Software Qual J (2007) 15:69–97
DOI 10.1007/s11219-006-9006-z

Assessing the quality of use case descriptions

Keith Thomas Phalp · Jonathan Vincent · Karl Cox

Published online: 17 January 2007

Abstract Use cases have, for some years, been a popular approach to specification, as part of the Unified Modelling Language (UML). However, a number of authors have pointed to weaknesses with the approach, particularly in terms of the support offered to the writer of the use case description. This paper describes a *Use Case Description Quality Checklist* that acts as a check on the quality of the written description. The checklist is derived from theories of text comprehension, taken from the Discourse Processing community. The checklist approach has a number of benefits. First, the approach can be used to derive, or examine further, use case guidelines. That is, by considering whether such guidelines are likely to result in desirable qualities within the resulting description, one is able to make an informed judgement about the utility of those guidelines. Second, one can test for the desirable quality features in existing descriptions, thus enabling empirical validation. Third, as a minimum, the quality features can themselves be used as a checklist for the examination, and revision, of use case descriptions. To demonstrate applicability, the paper reports upon the use, and success, of the approach on an industrial case study.

Keywords Use case · Use case description · Text comprehension · Requirements · Specification · Validation

1 Introduction: Use case descriptions and the need for guidelines

The use case model first became popular with its introduction by Jacobson et al. (1992) as a means to describe system specification in a natural and understandable way, and one that

K. T. Phalp · J. Vincent (✉)
Software Systems Modelling Group, Bournemouth University, UK
e-mail: jvincent@bmth.ac.uk

K. T. Phalp
e-mail: kphalp@bmth.ac.uk

K. Cox
National ICT Australia, Australia
e-mail: karl.cox@nicta.com.au

focused on systems from a user perspective. With its adoption into the UML (Booch et al., 1999), the use case soon became widely adopted in industry, and it remains the dominant approach to requirements description and specification within the UML.

Whilst early discussions of the use case focused primarily on the diagram semantics, in recent years attention has turned to supporting the use case description (e.g. Achour et al., 1999; Cockburn, 2001; Adolph et al., 2003; Somé, 2005). Given that the detail of the use case description typically forms the specification for systems, this shift in focus seems appropriate. However, for such an important document, there is little guidance as to what constitutes a good use case description, save for a general agreement that readability is paramount. For example, Cockburn (2001) states: "*Ultimately a use case, or any specification, will be read by people. If it is not easily understood, it does not serve its core purpose. You can increase readability by sacrificing some amount of precision and even accuracy, making up for the lack with increased conversation. Once you sacrifice readability, however, your constituents won't read use cases*."

Although readability is important, there must clearly be other factors to consider in judging the merits of use case descriptions, yet there is surprisingly little advice on this important matter, and little to guide the writer. For example, one still needs to have precision in specification, as there are risks of ambiguity inherent in natural language (Gause and Weinberg, 1989). However, moving from natural language to more formal constructs for scenarios, as expressed, for example, by Hsia et al. (1994) makes customer validation difficult because they are unlikely to be familiar with formal logic (Ham, 1998). Therefore, a pragmatic approach is to keep the natural language of use cases, and to provide some guidance to use case writers so that their descriptions are both comprehensible to their audience, and sufficiently rigorous for specification.

Recent years have seen a growth in this approach, and a number of authors have suggested a variety of guidance for the writer of the use case. Adolph et al. (2003) present patterns for use case descriptions that provide a high level view of the use case structure, and some advice on how to construct individual sentences; such as giving the actor's name and avoiding "*use case horrors*" such as passive voice and implementation detail. Other authors have posited similar guidance on writing descriptions, such as recommending the use of strong verbs and nouns (Kulak and Guiney, 2000) and avoiding passive voice (Pooley and Stevens, 1999). Others again suggest structural guidance. For example, Fowler and Scott (2000) and the OMG (2001) recommend writing primitive steps and numbering events.

Whilst these guidelines may seem sensible, they tend to represent disparate and implicit assumptions about quality or understandability in text comprehension. That is, it is assumed that a particular grammatical construct will be beneficial, and hence a rule is added. Similarly, there is no agreed consensus, and some rules proposed even appear to be contradictory. Therefore, this paper aims to examine the breadth of use case guidance presented, to explore the underlying (theoretical) rationale for guidelines, and to consider related work in text comprehension, in order to produce a view of use case quality that takes in a number of perspectives. These perspectives are then categorized, to produce a digestible and understandable set of rules that can be used to assess, and subsequently guide, use case descriptions.

The remainder of this paper is organised as follows. Section 2 reviews existing work on grammar structures, including the genesis of those presented within this paper. Section 3 presents an overview of work from discourse process research, which aims to understand the process of text comprehension. Section 4 attempts to set this work within the use case context, using a meta-model that relates key aspects of text comprehension to use case descriptions. Section 5 collates this work into a set of guidelines for assessing communicability. Section 6 explores the application of the guidelines to two use case descriptions drawn from an industrial case study. Section 7 discusses the findings, reviews recent work on guidelines and offers suggestions for further work.

2 Rules and guidelines

Given that use cases are in essence a form of structured English it is perhaps unsurprising that one source of guidance is that of grammatical rules, and some authors have suggested that specific grammar constructs should be used. Graham (1998, p. 141) suggests that "*sentences can be arranged in the form: Subject Verb Direct object Preposition Indirect object*(*s*)". Cockburn (2001, p. 90) gives one main writing structure:

> *"The sentence structure should be absurdly simple:*
>
> *Subject... verb... direct object ... prepositional phrase.*
>
> *For example:*
>
> *The system... deducts... the amount... from the balance account.*
>
> *That's all there is to it. I mention this matter because many people accidentally leave off the first noun, making it no longer clear who is controlling the action."*

However, whilst they may seem logical, there has been relatively little investigation of the actual (practical) usage of such structures. In one survey, conducted by Cox et al. (2001), it was found that the structures suggested by Cockburn and Graham (if we take them to be roughly equivalent) were used in 6% of sentences in main flows of events (from over 1900 sentences in 152 use case descriptions, of which 120 are published in the literature). The only structure to occur more than the above was Subject Verb Object, which occurred in 16% of sentences. All other structures occurred less than 2%. Perhaps it is unsurprising, therefore, that there is little guidance on the grammar one should use in writing descriptions, save of the type of those referred to in Section 1.

A significant development in this area was the European Union funded research project, CREWS (Co-operative Requirements Engineering With Scenarios), which proposed a complete set of guidelines called the CREWS Use Case Authoring Guidelines (Achour et al., 1999). These are a set of detailed Style and Content (sentence structure) guidelines, incorporating much of the general advice present in the literature, and providing detailed advice on how to construct each description event. Experimentation with the guidelines suggested that their application provided more completeness and consistency than descriptions written without the guidelines. However, the guidelines were considered somewhat unwieldy (Achour et al., 1999; Cox and Phalp, 2000) and the Content Guidelines too formal (Anda et al., 2001). In addition, the success of the guidelines was judged by the extent to which the suggested sentence forms were found in the resulting description. That is, the assessment was to some extent a self-fulfilling prophecy, in that where the suggested form occurred in the use case description the guideline would be seen to be successful. Similarly, Somé (2005), presents a structured use case grammar, but does not explain how this grammar was derived. The danger here is that, without theoretical validation, one is reliant on empirical studies, yet these can also become self-fulfilling prophecies, where the author appears to justify the utility of the structure by its subsequent appearance in the description after application of the guidelines. Furthermore, without stakeholder feedback, there is little to judge the practical utility of an approach. For example, a formal grammar may provide internal consistencies, and might yield promising empirical results, but could prove impractical to industrial users. Hence, this paper attempts to incorporate such lessons, not only by careful consideration of the theory behind grammatical structures, but also by considering (and gaining feedback about) their practical utility.

In order to have a more independent means of assessment for guidelines, Cox and Phalp (2000) proposed an approach to assessing use case descriptions by four 'independent' factors or means. These graded descriptions on plausibility (later termed 'realism' by Anda et al. (2001)), readability, structure consistency and whether (plausible) alternative flows were considered. Cox et al. (2001) then presented an alternative set of use case writing guidelines (with some marked differences and simplifications, called the CP Rules). With independent factors used to assess descriptions, experiments were then able to provide an assessment of the utility of the guideline sets. In brief, small differences in the success of the two sets were identified (Cox et al., 2001; Phalp and Cox, 2002), but the overriding finding was that the application of writing guidelines did improve the comprehensibility (or communicability) of the subsequent descriptions. Anda et al. (2001) came to similar conclusions—writing guidelines helped in the realism and consistency of the descriptions.

Although intended originally as a mechanism to aid the empirical validation of use case guidelines, the quality factors were found to provide useful guidance in their own right, and were later realigned into seven coherent categories (Phalp and Cox, 2002). This paper therefore describes the derivation of these (improved) communicability factors, and how they bring together research on discourse process, requirements and software engineering. Hence, the following section examines some of the rationale for such factors, by describing work on text comprehension.

3 Discourse process and text comprehension

Software engineering abounds with complex notations and descriptions. Hence, when considering the understanding of a use case description, one tends to assume that the act of comprehension is relatively trivial. Indeed, compared to a formal methods equivalent, a use case description does appear to be straight-forward, and there is an expectation that model builders and readers will both share the same common understanding. However, those who have examined the nature of comprehension would suggest otherwise.

> *"Reading comprehension is one of the most complex and uniquely human of cognitive activities. During reading, the successful comprehender connects the various events, persons, and objects that he or she encounters so that the text appears to be a coherent whole rather than a random list of facts and events. Frequently, the relations between parts of a text are implicit and, therefore, must be inferred. If all goes well, the result of the inferential processing is a mental representation of the text that is relatively stable and that can be accessed at a later point in time to answer questions, retell the story, and so forth."* (van den Broek et al., 1996, p. 165)

The ability to read a description in a coherent way and to build the representational model that the writer has in mind is vital to a description's ease of understanding. Conversely, the writer must also appreciate the perspective of the reader to be able to present a description that matches the reader's mental model (Traxler and Gernsbacher, 1995). Communicating the same understood message from the writer and reader's perspective is important. Therefore, the writer must construct their description in a coherent and straightforward manner. Clearly, there are significant implications here for use case descriptions, where both the readability and the need for reader and writer to share the same mental model are of great importance. Hence, it is important to recognize, and utilize, those elements of textual description that enforce ease of reading and understanding. The following sub-sections describe important aspects of discourse process that enable text comprehension, which should also transfer to the use case description.

3.1 Coherence and inference

Coherence is vital to text if it is to make any sense as a whole and, as individual elements within that text, to have a semantic relationship (Halliday and Hasan, 1976). Local coherence is the backward referencing that a clause makes to the previous clause in the text. There is also global coherence where a clause backward references a clause or object in the text prior to the last clause. McNamara and Kintsch (1996, p. 282) report an experiment where "*participants' average reading times per word were greater for the low-coherence than for the high-coherence text*". Thus, a more coherent text enables faster reading and understanding of that text.

Inferences do far more than shape a reference to the last clause, however. *"Inferences during comprehension take the reader beyond what is explicitly stated in the text… Thus inferences are particularly important to the interpretation of the text and the construction of a more complete mental model of the situation described in the text… Inferences that are made rely on the generic concepts in the reader's knowledge base,"* (Goldman et al., 1999, p. 6). Magliano (1999) and Kaakinen et al. (2002) agree with Goldman, stating the significance of world knowledge and perspective in understanding and achieving text coherence.

To enable a better understanding of the use case, readers of the description should be able to make some sense of it. The reader creates a mental model of what he/she is reading and ought to be able to infer what the next step in the description might be based upon the knowledge already assimilated and the mental model created. In simple spoken language, if there is a question, one normally expects a response to that question. This is known as an adjacency pair. "*The most common adjacency pair is the* [*Question* $\rightarrow$ *Reply to question*] *sequence*," (Graesser et al., 1996, p. 2). For instance, a description of an ATM in operation might state:

1. The User inserts the card into the ATM.

The response that one might infer is the next logical step in the process:

2. The ATM prompts the User for his/her PIN number

If there was no such statement then the structure of the mental model created by the reader might be drawn into doubt. It could be that the writer of the description chose to ignore the second sentence, assuming that the reader already has an accepted mental model of how the ATM process runs. However, Ozyurek and Trabasso (1997) state that since evaluation of sentences occurs throughout a text, rather than at the end, if there is an event missing (a span problem) or there is too much information (a scope problem (Jackson, 1995)), this can upset the overall view of the piece.

Garnham and Oakhill (1996) state that the structure of text (and therefore use cases) should be simple because word order is not typically remembered. However, Gernsbacher's "Advantage of First Mention" (1997, p. 273) principle shows that people recall the first of a pair of names far more readily than the second. Consider, for example, the sentence

Lisa and Maggie played a tennis match

In recall, Lisa is remembered more than Maggie (Gernsbacher, 1997). The "Advantage of First Mention" should be employed when writing use case descriptions. The principle actor of the use case description should be the first mentioned actor or subject in the description. This includes contextual details provided before the main flow of events.

Without relevant project specific knowledge, it is very difficult to comprehend the detailed contents of a text (Garnham and Oakhill, 1996). The same can be said of a use case. It is also the case, no doubt, that writing a use case or scenario would be very difficult without sufficient world knowledge. The construction manner of text comprehension is "*a holistic description of the overall situation linguistically communicated*" (Bransford et al., 1972, p. 202). To understand the content of the description, the reader has to understand the domain that the description fits within. Of equal importance, the writer of the description must have understanding of the domain the description (and the project as a whole) fits within. Understanding the problem domain is a necessary requirement for the use case writer and reader. Domain knowledge, though, is often only learned from practical experience.

3.2 Complex sentences

It is generally the case that people store knowledge about sentences and clauses, and interpret the meaning of them, dependent upon their structure (Garnham and Oakhill, 1996). Therefore, it can be considered that the structure of phrases is important. However, sentences can quite easily be misinterpreted although their meaning appears to be obvious. For example, from Wason and Reich (1979):

No head injury is too trivial to be ignored.

This is generally taken to mean that one should *not* ignore head injuries. However, the literal meaning states the reverse. The key is that this structure is complex—with a negative, passive and adjective. However, Clark (1997) points out that if the literal meaning of the sentence is to be dogmatically adhered to, the reader misses what the message is meant to convey; that is, that no head injury should go untreated. Readers do not follow the literal meaning but usually interpret a complex sentence into one that can be understood *and* which makes sense in the given context. To avoid the risk of confusion, it is sensible to keep the potential for ambiguity out of sentences where possible.

3.3 Referential continuity

Referential continuity is very important in use cases. As Garnham and Oakhill (1996, p. 321) point out "*Referential continuity is one of the major factors that determines whether a text is easy to understand*". Consider the following example:

The doctor is standing next to the patient.
He puts the prescription in his pocket.

Who is "he" and whose pocket is "his"? It would be better make the primary actor explicit:

The doctor puts the prescription in the patient's pocket.

This would suggest that the use of the pronoun anaphoric device has the opportunity to mislead the reader. However, if there is only one actor in a use case, and assuming that the system has no gender, then why not use a pronoun? In any case, the system ought to be referred to as ‘the system’ or ‘the machine’, or whatever name it has. That is, call the system by its proper name instead of using “it”. Otherwise, “it” (3rd person singular personal pronoun) might well be interpreted as a reference to the actor as much as to the system (since an actor can be a hardware device or a software application as well as a person). The problem with allowing pronouns in use case descriptions are many, and these outweigh their usefulness. For instance, although there may often be only one actor, the system, in responding to actor driven events, needs to state its own case. As such, the system name will become the *subject* of the structure. Differentiating between actors and systems is easy when proper nouns are used, but when both become the subject of sentences, the use of pronouns can lead to misunderstanding. In addition, in a use case, an actor is playing a role, or, at least, should be interpreted as such. Since a role does not necessarily have a gender (a role, say, Manager, or Customer, is non-gender specific) to give the actor a gender (via a pronoun) can also be considered misleading.

3.4 Structure foundations

It is much more difficult to comprehend text (i.e. develop a foundation of mental representations) *“if the information does not lend itself to building cohesive mental representations, for example, if sentences, paragraphs, or stories are self-embedded or scrambled”* (Gernsbacher, 1996, p. 290). Therefore, the notion of nesting sub-routines in a use case might not be ideal, primarily because it makes it difficult for people to follow a coherent path. For example,

1.

 1.2. –difficult, and actually there is the notion of moving back to 1 after

  1.2.1. completing 1.2.1., but this is implicit. Could the reader not move to 2?

2.

Gernsbacher (1996) describes a Structure Building Framework to model language comprehension, which enforces the establishment of logical coherence.

> *“According to the Structure Building Framework, the goal of comprehension is to build cohesive mental representations, or structures. The first process involved in building a structure is laying a foundation. The next process involves developing the structure by mapping on incoming information when that information coheres with the previous information. However, if the incoming information is less coherent, comprehenders employ a different process: They shift to initiate a new structure... Additional time processing is needed to handle coherence breaks or to shift focus from one structural foundation to another structural foundation.”* (Graesser and Britton, 1996, p. 343).

Use case descriptions ought to try to impose quicker reading, building upon such propositions in the description already read. To achieve coherence in a text, it is important to create a network of propositions. A text with few connecting propositions is considered difficult to understand (e.g. Mannes and St George, 1996; Fletcher et al., 1996; van den Broek et al., 1996). The noun-phrase argument of propositions is critical “*for connecting propositions and establishing text coherence; that is two propositions are connected if they share at least one noun argument*,” (Graesser and Britton, 1996, p. 345). Turner et al. (1996, p. 34) provide an explanation of propositional analysis:

"A predicate specifies a relation between arguments. For example, in the following sentence the predicate is hit.

(1) John hit the ball with the bat in the park at noon.

In Example 1, the predicate specifies the relations between John, the ball, the bat, the park and noon. In general, the predicate hit has a standard semantics which includes the notion of the hitter, the hittee, the instrument, and often the location and time. We can represent the semantics of the predicate hit in a general predicate frame with standard slots:

(2) hitter, hit, hittee, instrument, location, time

In these frames, the slots (i.e. the arguments of the proposition of which hit is the predicate) are filled from Example 1 as follows:

(3) John, hit, ball, bat, park, noon"

It is clear that coherence can be achieved by repetition of noun phrases in consecutive sentences. Thus, it should be considered that using proper names instead of pronouns enhances understanding of text. However, it is not recommended that use case descriptions be written in the style of Turner et al.'s third example above because this does not appear to be obviously comprehensible. As a means of examining the subsequent description, however, they might be useful. Indeed, Robertson (1995) uses propositional analysis to identify objects that form part of the design of a software system. Structures should be kept very simple because the use case reader's working memory should be free of less important predicates.

3.5 Lessons learned from discourse process

It should be clear that the lessons learned from discourse process research, as outlined above, have parallels for writers of use case descriptions. The following section attempts to make a more explicit connection, by taking five important metaphors identified in discourse processing and suggesting their role in making use case descriptions more comprehensible.

4 Discourse process metaphors for use case descriptions

From examination of the models and processes suggested in text comprehension, Graesser and Britton (1996, p. 342) suggest five metaphors that capture the various models of text understanding. The following sections outline how these five metaphors have an influence on how one should construct use case descriptions to consider the reader, and to aid their comprehension.

4.1 Understanding is the assembly of a multilevel representation

As depicted in Fig. 1, use case descriptions should continually build upon the foundations constructed at the start, rather than add events that force new foundations to be laid.

4.2 Understanding is the construction of a coherent representation

The goal is that the reader should be able to comprehend a description quickly. To facilitate this, it is recommended that the whole use case is written to achieve logical coherence

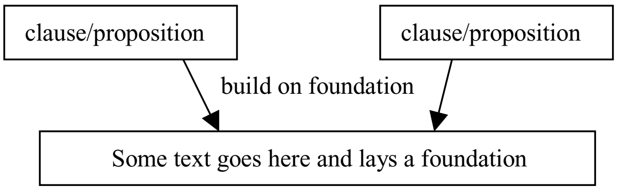
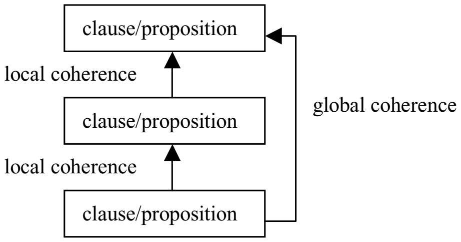


Fig. 1 Building mental foundations for text understanding

Fig. 2 Logical coherence in text



between events (Fig. 2). This means that the use case writer has to make sure that only information relevant to the continuity of the use case is introduced.

O'Brien and Myers (1999, p. 35) suggest:

> *"A basic principle in writing is that ideas should flow easily, one from another. In this way, as each new idea is introduced, it is easily integrated with the information that immediately precedes it. That is, the text should always be locally coherent. Models of text comprehension agree that readers make use of the information in working memory when integrating each new sentence into the text representation."*

Each successive sentence ('event' in use case parlance) in a description should logically cohere to a preceding sentence. For example:

The Customer selects Change PIN option.
The System asks for existing PIN.
The Customer enters existing PIN.
The System asks for new PIN.

Figure 3 illustrates the logical coherence in this example. There is local coherence to the previous line and also global coherence to events prior to the previous line.

Local coherence may not always be achievable but one should strive to obtain it (so long as it does not substantially increase work load).

4.3 Understanding is a complex dynamic system

Understanding is complex and levels of complexity are, at least in part, governed by the syntactic nature of the text. Hence, the use case structure (syntax) should aim to be as simple as possible, whilst still allowing the writer to express what is needed. Indeed, Perfetti (1997, p. 339) states that discourse theorists finally understood that "*Syntax had a purpose after all, and smoothly functioning discourse depended critically on syntax's formal power to flexibly arrange elements*". Perfetti also points out that syntax reduces the ability to understand propositions. However, without some syntax, propositions become even more difficult to

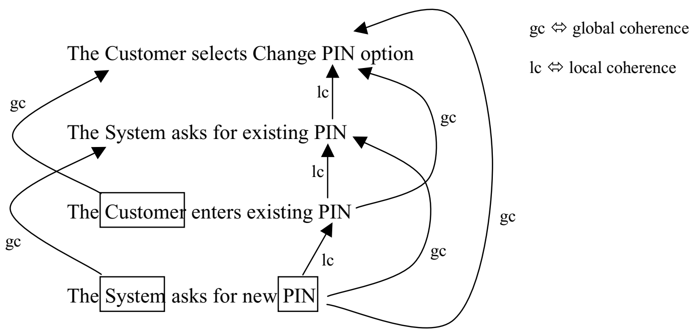

The Customer selects Change PIN option
The System asks for existing PIN
The Customer enters existing PIN
The System asks for new PIN
gc
lc
gc ⇔ global coherence
lc ⇔ local coherence

Fig. 3 Example of logical coherence

understand. That is, some trade off must be made. There has to be syntax but of a relatively simple structure.

For the use case structure, one should consider applying the adjacency pair rule. For example, Input $\rightarrow$ Response to input, or Interaction $\rightarrow$ Response to interaction (this focuses on describing specification). This appears natural, as engineers tend to understand the input $\rightarrow$ output idea. Indeed, it is an idea common to specification: describe an input and then show the system's output (if the audience is the user, as opposed to engineer, the adjacency pair might be rephrased as action $\rightarrow$ response or question $\rightarrow$ response, because users might not be aware of the input $\rightarrow$ output notion). The output should have meaning to, or describe a relationship with, the input. What is required from use cases is the same: local coherence.

4.4 Understanding is a process of managing working memory

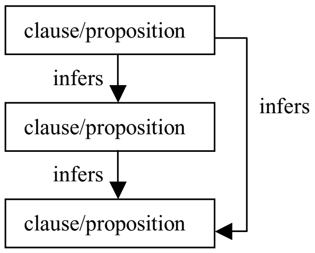
To help the reader organise their working memory, the use case description should not be complex; the structures should be straightforward and coherent. Again, the application of the adjacency pair rule helps this. Korn (2000) suggests that adjectives and adverbs introduce subjective possibilities of success into scenarios. Subjectivity can lead to misunderstanding. Removal of unnecessary adverbs, adjectives and pronouns, for instance, helps reduce the difficulty in organising working memory by simplifying the use case description. This should then speed up understanding and reduce ambiguity. Note, however, that some authors prefer to consider that adjectives can often be useful to subsequent designers, in that adjectives may often be identified as services within objects.

4.5 Understanding is inference generation

The notion of building upon foundations of previous events in the use case is that a reader can expect something to happen next (see Fig. 4).

That is, the reader infers that an event will now take place based upon information already comprehended. Using the adjacency pair rule should help inference generation become more accurate, because if there is an input with a specific purpose (sub-goal), one expects a response to achieve that purpose.

Fig. 4 Inference generation



4.6 Use case diagram—Discourse meta-model

Figure 5 illustrates a meta-model for a use case description. The notion of such a model in Requirements Engineering is relatively common. For instance, Pohl et al. (2001) use a scenario meta-model to maintain traceability to (an evolving) software architecture. Figure 5 shows how local coherence is significant to the use case description. The use case description describes a functional/user requirement, which should define how to obtain a business or user goal. The use case description must fulfil this goal. The description itself (which can be construed as the main flow of a use case description) is composed of three major elements. The first type of element is *system*, of which there is one, since this model does not consider the case of distributed systems. The second element is *event*; there should be at a minimum an input and a corresponding output. The third type of element is *actor*, of which there is at least one, since without an actor there is no notion of interaction, either at the interface or among subsystems. Input and output are kinds of events. The actor provides inputs and

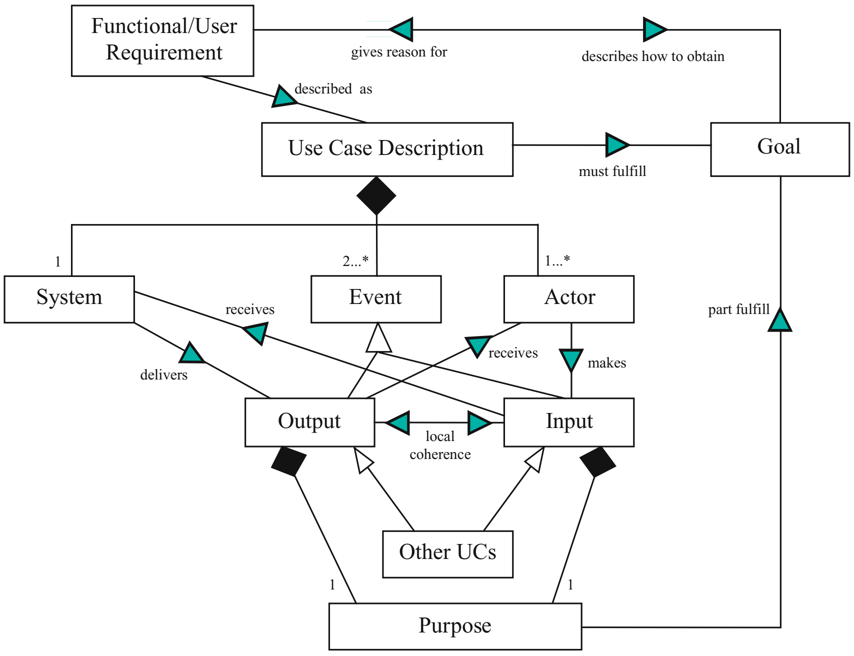


Fig. 5 Use case meta-model

receives outputs. The system receives these inputs and delivers outputs. Therefore, there should be local coherence between input and output. It is also the case that other use cases are types of inputs and outputs. This corresponds to other use case names (UML would denote these as *includes* and *extends* relationships) that are underlined as events in the description (Cockburn, 2001). Each input and output has a purpose (a sub-goal). The sum of purpose 'part fulfils' should achieve the goal of the use case description. This allows for the notions of purpose and goal at different granularities, detailing the difference suggested by Kaindl (1998); this equates to the sub-goals needed to achieve the goal.

Figure 5 employs local coherence between events in the use case description. The following section describes other desirable qualities of use cases as identified above, and attempts to collate these qualities into a coherent and usable framework.

5 Quality factors: The 7 Cs of Communicability

This section organises the ideas, comments and suggestions of authors described earlier in this paper into categories relevant to improving the communicability of use case descriptions. Table 1 explicitly describes the categories and their sources.

The '7 Cs' are proposed as a set of heuristics that might be valuable in general validation when actually writing descriptions. In addition, they allow for the empirical validation of guidelines, by providing a set of criteria on which one may base measures of use case worth. It is hoped that this will help requirements engineers, and software engineers in general, in their task of writing use case descriptions. These guidelines do not provide any prescribed method to guarantee success. Note that this is guidance on some of the allegedly easier tasks of requirements: writing descriptions; but as Cockburn (2001), Wieringa (2001), and Bray (2002) point out, writing is a complicated task not to be underestimated. By stimulating the use case writer with these heuristics, they might help as a catalyst to further clarification of the task of determining use case descriptions.

The 7 Cs are summarised below, then discussed in more detailed in subsequent sections.

1. Coverage.

   1.1 Span: The use case should contain all that is required to answer the problem.
   1.2 Scope: The use case should only contain detail relevant to the problem statement. Extra unnecessary information provided is out of problem scope and not required.

2. Cogent.

   2.1 Text Order: The use case should follow a logical path with events in the description in the correct order.
   2.2 Dependencies: The use case should complete as an end-to-end transaction (which can include alternative/exceptional flows). Does the actor reach a state that stops the transaction from terminating as expected?
   2.3 Rational Answer: The logic of the use case description should provide a plausible answer to the problem.

3. Coherent. The sentence being written should *repeat a noun* in the last sentence or a previous sentence, if possible. The description is easier to read and quicker to understand if there is logical coherence throughout.

Table 1 Underpinnings of the 7 Cs of communicability

7 Cs of Communicability		Rationale
Coverage	Scope Span	Jackson (1995) – refined notions of completeness for requirements.
Cogent	Text Order	Gernsbacher (1996, 1997) – structure building framework Graesser et al. (1996) – inference building (Question → Reply to Question)
	Dependencies	Jacobson et al. (1992), OMG (2001) – representing a complete transaction. e.g. Trabasso et al. (1989), Goldman et al. (1996) – local and global coherence. Garnham and Oakhill (1996) – referential continuity. Graesser et al. (1996) – inference building (Question → Reply to Question)
	Rational Answer	Gernsbacher (1996, 1997) – structure building framework Graesser et al. (1996) – inference building (Question → Reply to Question) Anda et al., (2001) – the realism of the use case
Coherent		e.g. Trabasso et al., (1989), Goldman et al., (1996) – local and global coherence.
Consistent Abstraction		Anda et al., (2001) and see Section 2.3.1 for arguments on this.
Consistent Structure	Variations	Kulak and Guiney (2000), CREWS – keep variations to a separate section.
	Sequence	e.g. Schneider and Winters (1998), CREWS – consistent sequential numbering
Consistent Grammar		e.g. Pooley and Stevens (1999) – avoid passive voice; the consensus is that there are many grammatical elements to avoid (Section 2.4). Some structures might improve comprehension e.g. Graham (1998) SVDPI.
Consideration of Alternatives	Separation	Kulak and Guiney (2000) – keep variations to a separate section. Alexander (2003) – failure to deal with exceptions leads to system failures.
	Viable	Alexander (2003) – failure to deal with exceptions leads to system failures.
	Numbering	Cockburn (2001), CREWS – there should be consistency in numbering.

4. Consistent Abstraction. The use case should be at a consistent level of abstraction throughout. Mixing abstraction levels (problem domain, interface specification, internal design mixes) may cause difficulty in understanding.
5. Consistent Structure.

5.1 Variations: Alternative paths should be excluded from the main flow. Inclusion of alternative paths in the main flow reduces readability.
5.2 Sequence: Numbering of events in the main flow should be consistent.

6. Consistent Grammar. Simple present tense should be used throughout. Adverbs, adjectives, pronouns, synonyms and negatives should be avoided.
7. Consideration of Alternatives.

7.1 Separation: There should be a separate section for any alternative/exceptional paths to the main flow.
7.2 Viable: Alternatives should be viable and make sense.
7.3 Numbering: Alternative numberings should exactly match the numbers in the main flow.

Table 1 shows how the 7 Cs relate to the ideas described in this paper

5.1 Coverage

The notion of Coverage introduces attributes of *Scope* (too much information) and *Span* (not enough information) (Jackson, 1995). These can be viewed as finer-grained completeness, a suggested quality good use cases should portray (Achour et al., 1999).

5.2 Cogent

The Cogent facet contains three elements: *Text Order*, *Dependencies* and *Rational Answer*. If a use case description follows a logical order, this ought to make the construction of a Structure Building Framework easier (Gernsbacher, 1997). It is also the case that if the description's text is in the correct order then inference building is less difficult, for instance, by application of the Question $\rightarrow$ Reply to Question sequence (Graesser et al., 1996). *Dependencies* can be defined as determining the completeness of the description; that is, that it describes a complete transaction (Jacobson et al., 1992). At a finer-grained level, each event has a dependency on a previous event (logical coherence, e.g. Goldman et al. (1999)). This implies referential continuity (Garnham and Oakhill, 1996) and inference building (Question $\rightarrow$ Reply to Question) (Graesser et al., 1996). *Rational Answer* considers the notion of plausibility as first suggested as part of an independent means for assessing the quality of a use case description (Cox and Phalp, 2000) and further used by Anda et al. (2001) in use case assessment (where it is referred to as "realism", p. 409). The following example presents something that might not appear rational:

1. The Customer inserts card into ATM.
2. The Customer selects Change PIN.
3. The Customer enters new PIN with a sledgehammer.

The prepositional phrase in event 3 'with a sledgehammer' is not considered rational and this will have an effect on any mental structure built, forcing the creation of a new structure (Gernsbacher, 1996). Inference generation (Graesser et al., 1996) is also broken—the reader is unlikely to infer the use of a sledgehammer to work the keypad. Though the use of a sledgehammer is irrational, it also raises design and security issues for the ATM. For instance, is the ATM capable of withstanding a hit with a sledgehammer, are there ways in which the ATM can be broken or vandalised, what about the safety and security of valid customers? These are clearly exceptional events that need to be flagged and considered if the system under design will be able to cope with these potential problems (Alexander, 2003).

5.3 Coherent

This facet implies good local coherence and global coherence for better understanding of the text (e.g. Trabasso et al., 1989).

5.4 Abstraction

Use cases are often used to describe requirements for systems to be designed and implemented in the object-oriented paradigm (Arlow, 1998). However, there has been much debate as to where the use case is the most effective approach. Jacobson et al. (1992) see the use case as useful for requirements, specification and design. The UML community takes a similar viewpoint, for example, Booch et al. (1999), Jackson (1998, 2001), and Kovitz (1999) see use cases as a means for describing a specification, because use cases deal with interactions between a user (actor) and the machine (system). However, Rosenberg (1999, 2001) has a slightly different take on the matter. He sees use cases as ways of describing "*units of behaviour*", requirements as describing the "*laws that govern that behaviour*" and functions as "*the individual actions that occur within that behaviour*" (p. 123). Jarke et al. (1998, p. 165) ask a key research question: "*what is the appropriate level of abstraction in a scenario, given a certain purpose*?". This question is equally applicable to use case descriptions. Scenarios can be represented at varying levels of abstraction from the problem domain to system design. Mixing abstraction representations in one scenario is considered possible (Carroll, 2000) and indeed this is commonly seen in scenarios and use case descriptions (e.g., Regnell et al., 1995). Alexander (2002) notes that authors should be careful not to mix abstraction levels, arguing that such internal design representation (describing the solution system's internal structure) in the requirements and specification phase forces consideration of solution information when it is not required. Indeed, the Object Management Group (OMG, 2001, p. 2–137), who have standardised the UML for industry, state: "*The use case construct is used to define the behaviour of a system or other semantic entity without revealing the entity's internal structure*". However, they then backtrack somewhat: "*A use case describes the interactions between the users and the entity as well as the responses performed by the entity, as these responses are perceived from the outside of the entity*". This could mean that only external events are documented because the user has visual, aural or tactile awareness of the event. Alternatively, it could mean that the user (outside the entity) is somehow aware of what the entity or system is doing internally because he is curious to know how the system works. In light of this confusion, the authors here presents their own view on the abstraction argument: a use case can be considered more cohesive—a good design quality (Budgen, 1994)—if it does not combine different abstraction levels into one description.

Use case descriptions were originally designed to describe interactions at the machine interface (Jacobson et al., 1992) without consideration of internal design or the problem environment. However, if one examines the texts describing use cases and scenarios in requirements and design, then their usage appears in almost all phases (Cox, 2000). For example, Insfran et al. (2002) explicitly describe system internal responses in a three-column format. Column one depicts general information that equates to an environmental type scenario, column two represents system interaction, and column three represents system internal response. This underlying assumption, that use cases are universally applicable, is a dangerous one since the use case does not model what happens away from the machine interface where the requirements will have their effect. Jackson (2001, pp. 5–6) provides a clear example of this use case weakness. External design (describing the external appearance and behaviour of the system), though, is another matter. Indeed, Rosenberg and Scott (1999) recommends that use cases be used to help develop prototypes and the Graphical User Interface (GUI). The question arises, then, whether internal design issues are necessary in requirements and specification. For instance, Regnell et al. (1995, p. 4) consider events in a use case description such as *card validation* as something that will "*have* [*an*] *effect on the users*" of ATMs. Interestingly, Regnell and Davidson (1997, p. 1) state "*A use case may be described either from an external* (*black-box*) *point of view suitable for requirements, or from an internal* (*white-box*) *point of view suitable for design*". They do not state that a use case can contain both black box and white box events in the same description. An ATM card validation concerns how the ATM internally checks that this is a valid card (and is white box). Is the card validation a shared phenomenon (Jackson, 1995) between the problem domain (ATM user) and the machine domain (ATM banking system)? Events such as card validation are indeed important to the user but they are internal design. From the user's viewpoint, although he/she might be implicitly aware that some card validation occurs, he/she is (generally) visually unaware of it unless the card is rejected. Bray (2002, p. 249) suggests that such validation checks (which could be design constraints) should be specified separately, although different viewpoints might require different information. For instance, a bank might need to know when a card is validated, whilst the customer probably will not be concerned. Even so, in terms of the bank's viewpoint, when the card is validated is still a matter of internal design for the system itself (and obviously this should occur before the customer is allowed to withdraw money). Indeed, Mattingly and Rao (1998, p. 78) note that "*internal (white box) interactions, although important for design, should be separate from the use case. They do not describe the interface to the system, and confuse the simplicity of what use cases should strive to be*". A specification documents inputs and outputs to a system and the relationships between them (Davis, 1991). It is the authors' contention that such relationships between input and output should not form part of a use case description (although they might be valid as specification per se) and that use cases should take the viewpoint of the actor, not the system.

5.5 Consistent structure

Consistent Structure requires that *Variations* be kept out of the main flow of events. There appears to be a consensus on this (e.g. Kulak and Guiney, 2000). *Sequence* asks that each event be correctly numbered, as recommended by, for example, Achour et al. (1999) and Schneider and Winters (1998).

5.6 Consistent grammar

There is a general consensus about the avoidance of certain grammar constructs in descriptions, for instance, the passive voice (e.g. Pooley and Stevens, 1999) and that certain grammar structures are recommended (e.g. Graham, 1998; Achour et al., 1999; Cockburn, 2001).

5.7 Consideration of alternatives

Consideration of alternatives suggests there be *Separation* (a separate section) for alternative and exceptional flows (e.g. Kulak and Guiney, 2000). The viability of these exceptions and alternatives is important. Failure to consider them could lead to system failure (Alexander, 2003). *Numbering* is a check to make sure that the alternative and exceptional paths listed correspond exactly to their equivalent in the main flow. For instance, if event 4 in the main flow has the Customer asking for a receipt and the alternative has the Customer doing something else, the numbers should correspond: 4 in the alternative to 4 in the main flow. Slight variations are also accepted, such as a4/4a (alternative) or e4/4e (exception) (Cockburn, 2001). However, 4.1 as an alternative to 4, is not acceptable.

6 Industrial study

Examples of use cases from an industrial case study, Cox (2002), are used here to illustrate the arguments of the paper and demonstrate the application of the 7 Cs of Communicability. The case study is sourced from a financial company in the City of London. The chief business of Company X (named as such to maintain its confidentiality) is the building and supporting of complex applications for the online buying and selling of shares, stocks and investments such as government bonds. As an estimate of the size of the organisation, before Initial Public Offering to the Stock Market, Company X was valued at over £350 million.

The project entailed re-engineering a critical straight through process of customer application, from customer application on the internet to lodging of that application in the system, to printing application forms for postage, and signing by the customer in order to activate the customer's account for online trading. It is not the intention of this paper to cover the case study in detail; we draw instead, two use cases from the broader study for the purpose of illustration. The first is the process of printing online customer account applications. The second is a longer use case description of the process of applying for a trading account. In general, the descriptions from the case study are lacking in detailed alternative and exceptional flows. This is a weakness but time constraints forced a rationalisation of the descriptions. That is, there was not enough time to elaborate upon all the potential alternative flows. As Davis and Hickey (2002) point out to academic researchers, it is important to test approaches in industry and get stakeholder feedback on those tests. We therefore also include critical feedback from stakeholders, and this is drawn upon to provide a validation of the 7 Cs in terms of how useful the use cases were to those stakeholders.

When considering the 7 Cs there is a degree of subjectivity in some factors, which is determined by the perspective of the use case writer and the context. Each use case description was assessed with the 7 Cs. In total, there were twenty-six identified use case descriptions in the case study. The mean length of the descriptions was eleven events (and the median 5) in the main flow. Therefore, it was felt that, in terms of checking the descriptions for conformity to the 7 Cs, application of the checklist was not an overly time-consuming activity.

6.1 Case study design

The study described here also contained a degree of action research in addition to other more typical elements of the case study. One of the authors acted as a member of the IT team working on specifications and as such this can be considered action research (Zuber-Skerritt, 1996). To provide a structure, this section follows Yin's outline case study design (1994). The author modelled from external processes and interfaces inwards. This "outside-in" (Potts, 1993, p. 22) analysis was appropriate because the external boundaries were already established.

6.1.1 The study's questions

The study initially set out to question how useful the 7 Cs are in an industrial setting.

6.1.2 The study's propositions

The goal of the case study is to apply the techniques to an industrial project. As such, the proposition suggested is:

1. The 7 Cs of Communicability are a useful technique for assessing the internal structure of the use case descriptions.

There is some feedback on the application of the use cases by members of the project team in Section 6.4.

6.1.3 The units of analysis

The units of analysis are the implementation processes and effectiveness of the 7 Cs of Communicability.

6.1.4 The logic linking the data to the propositions

As Yin (1994) states, this is a difficult task. Therefore, the majority of the logic linking the data to the propositions is provided by the authors' viewpoint in applying the approach, and stakeholder feedback on that application in the form of interviews and discussions.

6.1.5 Criteria for interpreting the findings

There is critical feedback from stakeholders and this is drawn upon to provide a validation of the 7 Cs in terms of how useful the use cases were to the stakeholders. Other observations are made by the authors. Galliers and Land (1987) suggest that descriptive research is a valid approach, and some of the work here follows that lead, with the caveat of author subjectivity kept in mind.

6.2 Use case descriptions Example 1: Print Mailing List

The first example use case is that of Print Mailing List. This use case description is used by Printing and Mail Room team that is internal to the Company.

PrUC2: Print Mailing List (Pack Types)

Actors: Print Room Staff member(s), Printers

Context: It is either 9 am, 12 pm or 3 pm and the Print Room staff are readying for a new print run to commence.

Pre-conditions: Files have been imported (copied) to the Access database from the Back Office; the Access Database is functioning normally.

Main flow of events:

1. The Print Room Staff member selects "Pack Type" to print from Pack Type list.
2. The Print Room Staff member selects "Print Mailing List".
3. The Print Room Staff member sees instructions for opening a Company X account in Microsoft Word.
4. The Print Room Staff member sees a mail merge in Microsoft Word.
5. The Print Room Staff member selects "Print" from the File menu.
6. The Print Room Staff member selects the printer to fit the Pack Type.
7. (Optional) The Print Room Staff member selects the number of document pages to print.
8. (Optional) The Print Room Staff member checks the correct forms are in the printer.
9. The Print Room Staff member starts the print run.
10. The Printer prints the documents.

Exceptional flows

e6. The Print Room Staff member selects the wrong printer.
  e6.1 The Print Room Staff member cancels the print run. (Here or 10?)
  e6.2 The Print Room Staff member selects the correct printer.
  e6.3 The Print Room Staff member restarts the print run.
e9. The print run does not complete.
  e9.1 The Print Room Staff member cancels the current print run.
  e9.2 The Print Room Staff member locates the cause of failure. (?)
  e9.3 The Print Room Staff member rectifies the problem. (How?)
  e9.4 The Print Room Staff member restarts the print run.
e10. The printer feeds papers from the wrong tray.
  e10.1 The Print Room Staff member does what?

Post-conditions: The packs are printed.

6.3 Validation of Example 1: Print Mailing List

Validation of the use case involved an informal reading through the 7 Cs heuristics as a checklist. There is a difficulty in the consistent application of the 7 Cs in that there is a large degree of subjective judgement involved in their assessment. Since many of the Cs are semantic notions, this subjectivity seems somewhat inevitable.

(1) *Coverage*. There is a potential *Span* problem in the exceptional flow event e9.3, which is flagged with a 'How' question because it needs further investigation. The description should not normally be signed off without resolving this issue. There do not appear to be any *Scope* problems, though as stated, events 3 and 4 might be considered too much information since they are cognitive events. However, the context suggested that they be included, though this is a subjective opinion. *Span* and *Scope* might be increased or decreased dependent upon the reader's viewpoint. Since the selection of pack type to printer is important (due to application form sizes) this might be necessary to expand upon. The *Span* of events 1 and 6 could be considered insufficient information to cover this important point. However, to go into further detail in this description might detract from its overall task of describing a print run. The details could be expanded upon elsewhere, perhaps through a use case scenario. One might have to accept the context of the situation and state that the descriptions match the coverage required by the context. If one were considering reusing use case descriptions this might be a drawback in that events 3 and 4 could be considered superfluous to the general task of printing applications.

(2) *Cogent*. The *Text Order* of the description follows a logical path except for exceptional flow e6.1, which questions whether this event should occur here or at 10; since this is flagged, this is considered a reasonable answer at an early analysis stage but it should be resolved before being signed off. The description is an end-to-end transaction (*Dependency*), though exceptional flows e9.2, e9.3 and e10.1 need resolution. The description provides a *Rational Answer*, though the exceptional flows are flagged for further study. These need to be resolved to avoid potential software or process failures (Alexander, 2003). Again, these are subjective considerations. For instance, are events 3 and 4 necessarily in the logical order? Do events 7 and 8 confuse the answer because these events are not dependent upon the others? It is clear that the success of event 10 is dependent upon there being sufficient paper in trays and it can be argued that Customer satisfaction is dependent upon being sent the right number of pages to be able to complete the application (event 7). Thus, events 7 and 8 are possibly more important to the success of the description than their 'Optional' state suggests. There is sufficient detail in the description to provide a rational answer for the context but there is the potential playing down of parts of the process important to its success.

(3) *Coherent*. There is local coherence throughout the main flow because Print Room Staff member has been named in all events except 10. This refers to the Printer and documents. The Printer coheres to the 'print run' noun phrase of event 9. However, the exceptional flow e10.1 does not cohere to the exceptional flow e10, and resolution of this exception might make the description more coherent.

(4) *Consistent Abstraction*. Most of the description is at the interaction level though there are some events that relate to the problem domain; for instance, event 8. The mix of abstractions appears reasonable because the description is describing the actions of the actor when conducting a print run. This suggests that mixing abstraction levels is necessary to convey full meaning in the description. There is, therefore, some subjectivity in determining what level of abstraction one should show. Again, the situation should determine the abstraction required.

(5) *Consistent Structure*. There are no *Variations* in the main flow, though the 'Optional' events might be considered alternatives. They are not quite the same as alternatives because they are not substitutes for other events. They could have been written in separate scenarios but this would mean repeating the description four times for one different event. There are no alternative flows listed though there are exceptional flows. There were no alternatives suggested—the process is quite constrained. There are no *Sequence* (numbering of events) mistakes.

(6) *Consistent Grammar*. This relates primarily to the CP Style Rules 2 and 3[1] (Cox, 2002). Grammar is consistent in accordance with these Rules. Each event uses present tense and there are no negatives, adjectives, adverbs or pronouns. The exception e10.1 shows variation to the rules but this is a question and flagged for further study. Whether events such as e9.3, which ends with the *How*? question should be counted or not as grammatically correct is a subjective judgement made by the use case writer. In this case, the authors suggest that these not be counted as incorrect since they flag that further study has to be done to resolve outstanding issues.

[1] (Abridged) CP Style Rule 2: All sentences are in present tense format. (Abridged) CP Style Rule 3: Avoid using adverbs and adjectives. Only use negatives in alternative and exceptional flows of events. Avoid using pronouns.

(7) *Consideration of Alternatives*. The *Separate* section considers exceptions only. These are reasonably well described though there are four exceptions that are flagged as questionable or need further work (*Viability*). They are correctly *Numbered*.

6.4 Example 2: Applying for a trading account

This second example is a longer use case description, which describes an online application for a Trading Account.

---

iUC1: Apply for an Individual Trading Account

Actors: Customer

Context: The Customer wants to open a trading account on the Company X website because the Customer wants to start trading in stocks and shares on the stock market.

Pre-conditions: The Customer logs on to the Company X web site; the Company X website is accessible.

Main flow of events

1. The Customer types www.Company X.com into the address bar of the web browser.
2. The Company X website appears on the screen.
3. The Customer selects "Apply Now".
4. The website takes the Customer to the Apply Screen.
5. The Customer reads the guide on how to apply.
6. The Customer sees the choice of Accounts (details: Trading Account or ISA).
7. The Customer selects "Trading Account"
8. The website takes the Customer to the Application Form (details: page 1 of 3)
9. The Customer selects the Country of Residence.
10. The Customer selects "Individual Trading Account".
11. The website asks, "Where possible do you prefer to deal in certificates?"
12. The Customer ticks the option.
13. The website asks how the Customer wishes to have interest and dividend paid.
14. The Customer selects "Cash".
15. The website asks which currency the Customer wishes any income paid in.
16. The Customer selects "GB Pound".
17. The website asks if the Customer would like to deal in UK registered warrants.
18. The Customer ignores the option.
19. The website asks if the Customer wants stock to be registered at a different address.
20. The Customer ignores the option.
21. The website informs Customer that a "contract note" will be sent on every trade.
22. The website asks the Customer if the Customer requires a contract note to be sent to a third party.
23. The Customer ignores the option.
24. Customer selects "continue".
25. The website takes the Customer to the second page of the Application form.
26. Website presents the Customer with "About the Primary Holder" screen.
27. Customer completes details.
28. Customer selects "Continue".
29. The website shows the completed page 2.
30. Customer selects "Continue".
31. The website goes to page 3 of the Application form.
32. The website presents the Customer with the Customer's details on the application.
33. The Customer selects "Apply".

---

34. The website presents the Customer with “Application” screen.
35. The Customer sees “Successful Submission” message.
36. The website displays the “Customer Number”.
37. The Customer enters a numeric PIN twice.
38. The website presents the Customer with “Change Dealing Password”.
39. The Customer enters a password.
40. The clicks “Change” button.
41. The website presents the Customer with a Welcome Information screen

Alternative flows

a20. Customer selects Register Stock at Different Address.
a23. Customer selects Send Contract Note to Third Party.
a33. Customer selects “Back”.
  a33.1 Website displays page 2 of application form.
  a33.2 Customer makes changes to application.
  a33.3 Customer clicks “Continue” (use case returns to event 33 in the main flow).
  a33.4 Customer selects Transfer Stock

Exceptional flows

Note: Need to consider this exceptional flow in detail.
e34. The website presents the Customer with “Failure” screen.
  e34.1 The website informs the Customer of alternative application procedures.

Post-conditions: The Customer has successfully opened a Trading Account with Company X. The system has generated a unique Customer Number (event 36).

The above description is long, but this is not unusual in industry (Leibundgut, 2002), although it somewhat contradicts Cockburn’s suggested 10 line maximum (Cockburn, 2001). This is because the author wanted to know what the Customer actually does at the interface; that is, the describing of the interaction between the actor and the system, which is a key purpose of the use case description (OMG, 2001). It would have been easy to write a short description such as:

1. Customer selects Apply for Individual Trading Account.
2. Customer completes details.
3. System presents unique trading number.

However, this would have been almost valueless for the task of developing a design for the application because the authors would have had to elicit the details contained in the full description in any case.

6.5 Validation of Example 2: Applying for a trading account

As before, we apply the 7 Cs checklist to the use case description, but as Example 1 was discussed in some detail, for the purposes of brevity we highlight here only key points of interest.

(1) *Coverage*. There are no *Span* problems except in the exceptional flow of events where there is a reference to examine the Failure Screen further because the exact details of the Failure Screen need to be documented so that this Customer might be able to apply by another route. One can argue that there is too much detail (*Scope*) because of the

clicking and ticking of options. However, all the details are appropriate to the context of this description (Alexander, 2002a) because they are necessary to be able to determine the exact type of application the Customer has applied for and what potential options there are.

(2) *Cogent*. The *Text Order* of the description follows a logical path—the event order matches that of the web interface at the time of the study. *Dependencies* are also satisfactory—the use case describes an end-to-end transaction. The only doubt is the exceptional flow but this is flagged for further study. The description provides a *Rational Answer* since it is taken from the website.

(3) *Coherent*. This is hard to achieve because the website dictates the order of events and the language used. However, there is local and global coherence throughout.

(4) *Consistent abstraction*. The entire description is at the interface specification level and as such can be considered consistent.

(5) *Consistent structure*. There are no *Variations* in the main flow, though the ability of the actor to select different options is possible. There are *Alternative Flows* missing, e.g. event 12; although appearing relatively insignificant, the impact on how the Customer and Company X conduct business is important. This should therefore be documented. The *Sequence* (numbering of events) is correct.

(6) *Consistent grammar*. Grammar is not always consistent. There are modal verbs (e.g. event 13) and future passive tense (e.g. event 21). The author was constrained to use these structures in an attempt to represent the information at the interface as closely as possible.

(7) *Consideration of alternatives*. Although there is a *Separate* section some alternatives are missing which ought to be considered for a more complete description. For example, events 12, 14, 16 and 18 probably require further explicit consideration of alternatives because these determine whether the Customer wants to be paid in cash or dividends, to deal in certificates etc., and are very important to the Customer's account set up. The alternatives are *Viable* and correctly *Numbered*.

6.6 Stakeholder feedback

The stakeholders in this case study—developers and managers working on the project—gave feedback on the use cases.

> *"The descriptions are very important for the finer details and the step-by-step operations. This is very useful for both IT and QA."—Company co-Founder and IT Vice-President.*

This remark suggests that use case descriptions could have a potentially important role to play in the IT and QA departments at Company X. When interviewed at the end of the study, the QA Manager reiterated the Vice-President's comment:

> *"[The use case descriptions are] definitely a good thing for QA and testing as a basis for test development and user acceptance tests."*

When asked if there were any weaknesses with use case descriptions, he stated there were none. The IT Development Manager also considered the descriptions as a useful testing tool:

> *"Use cases [are] good for testing and for reshaping the application process at the interface. The online application process is too long and no one has ever produced this kind of document to see the exact steps required to complete an application. The*

trouble is time constraints on getting this done—we don't have time to do all this"—IT Development Manager.

Despite the recognition from members of the IT department of the role use case descriptions can play, both to document the processes and also to use those descriptions to test and reshape applications, there is concern that there may not be not enough time to develop them. The IT Development Manager is at the 'coal face' of software construction and product delivery in the company. Although the QA Manager and the Vice-President are well aware of how development is done, they were not building the actual software at the time of the study and had a perhaps more idealistic view. The IT Development Manager delivered the practical view that although the descriptions would be a very good tool, they would not have time to write them: "*the work culture needs to change a bit [before] this can occur.*" A further interesting comment came from the IT Development Manager:

"The descriptions are good for test scripts. They capture lots of detail. [And are] actually similar to how a servlet is written! This is dangerous because it might encourage jumping from a use case straight into code. It's also frightening to see that there's loads to do! Descriptions are good for showing what's going on in the system."

The resemblance to how servlets are written is interesting. In terms of internal design, this might be useful because the description acts as a guide for how the system is to work, but, as stated, developers might jump from the use case into code without consideration of design.

7 Discussion

This paper has introduced a Use Case Description Quality Checklist (referred to collectively as the 7 Cs of Communicability) that acts as a check on the quality of the written description. The checklist has been formed by bringing together research in requirements engineering and discourse processing to produce a coherent set of desirable use case qualities.

Although there appears to be a good deal of advice on writing use case descriptions, much of it falls into two categories: minimalist or overwhelming support. The overwhelming support approach provides complete and comprehensive guidance on how to write use cases. Examples included Achour et al. (1999, and Adolph et al. (2003). The drawback to these is that they often require bespoke tool support to be practicable and one would need a good memory to remember all that they propose. The desirable use case qualities (7 Cs) presented in this paper are a compromise: there is relatively little to memorize, and the checklist is contained entirely within one table. Hence, the 7 Cs checklist can be applied as one is writing a description, and also as one is reading a description.

The primary goal of applying the 7 Cs is to rapidly assess a description and then make any necessary changes so that a reader might be able to understand it more quickly and more completely. The use case writer should also consider the 7 Cs as the description is being drafted. By doing so, the writer should be able to produce a better, more communicable description with minimal additional effort. However, a further contribution of the work is that the 7 Cs may be seen as potentially independent measures of the quality of use case descriptions. This may then form the basis for assessment, which will allow researchers to both justify theoretically or underpin, and gauge empirically, proposed use case guidelines.

Therefore, either by direct application, or by their use in shaping further use case guidelines, these qualities of use case descriptions provide a simple mechanism to assess and improve upon the quality of use cases.

7.1 Recent developments and further work

To some extent there appears agreement, amongst those researchers who have conducted empirical studies, that guidelines and/or templates help the use case writer to produce descriptions of better quality. Indeed, latterly, the role of guidelines in inspecting use cases for defects (Cox et al., 2004; Somé, 2005), and in using use cases to support and inform design has also been investigated (Phalp and Cox, 2003b; Anda and Sjoberg, 2005). However, whilst interest in the use case as a requirements tool appears as popular as ever (Jorgensen and Lassen, 2005; Kanyaru and Phalp, 2005a), industrial acceptance of guidelines (which are often perceived as restrictive) is still somewhat limited.

The situation is perhaps even more clouded by the many suggestions that standard use cases are deficient, particularly in their handling of dependencies amongst events. Hence, a number of extensions to use cases have been proposed (e.g., Liang, 2003) and the authors are amongst those who have suggested that use cases be augmented with state-based information (Phalp and Cox, 2003a; Ratcliffe and Budgen, 2001, 2005). In our case, we suggest the addition of simple pre- and post-conditions within the standard use case structure (Kanyaru and Phalp, 2005c). Similarly, we have also suggested tool support, which allows for an automated check on some guidelines, whilst also promising other benefits, such as enacting intended behaviours (Kanyaru and Phalp, 2005b).

However, once again, any additional effort, even as training, on the part of the writer, may often be difficult to justify, and the freedom offered by use cases is perhaps hard to give up. Hence, it may be that application of what seemed the most basic of ideas—the simple checklist approach—may prove to be more palatable for industrial application, and may be a pragmatic way to improve upon the quality of use case descriptions.

References

Achour, C., Rolland, C., Maiden, N., Souveyet, C. 1999. Guiding use case authouring: Results from an empirical study. In: 4th IEEE International Symposium on Requirements Engineering, Limerick, 7–11 June 1999. Version taken from: http://sunsite.informatik.rwth-aachen.de/CREWS/reports.htm, updated August 1999. Report Series 98-31.

Adolph, S., Bramble, P., Cockburn, A., Pols, A. 2003. Patterns for Effective Use Cases. Addison Wesley.

Alexander, I. 2002. On abstraction in scenarios. Requirements Engineering Journal **6**:252–255.

Alexander, I. 2003. Misuse cases: Use cases with hostile intent. IEEE Software **Jan/Feb**:58–66.

Anda, B., Sjoberg, D., Jorgensen, M. 2001. Quality and understanding of use case models. In: Lindskov Knudsen, J. (ed.) 15th European Conference on Object-Oriented Programming, Budapest, June 18–22 2001, Berlin, LNCS, Springer-Verlag, pp. 402–428.

Anda, B., Sjoberg, D. 2005. Investigating the role of use cases in the construction of class diagrams. Empirical Software Engineering Journal **10**(3):285–309.

Arlow, J. 1998. Use cases, UML visual modelling and the trivialisation of business requirements. Requirements Engineering Journal **3**:150–152.

Booch, G., Rumbaugh, J., Jacobson, I. 1999. The Unified Modeling Language User Guide. Harlow, Addison-Wesley.

Bransford, J., Barclay, J., Franks, J. 1972. Sentence memory: a constructive versus interpretative approach. Cognitive Psychology **3**:193–209.

Bray, I. 2002. An Introduction to Requirements Engineering. Harlow, Addison-Welsey.

Budgen, D. 1994. Software Design. Harlow, Addison-Wesley.

Carroll, J. 2000. Five reasons for scenario-based design. Interacting with Computers **13**:43–60.

Clark, H. 1997. Dogmas of understanding. Discourse Processes **23**(3):567–598.

Cockburn, A. 2001. Writing Effective Use Cases. Harlow, Addison-Wesley.

Cox, K. 2000. Fitting scenarios to the requirements process. 2nd International Workshop on the Requirements Engineering Process: Innovative Techniques, Models, Tools to support the RE Process, London, 6–8 September 2000. In: Tjoa, A. Wagner, R. and Al-Zobaidie, A. (eds.), Proceedings of DEXA'2000, 11th International Workshop on Database and Expert Systems Applications, Los Alamitos, CA, IEEE Computer Society Press, pp. 995–999.

Cox, K. 2002. Heuristics for Use Case Descriptions. PhD Thesis, Bournemouth University.

Cox, K., Phalp, K. 2000. Replicating the CREWS use case authoring experiment. Empirical Software Engineering Journal **5**(3):245–268.

Cox, K., Phalp, K., Shepperd, M. 2001. Comparing use case writing guidelines. In: Achour-Salinesi, C., Opdahl, A., Pohl, K., Rossi, M. (eds.), 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland, 4–5 June 2001 Essen, Essener Informatik Beitrage, pp. 101–112.

Cox, K., Aurum, A., Jeffery, R. 2004. An experiment in inspecting the quality of use case descriptions. Journal of Research and Practice in Information Technology **36**(4):211–229.

Davis, A. 1991. Software Requirements Analysis and Specification. Hemel Hempstead, Prentice Hall.

Davis, A., Hickey, A. 2002. Requirements researchers: do we practice what we preach? Requirements Engineering Journal **7**:107–111.

Fletcher, C., van den Broek, P., Arthur, E. 1996. A model of narrative comprehension and recall. In: Britton B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 141–163.

Fowler, M., Scott, K. 2000. UML Distilled 2nd Edition. Harlow, Addison-Wesley.

Garnham, A., Oakhill, J. 1996. The mental models theory language of comprehension. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 313–339.

Galliers, R., Land, F. 1987. Choosing appropriate information systems research methodologies. Communications of the ACM **30**(11):900–902.

Garnham, A., Oakhill, J. 1996. The mental models theory language of comprehension. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 313–339.

Gause, D., Weinberg, G. 1989. Exploring Requirements: Quality Before Design. Dorset House Publishing, New York.

Gernsbacher, M. 1996. The structure-building framework: What it is, what it might also be and why. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 289–311.

Gernsbacher, M. 1997. Two decades of structure building. Discourse Processes **23**(3):265–304.

Goldman, S., Varma, S., Cote, N. 1996. Extending capacity-constrained construction integration: toward 'smarter' and flexible models of text comprehension. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 73–113.

Goldman, S., Graesser, A., van den Broek, P. 1999. Essays in honor of Tom Trabasso. In: Goldman, S., Graesser, A., van den Broek, P. (eds.), Narrative Comprehension, Causality and Coherence: Essays in Honor of Tom Trabasso, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 1–10.

Graesser, A., Britton, B. 1996. Five metaphors for text understanding. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 341–351.

Graesser, A., Swamer, S., Baggett, W., Sell, M. 1996. New models of deep comprehension. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 1–32.

Graham, I. 1998. Requirements Engineering and Rapid Development. Harlow, Addison-Wesley.

Halliday, M., Hasan, R. 1976. Cohesion in English. Harlow, Longman Group.

Ham, G. 1998. Four roads to use case discovery – there is a use (and a case) for each one. CrossTalk, December 1998. Version taken from: www.stsc.hill.af.mil/CrossTalk/1998 in Nov 2005.

Hsia, P., Yaung, A. 1988. Screen-based scenario generator: a tool for scenario-based prototyping. In: Proceedings of the 21st IEEE Conference on System Science, Hawaii, 4–7 January 1998 IEEE Computer Society Press, pp. 455–461.

Insfran, E., Pastor, O., Wieringa, R. 2002. Requirements engineering-based conceptual modelling. Requirements Engineering Journal **7**(2):61–72.

Jackson, M. 1995. Software Requirements and Specifications: A Lexicon on Principles, Prejudice and Practice. Wokingham, Addison-Wesley

Jackson, M. 1998. A discipline of description. Requirements Engineering Journal **3**:73–78.

Jackson, M. 2001. Problem Frames. Harlow, Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. Wokingham, Addison-Wesley.

Jarke, M., Tung Bui, X., Carroll, J. 1998. Scenario management: An interdisciplinary approach. Requirements Engineering Journal **3**(3/4):155–173.

Jorgensen, J.B., Lassen, B. 2005. Aligning Work Processes and the Advisor Portal Bank System, 1st International Workshop on Requirements Engineering for Business Need and IT Alignment, (REBNITA 2005), 13th IEEE International Requirements Engineering Conference, RE 2005, Paris, 29 Aug–2 Sep, 2005.

Kaakinen, J., Hyona, J., Keenan, J. 2002. Perspective effects on online text processing. Discourse Processes **33**(2):159–173.

Kaindl, H. 1998. Combining goals and functional requirements in a scenario-based design process. In: Johnson, H., Nigay, L., Roast, C., (eds.), People and Computers XIII: Proceedings of HCI'98, Sheffield, September, London, Springer-Verlag, pp. 101–121.

Kanyaru, J., Phalp, K. 2005a. Supporting the consideration of dependencies in use case specifications. In: 11th International Workshop on Requirements Engineering: Foundation For Software Quality - REFSQ05, Porto, Portugal, 13–14 June 2005.

Kanyaru, J., Phalp, K. 2005b. Requirements validation with enactable models of state-based use cases, Empirical Assessment in Software Engineering, EASE 2005, Keele University, 11–13 April 2005.

Kanyaru, J., Phalp, K. 2005c. Aligning Business Process Models with Specifications using Enactable Use Case Tools, 1st International Workshop on Requirements Engineering for Business Need and IT Alignment, (REBNITA 2005), 13th IEEE International Requirements Engineering Conference, RE 2005, Paris, 29 Aug–2 Sep, 2005.

Korn, J. 2000. Scenarios through linguistic modelling. In: IEE Seminar on Scenarios in the System Lifecycle (Ref 00/138), London, 7 December 2000, IEE, pp. 3/1–3/7.

Kovitz, B. 1999. Practical Software Requirements: A Manual of Content and Style. Manning Publications, Greenwich, CT.

Kulak, D., Guiney, E. 2000. Use Cases – Requirements in Context. Harlow, Addison-Wesley.

Leibundgut, R. 2002. Use Cases in the Project Lifecycle, Presented at Requirements Engineering Specialist Group of the British Computer Society Workshop: Scenarios Work! Improving Requirements Engineering with Use Cases and Scenarios, July 10th 2002, UC London, available from: http://mcs.open.ac.uk/computing/resg2/documents/Lei.bundgut.pdf.

Liang, Y. 2003. From use cases to classes: A way of building object model with UML. Information and Software Technology **45**(2):83–93.

Magliano, J. 1999. Revealing inference processes during text comprehension. In: Goldman, S., Graesser, A., van den Broek, P. (eds.), Narrative Comprehension, Causality and Coherence: Essays in Honor of Tom Trabasso, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 55–75.

Mannes, S., St George, M. 1996. Effects of prior knowledge on text comprehension: a simple modelling approach. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 115–139.

Mattingly, L., Rao, H. 1998. Writing effective use cases and introducing collaboration cases. Journal of Object-Oriented Programming **October**:77–87.

McNamara, D., Kintsch, W. 1996. Learning from texts: effects of prior knowledge and text coherence. Discourse Processes **22**:247–288.

Object Management Group (OMG) 2001. Unified Modeling Language v1.4 – Semantics. Document 01-09-73. Version taken from: http://www.omg.org/pub/docs/formal/01-09-73.pdf, taken Jan 2002.

O'Brien, E., Myers, J. 1999. Text comprehension: a view from the bottom up. In: Goldman, S., Graesser, A., van den Broek, P. (eds.), Narrative Comprehension, Causality and Coherence: Essays in Honor of Tom Trabasso, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 35–53

Ozyurek, A., Trabasso, T. 1997. Evaluation during understanding of narratives. Discourse Processes **23**:305–335.

Perfetti, C. 1997. Sentences, individual differences and multiple texts: Three issues in text comprehension. Discourse Processes **23**:337–355

Phalp, K., Cox, K. 2002. Supporting communicability with use case guidelines: An empirical study. In: 6th International Conference on Empirical Assessment in Software Engineering, Keele. Keele University, 8–10 April 2002.

Phalp, K.T., Cox, K. 2003a. Using Enactable Models to Enhance Use Case Descriptions, ProSim'03, International Workshop on Software Process Simulation Modelling (in conjunction with ICSE 2003), Portland, USA, May 3–4 2003.

Phalp, K.T., Cox, K. 2003b. Exploiting use case descriptions for specification and design. In: 7th International Conference on Empirical Assessment and Evaluation in Software Engineering (EASE 2003), Keele University, Staffordshire, UK, April 8–10th, 2003.

Pohl, K., Brandenburg, M., Gulich, A. 2001. Integrating requirements and architecture information: a scenario and meta-model based approach. In: Achour-Salinesi, C., Opdahl, A., Pohl, K., Rossi, M. (eds.), 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland, 4–5 June 2001 Essen, Essener Informatik Beitrage, pp. 68–84.

Pooley, R., Stevens, P. 1999. Using UML—Software Engineering with Objects and Components. Harlow, Addison-Wesley.

Potts, C. 1993. Software engineering research revisited. IEEE Software **September**:19–28.

Ratcliffe, M., Budgen, D. 2001. The application of use case descriptions in system design specification. Information and Software Technology **43**(6):365–386.

Ratcliffe, M., Budgen, D. 2005. The application of use cases in systems analysis and design specification. Information and Software Technology **47**(9):623–641.

Regnell, B., Davidson, A. 1997. From requirements to design with use cases – experiences from industrial pilot projects. In: Dubois, E., Opdahl, A., Pohl, K. (eds.), 3rd International Workshop on Requirements Engineering: Foundation for Software Quality, Barcelona, 16–17 June 1997, Namur University Press.

Regnell, B., Kimber, K., Wesslen, A. 1995. Improving the use case driven approach to requirements engineering. In: 2nd International Symposium on Requirements Engineering, York, March 1995 Los Alamitos, CA, IEEE Computer Society Press, pp. 40–47.

Robertson, S. 1995. Generating object-oriented design representations via scenario queries. In: Carroll, J. (ed.). Scenario-Based Design: Envisioning Work and Technology in System Development, Chichester, Wiley, pp. 279–308.

Robertson, S. 2001. Are we afraid of the dark? IEEE Software **July/August**:12–15.

Rosenberg, D., Scott, K. 1999. Use Case Driven Object Modelling with UML: A Practical Approach. Harlow, Addison-Wesley.

Rosenberg, D., Scott, K. 2001. Applying Use Case Driven Object Modeling with UML. An Annotated E-Commerce Example. Reading, MA, Addison-Wesley.

Schneider, G., Winters, J. 1998. Applying Use Cases: A Practical Guide. Harlow, Addison-Wesley.

Somé, S. 2005. Supporting use case based requirements engineering, Information and Software Technology (accepted – now available online Articles in Press).

Trabasso, T., Van Den Broek, P., Suh, S. 1989. Logical necessity and transitivity in causal relations in stories. Discourse Processes **12**:1–25.

Traxler, M., Gernsbacher, M. 1995. Improving coherence in written communication. In: Gernsbacher, M., Givon, T. (eds.), Coherence in Spontaneous Text, Philadelphia. John Benjamins, pp. 215–237.

Turner, A., Britton, B., Andreaessen, P., McCutchen, D. 1996. A predication semantics model of text comprehension and recall. In: Britton, B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 33–71.

van den broek, P., Risden, K., Fletcher, C., Thurlow, T. 1996. A 'landscape' view of reading: fluctuating patterns of activation and the construction of a stable memory representation. In: Britton B., Graesser, A. (eds.), Models of Understanding Text, Mahwah, NJ, Lawrence Erlbaum Associates, pp. 165–187.

Wason, P., Reich, S. 1979. A verbal illusion. Quarterly Journal of Experimental Psychology **31**:591–597.

Wieringa, R. 2001. Software requirements engineering: the need for systems engineering and literacy. Requirements Engineering Journal **6**:132–134.

Yin, R. 1994. Case Study Research: Design and Methods, 2nd Edition. London, Sage Publications.

Zuber-Skerritt, O. editor, 1996. New Directions in Action Research. London, Falmer Press.

Keith Phalp originally read for a first degree in Mathematics, which he then taught for a few years, before completing a Masters in Software Engineering in 1991, followed by PhD in process Modelling in 1994. He then spent three years as a post-doctoral research fellow at the University of Southampton, again in the area of process modelling. In 1997, Dr Phalp took up a lectureship at Bournemouth, and became course leader for the Masters in Software Engineering. He has been there ever since, and during that time has taught units covering the majority of the software development process. He is currently Reader in Software Engineering within the Software Systems Modelling Group at Bournemouth University, which he co-founded with Dr Jonathan

Vincent. His research focuses on requirements engineering, and its relationship to both business needs and to the specification and design of systems.

Jonathan Vincent has a BEng in Electrical and Electronic Engineering, an MSc in Computing (Software Engineering) and a PhD in Computer Science. He is currently a Reader in Natural Computing within the School of Design, Engineering and Computing, at Bournemouth University, UK, where he directs the Software Systems Modelling Group (www.sosym.co.uk). He has wide ranging research interests and has published in a variety of areas within software engineering and computer science, including component based software engineering, software quality, modelling, evolutionary computation and neural networks.

Karl Cox has a Masters degree in Software Engineering and a PhD in Computer Science, both from Bournemouth University. Dr Cox's research interests are centred on requirements engineering, specifically: the Problem Frames approach as a means of providing a framework for understanding the problem context of business needs; goal modelling, combined with problem frames, as a means of describing business goals, strategies, and objectives that are aligned to software; process modelling, which captures the details of processes that businesses implement to carry out their daily work; and use cases, which is concerned with ways to improve the comprehensibility of use case descriptions, and the misunderstanding and misuse of use cases that often occurs. Prior to joining NICTA, Dr Cox was a Research Fellow in the School of Computer Science and Engineering at the University of New South Wales (UNSW), Sydney, Australia, and Lecturer at Bournemouth University in the UK.