Object-Oriented practice in 1998: Does it help or hinder collaboration?

Panel Discussion

John Artim (Moderator), Charlie Bridgeford, Lillian Christman, James Coplien, Mary Beth Rosson, Stanley Taylor and Rebecca Wirfs-Brock

OOCL (USA), Inc., Employer's Reinsurance Corporation, OOCL (USA), Inc., Bell Laboratories, Virginia Polytechnic Institute and State University, Apple Computer, Wirfs-Brock Associates

ABSTRACT

Groups of people working in concert perform most commercial, industrial or in-house software development. These groups are often quite diverse. This panel brings together object-oriented consultants and developers as well as practitioners and researchers interested in human factors and user-centered design, project management and technical writing. The panel will address the question, "To successfully complete today's projects, does object-oriented development as practiced today do an adequate job of supporting ALL of the participants who must collaborate?"

Keywords

Collaboration, object models, development practice

PANELIST POSITIONS

Lillian Christman

When we work together in groups we always (1) divide up the work and (2) develop some mechanism for coordinating across this division of work. As an innovative activity, software development groups are best supported by an organizational structure dubbed adhocracy by organizational theorists. The division of labor in an adhocracy is highly specialized. Coordination is primarily achieved through mutual adjustment. "Under mutual adjustment, control of the work rests in the hands of the doers. The knowledge of what is needed develops as the work unfolds." (1) Project success depends on the ability of the specialists to adapt their work to each other.

The tug-and-pull between labor specialization and the need to dynamically adjust to one another's work raises issues for those in software development. Chief among these issues are (1) a low tolerance for ambiguity and (2) extremely high communication costs. Is it possible that an OO approach can help address these issues?

Ambiguity

Being primarily concerned with software usability, I've noted the often ambiguous reference to "the" user. Is this person the domain expert, the task expert, or the company executive who sponsors the project? Generally we stuff all of these people and the information they give us into the same user cubbyhole. What can we do to be more precise in our use of user information? At the project level: clearly distinguish between team members who are domain/business process experts (business users) and those who are task experts (end-users); align these users with the appropriate development staff. At the artifact level: in use cases, clearly distinguish task-specific (end-user) requirements from business process (business user) requirements; explicitly model the end-user as well as the domain; in addition to business scenarios, develop task scenarios to support end-user requirements. At the process level: conduct domain requirements reviews with business users; conduct usability reviews and tests with end-users.

Communication

Many of the problems that need to be solved during the development of a software system only become apparent as the work unfolds. It can take a great deal of time for project team members to talk through the issues raised and reach decisions. As a consequence, a software development unit can appear to be highly inefficient. Again, as someone concerned primarily with usability I would make a few suggestions for streamlining communication. At the project level: include both business and end-users as full-time members of the team. At the process level: provide an opportunity for all team members to observe end-users in their work context; disseminate task requirements and scenarios to domain modelers, domain developers, database designers and quality assurance team members in addition to the UI designers/developers. Mutual adjustment is achieved primarily through informal communication. Nurture these channels as enthusiastically as you do the formal development process.

It's easy to see how the pressures to reduce ambiguity and to make communication more efficient can push a software

development group into standardizing as much of the work as possible. These pressures are especially intense at the beginning of a project. But be careful of imposing standards before the work has had some time to evolve. Standardization tends to push the organization in the direction of bureaucracy, an organizational structure not particularly supportive of innovative work.

At conferences like OOPSLA we spend a lot of time discussing the merits of various technical approaches to our work. I think it's useful to take some time to understand how we can use an understanding of organizational behavior to support our work.

[see Mintzberg, 1979]

Biographic Sketch

Lillian Christman is Manager of User-Centered Design for OOCL, a global shipping company. For the last seven years she has lead the user-centered design activities of large business information and niche market software development teams. A devoted observer of organizational behavior, she has a PhD in industrial sociology from Vanderbilt University.

James Coplien

It takes several paradigms in concert to meet a typical customer need. Many customers need databases, and few of those need to be object-oriented in any way. The term "object-oriented user interface" is thrown around a lot, but really has little or nothing to do with objects, and probably shouldn't have if it did. Those who limit themselves to objects are limiting their ability to communicate needs and to implement effectively.

To a first order, even the design paradigm doesn't matter. The primary issues of software development are organizational, logistic, and social. Many have to do with communication flow. There are techniques like CRC cards that have come from the object milieu, and those help. Refactoring has come into its own under object-oriented banners. But these have little to do with objects, are not new, but are just widely ignored. It's not their object-ness that makes them useful: JAD or SCRUM would be just as good.

To a first order, good communication comes from good organization, including good staffing, proper sizing, and strong role identification. If we are to seek improvements in software development, it's time to start looking away from the languages and methods associated with objects and to start looking at the social and organizational factors. Doing that is a matter of will, as is much of the success of our discipline. It takes will to not be a sheep looking to methods, objects, or process to get them through. It's about simple basics.

Biographic Sketch

Jim Coplien is a Distinguished Member of the Software Production Research Department in Bell Laboratories. He is currently studying organization communication patterns to help guide process evolution, as well as multi-paradigm design and architectural patterns of telecommunication software. He is author or co-editor of several books on C++, objects and software patterns. When he grows up, he wants to be an anthropologist.

Mary Beth Rosson

As a researcher in human-computer interaction (HCI), a key interest for me is how object technology can contribute to the development of useful and usable systems. Our work on HCI design methods relies centrally on scenarios of use (Carroll, 1995), and object-oriented analysis and design fit well into scenario-based development methods (Rosson & Carroll, 1995). We have used object analyses of proposed usage scenarios (i.e., a set of objects that could collaborate to implement a concrete scenario) to convey initial design specifications to programmers and to discuss and document the rationale for specific design decisions; we have used similar scenario-based analyses to explore proposed designs with potential users.

Recently we have begun to explore the role of object-oriented analysis in participatory design settings. Participatory design is founded on direct involvement of prospective users in the analysis and design process; participatory engagements may involve shared observation and analysis of existing work processes, brainstorming sessions about new ways to apply technology, or more specific design sessions in which the details of proposed human-computer interactions are refined. An important goal of such methods is mutual education-software experts learn more about the users and their tasks, and users learn more about current technology. With respect to the latter, we are investigating the effectiveness of an object-oriented conception of tasks (e.g., computational entities with specific responsibilities and collaborations) in extending users' views of their current goals and activities. Although end-users are initially skeptical about their abilities to reason about such models, we have found that once we help them identify and discuss a few objects they are able to extend these analyses in interesting and useful ways.

Biographic Sketch

Mary Beth Rosson is an Associate Professor of Computer Science at Virginia Polytechnic Institute and State University, where she has been since January 1994. Prior to that time, she was a Research Staff Member and Manager at the IBM T. J. Watson Research Center. She received a PhD in Experimental Psychology in 1982 from the University of Texas at Austin. She has been very active in both SIGCHI and SIGPLAN, serving in numerous Technical Program roles for the CHI and OOPSLA annual conferences, and as a member-at-large on the executive

committee of SIGPLAN. She is on the editorial board of Interacting with Computers. Dr. Rosson's research interests include the development of new paradigms for research in human-computer interaction, the use of network technology to support collaboration, especially in learning contexts, and psychological issues in the learning and use of the object-oriented design paradigm. She is author of Instructor's Guide to Object-Oriented Analysis and Design with Applications, has developed and taught a number of professional short courses in HCI design methods and in object-oriented design, and has authored numerous journal articles, conference papers, and book chapters.

Rebecca Wirfs-Brock

Object developers face two challenges: correctly interpreting stakeholders' concerns and requirements in their designs, and presenting their analysis and design work in terms understood by a wide audience. Consider if we insisted on teaching our stakeholders basic object-oriented terms and forced them to only speak "objectese". Pidgin-objects, would by necessity be a simplified language. Like all pidgin languages, object-pidgin could only expresses basic ideas. Some stakeholders' complex usability requirements, specific needs and concerns simply could not be expressed. How absurdly object-centric this view is!

You may think I'm going overboard. Of course *you* don't insist on objects being the center of the universe! However, I have encountered teams where CRC that cards were touted as the being the only necessary bridge to programming. If users were taught to model classes, responsibilities and collaborations, then the programmers' jobs sure would be easier. All they'd have to do would be to "translate" and "fix up" objects to make their software work. It wasn't clear whether it was a priority that the software function as the users wanted. I've been in meetings where object technologists (we had an affectionate name for them at my former company—"propeller heads") would discount other stakeholders as being stupid, clueless, and obviously misinformed. Since they didn't know objects, and couldn't understand our object technology deeply, they obviously didn't know anything.

While an object-oriented slant is useful, it is not complete. I happen to believe that expressing software designs with objects is very effective. However, each stakeholder in our object development process has differing needs and values. A lot of them know more about what they want their application to do than they can easily convey to us in object terms. With objects, we can capture high-level responsibilities of important concepts/aspects embodied by our software. Additionally, use case descriptions and scenarios can describe how our software should respond to its environment. Yet, both these models leaves unspoken the needs, intentions and day to day concerns of our system's users, production support staff, database administrators, network administrators, user interface designers, etc.

We must recognize that descriptions of what we are to build can be much more expressive and encompassing. I don't advocate that every project should rush out to generate lots and lots of new models. Instead, we need to gather and interpret appropriate descriptions in the native *languages* spoken by our stakeholders. Our job as object technologists is to reflect these descriptions in our object analysis and design work. Object technology is only one small part of a development context. We'll support collaborative work better as we become comfortable integrating our work into a process that adopts a variety of descriptions and models.

Biographic Sketch

Rebecca Wirfs-Brock is president of Wirfs-Brock Associates, a firm specializing in the transfer of object analysis and design expertise to organizations and individuals through training, mentoring, and consulting. Rebecca has been involved in object technology since 1984. Rebecca once spent several weeks with a client on a task force that attempted to integrate the techniques and practices of several analysis areas including workflow modeling, data modeling, human factors engineering, and object analysis. This experience revealed that each discipline uses similar names for very different concepts, and that even getting eager, attentive experts to understand each other can be difficult. Rebecca is co-inventor of the Responsibility-Driven Design method and co-author of the classic, *"Designing Object-Oriented Software."* Rebecca's is currently writing a new book on object design with her colleague, Alan McKean.

REFERENCES

1. Carroll, J. M. 1995. The Scenario Perspective on System Development. In (J. M. Carroll, Ed.), Scenario-Based Design: Envisioning Work and Technology in System Development (pp. 1-17). New York: John Wiley & Sons.
2. Mintzberg, Henry, *The Structuring of Organizations,* Prentice-Hall, Englewood Cliffs, NJ, 1979.
3. Rosson, M. B. & Carroll, J. M. 1995. Narrowing the Specification-Implementation Gap in Scenario-Based Design. In (J. M. Carroll, Ed.),Scenario-Based Design: Envisioning Work and Technology in System Development (pp. 247-278). New York: John Wiley & Sons.