J.: Addison Wesley Longman
- [7.] Fenton, N. and S.L. Pfleeger, *Software Metrics - A Rigorous & Practical Approach*. 1996, London: Thomson Computer Press
- [8.] Zave, P., *Classification of Research Efforts in Requirements Engineering*. ACM Computing Surveys, 1997. **29**(4): p. 315-321, ACM Press.
- [9.] Spivey, J.M., *The Z notation: a reference manual*. 1989, Upper Saddle River, N.J.: Prentice-Hall
- [10.] Dardenne, A., A. van Lamsweerde, and S. Fickas, *Goal-directed Requirements Acquisition*. Science of Computer Programming, 1993. **20**(1-2): p. 3-50, Elsevier.
- [11.] Khazaei, B. and C. Roast, *The Influence of Formal Representation on Solution Specification*. Requirements Engineering, 2003. **8**(1): p. 69-77, Springer-Verlag.
- [12.] van Lamsweerde, A., *Formal Specification: a Roadmap*, in *The Future of Software Engineering*, A. Finkelstein, Editor. 2000, ACM Press: New York.
- [13.] Regnell, B., M. Andersson, and J. Bergstrand. *A Hierarchical Use Case Model with Graphical Representation*. in *IEEE International Symposium and Workshop on Engineering of Computer-Based Systems*. 1996. Friedrichshafen, Germany: IEEE Computer Society
- [14.] various, *OMG Unified Modeling Language Specification*. Vol. 1.5. 2003: Object Management Group
- [15.] Alexander, I., A. Farncombe, and S. Mohammed. *Towards better railway system specifications through scenarios*. in *The Railway Technology Conference (RailTex)*. 2002. Birmingham http://easyweb.easynet.co.uk/~iany/consultancy/railway_scenarios/railway_scenarios.htm
- [16.] Adolph, S., P. Bramble, A. Cockburn, and A. Pols, *Patterns for Effective Use Cases Agile Software Development*. 2002, Upper Saddle River, N.J.: Addison Wesley Longman
- [17.] Biddle, R., J. Noble, and E. Tempero. *Patterns for Essential Use Cases*. in *KoalaPLoP*. 2001. Melbourne, Australia
- [18.] Biddle, R., J. Noble, and E. Tempero. *Essential use cases and responsibility in object-oriented development*. in *Twenty-Fifth Australasian Computer Science Conference (ACSC2002)*. 2002. Monash

University, Melbourne, Victoria, Australia: Australian Computer Society Inc.

<http://crpit.com/Vol4.html>

- [19.] Mylopoulos, J., L. Chung, and B. Nixon, *Representing and Using Nonfunctional Requirements: A Process-Oriented Approach*. IEEE Transactions on Software Engineering, 1992. **18**(6): p. 483-497, IEEE Computer Society.
- [20.] Beck, K. and W. Cunningham. *Using Pattern Languages for Object-oriented Programs*. in *Object-Oriented Programming, Systems, Languages, and Applications Conference (OOPSLA87)*. 1987. Orlando, Florida: ACM Press
- [21.] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns - Elements of reusable object-oriented software*. 1995, Upper Saddle River, N.J.: Addison Wesley Longman
- [22.] Gzara, L., D. Rieu, and M. Tollenaere, *Patterns Approach to Product Information Systems Engineering*. Requirements Engineering, 2000. **5**: p. 157-179, Springer-Verlag.
- [23.] de Farias, G., c. Ferreira, and P. van Sinderen. *A component-based groupware development methodology*. in *4th International Enterprise Distributed Object computing conference*. 2000. Makuhari, Japan citeseer.nj.nec.com/302797.html
- [24.] Merrick, P. and P. Barrow. *Towards a Requirements Formalism in Procurement*. in *8th Annual Conference of United Kingdom Academy of Information Systems*. 2003. Warwick, England
- [25.] Abbott, R.J., *Program design by informal English descriptions*. Communications of the ACM, 1983. **26**: p. 882-894, ACM Press.
- [26.] Connolly, T., A. Strachan, and C. Begg, *Database Systems - A Practical Approach to Design, Implementation and Management*. 1995, Harlow: Addison-Wesley Longman
- [27.] staff_writers, *Value for Money Evaluation in Complex Procurements*, 2002, Office of Government Commerce (OGC), Norwich,
- [28.] staff_writers, *The Green Book - Appraisal and Evaluation in Central Government*, 2003, HM Treasury, London, The Stationary Office.
www.ogc.gov.uk/sdtoolkit/reference/ogc_library/related/Green_Book_03.pdf
- [29.] Kruchten, P., *The Rational Unified Process - An Introduction*. 2000, Upper Saddle River, N.J.: Addison Wesley Longman
- [30.] Taylor, A., *IT projects sink or swim*. The Computer Bulletin, 2001. **42**(1): p. 24-26, Oxford Journals.
- [31.] Anda, B., D. H. S. D, and J. M. *Estimating Software Development Effort based on Use Cases - Experiences from Industry*. in *4th International Conference on the Unified Modeling Language*. 2001. Toronto, Canada: Springer-Verlag <http://www.idi.ntnu.no/emner/sif8080/docs/faglig/uml2001-anda.pdf>
- [32.] Anda, B., E. Angelvik, and K. Ribu. *Improving Estimation Practices by Applying Use Case Models*. in *4th International Conference on Product Focused Software Process Improvement, Rovaniemi, Finland, December 9 - 11, pp. 383-397, LNCS 2559, Springer-Verlag*. 2002. Profes, Finland http://www.simula.no/publication_one.php?publication_id=500
- [33.] Anda, B., *Empirical Studies of Construction and Application of Use Case Models*, 2003, University of Oslo,
- [34.] Ribu, K., *Estimating Object-Oriented Software Projects with Use Cases*, 2001, MSc., University of Oslo, Oslo, Norway www.stud.ifi.uio.no/~kribu/oppgave.pdf
- [35.] Boehm, B. and E. Horowitz, *Software Cost Estimation with Cocomo II*. 2000: Pearson Education
- [36.] Albrecht, A.J. *Measuring Application Development Productivity*. in *IBM Applications Development Symposium*. 1979. Monterey, CA
- [37.] Kitchenham, B., *Estimation*, in *Software Metrics - Rigorous and Practical Approach*, N. Fenton, Editor. 1991, Chapman + Hall: London. p. 132.
- [38.] Berry, D.M., *Software and House Requirements Engineering: Lessons Learned in Combating Requirements Creep*. Requirements Engineering, 1998. **3**: p. 242-244,