IMPROVING MUTUAL UNDERSTANDING OF DEVELOPMENT ARTIFACTS: A SEMIOTICS BASED APPROACH

Joerg Evermann
Victoria University of Wellington
jevermann@mcs.vuw.ac.nz

Gary Haggard
Bucknell University[1]
haggard@bucknell.edu

Jennifer Ferreira
Victoria University of Wellington
Jennifer.Ferreira@mcs.vuw.ac.nz

Abstract

The success of information system development projects is critically dependent on arriving at a shared understanding of the desired outcome by all project stakeholders. To communicate such understanding among stakeholders, development artifacts such as design specifications, prototypes, and user stories are created. This paper presents a new project management tool to measure and ensure common understanding of such development artifacts. The method is based on Peirce's theory of semiotics and Mayer's theory of learning. The application of the method is demonstrated using a brief example.

***Keywords**: Semiotics, Understanding, IS development*

Introduction

System development projects commonly involve multiple stakeholder groups. Typically, users and developers form the main groups. Communication among project participants takes place by means of, or is supported by, various system development artifacts (Kung and Solvberg, 1986). In traditional, well structured development approaches such as the waterfall model and the Unified Process (Hunt, 2000), these artifacts may be formal or informal models of the problem domain or software system. In agile methodologies (Beck, 2000), the development artifacts may be user stories, and in rapid application development approaches, they may be prototypes.

These development artifacts may serve different communicative purposes. For example, a conceptual domain model is used to convey understanding of the problem domain between users and developers (Kung and Solvberg, 1986). User stories are used to convey and confirm understanding of the expected functionality of the final software system between users and developers (Beck, 2000). In all cases, mutual understanding of these different aspects by all participating stakeholders is necessary for the success of information system development (Iscoe *et al.*, 1991; Jackson, 1995; Offen, 2002).

For example, given a user story in eXtreme Programming, both the developers and the users should have the same understanding of what this user story implies for the final system, in order for the development effort to be successful. Similarly, a conceptual model in system analysis should convey the same understanding to the developers and the users or other stakeholders. Hence, during the lifetime of a development project, users and developers have to *continuously* check and revise their understanding of the problem domain, or their expectation of the system's capabilities, in order for their

[1] Gary Haggard was on sabbatical at Victoria University of Wellington for the duration of this research.

understanding to converge.
In this paper we present a method of measuring mutual understanding among different groups of system development stakeholders, based on a development artifact. We also show how this understanding can be assessed over time and can inform specific changes to the development artifact in order to improve mutual understanding. This method is intended as a tool for project management in the early stages of IS development projects. Here, mixed audiences with different levels of expertise and experience, are brought together to agree on IS development artifacts.
The proposal presented here is the initial proposal in a larger research program. While this paper shows the applicability of the proposed method using a single case and a specific type of development artifact (UML class diagrams), future research will examine other types of design artifacts, a realistic project size and time-line, and assess the specific benefits of the proposal.
The remainder of the paper is structured as follows. In the next secion we briefly highlight relevant literature on understanding of development in artifacts in system development. This is followed by the theoretical foundation for our work. We then present our method for measuring and improving mutual understanding of a development artifact. Following this, we give an example application of this method. We conclude the paper with a discussion and outlook to future work.

Previous Work

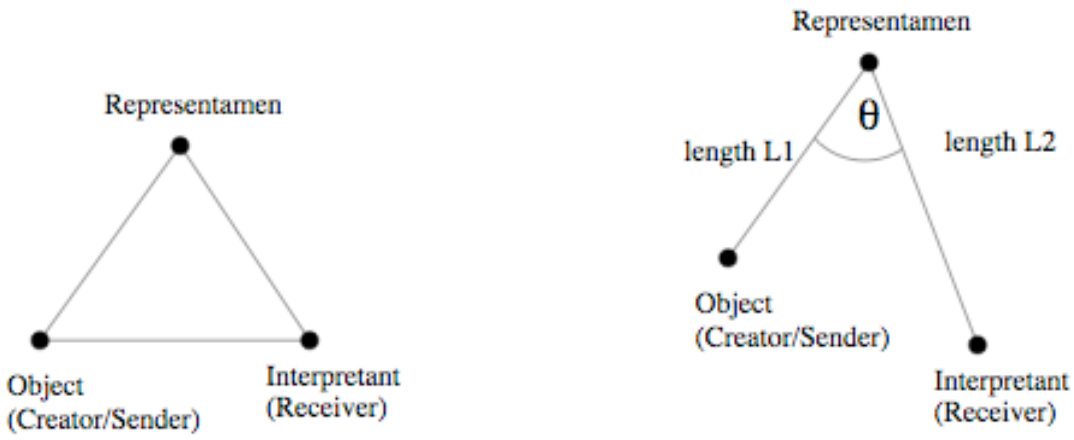
Understanding of conceptual models has been examined to a great extent in the context of information system analysis. In this context, the artifacts are used for fostering understanding and communication (Kung and Solvberg, 1986). Mylopoulos emphasizes the importance of mutual understanding when he states that "The adequacy of a conceptual modelling notation rests on its contribution to the construction of models of reality that promote a *common understanding* of that reality among their human users" (Mylopoulos, 1992, emphasis added).
Mutual understanding has been shown to be an important factor in reducing risks to system implementation (Offen, 2002) by facilitating requirements engineering and elicitation (Iscoe *et al.*, 1991; Jackson, 1995) and by reducing costly rework later in the development process (Boehm, 1988). Lind and Zmud (1991) showed that a convergence in understanding between technology providers and users promotes organizational innovativeness. Research on understanding of development artifacts has examined the understanding conveyed through data flow diagrams (Hungerford *et al.*, 2004), the effects of the size of the artifact on understanding (Bajaj, 2004), the effect of different modelling options on understanding (Bodart *et al.*, 2001), cognitive integration of multiple development artifacts (Kim *et al.*, 2000; Parsons, 2002), and other human factors (Topi and Ramesh, 2002). Most of these studies have examined understanding of conceptual models, such as UML and ER diagrams, during system analysis.
In the research on understanding of development artifacts, the work by Gemino on operationalizing the concept of understanding has been of particular importance (Gemino, 1999; Gemino and Wand, 2001, 2003, 2005). Gemino's work builds on work in educational psychology by Mayer (1989). Mayer's theory of learning proposes that understanding is the outcome of a learning process, and is distinct from comprehension, which is the outcome of the memorization process (Mayer, 1989). Consequently, understanding is demonstrated not by recall from memory, but by application of knowledge to problems beyond those used for training. Hence, Gemino suggests that understanding can be measured by examining problem solving performance, rather than comprehension performance (Gemino, 1999). For example, Evermann and Wand (Evermann and Wand, 2006) use five problem solving questions to determine understanding of a UML class diagram. In fact, comprehension or recall from memory have been shown to be distinct from problem solving in all studies that employed Gemino's operationalization of understanding (Bodart *et al.*, 2001; Burton-Jones and Meso, 2002; Evermann and Wand, 2006; Gemino and Wand, 2001, 2003, 2005).

Theoretical Foundation

As discussed in the introduction, software development is an imperfect process whose success relies heavily on the mutual understanding of the various stakeholders involved. Mutual understanding between stakeholders is generally not present at the start of the project. Every stakeholder group may be proficient in a different domain, i.e. the users in their domain and the developers in developing computer systems. Due to the different backgrounds of the stakeholders, and their different proficiencies, there will inevitably be mismatches in the way they interpret various artifacts. Developers who are trained to develop and understand object models created during the development process can not expect their users, with no software

development training, to attach the same meanings to those artifacts. Yet, it has been shown that mutual understanding between stakeholders is achievable by means of adequate communication — a dynamic process of exchanging information (Clark and Brennan, 1991).

In the study of *semiotics* (the doctrine of signs), the American philosopher Charles Sanders Peirce (Peirce, 1931) developed a model that showed a sign can have an intended meaning and a meaning generated by an independent observer. These two meanings may not necessarily coincide and studies have shown that this implies a failure on the part of the creator of the sign to communicate some message. Looking at Peirce's model in more detail, we see that a sign is a triadic relationship of three parts: the *representamen*, the *object*, and the *interpretant* (Figure 1 (a)). The object is what is to be denoted or signaled, what the sign is intended to represent. It is the meaning or understanding attached to the sign by its creator. The representamen is the physical signal or sign, the artifact that is created to represent the object. The interpretant is the understanding or meaning created in the mind of the perceiver of the sign. During the development process, developers may create artifacts, such as UML diagrams or user stories, as signs to communicate their understanding of certain aspects of the system to the user. The object of a development artifact is the understanding that the developer intends to communicate by means of the artifact. The interpretant is the understanding that the artifact creates in the mind of the user, while the representamen is the actual development artifact.



(a) Peirce's triangle of meaning **(b) Differences in meaning**

Figure 1. Peirce's elements of the sign

From Peirce's model it is clear that the nature of the sign is such that there may be a mismatch between the object and the interpretant, e.g. the understanding of the user and the understanding of the developer, which is the case when the creator of the sign's intended meaning is different from the understanding of the perceiver. This mismatch can occur both in the amount of understanding (quantity), as well as in the kind of understanding (quality). Figure 1 (b) shows how these differences can be understood in terms of Peirce's triangle of meaning, as the differing lengths of the sides (quantitative difference), and the angle formed by the sides of triangle (qualitative difference).

In the context of a software development process, when the users' understanding of a development artifact is different from that of the developers, it could lead to unfulfilled expectations of the eventual system for both parties. To avoid such an undesirable outcome, this mismatch in understanding needs to be resolved through continued refinement and testing of the sign and the understanding. We therefore propose a method to not only measure how closely the user's perceived meaning matches the developer's intended meaning of the artifact, but also to track the changes in understanding over time. The method's dynamic ability to track these changes allows us to identify specific ways of improving the artifact.

Proposed Method

Building on Mayer's theory of learning (Mayer, 1989) and following the work of Gemino (Gemino, 1999), reviewed in above, we propose to operationalize meaning in terms of understanding. We point out two important aspects of previous

work that lend themselves to extension. First, prior work operationalized understanding as uni-dimensional. It was quantified in a single, scalar, number. Second, the understanding was assessed against an assumed correct standard, based on the researcher's understanding and validated against secondary experts. The following subsections describe our extensions of these two aspects.

Understanding as a Multidimensional Concept

First, we extend the dimensionality of the understanding construct. Previous work (reviewed above) used multiple problem solving questions, the scores on those questions were averaged or summed to form a uni-dimensional measure expressing the *amount* of understanding. For example, Evermann and Wand (Evermann and Wand, 2006) use five problem solving questions to determine understanding of a UML class diagram. For their further analysis, they use average scores.

While we are interested in the *amount* of understanding conveyed by a development artifact (quantity), we are also interested in the *type* of understanding conveyed by a development artifact (quality). Thus, we understand each question as a unique dimension of understanding, in effect leading us to characterize understanding as a vector in an n-dimensional understanding space. Figure 2 shows such an *understanding vector* in an example 3-dimensional understanding space, spanned by the example dimensions of functionality, data content, and security.

We emphasize that the dimensions of understanding depend on the purpose of the development artifact. For example, user stories in extreme programming are intended to convey functionality of the final artifact, while UML sequence diagrams may be used to convey information about the real-time behaviour of a particular method implementation. Consequently, different dimensions of understanding are necessary.

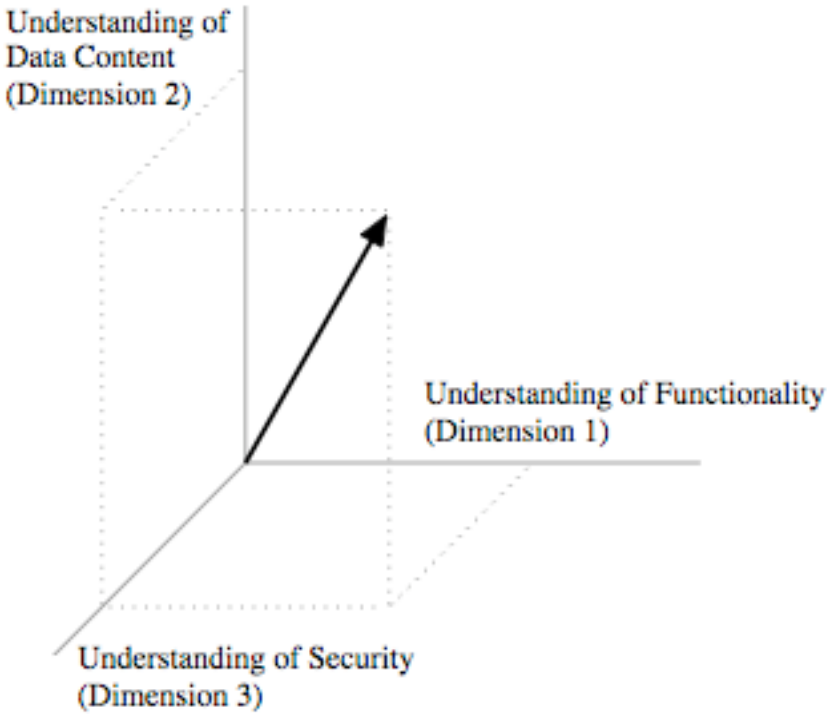


Figure 2. Example n-dimensional understanding

Mutual Understanding

Second, rather than comparing the understanding of a group of development stakeholders against an assumed correct standard (e.g. as interpreted by experts), we extend the methodology to compare the understanding arrived at by two groups

of development stakeholders, typically users or clients and developers or analysts. However, these may also be two groups of developers or two groups of users, etc. While it may be argued that giving up an external, objectively "correct" standard understanding can lead to both groups arriving at mutual understanding of the "wrong" type, real projects do not provide any external, "correct" understanding. Instead, it is the developers' understanding of users' requirements that is necessary for successful development.

These two extensions allow us to assess both the quantitative and qualitative differences in understanding, i.e. differences in the amount of understanding as well as differences in the kind of understanding. We contend that while quantitative differences, as measured by previous work, are important, it is the qualitative differences in understanding that are most important to confirm the customer's or user's expectations. In other words, while the depth of understanding of a domain is important, it is differences in the kind of understanding that are damaging to project success.

We operationalize the differences in understanding in terms of characteristics of the understanding vectors ϕ_1, ϕ_2 in an *n-dimensional* understanding space. The *quantitative understanding* of each party is the length $|\phi|$ of an understanding vector. Consequently, the *quantitative difference in understanding* is the difference in the lengths of the understanding vector:

$$\Delta_{quan} = abs\left(|\phi_1| - |\phi_2|\right)$$

We define the angle θ between the two understanding vectors ϕ_1, ϕ_2 as the *qualitative difference in understanding*.

$$\cos(\theta) = \frac{\phi_1 \cdot \phi_2}{|\phi_1||\phi_2|}$$

$$\Delta_{qual} = \theta$$

Figure 3 shows two example understanding vectors in an example 3-dimensional understanding space, and the angle θ between them, signifying the qualitative difference between the two understanding vectors.

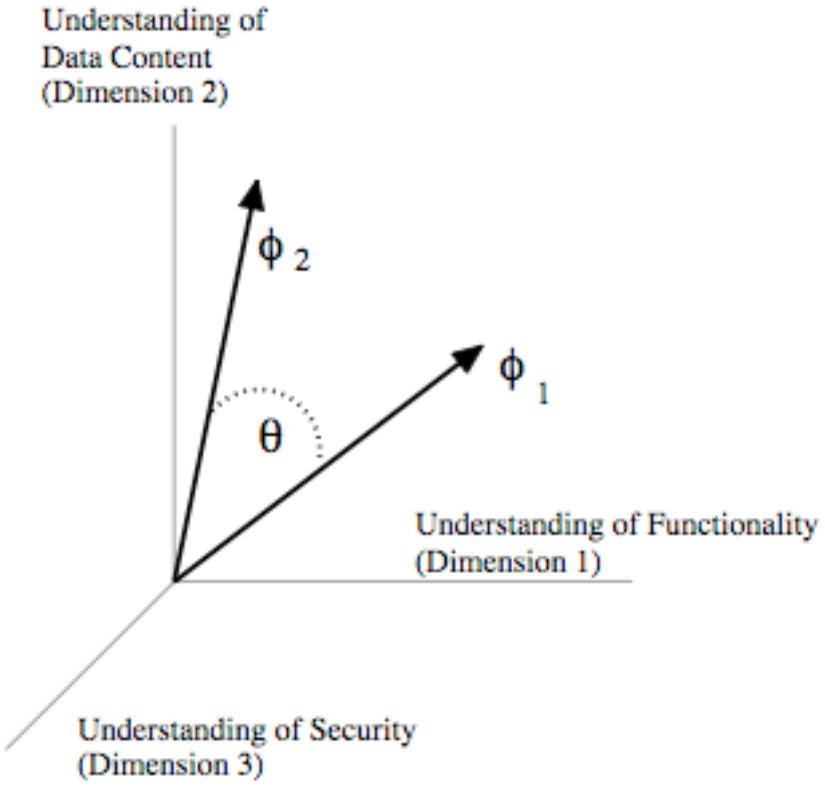


Figure 3. Angle between two understanding vectors

Improving Mutual Understanding

In the context of system development, it is important that the understanding of a design artifact by users matches the developer's understanding of the design artifact. As this is unlikely to occur immediately, it becomes necessary to track both qualitative and quantitative differences in understanding across time.
While tracking differences over time is necessary, it is not sufficient to merely measure qualitative understanding. A further necessary condition is the ability to use the identified differences in understanding to identify problematic aspects of the understanding. By maintaining the understanding vectors of both parties, rather than collapsing them into a uni-dimensional numbers, we can identify the dimensions of greatest differences. For example, Figure 4 (following page) shows the differences between two understanding vectors ϕ_1, ϕ_2 as projections on each understanding axis.

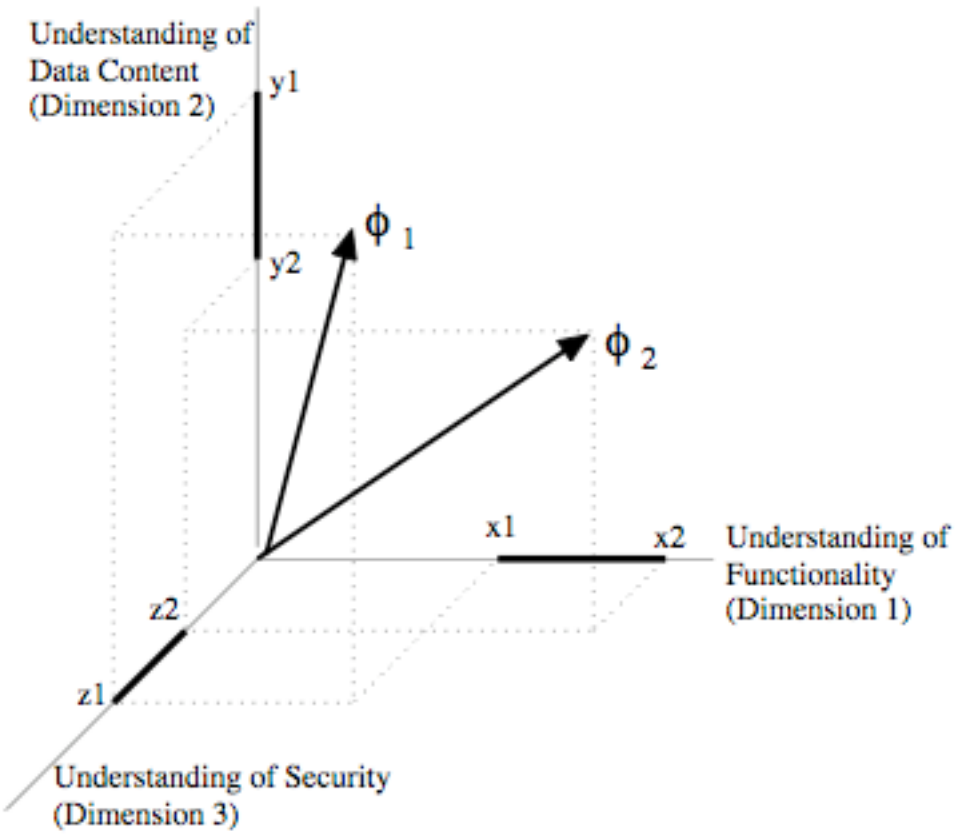


Figure 4. Example differences in individual understanding dimensions

The final necessary condition for improving mutual understanding is the requirement of traceability back to the development artifact. To effectively alter the development artifact to ensure qualitatively convergent understanding, it is necessary to be able to trace each dimension of the understanding vector to that particular aspect of the development artifact that most heavily influences understanding on that dimension. Figure 5 on the following page illustrates how each problem solving question measures understanding on one dimension of the understanding space. In turn, each dimension of the understanding space is traceable to a clearly defined aspect of the development artifact.
In this paper, we focus on analysis artifacts, as the dimensions of understanding are well understood from prior research. However, the method is also applicable to other development artifacts, such as user stories or use-case descriptions, rapid application prototypes, and software design specifications. Each of these artifact types requires adapting the problem solving questions to the particular purpose of the artifact and (re-)validation. In summary, this section has discussed some requirements for the problem solving questions:

- They must be related to relevant dimensions of understanding the domain. Relevance depends on the type of

understanding that is considered most critical to project success and is therefore application dependent.

- They must refer to a specific aspect of the development artifact, so that traceability is ensured
- They must be understandable to and answerable by all involved stakeholder groups.

Hence, project management needs to establish the relevant dimensions and questions in consultation with all involved stakeholder groups, and provide a process to measure mutual understanding over time. The specific questions depend on the purpose of the project, the type of development and the purpose of the development artefact.

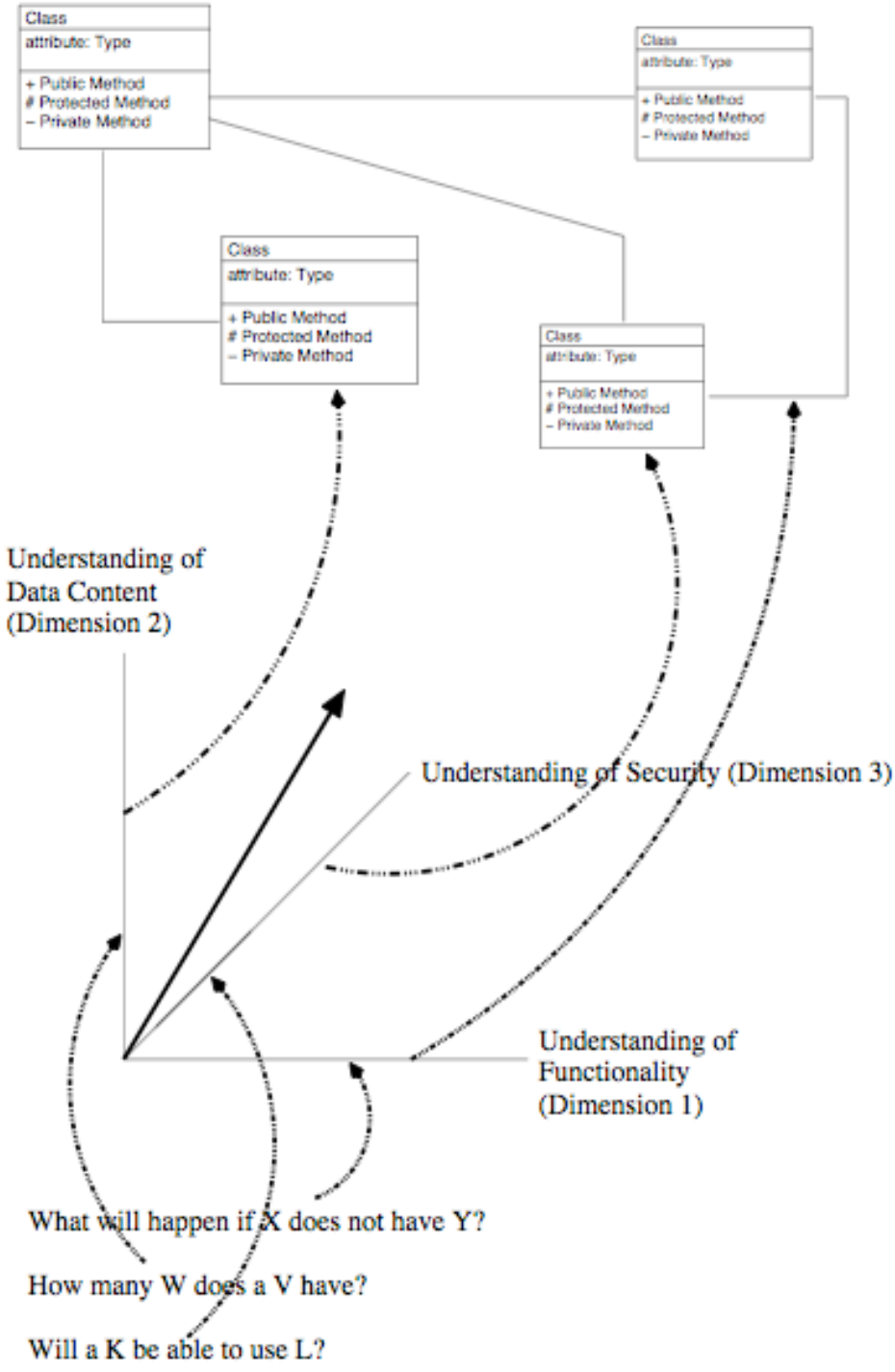


Figure 5. Traceability from understanding dimension to development artifact

Demonstration

We demonstrate the application of this method to object-oriented system analysis. The proposed method of measuring mutual understanding is a tool for project management in the early stages of IS development, e.g. during analysis and design. It allows project managers to assess whether diverse groups of stakeholders agree in their understanding of development artifacts. This demonstration is therefore best understood as a single case study that demonstrates the application of a project management tool that is based on IS research methods (Gemino, 1999; Gemino & Wand, 2001; 2003; 2005), rather than as an experimental study. The results presented here indicate the feasibility of the proposed method and demonstrate how it could be applied. Further research on this method is required to evaluate its use in different project contexts, e.g. for different stakeholder groups, different development artifacts, etc.

The example study involved two diverse groups of participants: Computer science students (being surrogates for developers) and business students (being surrogates for customers or users). Each participant was given the UML class diagram for a car-rental domain taken from (Fowler and Kendall, 2000), shown in Figure 6.

The application to class diagrams is for illustration purposes only. We recognize that the communication between users and developers is not necessarily based on class diagrams but may involve user stories, use cases etc. Our method is applicable to these kinds of development artifacts as well. Also, the method does not examine whether class diagrams in general are understandable by the involved stakeholder groups, but rather whether the content conveyed by a particular class diagram is understood in similar ways.

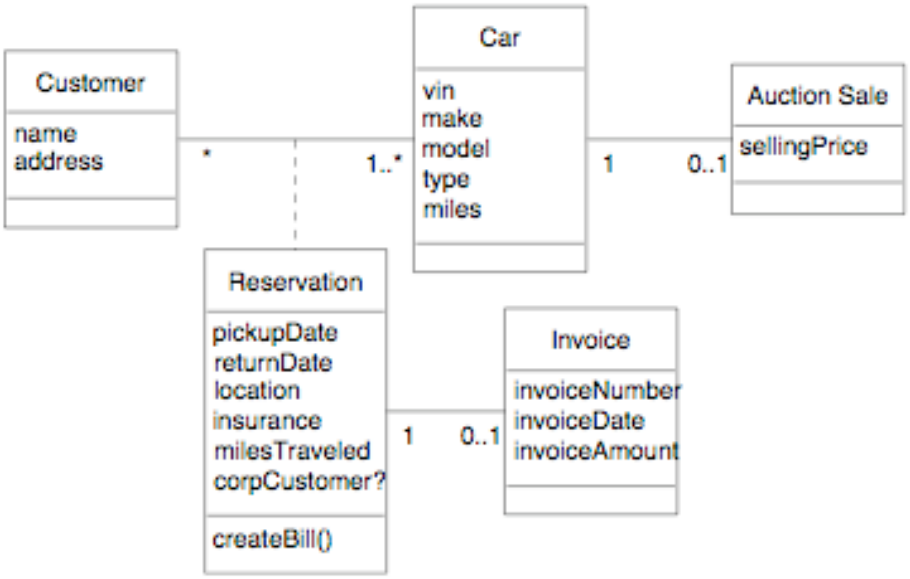


Figure 6. Initial class diagram

In this case, the UML class diagram's purpose was to convey understanding about the problem domain. Consequently, the problem solving questions were designed to test for understanding of the car-rental domain. The following five questions were designed with references to the requirements for such questions stated above, and cover all aspects of the initial diagram.

1. Suppose that a customer arrives for pick-up but no car is available. What could have happened?
2. Suppose a customer receives two invoices. What could have happened?
3. Suppose a car which is reserved for a customer is being sold at auction. How could this have happened?
4. Suppose a customer is not billed for a reservation? What could have happened?
5. Suppose that a customer wants to extend his rental period. What could go wrong?

The problem solving questions were patterned after Gemino's work (Gemino, 1999, Gemino and Wand, 2001, 2003, 2005). Each of the problem solving questions refers to a specific aspect of the original diagram. Note that the diagram contains many more than five elements, so that it is not possible to trace from each question to a single diagram element, but only to a set of related diagram elements.

Subjects in both groups were asked to answer the questions. The average number of correct answers per subject for each question for each of the two groups formed the raw scores for further analysis. To compute the differences in understanding, the two understanding vector of scores on the five questions were first normalized to unit length, in order to make the differences on individual dimensions comparable. The column "Diagram 1" of Table 1 shows the differences between the two groups on the first diagram. The qualitative difference in understanding (angle between the two vectors) was 24.2°. The quantitative difference in understanding (difference of the vector lengths) was only 0.3.

Table 1: Differences in understanding

	Differences		
Question	Diagram 1	Diagram 2	Diagram 3
1	0.11	**0.17**	0.09
2	**0.26**	0.05	0.05
3	0.11	0.15	0.06
4	0.04	0.13	0.01
5	0.01	0.09	0.14
$\lvert \phi_{CPSC} \rvert$	3.1	2.7	2.9
$\lvert \phi_{BUSI} \rvert$	2.8	7.5	4.5
Δ_{qual}	24.2°	16.4°	10.8°
Δ_{quan}	0.3	4.8	1.6

Table 1 shows that initially the primary understanding problem was with the second dimension, regarding the invoices (emphasized). Participants show smaller differences on the remaining four understanding dimension. The second understanding dimension is assessed by the second problem solving question. This question can be traced to the class "Invoice" that is associated with the class "Reservation". Hence, the class diagram was modified (Figure 7) to address the issues identified by the problem solving questions. As the length of the understanding vectors of both groups was relatively small (3.1, 2.8), more detail was introduced overall to determine whether this would improve quantitative understanding. The relevant excerpt of the resulting diagram is shown in Figure 7.

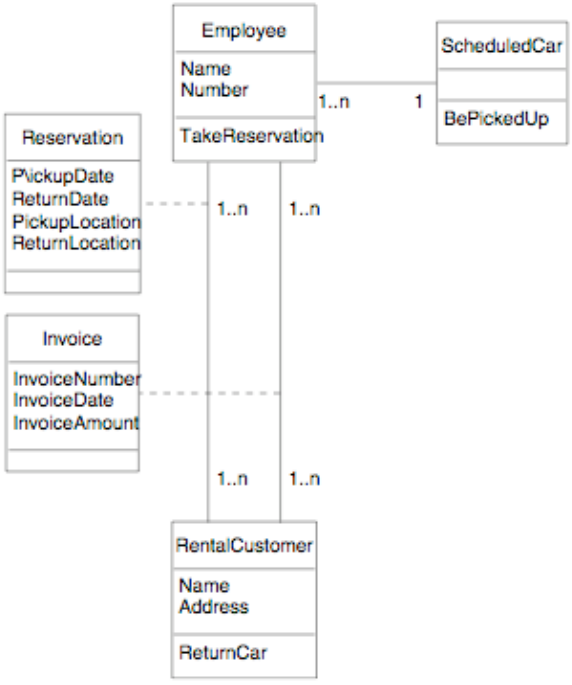


Figure 7. Excerpt of intermediate class diagram corresponding to understanding dimensions 2 (problem solving question 2)

In order to avoid carry-over and learning effects that might obscure the effect of diagram changes, two different groups of participants were used to examine this second diagram. While we prevented such effects for demonstration purposes, they are not a problem in a real application of this method, as they work towards common understanding.
The column "Diagram 2" of Table 1 shows the differences between the two groups. While this diagram shows a reduction in the divergence on the second dimension, the understanding differences on other dimensions have increased. Especially the first understanding dimension, focusing on car scheduling, now shows major differences (emphasized in Table 1). Consequently, a further modification was undertaken. As expected, due to the inclusion of additional model detail, the amount of understanding, described by the length of the vector, increased significantly for business students, from 2.8 to 7.5, but did not for computer science students (3.1 to 2.7). Despite these differences in quantitative understanding, their qualitative understanding has converged.
Figure 8 shows an excerpt of the final modifications of the class diagram. This diagram was examined by a third group each of business and computer science students, different again from the second groups. The results are shown in the final column of Table 1 and show a divergence of only 10.8°. The data shows only minor differences on each dimension.

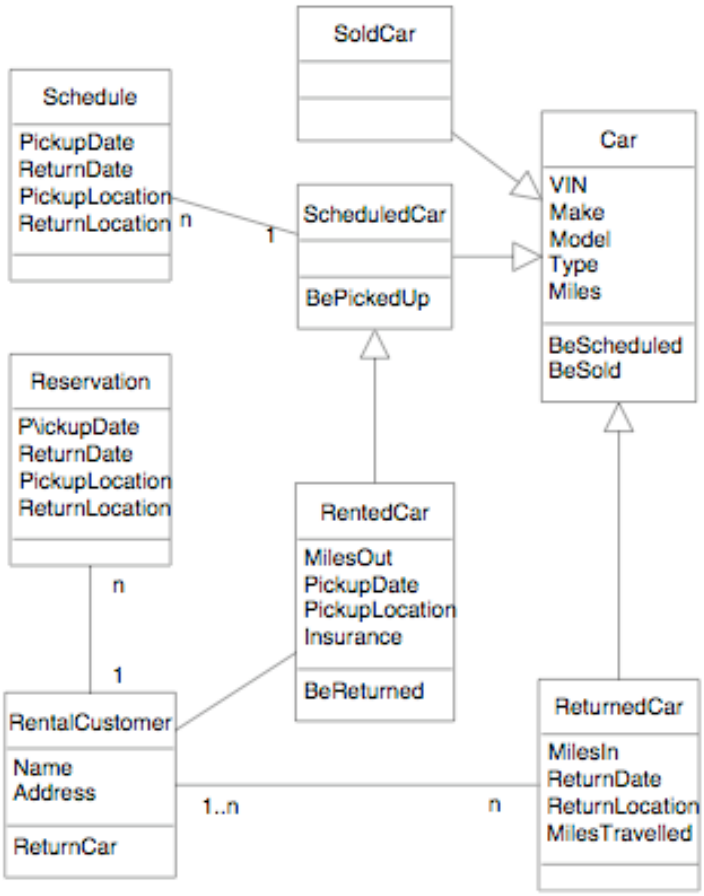


Figure 8. Excerpt of final class diagram corresponding to understanding dimension 1 (problem solving question 1)

To determine which group's understanding was affected more by the changes, we examine the differences between the three understanding tests *within the groups* (Table 2). The quantitative understanding of the domain by computer scientists of the three diagrams remains relatively stable with the lengths of the understanding vectors varying by 0.4 and 0.2 (Table 2) between the three diagrams (from 3.1 on Diagram 1 to 2.7 on Diagram 2 to 2.9 on Diagram 3, Table 1). The angle between the first and second understanding vector of computer science subjects was 4.4° and that between the second and third understanding vector was 9.9° (Table 2). This indicates that also the qualitative understanding for the three diagrams changed little for computer science subjects.

Table 2. Differences in qualitative understanding (within developer groups)

	Differences between Diagrams	
	Diagram 1 → Diagram 2	Diagram 2 → Diagram 3
Δ_{quan}(CPSC)	0.4	0.2
Δ_{qual}(CPSC)	4.4°	9.9°
Δ_{quan}(BUSI)	4.7	3.0
Δ_{qual}(BUSI)	27.7°	20.6°

On the other hand, for business subjects, the length of the understanding vectors is highly variable (2.8, 7.5, 4.5, Table 1). Also, the angles between the first and second understanding was 27.7° and that between the second and third understanding vector was 20.6° (Table 2). These large angles show that the qualitative understanding changes considerably between the three diagrams. Consequently, the convergence towards mutual understanding is largely due to the changing understanding of the domain by business subjects.

Discussion and Conclusion

We have developed a new method to improve common understanding of IS development artifacts over time. The proposed method of measuring mutual understanding is a tool for project management in the early stages of IS development, e.g. during analysis and design. It allows project managers to assess whether diverse groups of stakeholders agree in their understanding of development artifacts and to track and improve mutual understanding over time. The method is intended for use in actual project teams, whose size may range from less than five to only a few dozen members in large teams.

We have demonstrated the application of this method to object-oriented system analysis. Note that this is not an experimental study in the sense of a rigorous scientific experiment. Instead, we have shown a single case that demonstrates the application of the method. Therefore, the results presented here only indicate the feasibility of the proposed method and demonstrate how it could be applied. Further research on this method is required to evaluate it in different project contexts, e.g. for different stakeholder groups, different development artifacts, etc.

The proposed method shows how a technique that was originally developed for IS research (Gemino, 1999; Gemino and Wand, 2003; 2005) can be extended and applied in IS development practice. One of the contributions is therefore the knowledge transfer from research to practice. However, the extensions presented in this work can also contribute back to the field of IS research where the notion of understanding is used, so that a contribution to research is also made.

The example case has shown that the technique is applicable and can yield considerable insight into the understanding of development artifacts. To project management, it becomes obvious whether agreement among stakeholders exists, what hinders any agreement, and what parts of the artifact need modification to increase the agreement in understanding.

We have shown the process of applying this method in a single domain using three iterations of a specific development artifact. Further research is necessary to extend this over the lifetime of an entire project and across multiple development artifacts. For example, a typical project might involve multiple UML diagrams, involve multiple prototypes, or user stories, etc. Realistic projects are also likely to involve more than three iterations of an artifact.

Finally, while we have shown the feasibility of this approach, future research will examine the impact on project success. The impact should be quantifiable in terms of development duration, reduction of re-work, or similar measures.

References

Bajaj, A. "The effect of the number of concepts on the readability of schemas: An empirical study with data models.," Requirements Engineering (9), 2004, pp. 261–270.

Beck, K. Extreme programming explained: embrace change. Addison-Wesley, Reading, MA: Addison-Wesley, 2000

Bodart, F., Patel, A., Sim, M., and Weber, R. "Should optional properties be used in conceptual modelling? A theory and three empirical tests," Information Systems Research (12:4), 2001, pp. 384–405.

Boehm, B. "Understanding and controlling software costs," IEEE Transactions on Software Engineering (14:10), 1988, pp. 1462–1477.

Burton-Jones, A. and Meso, P. "How good are these UML diagrams? An empirical test of the Wand and Weber good decomposition model," In L. Appleate, R. Galliers, and J. I. DeGross, editors, Proceedings of the Internataionl Conference on Information systems, Barcelona, Dec. 2002.

Clark, H. H., & Brennan, S. E. "Grounding in communication". In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), Perspectives on socially shared cognition (pp. 127--149). Washington, DC, USA: American Psychological Association, 1991

Evermann, J. and Wand, Y. "Ontological modelling rules for UML: An empirical assessment," The Journal of Computer Information Systems (46:5), 2006, pp. 14-29.

Fowler, M. and Kendall, S. UML Distilled : A Brief guide to the standard object oriented modelling language. Reading, MA: Addison-Wesley, 2000.

Gemino, A. Empirical Comparisons of Systems Analysis Modeling Techniques. Ph.D. thesis, University of British Columbia, Canada, 1999

Gemino, A. and Wand, Y. Comparing object oriented with structured analysis techniques in conceptual modeling. Draft Manuscript, 2001.

Gemino, A. and Wand, Y. "Evaluating modeling techniques based on models of learning," Communications of the ACM (46:10), 2003, pp. 79–84.

Gemino, A. and Wand, Y. "Complexity and clarity in conceptual modelling: Comparison of mandatory and optional properties," Data and Knowledge Engineering (55:3), 2005, pp. 301–326.

Hungerford, B. C., Hevner, A. R., and Collins, R. W. "Reviewing software diagrams: A cognitive study," IEEE Transactions on Software Engineering (30:2), 2004, pp. 82–96.

Hunt, J. The unified process for practitioners: object-oriented design, UML and Java. London:Springer Verlag, 2000.

Iscoe, N., Williams, G. B., and Arango, G. "Domain modeling for software engineering," Proceedings of the 13th International Conference on Software Engineering ICSE 91, Austin, TX, 1991, pp 340–343.

Jackson, M. "The world and the machine," Proceedings of the 17th International Conference on Software Engineering ICSE 95, Seattle, WA, 1995, pp. 283–292.

Kim, J., Hahn, J., and Hahn, H. "How do we understand a system with (so) many diagrams? Cognitive integration processes in diagrammatic reasoning," Information Systems Research (11:3), 2000, pp. 284–303.

Kung, C. and Solvberg, A.. "Activity modelling and behaviour modelling," In T. Olle, H. Sol, and A. Verrijn-Stuart, editors, Information System Design Methodologies: Improving the Practice. Amsterdam:Addison-Wesley, Amsterdam, 1986.

Lind, M. and Zmud, R. "The influence of a convergence in understanding between technology providers and users of information technology innovativeness," Organization Science (2:2), 1991, pp. 195–217.

Mayer, R. E. Models for understanding. Review of Educational Research (59:1), 1989, pp. 43–64.

Mylopoulos, J. (1992). "Conceptual modeling and Telos," In P. Locuopoulos and R. Zicari, editors, Conceptual Modeling, Databases and Cases. New York, NY:John Wiley & Sons, Inc, 1992.

Offen, R. "Domain understanding is the key to successful system development," Requirements Engineering (7), 2001, pp. 172–175.

Parsons, J. "Effects of local versus gloabl schema diagrams on verification and communication in conceptual data modeling.," Journal of Management Information Systems (19:3), 2002, pp. 155–183.

Peirce, C. S. Collected papers of Charles Sanders Peirce. Edited by Hartshorne, Charles and Weiss, Paul. Cambridge, MA:Harvard University Press, 1931

Topi, H. and Ramesh, V. "Human factors research on data modeling: A review of prior research, an extended framework and future research directions," Journal of Database Management (13:2), 2002, pp. 3–19.