ABSTRACT

PROCESS MODELING USING ProSLCSE ON A WEB-ENABLED PLATFORM

by
Orcan Ali Enunlu

Process modeling is a relatively complex task that needs to be addressed from a different point of view. The classical approach would be to design the model, to send it for evaluation, then to return feedback to the developing team, and to reevaluate the model with the feedback received from the parties involved.

However, it is our understanding that the steps taken during the process modeling could benefit from the advantages that the Internet offers. To demonstrate the usefulness of Internet in process modeling, I have taken an existing tool, ProSLCSE, and implemented it with Java so that it can run on a web-enabled environment. This Web-enabled version of ProSLCSE, also called ProWEB, will not only facilitate the implementation, controlling or standardization of the models, but also accelerate the task of modeling in an efficient and effective way. The developing team of the models would benefit from the tool in a real-time environment. Other parties, like the monitoring agencies, or controlling bodies would add their modification to the application in a sequential form.

The implementation of this Web-enabled process modeling will bring a new level of abstraction to the modeling and will minimize the difficulties due to geographical differences for 'time-depending' projects.

PROCESS MODELING USING ProSLCSE ON A WEB-ENABLED PLATFORM

by
Orcan Ali Enunlu

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer and Information Science

Department of Computer and Information Science

August 1998

APPROVAL PAGE

PROCESS MODELING USING ProSLCSE ON A WEB-ENABLED PLATFORM

Orcan Ali Enunlu

5/18/98

Dr. Murat M. Tanik, Thesis Advisor — Date
Department of Computer and Information Science
New Jersey Institute of Technology

5-18-98

Dr. Franz Kurfess, Committee Member — Date
Department of Computer and Information Science
New Jersey Institute of Technology

5/18/98

Dr. Ali H. Dogru, Committee Member — Date
Department of Computer and Information Science
New Jersey Institute of Technology

BIOGRAPHICAL SKETCH

Author:	Orcan A. Enunlu
Degree:	Master of Science
Date:	August 1998
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education:

- Master of Science in Computer and Information Science,
  New Jersey Institute of Technology, Newark, NJ, 1998

- Bachelor of Science in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996

Major: Computer and Information Science

To My Beloved Family and Friends

A new preacher was assigned to a town. Everybody gathered
around the town square and wondered
what the preacher was going to say.
The preacher came and asked townsfolk whether
they knew what he was going to talk about.
Since nobody had a clue, they all responded: "Noooo!"
"Well" the preacher said and added:
"Then, why did you bothered coming here."
One week passed, and people gathered again around the town
and decided to answer the preacher differently.
The preacher appeared again and asked whether they knew
what his topic was.
This time, everybody shouted "Yessss."
Cleverly, the preacher told them they didn't need to listen to his speech
Since they already knew what he was going to talk about.
He left the townsfolk again puzzled.
Another week passed.
The townspeople got very curious about what the preacher
had to say. So they decided that half of the people
would say, "Yes, we know what you are going to say" and
The other half would say, "No, we don't know."
The preacher came and asked one more time:
"Do you know what I am going to talk about?"
As they planned, half of the townsfolk replied "Yes", the other half "No."
Thinking that the preacher is cornered this time, they awaited his response.
Deviously, the preacher responded:
"Well, those who are informed will tell those who are ignorant."

Enjoy.

ACKNOWLEDGEMENT

I would like to express my deepest thanks to Dr. Murat M. Tanik, Dr. Ali H. Dogru, and Dr. Franz Kurfess, who provided me with moral, technical and financial support throughout my Master's degree. Those three semesters at NJIT wouldn't have been successful if they didn't support and encouraged me. Special thanks go to Ms. Sheridan Quarless, Ms. Jenice Sabb, and Mr. Michael Tress, for their help and support.

During those three semesters, I had the privilege of receiving two awards: One received through my assistantship at the Learning Center as Leadership Award. The other was received also in 1997 as the Co-op Student of the Year, from the CIS department. I would like to thank all the people involved in my success.

I have a feeling that there'll be more good news and hope to celebrate those special events with my advisors and loved ones.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this thesis is to analyze the usefulness of process modeling through the World Wide Web. Since the use of automation in factories started to take place, engineers have tried to model their work. The processing of that work plays an important factor in achieving their objectives in a cost-efficient, reliable and timely manner [Tanik, 93].

Processes have been the basic structure for developing large-scale systems. Taken from the manufacturing community, the concept of presenting the problems of large-scale systems, by means of processes, is a recently applied concept to the software industry, as well as to businesses and organizations. Although the use of processes for the manufacturing community is clearly defined, there is not much difference when it is applied to the software industry. The objective is the same, whether it is applied for an organization, a business or a software industry. The primary objective is to clearly define and present the structure of the system as made of processes, each linked to each other and thus contributing to the system. Processes help coordinate the people involved with the system. A clearly defined set of processes will not only ease the interaction of the people involved, such as the designer, the developer, the client, etc, but also track their progress more precisely for future references [Humphrey, 95].

Processes are present in every aspects of the development and the maintenance of the system. Specifically, processes outline the technical and the management framework for applying methods, tools, and people. They define the relationship of these entities to each other and to the whole organization. Tasks and roles are defined to establish the connecting glue between methods, tools and people. Furthermore, processes determine the inputs and the outputs at every major step, so that the design and the maintenance of the system are done in an orderly manner. This flexibility allows better methods to be incorporated into the system without affecting the whole organization.

Benefits of using processes in an organization are countless. They enable an effective communication between users, developers, customers, and researchers. They expand the management's understanding of the organization by presenting an overall view of the process automation. They provide a framework, allowing the management to measure the status of the system. The use of processes allows reusability, as well. By being reusable, development time is reduced and therefore being reusable minimizes the cost of implementing new features. By being detachable, more effective processes can easily be mounted, if necessary, in place of older processes. Thus, allowing the developers to save time and money during the process design [Kellner, 88].

The organizations that effectively communicate, evolve, and manage their processes are considered to be more efficient. Thus, defining processes are an important factor in an efficient organization. A well-defined process is an influential element in an efficient organization. Processes that are not well defined usually result in unstructured, faulty integration of processes.

Thus, modeling the processes becomes an important issue. Much of the research on process modeling has been focused on software development organizations, since the software engineering community has already experienced formal modeling. Software process modeling focuses solely on phenomena occurring during the software creation and evolution [Curtis, 92]. Usually, organizations, mostly software organizations, do depend on software life cycle description [Madhavji, 91]. Often these descriptions do not correspond to the processes performed during development and maintenance. Rather than organizing the necessary information, they represent high-level abstract plans. This lack of conformance between the actual processes and the organization's stated processes could be explained by the following reasons: high–level processes that do not correspond to the actual project activities. Imprecise, ambiguous, incomprehensible, or undefined processes that are performed in the project; and failure to update the documentation as the processes and the structure of the project changes [Curtis, 92].

The "large-grained" traditional life-cycle descriptions outline the different phases abstractly. These life-cycles models, such as the Waterfall Model or the Spiral Model, would describe the process at a high-level abstraction. They would lack details needed to successfully run the project [Lassenius, 97]. Traditionally, these representations have been considered process models. But, the software process modeling community feels that something else is needed, because the granularity of these models is too large [Curtis, 92]. These life-cycle models fail to show the building blocks necessary for developing and maintaining large projects.

New understanding and research for process modeling has evolved to satisfy the needs mentioned in the previous paragraph. In forming such model, one has to consider the following uses for process modeling: to facilitate the human understanding and communication, to support process improvement, to support process management, to automate process guidance and to automate execution support [Curtis, 92].

Most of Chapter 2 deals with processes and process modeling. There is an emphasis on process definition, since this is what this thesis is partially based on. Process modeling is thoroughly reviewed in the second chapter, as well. The objectives and perspectives of process modeling are outlined. Some of the issues concerning process modeling are discussed. Capability Maturity Model from SEI is discussed as an example of assessment tool of software processes. This area is tied to the process modeling, since it brings a tool that enables developers to assess their processes and to improve their organizational structure.

Chapter 2 is finished with a discussion on Workflow Systems, which is relatively similar system to process modeling. Workflow is often confused with process modeling. This section gives the reader a chance to compare those two models and see the similarities and the differences.

1.2 Current Research and Justification

One of the most difficult problems in process modeling is the inherent complexity of representing real world processes. During the last few years, much new formalism has been developed or is still being developed to model processes. Few of them completely represent what they are supposed to. Some approaches are good for describing processes and others are good at modeling them at low levels. The granularity of process modeling is an important issue to be considered. The need for a formalism that represents a high level modeling has led researchers to combine the good aspects of several formalisms.

1.3 Contributions

Our approach was to use an already established tool, Process Oriented Software Life Cycle Support Environment (ProSLCSE). ProSLCSE is a tool that facilitates improvement efforts in an organizational environment. According to the SEI, organizations must first define the steps they use to conduct business processes. These processes are then analyzed and molded in the organizational framework to improve the structure of the organization. ProSLCSE is a tool that helps users to define and control processes, therefore enabling the control and improvement of the overall project [ProSLCSE, 94]. ProSLCSE helps the user in defining the infrastructure of the organization and the processes used by the organization to perform those tasks, associating resource types to the process, planning the activities, analyzing the project through simulation and running the project.

All these functionality led us choose this tool and implement it on a web-enabled platform. Chapter 3 describes the ProSLCSE environment. ProSLCSE uses a structure-oriented graphical editor that is based on Visual Process Modeling Language (VPML). The constructs of VPML are used in two editors, the Infrastructure and Process editors, which are discussed in detail.

Chapter 4 describes the environment and the programming language that is used to implement ProSLCSE. The Internet has become the center of scientific and commercial world. It enables users vast opportunities. The number of users using this medium is growing exponentially [Hoffman, 96]. Everything has to be associated with Internet in order to remain alive in this competitive world. Internet offers a medium where process modeling could evolve. Developers in different locations could model their processes through our web version of the ProSLCSE, which we call ProWEB. They could design the models, make corrections, align old processes with new ones, and add new components. All those actions are possible in an almost real-time environment. That is why ProSLCSE was ported to World Wide Web (WWW).

Java represents another aspect of our attention. We would like to implement ProSLCSE on a web-enabled platform, whether it is Netscape, Internet Explorer (IE) or HotJava. Java seems to be the language evolving around those browsers. Java's main characteristics led us choose this programming language over others. Chapter 4 describes some of those characteristics. But today's Java is not enough. It has limitations that need to be solved in near future.

Chapter 5 concludes this thesis by describing the implementation of our thesis. Step by step approach was chosen to let users navigate through the project. Although we are new in Java programming, there are areas that needs improvement and elaboration. The project is not yet fully implemented. The areas that are open for improvement are outlined in the last sections of the chapter.

CHAPTER 2

PROCESS MODELING

2.1 Overview

2.1.1 What is a Process?

The Webster dictionary [Webster, 83] defines a process as a continuing development involving many changes and a particular method for doing something, usually involving a number of steps or operations. The IEEE's definition is similar and defines a process as a sequence of steps performed for a given purpose [IEEE, 90].

A software process is defined in Capability Maturity Model (CMM) by Software Engineering Institute (SEI) as a set of activities, methods, practices, and transformation that people use to develop and maintain software and its derived products [Paulk, 93].

2.1.2 Relation between Process and Product Quality

An engineering problem involves many issues. Cost, reliability and timeliness are three major factors that make up all the engineering issues [Tanik, 93]. For example, if an engineering problem is reliable as far as the quality and it is timely delivered, but cost ten times what it is supposed to, then this solution does not respond to the engineering problem. Therefore, it is necessary to balance between those factors, in order to establish a good solution to the engineering problem. Software engineering problems or any other engineering problems should follow this understanding. Also, processes do provide a solution to those problems by establishing time predictability, quality, and cost efficiency. But there are differences between a set of good and bad processes. Immature processes might conclude in cost and time schedule overruns, expensive rework and in the end become the source for project termination [Krasner, 92]. In contrast, a well-defined processes might provide quality, timely access to the market and customer satisfaction to both the developer and customer sides. A good process is usually required to produce good products, establishing a correlation between process quality and product quality as shown in Figure 2.1 [Lassenius, 97].

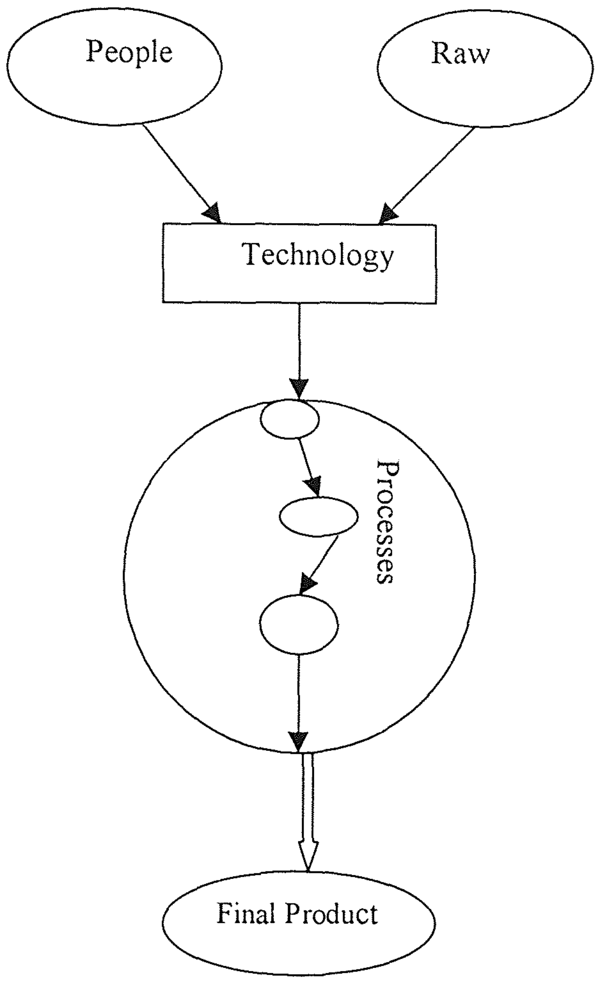


Figure 2-1. Process quality influences product quality

2.1.3 More about Process

A process has certain characteristics that enable developers and users to grade and to make a comparison between processes. Some of those characteristics could be summarized as the following [Lassenius, 97]:

- Understandability: Is the process defined and understandable?
- Visibility: How visible is the progress to outside?
- Reliability: Are there any process errors resulting in product defects?
- Robustness: How well the processes continue in case of unexpected problems?
- Maintainability: Can the process evolve to meet changing conditions and needs?
- Supportability: Can other tools support the process?
- Integration: How well is the process integrated to other processes?
- Acceptability: Is the process acceptable to those involved?
- Productivity: How fast can the system be produced?

2.1.3.1 Process Characteristics: All those questions should to be asked when the developer begins the process modeling. But considering those issues doesn't necessarily mean that we have an effective process in our hand. Bill Curtis thinks that an effective process should have the following attributes: It should be defined clearly to avoid any misunderstanding. It is advised to document the steps taken toward the use of such processes. It should be practiced and fully implemented to reach the objectives. An effective enforcement is necessary to avoid any deviation from the main objective. It should be measured throughout the project to control the outcome. It should be supported thoroughly. And most of all, the people involved in the project should be trained to increase the productivity [Curtis, 95], [Lassenius, 97].

2.1.3.2 Representations: There is a distinction made between process model and process guide. This distinction lies in the way things are represented to the outside world. Process guides are standard documentation describing the implementation of the software process. Those instructions are usually represented in text and graphic format. They are primarily used for planning and training purposes. On the other side, process models are detailed and formalized descriptions. They are content-rich and powerful notations to represent activities, products, resources, objects, transformations, events, etc. Those graphical representations ease the human understanding of the project by giving a detailed view of the processes. They are primarily used for development, planning and improvement purposes [Armitage, 94], [Lassenius, 97].

2.1.3.3 Current Issues in Software Community

- **Problems with Granularity**

Traditional life-cycle models like the "waterfall model"; the "spiral model" or "iterative enhancement" depicts different phases of a project abstractly. These models have been considered process models and much of the software process thinking is still based on the waterfall framework [Humphrey, 89]. But, the granularity of those models is too large to describe many elemental process blocks necessary to manage and coordinate the project [Curtis, 92]. They lack the details to support process optimization. They do not respond adequately to the changes in the software development. They usually follow a relatively uniform and orderly sequenced set of activities. They fall short in accommodating to recent developments as rapid prototyping or advanced languages [Humphrey, 89].

Reliance on the waterfall model has limited some of the design improvements. By describing the process as the sequence of requirements, design, implementation, and test, where each step depends on the previous one, has caused the phases to be dependent on each other. The Figure 2-2 shows the standard Waterfall life-cycle model. Nevertheless, in reality, the requirement is a continuing phase that must be constantly updated through development phase. Other relevant tradeoffs exist in the following phases [Humphrey, 89].

- **Formality**

The level of mathematical formality used in a process modeling language is another primary concern in the research community. The level of a language's formal precision is a determinant factor in representing the process. This formality level depends also on the purpose served by the process model and the agent using it. A process program enacted by a machine should be more formal than a process enacted by humans [Curtis, 92].

- **Fitness**

Process modelers would like also to differentiate how processes act on different situations. Depending on a particular situation, they would like to separate process models into three categories: prescriptive, descriptive, and proscriptive modeling. Prescriptive modeling implies that the process should be applied in a certain way. A process modeling that conforms to a certain level of performance is said to be highly fit to the organization. A descriptive modeling approach, however, is focused on determining the actual processes currently used in an organization to get work done. Those types of processes are said to form the organization's process baseline. The third perspective, the proscriptive modeling deals with behaviors that are not allowed in an organization. That approach tells process modelers the presence of constraints that cannot be violated and should be considered with moderate attention [Curtis, 92].

In view of those limitations mentioned above, the software process modeling community turned to other process modeling descriptions that will satisfy their needs.

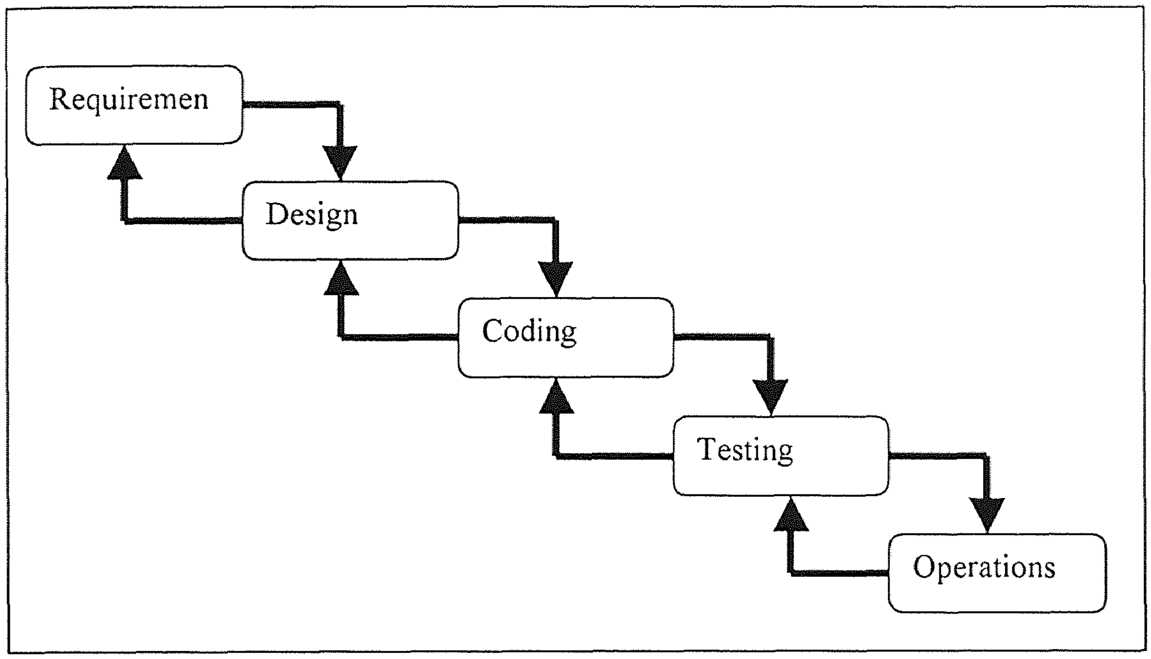


Figure 2-2. Standard Waterfall Life Cycle Model [Davis, 93]

2.1.4 Process Modeling

2.1.4.1 Objectives: New research for process modeling has concentrated in new areas to satisfy those needs mentioned above. In forming such model, one has to consider the following objectives to create a new approach to process modeling: Facilitate the human understanding and communication, support process improvement, support process management, automate process guidance and automate execution support [Curtis, 92].

By enabling effective communications among process users, developers, managers, or researchers, a precise basis for process execution and automation is developed. This led to an increase in human understanding of the processes. This type of use requires that a group share a common ground in representing processes.

A support for process reuse and improvement is necessary to avoid repetitions. This requires a basis for defining and analyzing processes. Process development is time-consuming and expensive. This reason alone causes project team to save time by reusing and by improving their old processes. The process models should be used to facilitate process management, as well. Effective management requires a clear understanding of plans and a defined process against which actual project behaviors can be compared to evaluate the status of the project [Humphrey, 89], [Curtis, 92].

From those objectives, it is clear that process models should represent the way work is performed to achieve the goals. They should provide a clear and a flexible framework for representing and improving process models, and yet be granular to the desired level of details. Table 2-1 summarizes the software process modeling objectives and goals [Curtis, 92].

Table 2-1. Software process modeling objectives and goals [Curtis, 92]

Facilitate human understanding and communication between users • Represent process in an understandable form • Facilitate communication among process users • Formalize the process to let people work more efficiently • Provide the necessary information to allow an individual or team to perform the intended objectives • Form a basis for training the intended process
Support process improvement • Identify the necessary components of a high-yield software development or maintenance process • Reuse process in the future • Compare alternative process to determine the effect on the overall project • Estimate the consequences of potential changes to a software process before putting them into practice • Assist in the selection of tools that will be used with the processes • Facilitate organizational learning on implementing effective processes • Support managed evolution of a process
Support process management • Develop a project-specific software process to accommodate the attributes of a particular project • Outline the reasons for the creation and evolution of those software attributes • Support development of plans for forecasting purposes • Monitor, manage and coordinate the processes • Integrate a comparison basis for process evaluation and measurement
Automate guidance in process performance • Define an effective software development environment • Provide users guidance, suggestions and material to increase the performance of the processes by increasing the human understanding • Save process representation for future use in a repository
Automate execution support • Automate parts or portions of the process • Encourage cooperative work among individual users or teams by automating process • Automate the collection of measurement data • Enforce rules to guarantee process integrity

2.1.4.2 Process Modeling Perspectives: There is lots of information that users want to extract from process models. Some of the questions, that they want answers to, are listed below [Curtis, 92]:

1. What is going to be accomplished?
2. Who is going to participate in this work?
3. How will the work be partitioned among participants?
4. When will it be done?
5. Where will it take place?
6. How will it be done?
7. Why does it need to be done?
8. Who and what are going to be affected by its completion?

The reason why process-modeling languages differs from each other lies in the way those questions are answered. Since those questions can be viewed and handled in different ways, different languages provide a solution to particular set of questions.
Process modeling languages and representations usually provide a solution to one or more different perspectives related to these questions. Four of the most commonly presented perspectives are functional, behavioral, organizational, and informational perspectives [Curtis, 92]. Figure 2-3 shows how those perspectives are interrelated with each other.

Functional perspective might relate to the first question. 'What is done?' 'What is accomplished?' are the questions that find answers if someone implements the functional perspective. The process elements that are being performed and the informational entities (e.g., artifacts, products, and data) that are being included are investigated through this perspective.

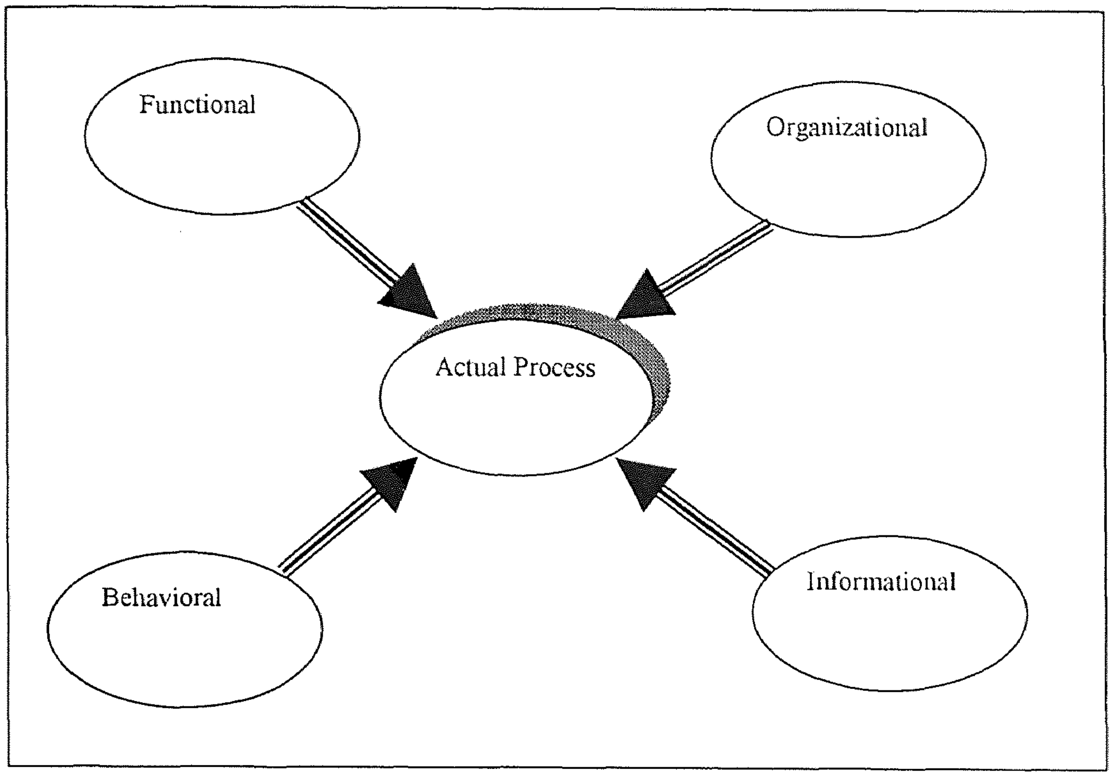


Figure 2-3. Process perspectives [Curtis, 92]

Behavioral perspective represents the actions with respects to time. 'When things are done?' 'When the processes are performed?' 'How are they performed?' are questions that are studied when behavioral perspective is taken into consideration.

Organizational perspective responds to questions like 'Where is the process taken place?' and ' Who is involved in the accomplishment of the process?' The agents involved in the process and the location, where those processes have taken place, are the highlighting aspects of the organizational perspective.

Informational perspective deals more with informational entities manipulated or produced by a process. These entities might be data, artifacts, intermediate or end products, and objects. This perspective considers the relationship among those entities, as well as their structure within the process.

To have all those perspectives in consideration during the development of process modeling will result in a better process. It is estimated that the combination of these perspectives will lead to an integrated, consistent and complete model of the process in hand. However, in real life, known languages and representations fail to support all perspectives adequately. In practice, most process descriptions have used narrative text and diagrams to express process. These types of representations were bounded by the constructs of the language used for modeling [Curtis, 92].

2.1.4.3 Process Modeling Paradigms: Most researchers have viewed the development of software as the first step in developing a process model. Thus, the actual development of a process has been compared to the development of software. From this perspective, the concept of process programming has emerged and is defined as "the activity of expressing software process descriptions with the aid of programming concept" [Tanik, 93]. The researchers who followed this trend were faced with the famous chicken-and-egg problem [Osterweil, 87]. As Osterweil pointed out: "In order to find out what language features we need, we need to write process diagrams; in order to write process programs, we need the appropriate language features." Most languages used by the software community have been the basis for modeling software processes, and most of the process modeling languages derived from those languages [Curtis, 92]. Some of those language types are procedural programming language, system analysis and design, state transition and Petri-nets, functional languages, formal languages, data modeling, etc. Table 2-2 outlines some of those languages and their respective use in the software industry.

Table 2-2. Language types and constructs used in software process modeling [Curtis, 92].

Base Language Types and Constructs	**Sample Software Process Modeling**
Procedural programming languages	APPL/A
System analysis and design	STATEMATE
AI languages and approaches	AP5, GRAPPLE,
Events and triggers	AP5, APPL/A, STATEMATE
State transition and petri-nets	Role interaction Nets, STATEMATE
Control flow	MVP
Functional languages	HFSP
Formal languages	Context-Free Grammar
Data modeling	APPL/A, PMDB, STATEMATE
Object modeling	AP5, MARVEL, MVP
Precedence networks	SPMS
Quantitative modeling	System Dynamics

2.2 Current Capability Models

There are several different diagnostic tools, which are used in the software community to assess software processes of an organization. The assessment and the evaluation of software processes have become a major concern worldwide. A software process assessment is an appraisal by a trained team of software professionals to determine the state of an organization's current software process, to determine the high-priority software process-related issues facing an organization, and to obtain the organizational support for software process improvement. A software capability evaluation is an appraisal by a trained team of professionals to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing software effort [Paulk, 93].

Organizations would like to know if their products meet the standards set by their respective community. The software community relies on different assessment tools. The Capability Maturity Model (CMM) is the popular one right now. Other maturity models are being developed. Some of the current ones are ISO SPICE, ISO-9000, Trillium, Bootstrap [Lassenius, 97]. Since CMM is becoming a widely approved tool, more detail is given in the following section.

2.2.1 Capability Maturity Model

The CMM was originally developed to assist the U.S. Department of Defense (DoD) in software acquisition. The main purpose was to evaluate the contractor's software in order to make a decision. DoD contractors quickly learned that they needed to change and guide their organizations to become more aligned with the CMM. Software Engineering Institute (SEI) saw the benefits of such framework and recommended a broader participation in the development and improvement of the CMM. Hence, the model gained visibility in the software engineering community. Soon enough, commercial organizations adopted the CMM for their own improvement. As of 1993, commercial organization have performed more assessments than all DoD and other Federal contractors [Herbsleb, 97].

The Capability Maturity Model for Software provides software organizations with guidance on how to gain control of their processes for developing and maintaining software. The CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement. By focusing on a limited set of activities and working aggressively to achieve them, an organization can steadily improve its organization-wide software process to enable continuous and lasting gains in software process capability [Paulk, 93].

Continuous process improvement is based on many small, evolutionary steps rather than revolutionary innovations. The CMM provides a framework for organizing these evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement. These five maturity levels define a scale for measuring the maturity of an organization's software process and for evaluating its software process capability. These levels also help an organization prioritize its improvement efforts.

Each level comprises a set of process goals that, when satisfied, stabilize an important component of the software process. Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization.

Organizing the CMM into the five levels, as shown in Figure 2-8, helps increase achieving software process maturity. The labeled arrows in Figure 2-4 indicate the type of process capability being institutionalized by the organization at each step of the maturity framework.

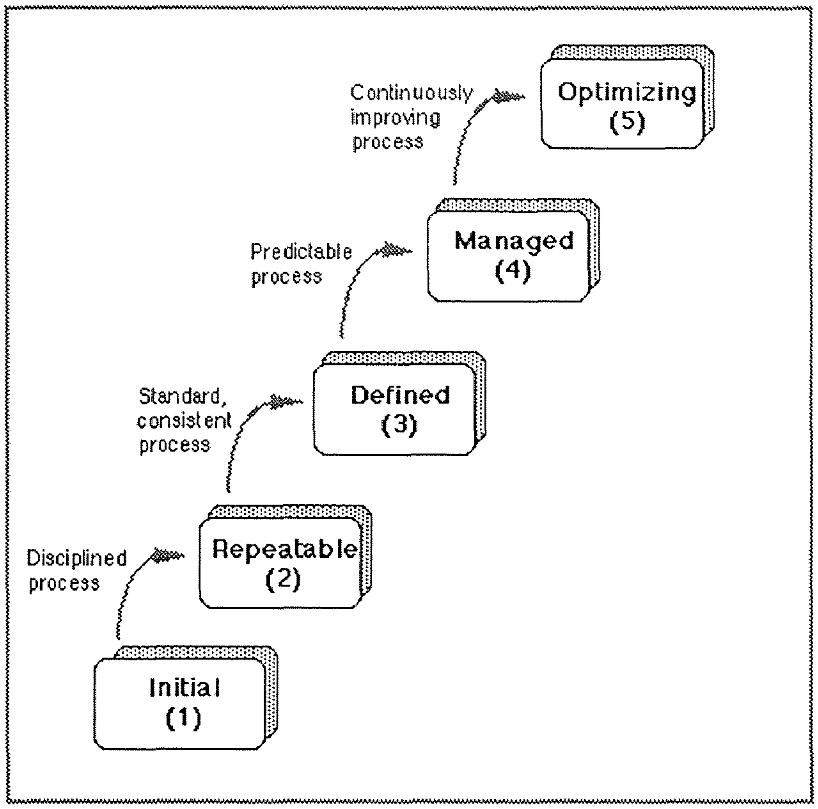


Figure 2-4. The five levels of software process maturity [Paulk, 93]

2.3 Workflow

Workflow management is a fast evolving technology being used in a variety of industries. Workflow can be summarized as the automation of processes involving human and machine-based activities, as well as the interaction of Information Technology (IT) applications and tools with users. This type of technology is mostly used within the office environment in staff intensive operations such as banking, insurance, legal administration, and its application to other areas is also expanding. It became useful for industrial and manufacturing applications, as well. Although the number of users of workflow management is on the rise, there isn't any standardization in this new area. The lack of conformity between workflow products limits the integration of workflow systems and results in incompatible process automation.

There are some encouraging works done to remedy this situation. New research groups have come together to set uniformity within this community. It has been recognized that all workflow management products have some common characteristics. This enables some level of interoperability between products. The Workflow Management Coalition (WFM Coalition) is one of those groups working to find common grounds between product specifications and to improve integration of workflow applications with other IT services. An improved and effective use of workflow technology within the IT is believed to be beneficial to both vendors and users [Hollingsworth, 94].

2.3.1 Historical Facts

In the early years of 19th century, Frederick Taylor outlined the principles of "scientific management" [Winograd, 87], a view that considers work as motions and activities of workers that can be planned and organized to produce the maximum output. But the introduction of computers, hence automation shifted the definition of work to a new direction. The individual worker became an information processor, modeled as a function that processes input information into output information. According to Denning, another pioneer in this area, Herbert Simon brought a new interpretation to work [Denning, 95]. He described IT as a source of providing information to the worker to make better decisions. In all those perspectives, work is seen as the process of transforming the given inputs (information or product) into the desired outputs (processed information and products).

2.3.2 Definition of Workflow

Workflow is the automation of procedures, where information or tasks are passed between users, according to a predefined set of rules, to achieve an organizational goal. While workflow may be manually organized, it is mostly organized within the context of an IT system. This enables the system users with computerized support for the procedural automation. Often, workflow is associated with Business Process Re-engineering, which deals more with the assessment, analysis, modeling, definition and other operational implementation of business processes of an organization. This similarity between business process modeling and workflow makes it harder to distinguish those two business-modeling concepts. What makes workflow technology different is that it provides separation of the business procedural rules and its IT oriented operational support. Thus, it enables subsequent changes to be incorporated into the procedural rules defining the business process.

2.3.3 Organizational Processes

There are three distinguished domains to describe the activities within an organization: Material, Information and Business processes [Medina, 92]. Those three different types of organizational processes follow the historical definition of work mentioned in Section 2.3.1.

2.3.3.1 Material Processes: This is the standard interpretation of work. Every action is tied to a physical movement or a change of state. In its traditional context, factory automation, physical components are transformed and assembled in products. Material process redesign, along with the application of new technologies, has been used to provide more efficient outputs [Medina, 92].

2.3.3.2 Information Processes: With the twentieth-century shift from traditional work concept to information age, the material process domain fails to capture the important elements of today's work. With computers all around the organizations, work became indistinguishable. Theorists and IT providers have developed new and improved ways to analyze and facilitate the flow of information in an organization. Until recently, one of the primary objectives of organizations was to identify, study, and optimize their informational processes [Denning, 95]. Nowadays, much of the work is concerned with how information is passed around the organization, how effective the current techniques of data flow, database storage and retrieval, transaction processing, network communication are [Medina, 92].

2.3.3.3 Business Processes: Business process is the next step after the information domain. Recently, organizations have realized that the information technology does not address their crucial concerns. The information is considered useless if it does not generate an action. The business process brings a broader significance to the information [Medina, 92]. The business process, sometimes called the human process, is concerned with the coordination of the people involved in performing the actions generated by the information.

2.3.4 Workflow Systems

A workflow management system is a system that completely defines, manages and executes business processes through the execution of software. Thus, this system provides procedural automation of business processes by managing the sequence of the activities and invoking the appropriate human or IT resources associated with the various activity steps. Workflow systems can be described according to the type of process they are designed to deal with. Some of the important types of workflow systems are explained in the following subsection [Hollingsworth, 94], [Denning, 95].

2.3.4.1 Image-based Workflow: Initially, workflow has been closely associated with image systems, where workflow software helped to automate image routing. Image-based Workflow Systems are designed to automate the flow of paper through an organization, by capturing the paper as images. These were the first workflow systems that gained wide acceptance. In a typical scenario, incoming mail (consisting of paper-based information) is digitized and stored on optical discs as images. The workflow software manages queues of pending forms, automatically balancing the workloads of individual workers that are processing the incoming forms. After the transformation process, which is automated, the converted 'images' are passed among users, possibly involving interaction with other IT applications. This interaction creates a need for a workflow functionality, which is discussed in the next section.

2.3.4.2 Form-based Workflow: Form-based workflow takes image-based workflow one step further. Rather than just routing images to workers, forms, which are text-based and consist of editable fields, are routed throughout the organization. Forms are automatically routed according to the information entered on the form. In addition, these form-based systems can notify or remind people when action is due. Forms also contain data that is accessible by the workflow system. Conditional decisions can be made automatically by the workflow system, enabling routine forms to be filled automatically with the appropriate data. These characteristics provide form-based workflow with a higher level of capability than image-based workflow systems. Further notes and an example of how form processing is accomplished can be found in [Teixeira, 93].

2.3.4.3 Coordination-based Workflow: Coordination-based Workflow Systems are designed to facilitate the completion of work by providing a framework for the coordination of actions. This framework is addressing the domain of human concerns (business processes), rather than optimizing information or material processes. Such systems have the potential to improve organizational productivity by addressing the issues necessary for customer satisfaction, rather than automating procedures that are not closely related to customer satisfaction [Denning, 95]. Coordination-based workflow is based on the theory of communication and coordination developed by Fernando Flores and Terry Winograd in the late 1970s [Denning, 95]. After successful implementation in several case studies, this theory brought a new understanding of work. Flores proposed that most human coordination occur between people in requesting, making, and fulfilling commitments. The importance of the computer lies in facilitating this kind of coordination rather than simply in data processing. This thinking was further developed by representing coordination of people as a finite-state machine and called "conversation for action". In the late 1980s, Flores demonstrated that the basic cycle of coordination reappears at many levels of an organization, not just between individuals, and that the organization itself could be seen as a network of recurring workflow loops [Denning, 95]. This became the basis of a workflow management system produced and patented by Action Technologies. It was seen that the workflow-loop map has provided significant improvements in productivity and in satisfaction of customers and employees. Figure 2-5 shows the generic structure of a coordination loop, called ActionWorkflow loop by Action Technologies. This figure views work as a closed loop process in which a performer completes actions leading to the satisfaction of a customer or client's request. During any phase, the participants may make requests of others, thus initiating secondary loops, whose completion will enable forward progress in the previous loop. The initiation or call to new loops generates a network of connected loops: loop segments can be further refined.

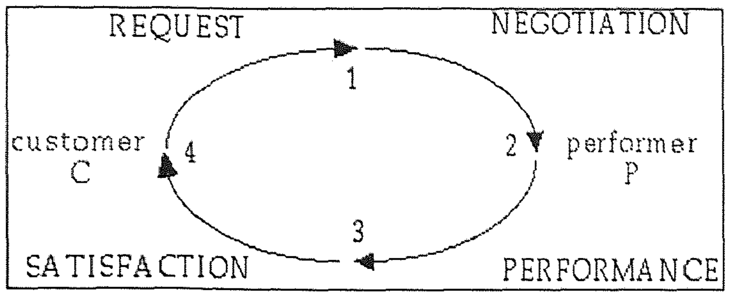


Figure 2-5. Workflow loop [Denning, 95]

The Figure 2-6 is an example implementing the map for a procurement process of an organization, which illustrates expansion into secondary workflow loops; in this case, the primary performance phase is expanded into three sequential secondary loops.

2.3.5 More in Depth in Functionality

All workflow management systems provide support in three functional areas: Build-time functions, run-time control functions, and run-time activity interaction functions [Hollingsworth, 94].

2.3.5.1 Build-time Functions: The Build-time functions deal with the definition, and possibly the modelling of workflow process and its constituent activities. These functions result in a computerised definition of a business process. During this phase, real-world processes are translated into formal and computer processed definition. This definition is often called a process model, a process template, a process metadata, or a process definition. A process definition represents a number of discrete activity steps, which could be initiated by a computer or by users. The process definition may be expressed in textual or graphical form or in a formal language notation.

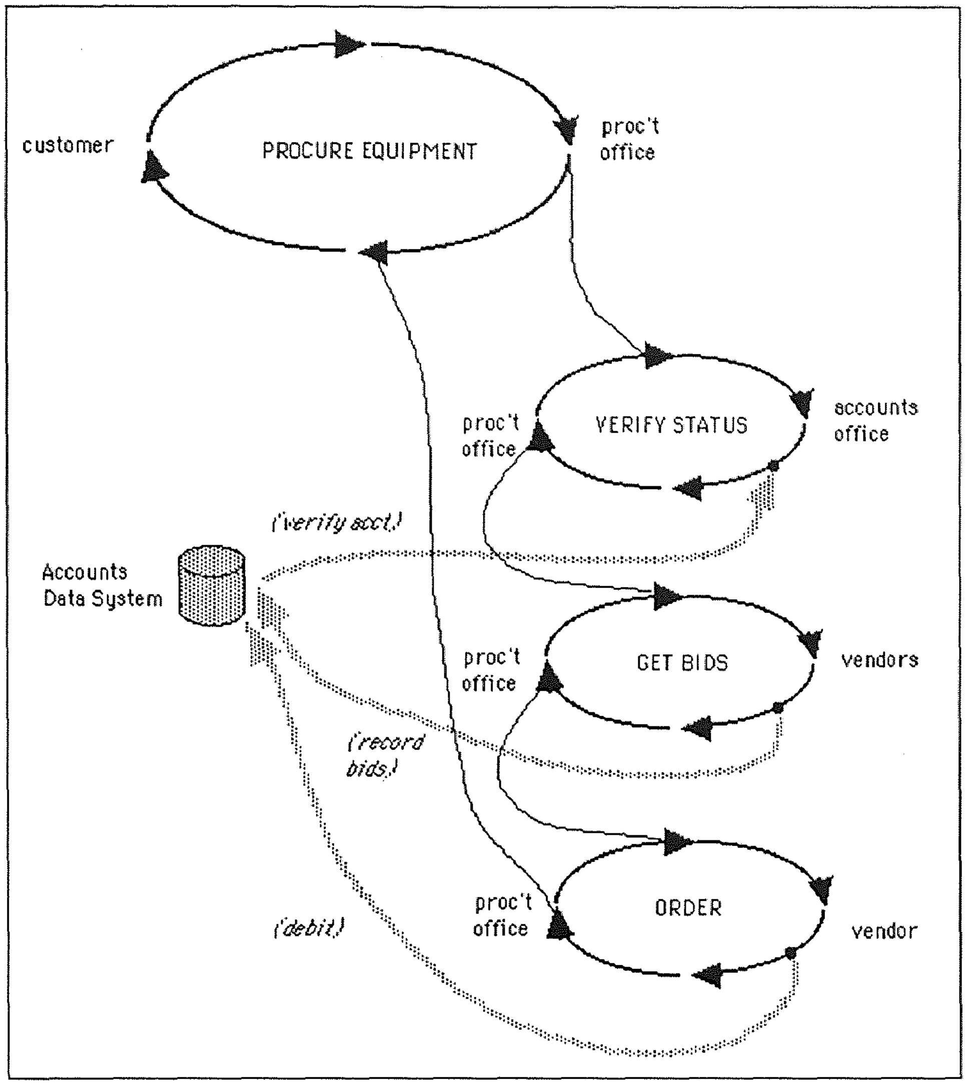


Figure 2-6. Procurement process of an organization [Denning, 95]

2.3.5.2 Run-time Control Functions: The Run-time control functions are concerned with the management of the workflow processes in an operational environment and the sequencing of various activities to be handled as part of each process. During this phase, software interprets the process definition. Also, it creates and controls operational instances of the process, scheduling the various activities within the process and invoking the appropriate human and IT resources. This software is the core component of the workflow management and it is often distributed across a number of computer platforms to manage processes distributed over a wide geographical area.

2.3.5.3 Run-time Activity Functions: The Run-time activity functions enable the interactions of human users and IT application tools for processing the various activity steps. Human intervention is often realised with the use of particular IT tools, which may be some application program that operate on some defined information. The interaction with the process control software, mentioned above, is necessary, since it enables the transfer of control between activities, the checking of processes' status, the invocation of application tools, and the passing of appropriate data.

Figure 2-7 illustrates the basic characteristics of workflow systems and the relationships between its main functions.

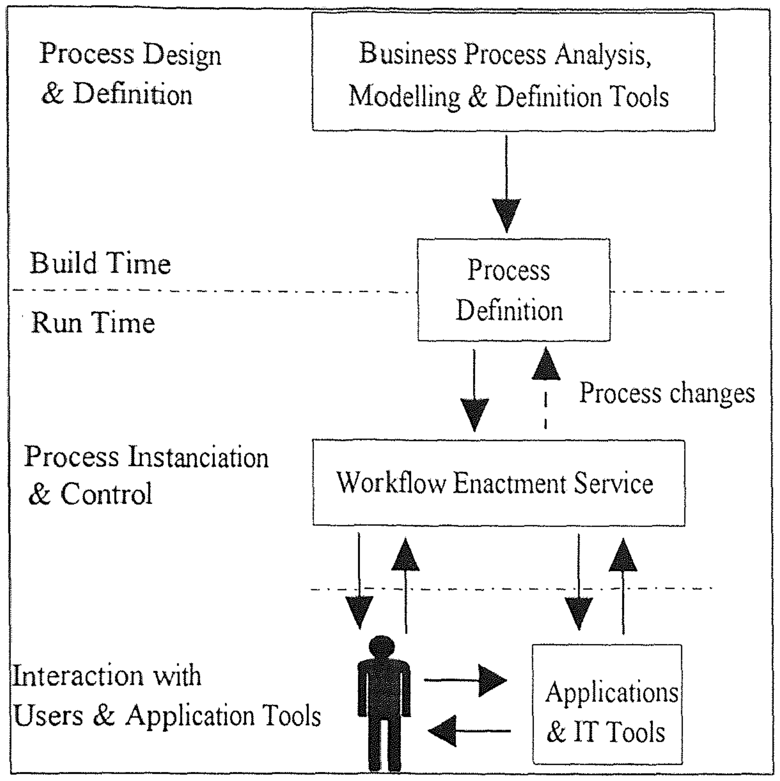


Figure 2-7. Workflow system characteristics [Anonymous, 96]

2.3.6 Comparison of Workflow with Process Modeling

Workflow systems are usually understood as process modeling. However, workflow is not only limited with modeling. It comprises other factors that differentiate it from process modeling. Figure 2-8 shows the Workflow Reference Model. As seen in the figure, there are different aspects that make up the Workflow systems. The engine, at the center of the action, is an engine that controls the flow of information and data. The main difference lies in this distinction. Workflow generates the flow of data in an automated fashion. However, process modeling does not deal with data, but represents products, objects, and information.

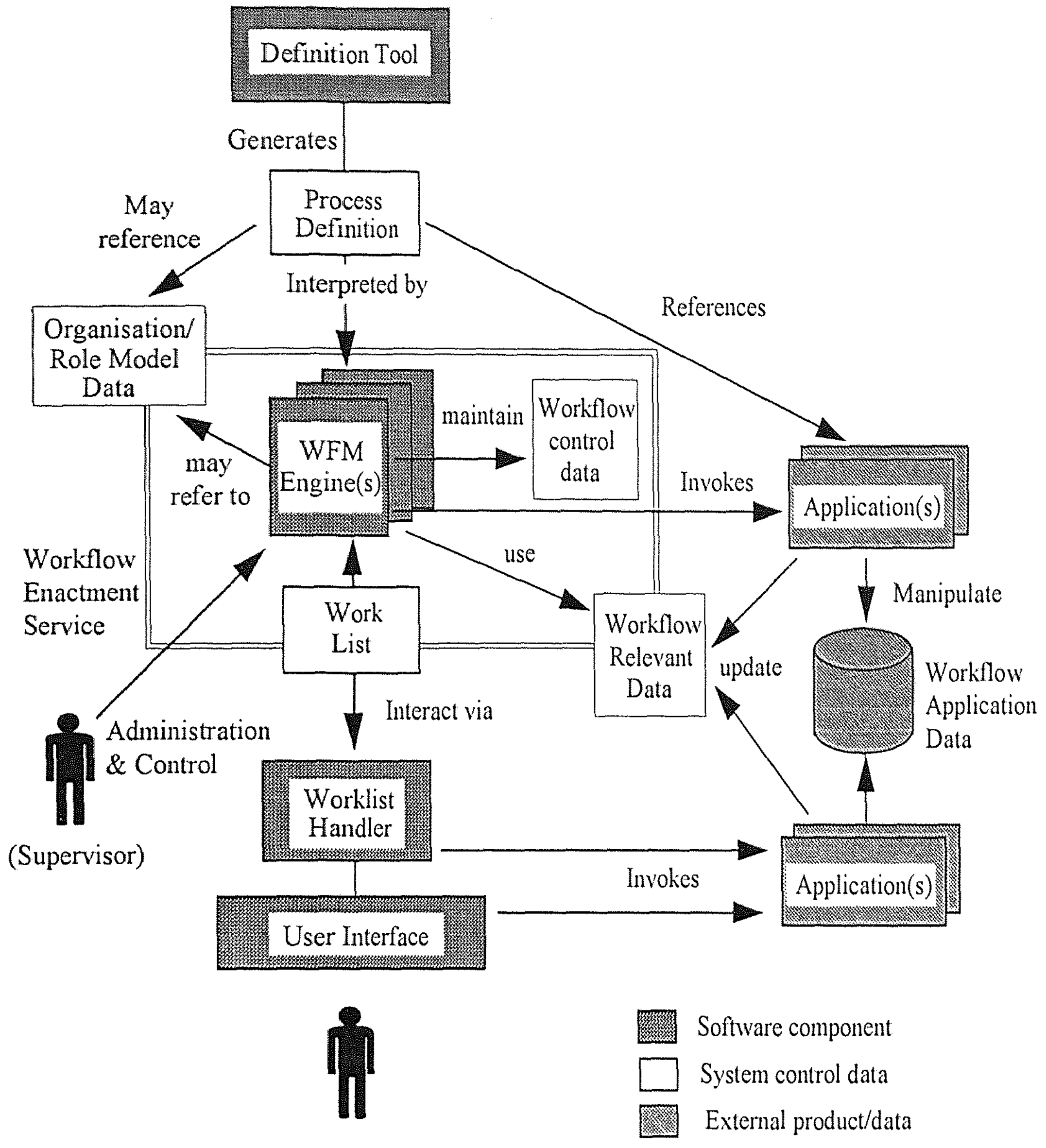


Figure 2-8. Workflow reference model [Hollingsworth, 94]

CHAPTER 3

ProSLCSE

3.1 Overview

ProSLCSE (Process Oriented Software Life Cycle Support Environment) is a tool produced by a company named International Software Systems, Inc. (ISSI) [ProSLCSE, 95]. Its main purpose is to facilitate and to enhance the improvement efforts in organizations. According to the principles of total quality management and continuous process improvement, organizations need to define clearly the steps that they use to conduct key business processes. Those steps are then analyzed and controlled to find better ways to do the job, in an efficient and effective way. Three key points of an engineering problem are time, cost and reliability, which are addressed with a clearly defined set of processes.

ProSLCSE helps users reach their objectives in the following ways:

- Define the infrastructure (resources, resource types, and the relationship among them) in an organization,
- Define the processes used by an organization to perform the tasks at every phase,
- Associate the resource types to the processes,
- Create a project by allocating specific resources (as defined in the infrastructure) to roles and other resources types and by specifying the planned duration to complete each tasks and activities,
- Analyze the project with simulations and other analysis tools,

- Enact or run the project. The staff members assigned to the project can check the status of their tasks in an out of the ProSLCSE system. Based on the dependencies in the project model and the completion of activities reported by the assigned members, ProSLCSE determines the readiness of the activities to be implemented [ProSLCSE 95].

ProSLCSE has two editors, the Infrastructure Editor and the Process Editor, which are structure-oriented graphical editors using the Visual Process Modeling Language (VPML). Following the rules set for interconnecting different components, these two editors help users construct VPML process models rapidly and correctly. Both editors use graph-based interfaces to edit infrastructure and process graphs and forms-based interfaces to edit the attributes of VPML constructs [ProSLCSE, 95].

ProSLCSE and the terms described above would be explained in more depth in the following section, but I would like to give an introductory overview of what those editors stand for and what they look like in real life.

3.1.1 Infrastructure Editor

The Infrastructure is defined for an enterprise. An enterprise could be an organization, a company, a division within the company, or any group that has resources that need to be allocated to resource types. The Infrastructure Editor displays all the resource types and resources, along with their relationship to each other. Resources are members of resource types. An example of an infrastructure from different perspective is shown in Figures 3-1a to 3-1d.

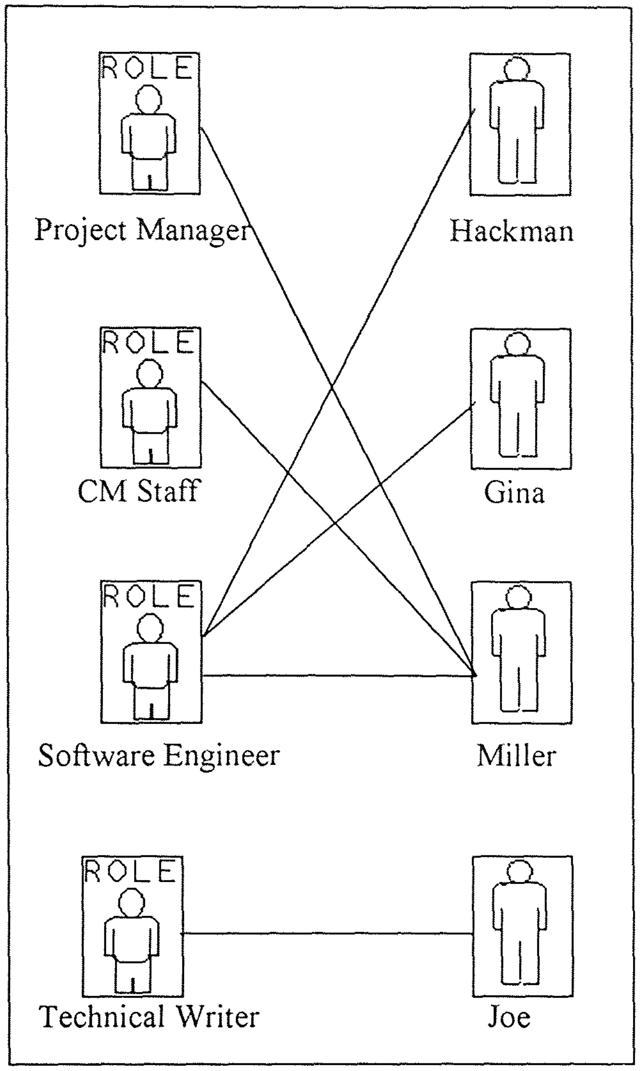


Figure 3-1a. Example of infrastructure involving people

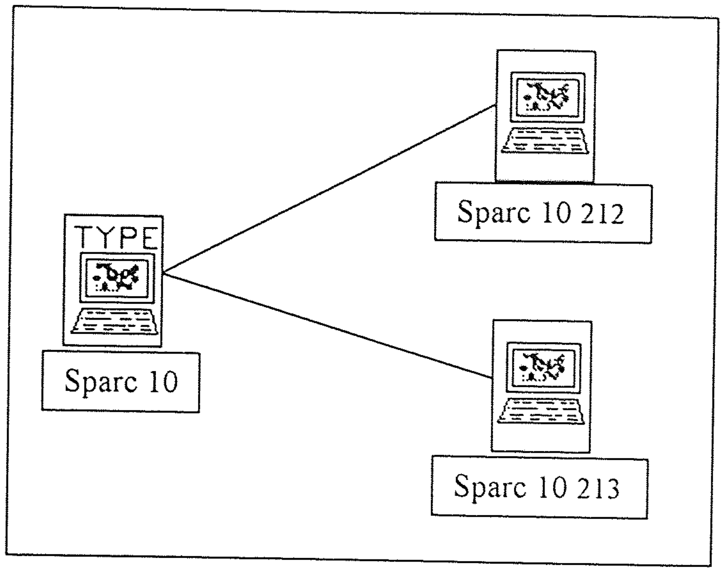


Figure 3-1b. Example of infrastructure involving machines

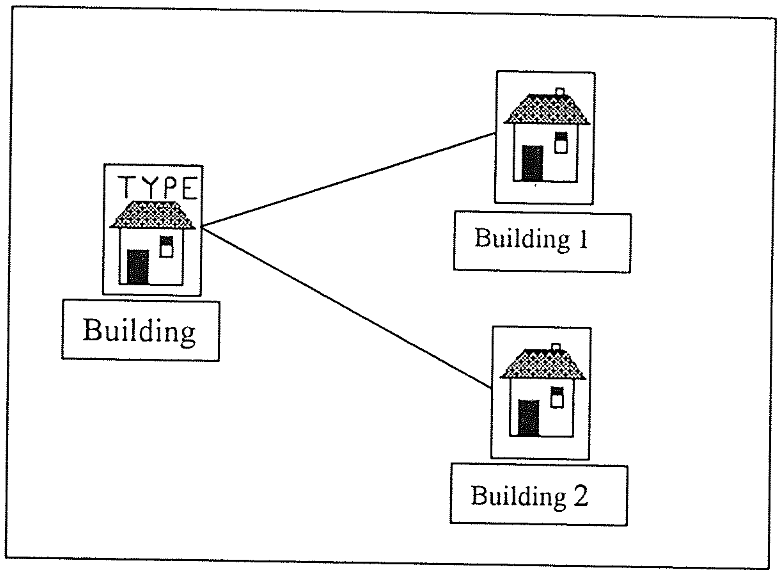


Figure 3-1c. Example of infrastructure involving location

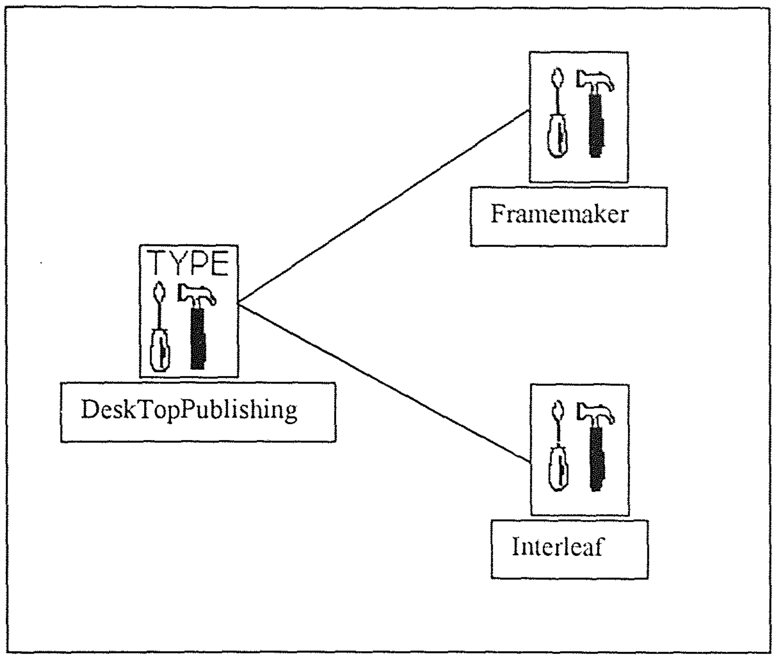


Figure 3-1d. Example of infrastructure involving tools

3.1.2 Process Editor

Process diagrams are made of leaf activities, composite activities, and automatic activities that have products, like artifacts, messages, documents, folders, composite products, as inputs and outputs. Roles are associated with activities, and tools are associated with products. An example of a process diagram, as seen in Figure 3-2, illustrates a high-level view of processes. This diagram, also called external diagram might contain a composite activity (Activity 12). By clicking, the user could illustrate a more detailed level (also called internal view) of this activity.

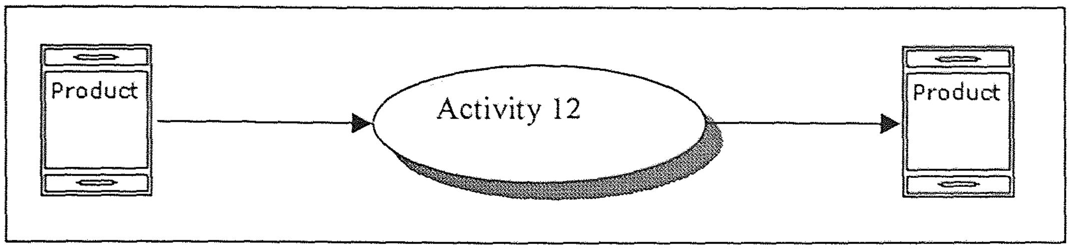


Figure 3-2a. External view of an Activity

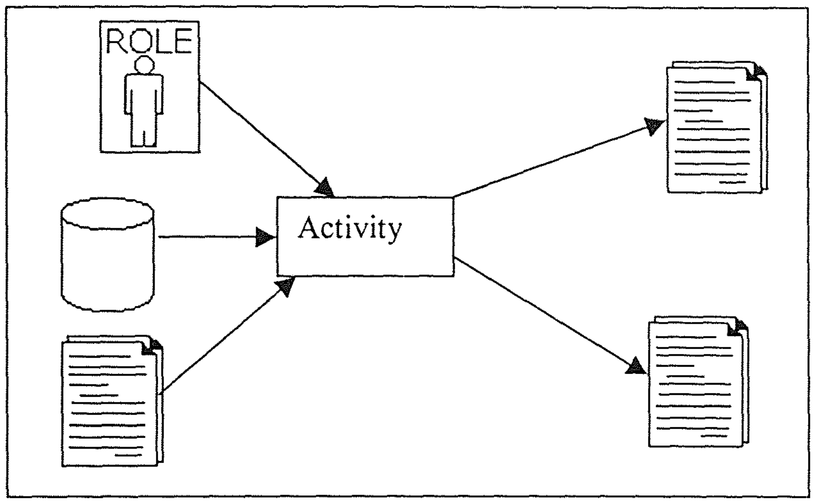


Figure 3-2b. Internal view of Activity 12

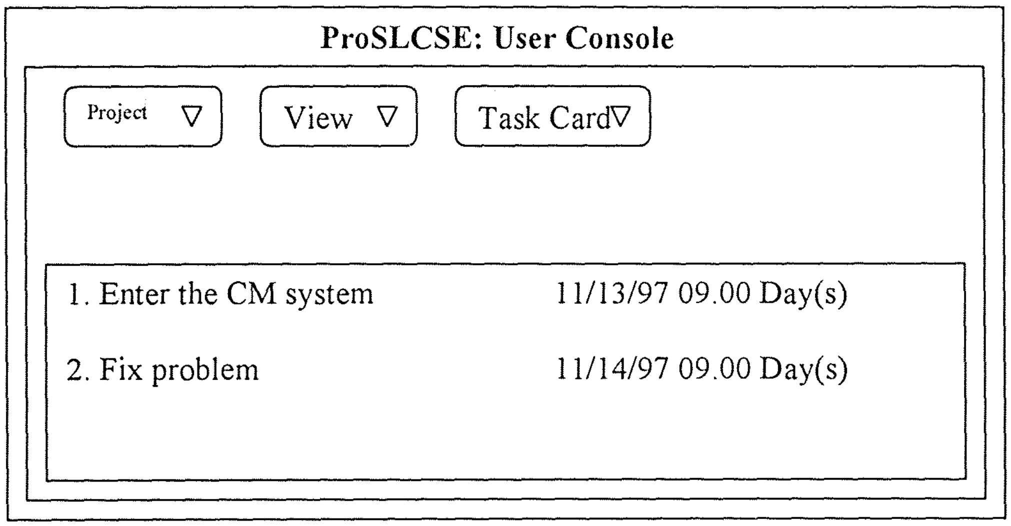
The process Editor enables users to assign specific personnel in the infrastructure to roles and to set time limits for performing activities. Only after all roles in the process have been assigned to persons that the simulation or the enactment can start.

3.1.3 Analyzers

After the completion of a process, two kinds of analysis is possible (optional): the Static Management Assistant and the Project Simulator. The Static Management Assistant produces cost and resource utilization reports based on time duration and cost definition. The Project Simulator allow the user to activate or run the process. Users have the possibility to simulate the process step by step, one activity at a time, and if desired, random factors can be added to make a total analysis of the processes.

3.1.4 Enactment

When resources have been assigned to all resources types, the project can be enacted. The staff members involved in the project can check their activities through a User Console, as shown in Figure 3-3.



1. Enter the CM system 11/13/97 09.00 Day(s)

2. Fix problem 11/14/97 09.00 Day(s)

Figure 3-3. User console

3.2 Visual Process Modeling Language

This section introduces the Visual Process Modeling Language (VPML), which is a graphical language meant to define processes. The components of a process diagram are activities, products, resources and resources types, annotations, the connections (flow connections, data flow connectors, temporal connections), input/output indicators, and other miscellaneous connections. The VPML is based on constructs that are shown in Table 3-1 through Table 3-9. Each construct is represented by its visual representation, its construct name and its description.

3.2.1 Activities

Activities are the central focus of VPML. They represent the work that is performed in a process. Other VPML constructs are usually assisting activities to specify details and to coordinate activities. There are three types of activities in VPML: leaf activity, automatic activity and composite activities.

3.2.1.1 Leaf Activity: Leaf activity represent work done by one or more persons. It needs at least one role associated with it to pass the completeness check. This makes the activity ready to be simulated or enacted. During its enactment, a leaf activity is in one of the eight states: Wait-For-Start, Pending, Ready, Active, Wait-For-Finish, Cancelled, Suspended, or Finished. Users are able to follow those states through the user console. The state of the activity determines what operations are possible to perform. Specific conditions force the states to change. Those states and how they are related to each other is shown in Figure 3-4.

During the first enactment, all leaf activities are set to Pending state. Transition T1 occurs when the inputs to the activity is ready, and at least there is one Finish_Start, Start_Start, or Start_After_Start connection waiting to be satisfied. T2 occurs when all the connection mentioned above are satisfied, resulting the transition to the Ready state. A person assigned to the Ready state has to initiate transition T3 for the activity to pass to Active state.The reverse, the transition T4, follows the same routine. For the transition T5 to take place, there are 3 conditions that must hold: Each person assigned to the activity must approve that the work on the activity has been completed. Furhtermore, the

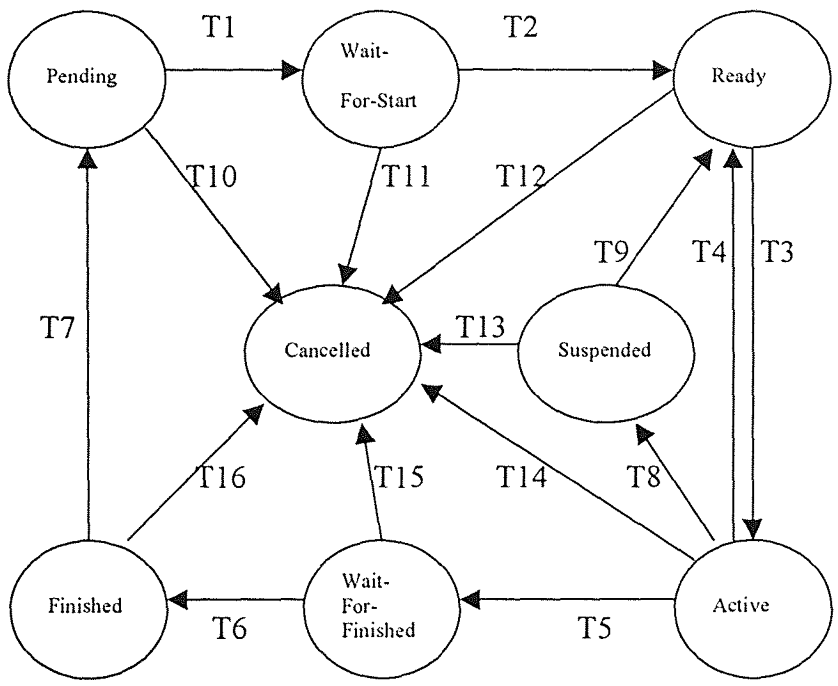


Figure 3-4. Leaf activity state transition diagram

conditions on the Output_OR and Output_AND must also be true, that is, only one path has been selected for each Output_OR connector and all paths have been selected for each Output_AND connector attached to the activity. The activity stays in the Wait-For-Finished state as long as there are any unsatisfied Finish_after_Finish and Finish_Finish connections. Once those connections are satisfied, the activity is sent to the Finished state. Transition T7 from the Finished state to the Pending state takes place when at least another input to the activity is ready again. The manager in the Project [ProSLCSE 95] controls all other transitions to the Cancelled and Suspended State.

3.2.1.2 Automatic Activity: An automatic activity is an activity that is performed by a computer script or program. Enactment behaviour for automatic activity is similar to that of leaf activity. During enactment, an automatic activity is in one of the six states: Ready, Wait-for-Start, Pending, Wait-For-Finish, Active or Finished. The states and the transitions are shown in Figure 3-5.

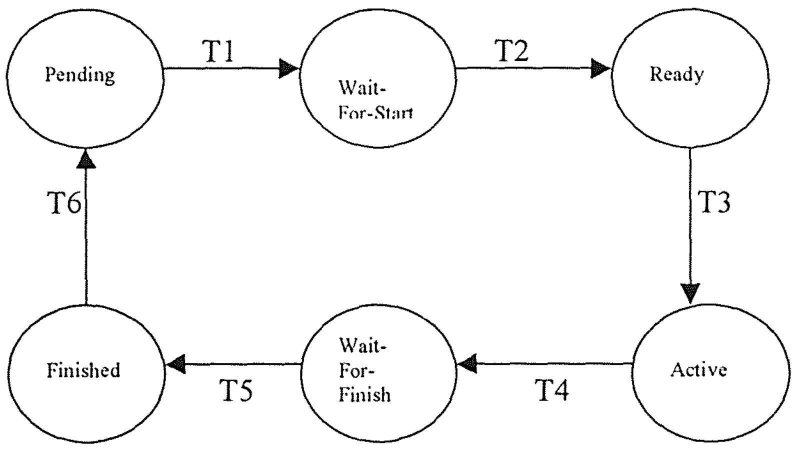


Figure 3-5. Automatic activity state transition diagram

All automatic activies start from the Pending state. Transition T1 occurs when the inputs to the activity are ready, and at least there is one Finish_Start, Start_Start, or Start_After_Start connection waiting to be satisfied. After those connections are satisfied, the activity moves to the Ready state, passing by T2. After the transition from Ready state to Active state, instructions from the program script are executed. The program or the script that is being executed has read access to the products along the reference and data flow connections. It also has create, read, and write access to products along the outgoing data flow connections for the activity. For the activity to pass to Wait-For-Finish state from the Active state, three conditions have to be satisfied. The program or the script for the activity must be finished; Only one selection must be indicated for each Output_OR attached to the activity; And finally several selections must give way for each Output_AND attached to the activity [ProSLCSE 95]. The automatic activity resumes to the next state, Wait-For-Finish, if the activity has any temporal dependencies that need to be finished. To make the T6 transition, all Finish_after_Finish and Finish_Finish connections should be satisfied, putting the activity in the Finished state. Transition from this state to Pending state is possible when new inputs are ready.

3.2.1.3 Composite Activity: A composite activity enables the user to represent activities and processes in a varying levels of detail. A single composite activity can contain leaf, automatic, other composite activities and constructs within itself. Because of this capability, two views are assigned for this construct: Internal and external views. In its external view, a composite activity can be considered like a leaf activity, which can be seen in Figure 3-2a. In its decomposed form, a composite activity shows its component in a detailed view, as seen in Figure 3-6. This decomposition has no effect during the enactment. This type of activity does not have states. It is only for visual purpose.

The constructs representing activities are summarized in Table 3-1, along with their definition and their visual representation.

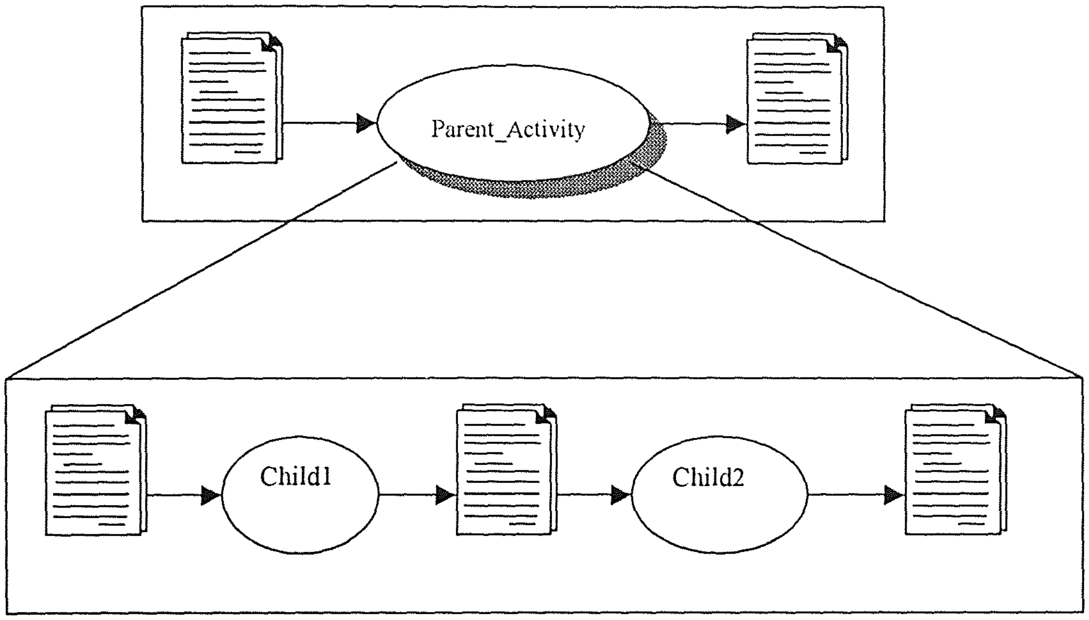
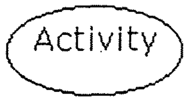
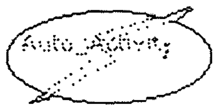
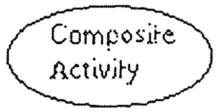


Figure 3-6. Composite activity with external view and internal view

Table 3-1. VPML constructs representing activities (Adapted from ProSLCSE manual)

Visual Representation	Construct name in ProSLCSE	Description
	Activity	Represents work that is performed by one or more persons or group. Also called leaf activity.
	Automatic Activity	Represents work performed by a computer script, rather than manual work.
	Composite Activity	Represents a sub-graph of subdivided activities.

3.2.2 Products

Items that are created, used, modified, and transferred among activities are called products. A product is being created or modified when it is on an incoming or on an outgoing data flow. A product, that is on an output flow connection from one activity and an input flow connection to another activity, is considered to be transfered from the first activity to the second.

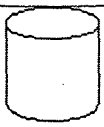
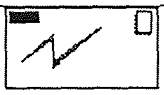
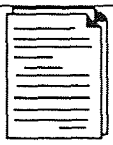
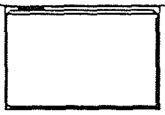
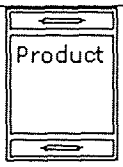
3.2.2.1 Artifacts, Messages, and Documents: An artifact, in VPML, is the most general class for representing products. It represents products that can not be represented by document or message constructs. A source code or a database would be some of the examples for artifact. A message is a specialized artifact. It is used primarily to represent messages that are sent from one activity to another one. A document is also a specialized artifact and is used to model document that are being created, modified or used during the activity.

3.2.2.2 Folder: A folder is a graphical representation of a dynamic collection of products. Folders could collect different products. They could be the input or an output to an activity. When a folder is connected to a leaf activity, the person assigned to the activity has to carry out the activity specified in the folder. Same goes for an automatic folder. But in this case the activity's script specifies the procedure by which products are extracted or inserted into a folder. The leaf activities that use folders can not move to the Ready state until the folder is completed. Folders have to be completed for an automatic activity to move to the Active state, as well.

3.2.2.3 Composite Product: The composite product is a collection of all the products mentioned above. By representing the group of products as a single product, it is possible to reduce the space covered by those products to minimal. Unlike folders, the components of a composite product do not need to be related to each other. Since the representation of composite product is for visual purpose, this construct does not have any impact on the enactment.

Table 3-2 represents a summary of the products.

Table 3-2. VPML constructs representing products (Adapted from ProSLCSE manual)

Visual Representation	**Construct name in ProSLCSE**	**Description**
	Artifact	Represents a general kind of product.
	Message	Represents messages sent from one activity to another one. It is also a specialization of artifact class.
	Document	Represents documents created, modified, or used during a process. It is also a specialization of artifact class.
	Folder	Holds a collection of products, that may change dynamically as the process runs.
	Composite Product	Represents a collection of products. Allows users to present multiple products with a single icon.

3.2.3 Resource Types and Resources

VPML resource types and resources are representing real-world resources that are involved in the accomplishement of an activity. VPML does model human and certain nonhuman resources such as tools, machines and locations. VPML makes a distinction between a resource type and an instance of that resource. During a process definition, a resource type might be assigned to a general construct such as a programmer. Later on in the instantination phase, a specific person name with that skill might be specified. This difference allows some flexibility during the definition phase, leaving the choice of to be made later in the instantination when the activity is ready to be performed.

In ProSLCSE, the resources instances for an organization are outlined in the Infrastructure Editor. Also, resource types may be assigned in the Process Editor. But specific resources may only be added in the Infrastructure Editor. Users are urged to specify the relationship first in the Infrastructure Editor, to be able to assign instances to the resource types.

A summary of all the resource types and resources is outlined in Table 3-3, with their visual representation and their definition.

3.2.3.1 Person Type and Person: A person type, also called Role, is used to model an individual person with set of skills necessary to complete the activities in a process. A programmer, technical writer, tester, manager are some of the examples. However, a person is specific name that has those particular set of skills to complete the task. A person may have different skills, allowing that person to be assigned to several activities at the same time. To be assigned to any activity, a person has to be associated with a role. A person instance has some attributes that identify it from the rest of the instances. Name, availability, cost and efficiency are the attributes that are assigned to each person. An example of role (person type) and associated persons was shown in Figure 3-1a.

3.2.3.2 Machine Type and Machine: A machine type is used to model the equipment needed to perform the activities of a process such as workstations, printers, etc. A machine is refered as a specific name of that particular instance of that equipment. The attributes for a machine are the name, the availibilty and the cost per hour to use that particular machine. A SunSPARCstation 12, or ID#234 are some examples of machine instances. The figure 3-1b, showing the machine type and machine instances, was already illustrated in the previous section.

3.2.3.3 Location Type and Location: A location type is used to model a physical place where the activity is going to be performed. A factory, a building, or a room are examples of location types. A specified location name is refered as location in VPML. The attributes for a location are the name, the availabilty and the cost to use that particular location. Instances relative to the previous examples might be the manufacturing factory, building 4, or the conference room. Figure 3-1c shows a visual example of how location types and location instances are represented in the Infrastructure Editor.

3.2.3.4 Tool Type and Tool: A tool type is used to model software, application or other type of tools needed to operate on a product in the process. A tool is referred as an instance of the tool types. An example of tool type might be spreadsheet and one of the instance might be Microsoft Spreadsheet. The attributes for a tool are its name, its availibity, its cost per hour and its number of licences. A visual example was shown in Figure 3-1d.

3.2.3.5 Group type and Group: A group type is a special type which can contain several resource types.Roles, machine types, location types and tool types could be incorporated in this construct. However, a group is an instance of the group type that might contain more than one persons, machines, locations, and other groups. The group is differentiated from other groups with its number or name assigned to it.

3.2.4 Flow Connections

Flow connections are establishing the relationship between VPML constructs and passing product information between activities, thus helping the schedule coordination of activities. VPML has separated flow connections into three categories: data flow, timer, and reference connections.

3.2.4.1 Data Model Connections: Data model connections model short-term access to products, and are represented by black arrow, connecting constructs to each other. The direction of arrow shows the direction of information's flow.

3.2.4.2 Timer Connections: Timer connections are drawn from timers to leaf, automatic, and composite activities. Timers pass the system clock to activities.

3.2.4.3 Reference Connections: Reference connections model long-term access to products. They always connect products to activities. Although a reference connection is similar to data flow connection, it differ itself by making products always available to activities. A summary of the flow connection construct, along with their visual representation is presented in Table 3-4.

Table 3-3. VPML constructs representing resources and resource types (Adapted from ProSLCSE manual)

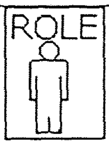
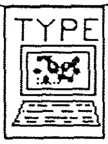
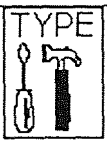
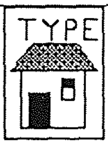
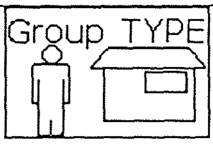
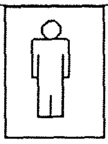
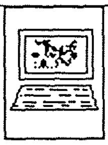
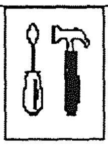
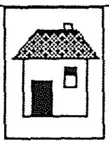
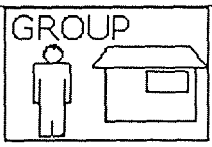
Visual Representation	**Construct name in ProSLCSE**	**Description**
	Role	Represents a person with a particular set of skills (also called person type).
	Machine type	Represents a set of equipment(e.g., Sun SparcStations).
	Tool type	Represents a particular set of tools (e.g.,word processing).
	Location type	Represents a type of location (e.g., manufaturing building).
	Group type	Is a collection of resources types, including roles, machines types, location types, and other group types mentioned above.
	Person	Represents a specific person who is aaigned to do the job (e.g., Robert Duwall).
	Machine	Represents a specific machine (e.g., Sparc #210).
	Tool	Represents a specific tool (e.g., Microsoft Word).
	Location	Represents a specific location name (e.g., Building).
	Group	Is a collection of resources, including instances of persons, machines, locations, and other groups.

Table 3-4. VPML constructs representing flow connections
(Adapted from ProSLCSE manual)

Visual Representation	**Construct name in ProSLCSE**	**Description**
⟶ (Black)	Data flow	Connects products to activities. After the completion of the activity, the input products are reset to indicate that they have been consumed and need to be produced again to deactivate.
⇢ (Red)	Timer Connection	Associates timers as inputs to leaf and automatic activities. Timers send their input at a specified time or at intervals determined by the timer attributes or the system clock.
┈┈┈ (Black)	Reference	Has a similar role as the data flow, except that once the product is associated with this construct, the product is always available.

3.2.5 Data Flow Connectors

Data flow connectors enable flexibility in providing different flow path to be considered. By following alternative flow paths, the process can resume depending on the outcome of the connector. There are four types of data flow connectors: Input_OR, Input_AND, Outout_OR, and Output_AND. The Table 3-5 summarizes these connectors.

3.2.5.1 Input_OR and Input_AND: These two connectors allow multiple data flow connections to be merged. The resulting information is passed to the output connections. Although Input_ORs and Input_ANDs can be combined in arbitrary ways to form complexe graphs, there are some stylistic conventions to be followed in order to keep the resulting graph simple. Input_ORs can only be connected to activities and Input_ANDs can only be directly connected to Input_ORs. In this way, the graph is limited to at most one layer of Input_ANDs over one layer of Input_ORs. The following figure 3-7 shows an implementation of this convention.

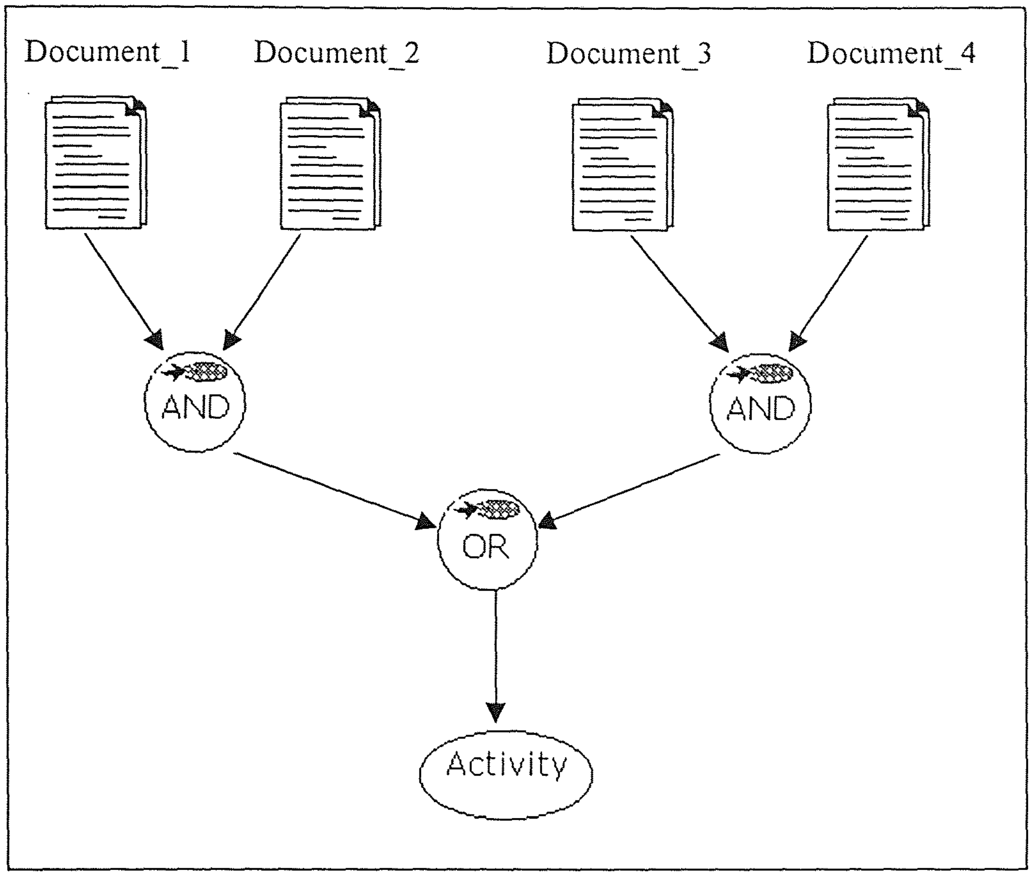


Figure 3-7. Example of Input_ORs and Input_ANDs [ProSLCSE 94]

This example illustrate that either Document_1 and 2 or Document_3 and 4 are necessary to resume the activity.

3.2.5.2 Output_OR and Output_AND: Output_OR and Output_AND are data flow connectors used to provide choices among different outgoing data flow paths originating from an activity. During the creation of a data flow connection originating from an

Output_OR connector, ProSLCSE adds a default label which is attached to the connections. The default name can be changed by right-clicking on the label. Figure 3-8 shows a combination of these two data flow connectors.

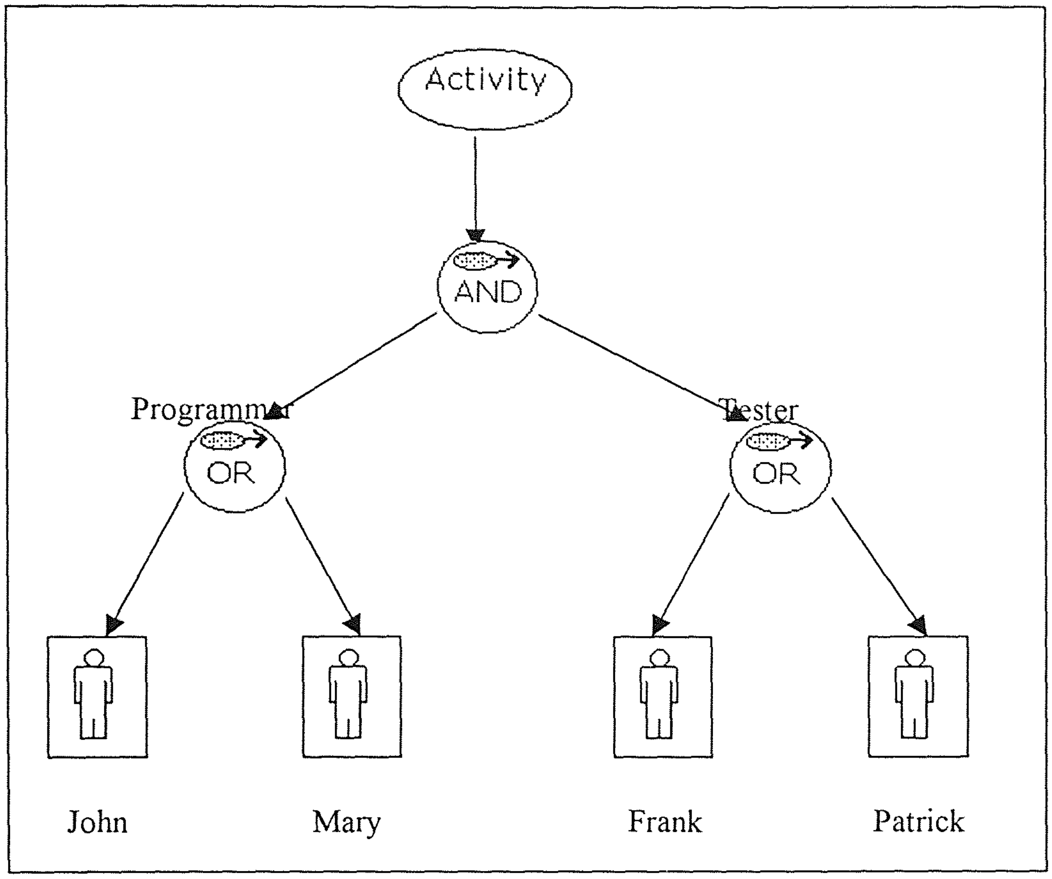
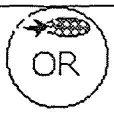
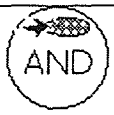
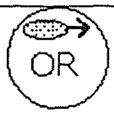
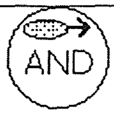


Figure 3-8. Example of Output_ORs and Output_ANDs

The example illustrates that either John or Mary will be chosen as programmer and either Frank or Patrick will be selected as tester. Table 3-5 represents data flow connectors with their visual representation and their characteristics.

Table 3-5. VPML constructs representing data flow connectors (Adapted from ProSLCSE manual)

Visual Representation	Construct name in ProSLCSE	Description
	Input_OR	Allow multiple input data flows to be joined, but only one input to be sent to an activity.
	Input_AND	Allows multiple input data flows to be joined, and all inputs have to be sent to an activity.
	Output_OR	Allows multiple output data flows to be joined when only one of the flows is to be followed.
	Output_AND	Allows multiple output data flows to be joined when all the flows are to be followed.

3.2.6 Temporal Connections

Temporal connections outline the timing dependencies among activities. Although very similar to data flow connections, they require only direct connection from one activity to another. A Finish_Start connection requires the first activity to change to Finished state for the second activity to be initiated toward Ready state. A Start_Start connection is used to model activities that need to be initiated at the same time. Two activities connected with the Start_Start connection has to wait each other until both of them have their inputs ready and their temporal conditions satisfied. A Finish_Finish connection is used in cases where multiple activities have to finish simultaneously. In order for both activities to advance to the Finished state, both activities should have satisfied their prerequisites. A Start_after_Start connection urges one activity to change to the Ready state in order to get itself in the Ready state. The first activity is dependent on the second activity. A Finish_after_Finish connection is modeled to display activities that are dependent on each other, as well. An activity has to be finished before another one is considered finished. This last two type of connections are somewhat conditional connections. An activity is wholly dependent on another one and the continuation of the process depend on the outcome of the first activity. Table 3-6 shows a summary of temporal connections.

3.2.7 I/O Indicators

There are two indicators that are automatically put in the internal view of a composite activity to show that the products are connected to an external view. Input I/O indicator is placed at the beginning of the internal view and the Output I/O indicator is placed on the last product of the internal view. Table 3-8 lists a summary of those two indicators along with their visual representation.

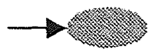
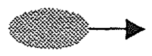
3.2.8 Tunnel Connections

There are two tunnel connectors: Tunnel_IN and Tunnel_OUT. Those two connectors are used together for different reasons. First, they may represent a data flow between an activity and the output product, while bypassing all the connector between them, or vice-versa. Another reason is to show a temporal connection between two activities. The last option is used when there are large-scale representations to be drawn and the whole image does not fit in the Editor's view. The two connectors should have the same name. The use of tunnel connectors is not advised, since it will make the readability of the diagram harder. These two connections are represented in Table 3-8.

Table 3-6. VPML constructs representing temporal connections (Adapted from ProSLCSE manual)

Visual Representation	**Construct name in ProSLCSE**	**Description**
············▶ (Blue)	Finish_Start	Represent a temporal dependency between activities. The source activity should change to the Finished state before the destination activity can is enabled out of the Pending status to the Ready status.
············ (Green)	Start_Start	Represent a temporal dependency between activities. This construct prevent the activity to change from Ready state until all of their products is available.
············ (Red)	Finish_Finish	Represent a temporal dependency between activities. When all the connected activities have met the requirements for being finished, then they can change to Finished state.
············▶ (Green)	Start_after_Start	Represent a temporal dependency between activities. The destination activity has to wait until the source activity is Ready, to get in Ready state.
············▶ (Red)	Finish_after_ Finish	Represent a temporal dependency between activities. The destination activity is depending on the source activity to change to Finished state for itself to be Finished.

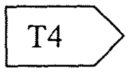
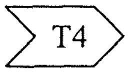
Table 3-7. VPML constructs representing I/O indicators (Adapted from ProSLCSE manual)

Visual Representation	**Construct name in ProSLCSE**	**Description**
	Input I/O indicator	Represents automatically generated input products by the Process Editor. Those are products that are automatically generated in internal views of a composite activity when products are specified as inputs in the external view.
	Output I/O indicator	Represents automatically generated output products by the Process Editor. Those are products that are automatically generated in internal views of a composite activity when products are specified as outputs in the external view.

3.2.9 Other Connections Worth Mentioning

Those are connections that have really special meaning and hereby mentioned in this section. Those constructs have special characteristics that can not fit into any group: type instance relation, group member relation, milestone, and timer. Table 3-9 outlines the characteristics of some of the special constructs.

Table 3-8. VPML constructs representing other connection types (Adapted from ProSLCSE manual)

Visual Representation	**Construct name in ProSLCSE**	**Description**
┈┈┈┈┈┈ (Red)	Association	Associates a resource to an activity or product in the Process Editor and also does same for a person to a tool, location or machine in the Infrastructure Editor.
——————— (Black)	Type Instance relation	Used to connect a specific resource to a resource type in the Infrastructure Editor.
——————— (Black)	Group member relation	Correlate one entity to a group or group type in the Infrastructure Editor.
———————▶ (Black)	Has member group	Used to indicate that a group is a member of a group or a group type is a member of a group type in the Infrastructure Editor.
	Tunnel_IN	Represent a connection in large diagrams, to diminish the space drawn. Used along the Tunnel_OUT
	Tunnel_OUT	Represent a connection in large diagrams, to diminish the space drawn. Used along the Tunnel_OUT

3.2.9.1 Type Instance Relation: This type of connection is only used in the Infrastructure to connect resources to resource types. A process is incomplete if its resource types are not associated with their respective resources. It allows one or several connections to be made from the resource type to resources.

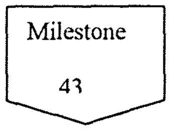
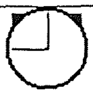
3.2.9.2 Group Member Relation: This connection creates a relation between certain icons in the Infrastructure Editor. The Group types may be connected to other types with this connection and thereby creating a relation between those types.

3.2.9.3 Milestone: A milestone is used to model a significant event during the process initiation. Although no work is associated with milestone, it just represents a special event to be taken seriously. A milestone might be associated with a deadline. It keeps a record of special timeline as to when an action should take place, or produced by the milestone.

Figure 3-9 shows an example of milestone and timer.

3.2.9.4 Timer: A timer is used to model events that take place periodically during a process. A weekly checkup or daily backup is some of the examples. A timer may also be used to provide data to an activity at a specific time or at periodic intervals. Starting and the ending times are its attributes.

Table 3-9. VPML constructs representing miscellaneous constructs (Adapted from ProSLCSE manual)

Visual Representation	Construct name in ProSLCSE	Description
Text	Annotation	Used for explanations, not connected to any particular icon.
	Milestone	Represent a significant event in a process.
	Timer	Used to model periodic events or events taking place at a specific time.

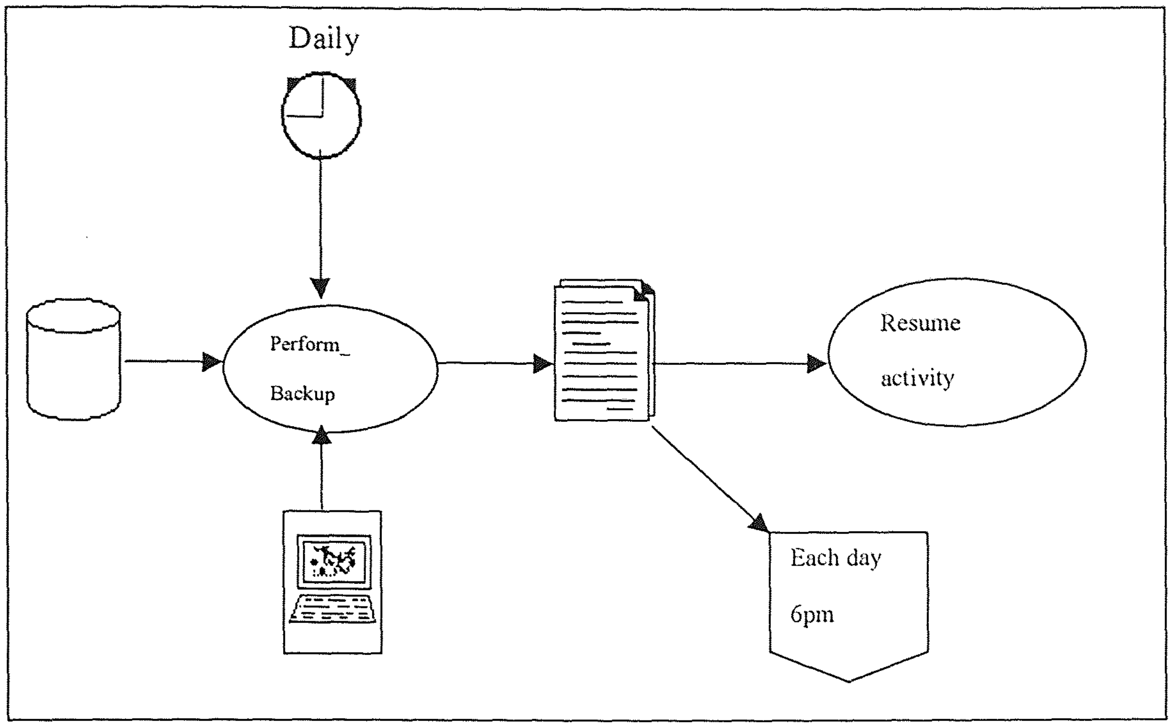


Figure 3-9. Example incorporating a milestone and a timer

3.3 Editors

3.3.1 The Infrastructure Editor

The Infrastructure is a special editor where users define the resources types and instances of resources. Resources types must be declared with this editor. Any types, which have not been assigned a resource, can not be defined and will be ignored by the editor. The resource types and their resources have been thoroughly examined in section 3.2.3. The Infrastructure can also be a medium where relationships other than between resource types and resources are set. A relation between a machine and a person might be defined to show that person is using those machines. Those types of indirect relations are usually defined as part of group types. In addition, a group type might be a member of another group type and so forth. An example of an Infrastructure Editor view is displayed in Figure 3-10. Resource types and their instances are given a label name and their connection is established with type instance relation. Setting group member relation, associations, and has member group type of connections are also made in the Infrastructure Editor.

3.3.2 The Process Editor

Most of the work is done in this editor. The Process Editor is a structure-oriented graphical editor, which uses the Visual Process Modeling Language (VPML), as already mentioned in the Overview section. This editor was intended as a graphical tool to simplify the drawing of process diagrams. It uses a graphical-based interface to edit process diagrams and a form-based interface for editing the attributes of objects [ProSLCSE, 94].

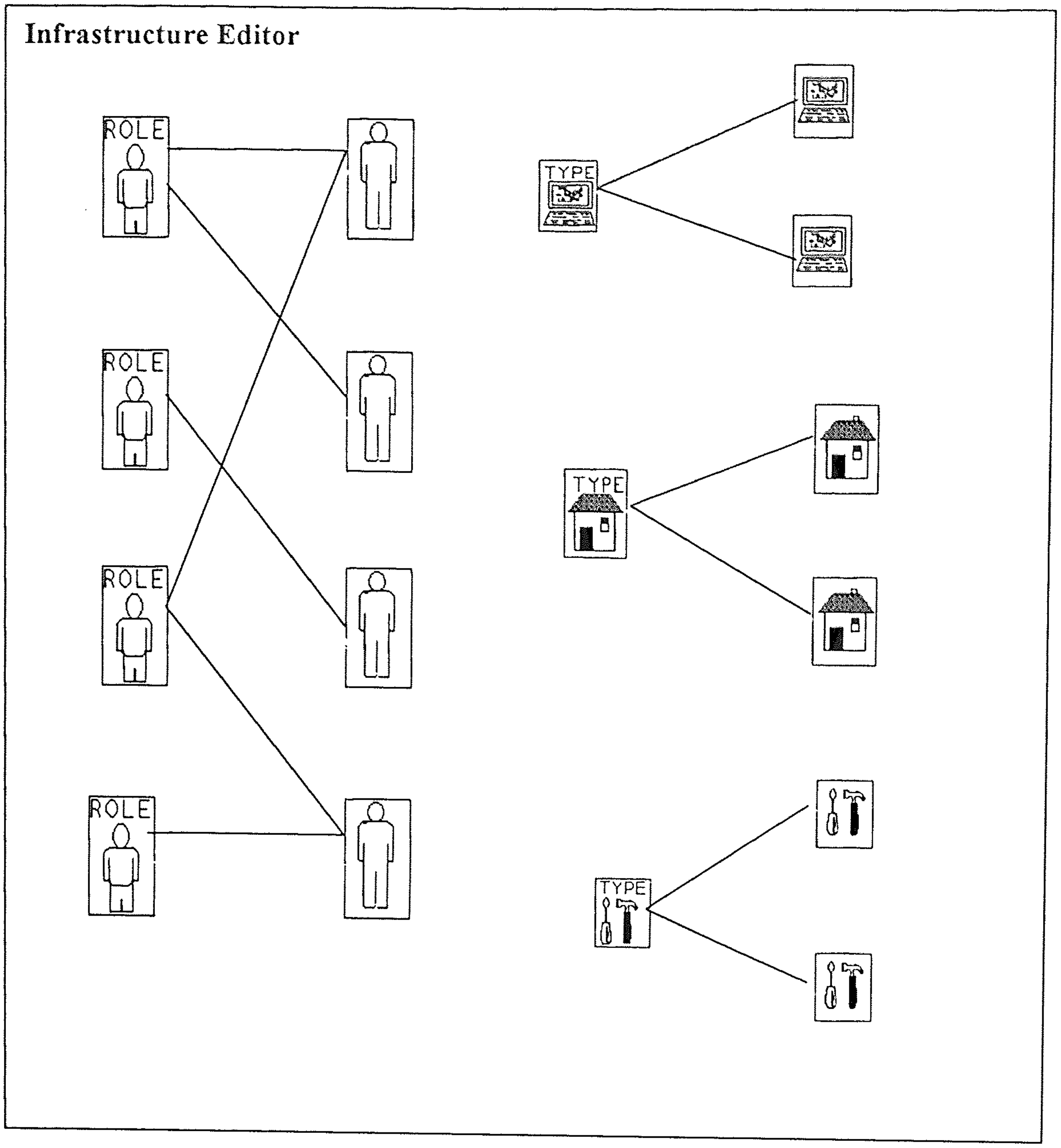


Figure 3-10. A view of an Infrastructure Editor

Creating and editing process diagram is a simple task using the menus or popup menus. By choosing the EditView menu, the users can create, edit or specify constructs. Figure 3-11 shows the graphical interface of such menus.

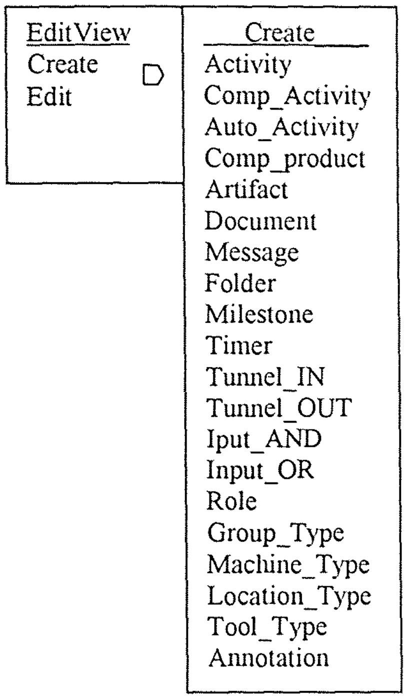


Figure 3-11. EditView and create menus

Popup menus are initiated once the user right clicks on the object created. The next step is to associate object with each other. Clicking the first object and dragging the mouse to the second object draws data flow and association connections. Different connections are possible between objects, as was mentioned in sections 3.2.4 to section 3.2.9.

This concludes the chapter on the ProSLCSE. Further information could be found in the ProSLCSE User's Manual and through the ISSI's web site.

CHAPTER 4

ENVIRONMENT AND PROGRAMMING LANGUAGE

4.1 Internet: The Environment

The Internet is growing fast, and nobody can stop it. Due to ease of use, academic, government, and industrial researchers first populated the Internet [Heffley, 95]. Then, it has become the playground for millions of users, thanks to an application called the World Wide Web (WWW) [Heffley, 95]. Now, with user numbers in the tens of millions, it is a medium of communication no one can deny its existence or what it could for them. The exponential growth of users is so much fascinating, that companies are competing to control it. Although the Internet does not belong to anybody, some companies try to dominate the "networks of networks." Since the access to the medium is relatively free, not counting the connection cost, to browse the Internet is not. A user has to own a browser to surf the net. Therefore, whoever sells its browser will dominate the Internet. Although competition is one important aspect of the Internet, it is not the only issue. Standardization, government control over the Internet, social changes, security, e-commerce are some of those issues.

On October 24, 1995, the Federal Networking Council (FNC) passed a resolution defining the term Internet [Leiner, 98].

"Internet" refers to the global information system that –

(i) is logically linked together by a globally unique address space based on the Internet Protocol (IP) or its subsequent extensions/follow-ons;
(ii) is able to support communications using the Transmission Control Protocol/Internet Protocol (TCP/IP) suite or its subsequent extensions/follow-ons, and/or other IP-compatible protocols; and
(iii) provides, uses or makes accessible, either publicly or privately, high level services layered on the communications and related infrastructure described herein.

In addition to the definition, a historical background to Internet, as well as the issues mentioned above will be discussed in the following sections. These sections are intended to give an overview on Internet, on the Web, and on future expectations and innovations that are being shaped around the net, as well as how our implementation of ProWeb will shape in this medium.

4.1.1 History

It all started when the Department of Defense wanted a secure command and control network that would survive a nuclear attack. Its research department, Advanced Research Projects Agency (ARPA) was set up to research computer networking. Late 1969, the first four nodes of the system went on-line at UCLA, UCSB, SRI and the University of Utah. The network called ARPANET soon connected 23 host by 1971, and the growth became exponential as more computer networks joined ARPANET. Soon, the network became international with the addition of England and Norway, and others followed this trend. By the late 1970s, National Science Foundation (NSF) saw the impact that ARPANET was having on university research and government. The NSF created a virtual network, called CSNET, centered on a single machine, which allowed researchers to call up and leave messages. NSF, then, set up the NSFNet backbone on ARPANET in 1984, consisting of 6 supercomputing centers throughout US. By 1987, the number of computers on ARPANET reached 10,000. It took only 2 years to reach 100,000 computers. In 1990, ARPANET ceased to exist and became known as the Internet. By 1992, a Swiss researcher created the World Wide Web (WWW), a hyperlinked interface to the Internet, making it accessible to millions of users. Hence, by the end of that year, a million of computers were connected to the Internet, worldwide. It is estimated that the number of host computer doubles every year, making Internet the fastest growing sector [Tanenbaum, 96]. Figure 4-1 and Table 4-1 are very good illustrations of that growth.

4.1.2 The Web

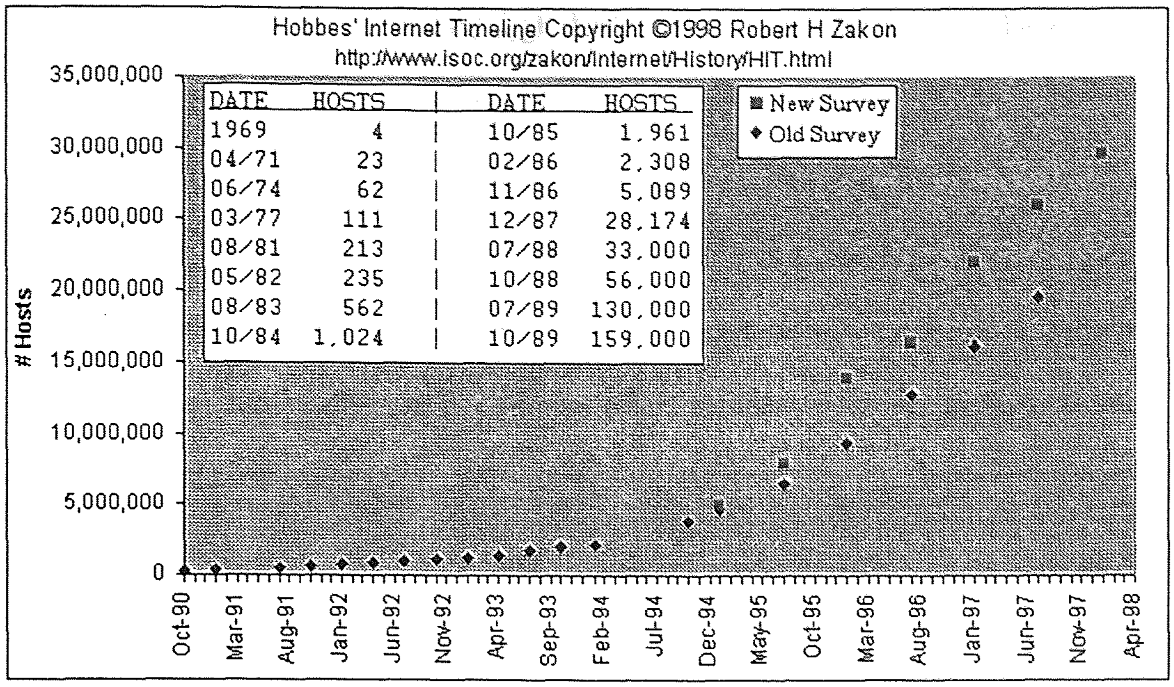
The World Wide Web (WWW) is often confused with Internet. Although WWW is part of the Net, it does not represent the whole Network. WWW may be identified as a huge collection of documents linked to each other without any structure. Web client uses HyperText Transport Protocol (HTTP) to transmit a request to the web server through the Internet. The web server, either returns a static HyperText Markup Language (HTML) document from its origin, or uses some type of scripting language such as Common Gateway Interface (CGI) or Java. The Internet provides a series of other services, as well. The main four applications are 'Email', 'News', 'Remote login' and 'File transfer'. Email came into existence with the early days of ARPANET and nowadays is the most popular tool used in Internet. News is a specialized forum in which users interchange message related to one topic. Remote login permit users to log into any remote machine in the Internet, using Telnet, Rlogin or any other program available. Using FTP program, a user can transfer files from one machine to another. Companies and organizations are using the Internet for many reasons. Shopping, information exchange, development medium are some of its use [Tanenbaum, 96].

4.1.3 Standardization and Control

No government, company or institution owns the Internet. No entity has a controlling power over the Internet. Although the Internet is a truly collective and collaborative environment, there are some organizations that have some sort of influence over the decisions taken around the Internet [Cnet, 97]. Some of the influential ones are the World Wide Web Consortium (W3C), the Internet Engineering Task Force (IETF), the Internet Society (ISOC) and the Internet Engineering Steering Group (IESG). W3C is setting the standards for HTML and other specifics on the Web. The IESG manages the Internet standard process. Those organizations have taken the responsibility to guide the Internet, not really set the standards. Thus, standardization is still an open issue. It seems that the lack of general Internet standards poses as the greatest impediment to the Web. Although the computer industry has set the standards for the Internet protocols, other issues, such as choice of browser, push-technology still remain unsolved [Donaton, 96]. The security of the Internet needs some improvements. Businesses and customers would like to get assurance when it comes to transactions, property rights, secure exchange medium [Spar, 96]. Some specialists recommend the creation of online communities either formed by service providers or selected group of users that will monitor the net in their community and take meaningful course of actions against fraud and abuse. The Internet will certainly affect our social life and people will have to adapt themselves to those changes. The presence of the net communities will also eliminate or at least minimize the presence of government. But still, others ask themselves if government should be totally present in the Internet. Tax issues are still unresolved. The US Treasury Department is preparing a paper that will look at tax issues posed by electronic commerce. To avoid tax liability, some companies are locating their servers in states and countries with little or no tax. Electronic commerce is presenting itself as the ultimate business transaction medium. More and more businesses are creating their web sites and are using Internet as the transaction medium.

Table 4-1. The exponential growth of the Internet [Zakon, 98]

Date	Hosts	Date	Hosts	Networks	Domains
1969	4	07/89	130,000	650	3,900
04/71	23	10/89	159,000	837	
06/74	62	10/90	313,000	2,063	9,300
03/77	111	01/91	376,000	2,338	
08/81	213	07/91	535,000	3,086	16,000
05/82	235	10/91	617,000	3,556	18,000
08/83	562	01/92	727,000	4,526	
10/84	1,024	04/92	890,000	5,291	20,000
02/86	2,308	10/92	1,136,000	7,505	18,100
11/86	5,089	01/93	1,313,000	8,258	21,000
07/88	33,000	07/93	1,776,000	13,767	26,000
01/89	80,000	01/94	2,217,000	20,539	30,000
		07/94	3,212,000	25,210	46,000
		01/95	4,852,000	39,410	71,000
		01/96	9,472,000	93,671	240,000
		01/97	16,146,000		828,000
		07/97	19,540,000		1,301,000
		01/98	29,670,000		



DATE	HOSTS	DATE	HOSTS
1969	4	10/85	1,961
04/71	23	02/86	2,308
06/74	62	11/86	5,089
03/77	111	12/87	28,174
08/81	213	07/88	33,000
05/82	235	10/88	56,000
08/83	562	07/89	130,000
10/84	1,024	10/89	159,000

Figure 4-1. The growth of Internet hosts [Zakon, 98]

4.1.4 Future Expectations

All those issues are still worked on. There are also some improvements made in the Internet. Microsoft and Netscape are competing over push technology, a method that delivers information to the desktop. Applications such as web videoconferencing, phone conversation over the net are areas that will attract more users to the net [Hierhager, 96].

These days, Internet seems the only way to go. Internet has made interconnected networks, usually connecting different and incompatible networks, a common ground of information exchange medium. It enables researchers and users to communicate, exchange ideas, and collaborate. That is one of the most important factors that led us to model are process modeling tool through the Internet, so that the users do have endless possibilities to experiment with it.

Businesses will have an open window to the world market, being able to sell their products and services to the world. Our concern is to be able to draw process diagram through our application. Thus, allowing different teams on different locations to interact with each other. Hence, the Internet offers endless opportunities for customers, developers and businesses.

4.2 JAVA: The Programming Language

Java means different things for different people. Sun's Scott McNealy believes that Java is a new kind of industrial revolution [Cnet, 97]. On the other side, Microsoft's Bill Gates says Java is just another programming language. Nevertheless, the popularity of this language is well recognized from its adversaries, even by Bill Gates [Cnet, 97].

There are different reason leading us to choose Java programming language over others [Manning, 97]. Although still new to this language, my advisors and I felt that it will provide us many opportunities in the future. Java's capabilities compared to other languages are remarkable. Its portability, its similarity to C++, its ease of learning is some of the intriguing factors. Other issues like its performance, its limitations and its future will also be discussed in the next section.

4.2.1 The Origins of Java

Java is a programming language developed at Sun Microsystems in 1990. A small team headed by James Grosling started to develop this object-oriented programming language. Originally, it was designed to be used on small consumer-electronic devices such as television set-top boxes or video games, but soon enough Sun realized that Java could be much more useful than that [Cnet, 97]. With the explosion of the Internet, Java became an ideal programming language for Internet applications. Java addresses many issues of software distribution over the network, including interoperability, security, portability, etc.

Nowadays, most browsers such as Netscape Navigator, Microsoft Internet Explorer and HotJava contain a Java Virtual Machine (JVM). Desktop platforms such as Microsoft Windows, MacOS, OS/2 Warp, and Sun Solaris provide standalone JVM which can execute Java codes directly. Sun is working on way to expand the use of Java on a wide range of consumer products, such as pagers, telephones, and televisions [Tilley, 97]. The relationships of applets, applications, Java Virtual Machines, and platform-independence is show in Figure 4-2.

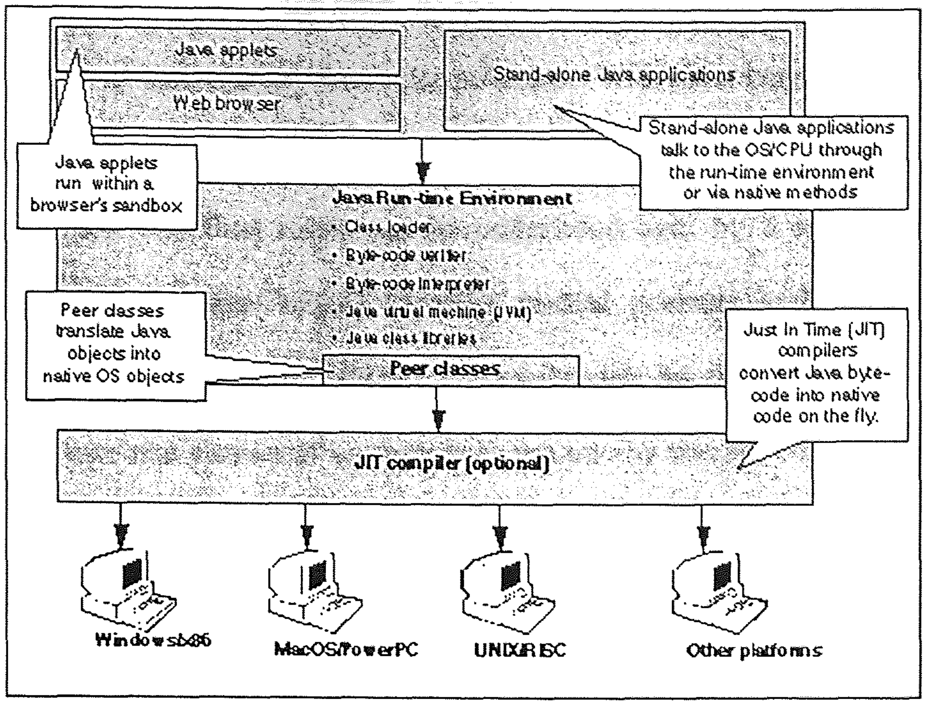

Java applets
Web browser
Stand-alone Java applications
Java applets run within a browser's sandbox
Stand-alone Java applications talk to the OS/CPU through the run-time environment or via native methods
Java Run-time Environment
• Class loader
• Byte-code verifier
• Byte-code interpreter
• Java virtual machine (JVM)
• Java class libraries
Peer classes translate Java objects into native OS objects
Peer classes
Just In Time (JIT) compilers convert Java byte-code into native code on the fly.
JIT compiler (optional)
Windows/x86
MacOS/PowerPC
UNIX/RISC
Other platforms

Figure 4-2. Multiple-platform application [Tilley, 97]

4.2.2 Characteristics

Java is a high-level programming language similar in flavor to Smalltalk and similar in syntax to C and C++. However, the Java language is less complex than C++. Nevertheless, it is an object-oriented that is architecture-neutral, multi-threaded, and robust. Some of other advantages are its built-in garbage collection, its support of a single-inheritance class hierarchy. Java does not use pointers, thereby eliminating most of error sources in many C++ programs. Its syntax resemblance to widely known C and C++ made the Java language popular to most developers [Tilley, 97].

4.2.2.1 Portability: Unlike most other programming languages, Java bytecode is not platform-specific or native to any particular processor. With most languages, developers have to translate their programs into the zeros and ones of machine code with a tool called a compiler. The final result can be understood only by a specific operating system. However, the Java Virtual Machine instead presents a different level of interpretation for Java programs. This is a breakthrough for developers because they only need to write one version of a program to run on any machine. It also means that users don't have to worry about whether the new application they bought will run on a specific platform, since it follows a "write once, run anywhere" approach [Cnet, 97]. This platform-neutrality at both source and binary levels means Java is inherently portable between different platforms. The Java system also provides an extensive library of classes that provides access to the underlying operating system [Tilley, 97].

4.2.2.2 Safety: The Java platform provides portability, a measure of security, and inherent trustworthiness, including strong memory protection, encryption and signatures, rules enforcement, and runtime verification. Java allows applets to be downloaded and executed without introducing viruses or misbehaved code to the client side. It does this by placing strict limits on applets. For example, applets cannot read from or write to the local disk. Unfortunately, while the Java model is secure in theory, it shows signs of weakness at the JVM level. Exploitation of security flaws in the implementations is still alarmingly common [Tilley, 97].

4.2.2.3 Performance: Performance is a major constraint on Java. In most cases, interpreted Java is much slower than compiled C or C++. This is the price that must be paid for being able to run anywhere. But there are ways to speed up Java performance. Recent versions of popular Web browsers and Java development environments provide Just In Time (JIT) compilers that produces native binary code (while the program is loaded and executed). These JIT work with the virtual machine to analyze a Java program and break it down into simpler and more efficient code that can run more quickly [Cnet, 97]. JIT compiler's performance is beginning to catch that of optimized C++ level. For real-time applications, the performance implications of the Java garbage collector should also be considered. Garbage collection may make it difficult to easily bound timing properties of the application [Tilley, 97].

4.2.2.4 Cost and Limitations: Java and the source for the Java interpreter are freely available for noncommercial use. There are some restrictions in incorporating Java into commercial products. Sun Microsystems licenses Java to hardware and software companies that are developing products to run the Java virtual machine and execute Java code. Developers, however, can write Java code without a license. A complete Java Development Kit (JDK), including a Java compiler, can be downloaded for free [Tilley, 97].

4.2.3 Java Applications and Applets

Java programs start as Java source code, which is then compiled to bytecode and stored on a server or a local computer. In order to execute a Java program, a user invokes a

JVM that executes the Java bytecode. There are two "types" of Java programs: applications and applets.

4.2.3.1 Applications: An application is a stand-alone Java program that can run outside of a Web browser, as long as a JVM is available.

4.2.3.2 Applets: An applet is a program written in Java to run within a Java-compatible Web browser. When they are embedded in a Web page, Java programs are called "applets." An applet provides a developer the flexibility to develop a more sophisticated user interface on a Web page. Java applets provide executable content, such as event-driven pop-up windows and graphical user interface (GUI) widgets, which can support a variety of applications.

Java is designed to allow applets to be downloaded and executed without fear of viruses. Some strict limits on applets are placed to prevent malicious actions [Tilley, 97].

4.2.4 JavaBean and JavaScript

JavaBeans are actually reusable software components [Cnet, 97]. They may be viewed as little chunks of code that perform single functions and can be mixed and matched to create complex applications. The market for beans is still new, so there aren't nearly as many available on the market. But Sun Microsystems hopes the cross-platform appeal of beans will change that [Cnet, 97].

JavaScript and Java have only a very loose relationship [Manning, 97]. Both are object-based but only Java is truly object-oriented. JavaScript began life at Netscape as a scripting language called LiveScript. Netscape designed LiveScript as a simple tool to help Web site developer design interactive Web pages. It was simplistic, but performed the necessary function of letting different parts of a Web page interact with each another. In 1995, Netscape and Sun joined forces to update LiveScript to JavaScript. JavaScript is an improvement over LiveScript, but it still isn't a full-fledged programming language like Java. JavaScript is simply an interpreted scripting language [Cnet, 97].

4.2.5 Future of Java

Sun wants its language to run on any device that has a microprocessor in it, such as servers, mobile phones, pagers, network computers, smart cards, and Internet set-top boxes like WebTV. Sun plans to deliver a Java platform that could challenge the Windows platform in the next decade. If Sun fails, Java will become another has-been technology. If they succeed, they might slow down Microsoft's dominance in the computer industry.

CHAPTER 5

ProWEB: THE WEB APPROACH

5.1 Overview

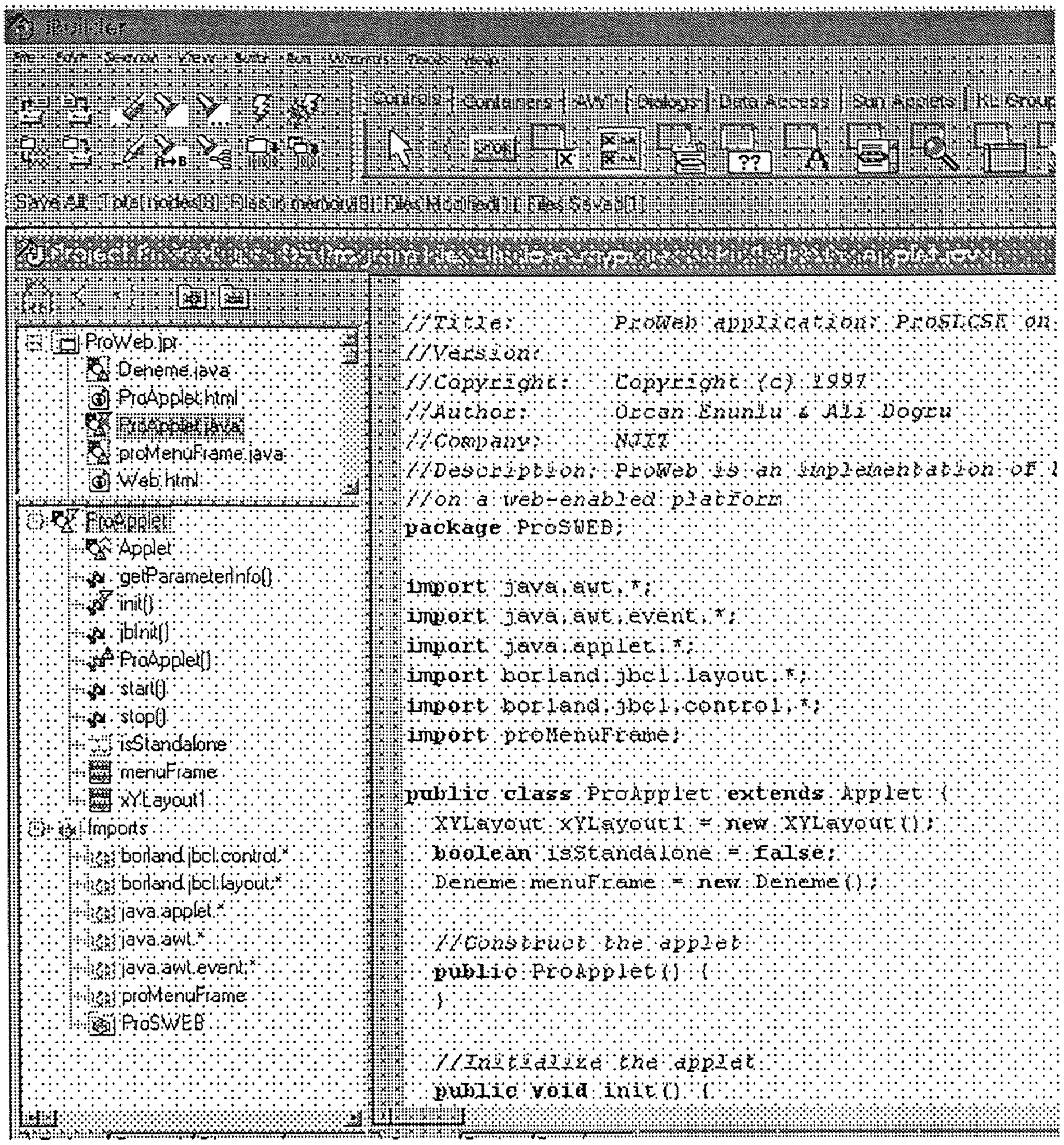
This chapter will give an overview of what we want to accomplish with the project in hand. It will outline the intentions, the tools, and the problems we have encountered. The section 5.2 will outline the features of JBuilder from Borland. This is the application that we have used to design and to program our work. Section 5.3 will display some screenshot of our actual work and future work to be accomplished. Section 5.4 will present the conclusion of this thesis.

5.2 JBuilder

We have started to design our project with JBuilder [Manning, 97]. JBuilder is Borland's (now called Inprise Corp.) visual development environment for Java. It includes a project browser, code editor, visual designer, component palette, property inspector, integrated debugger, and compiler. JBuilder is JDK 1.1 compliant.

JBuilder has incorporated Borland's "two-way Tools" (originally introduced in Delphi), which enables the users to switch between the source code and the visual design synchronized [Manning, 97]. Any changes made to the visual design are reflected on the source code. Any changes made to the source code are represented in the visual design, as well. This property gives the user the chance to experiment in both ways.

Borland pioneered the idea of an Integrated Development Environment (IDE), where the developer could create, compile, debug, and run a program all from the same interface. The Figure 5-1 presents JBuilder's Integrated Development Environment.



```
ProWeb.jpr
    Deneme.java
    ProApplet.html
    ProApplet.java
    proMenuFrame.java
    Web.html
```

```
ProApplet
    Applet
    getParameterInfo()
    init()
    jbInit()
    ProApplet()
    start()
    stop()
    isStandalone
    menuFrame
    xYLayout1
Imports
    borland.jbcl.control.*
    borland.jbcl.layout.*
    java.applet.*
    java.awt.*
    java.awt.event.*
    proMenuFrame
    ProSWEB
```

```
//Title:        ProWeb application: ProSLCSE on
//Version:
//Copyright:    Copyright (c) 1997
//Author:       Orcan Enunlu & Ali Dogru
//Company:      NJIT
//Description:  ProWeb is an implementation of 
//on a web-enabled platform
package ProSWEB;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;
import proMenuFrame;

public class ProApplet extends Applet {
  XYLayout xYLayout1 = new XYLayout();
  boolean isStandalone = false;
  Deneme menuFrame = new Deneme();

  //Construct the applet
  public ProApplet() {
  }

  //Initialize the applet
  public void init() {
```

Figure 5-1. JBuilder's Integrated Development Environment

5.3 ProWEB: WEB Version of ProSLCSE

ProWEB is the implementation of ProSLCSE on a web-enabled platform. We wanted to implement that tool on the Internet, so that developers could model their processes in a real environment. Although our version is for a single user, we intend to make modification so that several process modelers could work with each other no matter where they are. So far, the early version has the following capabilities:

- The user could select any of the construct mentioned in Chapter 3
- The user has the ability to display that construct on a frame.
- The user can select the construct from a toolbar
- The user can move the constructs on the frame
- The user can display the frame on Netscape

The reason why we focused on Netscape, but not on Microsoft's Internet Explorer or SUN's HotJava is that most of the browser are still not fully JDK compliant [Cnet, 97]. As of today, all the major browser such as HotJava, Netscape and IE are JDK compliant, but not at the full extend. New versions that are fully JDK compliant are on the shipment phase. Netscape's Netscape Communicator 4.04 or higher is supposed to be JDK compliant. And since the market seems to be revolving around Netscape for the moment, we chose to implement our applets on Netscape for the moment. Microsoft's growth rate in gaining the browser market has also got our attention. Future testing would also include IE4 or higher and HotJava. Figure 5-2 shows a sample output seen of ProWeb application run through an applet.

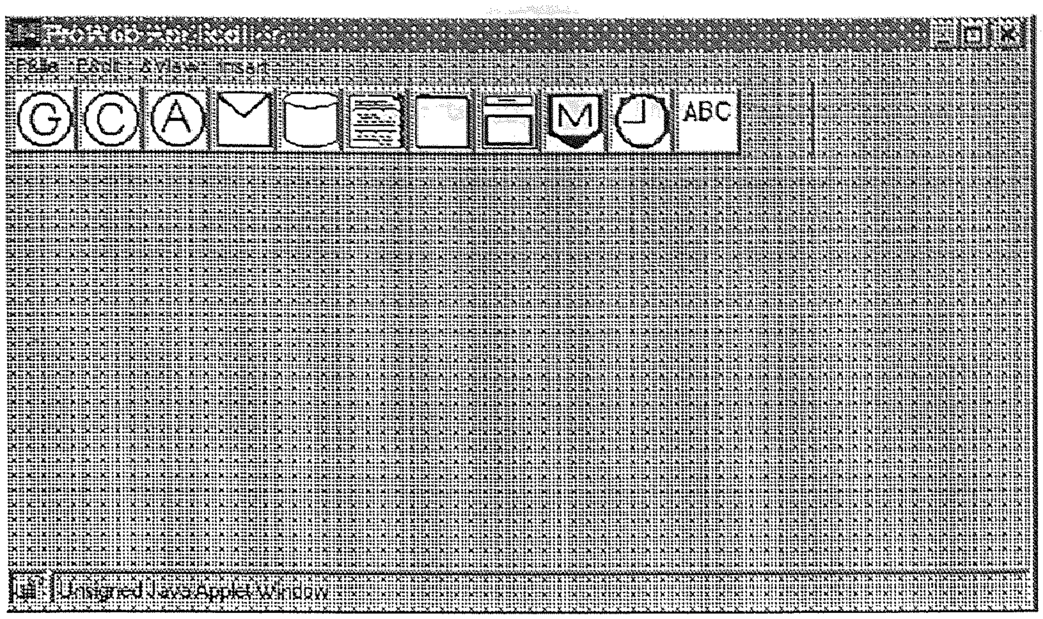


Figure 5-2. ProWEB interface

Multiple toolbars are possible to be displayed. The user can either select to view the Main Toolbar (displayed under the menu), the Connection Toolbar, and/or the Resource Toolbar.

The main menu bar gives the user a large number of commands necessary to perform the process editing tasks. The main menu is similar to that of other Windows applications. It contains menu items that are enabled and disabled in response to the current context. The following menus along with their menu items are listed below.

File: New, Open, Close, Save, Print, Exit

Edit: Cut, Copy, Paste, Delete

View: Active Toolbar, Resources Toolbar, Connections Toolbar

Insert: Person, Machine, Location, Tool, Group, Role, Machine Type, Location Type, Tool Type, Group Type.

Figure 5-3 shows the interface after selections are made from the button bar. As seen in Figure 5-3, objects are placed at different locations. They are moveable objects. The user has the possibility to place them on the screen. Several same components could be placed on the screen, as well. Appendix B contains the source code of the ProWEB application. It is made primarily of three Java source codes. One is labeled as 'ProApplet.java' which is the main applet. The second is labeled as 'Deneme.java' and contains the source code for the main frame. The third is 'ProsElement.java', which contain some of the classes.

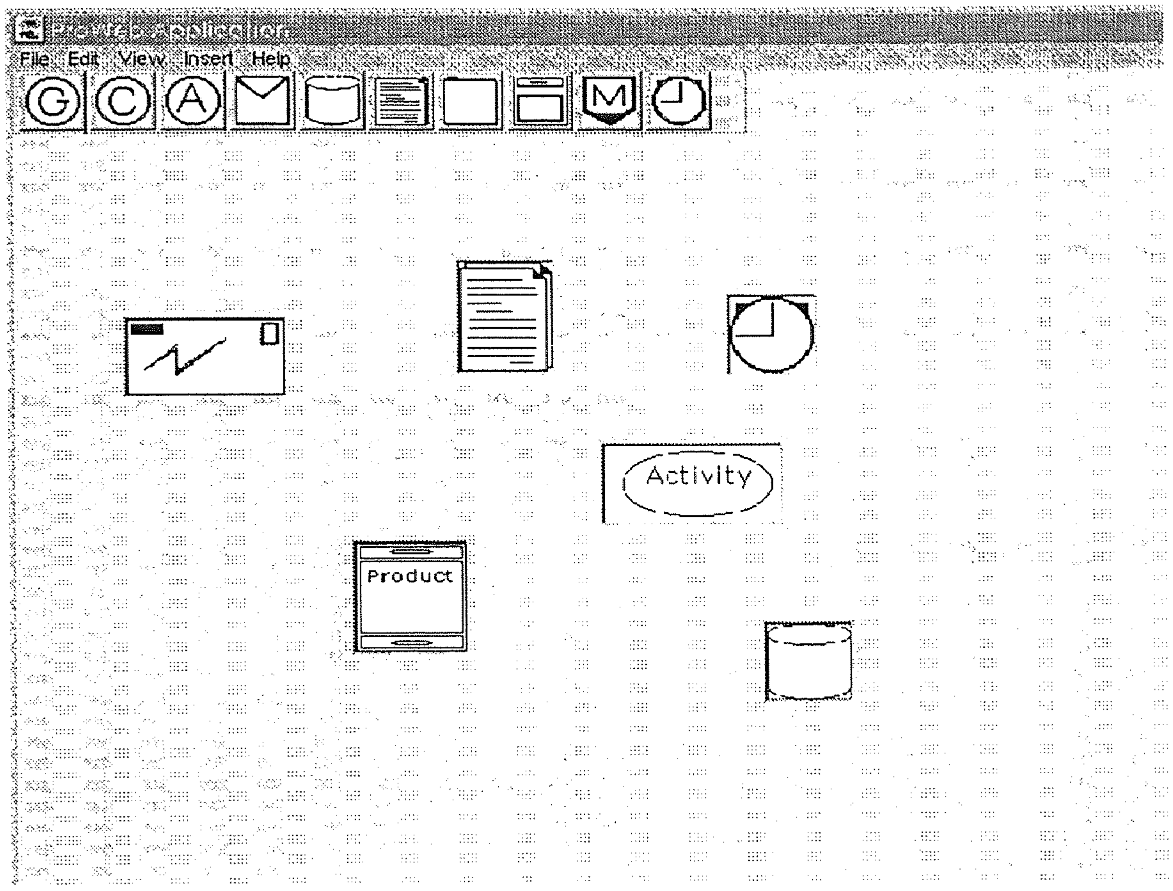


Figure 5-3. ProWEB with components placed on the screen

5.4 Conclusion

This thesis work can be summarized as a) the analysis of the Process Modeling concepts and a specific tool: ProSLCSE, b) the research on the enabling technologies for the utilization of internet, and c) the exploitation of networking notions for the distributed access of process modeling tools, particularly ProSLCSE.

Process Modeling concept has been aided with the introduction of distributed and concurrent access to a common model by different parties involved. In this framework, participators develop a model as a group task, or get involved in the evaluation or inspection of the model. The process model under development could only be treated for the business aspects of the enterprise, or for compliance with standards or regulations. A variety of purposes could be supported by allowing remote access to teams of users who are potentially located in disparate geographical locations. The notion introduced allows the development, evaluation, and inspection of a model in a faster , consistent, and a timely manner. For the demonstration of this notion, a web-enabled version of the ProSLCSE is implemented in a prototype setting.

APPENDIX A

GLOSSARY

In this appendix, some of the terms related to processes and process modeling are summarized. Some of the important terms used in Chapter 2 are also included.

Activity is a description of a piece of work that forms one logical step within a process. An activity may be manual, automated or composite, involving either human or machine participation for its completion.

And-split is a point where a single thread of control splits into two or more parallel activities.

And-join is a point where two or more parallel activities converge into a single common thread of control.

Application support technology includes tools used as part of the process by software developers, which are usually computerized. A compiler is an example of application support technology.

Automatic activities are described as activities, which can be carried out without human intervention. It is performed either by a computer script or program.

Capability maturity model is a maturity model available in the software community, which describes an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes.

Design process is defined as the ordered set of activities performed to transform system requirements into a design representation.

Key practices describe the steps to be taken to satisfy capability maturity model's key practice area goals. Common features are detailed by *key practices*.

Key process area is defined as a cluster of related activities, when collectively performed achieve a set of goals considered important for enhancing process capability.

Leaf activity is a general activity that is performed by one or more persons.

Organizational process is a set of activities, which collectively realize a business objective, or goal within the context of an organizational structure. Completion of each task within the process is dependent on the innovation and skills of the responsible individuals.

Or-join is a point where two or more alternative activities converge to a single activity.

Or-split is a point where a single thread of control leads to several branches, upon which a decision has to be made.

Process execution represents the time during which the process is operational, where process instances are being created and managed.

Process improvement includes analyzing the process, determining inefficiencies and ways to improve the process, and implementing those changes.

Process instance is the representation of a single enactment of a process

Process model is a formal representation of a process. This detailed representation of a process will lend itself to analysis and measurement.

Process modeling is defined as the activity of representing the process as a process model.

Process state represents the status of a process instance at a particular point in time.

Process support technology is defined to include tools in support of process model development and execution.

Quality component quantifies the range of a quality attribute.

Software process refers to the total set of activities required to implement a function in software along with its associated tools, methods, structure, and people.

Software process maturity is defined, as the extent to which a software process is explicitly defined, managed, measured, controlled, and effective.

Workflow is the automation of a process, in whole or in part, during which information or tasks are passed from one user to another, according to a set of rules.

Workflow management system is a system that defines, creates and manages the execution of workflows through the use of software.

APPENDIX B
SOURCE CODES

In this appendix, source codes for the ProWEB application are presented. There are three source codes attached in this section. The first source code labeled 'ProApplet.java' is the applet that generate other frames, such as the 'Deneme.java'. The Java file named 'Deneme.java' has the code that generates the frame that holds the menus. The third source code labeled 'ProSElement.java' holds some of the classes that 'Deneme.java' needs.

ProApplet.java

```
//Title:       ProWeb application: ProSLCSE on WEB
//Version:
//Copyright:   Copyright (c) 1997
//Author:      Orcan Enunlu & Ali Dogru
//Company:     NJIT
//Description: ProWeb is an implementation of ProSLCSE
//on a web-enabled platform
package ProSWEB;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;
import Deneme;

public class ProApplet extends Applet {
  XYLayout xYLayout1 = new XYLayout();
  boolean isStandalone = false;
  Deneme menuFrame = new Deneme();


  //Construct the applet
  public ProApplet() {
  }

  //Initialize the applet
  public void init() {
    try { jbInit(); } catch (Exception e) { e.printStackTrace(); }
  }

  //Component initialization
  private void jbInit() throws Exception{
    xYLayout1.setWidth(800);
    xYLayout1.setHeight(600);
    this.setLayout(xYLayout1);
  }

  //Start the applet
  public void start() {
     menuFrame.setVisible(true);
     menuFrame.setLocation(100,100);
     menuFrame.setSize(800,600);
  }

  //Stop the applet
  public void stop() {
     menuFrame.setVisible(false);
  }

  //Destroy the applet
  public void destroy() {
 }
 //Get Applet information
 //public String getAppletInfo() {
   // return "Applet Information";
 //}

  //Get parameter info
  public String[][] getParameterInfo() {
    return null;
  }
}
```

Deneme.java

```
//Title:        ProWeb application: ProSLCSE on WEB
//Version:
//Copyright:    Copyright (c) 1997
//Author:       Orcan Enunlu & Ali Dogru
//Company:      NJIT
//Description:  ProWeb is an implementation of ProSLCSE
//on a web-enabled platform


package ProSWEB;

import java.awt.*;
import java.awt.event.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;


public class Deneme extends DecoratedFrame {
  ProSElement LastElement, CurrentElement;
  XYLayout xYLayout1 = new XYLayout();
  XYConstraints CurrentXYC;
  boolean created = false;
  ButtonBar MainButtonBar = new ButtonBar();
  MenuBar menuBar1 = new MenuBar();
  Menu FileMenu = new Menu();
  MenuItem MenuFileNew = new MenuItem();
  MenuItem MenuFileOpen = new MenuItem();
  MenuItem MenuFileClose = new MenuItem();
  MenuItem MenuFilePrint = new MenuItem();
  MenuItem MenuFileExit = new MenuItem();
  Menu EditMenu = new Menu();
  MenuItem MenuEditCut = new MenuItem();
  MenuItem MenuEditCopy = new MenuItem();
  MenuItem MenuEditPaste = new MenuItem();
  MenuItem MenuEditDelete = new MenuItem();
  MenuItem MenuFileSave = new MenuItem();
  Menu ViewMenu = new Menu();
  MenuItem MenuViewAct = new MenuItem();
  MenuItem MenuViewRes = new MenuItem();
  MenuItem MenuViewCon = new MenuItem();
  Menu InsertMenu = new Menu();
  Menu HelpMenu = new Menu();
  MenuItem MenuHelpAbout = new MenuItem();

 // Canvas canvas = new Canvas();

  public Deneme() {
    try {
      jbInit();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception{
    this.setTitle("ProWeb Application");
    this.addMouseListener(new Deneme_this_mouseAdapter(this));
    xYLayout1.setHeight(496);
    xYLayout1.setWidth(562);
    MainButtonBar.setHgap(2);
    MainButtonBar.setBackground(SystemColor.scrollbar);
    MainButtonBar.setButtonType(ButtonBar.IMAGE_ONLY);
    MainButtonBar.setImageBase("C:\\JBuilder\\myclasses\\GIF_Images");
    MainButtonBar.setLabels(new String[] {"Activity",
                                          "Comp_act",
                                          "Auto_act",
                                          "Message",
                                          "Artifact",
                                          "Document",
                                          "Folder",
                                          "Comp_prod",
                                          "Milestone",
                                          "Timer",
                                          "Text"});
    MainButtonBar.setSoft(true);
    MainButtonBar.setMargins(new Insets(1, 1, 1, 1));

    MainButtonBar.addActionListener(new
Deneme_MainButtonBar_actionAdapter(this));

    //Main menu labeling starts here.../
    FileMenu.setLabel("File");
    FileMenu.setActionCommand("File");
    MenuFileNew.setLabel("New");
    MenuFileOpen.setLabel("Open");
    MenuFileClose.setLabel("Close");
    MenuFileClose.setActionCommand("Close");
    MenuFilePrint.setLabel("Print");
    MenuFileExit.setLabel("Exit");
    MenuFileExit.addActionListener(new
Deneme_MenuFileExit_actionAdapter(this));
    EditMenu.setLabel("Edit");
    MenuEditCut.setLabel("Cut");
    MenuEditCopy.setLabel("Copy");
    MenuEditPaste.setLabel("Paste");
    MenuEditDelete.setLabel("Delete");
    MenuFileSave.setLabel("Save");
    MenuFileSave.setActionCommand("Save");
    ViewMenu.setLabel("View");
    MenuViewAct.setLabel("Activities Toolbar");
    MenuViewRes.setLabel("Resources Toolbar");
    MenuViewCon.setLabel("Connections Toolbar");
    InsertMenu.setLabel("Insert");
    HelpMenu.setLabel("Help");
    MenuHelpAbout.setLabel("About");

    //main button bar images entered here.../
    MainButtonBar.setImageNames(new String[] {"Activity_bt.gif",
                                              "Comp_act_bt.gif",
                                              "Auto_act_bt.gif",
                                                  "Message_bt.gif",
                                                  "Artifact_bt.gif",
                                                  "Document_bt.gif",
                                                  "Folder_bt.gif",
                                                  "Comp_prod_bt.gif",
                                                  "Milestone_bt.gif",
                                                  "Timer_bt.gif",
                                                  "Text_bt.gif"}
                                                  );

    //start adding the components to the this layout.../
    this.setLayout(xYLayout1);
    menuBar1.add(FileMenu);
    menuBar1.add(EditMenu);
    menuBar1.add(ViewMenu);
    menuBar1.add(InsertMenu);
    menuBar1.add(HelpMenu);
    FileMenu.add(MenuFileNew);
    FileMenu.add(MenuFileOpen);
    FileMenu.addSeparator();
    FileMenu.add(MenuFileClose);
    FileMenu.add(MenuFileSave);
    FileMenu.addSeparator();
    FileMenu.add(MenuFilePrint);
    FileMenu.addSeparator();
    FileMenu.add(MenuFileExit);
    EditMenu.add(MenuEditCut);
    EditMenu.add(MenuEditCopy);
    EditMenu.add(MenuEditPaste);
    EditMenu.add(MenuEditDelete);
    ViewMenu.add(MenuViewAct);
    ViewMenu.add(MenuViewRes);
    ViewMenu.add(MenuViewCon);
    HelpMenu.add(MenuHelpAbout);
    this.add(MainButtonBar, new XYConstraints(0, -2, 401, -1));
    this.setBackground(new Color(233, 244, 235));
    this.setEnabled(true);

this.setIconImageName("C:\\JBuilder\\myclasses\\GIF_Images\\deneme.gif")
;
    this.setMenuBar(menuBar1);
    this.setResizable(true);
    this.setSize(new Dimension(500, 400));
  }


  //Activity call function entered here.../
  void ActivityCall() {
    try { ActivityInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

    void ActivityInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,100,50);
      Activity activity1 = new Activity(constraint);
      CurrentElement = activity1;
      CurrentXYC = constraint;
//    activity1.addComponentListener(new
Deneme_activity1_componentAdapter(this));
      activity1.addActionListener(new
Deneme_element_actionAdapter(this));
      activity1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      activity1.addMouseListener(new Deneme_element_mouseAdapter(this));
      activity1.setVisible(false);
  };
//ActivityCall() ended here


//Composite activity call starts here
  void Comp_actCall(){
    try { Comp_actInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void Comp_actInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,100,50);
      Comp_act comp_act1 = new Comp_act(constraint);
      CurrentElement = comp_act1;
      CurrentXYC = constraint;
      comp_act1.addActionListener(new
Deneme_element_actionAdapter(this));
      comp_act1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      comp_act1.addMouseListener(new Deneme_element_mouseAdapter(this));
      comp_act1.setVisible(false);
  };
//..Comp_actCall() ended


//Automatic activity call starts here
  void Auto_actCall(){
    try { Auto_actInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void Auto_actInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,100,50);
      Auto_act auto_act1 = new Auto_act(constraint);
      CurrentElement = auto_act1;
      CurrentXYC = constraint;
      auto_act1.addActionListener(new
Deneme_element_actionAdapter(this));
      auto_act1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
```

```
        auto_act1.addMouseListener(new Deneme_element_mouseAdapter(this));
        auto_act1.setVisible(false);
    };
//..Auto_actCall() ended


//Message call function entered here../
    void MessageCall(){
     try { MessageInit();
      }
        catch (Exception e) {
         e.printStackTrace();
        };
    };

    void MessageInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,90,50);
        Message message1 = new Message(constraint);
        CurrentElement = message1;
        CurrentXYC = constraint;
        message1.addActionListener(new
Deneme_element_actionAdapter(this));
        message1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
        message1.addMouseListener(new Deneme_element_mouseAdapter(this));
        message1.setVisible(false);
    };
//..MessageCall() ended


//Artifact call function entered here../
    void ArtifactCall(){
      try { ArtifactInit();
      }
        catch (Exception e) {
         e.printStackTrace();
        };
    };

    void ArtifactInit() throws Exception {
        XYConstraints constraint = new XYConstraints (100,100,50,50);
        Artifact artifact1 = new Artifact(constraint);
        CurrentElement = artifact1;
        CurrentXYC = constraint;
        artifact1.addActionListener(new
Deneme_element_actionAdapter(this));
        artifact1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
        artifact1.addMouseListener(new Deneme_element_mouseAdapter(this));
        artifact1.setVisible(false);
    };
//..ArtifactCall() ended


//Document call function entered here../
    void DocumentCall(){
      try { DocumentInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void DocumentInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,55,70);
      Document document1 = new Document(constraint);
      CurrentElement = document1;
      CurrentXYC = constraint;
      document1.addActionListener(new
Deneme_element_actionAdapter(this));
      document1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      document1.addMouseListener(new Deneme_element_mouseAdapter(this));
      document1.setVisible(false);
  };
//..DocumentCall() ended


//Folder call function entered here../
  void FolderCall(){
    try { FolderInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void FolderInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100, 85,60);
      Folder folder1 = new Folder(constraint);
      CurrentElement = folder1;
      CurrentXYC = constraint;
      folder1.addActionListener(new Deneme_element_actionAdapter(this));
      folder1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      folder1.addMouseListener(new Deneme_element_mouseAdapter(this));
      folder1.setVisible(false);
  };
//..FolderCall() ended


//Composite product call function entered here../
  void Comp_prodCall(){
    try { Comp_prodInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void Comp_prodInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,65,70);
      Comp_prod comp_prod1 = new Comp_prod(constraint);
      CurrentElement = comp_prod1;
      CurrentXYC = constraint;
      comp_prod1.addActionListener(new
Deneme_element_actionAdapter(this));
      comp_prod1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      comp_prod1.addMouseListener(new
Deneme_element_mouseAdapter(this));
      comp_prod1.setVisible(false);
  };
//..Comp_prodCall() ended


//Milestone call function entered here../
  void MilestoneCall(){
    try { MilestoneInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void MilestoneInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,70,100);
      Milestone milestone1 = new Milestone(constraint);
      CurrentElement = milestone1;
      CurrentXYC = constraint;
      milestone1.addActionListener(new
Deneme_element_actionAdapter(this));
      milestone1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      milestone1.addMouseListener(new
Deneme_element_mouseAdapter(this));
      milestone1.setVisible(false);
  };
//..MilestoneCall() ended


//Timer call function entered here../
  void TimerCall(){
    try { TimerInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void TimerInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,50,50);
      Timer timer1 = new Timer(constraint);
      CurrentElement = timer1;
      CurrentXYC = constraint;
      timer1.addActionListener(new Deneme_element_actionAdapter(this));
      timer1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      timer1.addMouseListener(new Deneme_element_mouseAdapter(this));
      timer1.setVisible(false);
  };
```

```
//..TimerCall() ended


//Text call function entered here../
  void TextCall(){
    try { TextInit();
    }
      catch (Exception e) {
       e.printStackTrace();
      };
  };

  void TextInit() throws Exception {
      XYConstraints constraint = new XYConstraints (100,100,50,100);
      Text text1 = new Text(constraint);
      CurrentElement = text1;
      CurrentXYC = constraint;
      text1.addActionListener(new Deneme_element_actionAdapter(this));
      text1.addMouseMotionListener(new
Deneme_element_mouseMotionAdapter(this));
      text1.addMouseListener(new Deneme_element_mouseAdapter(this));
      text1.setVisible(false);
  };
//..TextCall() ended

//--------------------------------------------------------------//

// MAin Button bar action events
  void MainButtonBar_actionPerformed(ActionEvent e) {
    created = true;
    String actionCommand = e.getActionCommand();
    // selection depends on actionCommand, a function is called
    if (actionCommand.equals("Activity")) {
       ActivityCall();
    }
    else if (actionCommand.equals("Comp_act")){
         Comp_actCall();
    }
    else if (actionCommand.equals("Auto_act")){
         Auto_actCall();
    }
    else if (actionCommand.equals("Message")){
         MessageCall();
    }
    else if (actionCommand.equals("Artifact")){
         ArtifactCall();
    }
    else if (actionCommand.equals("Document")){
         DocumentCall();
    }
    else if (actionCommand.equals("Folder")){
         FolderCall();
    }
    else if (actionCommand.equals("Comp_prod")){
         Comp_prodCall();
    }
    else if (actionCommand.equals("Milestone")){
          MilestoneCall();
     }
     else if (actionCommand.equals("Timer")){
          TimerCall();
     }
     else if (actionCommand.equals("Text")){
          TextCall();
     }
   }
   // end of main buttonbar selection


/*=================================================================/
/============= the mouse events methods which are called =============/
/========== are listed below, also other methods are listed===========/
/=================================================================*/

   void MenuFileExit_actionPerformed(ActionEvent e) {
      System.exit(0);
   }

   void this_mouseClicked(MouseEvent e) {
     if (created) {
       CurrentElement.setLocation(e.getPoint());
       CurrentElement.XYC.setX(e.getPoint().x -4);//-4: left border width
       CurrentElement.XYC.setY(e.getPoint().y - 39);// 42: topborder +
menubar height
       this.add(CurrentElement, CurrentXYC);
       CurrentElement.setVisible(true);
       show();
       created = false;
       }
   }

////  activity1 controls and methods starts here --------//
   void element_mouseDragged(MouseEvent e) {
       ProSElement elem = (ProSElement) e.getComponent();
       Point p1 = e.getPoint();
       Point p2 = elem.getLocation();
       Point p3 = elem.getClickPoint();
       elem.setLocation(p2.x + p1.x - p3.x, p2.y + p1.y - p3.y);
       elem.XYC.setX(p2.x + p1.x - p3.x - 4); // 4: left border width
       elem.XYC.setY(p2.y + p1.y - p3.y - 39);   //42:topborder + menubar
height
   }

   void element_mouseClicked(MouseEvent e) {
//      CurrentElement.putClickPoint(e.getPoint());
   }

   void element_mousePressed(MouseEvent e) {
       ProSElement elem = (ProSElement) e.getComponent();
       elem.putClickPoint(e.getPoint());
   }

   void element_actionPerformed (ActionEvent e) {
   }
```

```
/// -------- element (previously called activity1) methods end here ----
-------//

}


/*=================================================================/
/======= Classes that mouse events have created are defined below
======/
/=================================================================*/

//special classes defined in this section (3 so far)
class Deneme_MainButtonBar_actionAdapter implements
java.awt.event.ActionListener{
  Deneme adaptee;

  Deneme_MainButtonBar_actionAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.MainButtonBar_actionPerformed(e);
  }
}

class Deneme_MenuFileExit_actionAdapter implements
java.awt.event.ActionListener {
  Deneme adaptee;

  Deneme_MenuFileExit_actionAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.MenuFileExit_actionPerformed(e);
  }
}

class Deneme_this_mouseAdapter extends java.awt.event.MouseAdapter {
  Deneme adaptee;

  Deneme_this_mouseAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void mouseClicked(MouseEvent e) {
    adaptee.this_mouseClicked(e);
  }
}
//end of special classes//


// CLASSES DEFINED FOR All elements...EL1.============================/
// generally called 'element', which applies to all elements of
// main toolbar.
class Deneme_element_mouseAdapter extends java.awt.event.MouseAdapter {
  Deneme adaptee;
  Deneme_element_mouseAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void mouseClicked(MouseEvent e) {
    adaptee.element_mouseClicked(e);
  }
  public void mousePressed (MouseEvent e) {
    adaptee.element_mousePressed(e);
  }
}

class Deneme_element_mouseMotionAdapter extends
java.awt.event.MouseMotionAdapter {
  Deneme adaptee;

  Deneme_element_mouseMotionAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void mouseDragged(MouseEvent e) {
    adaptee.element_mouseDragged(e);
  }
}

class Deneme_element_actionAdapter implements
java.awt.event.ActionListener {
  Deneme adaptee;

    Deneme_element_actionAdapter(Deneme adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.element_actionPerformed(e);
  }
}
//...end of EL1. end of classes for all elements //
```

ProSElement.java

```
//Those are the classes that are called by Deneme.java
//It returns the mouse click on the frame as the ClickPoint
//the images that are displayed on the frame are listed in the
//subclasses of ProSElement

package ProSWEB;

import java.awt.*;
import java.awt.event.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;
import java.lang.*;


  public class ProSElement extends ImageControl{
  public XYConstraints XYC;

  Point ClickPoint;
  public ProSElement(){
  };
  public Point getClickPoint(){
    return ClickPoint;
  }
  public void putClickPoint(Point p){
    ClickPoint = p;
  }
}

class Activity extends ProSElement{
  public Activity(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Activity.gif");
    XYC = xyc;
  }
};


class Comp_act extends ProSElement{
  public Comp_act(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
```

```
public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Comp_act.gif");
    XYC = xyc;
  }
};

class Auto_act extends ProSElement{
  public Auto_act(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Auto_act.gif");
    XYC = xyc;
  }
};

class Message extends ProSElement{
  public Message(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Message.gif");
    XYC = xyc;
  }
};

class Document extends ProSElement{
  public Document(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Document.gif");
    XYC = xyc;
  }
};

class Folder extends ProSElement{
  public Folder(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Folder.gif");
    XYC = xyc;
  }
};

class Comp_prod extends ProSElement{
  public Comp_prod(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Comp_prod.gif");
    XYC = xyc;
  }
};

class Milestone extends ProSElement{
  public Milestone(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Milestone.gif");
    XYC = xyc;
  }
};

class Artifact extends ProSElement{
  public Artifact(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Artifact.gif");
    XYC = xyc;
  }
};

class Timer extends ProSElement{
  public Timer(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Timer.gif");
    XYC = xyc;
  }
};

class Text extends ProSElement{
  public Text(XYConstraints xyc) {
    try {
      init(xyc);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  };
  public void init(XYConstraints xyc)throws Exception{
    setImageName("C:\\JBuilder\\myclasses\\GIF_Images\\Text.gif");
    XYC = xyc;
  }
};
```

REFERENCES

[Anonymous, 96]	"Workflow Management Coalition, Terminology & Glossary," Document number WFMC-TC-1011, Issue 2.0, June 1996.
[Armitage, 94]	Armitage, J. W. et. al., "Software Process Definition Guide: Content of Enactable Software Process Representations," CMU/SEI Special Report 94-SR-21, SEI/CMU 1994.
[Cnet, 97]	Cnet Staff, "20 Questions about Java," http://www.cnet.com/Content/Features/Techno/Java20/index.html (31 July 1997).
[Curtis, 92]	Curtis, B., Kellner, M. I., and Over, J., "Process Modeling," Communications of the ACM, vol. 35, Sept. 1992, pp.75-90.
[Curtis, 95]	Curtis, B., "Software Process Improvement: Methods and Lessons Learned," Tutorial at ICSE-17, 1995.
[Davis, 93]	Davis, A. M., Software Requirements: Objects, Functions, and States, Prentice Hall PTR, Englewood, NJ, 1993.
[Denning, 95]	Denning, P. J., Hieb, M. R., and Menasce, D. A., "Introduction to Workflow," Workflow Management Systems, http://cne.gmu.edu/modules/workflow/workflow-intro.html (13 April 1998).
[Donaton, 96]	Donaton, S., "Standards required to make the next leap," Advertising Age, vol. 67, Iss. 45, Nov 4, 1996.
[Heffley, 95]	Heffley, B. and Morris, J., "An Introduction to the Internet and the World Wide Web," ACM, May 1995, pp. 365-366.
[Herbsleb, 97]	Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M., "Software Quality and the Capability Maturity Model," Communications of the ACM, vol. 40, June 1997, pp.30-40.

**REFERENCES
(Continued)**

[Hierhager, 96] Hierhager, A., "Web videoconferencing making waves," Electronic Engineering Times, Iss. 928, Nov 18, 1996: pp.124, 154.

[Hoffman, 96] Hoffman, D. L., Kalsbeek, W. D., and Novak, T. P., "Internet and the Web Use in the U.S.," Communications of the ACM, vol. 39, Dec 1996, pp.36-46.

[Hollingsworth, 94] Hollingsworth, D., "The Workflow Reference Model," Workflow Management Coalition, Document number TC00-1003, Issue 1.1, Nov 1994.

[Humphrey, 89] Humphrey, W. S. and Kellner, M. I., "Software Process Modeling: Principles of Entity Process Models," Proceedings of the 11th International Conference on Software Engineering, 15-18 May, 1989.

[Humphrey, 95] Humphrey, W. S., A Discipline for Software Engineering, Addison-Wesley, Reading, Massachusetts, NY, 1995.

[IEEE, 90] Institute of Electrical and Electronics Engineers. IEEE standard glossary of Software Engineering Terminology. IEEE standard 610.12-1990.

[Kellner, 88] Kellner, M. I., "Representation Formalism for Software Process Modeling," Proceedings of the 4th International Software Process Workshop, ACM press, 1988, pp. 93-96.

[Krasner, 92] Krasner, H., Terrel, J., Linehan, A., Arnold, P., and Ett, W.H., "Lessons Learned from a Software Process Modeling System," Communications of the ACM, vol. 35, Sept 1992, pp.91-100.

[Lassenius, 97] Lassenius, C. "Process Modeling Issues," http://mordor.cs.hut.fi/~cls/tik-76.631/johdanto/ (10 March 1998).

REFERENCES
(Continued)

[Leiner, 98]	Leiner, B. M, et al. "A Brief History of the Internet," http://www.isoc.org/internet/history/brief.html (12 May 1998).
[Madhavji, 91]	Madhavji, N. H., "The Process Cycle," Software Engineering Journal, Sept 1991, pp.234-242.
[Manning, 97]	Manning, M. M., Borland JBuilder in 21 Days, Borland Press, Indiana, 1997.
[Medina, 92]	R. Medina-Mora, T. Winograd, R. Flores, and F. Flores, "The Action Workflow Approach to Workflow Management Technology," CSCW Proceedings, Nov 1992, pp. 281-288.
[Osterweil, 87]	Osterweil, L., "Software Processes are Software too," Proceedings of the Ninth International Conference on Software Engineering, IEEE Computer Society, Washington, DC, 1987, pp.2-13.
[Paulk, 93]	Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V. "The Capability Maturity Model For Software, Version 1.1", Software Engineering Institute, CMU, Pittsburgh, 1993 http://www2.umassd.edu/SWPI/sei/tr24f/tr24.html (22 April 1998).
[ProSLCSE, 94]	ProSLCSE User's Manual, ISSI, Austin, Texas, 1994.
[Spar, 96]	Spar, D., Bussgang, J. J., " Ruling the net." Harvard Business Review, vol. 74, May/June 1996, pp.125-133.
[Tanenbaum, 96]	Tanenbaum, A. S., "Computer Networks," 3rd Edition, Prentice Hall PTR, NJ, 1996.
[Tanik, 93]	Tanik, M. M., and Delcambre, S. N., "A Proposed Process Modeling Framework," Southern Methodist University, 1993.

REFERENCES (Continued)

[Teixeira, 93]	Teixeira, D. and Thompson, J., "The Power of Work Flow Software," The Bankers Magazine, Sept/Oct, 1993, pp.10-14.
[Tilley, 97]	Tilley, S., "Java," http://www.sei.cmu.edu/str/descriptions/java.html (23 April 1998).
[Webster, 83]	Webster, N., Webster's New Twentieth Century Dictionary, 2nd. Edition, Simon and Shuster, NY.
[Winograd, 87]	Winograd, T. and Flores, F., Understanding Computers and Cognition, Addisson-Wesley, 1987.
[Zakon, 98]	Zakon, R. H., "Hobbes' Internet Timeline," http://www.isoc.org/zakon/Internet/History/HIT.html (12 April 1998).