Requirements Models in Context

Colin Potts
College of Computing, Georgia Institute of Technology
potts@cc.gatech.edu

Abstract

The field of requirements engineering emerges out of tradition of research and engineering practice that stresses the importance of generalizations and abstractions. Although abstraction is essential to design it also has its dark side. By abstracting away from the context of an investigation, the designer too easily lapses into modeling only those things that are easy to model. The subtleties, special cases, interpretations and concrete features of the context of use are smoothed over in the rush to capture the essence of the requirements. Often, however, what is left out is essential to understanding stakeholders' needs. In contrast, approaches that stress context at the expense of abstraction may lead to floundering or to short-term customer satisfaction at the expense of long-term fragility of the system. What is needed is a synthesis of these two approaches: a synthesis that recognizes the complementary values of abstraction and context in requirements engineering and that does not relegate either one to a background role. Such a synthesis requires us not only to adopt new methods in practice but also to rethink our underlying assumptions about what requirements models are models of and what it means to validate them.

Most requirements engineering (RE) research and practice embodies a philosophy that I call *abstractionism*, which involves the building of simplified models of domains of discourse and proposed systems. Abstractionists make much use of formal models, such as state-based or activity-based descriptions of workflow and system behavior, entity-relationship models of domain information, and goal dependency networks for businesses and systems.

All science and engineering depends, of course, on the discovery and use of appropriate abstractions. If we could not capture what was somehow essential to the functioning or structure of a system, we could not build it. Any complex system has to be described in advance at a suitably high level of abstraction, and the world into which the system will be introduced needs to be described similarly.

Discovering the appropriate abstractions is a two-level activity. First, we must decide on the ontology of the phenomena that we wish to describe. We always have choices in picking an ontology, although devotees of specific design methods and languages seldom decide to exercise this choice. For example, a description of a piece of the world or a system that rests heavily on the concepts of events, activities and temporal sequence is more appropriate for time-constrained systems, such as real-time controllers, than an ontology that had a complementary emphasis on objects, their properties, relationships, and constraints on these properties and relationships.

Having chosen an ontology, we can then use it to represent the domain of discourse. Suppose we have chosen an ontology based on information structure and relationships (as embedded in several database design methods, for example), we are then faced with the challenge of using this ontology to represent our system. Here, too, we are faced with choices. What do we include in the model, and what do we simplify away?

The key notion underlying abstractionism is the model, a sufficiently restricted description of a phenomenon that can stand in its stead and be used to answer questions about the phenomenon. But the restrictedness of a model is both its strength and its Achilles heel. A model is more malleable than the modeled phenomenon; it can be written down on paper and stored in a relatively small repository; it can be analyzed using automated tools. But malleability is not the same as compliance, and sometimes we simplify away many of the features of the world that we may need to know about.

It is against this backdrop that several RE research projects have been conducted in recent years to investigate the role of naturalistic inquiry [1] methods in RE [4][5]. Typical of this research program has been the use of ethnographic methods to help requirements engineers understand better the use situation prior to their construction of abstractionist models.

An alternative design philosophy to abstractionism is *contextualism*, according to which the particularities of the context of use of a system must be understood in detail before the requirements can be derived. This philosophy is especially sympathetic to naturalistic methods of inquiry, which contextualists use to uncover and help interpret these particularities. Unlike abstractionists, however, contextualists do not do this just as an initial basis for model building. In fact, in the social sciences that make most use of naturalistic methods, such

as anthropology, it is unusual to develop models from naturalistic data.

The differences between abstractionism and contextualism are summarized in Table 1.

A consideration of the complementary strengths and limitations of abstractionism and contextualism, which are given in Table 2, suggests that a synthesis of the two approaches would have great advantages. We would like to avoid the worst excesses of model-building and understand the context in which a system is to be used so well that it will truly fit its context rather than some simplified conception that fits our modeling ontology. On the other hand, we generally are not able to engage in laborious and time-consuming fieldwork, and our designs cannot progress reliably if a "rich description" is our only starting point.

Despite the apparent incompatibilities implied in these descriptions, I believe that the objectives of abstractionism and contextualism can be reconciled. If we cannot do this, we will continue to develop systems that, however well they are developed according to the canons of abstractionism, are less useful than their customers deserve, because they do not fit comfortably into their contexts of use.

However, to bring such a synthesis about means more than merely adopting some methods from the social sciences for data collection before the "real" work of RE starts. This weak, "uncommitted" view can lead to minor improvements in the modeling of requirements, because it entails the use of more appropriate methods for gathering background knowledge. A strong, "committed" view, however, requires that we acknowledge that most of the phenomena contained in models of systems and domains are socially constructed, and not objectively real [1]. "Business processes", for example, do not exist in businesses at all, but in key stakeholders' interpretations of the business. They are constitutive.

If we adopt this strong view, our models become simplified descriptions of the interpretations of reality offered by the stakeholders in the system, not representations of the real world. In some situations, this distinction is too fine to make a difference (for example, in modeling physical phenomena in real-time control systems, or some legislative requirements for an information system). Generally, however, the shift is significant. The fidelity of a model no longer is as much an issue as the degree to which it represents to the stakeholders' satisfaction and agreement the islands of stability that always arise in any sea of constitutive phenomena.

Are there examples of such models? Well, yes, there are, but they are not generally to be found in the literature of RE. An example of how a strong synthesis can be brought about would be to apply Spradley's [6] methods of taxonomic elicitation and domain analysis from the discipline of cognitive anthropology to the construction of object-oriented analysis (OOA) models. Spradley's discussion of the uses to which ethnographic interviews and participant observation can be put read uncannily like the literature of OOA with two significant differences: Spradley was writing ten years before OOA caught on, and his guidelines, unlike those of most OOA methods, are rich and strongly heuristic. What results from such an inquiry into social phenomena (Spradley's favorite examples are cocktail lounges and the street life of tramps)

Table 1: Comparison of Abstractionism and Contextualism

	Abstractionism	Contextualism
Role of description	Abstractions are powerful and general	Particularities are as informative as generalities
Design criteria	Design integrity	Contextual fit
Origin of requirements	Prescriptive recommendations	Current practice
Role of users	Management	End-users
Community of practice	RE and software engineering	CSCW and HCI

Table 2: Comparison of strengths and limitations of abstractionism and contextualism

	Abstractionism	Contextualism
Advantages	Generalization across contexts	Accommodation of richness of contexts
	Standard methods for constructing abstractions	
Limitations	Oversimplification	Descriptive, not prescriptive
	Overemphasis of normative cases	Overemphasis on immediate actors and current practice

are not models of these settings that are in any sense "correct", but rather analyses of how the participants in such social situations see their world. This is exactly how we need to look at the activity of modeling when we create models that help us develop systems to improve business processes.

Another example of a strong synthesis of contextualist and abstractionist thinking is illustrated by some work of mine with Idris Hsi [2]. We were interested in questioning the core assumption of goal-refinement approaches to RE: that a system can be said to embody a set of goals. The big contextualist questions that arise from such a claim are obvious: whose goals are we talking about? how do we gather them? what can happen to thwart these goals in a real context, and how should our system cope with the vagaries of real life? After interviewing administrative assistants about how they organize meetings and in analyzing a corpus of e-mail messages about meeting scheduling, we compared the results with the abstractions to be found in goal-refinement analyses of meeting scheduling [7]. We then developed a set of specific heuristics for doing goal refinement in conjunction with naturalistic inquiry techniques. These naturalistic techniques are not to be used initially to help develop insight about the problem that is subsequently and by some ineffable means transformed into a formal model. Nor are they used to develop models of how meeting scheduling actually happens. Rather, they are used to develop an understanding of how stakeholders interpret the process in which they are engaged and how that interpretation should shape the formal refinement process as it actually unfolds.

The strong view of synthesis leads us to ontological and epistemological commitments (commitments about what our models are models of and what it means to validate them) that conflict with the conventional use of models in RE. To validate an "as-is" model is not to test the correspondence of a model against reality but to increase one's confidence in the trustworthiness of the model [1]. Validating a "to-be" model is not a process of checking whether the model describes what the customer really wants, as if that were a stable phenomenon, but to gain or predict the approval of stakeholders from demonstrations of functionality.

References

[1] Lincoln, Y. and E. Guba, *Naturalistic Inquiry*. Sage Publications, 1985.

[2] Potts, C. and I. Hsi, Abstraction and Context in Requirements Engineering: Toward a Synthesis, *Annals of Software Engineering*, to appear, 1997.

[3] Potts, C. and W. Newstetter, Naturalistic Inquiry and Requirements Engineering: Reconciling Their Theoretical Foundations, *Proceedings ISRE'97: Third International Symposium on Requirements Engineering*. Annapolis, MD: January 6-8, 1997. IEEE Computer Society Press.

[4] Randall, D., J. Hughes and D. Shapiro, Steps toward a Partnership: Ethnography and System Design, in M. Jirotka and J. Goguen (Eds.) *Requirements Engineering: Social and Technical Issues*, Academic Press, 1994.

[5] Sommerville, I., T. Rodden, P. Sawyer, R. Bentley and M. Twidale, Integrating Ethnography into the Requirements Engineering Process. *Proceedings RE'93: International Symposium on Requirements Engineering*. San Diego, CA: January, 1993. IEEE Computer Society Press

[6] Spradley, J. *Participant Observation*. Holt Rinehart Winston, 1980.

[7] Van Lamsweerde, A. Darimont, R. and Massonet, P. Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt, *Proc. RE'95: 2nd International Symposium on Requirements Engineering*, IEEE Computer Society Press, 27-29 March, 1995, York, UK.