Collaborative Modeling: Hiding UML and Promoting Data Examples in NEMo

Patricia Schank
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
+1 650-859-3934
patricia.schank@sri.com

Lawrence Hamel
Codeguild, Inc.
401 Claremont Way
Menlo Park, CA 94025
+1 206-202-1825
cscw@codeguild.com

ABSTRACT
Domain experts are essential for successful software development, but these experts may not recognize their ideas when abstracted into Unified Modeling Language (UML) or ontologies. We describe a Web-based tool for modeling that creates and manipulates a simple data model without representing it in UML, while promoting collaboration and the use of examples to compare and validate the model. The open-source tool, "NEMo," is a by-product of a team effort to invent and refine a complex data model and library of examples.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/ Specifications – *elicitation methods, methodologies, tools.* D.2.10 [**Software Engineering**]: Design – *methodologies, representation.* H.1.2 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *computer-supported cooperative work.* I.6.5 [**Simulation and Modeling**]: Model Development – *modeling methodologies.*

General Terms
Design, Human Factors.

Keywords
Domain modeling, data modeling, participatory design, knowledge extraction, example-driven design, UML, XML.

1. INTRODUCTION
A recent NSF workshop identified several challenges and opportunities for research programs to advance the science of design of software systems [12], including the need to improve the ability of stakeholders to creatively explore designs, not in the abstract, but through participatory, contextualized processes. This paper describes one approach to support the design of a particular information artifact: the data model.

Many communities of practice have undertaken the creation of XML data models in an effort to standardize digital communication in their fields. The collaborative design of these data models is often the primary activity of such groups, and their published models can have a significant impact. For example, HL7 (hl7.org) is widely used in medical informatics, QTI (imsglobal.org) is popular in learning-management systems, and FIX (fixprotocol.org) is used in securities exchange. More generally, saving information is fundamental to most software systems, and for a system to be useful and adaptable, it must have a sound data model—as accurate and complete as possible, yet flexible enough to accommodate inevitable changes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CSCW'04, November 6–10, 2004, Chicago, IL, U.S.A.
Copyright 2004 ACM 1-58113-810-5/04/0011…$5.00.

1.1 Challenges for Data Modeling
Within workgroups of domain experts, the contributions of individuals are mediated by tools and representations, and perhaps by technical personnel who may exclusively edit the representations. But experts can be sidelined by poor tools and processes. New techniques are needed to increase participation of domain experts in data modeling to support creative, interdisciplinary, collaborative exploration to promote better designs [12] and to enable domain experts to validate that their domain is represented correctly by a candidate design. Participatory design—in which stakeholders serve a proactive, central role in the design team, working together with engineers on a design—can often lead to more usable designs and shortened development and test cycles [11].

The effort to define a data model collaboratively raises issues similar to those well documented in the fields of knowledge engineering and knowledge management: the difficulties of knowledge extraction and capture of social context. In artificial intelligence research, the development of expert systems relies on extracting knowledge from experts and representing that knowledge in the system. In practice, it is quite difficult and time-consuming for experts to articulate their (often tacit) knowledge and skills, removed from the context of an activity [7]. Designers of knowledge management systems experience similar issues when they try to codify employees' situated knowledge within a company knowledge base [2].

On the basis of the research literature and our own experiences (described below), we organize the challenges facing the practice of data modeling into the following categories:

- *Limited participation*. Typically, a software engineer mediates all contributions. To understand or apply the model, participants need a fair amount of technical expertise.
- *Insufficient negotiation*. Barriers to debate include inability to edit the model, lack of understanding of the model, and limited sense of ownership.
- *Insufficient validation*. Discussion without concrete examples can lead to ambiguity and misunderstanding.

- *Misunderstanding across disciplines*. With abstract representations, terms can be misinterpreted or have different definitions in different disciplines.
- *Low-level conversations*. Discussions can be sidelined by incidental issues, such as (often complex) model notation.

1.2 Changing the Practice of Data Modeling

As part of the NSF-funded Principled Assessment Designs for Inquiry (PADI) project (padi.sri.com) described below, we created a system, which we now call NEMo (Negotiated Example-based Modeling) to support the creation of a data model for the domain of student assessment. NEMo supports an evolving data model and enables a participatory process, including the generation of examples by a large and varied group of domain experts. It has helped us capture ideas of both domain experts and technical staff. We noticed the following changes in the practice of data modeling:

- *Broader participation*. New people became involved in the conversations and had the understanding and motivation to critique proposals. Most team members created examples.
- *More negotiation*. Model proposals were thoroughly debated through much iteration.
- *Increased validation*. Discussions of abstract models were grounded in concrete examples, and proposals were communicated fairly quickly between team members.
- *Improved understanding across disciplines*. Understanding was less clearly split down disciplinary lines. There was less need for one camp to "translate" ideas for the other.
- *Elevated conversations*. Discussions focused on important domain issues, such as the kinds of information that were important to capture and how to build flexibility into the model to accommodate future changes.

Our hypothesis is that NEMo changed the practice of data modeling in PADI because of the following key features:

- Examples led to broader participation, increased validation, and understanding across disciplines.
- The familiar Web interface led to broader participation and more negotiation.
- Remote access via the Web led to more negotiation.
- Lack of database or object-oriented data modeling verbiage (e.g., relations, entities) led to broader participation and elevated conversations.
- The ability to switch quickly between competing models led to more negotiation and elevated conversations.

We also suspect that the process for using NEMo may have been just as important as the system itself. The PADI team teleconferenced weekly, met face-to-face annually, and had weekly e-mail conversations about our model. We broke up into small groups that focused on developing the model or applying it to their concrete situation. A variety of experts (including psychometricians, curriculum developers, education researchers, and software architects) facilitated the conversations by coming to meetings with candidate models and examples to critique. Participants were also prompted by e-mail to log in to the system and comment on proposed models and examples.

2. RELATED WORK

The Unified Modeling Language (UML) is the de facto standard for object-oriented data modeling [1]. However, UML takes considerable time to learn, and is difficult to use for communication with nontechnical members of a project [9, 13]. Most UML tools are intended for an audience of software engineers who will translate the UML into programming code, and most UML tools do not offer support for entering sample data to validate a data model. To share UML, the engineer editing a UML diagram might export it as an image file and transfer it to a Web server for others to critique. For instance, IBM's Rational Rose supports the creation and publishing of UML diagrams to the Web. For a teleconference, some kind of screen-sharing technology like Microsoft's NetMeeting might be used to share the application. Poseidon (gentleware.com) allows the sharing of UML models in a client-server configuration, as long as each concurrent user has a license. Yet other tools interpret pen strokes to create UML elements, combining the ease of freehand drawing with the benefits of computer editing and saving [5, 9]. However, these scenarios do not facilitate control of the model (or competing models) by nontechnical experts, and there is no clear support for creating concrete instances of the models within these tools.

In the field of artificial intelligence, ontological engineering tools have been created to help developers and domain experts build effective knowledge-based systems. Some tools attempt to support communities of nontechnical domain experts [6, 8], but few support the entering of examples to validate the model design. An exception is Protégé [8], but even so, these sample records are subordinated to the role of design annotations, with functionality for manipulating examples buried among many other features. To share models across a distributed team, users must perform additional configuration, since Protégé is a desktop-, not Web-based, application.

In contrast, NEMo is specially oriented around examples as the primary device for communication and negotiation. The research literature indicates the benefits of such contextualization. Research in learning has shown that focusing on examples, rather than abstract representations, can both enhance and accelerate comprehension and learning [e.g., 4] due to issues related to working memory capacity and motivation. In software design, the use of examples can help designers recognize, capture, and reuse generalizations, and ultimately enhance the effectiveness of the products of design [3]. We argue that in data modeling, examples not only validate a candidate model, they also support the creative interplay among domain and technical experts, improving communication and understanding of requirements and stakeholders' practice and ultimately improving the usefulness and accuracy of the final model.

3. REQUIREMENTS AND DESIGN

NEMo emerged from the requirements and practices the PADI project, which attempts to model the psychometrics of the assessment of inquiry learning by science students [10]. A primary deliverable of the project is a robust data model for assessment design, delivery, and scoring; a secondary deliverable is a library of exemplars. The PADI team includes nearly 30 members from five educational organizations: SRI International, University of Maryland, University of California at Berkeley, Lawrence Hall of Science, and University of Michigan. The vast majority of the team members are domain experts in science education, educational research, assessment, and instruction. The domain experts were not familiar with UML, with ontology construction, or with XML schemas.

As the team teleconferenced weekly to discuss models of student understanding, they needed a way to manipulate and compare multiple models and examples in a shared manner.

One of our first tasks was to explore promising data modeling tools and methods that would support our collaborative efforts to develop a model and library of exemplars. We investigated UML tools Rational Rose and Poseidon, as well as ontological engineering tools such as Protégé. We also experimented with a typical knowledge extraction method in which a senior engineer presented UML diagrams created in Rational Rose and solicited critiques from the team. We found that group members had difficulty understanding the UML notation and needed an easier way to generate and compare models.

The team members charged with software development sought to promote the creation of samples for validation and discussion. This approach was also attractive in that it would yield examples for the eventual PADI library. A Web application, dubbed "NEMo," was created to expose a virtual data layer, a layer of abstraction between the model displayed and the real database structure. PADI domain experts used this virtual layer to manipulate the model—the data object definitions and the interrelations between objects—in addition to entering examples. NEMo focuses on the manipulation of examples to validate the design under discussion and provide a more accessible means to understand the design. A Web-based solution was implemented to leverage team members' familiarity with Web forms, navigating Web pages, and refreshing a Web page to view new information in the shared repository. Using NEMo editing features, changes to the model could happen during a teleconference, and all members would see their samples change to reflect a new design (e.g., a new, empty field showing up in their sample records). The system also allows the creation of competing models, along with their own sample records. Using the system over 18 months, the PADI team created a data model that encompasses 15 core objects, with numerous attributes and relations between these objects, as well as sample instantiations of their use. As of April 2004, 19 team members had entered 422 examples into the system.

3.1 NEMo Implementation and Features

Figure 1 shows a sample NEMo editing screen to create a Movie object with nine attributes. From this screen, team members can reorder, insert, and delete attributes for an object (see below for discussion of shared vs. owned attributes). A team member could create an alternative "Movie" model with different attributes and thereby design on a parallel track. Figure 2 shows a populated instance—a sample record in which data have been entered for a particular movie.

NEMo is a three-tier application based on the free, open-source Expresso framework (jcorporate.com), which in turn is based on Apache Struts and uses Java Server Pages technology for the rendering layer. It employs a simple node-attribute-relation database schema, which allows all instances to be handled by the same functions; for example, the code for manipulating a Movie instance is the same as that for editing a Person instance. Expresso offers a model-view-controller system in which the universe of supported Web requests is mapped into a finite-state machine. Expresso includes an object-to-relational layer that permits developers to write persistence code in Java only, without explicit SQL. In practice, this means that our development commonly takes place on desktop computers running Windows or Mac OS X, using the Hypersonic SQL database, followed by seamless deployment on Solaris servers using the MySQL database.

Figure 1. A NEMo page allows manipulation of the attributes of a "Movie" object.

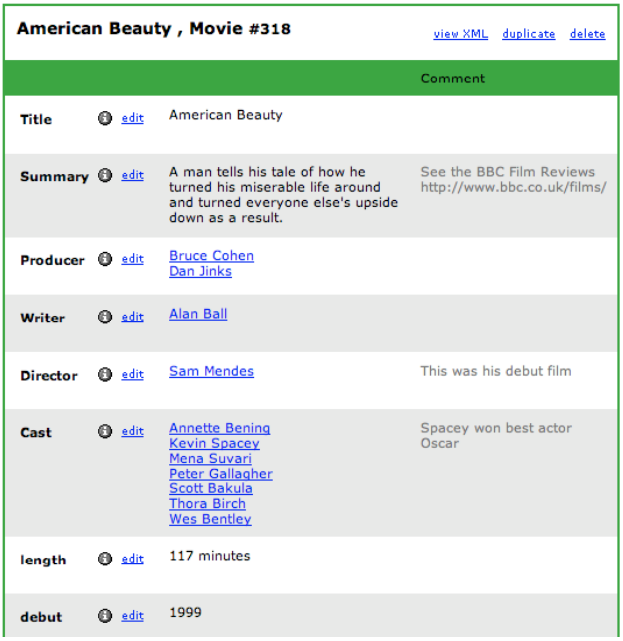

Figure 2. A Movie instance in NEMo provides fields for the input of sample data.

NEMo's core features include:

- *Shared editing and viewing.* As a Web application with server-side data persistence, NEMo offers the potential to share all its contents (given sufficient permissions), including editing rights for the current model and/or the ability to create a competing model. As model edits are made and stored in the database, all subsequent views reflect the updates, providing immediate feedback to participants.

- *Multiple representations*. NEMo represents both models and examples as a set of Web pages. It currently supports import/export via XML, and we propose to extend this to include XML Metadata Interchange and DTD, as well as to implement other visual (tree and graph) representations.
- *Shared-relationship attributes vs. owned attributes*. When analyzing a domain, one of the challenges is to identify first-class objects and relations between them ("shared" attributes in NEMo), as opposed to identifying attributes that are not separable from the object ("owned" attributes). Shared attributes are created by relating two objects via checkbox associations, where the possible candidates are constrained by the model. Owned attributes are simply edited and viewed as values within an object. Whether an attribute is shared or owned is debatable; it depends on how experts view the domain and plan to use the data, and may change as experts compare and contrast modeling decisions.
- *Permissions can be specified per instance (per row)*. Groups can be given fine-grained permissions that control reading and writing on a row-by-row basis, where a "row" in the database corresponds to a model instance in NEMo. Further, a distinction is made between users who have permission to edit examples and those who can edit the model itself.
- *Menus can constrain attribute values*. Objects may contain attributes that have constrained values (vs. free-form text entry), and these values will be presented in a menu. Users with model-editing privileges can dynamically alter the constraints (the menu items) by editing the model. In response, the menus available to model instances will change.
- *Special handling is possible via extensions*. Attributes that require custom rendering or special handling for viewing and/or editing can implement a Java interface for that purpose. This feature provides for the creation of arbitrary view/edit screens, custom built for a given attribute.

4. CONCLUSIONS AND FUTURE WORK

As various disciplines attempt to standardize the exchange of information via XML, modeling teams may benefit from a tool that supports collaborative data modeling. NEMo is a by-product of an effort by domain experts to collaborate in drafting a coherent data model. It promotes the use of examples, avoiding UML representations and leveraging the experience of team members with Web forms and online information sharing. NEMo proved sufficiently malleable within a project attempting to model the psychometrics of assessment design, and continues to serve as a repository of examples. We are actively developing additional editing capabilities and navigational aids while expanding the library of examples in the system. The model has matured and the model-editing functions are gathering dust, but the system now provides a library of resources as we discuss content and content-creation "wizards" to scaffold interaction with the system.

Our future work will seek to determine whether our modeling success is replicable across other domains and to more systematically contrast tools and processes for designing data models. Since NEMo is available under an open-source license (sourceforge.net/projects/emo/), we hope others will adopt and adapt it for their own uses and share their feedback with us.

5. ACKNOWLEDGMENTS

We thank Geneva Haertel, Robert Mislevy, John Gennari, Chris Digiano and the PADI team for their helpful suggestions for NEMo and this paper. This work was supported by Interagency Educational Research Initiative grant REC-0129331.

6. REFERENCES

[1] Booch, G. Rumbaugh, J., and Jacobson, L. *The Unified Modeling Language user guide*. Addison-Wesley, Reading, MA, 1998.

[2] Brown, J. S., & Duguid, P. *The social life of information.* Harvard Business School Press, Cambridge, MA, 2000.

[3] Carroll, J. M., & Rosson, M. B. Getting around the task-artifact cycle: How to make claims and design by scenario. *ACM Transactions on Information Systems, 10*(2) (1992). 181-212.

[4] Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., and Glaser, R. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13* (1989), 145-182.

[5] Damm, C. H., Hansen, K. M., and Thomsen, M. Tool support for cooperative object-oriented design: Gesture based modeling on an electronic whiteboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands, April 1-6, 2000). ACM Press, New York, 2000, 518-525.

[6] Domingue, J., Motta, E., Shum, S. B., Vargas-Vera, M., Kalfoglou, Y., and Farnes, N. Supporting ontology driven document enrichment within communities of practice. In *Proceedings of the First International Conference on Knowledge Capture* (Victoria, British Columbia, October 21-23, 2001). ACM Press, New York, 2001, 30-37.

[7] Dreyfus, H. *What computers still can't do: A critique of artificial reason. MIT Press,* Cambridge, MA, 1993.

[8] Gennari, J. H., Musen, R. W., Fergerson, W. E., Grosso, M. C., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, *58*, 1 (Jan. 2003), 89-123.

[9] Hammond, T., and Davis, R. Tahuti: A geometrical sketch recognition system for UML class diagram. In *Proceedings of the AAAI Spring Symposium on Sketch Understanding* (Palo Alto, CA, March 2002). AAAI Press, Menlo Park, 2002, 59-66.

[10] Mislevy, R., Haertel, G., and the PADI Research Group. *Design Patterns for Assessing Science Inquiry*. Technical Report PADI-1, SRI International, Menlo Park, CA, 2003.

[11] Schuler, D., and Namioka, A. *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

[12] Sullivan, K. *Preliminary report of the NSF Workshop on the Science of Design*. Department of Computer Science, University of Virginia, Charlottesville, VA, 2004. http://www.cs.virginia.edu/~sullivan/sdsis/

[13] Tilley, S., and Huang, S. A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding. In *Proceedings of the 21st Annual International Conference on Documentation* (San Francisco, CA, October 12-15, 2003). ACM Press, New York, 2003, 184-19