8-16-1996

Object-Oriented Project Management

Vijay Kanabar
Boston University

Larry Appleman
Metropolitan college

John Gorgone
Bentley College

Object-Oriented Project Management

Vijay Kanabar & Larry Appleman, Boston University, Metropolitan College
John Gorgone, Bentley College

Object-orientation is a way of looking at the world as "classes" of "objects" in order to model the real world more effectively than traditional structured software engineering or other methods of computer programming. Objects are descriptions of sets of behaviors, often in the real world, but just as often in an imaginary world. Classes describe sets of objects which have shared properties.

Object-oriented software engineering has existed in academic and research settings since the late 1960s, undergoing thorough research throughout the 1970s. During the 1980's it became accepted for commercial and industrial use. Objects are widely viewed as having behavior, which are described by its associated class (or classes).

In addition to object-orientation's putative advantage of better modeling of the real world, object-oriented software development offers other distinct and substantial benefits over other philosophies of software development (Goldberg & Rubin, 1995): objects are reusable, thus saving effort in the long run; objects are a means of attacking, understanding, and communicating about complex systems; objects may be used to build systems which are flexible and resilient to change; objects allow partial systems to work. Object-oriented software development is generally performed in an environment of iterative development, incremental delivery, and rapid prototyping, thus allowing more customer involvement and feedback during the development process (Dué & Henderson-Sellers, 1995). These are some of the reasons that object-oriented software development is growing in popularity for commercial applications.

Managing Object-Oriented Software Development Projects.

[The presentation illustrates various concepts introduced using our real-world experience of the MAS (Academic System) that used Object-Oriented techniques. Due to space constraints the case study is omitted. Please contact the first author to obtain the case.]

As object-oriented software development has been adopted by commercial and industrial firms, it has been observed that previously existing techniques and tools of project management have been insufficient. One expert writes that object-oriented project management requires "a whole new way of doing things," and "traditional project management techniques are fundamentally ill-suited to object-oriented projects," as the following analogy illustrates (Swanstrom, 1995). Some of the differences between object-oriented and the traditional forms of software development are examined below.

Lifecycle. The iterative or parallel-recursive "rapid prototyping" lifecycle of object-oriented projects is different from the traditional "waterfall model". "Spiral" lifecycle as introduced by Boehm at TRW is also a suitable model for use with object-oriented projects. [MAS System: used Spiral Model.]

Requirements. Some iterative object-oriented development projects use "cases and scenarios," which almost anecdotally describe situations under which a system's requirements are discovered, captured, documented, and communicated. It should be noted, however, that many object-oriented development methods are non-iterative as well. [MAS System: ESS/DSS requirements were captured this way.]

Planning. A significant difference between object-oriented and traditional software projects iteration: the regularly repeated delivery (through the point of actual coding and testing) of a portion of the end-product's functionality. Plans for object-oriented projects may have to reflect multiple iterations, with the quantity varying based on size and complexity of the project. Iterations should be carefully planned, so that earlier ones may serve as a foundation for later ones. A rule of thumb for planning is that iterations should usually be no longer than three months in length (Reid, 1994). From our experience it is also suggested that the number of iterations per lifecycle phase be limited to three. [MAS System: Multiple iterations created major problems… project manager intervenes with a "three strikes & out" approach]

Reusability: As one of the principal goals of object-oriented software development is reusability project managers may find it useful to identify a separate timeline for identifying reusable components. Furthermore, project plans for object-oriented projects may be treated as a reusable set of artifacts, which should have schedule and staffing templates that can be adapted to many different projects (Booch, 1996).

*Estimating***.** Because object-oriented technology is relatively new, project managers do not have a set of well-tuned data allowing successful estimation by traditional models such as COCOMO or Price-S. Object-oriented project managers have to collect entirely new species of metrics, and most object-oriented projects today must rely on informally gathered estimates (Booch, 1996). Some current parametric models such as 4GT Model (Kanabar, 1992) have successfully been applied to oriented database development projects. [MAS System used the 4GT Model to get a ball-park estimate.]

Risk management. The iterative style of object-oriented projects mitigates several risks such as: clarifying user requirements up front, and pre-testing project feasibility. Regardless, a proactive approach to risk management needs to be practiced here. [MAS System: Several risks were successfully identified: new technology, new tools, buggy software, inexperience, unique approach. Major risk that was not anticipated was: data conversion.]

Measuring Progress: Appropriate measures may include a number of key classes, support classes, and classes per subsystem; number of interface operations, message sends, and nesting levels; classes per developer; and particularly illustrative of object-oriented engineering, the number of classes reused in a project, by counting classes at the outset of a project and at the end. (Dué & Henderson-Sellers, 1995).

Team roles. For an object-oriented software development project team, new professional roles may be necessary (Dué & Henderson-Sellers, 1995): librarians, for managing class libraries; library-class consultants; strategic library planners; library-class programmers (at the foundation and application levels); application programmers/prototypers; requirements analysts; implementation designers; toolsmiths; testers; trainers; quality assurance personnel; scribes; modeling experts; linchpin personnel to balance competitiveness between teams; and gurus.

What Project Management Can Learn from Object Orientation

The new techniques described above were discussed in the context of object-oriented software engineering projects. Some of these techniques may be applicable to project management in general. Two directions in application of object-orientation to project management appear to be particularly promising.

First, any project can be modeled using object-oriented techniques, thus providing a potentially useful tool for project managers. An object model has been successfully built for the software engineering process, as a tool for describing, understanding, and simulating the course of a project, and to provide a disciplined and controlled opportunity for change management (Brandl & Worley, 1993) and (Abdel-Hamid, 1993). Because object-oriented technology models the real world, such an object-oriented model of a process can be used effectively to help software development project managers to better understand the software engineering process, and as a tool to use in planning, resource use, and scheduling.

Second, project managers may use object-oriented techniques to develop reusable artifacts to allow more effective management. We may find it instructive and useful to think of a project as having reusable artifacts, including (Booch, 1996): architecture, design, code, requirements, data, human interfaces, estimates, project plans, test plans, documentation. Even in a non-software project we may identify most of these artifacts.

Often, project management plans, schedules, and estimates are developed from scratch, based on the experience of the project planner and on some guiding principles. By applying object-oriented thinking to project management, a more uniform approach may be developed (Stewart, 1992). The ideal application of object-oriented thinking to project management would include the classification of experiences into

reusable objects, so that an "experience database" would exist in the form of an object library. Each of the project management phases -- concept, definition, execution, and finishing -- could be represented as "activity objects." These could then be broken down into more detailed activity objects, so that firms could recognize common attributes and characteristics of various activities involved in managing a project. A project management experience library could consist of activity objects which combined results from multiple development projects which used the same kind of activity object. Each activity object would be carefully described so that useful objects can be found and retrieved; activity objects may have external attributes, such as cost, schedule, and status.

Other possible objects might be "work product" objects, modeling the physical result of activities, such as requirements documents, work-breakdown structures, cost reports. Such work product objects would have unique descriptions and external attributes, such as title, type, state, and version. An experience library may also contain prototypes of activity objects, work product objects, and other object classes (say, product objects). These prototypes could be used by a project planner as a basis for developing plans, and therefore may only have to spend time analyzing tasks which require new technologies or methodologies. This suggests that, by taking advantage of the "reusability" and "encapsulation" aspects projects can be better planned and estimated.

Finally, one can apply object-oriented techniques to the design and development of business process modeling and to knowledge-based systems (frames) as well. Unlike frames which are passive data structures, the objects, which encapsulate both state and behavior, are active. They execute operations in direct response to messages received.

Conclusion

As object-oriented project management techniques mature, some aspects of object-oriented thinking may yield substantial improvements in project management even for non-object-oriented, non-software projects. Promising developments include object modeling of project processes, and libraries built from project management artifacts so that these artifacts may be effectively reused.

References

Abdel-Hamid, T, and Madnick, S, *Software Project Dynamics: An Integrated Approach*, Englewood Cliffs, NJ: Prentice-Hall, 1991.

Boehm, B., *Improving Software Productivity*, IEEE Computer 20(8), 1987, pp. 43-58.

Booch, G., *Object Solutions: Managing the Object-Oriented Project,* Addison-Wesley, CA (1996).

Brandl, D. L. and Worley, J. H., An implemented object model of the software engineering process, *Journal of Systems and Software*, v. 23, 171-181 (November 1993).

Dué, R. T. and Henderson-Sellers, B., The changing paradigm for object project management, *Object Magazine*, v. 5, n. 4, 55 (July-August 1995)

Goldberg, J. and Rubin, K. S., *Succeeding with Objects: Decision Frameworks for Project Management*, Addison-Wesley, California (1995).

Kanabar, V. Modelling of Software Development Effort for the Fourth Generation Environment, Micromedia, Ltd., Ottawa (1991).

Reid, G., New management issues becoming pervasive, *Computing Canada*, v.20, n.15, 20 (July 20, 1994).

Stewart, A. M., How object-orientation can help project management, *Computerworld*, v. 26, n. 34, 86 (August 24, 1992).

Swanstrom, E., Beyond methodology transfer: O-O mentoring meets project management, *Journal of Object-Oriented Programming*, v. 8, n. 1, 57 (March - April 1995).