NASA-CR-193706

186042

8p

Advanced Software Development Workstation

Object-Oriented Methodologies and Applications for Flight Planning and Mission Operations

Michel Izygon

Barrios Technology, Inc.

June 30, 1993

Cooperative Agreement NCC 9-16
Research Activity No. SR.02

NASA Johnson Space Center
Information Systems Directorate
Technology Development Division

Research Institute for Computing and Information Systems
University of Houston-Clear Lake

INTERIM REPORT

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information Systems (RICIS) in 1986 to encourage the NASA Johnson Space Center (JSC) and local industry to actively support research in the computing and information sciences. As part of this endeavor, UHCL proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a continuing cooperative agreement with UHCL beginning in May 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The UHCL/RICIS mission is to conduct, coordinate, and disseminate research and professional level education in computing and information systems to serve the needs of the government, industry, community and academia. RICIS combines resources of UHCL and its gateway affiliates to research and develop materials, prototypes and publications on topics of mutual interest to its sponsors and researchers. Within UHCL, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business and Public Administration, Education, Human Sciences and Humanities, and Natural and Applied Sciences. RICIS also collaborates with industry in a companion program. This program is focused on serving the research and advanced development needs of industry.

Moreover, UHCL established relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research. For example, UHCL has entered into a special partnership with Texas A&M University to help oversee RICIS research and education programs, while other research organizations are involved via the "gateway" concept.

A major role of RICIS then is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. RICIS, working jointly with its sponsors, advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research and integrates technical results into the goals of UHCL, NASA/JSC and industry.

RICIS Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Dr. Michel Izygon of Barrios Technology, Inc. Dr. Rodney L. Bown served as the RICIS research coordinator.

Funding was provided by the Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA research coordinator for this activity was Ernest M. Fridge III, Deputy Chief of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

Advanced Software Development Workstation

Object-Oriented Methodologies and Applications for Flight Planning and Mission Operations

Prepared for
NASA-Johnson Space Center

June 30, 1993

Submitted by
Dr. Michel Izygon
Barrios Technology Inc.
1331 Gemini Av.
Houston, TEXAS 77058

ABSTRACT

This report summarizes the work accomplished during the past nine months in order to help three different organizations involved in Flight Planning and in Mission Operations systems, to transition to Object-Oriented Technology, by adopting one of the currently most widely used Object-Oriented Analysis and Design Methodology.

Object-Oriented Methodologies and Applications for Flight Planning and Mission Operations

I Introduction

This report is an initial summary of the work accomplished during the past nine months with three different groups within the Space Transportation System Operations Contract (STSOC), to assist them in the transition to Object-Oriented Technology (OOT).

The transition to Object-Oriented Technology is an issue of importance to many organizations inside NASA as they face the challenge of developing and maintaining increasing numbers of software systems with decreasing or at best constant budget. The main driver for an organization to transition to OOT is the potential significant cost savings. The primary benefits of an object-oriented development approach that allows reaching this cost savings goal are a high degree of reusability, a highly modular design, and ease of maintenance. The decision to adopt OOT is not an easy decision for the environment in which these teams work. The mission criticality of the systems combined with the feeling that OOT might be immature or oversold, and the significant up-front cost of the transition are factors taken into account in the decision making process.

The three STSOC groups, Mission Operations Computer (MOC), Reusable Object Support Environment (ROSE), and Reconfiguration Software Tools (RECON), faced the same problem and came to us for support. Their problems are globally identical, although, as we will see, some significant differences in the projects' constraints impacted the training differently. The common goal was (and still is) to rewrite their applications using Object-Oriented Technology.

Let us first describe succinctly the three different groups and their projects.

The first team we worked with is the Mission Operation Computer (MOC) group. They are in charge of the maintenance of the mission control center software, developed mainly in the 1960's in HLAL, a High Level Assembly Language for IBM Mainframe computers. Their system is divided into four main subsystems: Command, Trajectory, Telemetry and CCS Control. Their goal is to reengineer the system in order to significantly lower the maintenance cost. This project does not have stringent deadlines, and therefore the project's managers had the opportunity to introduce the OOT in a smooth way, i.e., with a small team of about 6 people working on two small subsystems, in order to prove to themselves and to their management that OOT is a valid path to follow.

The second team we worked with is the ROSE project group. This project aims at reengineering the Flight Analysis and Design Software (FADS). Part of this system, previously named SVDS, was developed over the past 20 years, is written in FORTRAN, and is known for the large amount of duplicate and dead code it contains. A pilot project has been defined which is supposed to give within a period of twelve months, a precise estimate of the cost of reengineering the complete system, as well as the definition of the process to be used for the full project, and of the training needs for the complete team. This pilot project started in mid-December and should be completed by December 1993. The time constraint and the number of persons involved in the project (20) made the transition to OOT quite challenging. A strong commitment to extensive training prior to any work starts on the pilot can be considered a good risk mitigation factor.

The third team we are working with is the RECON group. This group is in charge of the maintenance of the Reconfiguration Tools for the Shuttle software. The RECON system, developed 15 years ago, is a Management Information System (MIS) and contains about 4.5 Millions lines of code, written mostly in PL/1. The reengineering effort is constrained by a period of three years, and by the mainframe platform on which the software has to run. The effort started in December 1992, and the first phase deliverables are scheduled for next September. The language in which the system has to be re-written is C. This constraint made the move to OOT more difficult to implement as the language cannot support the OO features used during analysis and design. Furthermore the fact that the project has to be successfully finished within a period of three years, made the decision to move to a new technology more difficult.

II Training

With each of these teams, we faced the challenge of helping them transition to the Object-Oriented paradigm in a relatively short time, while at the same time they had to transition

to a completely new environment. All the teams' members were accustomed to working on mainframe platforms, with languages such as Assembly, PL/1 or FORTRAN, and they had to move to new platforms, new Operating Systems, new languages and new CASE tools. This cultural shock had to be properly managed. We were charged with the organization and planning of the training in OOT. Our role was threefold:
1- serve as consultant in recommending what method should be taught and how the training is to be done.
2- serve as mentor in getting the team up to speed on the chosen method.
3- serve as organizers of formal training, by contacting trainers and having the class tailored to the specific needs of each team.

First, we chose the OMT method developed by Rumbaugh to introduce the object-orientated concepts. The choice of the method was based on the following criteria:
- Our experience with different methods convinced us that this method is easier to learn and to apply. It is considered an "evolutionary" rather than a "revolutionary" method.
- It is one of the two methods most widely used today (Booch's is the other one).
- The book is clearly written and easily understandable, with many exercises.
- A tool, Paradigm Plus, supports it and is available to us.
- This method emphasizes the Analysis Phase of the software development life cycle, which we believe is of critical importance in order to take full advantage of the Object-Oriented paradigm. Moreover, it is also usable in the Design Phase, and therefore allows a seamless integration of the Analysis and Design phases.

The second part of our involvement in the training focused on getting these teams up to speed on OOT. As most of the teams' members did not have previous experience in the Object-Oriented paradigm, we felt that the usual way of getting training, i.e., a one-week class on Object-Oriented Analysis and Design concepts, may not allow everyone to understand precisely the OOT fundamentals and therefore would not allow them to be really operational in applying them. A paradigm shift cannot happen overnight. Absorbing these concepts takes longer. Not only formal training is required, but also apprenticeship and mentoring. Therefore, we decided to organize mentor-based study group meetings during which the book written by James Rumbaugh [1] on the OMT method will be discussed. These two-hour study group meetings are held twice a week at two-day intervals. For the MOC project the group consisted of 6-7 people plus two acting as mentors. Our role during the meetings is to guide them through the reading and to facilitate the discussion on specific crucial issues. Before each meeting, the group reads an assigned chapter, prepares a list of questions on eventual misunderstood problems, and works some exercises to see if they could apply the contents of the chapter. We provided a tentative timeline on the reading of the book and outlined the specific chapters they should concentrate on. These were the chapters 3, 5, 6, 8, 9 and 10 of the book.

The third part of our work in helping these teams, was to organize the formal training. We selected the best solution for the training by considering the specific needs of each team, the training money available, and the trainers' availability and courses. Thus, for the MOC team, we had a University Professor giving a one-day intensive class on Object-Oriented Analysis Design and Programming. As one of the issues of interest was the differences between Ada and C++ for OO support, we asked the instructor to customize his class to include the comparison of the two languages. We also worked with the instructor to develop an example in which the OMT could be applied from Analysis to implementation. This class was successful in helping the team decide which language to choose. However, the limited experience of the instructor in applying the OMT method to real projects showed up during the class. Knowing the principles of the method is not enough. This can be captured through reading of the book and study group classes, but the tips on how to make this knowledge practical is really what needs to be learned during the formal class.

For the other two groups we asked James Rumbaugh to come teach a five-day formal class on OMT. Of course this solution is expensive, but it has multiple advantages.
• First, as Rumbaugh has accumulated many years of experience in applying the method, he could complement the method with very useful hands-on tips.
• Second, we received the latest version of the method. As mentioned previously, the OOA/OOD field is still young and very dynamic.
• Third, directly asking questions to the creator of the method clarified the gray areas of the method, and gave a feeling of confidence to the team members.

Finally, we introduced the team members to the use of the Paradigm Plus tool, which supports the OMT method. By using the tool to develop the exercise models they could get a better feeling of the method, and it complemented the learning of the theory. The code generator capabilities of the tool also allowed them to start looking at code that is equivalent to the classes they were graphically describing.

III Operational use of OOT

One of the main problem we faced in helping these teams make the transition to OOT is that understanding the concepts is one thing, but applying them is another. In other words, learning OOT is like learning how to drive a car: you have not learned anything unless you have tried it. In this respect, Rumbaugh's book is quite good, as it contains many exercises at the end of each chapter that are corrected in a solution book, also available [2]. During the study group meetings, we used these exercises to allow the team members to apply the concepts learned in the book. Although very pedagogical, these exercises are too often far away from the domain of expertise and interest of the three STSOC teams. Consequently, working out these exercises appeared to produce some frustration among the teams' members. These exercises were felt by most to be too theoretical, and easy to solve or at least to understand the solution. Solving these did not seem to help them apply the concepts to their own domain. They felt that using the method on their problem domain was many orders of magnitude more complex than on these "toy exercises". In order to cope with this situation, we to apply the principles to increasingly difficult problems. Once the basic concepts were properly covered and applied through exercises, we used other exercises found in some OO publications such as the Journal of Object-Oriented Programming (JOOP) in which James Rumbaugh writes and presents some study cases with their solutions. Then, the next step in the climbing of the learning curve was to ask the team members to try to describe with the OMT the last software project they worked on. This type of exercise has the advantage that they know the domain well and therefore they can concentrate on remodeling this knowledge from the Object perspective.

The last step in applying the concepts is to start working on their next project for which they have to use the OMT method. This task can start while the group studies chapter 8 of Rumbaugh's book and remains an on-going task. It is used to illustrate the concepts presented, and at the same time to start the first iteration of the modeling work for their project. It is valuable to start this task before the formal training happens, in order to use some time during the training to discuss the early models with the instructor.

IV Lessons Learned

Let us try now to define the lessons learned from this threefold experience:

First, a mentor-based study group seems to be an excellent format to introduce an organization to Object-Oriented Analysis and Design methods. The flexibility of this format, which allows to go at the right speed for the defined goal, plus the fact that the organization can usually rely on in-house experts to start the process of the transition, seems to be considered as a safe or risk mitigating path. The efficiency of the study group format, i.e. the amount of knowledge obtained compared to the time invested, is good. A good indicator is the assessment made by Rumbaugh that the level of these teams was the highest he has ever encountered while teaching these classes.

Secondly, the study group meetings start the process of discussing the OO models, and this habit is kept for the real project. The study group is therefore also a training tool to review sessions for the real project. This is a very important and beneficial product. However, we must point out that these discussions on OO models can become quite animated. In fact, we noticed that once people start developing OO models, even if it is only for exercises, they become "emotionally" linked to their models and they do not appreciate critics. This issue must be taken seriously into account, and has to be dealt with properly.

Moreover, the pace of the study group meeting format is well adapted to the learning curve of the OO paradigm. This paradigm is something that people have to think about; it is not an absolute science but rather a way to look at and think about a problem. It requires some good analysis and good abstraction skills that can only be developed over time. Some people are better equipped than others with these skills and therefore pick up more rapidly the concepts, and transition faster. This is, of course, true for any new technology. The study group format made it quite obvious, but also allowed us to help those who were slower. It is worth mentioning that, as these projects were the first on which the organization decided to use OOT, the people chosen were some of their best people available. Also, most of the people involved in these study groups were highly motivated because they knew that learning these concepts could have a positive impact for their careers.

Reinforcement through the formal training allowed consolidation of the knowledge and gave the team members the feeling of confidence about their understanding of the concepts. It also answers specific questions or to clarify some areas still misunderstood.

Other lessons that are worth mentioning are the following:

• Some frustration happens when the team tries to apply the concepts they think they understand to their own problem domain, where there is no given solution.

• Some programmers have also the tendency to need to see the code very early on. Analyzing the problem is frustrating to them if they cannot get an idea of what the corresponding code looks like. In fact, this is at the core of the paradigm shift, as they learn to spend more time up-front at the analysis phase instead of during the implementation phase.

• Developing models is a very iterative effort and drawing them with pencil and paper quickly becomes a limitation to the method. Therefore, the use of the tool (Paradigm Plus) was a catalytic factor in the apprenticeship of the method.

V Conclusions

This report has tried to explain how we supported some organizations in their efforts to transition to Object-Oriented Technology. We believe that this transition can be successful in any case provided the management is committed to it. We defined in the past nine months some ways to achieve a smooth transition. Among the main conclusions is the fact the OMT method is an excellent way to introduce the OO concepts to people without any former knowledge of the OO paradigm. This is because it presents the key concepts in a relatively simple and clear way. Furthermore, because it emphasizes the analysis phase, the more complex problems related to OO Languages, such as dynamic binding or polymorphism, do not need to be dealt with early in the training.

We acknowledge that learning the OMT method is not a panacea. However, it allows one to start using the OO concepts, and therefore permits novices to acquire and build the required experience. The transition to OOT is too important not to be addressed correctly. We believe that using the study group format, combined with a top-level formal training, should allow any organization in the JSC community to move smoothly to Object-Oriented Technology and to get early benefits and return on investment.

Please recall that we have discussed only one area involved in the transition to Object-Oriented Technology. Other areas include the transition to an Object-Oriented Language such as C++ and the specific characteristics of Managing an OO project. These areas are of importance and they must also be dealt with properly.

References:

[1] Rumbaugh J. et. all, 1991
Object-Oriented Modeling and Design, Prentice-Hall

[2] Rumbaugh J. et. all, 1991
Solutions Manual. Object-Oriented Modeling and Design, Prentice-Hall