Object Oriented Project Management

Moderator:
Laura Hill, JP Morgan, hill_laura@jpmorgan.com

Panelists:
Doug Johnson, Rothwell International, doug@rwi.com
Kenny Rubin, IBM
Charles Berman, Fidelity Investments
Jim Coplien, AT&T
John Daniels, Object Designers Ltd, jdaniels@cix.compulink.co.uk

1. Introduction

The purpose of this panel was to address issues of project management specific to object-oriented projects. Topics considered included:

- How is object-oriented Project Management different? Is a "paradigm shift" required for project management?
- What aspects of object-oriented project management should be stressed? What are some tips, tricks or gotchas experienced by the panel members?
- Does a project manager need to be more technical to lead an object-oriented project? Should s/he have had previous experience with object-oriented development?
- What are some alternate organizational structures which would better support object-oriented projects?

2. Presentations

Doug Johnson

Doug Johnson talked about variations of OO project management techniques due to project size. He found that they varied radically — big projects in particular requiring more paper, formality and process. The major mistake he reported was the application of small project management techniques to large projects.

He said that OO projects were essentially no different than any other kind of project. All projects need written requirements, design, process, written schedule, standards, a good object model, and reviews of all the deliverables.

For small projects (less than 5 developers), Doug reported that iterative development works very well. Each iteration has to have written requirements, written goals and a written schedule. Code management can be done without tools.

For medium sized projects (5-20 developers), Doug suggested starting with a small (1 - 5 developers) team to stabilize requirements and develop the core object model. This core team can then go on to become the system architect and team leaders for the expanded team. This allows the project to get running early and then grow the function base system. Code management tools are definitely required with this size team.

For large (greater than 20 developers) projects, Doug felt that a modified waterfall was required. Again, he suggested starting with a small team to get a very solid design before full staffing. Changing the design causes problems when communications and deep understanding of the change need to go out to a large number of people. The problems of large projects focus on communication. The system and staff must be organized into "meta objects" — subsystems of class groups with well

defined interfaces. Each subsystem is owned by a team of 2-5, with individuals owning classes. A lead architect is required to maintain system integrity. In a large project, the lead architect might have a team of two to three people. The lead architect role is as important as the project manager role, and should, in fact, report to the same person. The key to organizing system and staff is to think about them at the same time - Doug reported that he often found he had to compromise technical issues to meet people issues, which are more expensive.

Kenny Rubin

Kenny concentrated his presentation on discussing the many kinds of project that need to be addressed by an organization in order successfully to transition to object technology. He introduced the idea of a decision framework — much as a software framework is a collection of abstractions that form the basic concepts of a particular domain, a decision framework is a collection of the core ideas necessary to address a particular topic area, such as training, reuse or a development process model.

Kenny defined a project as a set of activities that move you from one place to another on the way to achieving a desirable state. Any given organization will have a number of such projects. Here are some of the projects you'll want to consider:

- Selecting the product process mode
- Selecting a reuse process model
- Identifying how you will plan and control a project
- Defining the team structure
- Selecting the software development environment
- Transferring skills to staff members
- Establishing a formal process of software measurement

More details on these decision frameworks may be obtained from the book Kenny co-authoured with Adele Goldberg, "Succeeding With Objects: Decision Frameworks for Project Management"

Charles Berman

Charles discussed the technology choices made at Fidelity in light of the business constraints and technology vision they had. He focussed on the components, frameworks and patterns they instantiated and the factors that allowed them to achieve reuse. Lastly, he discussed lessons learned.

The main business driver for Fidelity is the constantly changing nature of the business. This implies that applications, which directly relate to business entities and processes, also change rapidly. To handle this, they center their software development around small teams (3 - 6 developers) and short, 3 month, delivery cycles. Technology-wise Fidelity uses a three tier architecture with a componentized object model in the middle tier and a CORBA compliant backbone as the distributed environment. The front end (OLE on the desktop and the world wide web) is an external requirement.

The desire for reuse and a requirement for small teams that can act quickly suggested a strategy of binary executables as the unit of reuse. This strategy contained very little coupling between components - decoupling being the number one productivity indicator.

Rather than struggle with a large, centralized object model group, Fidelity has focussed on design reuse across the firm with a conceptual business object model and the use of design patterns. The use of patterns have proven to be a great way of explaining how behavior can be added to the conceptual model. Within the team, reuse may be obtained via old fashioned inheritance and software frameworks.

Charles identified four major factors for achieving reuse:

- Buy rather than build

- Reuse the legacy
- Make the code easy to acquire
- Trivialize the cost of component integration

Jim Coplien

Jim's premise was that there is no significant difference between OO projects and standard software development projects. The trappings of OO projects, such as iterative development, really have nothing to do with objects. He has found that all successful development projects adhere to the same organizational patterns. His own analysis orientation is to look at the organizational architecture — the things in systems and the relationship between the things.

Iterative development is special - but not specific to OO. It's something that is often hidden, but is actually how developers work. Jim pointed out that our old models for software development and practices come from the industrial revolution or from the military and that it is time to get new models, which more accurately reflect software development practices.

Realistically, most projects require multiple paradigms and it is the grouping that needs to be managed. The bulk of project management is risk management, and the biggest current risk in the OO world is tools. However, Jim feels that as OO technology matures, the need for explicit OO project management (ie risk management) will diminish.

John Daniels

John spoke of his experience being called in to help projects "in trouble", commenting that the trouble was nearly always to do with the way the project was organized, managed and run, rather than to do with any particular technology. All issues are about managing risk — identifying risk and then controlling it. The following four quotations are some of the warning signals you might look for, and John's answer to them:

" I always use incremental delivery. We're going to design the object model this cycle, and then we'll implement the business object classes in the next"
Evolutionary delivery is the best way of managing, controlling and reducing risk, but the above isn't evolutionary delivery. Each cycle must actually deliver something — some executable code that you can test and that works. The above is just a waterfall.

"I don't know much about objects, so I've told them to use OMT — after all, it is the leading method, isn't it?"
Every project manager must understand the technology being used on their project. Although a knowledge of design techniques is very helpful, blindly following a particular method can become a crutch and will be of no help. Risk management means identifying the risks and addressing them, not putting them off and avoiding.

" Now we're happy with the business model, we'll start thinking about the software architecture" (or "We don't need to try out the database until later")
It's never too soon to start work on the technical/software architecture, for it is in those details that the project will succeed or fail. It's imperative that you try out all your technology pieces as soon as possible.

" My chief architect wants to know how you indicate 'pure virtual protected function' on an OMT diagram"
The point of modeling is to produce abstract models. It's no use writing the code in pictures . All too often people are producing models at way too low a level. The result is that all too frequently the high level design isn't written down at all meaning that there is no easy route for newcomers to understand how it works.

3. Questions and Answers

Question: What has been your experience with decentralized development?
Doug: Decentralized development requires lots of communication. My rule of thumb is to establish as high a communication bandwidth as possible and use as little of it as necessary. Need a lot of personal commuting.
Jim: It's important to look beyond surface reasons to underlying causes. For example, look at conference calls across the Atlantic -

- the energy level and motivation differ between the early morning of one group and the almost time to go home of the other. We must go beyond the cultural things which are often used as an excuse for failure.

Question: How do you manage a project which is really enhancing the total resources of the firm, but which may be seen by management as a new application?
Charles: Structure a set of discreet user-visible deliverables rather than an infrastructure project. Component based delivery offers a good paradigm for structuring these kinds of deliverables. Components offer new functionality iteratively without building new applications. New applications are heavyweight things and can be big risk producers - involving retraining users and support and being cognitively disruptive for users. Components add the functionality the users want without forcing them to learn a new desktop.
Kenny: There isn't one kind of application development project. Even within one customer we've identified many different kinds of projects. You can't draw one grand unified strategy for managing these projects, you need strategies for defining how you would handle each individually.

Question: There was a comment this morning that software engineering has failed - which received resounding applause from the audience. The panel has largely described standard software engineering practices applied to OO and yet computer science departments don't teach software engineering. What's going on?
Jim: I cannot reconcile the good practices that I see with anything that comes out of software engineering. I don't see it practiced the way it says in books. If you look at how productive teams work it's as applicable to OO today as to other paradigms in the past and it may not have much to do with what we're calling software engineering.
Doug: It's not software engineering that has failed but application of it that has failed. There are a number of books on OO or traditional project management — you can take any one, follow it, and do better than most project managers do today. The problem is that project management techniques are not taught.

Question: In practical experience, how do you find the mechanism of synchronizing design models? What do you do in reality?
Charles: We are pursuing a business model that serves as a catalog for everyoneís understanding, but supports many implementations. We pulled back from close coupling between the analysis model and all its representations. The organization has many moving parts, including new development and legacy attempting to put everyone in ìlock stepî simply locks everyone. Decoupling is the way for everyone to go about their business.
Kenny: Most people are looking for the grand unified tool solution but I haven't seen any. Some people try to avoid the problem by turning an inter-project reuse problem into an intra-project reuse problem by expanding the project boundaries to avoid the need for later synchronization. This quickly turns into enterprise modeling.

Question: How do we get our clients to focus on a value-produced model, rather than a cost-based model?
John: It's next to impossible to determine the cost of things. People are always willing to invest if they can see a return. We need to concentrate on giving them the return and letting them measure it for themselves. Evolutionary systems help because they get stuff earlier. It's difficult to determine the order of things to construct - we are often thinking in terms of the best political order, rather than the best technical order. The trust and confidence of project sponsors must be uppermost in the mind of project managers.

Question: How do we analyze our enterprise such that we can use the analysis models as a high level description without getting stuck in analysis-paralysis?
Charles: Enterprise analysis is fundamentally a different level of abstraction than working on a single project model. The goals of an enterprise analysis should be: to gather use cases, identify classes and patterns, standardize nomenclature, and identify relationships. Step away from large inheritance based reuse since classes appear differently for different applications. Step away from an attributed analysis model - it's too low level. A common understanding at a high level of abstraction is a good anchor point for a

catalog for understanding why so many different versions exist.

Kenny: You have to be willing to take the risk of being wrong and rely on the technology available to be pliable enough so that you can recover. You can't live in fear that you're not building the most reusable component. You have to go on and then revisit if you need to.

Doug: Reusable objects are discovered, not invented. Organizations that have tried enterprise modelling have either dropped into analysis-paralysis or built something so complex that no one can use it.

Question: Doug said that large projects have to have a modified waterfall process. Do the other panel members agree?

John: I don't agree. Obviously size has an effect - size makes planning an evolutionary process much harder. I just don't get involved in projects that are too big. At some point, the large size increases risk to the point of being unacceptable. An organization should have less ambition and less risk.

Jim: There are other factors besides size, such as rigid or demanding markets. The market may often demand well-timed benchmarks. Needing to plan to slot projects into multi million dollar infrastructures doesn't lend itself well to iterative or incremental development. However, you can often keep the customer happy with a waterfall on top and another process underneath with some kind of mapping.

Question: The new books on OO project management suffer from a lot of the problems of early object texts — at the platitude level, all is obvious - but trying to implement these things and getting to the proper level of detail is very hard. Especially getting a lot of centralized chokepoints — you either go for a centralized model — or a decentralized model that needs lots of reconciliation.

Jim: Organizational patterns try to attack that — they tell you very specifically what to do. What's missing is the big picture — but it will say if you have this problem, try this.

Kenny: You've pointed out two extremes — the entire process is either centralized or de-centralized. In practice, we see something in between. When you start off a project - if its large, you'll want to partition it. How could you be smart enough at the beginning of a project to do a good partition? You must start some analysis to get a big enough picture to allow you to partition it up to begin with - with the understanding that couplings will be limited to interfaces. You can combine the best of both.

Doug: You don't learn much of anything from books — you must go in and try.

Charles: Software architecture does have a strong impact on the effectiveness of an organization, so divide organizational strategies according to architecture scaleability. Code reuse is much easier in small groups, so organize people into small groups with related class deliverables. Components, patterns and design reuse scale up well, so have a component based delivery architecture for the organization with a pattern-based analysis model for centralized understanding of the business.