Logistics Management Institute

Use of Rapid Application Development Techniques

Designing the Staff Resource Tracking Tool

IR715T1

September 1997

John B. Harris
Frank L. Eichorn
David L. Goodwin
Joel L. Henson

LMI

Use of Rapid Application Development Techniques

Designing the Staff Resource Tracking Tool

IR715T1

September 1997

John B. Harris
Frank L. Eichorn
David L. Goodwin
Joel L. Henson

LOGISTICS MANAGEMENT INSTITUTE
2000 CORPORATE RIDGE
MCLEAN, VIRGINIA 22102-7805

Contents

FIGURES

TABLE

Chapter 1
Background and Scope

The practice of developing software for computer systems is constantly evolving and consistently a high priority. Whether one is building new weapon systems or new tax processing systems, the most costly segment is generally designing, constructing, testing, implementing, and maintaining the software. Accordingly, the search for improved programming languages and better methods for creating software applications is continuous and highly publicized and scrutinized.

RAPID APPLICATION DEVELOPMENT

One of the newer methods of designing and building software is called *rapid application development*, or RAD. "Newer" is used loosely; the RAD concepts have been known and practiced to some degree for a decade or more. But only in the last year or two have these concepts become widely supported by many well-received automated tools that help perform the design and programming chores. Increasingly, government managers of system development, some of them Logistics Management Institute (LMI) clients, are being faced with decisions about development tools and languages—with names like Visual Basic, Delphi, and Rational Rose—and ask our advice. Potential vendors are likely to propose that they "use tool A for requirements analysis, tool B for the design, and tool C for programming." Our clients then ask us if this plan makes sense; are they likely to get an operating and maintainable system as a result? The broad purpose of this task is to increase LMI's exposure to RAD methodologies and tools so that we are in a better position to provide valuable answers to these kinds of questions.

RAD is based on a few basic principles: (1) joint design teams with trained and motivated participants from both the development and functional user organizations, (2) integrated computer-aided software engineering (I-CASE) tools to capture requirements and design information and reuse it for software development purposes, and (3) an iterative process for demonstrating the software to users *as it is developed*, using the immediate feedback to converge on useful solutions and minimize undesired surprises.

OBJECT ORIENTATION

Recently, a complementary analysis/design/programming methodology (with associated automated tools) has become popular: *object-oriented* (OO) methodology and tools. These OO approaches represent a paradigm shift from some more traditional development methods such as structured analysis and design. Where those methods evolved around complex systems that used algorithms as the fundamental building block, OO methods have evolved around objects and classes as their building blocks. The difference can be appreciated by looking further at the *object model.*

The central component of the object model is an "object." Definitions vary, but we can generally say that from a systems standpoint, an object is a tangible entity that exhibits some well-defined behavior and combines the properties of procedures and data in one package. Objects are instances of some class or group of items that exhibit similar behavior and characteristics. Objects have a state or value and an object's behavior is how it reacts to changes in its state. For example, a vending machine is an object that exhibits different behavior when its state changes by a user putting money into the machine.

There are four elements that comprise the object model:

- *Abstraction*—essential characteristics that distinguish an object from all other kinds of objects
- *Encapsulation*—a means of packaging an object so that only valid operations on it are allowed
- *Modularity*—decomposing a system into cohesive, loosely coupled modules
- *Hierarchy*—a way to rank or order different abstractions of objects.

The use of object models has many benefits. Chief among them is that such systems tend to be resilient to change, making maintenance and enhancement easier. There is also the claim of reduced risk for complex systems because the process calls for integration of requirements, processes, and data throughout the life cycle. Despite these promises, many experts recommend caution when embracing OO technology.

There are two major reasons for this caution. First, certain types of problem domains, such as computation-intense applications, do not lend themselves well to OO technology. Second, OO development requires a shift in thinking for systems professionals schooled in structured techniques. Recognizing this, an organization should not make a commitment to OO without a trained and experienced staff. Many projects have failed because this prerequisite was not met. Thus, an important objective of this task was to have several LMI analysts learn, by attending a formal class, the details of OO methods. We also wanted to make use of that knowledge by considering process improvement at LMI and the potential to use OO tools to improve an LMI process.

Chapter 2 Approach

TRAINING

Since one of the project goals was to increase staff knowledge of OO principles, team members attended OO training courses. The first course was Object Oriented Analysis and Design (OOA&D) from Learning Tree International. This 1-week course was designed to "provide a thorough, practical knowledge of OOA&D methods." It was an academic presentation of the history and evolution of OO technology, an overview of terminology, and a discussion of a practical approach for actual systems development. Another team member attended a similar course offered by ObjectSpace Corp. This 1-week training session covered similar topics but used hands-on, practical exercises that helped the students apply the concepts they were learning. Clearly, this was a much more effective teaching method.

Two team members then attended 2-day training sessions from Rational Software titled Introduction to Rational Rose/C++ using UML. Rational Rose is the industry leader in computer-assisted OO modeling tools. Rose is an easy-to-use tool for creating and maintaining the various diagrams used during OO development. The tool then promises to build baseline C++ code (other languages are available) for the designed system. This code provides a good start for the actual programming of the final application.

Three team members also attended a 2-day workshop in developing "use cases" (see Chapter 3 for a more detailed explanation of use cases) offered by Advanced Systems Concepts. The workshop promised to provide extensive practice in developing use case descriptions and diagrams from given requirements. Unfortunately, the course spent too much time espousing the value of performing thorough requirements analysis and too little time working on exercises.

In addition to Rational, another vendor of leading-edge RAD tools and methods is Template Software, Inc. Several of the LMI staff member attended briefings at which Template explained their methods and described their software, which is called SNAP. They claim very impressive performance on software development projects and have the accompanying charts with statistics to support their claims. We were sufficiently intrigued to discuss with Template how we could include them in our task. Unfortunately, this discussion was not fruitful. The SNAP software is prohibitively expensive (if you are not buying it to develop a significant production system), and Template was not interested in providing an evaluation copy for our testing purposes. Nonetheless, this remains an exciting product and company; they merit consideration when looking for skilled RAD practitioners.

Selection of Target Application

Another aspect of the task was to select an "LMI process" that we thought would be a good test candidate for an RAD/OO development effort. We considered several criteria as we evaluated potential LMI target processes:

- *Internal or external process.* We favored an internal process that occurred entirely within LMI. Since this would in many ways constitute an experiment with uncertain results, there might be disadvantages to including non-LMI participants.
- *Large "enterprise" project or smaller group-level project.* An organization's first RAD/OO endeavor should not be a highly visible or mission critical project. We decided to seek a smaller project whose initial scope could be limited to the program director group level.
- *Value to LMI.* Although avoiding mission critical projects, we did not want to select a job with no real value. We sought a project where there was a recognized and important need, but which had not yet been automated (effectively).

Our selection for a "good fit" process improvement application was (what we called) the Staff Resource Tracking Tool (STRTT). The primary purpose of this application is tracking the *projected* workload, by task, of the staff members so that managers (program directors, program managers, project leaders) can determine staff availability and potential over- or understaffing problems. (Note the emphasis on the word "projected.") This application is needed to complement existing Institute applications (e.g., the task status report and the monthly task summary) that detail *actual* accumulation of effort and cost. The following chapter describes our selected application in more detail.

Survey of Potential Users

We surveyed users (two program directors and two project leaders) to help determine specific requirements of the STRTT application. The area of focus was on LMI managers' requirements to assess staff availability and anticipated task manning. The questions included the following:

- What management issues are not adequately addressed by existing tools or other resources?
- How do they solve this problem now?
- What do they envision as a desirable solution?

- Who are the potential users of the new application?
- Is there a need to exchange data with other applications?

Information from the surveys was then combined and evaluated. The results constitute the problem description, which is the starting point for the application design described in the next chapter. The primary requirement of the potential application is to *project, reserve, and track resources related to tasks.* The task-related resources to be managed are the staff members and the work skills they possess. The need to project, reserve, and track those resources implies participation by the staff members, project leaders, and program directors. The objective of the application is to provide an accurate picture of the resources (i.e., people and skills) currently assigned to tasks and their projected availability for assignment to future tasks. Such a capability is currently not available to program directors, program managers, or project leaders. Consequently, management of human resources is primarily extemporaneous.

Chapter 3
STRTT Application Development

PROBLEM DEFINITION

In the STRTT, we sought to develop an automated management tool that can be used to project, reserve, and track resources related to tasks. The primary components of this tool are people, tasks, and organizational policies and procedures. The specific requirements for the STRTT, all of which the initial versions may not address, include the following:

- People
  - tracking availability of time for assignment to tasks or in projecting potential tasks,
  - tracking professional development and skills,
  - tracking requirements for specific skills,
  - projecting estimated and actual percentages of commitment to tasks, and
  - projecting staffing needs for adding personnel.
- Tasks
  - facilitating task planning by linking the project plan to people;
  - tracking task progress, comparing actual values to projections;
  - showing who is working on a given task and how much time they have committed;
  - conducting "what if" scenarios;
  - tracking task funding at both the task level and various levels of consolidation;
  - conducting trend analysis of skills required, personnel, and funding; and
  - conducting risk analysis as an integral part of task planning.

- Organizational
    - integrating with time-keeping software,
    - integrating with task approval,
    - integrating with the task review process,
    - integrating with human resources and training to project staff requirements for new hires and training,
    - showing commitments versus projections, and
    - integrating with program tracking and assessment.

ANALYSIS

Analysis Methodology

The object-oriented analysis methodology selected for the STRTT application was the Use Case/Requirements Model. In this model

- high-level use cases are developed to identify functional areas of the system,
- a use case diagram is used to portray the system,
- expanded use cases are developed to show detail and the order of events,
- sequence diagrams are used to display process flow, and
- a class diagram is developed to show identified objects and their relationships.

HIGH-LEVEL USE CASE

High-level use cases briefly describe the major process of the system and are identified in the initial scoping efforts. They assist in partitioning the major functional areas of the system.

USE CASE DIAGRAM

The use case diagram is a graphic portrayal of the system and its external stimuli. The diagram displays the system or application, its actors, and the use cases developed for the system as well as the interaction between them.

EXPANDED USE CASE

The expanded use case is a narrative method of describing the system or process. It normally contains the functionality of the system or application in response to a stimulus from an external source. These use cases provide a method to capture requirements, communicate to domain experts and users (from the perspective of a systems designer), and develop a testing mechanism for the system or application.

SEQUENCE/SCENARIO DIAGRAM

The sequence diagram is a design technique that begins to map the process described in the narrative use case to object classes and their components. Sequence diagrams are often used to depict use case scenarios. An example of a primary scenario is one in which a specific process is performed as planned with no exceptions. A secondary scenario contains exceptions to the primary scenario. It is in the sequence diagram that objects are specified and their state, behavior, and identity are described. It is important to note that in object-oriented analysis the dimension of decomposition is by things or concepts (objects and classes) versus the traditional structured analysis dimension of decomposition by processes or functions.

CLASS DIAGRAM

The class diagram is a logical view of the packages and classes. For a fully developed system, there are normally many class diagrams. The primary or main class diagram is a logical view of the high-level packages.

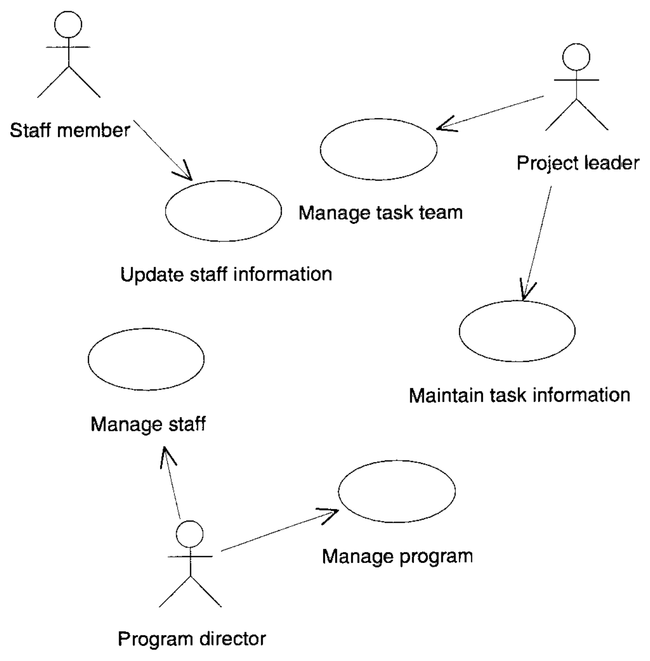
USE OF OBJECT-ORIENTED TOOLS

We used the OO tool, Rational Rose, to capture information about our design. All use case descriptions, use case diagrams, sequence diagrams, and class diagrams presented in this chapter were generated by the Rational Rose tool.

STRTT Primary Actors and High-Level Use Cases

The analysis of requirements of the STRTT identified three primary actors and five high-level primary use cases. The relationship of the use cases to actors is depicted in Figure 3-1. There are opportunities for additional use cases to expand the capabilities of the system. The scope of the first iteration of the project has been limited to ensure simplicity.

Figure 3-1. Use Cases and Actors



The use cases and actors are as follows:

- *Update staff information (staff member).* This use case begins when the staff member obtains a new skill level or task assignment. The information is created, reviewed, modified, or deleted related to staff member time availability and skill sets.
- *Maintain task team (project leader, initiator; staff member).* This use case begins when the project leader has a task order and allows creation, review, selection, modification, and deletion of team members.
- *Maintain task information (project leader, initiator; program director).* This use case begins when the project leader has initiated a draft task order. It provides the capability to create, review, modify, and delete task order information.
- *Manage staff (program director, initiator; staff member).* This use case begins when the program director has a requirement from a task order to provide staff resources. It provides the capability to create, review, modify, and delete staff members. The program director may also review assigned task workloads and skills. Project leaders and staff members update information related to skills and task assignment.
- *Manage program (program director, initiator).* This use case begins when the program director reviews summary information about all tasks

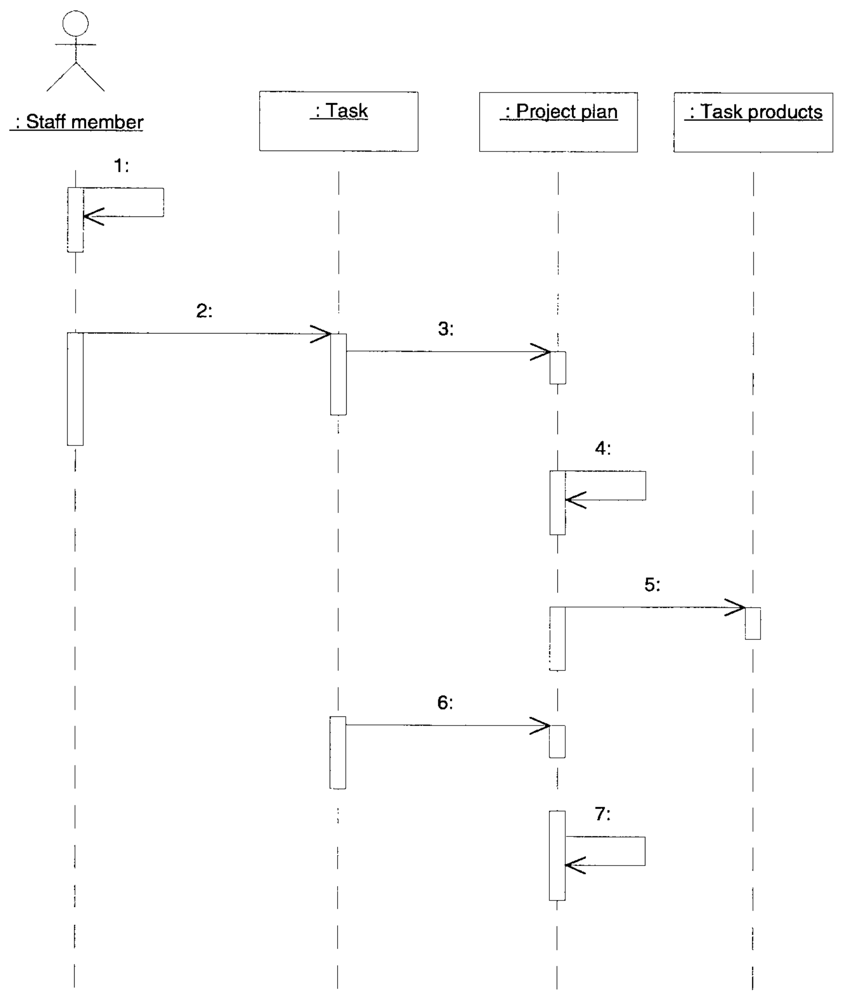
assigned to the group. It provides the capability to create, review, modify, and delete task order information related to the group's program. This information includes funding, task progress, staff resources, and skills.

Expanded Use Cases

UPDATE STAFF INFORMATION

Description: This use case (Figure 3-2) begins when the staff member obtains a new skill level or a task assignment. The information is created, reviewed, modified, or deleted related to staff member time availability and skill sets.

Figure 3-2. Update Staff Information



Actors: Staff member

Type: Primary, analysis oriented

Preconditions: Staff member is registered; all tasks are registered; staff member has accepted a new task assignment.

Basic Course of Events:

1. This use case begins when the staff member accesses the STRTT and selects him/herself for edit.
2. The staff member elects to update a task.
3. The staff member reviews current task and time commitments.
4. The staff member selects the task and projects his/her time commitment in hours per week of anticipated participation on a task.
5. The weekly percentage of staff member time commitment is calculated from the sum of all tasks associated with the staff member for all weeks that have been recorded.
6. A visual display of percentage of time commitment is generated depicting level of weekly workload and associated tasks for confirmation
7. The staff member exits the system.

Postconditions: The time commitment has been recorded and updated.

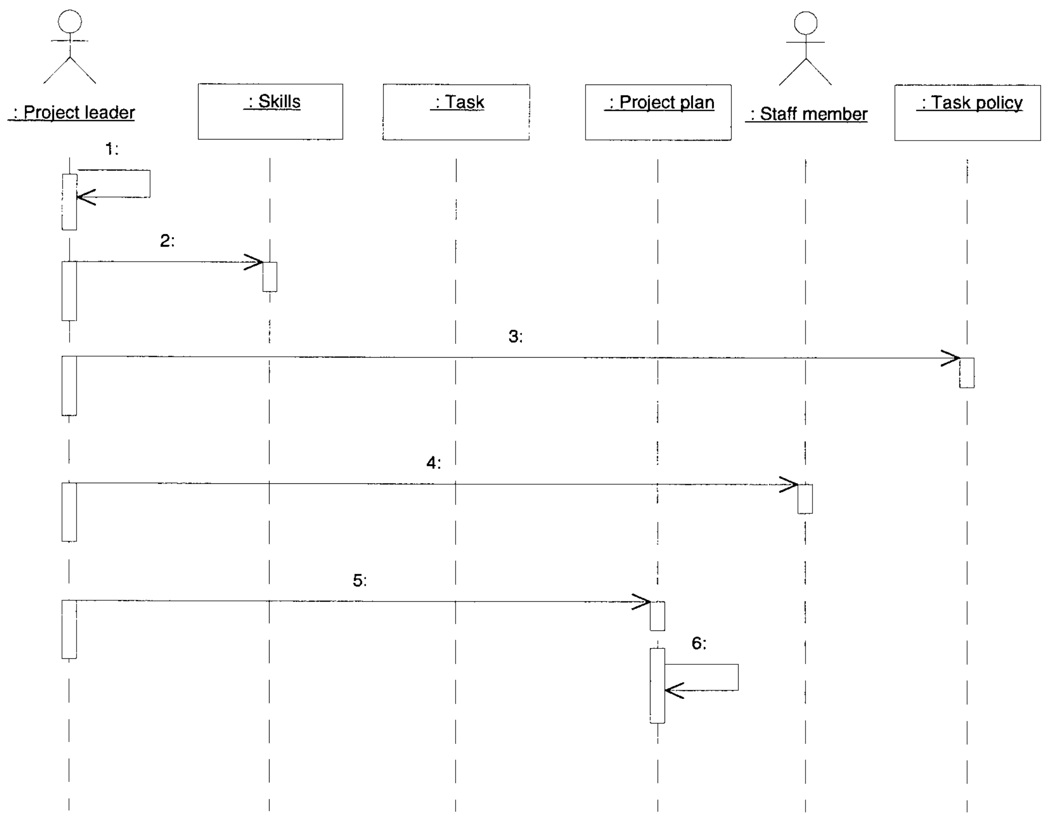
Alternatives:

- Alternative at 2: If the staff member is not registered, do not allow editing.
- Alternative at 6: If the staff member time commitment exceeds 100 percent for any week, notify of conflict. Allow entry.
- Alternative at 7: Allow print report of visual display.

MANAGE TASK TEAM

Description: This use case (Figure 3-3) begins when the project leader has initiated a draft task order and needs to get resources. It provides the capability to create, review, modify, and delete task order information. This use case depends upon the maintain task information use case.

Figure 3-3. Manage Task Team



Actors: Project leader (initiator), staff member

Type: Primary, analysis oriented

Preconditions: A current task order exists in the system.

Basic Course of Events:

1. This use case begins when the project leader has a task order that he/she needs to staff and accesses the STRTT.

2. The project leader selects the skills required. From the skills required, the project leader reviews the staff member time commitment to assess availability of staff members.

3. The project leader reserves staff members whose skill sets are required and who are available.

4. The project leader reviews the reserved task team members for the task in a visual display.

5. The project leader confirms the reservation and records it in the system.

6. The project leader exits the system.

Postconditions: The task team reservation has been recorded and updated.

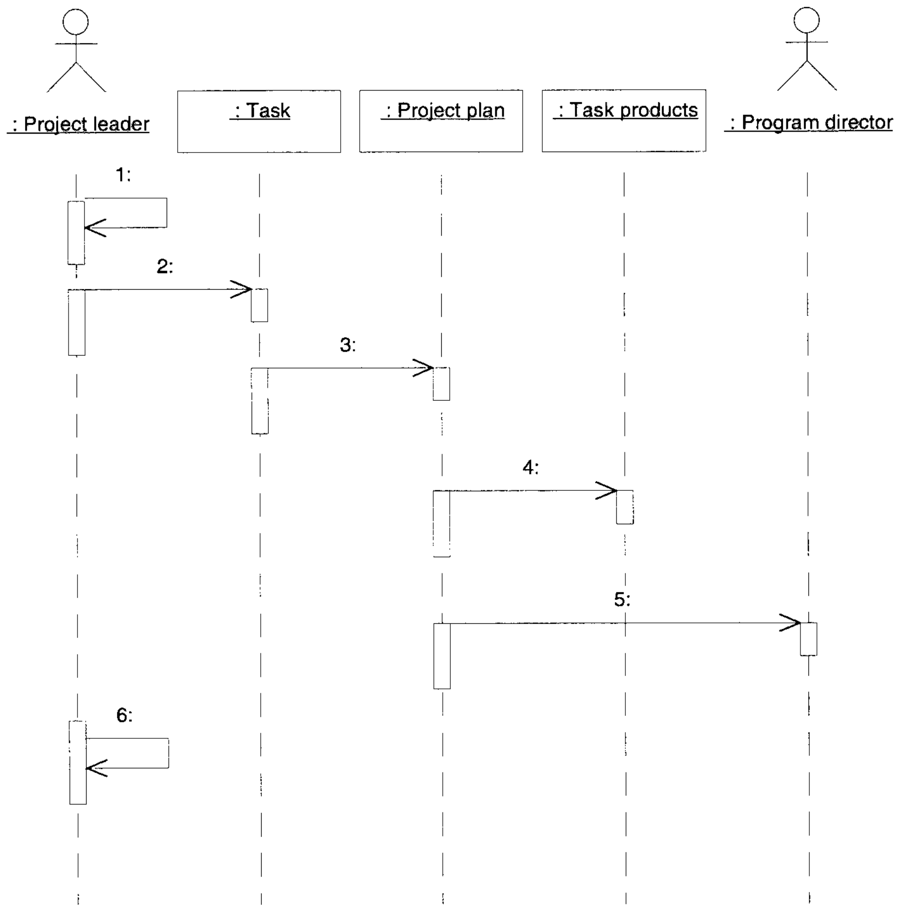
Alternatives:

- Alternative at 3: If staff member's time is fully committed, allow for a conditional reservation of time.
- Alternative at 5: Allow print report of visual display.

MAINTAIN TASK INFORMATION

Description: This use case (Figure 3-4) begins when the project leader has a draft or finalized task order and needs to develop a project plan. It provides the capability to create, review, modify, and delete task order information.

Figure 3-4. Maintain Task Information



Actors: Project leader (initiator), program director

Type: Primary, analysis oriented

Preconditions: Individual has been assigned as project leader.

Basic Course of Events:

1. This use case begins when the project leader has a draft or finalized task order and he/she needs to develop a project plan and accesses the STRTT and elects to create a new task.

2. The project leader creates a new task.

3. The project leader creates a project plan and develops tasks, durations, and milestones.

4. The project leader reviews the task order project plan in a visual display.

5. The project leader reviews the project plan, saves it, and submits it to the program director for approval.

6. The project leader exits the system.

Postconditions: The task order has been recorded and the program director notified for approval.

Alternatives:

- ◆ Alternative at 1: Proposed or projected task orders may be entered under a "proposed" status.

- ◆ Alternative at 4: Allow print report of visual display.

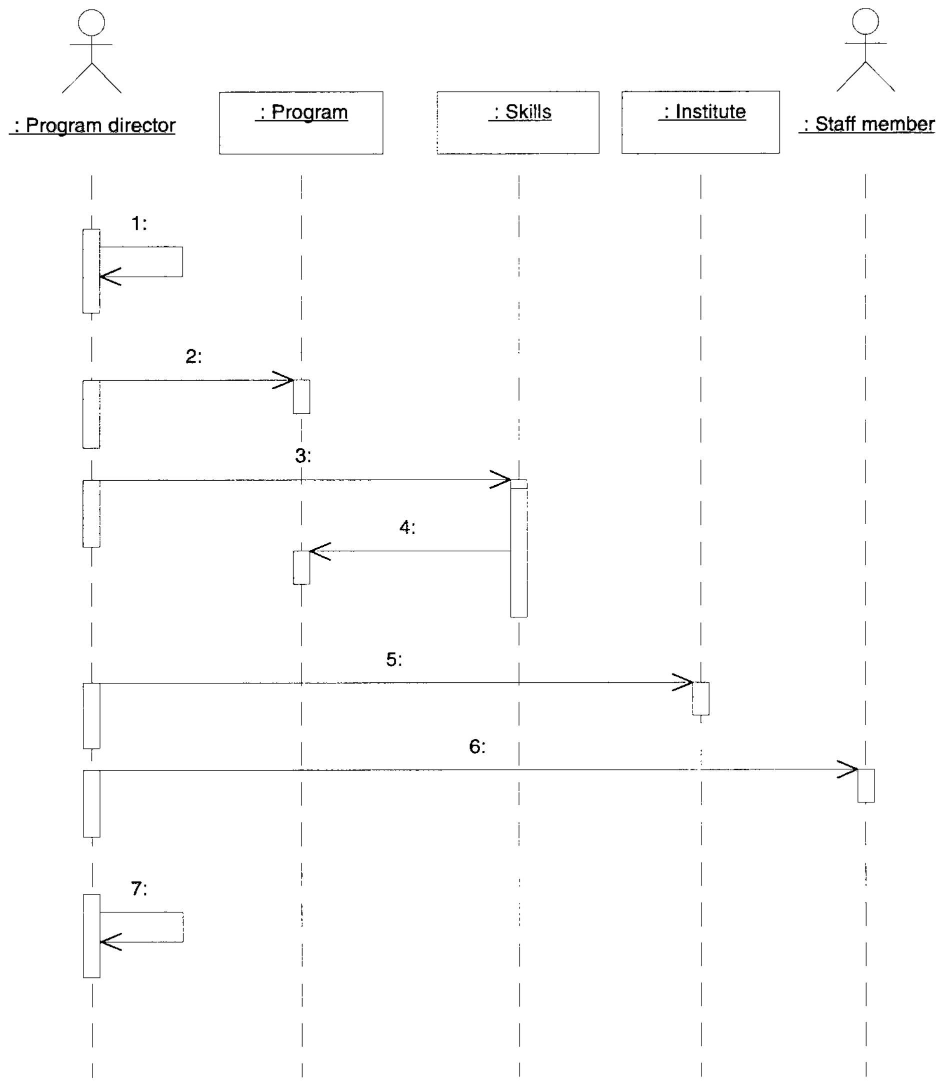
MANAGE STAFF

Description: This use case (Figure 3-5) begins when the program director has a requirement from a task order to provide staff resources. It provides the capability to create, review, modify, and delete staff members. The program director may also review assigned task workloads and skills. This use case depends upon the update staff information use case and the maintain task information use case.

Actors: Program director (initiator), staff member

Type: Primary, analysis oriented

Figure 3-5. Manage Staff



Preconditions: Staff information is current and updated; task order information is current and updated.

Basic Course of Events:

1. This use case begins when the program director has current and projected task orders within his/her program group and needs to get resources.

2. The program director accesses the STRTT and elects to review projected staff time commitment.

3. The program director identifies any under- or over-committed staff members in a consolidated visual display.

4. The program director elects to review staff skills.

5. The program director identifies any under- or over-committed skills and usage trends in a consolidated visual display.

6. The program director adds, modifies, or deletes staff members.

7. The program director exits the system.

Postconditions: New staff members have been added or departing staff members have been deleted.

Alternatives:

- ◆ Alternative at 3: The program director identifies any under- or over-committed staff members individually in a visual display.
- ◆ Alternative at 4: The program director identifies any under- or over-committed skills and usage trends by specific skill set and experience level.

MANAGE PROGRAM

Description: This use case (Figure 3-6) begins when the program director reviews summary information about all tasks assigned to the group. It provides the capability to create, review, modify, and delete task order information related to the group's program. This information includes funding, task progress, staff resources, and skills. This use case depends upon the update staff information use case, the maintain task information use case, and the manage staff use case.

Actors: Program director (initiator)

Type: Primary, analysis oriented

Preconditions: Staff information is current and updated; task order information is current and updated.

Basic Course of Events:

1. This use case begins when the program director elects to review the tasks, both actual and projected, registered for his/her group and accesses the STRTT.

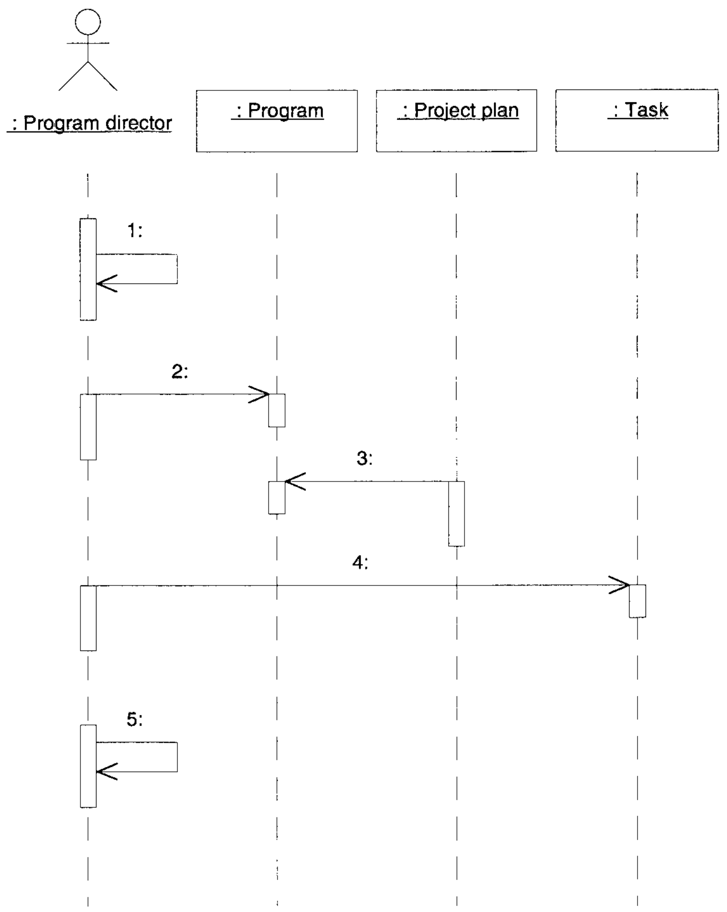
2. The program director elects to review all tasks within a program.

3. The program director selects a task-related project plan and reviews durations, milestones, deliveries, and actual-versus-projected expenditures in a visual display.

4. The program director selects summary view of task for projected and actual durations, projected and actual staff member requirements, and projected and actual expenditures in a visual display.

5. The program director exits the system.

Figure 3-6. Manage Program



Postconditions: Program has be reviewed and/or updated and reports generated if required.

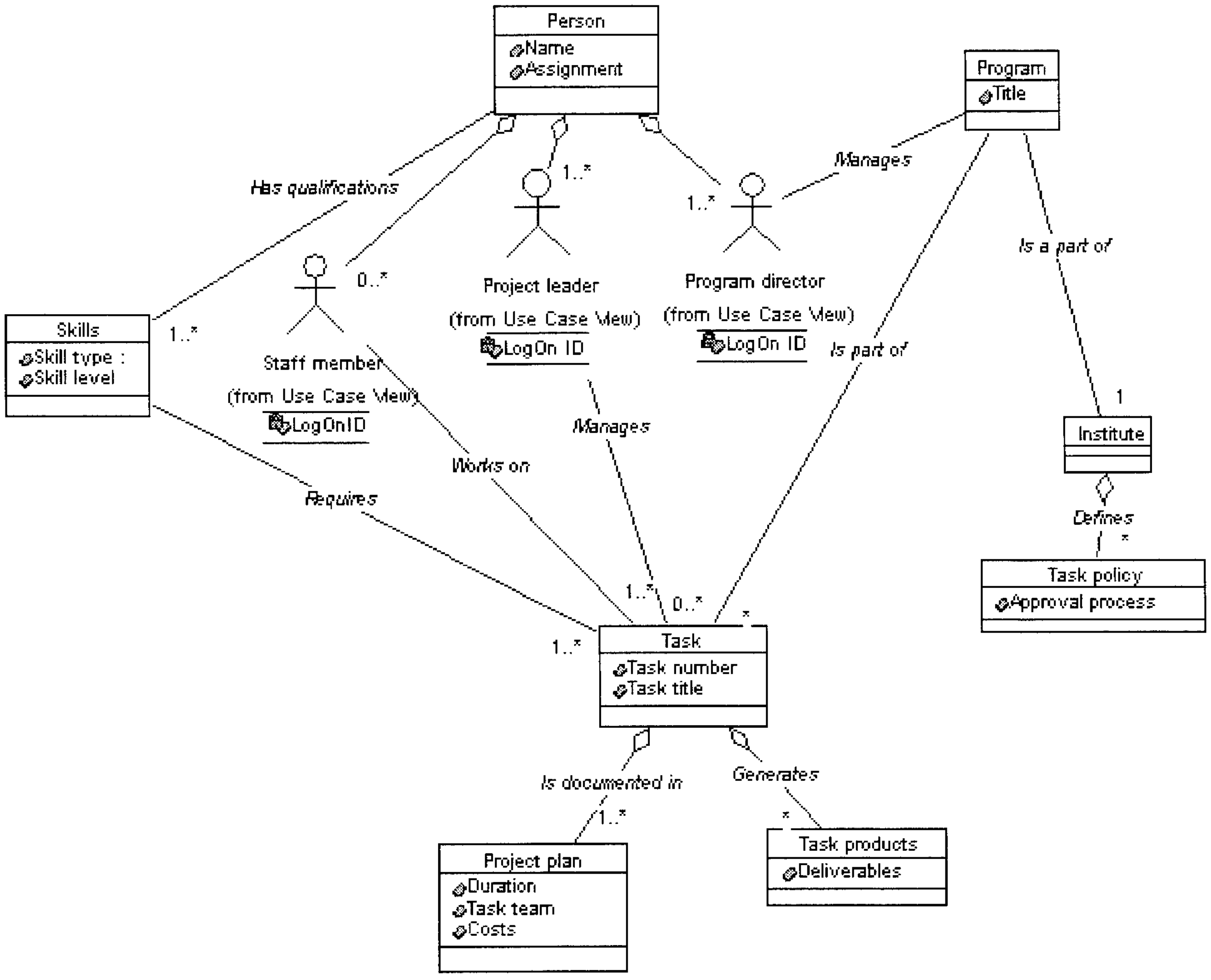
Alternatives:

- Alternative at 3: The program director elects to review tasks assigned to a specific project leader.
- Alternative at 4: The program director elects to review summary data, excluding projected tasks.
- Alternative at 4: The program director elects to review summary data, excluding actual tasks.

STRTT Class Analysis Diagram

The class analysis diagram (Figure 3-7) illustrates the emerging software architecture requirements. It can be generally equated to other activity modeling techniques such as IDEF0 [ICAM (Integrated Computer Aided Manufacturing) Definition 0]. The primary differentiation is that the model has meaning to both functional experts and information systems experts. This analysis diagram can be expanded into a design model, which will enable programming of a custom application.

Figure 3-7. Class Analysis Diagram



Note: 1..* and * are cardinality of relationship.

Chapter 4
Alternative Solutions

As we have described, the primary purpose of this project was to familiarize ourselves with and assess the potential value of using OO analysis and design techniques. In an effort to enhance the value of our efforts, we selected a pilot project that, if implemented, would be of value to the Institute. Having completed the analysis and design, we are now in a potential system development situation. Essentially, we have the typical "build or buy" decision to make. This chapter explores that decision and describes three basic approaches. For each alternative, we have included a high-level description and a discussion of the advantages and disadvantages.

BUILD—CUSTOM DEVELOPMENT

The OO design and templates described in Chapter 3 can provide the basis for a systems development effort. An OO-aware development tool, such as Rational Rose, could begin the effort by taking the completed design and generating code such as C++ or PowerBuilder from Powersoft. This code would be the foundation for the application, containing the appropriate physical implementation of the data model. Programmers could then design the user interface, including menus, screens, queries, and reports.

Custom development provides the most flexibility in providing a tailored solution. Design changes, such as screens and reports, can be easily accommodated. Logos, company-specific terms, titles, and formats can all be incorporated in a custom package. On the other hand, as with any custom development effort, this is probably the most expensive in terms of resources and risk. However, automated design tools like Rational Rose should reduce development time compared with structured techniques and facilitate iteration and design changes that will occur during the process. Maintenance costs should also be reduced, though they will still probably exceed those of the other two alternatives.

BUY—COTS PACKAGE

The second alternative is to locate a commercial off-the-shelf (COTS) package that satisfies all or most of our requirements. Using our application requirements and design, we would survey the market for existing software that meets our needs. If potential packages are identified, we would install, evaluate, and rate them against our design. We would further assess implementation issues, including integration with our existing systems and product supportability.

Using a COTS package has many advantages over a custom development. To begin with, product cost is usually substantially less. If there are a large number of users, the difference can be reduced, but site licenses and other blanket agreements still keep the cost down. Actual development problems are almost completely eliminated. Programmer expertise and availability is not a factor nor is maintenance of code. The disadvantages stem from trying to fit a standard package to our needs. The degree of customizing necessary depends on the variance between what is available and what is required. This will be determined during the package evaluation, but it is definitely a potential problem.

INTEGRATE—HYBRID SYSTEM

The third alternative involves combining various COTS components to produce an integrated solution. For example, a standard project management system that tracks project and manpower data may be integrated with a database application to perform specially designed data manipulations, with possibly a third product providing formatted reports and graphics. The basic idea is that no one COTS product would satisfy all of the requirements, but two or more could be seamlessly integrated to provide a total solution. In addition, some custom programming may be necessary to accommodate the integration, but far less programming than the amount involved in the custom development alternative.

The advantage of this approach is that it combines the time-saving and cost-reduction advantages of using COTS products—without being limited to the functionality of a single package—and increases the likelihood of meeting all requirements. The disadvantage of this approach is the broader product knowledge that would be required during development. The mechanics of package integration require more in-depth understanding than just package use. Open database connectivity, object linking and embedding, dynamic link libraries, and other integration tools add a layer of complexity that can often affect performance and reliability.

SUMMARY COMPARISON OF ALTERNATIVES

Table 4-1 summarizes the alternatives for system development.

RECOMMENDATION

Because the system requirements for this project do not appear to be significantly unique, the higher cost and risk associated with a custom development effort would not be justified. A single COTS package may provide all the functional requirements we have identified, but to implement such a package would require significant changes to LMI internal processes. The best solution appears to be an integration of COTS tools, which would give us the broadest capability with acceptable development time and risk.

Table 4-1. Alternative Solutions Matrix

Decision criteria	Custom development	COTS package	Hybrid system
Development costs	High	Low	Medium
Maintenance costs	High	Low	Low
Effort	High	Low	Medium
Estimated delivery time	6 months	2 months	3 months
Advantages	Tailored solution that meets all requirements	Low cost Ease of maintenance Proven product	Combines time-saving and cost-reduction advantages of the other alternatives
Disadvantages	High cost Programmer expertise affects product	May not meet all requirements	Potential system integration issues

REPORT DOCUMENTATION PAGE

Form Approved OPM No.0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE Sep 97	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Use of Rapid Application Development Techniques: Designing the Staff Resource Tracking Tool	5. FUNDING NUMBERS C PE 0902198D
6. AUTHOR(S) John B. Harris, Frank L. Eichorn, David L. Goodwin, Joel L. Henson	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Logistics Management Institute 2000 Corporate Ridge McLean, VA 22102-7805	8. PERFORMING ORGANIZATION REPORT NUMBER LMI- IR715T1
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Logistics Management Institute 2000 Corporate Ridge McLean, VA 22102-7805	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT A: Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Rapid application development techniques documents the learning process undertaken to achieve understanding and proficiency in the use of state-of-the-art object-oriented tools to produce viable software. An exploration of appropriate education and tools is undertaken and those experiences are used to conduct an analysis of requirements and initial design of an internal Logistics Management Institute program management tool. The report concludes with an assessment that RAD tools still need to mature; however, many components and processes related to object-oriented analysis and design are helpful in capturing and communicating requirements.

14. SUBJECT TERMS Rapid application development, software development, object-oriented (OO), unified modeling language (UML)	15. NUMBER OF PAGES 28
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL