

# On the Representation of an Interactive Workpiece Problem with Requirements Patterns

Peter Merrick and Patrick Barrow

School of Computing Science, University of East Anglia, Norwich, NR4 7TJ  
{peter.merrick, p.barrow}@uea.ac.uk

**Abstract.** This paper describes a pattern language useful in rapidly producing a set of Use Cases for the representation of user requirements. The language is appropriate for problems that fall into the composite frame category of Interactive Workpiece problems. Such problems can also be characterized as representing business applications supported by databases. For an organization interested in institutionalizing a program of reuse, this work is relevant in that it adds to the ability to represent problems such that appropriate solutions may be more easily identified. The pattern language has been applied successfully for the purpose of improving commercial procurement. In this paper, the individual patterns of the language are presented with examples of their realization.

**Keywords:** reusable problem specification, Use Case, workpiece, pattern language

## 1 Introduction

Software project failure is common; faults in the procurement process [1] destine a project to a bad start from which they might not recover. There is a need to produce a 'lightweight' model early on in the contemplation of a new system, for the purpose of inclusion in, or response to, a tender document. In this paper a pattern language for the production of Use Case models is introduced that can be employed over a wide range of business applications. This approach is grounded in the theory that the class of problems under investigation is fundamentally capable of comparison while at the same time providing a strategy for the production of specialized problem artifacts. Much attention has been paid to patterns that describe solutions, whereas this set of patterns is entirely concerned with an elicitation of problems. The purpose of this approach is to allow the practitioner to identify the most appropriate solution from a potentially wide candidate set wherein a well-formed problem statement acts as an unambiguous entry point.

A system can be characterized as a problem/solution pair. To specify a solution it is first necessary to characterize the problem reliably. A solution oriented approach to engineering works better when the problems encountered are well known and classified [2]. Although in the past software engineering problems have not been well classified, recently progress has been made. Jackson introduced a lexicon that characterizes all problems as including entities, events, truths and roles into a problem frame approach [3].

To Sutcliffe, application domains are structured in a layered and numbered hierarchy he uses to highlight where reuse efforts have been most successful [2]. The bottom layer (layer 3 – system support) contains databases and networks, the middle layer (layer 2 – software design) is populated by components, classes and design patterns, whereas the top layer (layer 1 – requirements) contains both a vertical set of business applications (banking, health, education, avionics etc.) and a horizontal set of generic applications (word processors, spreadsheets etc.). Sutcliffe contends that reuse has been much more successful in the bottom two layers but less successful in the top layer [2]. To address this failing, abstraction has an important role to play in aiding problem solving as an invaluable strategy for reuse by providing an entry point to solutions [4]. It has been suggested that all software engineering problems might be capable of expression by a small set of models that represent fundamental abstractions [2]. Sutcliffe suggests that the value of this theory will be demonstrated if it is capable of explaining and predicting phenomena that Jackson defines within a small set of elementary problem frames that are sufficiently abstract to be widely reusable, while still capturing an essential dynamic within a common class of problems [3]. The essence of a problem can be abstracted and characterized as belonging to an elementary class of problem, one of which is the Simple Workpiece frame; useful in creating and manipulating text, graphics or database records. To aid reusability, the frame does not consider the physical design of the workpiece, which exists as a phenomenon within the machine.

From a set of elementary frames, Jackson sees real world problems as being capable of expression through a rich set of composite frames. A composite frame defines a common abstraction all problems of a particular class share. For example, the Interactive Workpiece frame is comprised of an elementary simple workpiece frame and an interactive screen domain. It is this composite frame that describes the class of problem commonly solved by business-oriented database applications (e.g. managing bank accounts, theatre seat reservations or a library); a very common class of problem.

Use Case modeling is a popular technique for representing requirements, normally employed in the analysis phase of the software engineering lifecycle. The UML [5] specifies Use Case modeling as the basis of a development approach where Use Cases act as the input to design, as the basis of verification, validation and testing. Use Cases were first introduced by Jacobson [6] and have since become part of the UML standard. A Use Case is typically made up of two parts, a graphical element and a textual element. Modeling in this paper is constrained to the graphical element, a component of a complete Use Case which Kulak [7] refers to as a 'façade'.

One of the perceived benefits of embracing requirements expression with Use Cases at a consistent level of abstraction is the ability to transform the problem in a manner that suggests a solution. For instance, one well-known approach to solving an instance of the workpiece problem is to employ the Model-View-Controller(MVC) pattern [3, 8]. Sutcliffe defines one of the routes to problem satisfaction as taking a specification, transforming it into a high-level requirements language, followed by a further transformation into code [2]. From a reuse perspective this route could be modified whereby the representation in the requirements language could be transformed into an identification of reusable components followed by parameterization. Sutcliffe sees the major barrier to progress in strategies for reuse in

Layer 1 (vertical business applications) applications being the lack of a requirements specification language that is meaningful to both users and the technical community [2]. The work presented herein is intended as a contribution to bringing down that barrier.

## **2 A Requirements Pattern Language**

The purpose of this pattern is to produce a Use Case model quickly from a natural language business requirements specification. The representation of requirements is often ambiguous. Where natural language is employed in requirements specification misunderstanding can lead to problems and project failure [9-11]. Requirements representation using formal methods [12] may have clear meaning to engineers but at the expense of understandability to customers [2, 13]. The informal diagrams of Use Cases, within the Unified Modeling Language (UML), are less ambiguous than natural language, with the added benefit of retaining their understandability to customers.

In order to create an initial Use Case model from a natural language specification it is necessary to apply the patterns in the requisite order. The patterns are Minimal Representation, NewSearchModify (NSM), the Usual Suspects, Informed Management, Role Based Assignment, and Coherent Whole.

The approach described is driven by the discovery of appropriate entities. The first pattern, Minimal Representation, requires the construction of a logical minimal data model sufficient to support the business. NSM takes the identified data entities from Minimal Representation as an input and automatically generates a set of candidate Use Cases to allow for their management. The Usual Suspects proposes a candidate set of actors that often occur in database systems. Informed Management proposes a set of operational and management information Use Cases. Role Based Assignment provides guidance on how to select from the available candidate Use Cases and assign them to the available actors. Finally, Coherent Whole provides a check that the proposed system defines a problem of sufficient scope to deliver meaningful value to its users. Examples of the application of the Inflatable Façade pattern language in industrial environments are reported in their entirety elsewhere [14, 15].

## **3 Minimal Representation**

The purpose of this pattern is to construct a minimal abstraction of the eventual data structure. The workpiece problems being investigated are primarily concerned with the management of data entities; therefore the challenge is to discover an appropriate set of logical entities that would be sufficient to support the solution. The entities need to be discovered from the customer's business requirements specification or they need to be implied.

### 3.1 Minimal Representation - Solution

The objective of this pattern is to produce a data model that contains a few high-level entities by employing a top-down approach without applying the normal successive refinements necessary to build a physical model. An entity is defined as an identified concept, having an independent existence. It is one of the phenomena Jackson describes as being present in the problem domain [3]. According to logical database design the real world consists of entities and relationships that are thought to naturally exist in our minds [16, 17].

A useful way of identifying entities is to identify nouns. However not all nouns represent entities, and not all entities will initially need to be modeled. Start by identifying common nouns from a natural language description of the problem as found in the business requirements specification. Common nouns name things (customer, course, reservation). Other types of nouns should be ignored except where they allude to a common noun. Another potential type is the proper noun (names a particular instance), which may allude to a class of common noun ('John' may allude to the existence of 'students'). It is appropriate to ignore nouns of quantity (e.g. scoop), collections (e.g. flock), emotions (e.g. sadness), and states (e.g. peace).

What follows is a collection of heuristics useful in determining which nouns will become entities in the minimal data model. Convert all plural common nouns into their singular equivalent. Disregard entities that are outside the scope of the system being modeled. Decide which entities are synonyms and select the most appropriate representation. Disregard weak entities that depend on another entity for their existence (e.g. address depends on customer) except in the case where a weak entity resolves a many to many relationship. It is necessary to introduce new entities as required to resolve any remaining many-to-many relationships (e.g. student and course\_instance are associated via a booking). What remains is a minimal data model.

### 3.2 Minimal Representation - Example

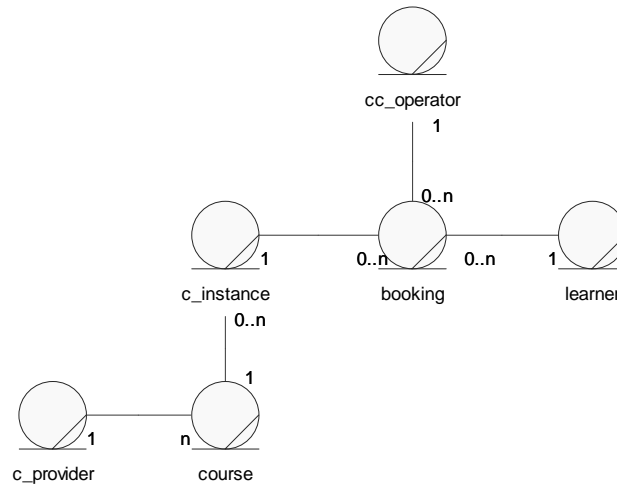
An example specification represented in natural language is presented below. The first time a noun is encountered it is identified in bold text. In Table 1, the identified nouns are represented in the order they appear, along with the heuristics that have been applied in their grammatical treatment for the purpose of deriving a minimal set of entities.

The system should allow learners to search for and make a booking on 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. A range of providers runs courses. 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 that may lead to the achievement of a recognized qualification. The system will hold course information to include the content of the course, the level of instruction and equipment necessary. A course will 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. The system will be delivered over the Internet and will send emails and text messages. Letters will be produced giving course details, for those without Internet access, confirming attendance. The system

must provide a comprehensive range of reports. Current course numbers should be reported. Information on the personal details of students should be held for modification either by the individual themselves or via a call centre operator (cc\_operator).

**Table 1.** Nouns identified in the example business specification text are subjected to the heuristics defined in the pattern to identify candidate entities that will appear in the minimal data model

Input	Heuristic 1	Heuristic 2	Output
system	Outside modelling scope		
learners	singular: learner		learner
booking	common noun		booking
courses	singular: course	common noun	course
area	outside modelling scope	weak entity	course
LSC	outside modelling scope	proper noun	
subjects	singular: subject	weak entity	
public	noun of quantity	synonym student	
providers	singular: provider	common noun	provider
objective	outside modeling scope		
students	singular: student	synonym learner	
event	synonym of course		course
period	weak entity		course
achievement	outside modelling scope		
qualification	outside modelling scope		
information	weak entity		course
content	weak entity		course
level			
equipment	weak entity		course
numbers	singular: number		
places	singular: place	weak entity	
date, time,	weak entities		course
Internet	outside modeling scope		
email, text	weak entity		course
letters	singular: letter	weak entity	course
report(s)	alludes to booking		booking
individual	synonym of learner		
cc_operator	common noun		cc_operator



**Figure 1.** This minimal data model includes all the common nouns identified in Table 1. Through the process of further investigation new and important information may be discovered. In this example it emerged that a course may be run on many occasions necessitating the inclusion of a course\_instance (c\_instance). Course\_instance was not a noun in the original text, although its existence was alluded to within the concept of course. Other entities are represented with the prefix ‘c’ to indicate ‘course’

The differentiation among types of noun is a matter of semantics and not a simple syntactic distinction [17]. For instance booking may be a verb or a noun demonstrating that it requires knowledge of the real world to which the domain of enquiry applies to construct the model. There is no mechanical process for automatically transforming an informal statement of natural language into a formal collection of entities. Even with simple problems it may be necessary to add entities that do not appear in the original business specification. Consider aggregation, whether the nouns identified are components of a greater set, or are themselves a set that could be usefully decomposed. Building lightweight minimal entity models requires being familiar with how to build normalized relational databases.

## 4 New, Search, Modify (NSM)

The purpose of this pattern is to generate candidate Use Cases for the management of the data entities identified by Minimal Representation.

### 4.1 NSM - Solution

This pattern generates three candidate Use Cases for each data entity; New <<entity>>, Search <<entities>> and Modify <<entity>>. Other authors have suggested that the identification of functionality based on data entities is likely to

produce a valid candidate set [18, 19]. This kind of functionality is commonly termed Create, Read, Update, and Delete (CRUD).

It is sometimes preferable to manage a NSM Use Case set with the single tag Manage <<entity>> although this may prove more suitable for Use Case representation at the summary level of representation [18]. Every entity is explicitly shown as being subject to a new instance creation, the modification of an existing instance, and the searching for a single instance or a multiple instance set. Complex behavior that requires searches over more than one entity are often found in reporting Use Cases, created in the Informed Management pattern.

The result of this pattern creates a candidate Use Case set; it is not necessary to include all Use Cases identified by NSM in the resulting models. The decision to include a Use Case is a matter of judgment that is informed by considerations such as the dynamic nature of a particular entity (how often it changes) and the requirement to provide a user interface for the entity's management. For example, entities that do not change often and therefore do not required to be actively managed need not be included.

#### 4.2 NSM – Example

**Table 2.** Application of the NSM pattern over the identified candidate tables generated from the prior application of the Minimal Representation pattern

Data Entity	New <<entity>>	Search<<entity>>	Modify <<entity>>
c_provider	new c_provider	search c_provider	modify c_provider
course	new course	search course	modify course
c_instance	new c_instance	search c_instance	modify c_instance
booking	new booking	search booking	modify booking
learner	new learner	search learner	modify learner

The objective of this pattern is to allow for a quick identification of roles (known as Actors [5, 6]) who have a high likelihood of occurring in a business-oriented software application that offers a service to an end-user, sometimes supported by a call centre.

## 5 Usual Suspects

The objective of this pattern is to allow for a quick identification of roles (known as Actors [5, 6]) who have a high likelihood of occurring in a business-oriented software application that offers a service to an end-user, sometimes supported by a call centre.

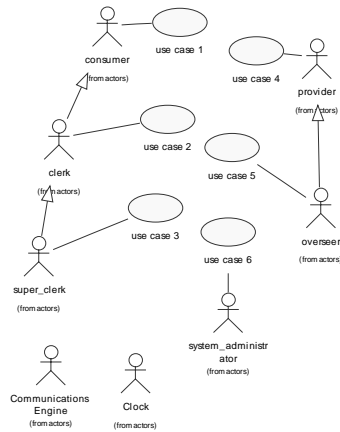
### 5.1 Usual Suspects - Solution

The type of system being specified can be characterized as featuring a consumer, a provider and an administrator of products or services. Consumers may interact with the system directly, or proxies may aid them in their interaction. The clerk and superclerk (via specialization) act as proxies for the consumer. Superclerks are managers who supervise the performance of clerks. The system administrator has responsibility for setting up other individuals in their role as particular actors and

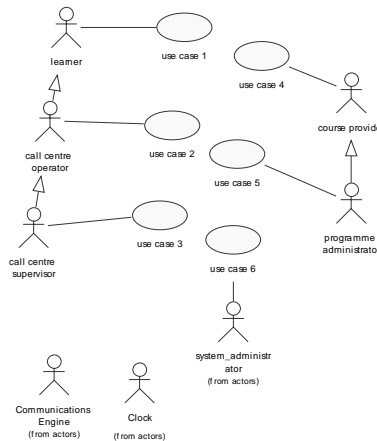
controls the setting of parameters that cause the system to behave in pre-defined ways.

The system is said to exist to serve the needs of the consumer. The provider is responsible for the definition of the product or service to which the system gives access. The administrator generally controls the creation and sets the parameters of behavior followed by providers. The administrator relies on different kinds of reports to monitor the performance of the system. Clerks and superclerks are often specialized as call centre workers. It is not necessary to specify all of the actors defined in the pattern where the specification does not call for them. The system will likely include a clock to trigger automated behavior (such as the generation of reports) and a communication mechanism, such as an email and text-messaging interface.

### 5.2 Usual Suspects - Example



**Figure 2.** 'Usual Suspects' Actors and their relationships that are useful in the specification of permissions



**Figure 3.** A specialization of the Usual Suspects pattern introduced in Fig. 2

This classification system has the effect of defining roles that will control permissions to access different functional behavior. This fact is used to good effect in Role Based Assignment.

## 6 Informed Management

Managers need to understand how well the business is performing. For example, Administrators need to know how many consultations have been performed on this day (operational report) and additionally, how this compares with the number of consultations performed in the same period one year previously (management report). The super clerk needs to know such things as how many calls a particular clerk handled, and how long that clerk typically took on a call.

This pattern specifies that reports are required and makes the distinction between operation and historical reports. Reports that make predictions may be required but they are not included in this pattern.

### 6.1 Informed Management - Solution

At an early stage in modeling based on a specification it is unlikely that sufficient evidence will be available to define reports with a high degree of accuracy. Reports will be primarily concerned with the entity that is consumed within the system. This is likely to be a weak entity, dependent on two or more strong entities that solves what would otherwise be a many-to-many relationship. In the example, the consumed entity is the entity booking. There will, however, be many different flavor of booking report that combines information from one or more of the other entities.

### 6.2 Informed Management - Example



Figure 4. This diagram indicates the need for reports on bookings is recognized

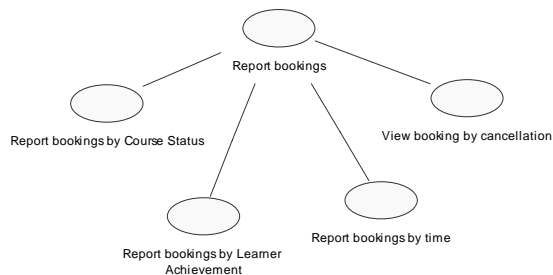


Figure 5. This diagram recognizes that when Report Consultations is realized there will be many specializations. Report Consultations is an abstract Use Case

## 7 Role Based Assignment

Thus far the specifier has identified a set of candidate actors and a set of candidate Use Cases. It is now necessary to combine the elements to produce the initial Use Case models. It is not likely all the actors and Use Cases will fit on a single diagram; therefore multiple diagrams must be produced.

### 7.1 Role Based Assignment - Solution

To assign Use Cases to actors, produce diagrams based on consumption, provision, administration of service, administration of proxy, and system administration. It is useful to add to this set a general class of function to which there are no permissions or restrictions such as *log\_in* and to which may be added search functionality as appropriate to the satisfaction of business rules.

7.2 Role Based Assignment - Example

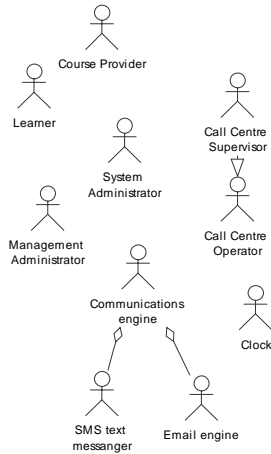
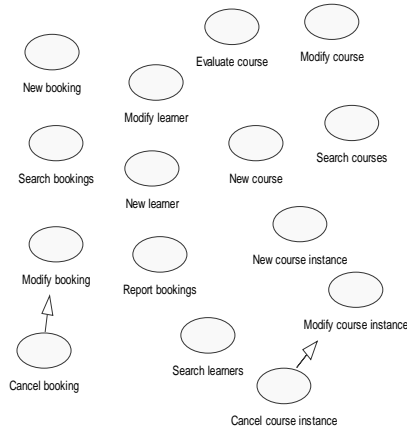


Figure 6. Many Use Cases are generated by the pattern NSM and Informed Management that must be assigned to actors

Figure 7. Many actors are generated by the pattern Usual Suspects that must be assigned to Use Cases

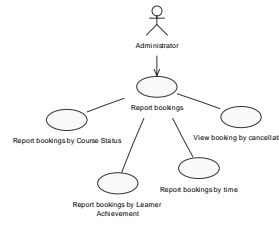
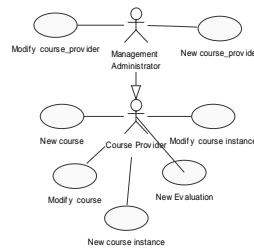
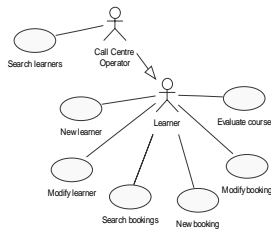


Figure 8. An example of a Consumption diagram featuring the Consumer and his/her proxy in the maintenance of their own record and delivery of the system's public purpose

Figure 9. An example of the Provision diagram features the Administrator and the Provider in the definition of the service

Figure 10. An example of the Administration of service features the Administrator and the assignment of Use Cases first introduced in Figure 5

It is useful to classify identified entities, for instance booking can be termed a consumption entity, while course and course\_instance can be termed provider entities. Depending on the business rules in force, assign New/Modify <<consumption\_entity>> to the consumer, New/Modify <<provider\_entity>> to the provider, and Report <<consumption\_entity>> to the Administrator. In any response to a tender document all assignments should be checked with the customer, where possible, and any unverified assumptions clearly stated, as this task is equivalent to the granting of permissions based on roles.

## 8 Coherent Whole

This pattern is the least formulaic. It exists to provide a check that no functionality that might have been identified has been missed. It is essential to try and ensure that the system under consideration will deliver a complete and recognizable service. It is not uncommon for customer and supplier to disagree regarding the classification of new information on the subject of functionality [20]. Once a project begins, it can be classified as new functionality, for which an extra charge should be made, or as providing more detail for which no charge is appropriate. Systems of a certain type may be expected to exhibit certain functionality based on an assessment of past work, regardless of whether it is overtly specified. For example, the fact that a ‘home build’ project does not explicitly define the necessity for a roof should not imply that no roof is required.

It is a certainty that requirements will be further clarified and in many instances will change [10]. Therefore it is important to define a process for requirements change management. The question arises, ‘when is a change a change?’ A ‘change’ implies something has been implemented that must be ‘made different’ and for which a charge will be made; otherwise what is communicated is a clarification. The question of whether new information implies a chargeable change or new detail to an existing specification is profoundly important. However, it is reasonable to expect that additional behavior should be paid for, with the caveat that the supplier has a responsibility to help the customer commit to a complete specification based on experience and the norms of the industry in building applications that can be considered equivalent.

### 8.1 Coherent Whole - Solution

Check with potential users that the system, as proposed, will accomplish the high-level goal they seek to perform. Show the responsible project owner [21] a Use Case model and invite them to sign it off. The process of getting agreement will provide a high degree of assurance that the model represents what is required. If it is difficult to persuade the project owner as to the value of the model, this indicates the model is defective. Explore with the responsible owner a variety of scenarios [22], based on Use Cases, designed to test the completeness of the proposed problem definition and to ensure the process for requirements change management is understood by all parties.

## 9 Results

It is reasonable to assume that, in any project that attempts to predict functionality, some requirements will be missed, and some entirely new requirements will be introduced. The pattern language does not predict all the Use Cases in the final system, but it does identify a high proportion. Tests carried out thus far, for the Health and Safety Executive and the Learning and Skills Council, have been successful in predicting approximately 70% of the total functionality of the built system [14, 15]. In addition, virtually all the Use Cases originally predicted have been included in the

final design. This suggests the approach has value and that problems of this class share phenomena that are inherently comparable. There are very many factors that have an impact on how well the pattern language can predict the final shape of a system which include the completeness of the original business specification, the ability of the practitioner, the degree to which the problem conforms to the Interactive Workpiece problem frame and the degree to which new functionality is requested during the lifetime of the project. With that said, it does appear to be a useful contribution to making problem representation less ambiguous and thereby offering a model against which the components of a reusable solution can be identified.

Some improvements in this first version of the pattern language are inevitable. For instance, the pattern Role Based Assignment has lead to many Use Cases being assigned to the wrong actor. However, this has not been particularly serious as it is a problem that is simple to correct and appears to reflect incorrect assumptions with respect to business rules that are later clarified in conversation with the customer.

Customers have found the pattern generated Use Case approach superior to natural language expression and have easily understood the Use Case notation. One customer stated their decision to choose a supplier was strongly influenced by the inclusion of models in the tender response as it demonstrated a firm grasp of the problem to hand [14].

## 10 Further Work

One of the stipulations for the implementation of good practice in IT procurement published by the Office of Government Commerce (OGC) states the objective of unambiguously capturing the scope and requirements of the system it is intended to purchase [11]. This has been identified as a weakness in many past projects that have gone far over budget. To address the shortcoming of poor problem specification the OGC is cooperating with the authors in seeking volunteers to represent their requirements via the pattern language in a public tender. This work is of interest to observe how the major IT suppliers who are invited to respond react to this style of specification. Ultimately the pattern language's efficacy will be tested through quantitative research that shows it has an impact on reducing cost and schedule overruns. The publishing in OGC newsletters and communications of the civil service for participants in further research is a step in the right direction aimed at building up a body of evidence.

## 11 Bibliography

- [1.] staff, *Why IT Projects Fail*, 2002, Office of Government Commerce, [http://www.ogc.gov.uk/sdtoolkit/reference/ogc\\_library/bpbriefings/it\\_projects.pdf](http://www.ogc.gov.uk/sdtoolkit/reference/ogc_library/bpbriefings/it_projects.pdf)
- [2.] Sutcliffe, A., *The Domain Theory - Patterns of Knowledge and Software Reuse*. (2002), London: Lawrence Erlbaum Associates, Inc.
- [3.] Jackson, M., *Problem frames: analysing and structuring software development problems*. (2001), Harlow: Pearson Education

- [4.] Sutcliffe, A., *Domain analysis for software reuse*. Journal of Systems and Software, (1998). **50**: p. 175-199, Elsevier.
- [5.] Fowler, M. and Scott, K., *UML Distilled - A Brief Guide to the Standard Object Modeling Language*. (1999), Upper Saddle River, N.J.: Addison Wesley Longman
- [6.] 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
- [7.] Kulak, D. and Guiney, E., *Use Cases - Requirements in Context*. (2000), Upper Saddle River, N.J.: Addison Wesley Longman
- [8.] Krasner, G.E. and T., P.S., *A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80*. Journal of Object Oriented Programming, (1988). **1**(3)
- [9.] Standish, *The Chaos Report*, 1994, The Standish Group,  
[http://www.standishgroup.com/sample\\_research/chaos\\_1994\\_1.php](http://www.standishgroup.com/sample_research/chaos_1994_1.php)
- [10.] Taylor, A., *IT projects sink or swim*. The Computer Bulletin, (2001). **42**(1): p. 24-26, Oxford Journals.
- [11.] staff\_writers, *Why IT Projects Fail*, 2002, Office of Government Commerce,  
[http://www.ogc.gov.uk/sdtoolkit/reference/ogc\\_library/bpbriefings/it\\_projects.pdf](http://www.ogc.gov.uk/sdtoolkit/reference/ogc_library/bpbriefings/it_projects.pdf)
- [12.] Spivey, J.M., *The Z notation: a reference manual*. (1989), Upper Saddle River, N.J.: Prentice-Hall
- [13.] Khazaei, B. and Roast, C., *The Influence of Formal Representation on Solution Specification*. Requirements Engineering, (2003). **8**(1): p. 69-77, Springer-Verlag.
- [14.] Merrick, P., *Testing the Predictive Ability of a Requirements Pattern Language*. Requirements Engineering (accepted for publication), (2004) Springer-Verlag.
- [15.] Merrick, P.: *Testing a Requirements Pattern Language Through Reverse Modelling*. in *INCOSE 2004*. (2004). Toulouse, France: accepted for presentation
- [16.] Chen, P., *The Entity-Relationship Model - Toward a Unified View of Data*. ACM transactions on Database Systems, (1976). **1**(1): p. 9-36, ACM.
- [17.] Abbott, R.J., *Program design by informal English descriptions*. Communications of the ACM, (1983). **26**: p. 882-894, ACM Press.
- [18.] Cockburn, A., *Writing Effective Use Cases*. (2001), Upper Saddle River, N.J.: Addison Wesley Longman
- [19.] Armour, F. and Miller, G., *Advanced Use Case Modelling*. (2001): Addison Wesley
- [20.] Berry, D.M., *Software and House Requirements Engineering: Lessons Learned in Combating Requirements Creep*. Requirements Engineering, (1998). **3**: p. 242-244,
- [21.] staff\_writers, *The OGC Gateway Process*, 2004, Office of Government Commerce, London, [www.ogc.gov.uk](http://www.ogc.gov.uk)
- [22.] Alexander, I., *On Abstraction in Scenarios*. Requirements Engineering, (2002). **6**(4): p. 252-255, Springer-Verlag.