Object-Oriented Analysis in the Real World

Michael M. Lee

Project Technology, Inc.
10940 Bigge Street
San Leandro, California 94577-1123
510 567-0255
http://www.projtech.com

1. Introduction

Theory and Practice

There is theory, and there is practice. There is the plan, and there is its execution. This paper concerns itself with the practice and execution of using Shlaer-Mellor object-oriented analysis (OOA) [1, 2] on real-time control systems. The task of the project manager is to make the theory and the plan work together in the context of a given organization, project, and engineering team. This paper describes experiences in this world and presents an approach, evolved over time and with the help of other colleagues working in this area, for succeeding with this task.

The Projects

The experiences described in this paper were gained on five separate projects. They involved medium to large real-time systems. The organizations typically had little or no prior experience with formal analysis methods. The motivations for using a more formal approach included efforts to

- reuse all or significant parts of the system,
- manage the size and complexity of the system,
- develop a long-term, product platform and
- improve the software development process.

The first two projects were on distributed minicomputer platforms. The last three were on embedded, distributed, microcomputer platforms. The following table provides a summary of these projects.

Business	Application	Size (SW staff)	Duration	Result
Aluminum Manufacturing	Rolling Mills	14	3.5 years	Factory Installation
Communications	Material Handling	12	2.5 years	Operational Prototype
Medical Instruments	Imaging	22	5.0 years	In Progress (design/code)
Computer Peripherals	Tape System	15	3.0 years	In Progress (design/code)
Computer Peripherals	Disk Array System	95	4.0 years	In Progress (analysis/design)

Object-Oriented Analysis
Shlaer-Mellor object-oriented analysis uses an integrated set of models to identify the conceptual entities (objects) in the system, their dynamic behavior, and the required processing. The reasons for using this approach include

- its explicit attention to real-time issues,
- the ability to verify the analysis by simulating the models,
- the availability of textbooks, training and technical support for it, and
- its prescribed approach for transitioning from analysis to design.

The reader is encouraged to read the first two references for more details on this approach.

2. The Real World

Practice, as opposed to theory, is typically neither pretty nor clean. In this section, those aspects of projects which *are* sometimes ugly and dirty are considered. The issue here is not to point a self-righteous finger at these areas, but to recognize that they do exist and will have a negative impact on the project if not effectively addressed. The areas in which OOA can offer assistance are noted.

Requirements
The problems that arise when requirements for a software system are poorly understood are widely recognized and well documented in a number of books and articles [3, 4, 5]. Still, the problems persist, and, in my estimation, will never totally disappear. Here are a few reasons for this.

- *Pushing the Technology Envelope*. When systems are expanding into areas of new technology, the "requirements" are frequently being discovered as development proceeds.

- *Building Product Platforms.* When it is the objective of a project to develop a platform to support a family of products over an extended period of time, seldom are all the members of this family defined yet. Many are little more than a glimmer in some marketing VP's eye.

In situations like this, requirements are never well understood at the beginning, nor do they remain static over time.

What is needed is an analysis method that allows the engineer to 1) quantify, in some manner, the tradeoffs between different sets of requirements, and 2) assess the ability to meet multiple sets of requirements. The OOA models, through their ability to simulate execution, provide excellent support for this.

Schedules
Producing relevant schedules for software development is even more problematic than doing the development itself. This is especially true for the initial analysis phase when we don't know what we don't know yet. Add to that a learning curve of uncertain slope for a new analysis method and it truly becomes a Herculean task. No analysis method, it would seem can offer the clairvoyant assistance needed here. What we can ask of a good method, however, is to provide relevant feedback for adjusting and calibrating the schedule. Suggestions for how to do this with OOA are described in the next section.

Size/Complexity
Of issue here is not that systems are large or complex, but that their true size and complexity are often poorly understood. As long as this is the case, schedule projections suffer and a mild sense of being out of control persists. It is in the analysis phase that the size/complexity question must be answered, and the analysis method should have two important characteristics to deal with this. First is the ability to scale up for large and complex problems if necessary. Second is the ability to factor and reduce the size and complexity to their smallest possible dimensions. The Domain Chart of OOA is a valuable tool for doing this and its use is described in the next section.

Engineering Skills
Successful software engineering requires a number of very different technical skills. These include the ability to do analysis, system design, program design, coding, integration, and testing. Very few individuals are equally talented in all of these areas. My informal sampling of software engineers produces a distribution that peaks in the middle with program design and coding, and falls off in both directions with good analysts and testers being the most rare. This creates two challenges during the analysis phase. One, good analysts must be identified and assigned judiciously. Two, productive work must be found for members with other skills. OOA offers some help in this area by having different roles within the analysis phase. The design approach associated with OOA (Recursive Design [6]) also offers opportunities to begin some design and implementation tasks early in some areas, often while analysis is still proceeding in other areas.

Silver Bullet Syndrome
Perhaps the most insidious real-world problem when introducing new software methods is the over zealous and unrealistic expectation that the next new method (object-oriented anything, these days) will prove to be the long awaited "silver bullet" with which the dreaded software development dragon can be slayed. It is insidious because it carries enthusiasm that is short lived and a commitment that is only slightly longer lived. Addressing this problem is outside the scope of *any* particular method, though it must be done if a method is to be assessed on its true merits. The best policy here is complete frankness on the benefits and liabilities of a method, and an ability to adjust for these.

3. Practical Approach to Applying OOA

Analysis Steps

Halfway through the third OOA project, when some of the difficulties encountered began to feel familiar, an attempt was made to identify and categorize these difficulties. (Note that the difficulties referred to here are not with the method, but those arising from "making the theory and plan work together in the context of a given organization..." as described in the first section).

The strongest pattern that arose when categorizing these difficulties was that of time and sequence. Different problems came up at different times in the analysis, and different problems were dealt with more or less effectively depending on the sequence in which they were resolved. The best way to address this was to partition the analysis process into a set of distinct steps based on the sequence in which the issues were best addressed. This way each step builds on the work of the preceding steps. These steps are to:

- assess applicability of OOA,
- gain management support,
- initiate analysis, and
- sustain analysis.

This simple sequence of steps performs the age-old function of breaking a problem down into smaller pieces to reduce the difficulty, and to guide the process. The following subsections address the issues and proposed approach for each of these steps.

Assess Applicability of OOA

As basic as this step may seem, it is seldom done in as thorough a manner as it should be before the other steps are started. This can cause difficulties in the later steps when it continues to be assessed, multiple times, by different groups, possibly with different results! The basic assessment which must be made is the *match between the projects objectives and the method's capabilities.* There are three basic areas which need to be considered in making this assessment

- *Technical Considerations.* Is OOA the right tool for the job? Does it address the difficult issues which the project expects to encounter? Does it help meet the project's technical objectives?

- *Management Considerations.* Can the project lifecycle be "front loaded"? Will there be support for training? Is there a perceived need for this technology? Does it help meet the project's business objectives?

- *Staff Considerations.* Is there an existing analysis method? Does it work? Is there a perceived need for this technology? Does it help the staff meet their professional objectives?

When assessing these issues, one needs to be aware of project objectives that may and may not be addressed by OOA. Examples of project characteristics that can be addressed extremely well by using OOA include analyzing large and/or complex systems; developing configurable, data-driven systems; preparing for object-oriented design and programming; designing for reuse and maintenance; and automating the implementation.

Examples of project characteristics that may not be addressed particularly well by using OOA include the "n-th iteration" on a well understood problem; a project which is already absorbing a number of other new and different technologies; projects with extremely aggressive schedules; and systems where "quick and dirty" is an acceptable approach. Meilir Page-Jones has recently compiled an interesting list of motivations for taking, and not taking, an object-oriented approach which expands on this topic [7].

Gain Management Support

Having determined that there is a good match between the project's objectives and the capabilities of OOA, the next step is to gain management support for the introduction of this new analysis method. The support that will be needed will include funding for such things as training and CASE tools, commitment to a schedule that will spend more time in analysis and design and correspondingly less in implementation, organizational changes to accommodate the different skill profiles required for this work, and cultural changes in the way work gets documented and reviewed.

It's important to gain this support based on *realistic expectations* of how the work will proceed. It will do more harm than good to gain support based on unrealistic expectations like : "it's object-oriented, so we'll see a 10 fold increase in productivity immediately", or "the learning curve impact is just the training time". Such support is very likely to dissipate quickly as these unrealistic expectations are not met, and the project will suffer accordingly.

At the heart of gaining management support is convincing the appropriate managers that there is a good match between the project's objectives and OOA's capabilities. The following tasks are key for doing this.

- *Present Assessment of OOA's Applicability.* Having established that there is a good match between the project's objectives and OOA's capabilities, this must be communicated to the management team. They should have an opportunity to probe the assessment, have their questions answered and hopefully come to the same conclusion.

- *Demonstrate Benefits of Good Match.* It's important at this point to recognize that you are involved in making a *sale*. The best way to succeed with this is to go beyond just establishing the good match and elaborate on other benefits to be gained on future projects when OOA, and hopefully a significant amounts of code, can be reused.

- *Establish Realistic Expectations.* It's extremely important at this point, for reasons cited earlier, to ensure that the above benefits are presented in a realistic light, and that the costs acknowledged. Typical costs are training, learning curve, CASE tools, and possibly technical support. Typical changes in "business as usual" are some organizational shuffling, new documentation and review procedures, and more time spent at the front end of the project.

- *Present a Plan.* A good management team will not commit to a process without at least a preliminary plan of execution. Be prepared to provide this. Base it on the prescribed approach of constructing a Domain Chart, Subsystem Diagrams, and Project Matrix. More will be said in later subsections about how to manage this planning.

Initiate Analysis

The task at this step is to overcome the inertia of initiating activity and get the team moving in a coherent manner in roughly the same direction. To do this, the following tasks must be accomplished.

- Create a productive analysis environment.
- Train the technical and managerial staff.
- Create an executable, short-term plan.
- Initiate the analysis activities.

Productive Analysis Environment. The software profession has made great strides over the past few decades in improving the implementation environment. The programmers' workbench, coding standards, configuration management systems, and interactive debugging environments are a few good examples of this. By comparison, analysis environments are typically primitive, not that they need be, however. The tools do exist. It just seems that this receives little attention. If your plan is to spend a significant part of your schedule in this phase, it behooves you to give this more consideration. The essentials for a productive environment include the following.

- *Document Library*. Establish a central, public repository for project documentation. This is *the* deliverable during the analysis phase. It's important to manage it and ensure its common and convenient accessibility. This is most conveniently done "on line", though a manual system can also work.

- *Document Standards.* It's a small thing, keep it simple, but do it! Designate a document and drawing tool. Ensure that title, author, version, and date information are uniformly included on every document.

- *Meeting Guidelines.* Unproductive meetings place unnecessarily large overheads on the analysis process. For productive meetings, clarify objectives, limit allocated time, and stick to agendas. Distinguish between the following three types of meetings. A working meeting for getting technical work done. A presentation meeting for informal communication of the results of a piece of work. A review meeting for formally reviewing a work product.

- *CASE Tool.* OOA generates a set of integrated models which all have graphic representations. This is not something that can productively be done manually or with a simple drawing tool. The minimum requirement is a professional drafting tool and a draftsperson. A more desirable alternative is a networked CASE tool to which all the analysts have access.

Train Staff. Three courses exist for OOA/RD and the team should attend them all. The best thing to do is to train the team in OOA as a group immediately before starting the analysis effort. Send both technical staff and their managers. The technical staff for obvious reasons. The managers so they understand the process they'll be attempting to schedule and control. An important point to understand about the training is what it can and cannot accomplish. One, it will not create analysts out of people without these skills. Two, it will only impart knowledge not experience. Three, it's folly not to do it.

Create a Plan. The plan created in the previous step of gaining management support was really a strategic statement outlining a general approach. Now a tactical statement is needed to clarify the week-to-week activities and focus the effort. The best guidance that can be offered is to initially target those areas of the application which are expected to provide additional insights. This may mean different things on different projects. Sometimes doing a well-understood area to gain experience with OOA method helps. Sometimes doing a particularly difficult area to demonstrate the capability of the OOA method helps. Sometimes the structure of the problem suggests an advantageous starting point. The important point is to target your efforts for a defined objective as a means of providing a focus for your activities. Taking an undirected, breadth first approach to the problem is very likely to stall out without producing anything of significance.

Initiate Analysis. Two things are important as you begin the analysis activities. One is organizing the analysis teams, and the other is recognizing that the analysis process is frequently akin to prospecting.

When organizing the analysis team, both size and skill are critical. Small teams of two to four people seem to work best. At least two people are needed to provide second opinions, and more than four tend to either leave some members out of the process or slow down as the diversity of opinions are dealt with. The skill issue is both obvious and yet difficult to address. The obvious part is ensuring that each team has at least one skilled analyst. The difficult part is identifying the good analysts, and then keeping them evenly distributed among the different analysis areas.

A productive mind set as analysis begins is that of an oil prospector. It's impossible to accurately predict where the oil will be found. Rather than agonize over exactly where to drill, or continue to drill in areas which have not yielded anything, the best thing to do is make an educated guess, proceed with sinking some holes, and use the results to guide your future work. From an OOA perspective, this means identifying potentially fruitful areas and proceeding to build the information models in these areas. The results of this analysis will help guide the direction of future work.

Sustain Analysis

As the different analysis teams begin to develop and review their OOA models, new issues can arise which affect the ability to successfully sustain the analysis effort. These include confusion with details of the OOA formalisms, OOA models that are only understood by their developers, models which can never be completed, confusion with what to model and what not to model, difficulties with integrating the models from different teams, the analysis plans becoming obsolete, and management support wavering as schedule pressures increase. A number of things can be done to address, and in many cases prevent, these problems.

- Augment the formal models with informal notes and diagrams.
- Review the models.
- Leverage available analysis experience.
- Distinguish between analysis and design issues.
- Maintain the domain chart and subsystem interfaces documents.
- Demonstrate tangible progress on a regular basis.
- Use results to calibrate and maintain the plan.

Informal Notes and Diagrams. The OOA formalisms (information model, state model, and process model) are an excellent means to model the conceptual entities and behavior of a system precisely. They are not always sufficient, however, for understanding the underlying abstractions upon which the models are based. What we want to capture are the white board pictures and explanations that initially produced the "Yes, that's the way to think about this!" insights. What works best in situations like this is to produce an informal document that explains the thinking and underlying abstractions behind the models as a prelude to doing the formal models. Using diagrams and pictures in this type of document greatly enhances its ability to communicate complex ideas.

With these informal documents (commonly called "technical notes"), it is possible to capture and communicate some of the most important work that goes on during the analysis phase. It provides a good foundation for developing the OOA models. It makes them more comprehensible after they have been developed. It allows conceptual approaches to be assessed prior to investing time in modeling them. And it provides a convenient mini-step which aids in tracking progress.

Review Models. A discipline of reviewing analysis models when they reach an acceptable level of maturity must be established on the project. This monitors quality, measures progress, provides interactions between different analysis teams, guards against unnecessary "polishing", and produces a sense of progress as tasks are completed. Each model should be reviewed as it is completed. For each subsystem, this means a review for the information model, state model set, object communication model, and process model set. For the system this means reviews for the domain chart and the subsystem information, communication and access models. Reviews can also be helpful prior to "shelving" a piece of work for some time, and when a piece of work is "stuck". These reviews should be formal assessments of the work product, documented with meeting minutes, and generally follow good software inspection procedures.

Leverage Available Analysis Experience. This experience typically come from two sources. First, from other projects within the organization that have used OOA, and second from within the project, as some team members pick up the analysis technique more quickly than others. These individuals need to be identified and effectively assigned to the important analysis problems to leverage their skills. This requires a certain degree of flexibility in being able to assign and reassign team members to tasks. This flexibility can be problematic in certain organizations and with particular individuals. It helps if the dynamic nature of assignments during the analysis phase can be established early.

If the availability of in-house experience is insufficient, outside help from experienced OOA practitioners should be sought to bootstrap the project. This will significantly enhance the speed of the technology transfer and early self sufficiency of the organization.

Distinguish between Analysis and Design. Analysis should focus on the application. Design should focus on the implementation. When implementation issues begin to creep into the analysis, the job gets bigger and the issues get cloudier. The result is that the analysis process can bog down. One needs to be on constant guard against this. The OOA courses are very explicit about this distinction, and the message needs to be reinforced throughout the analysis phase. The Recursive Design course can also help by clarifying how the analysis models are transformed in design.

Maintain Domain Chart and Subsystem Interfaces. On the one hand, the partitioning of the system into domains and subsystems is essential. On the other hand, it cannot effectively be done in one pass. Not enough is known at the start of the project to do this correctly. Given this dilemma, how should the project proceed? A best guess, based on good engineering judgment, must be made at the start of the project. Thereafter, it is important to revisit this choice periodically and make adjustments as more is learned through the analysis. The process then becomes one of refining the system vision as more is learned about the individual components. One wants to avoid wasting time at the start of the project agonizing over the perfect partitioning. One also wants to ensure that the insights gained through analysis get factored back into the partitioning and an integrated system vision.

Demonstrate Tangible Progress. This should be done on a regular basis for two important reasons. First, it encourages continued management support for the project and its team. Very few management processes accept the Big Bang (results will suddenly appear overnight) theory of development and hence want to see incremental progress. Second, it has a positive impact on the morale of the team to recognize their progress. It's easy for them to lose sight of this progress when the daily focus is on the work that remains to be done.

There are a number of analysis activities that can be used to demonstrate tangible progress: a revised domain chart which reflects a better understanding of the different subject matters in the system; new conceptual insights documented in technical notes; completion of reviews; successful subsystem partitioning of a domain allowing parallel work to proceed. The analysts should take the initiative on documenting and communicating these achievements. The best defense is a good offense!

Calibrate and Maintain the Plan. As the analysis effort proceeds, more is learned about the scope of the work and the rate at which it can be done. This information needs to be factored back into the analysis plan on a regular basis to understand its impact and allow for timely adjustments. The situation here is very analogous to creating and maintaining the domain chart. Good engineering judgment is used to initialize the plan and adjustments should be made as new information is available. Adjustments to scope should track changes to the domain chart. Adjustments to efficiency need to be made far enough apart to ensure significant data and close enough to recognize problems. My rule of thumb is to do it between every 3 to 6 months. The guiding principle is that, to be useful, the plan must reflect a reasonable approximation of what the team is capable.

4. Summary

There is a lot more to succeeding with OOA than mastering the modeling techniques. Some of it is just good project management practices. Some of it is just good analysis practices. Some of it has to do with the process of introducing a new engineering discipline into an organization. Some of it has to do with the general, object-oriented approach of OOA. And some of has to do with application of the specific OOA modeling techniques. If the issues in all of these areas are actively tracked and addressed, the chances of success are greatly enhanced. The following checklist summarizes critical activities.

- Establish utility of OOA.
- Create productive analysis environment.
- Train the analysis team.
- Use experienced OOA analysts.
- Augment formal models with informal technical notes.
- Factor analysis results back into the maintenance of the domain chart and subsystem interfaces, and into the project plan.

References

[1] Sally Shlaer, Stephen J. Mellor, "Object-Oriented Systems Analysis -- Modeling the World in Data", Prentice Hall, 1988.

[2] Sally Shlaer, Stephen J. Mellor, "Object Lifecycles -- Modeling the World in States", Prentice Hall, 1991.

[3] Tom DeMarco, "Structured Analysis and System Specification", Yourdon Press, 1978.

[4] T.E. Bell, T.A. Thayer, "Software Requirements: Are They Really a Problem?", Proceedings, 2nd International Conference on Software Engineering, 1976.

[5] Harlan D. Mills, "Software Engineering", IEEE Transactions on Software Engineering, Volume SE-2, No. 4, December 1976.

[6] Sally Shlaer, Stephen J. Mellor, "Recursive Design", Computer Language, March 1990.

[7] Meilir Page-Jones, "Object-Orientation: The importance of being earnest", Object Magazine, July-August, 1992.