

## INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600

UMI<sup>®</sup>



University of Alberta

ECOPRO: AN EXERCISE IN MODEL-BASED INTERACTIVE SYSTEM DESIGN

by

Zhaohui Zhong ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta  
Spring 2001



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-60520-5

**Canada**

University of Alberta

Library Release Form

Name of Author: Zhaohui Zhong

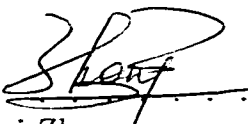
Title of Thesis: Ecopro: An Exercise in Model-Based Interactive System Design

Degree: Master of Science

Year this Degree Granted: 2001

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

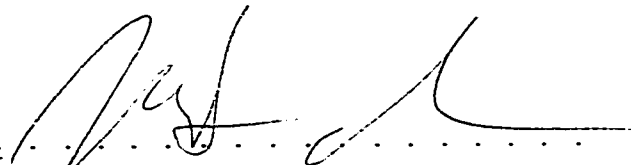
  
.....  
Zhaohui Zhong  
818-10 Edgecliff Golfway  
Toronto, ON, M3C 3A3


Date: Nov. 6, 2000

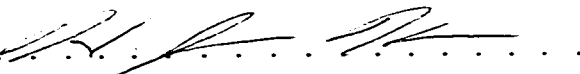
University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Ecopro: An Exercise in Model-Based Interactive System Design** submitted by Zhaohui Zhong in partial fulfillment of the requirements for the degree of **Master of Science**.

  
.....  
Mark Green (Supervisor)

  
.....  
Robert Grant (External)

  
.....  
Jim Hoover (Internal)

Date: *Nov. 3, 2000*  
.....

# Abstract

This thesis describes an implementation of a model-based development process using a set of design techniques throughout the system life cycle. The development of the system is based on the integration of a task model, a dialogue model and an implementation model.

The entire process begins with task analysis, which takes user requirements and produces a set of tasks to be supported by the design. After the initial task model is defined, the views for the objects in the task model are identified for supporting mechanisms to achieve the various tasks. Once objects and views are identified, implementation details are specified for system programming. As an iterative methodology, users participate in the design, changes in requirements are handled, and designs are refined through the entire design process.

# Acknowledgements

I would like to thank everyone who supports me all my past two years in University of Alberta, especially my supervisor Dr. Mark Green and co-supervisor Dr. Robert Grant, who encourage me throughout this project.

At last, and for the most, I want to thank my parents who are proud of me and give me love and support forever.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Outline of the thesis . . . . .	2
<b>2</b>	<b>Background Overview</b>	<b>3</b>
2.1	An introduction to ecosys . . . . .	3
2.2	Interactive system . . . . .	5
2.3	The design world . . . . .	7
2.4	Methodology overview . . . . .	7
2.4.1	User centered design . . . . .	7
2.4.2	Design based on models . . . . .	8
2.4.3	Design techniques . . . . .	8
2.5	Design process . . . . .	10
2.5.1	Models used in Ecopro . . . . .	10
2.5.2	Design process . . . . .	14
<b>3</b>	<b>User Requirements and Task Analysis</b>	<b>21</b>
3.1	Requirements definition . . . . .	21
3.2	User profile analysis . . . . .	21
3.3	Task model generation . . . . .	24
3.3.1	User task model development . . . . .	24
3.3.2	Initial task model generation . . . . .	27
3.3.3	Envisioned task model generation . . . . .	35
3.3.4	Scenarios test . . . . .	39
<b>4</b>	<b>Dialogue Design and Design Evaluation</b>	<b>44</b>
4.1	Syntactic level of dialogue modeling . . . . .	44
4.2	Semantic level of dialogue modeling . . . . .	46
4.3	Interaction style . . . . .	50
4.4	Lexical level of dialogue modeling . . . . .	52
4.5	Design evaluation . . . . .	55
<b>5</b>	<b>System implementation</b>	<b>61</b>
5.1	Basic classes in Ecopro . . . . .	61
5.1.1	Identification of the main classes . . . . .	61

5.1.2	Identification of the semantic component of each main class . . . . .	62
5.1.3	Identification of interactions among main classes . . . . .	63
5.2	Subclasses . . . . .	64
5.2.1	Subclasses of Model . . . . .	66
5.2.2	Subclasses of View . . . . .	69
5.2.3	Subclasses of Controller . . . . .	71
5.3	Interactions among subclasses . . . . .	71
<b>6</b>	<b>Usability Testing</b>	<b>75</b>
6.1	Thinking aloud . . . . .	75
6.2	User survey . . . . .	76
<b>7</b>	<b>Conclusion and future work</b>	<b>79</b>
7.1	Conclusion . . . . .	79
7.2	Future Work . . . . .	80

# List of Figures

2.1	Ecopro structure . . . . .	5
2.2	A design world . . . . .	6
2.3	Various techniques involved in user interface design . . . . .	9
2.4	Ecopro models and design process . . . . .	12
2.5	Model-View-Controller state and message sending . . . . .	19
3.1	First level of task analysis in Ecopro . . . . .	25
3.2	More levels of task analysis in Ecopro . . . . .	26
3.3	First level of task analysis after refinement . . . . .	28
3.4	Task model of adding input files . . . . .	29
3.5	Task model of inputting topographic file . . . . .	30
3.6	Task model of inputting one scenario . . . . .	31
3.7	Task model of modifying one scenario . . . . .	32
3.8	Task model of modifying input files . . . . .	33
3.9	Task model of invoking ecosys . . . . .	34
3.10	Task model of visualizing output data . . . . .	35
3.11	“User” object state diagram in envisioned task model . . . . .	40
3.12	“Evaluation process” object state diagram in envisioned task model . . . . .	41
3.13	Object relationship diagram in envisioned task model . . . . .	42
4.1	Object relationship diagram in dialogue modeling (I) . . . . .	47
4.2	ATN diagram of initial “user” view . . . . .	48
4.3	ATN diagram of “evaluation process” view . . . . .	49
4.4	ATN diagram of “user view” after completeness and reachability check . . . . .	51
4.5	Object relationship diagram in dialogue modeling (II) . . . . .	54
4.6	Prototyping of initial view of Ecopro (I) . . . . .	55
4.7	Prototyping of initial view of Ecopro (II) . . . . .	56
4.8	Prototyping of evaluation process view (I) . . . . .	56
4.9	Prototyping of evaluation process view (II) . . . . .	57
4.10	Prototyping of scenario view . . . . .	57
4.11	Prototyping of site file view (I) . . . . .	58
4.12	Prototyping of site file view (II) . . . . .	58
5.1	A high level class diagram of Model-View-Controller in Ecopro . . . . .	63
5.2	MVC triad set up process . . . . .	64
5.3	MVC triad work process . . . . .	65

5.4	Class diagram of subclasses of Model in Ecopro (I) . . . . .	67
5.5	Class diagram of subclasses of Model in Ecopro (II) . . . . .	68
5.6	Class diagram of subclasses of View in Ecopro . . . . .	70
5.7	Class diagram of subclasses of Controller in Ecopro . . . . .	72
5.8	Collaboration diagram of “ input file ” scenario in Ecopro . . . . .	73

# List of Tables

2.1	Level of dialogue modeling . . . . .	16
3.1	Initial user requirement of Ecopro . . . . .	22
3.2	Applying user profile on design . . . . .	23
3.3	Sample questions in user interview . . . . .	25
3.4	Usability considerations in Ecopro . . . . .	37
3.5	General rules for task allocation in Ecopro . . . . .	38

# Chapter 1

## Introduction

### 1.1 Motivation

The rapid growth and spread of computer software has raised the important issue of providing effective support for the end users. In the past, little attention was paid to the user interface, because both software developers and customers focused on maximum functionality within cost and performance constraints. Recently, however, there has been a growing awareness that the success of computer software depends to a significant degree on the quality of the user interface. This thesis explores the principles and choices in developing a graphical user interface (GUI) for an existing software system.

The system for which the GUI is built is called ecosys. The long-term objective of ecosys is to provide means for predicting ecosystem behavior under different environmental conditions. The problem with the existing implementation of Ecopro is that it is not user friendly for the following reasons:

- The user needs to have advanced knowledge of the input files that are based on different scenarios. For example, the user needs to remember the file format of each input file and understand the relationship among different files.
- In some cases, the amount of information input could be enormous.
- The output files are not straightforward, thus they are hard for the user to understand.
- The user needs to have certain computer background.

The disadvantages discussed above motivated the development of an interactive, user-friendly system called Ecopro. This system aims to:

- make all intricate system details transparent to the user.
- provide support to the user in system input.
- help user understand system output.
- maintain a robust system structure for ecosys.

## 1.2 Outline of the thesis

The following is a brief outline that describes the thesis structure:

Chapter 1 - Introduction

Chapter 2 - Background overview

A short description of the interactive system design methodology and the models used in implementing Ecopro.

Chapter 3 - User requirements and tasks analysis

This chapter provides answers to questions directed towards understanding the user's needs - a key requirement for designing and implementing Ecopro. The general steps include contacting the user, analyzing and modeling their tasks, as well as defining system constraints and usability targets.

Chapter 4 - Interface design, prototyping and evaluation

In this chapter, a visual presentation of the system is designed. The main topics relate to matching graphical presentation with the tasks and evaluating the design.

Chapter 5 - System implementation

This chapter deals with implementing the design. Development issues such as identifying main classes and their relationship are involved.

Chapter 6 - Usability testing

This chapter describes how to evaluate the system.

Chapter 7 - Conclusion

This chapter summarizes the process from the preliminary design to the final implementation, and offers some thoughts for further improvements.

# Chapter 2

## Background Overview

### 2.1 An introduction to ecosys

Ecosys is developed for the construction and testing of a comprehensive mathematical model of natural and controlled ecosystems. It was written in Fortran running under the UNIX operating system. Its operation is directed by a shell script in which input files for soil, management and climate are read, and output files for C, N, P, ion. water and energy distribution are generated. For example, ecosys allows the user to examine the ecosystem behavior in Edmonton from 1980 to 1990. In this situation, the user needs to input a set of files according to the specific characteristics of Edmonton and what happened during the ten- year time period. Basically the user will need to input the latitude and altitude of Edmonton, the weather condition in Edmonton from 1980 to 1990, the plants that grew there, and events related to fertilization, tillage, irrigation, etc. The user also needs to specify his or her particular requirements to ecosys. For example, the frequency of output may be every five hours instead of every hour, thus significantly reducing modeling time. After inputting all necessary files, the user needs to write a script file to trigger the running of ecosys. Once the system is invoked and executed, a set of output files are generated. The term “evaluation process” is used to describe all the activities from inputting all files until output data is generated. For each evaluation process, the following input files are needed:

1. A site file describing the site characteristics (latitude, altitude, average annual temperature etc) of the specific site in which the user wishes to model the ecosystem.



2. A topographic file describing topographic characteristics (such as slope of the earth face, depth of snow-pack, roughness of surface etc.) of landscape units in the specific place.
3. A soil file describing the characteristics of soil (such as how many layers in the soil, water content, sand and silt content etc.) in the specific place.
4. A control file containing information about user's requirements, such as the time period for the model run, output control etc.
5. A weather file describing the weather condition in the specific place during the evaluation time period.
6. One or several plant management files describing various species of plants in the specific place and harvest related events that occurred during the evaluation time period.
7. A tillage file describing any tillage related events that happened in the specific place during the evaluation time period.
8. A fertilization file describing events related to soil modification that happened in the specific place during the evaluation time period.
9. An irrigation file describing any irrigation related events that happened in the specific place during the evaluation time period.

Tillage, fertilization and irrigation related files are optional and they vary depending on whether any related events occurred in the specified time frame. The contents of site, topographic and soil related files are determined once an evaluation site is chosen. The other six files also depend on the time period of evaluation. This causes a problem, because during different time periods, different events may have occurred in different combinations. For example, in 1980, harvest, tillage, fertilization and irrigation events happened, while in 1981, only tillage and irrigation events happened. This change makes it difficult for ecosys to model the environment.

The designers of ecosys tried to solve these problems by defining the term "scenario". Each scenario corresponds to data input within one year. It includes one

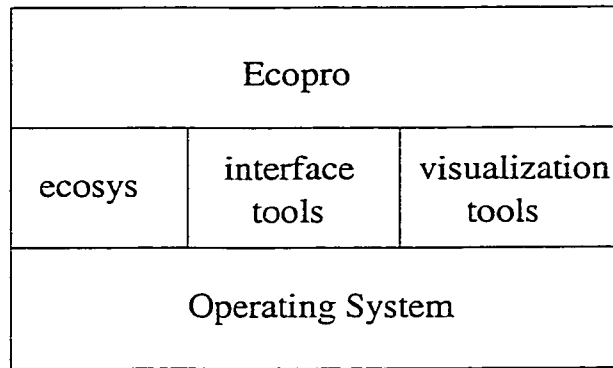


Figure 2.1: Ecopro structure

control file, one weather file, one or several plant management file(s), tillage, fertilization and irrigation files if there are any. Consequently, a typical “evaluation process” contains one site file, one topographic file, one soil file and several scenarios depending on the number of years that the user wishes to model. For example, if the user wants to model the ecosystem in Edmonton from 1980 to 1990, he or she will need to input one site file, one topographic file, one soil file and ten scenarios consisting of ten control files, ten weather files and so on. This makes it very difficult for people to use ecosys.

The goal of the current approach is to provide an interactive system based on ecosys that we call Ecopro. It is aimed at helping the user to input all necessary files, execute ecosys and view the output data in a user-friendly manner. Not only is the user interface design emphasized, but also the overall system purpose, functionality, and structure. All these aspects are treated within the scope of the current user interface design.

## 2.2 Interactive system

According to the definition of [Newman, 1995], “an interactive system supports communication from user to computer and back. The user takes actions such as pressing buttons, and the system reacts accordingly. All of this takes place via the system’s user interface, the part of the system that provides access to the computer’s internal resources. The most crucial property of any interactive system is its support for human activity”.

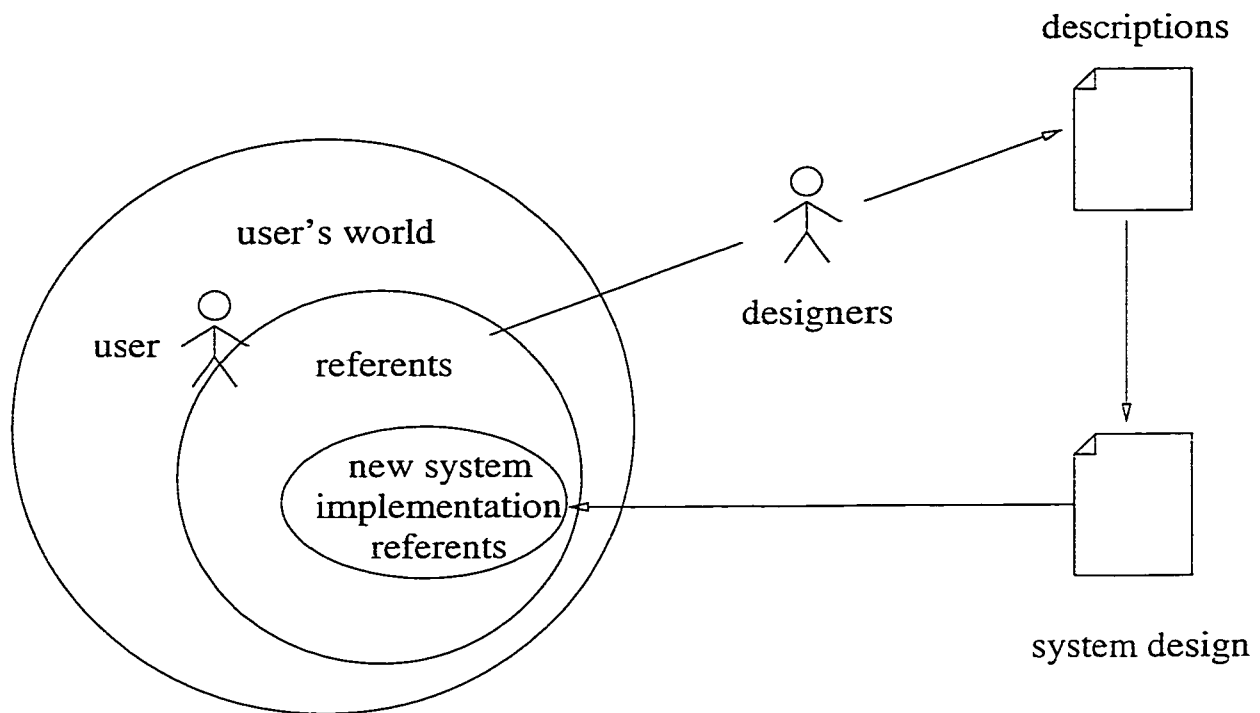


Figure 2.2: A design world

The approach of user interface development is different from general software engineering methodology and requires extra resources and expertise. Using an interactive system is a human activity. Beyond simply doing what is needed, a successful system has to merge smoothly into the user's existing world. There is a basic fundamental difference between the approaches taken by software engineers and human computer interaction(HCI) specialists. HCI specialists are user centered and software engineers are system centered. Software engineering methodologies are useful for specifying and building the functional aspects of a software system. Human computer interfaces emphasize developing a deep understanding of user characteristics and a clear awareness of the tasks a user must perform. Thus an iterative design technique is required for the development of user interfaces. The waterfall approach to software development does not work for user interface design since an user interface cannot be specified without repeated testing with users.

## 2.3 The design world

User interface designers support users who have particular goals within some context. This framework is called a design world as illustrated in Figure 2.2. A user's world is a system that is of interest to a population of users and designers for some period of time. A user's world can be described in different terms and at various levels of abstraction. Referents are entities in the user's world that have meaning to users. They may be users themselves, physical objects in the user's world, or events and tasks. Not only may referents already exist in the user's world, but they may also be added as the result of the activities of computer systems. Designers create descriptions of the world that enable the process of user interface design. A design method or approach is a way of performing user interface design. A design method uses design techniques such as task analysis. Designers use a specific design technique, or combine them to produce a description expressed in a notation. The description is then used for system design, and the feedback from system design adds to the user's world for further improvement. A designer is thus interested in:

- how to form descriptions of the user's world.
- how to match these descriptions to computer system and generate the system design.
- how to evaluate his or her design.

## 2.4 Methodology overview

### 2.4.1 User centered design

Since a common concern within interface design is the involvement of the user, a set of methods have been developed which aim to improve the human involvement in system development. These methods advocate the following approaches which undoubtedly improve system and interface design.

- Users participate design[Newman, 1995]: Users should be actively involved in the process of design and should be assigned to the design team to share in de-

cision making. This is intended to narrow the gap between computer specialists and computer users.

- User centered design[Norman and Draper, 1986]: The system design should be driven by the needs of the users instead of functional processing requirements, limits of hardware, etc. In this way it is ensured that the design will be based on real data, not on imagination, and will be about people and how they perform their tasks.

### **2.4.2 Design based on models**

The discipline of model based user interface design[Forbrig 1997] advocates the explicit denotation of information that is required for user interface design. This formal description can then be used for information analysis, interaction prediction, system specification and generation.

The primary strength of user interface models is in their descriptive power and high level of abstraction. The design group will develop knowledge of the process structure in a much more explicit and precise way. Also, when designs are expressed formally there is a chance that parts of them would be reused in other projects, which is almost impossible otherwise.

There are some inherent weaknesses in the model based approach. The biggest problem is that the user or the client might not understand the formal model, and the designer could get lost in the formal design space, due to overwhelming complex formalism. The expressiveness of the formal language is also a critical issue. On one hand, the formalism should be easy to use, on the other hand it should be powerful enough to allow the full expression of everything needed. Finally different disciplines need different formal models and support tools, which sometimes may not be compatible with each other.

### **2.4.3 Design techniques**

Figure 2.3 lists a set of techniques which may be used through the design process.

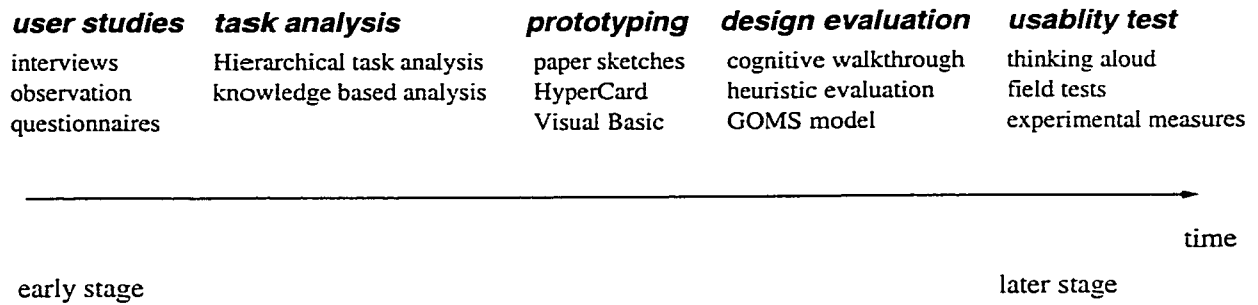


Figure 2.3: Various techniques involved in user interface design

## User study

Human-computer interface is built to suit the needs of people, therefore, it is important to find out who will use the system, their general abilities and other factors which affect their usage of the system. The objective of a user study[Rosson, 1988] is to obtain a thorough knowledge of the skills and experience of all users in order to be able to predict what kinds of tasks they want to perform and how they will react to different interface designs. User studies may also be used in support of evaluation by conducting user tests. Several methods are available for user study including interview, observation and questionnaires[Newman, 1995].

## Task analysis

A task is a unit of human activity, carried out in order to achieve a specific goal. The performance of a task usually involves a sequence of steps, each step contributing in some way to the achievement of the task's goal. Task analysis[Lewis, 1993] is the process of analyzing the way people perform their tasks. It is an important technique that enables users to be involved in the design process and enables designers to understand user's work.

## Prototyping

Prototypes are used to test design ideas and obtain feedback from the user. The result of prototyping might be as simple as a series of paper sketches showing the interface while a user steps through one of the representative tasks, or it may be a detailed design using a system such as Visual Basic. The entire design doesn't need to be implemented at this stage. Initial efforts concentrate on parts of the interface needed

for the representative tasks so that feedback from users can be quickly generated and used for re-design.

### **Design evaluation**

Evaluation after prototyping [Shneiderman, 1998] has the advantage that problems can be found before considerable effort and resources have been expended on implementation. A number of methods have been proposed for this purpose such as cognitive walk-through[Carroll, 1995], heuristic evaluation and model based evaluation. The basic intent is to identify any areas which are likely to cause difficulties because they violate known cognitive principles, or ignore accepted empirical results.

### **Usability testing**

The distinction between prototype evaluation and usability testing is the existence of an actual implementation of the system in the latter case. There is something concrete to test, thus usability tests tend to be experimental involving real users while prototype evaluation tends to be analytic. There are various kinds of techniques which can be explored for usability testing such as thinking aloud, field tests, experimental evaluation etc.[Shneiderman, 1998]

## **2.5 Design process**

As we mentioned above, an experimental approach is necessary in interface design because there is not a sufficiently strong theory from which a theoretically based design could be constructed. A designer may choose to combine different approaches depending on the specific needs in different cases. In Ecopro, an attempt is made to combine user centered design with a task centered approach by using different models. In the subsequent sections, the models that are used in Ecopro will first be described and then a brief explanation will be given of the design process using these models.

### **2.5.1 Models used in Ecopro**

The models in combination with their use form the basis of design methods and techniques. Figure 2.4 illustrates the models that are used in Ecopro and the design

steps.

## User model

A user model is used to understand and model an activity as it is understood by the user. If the designer can get the model right then he or she can use it to create a design that will be intuitive to the user. User models come in several categories depending on the interest of the authors. The following types are currently found in the human computer interface literature:

- Theoretical cognitive models as constructed by psychologists. The purpose is to understand human mental processes.
- Model of user knowledge as inspired by computer based training(CBT) interests and adaptive interfaces. The model is constructed of domain knowledge to assess how users learn by traversing the knowledge network.
- Model of user characteristics attempting to classify users in terms of skill and ability; It is also called user profile.
- User task model that reflects the user's concept of how a task is constructed in terms of its functions and operational sequence. It is an attempt to discover how much users know about the system in terms of its operation and what are their expectations about how it will work.
- User view is the user's model of the system structure which may be expressed in terms of visual presentation of system components. It is also called user system image.

In Ecopro, a user profile is acquired during the user requirement analysis stage. The user task model and user view are generated in tasks analysis and dialogue design stages respectively. Theoretical models and models of user knowledge are not used because they have less direct relevance to the present design.



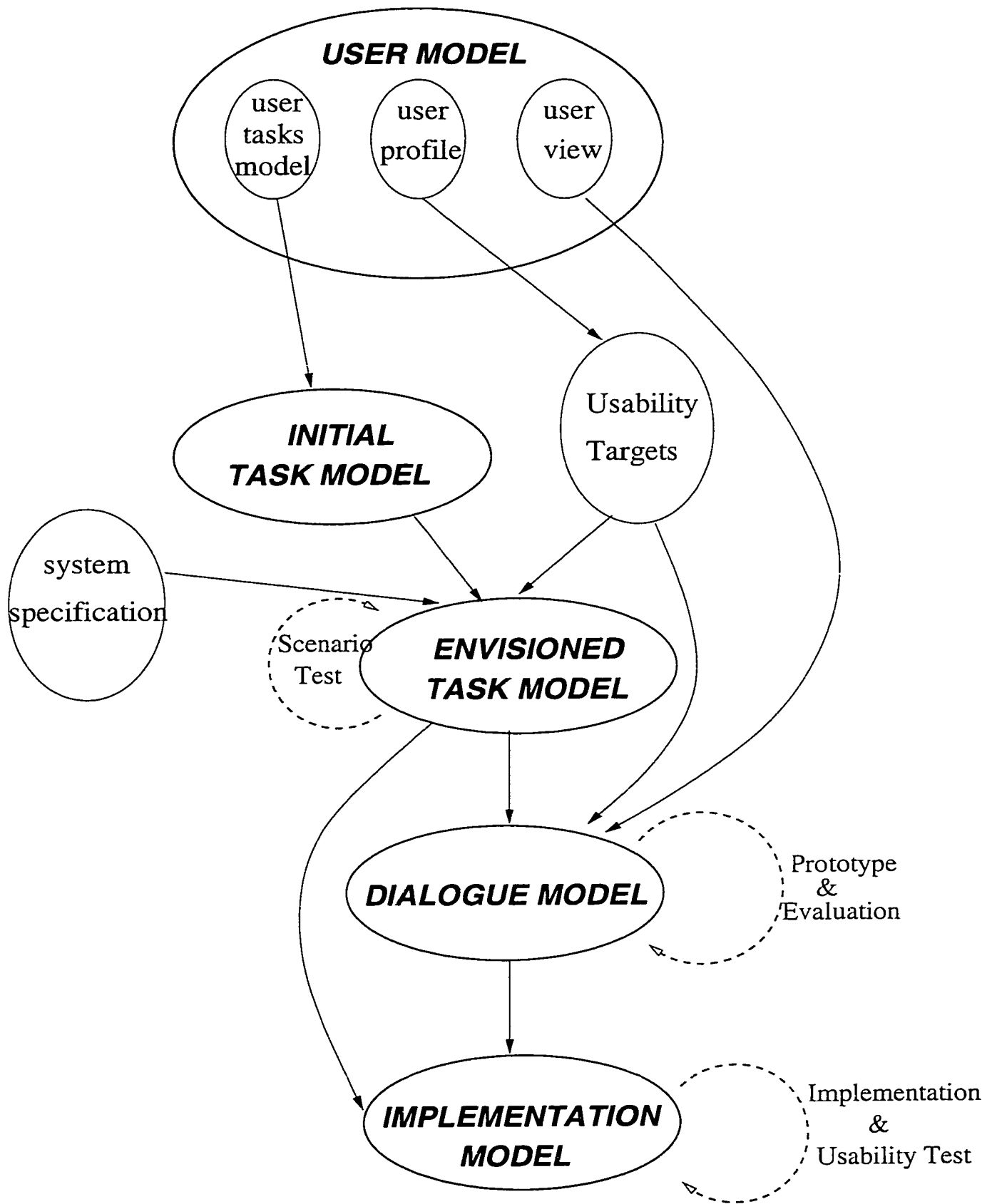


Figure 2.4: Ecopro models and design process

## **Task model**

A task model describes the activities in which users engage to achieve their goals, the details of those goals and the strategies adopted to achieve them. The task model is described in terms of the actions that users perform, the sequencing of the actions, and the objects involved in those actions.

Task modeling can essentially be done on two different levels, leading to different models. First, it represents the tasks that users perform with existing systems. Secondly, it represents the tasks that are envisioned as being performed with one or more proposed systems. This envisioned task model documents the task structure and the division of labor between the human user and system. Based on this second task model, the dialogue structure of the interface is developed and captured in a dialogue model.

## **Dialogue model**

A dialogue model represents important information about the conversation between the user and the computer system. Ultimately, this information supports the development of a concrete user interface. Using a continuously developed, explicit dialogue model can ensure the consistency between different levels of interface modeling, and provide a common ground for communication between user and designer [Dix, 1993]. Dialogue models can be described by diagrammatic as well as textual notations.

## **Implementation model**

An implementation model is a concrete model that drives the implementation. It is a level of detail which is deemed desirable for generating implementation codes.

## **Object-oriented model**

An object model is a description which is composed of objects and links between them. Many of the advantages claimed for object-oriented design have particular attraction to the developers of interactive application, in that a key aspect of designing such systems is the creation of tasks and interfaces that are accessible to users but also extensible as user's needs evolve.

In Ecopro, the object-oriented concept is employed in the envisioned task model, dialogue model and implementation model. The initial task model is not object-oriented because it seems natural to describe the initial model by tasks and their sequence rather than by objects directly. Consequently the initial task model focuses on the tasks, their structure and sequencing, each object is linked to those tasks which operate on itself; while the envisioned task model emphasizes the objects, their properties and relations, each task is linked to those objects which are involved in the task execution and the sequencing of tasks is of less importance.

### **2.5.2 Design process**

Design methods generally describe a series of phases with each phase involving one or more activities using models, e.g. eliciting the information in the model, manipulating the model or using the model to create further design models.

#### **User analysis**

The first stage in the design process involves collecting and analyzing data about the users, their tasks, the way they may interact with the system, and the system image in their mind.

An analysis of user characteristics is important. The general ability, computing and task experience of the user contributes towards measures of level of support. These measures can then be used to generate the task model and plan suitable types of interfaces at later stages.

Knowledge about how users perform tasks enables designers to contemplate the aspects of work that could or could not be supported, how they could be supported and what changes to the work will come about as a result of a design. This is achieved by generating the user task model and view model. A detailed explanation of the two models will be offered in the next two chapters.

#### **Task analysis**

The more knowledge about the user tasks available during design time, the more it can be exploited when defining the system's properties and features, and the higher the degree of user acceptance and satisfaction obtained when using the system. After

understanding more about the user, modeling and analyzing of the user's tasks begins. The output of this process is then used in the subsequent system design and evaluation phases. The task analysis concentrates on the tasks that the user wants to perform and the level of support that can be provided. There are six steps involved in the task analysis.

1. Interview the user and generate the user task model.
2. Apply domain knowledge, analyze and refine all tasks, generate the task model.
3. Consider level of support and system constraints, and apply to the task model.
4. Allocate the tasks from the user and system's perspective.
5. Identify objects and actions in the task model, and apply the result of 3 and 4 to generate the envisioned task model.
6. Use scenarios to evaluate the envisioned task model.

The process starts with interviewing users, collecting data on the tasks they work on and their means of performing tasks. Then data is analyzed to build a model or abstraction of the activities. This provides a blueprint for the initial version of the user task model.

Since the user task model provides the designer with only the user's thoughts, it is not complete and specific enough for further design. It needs to be refined with domain expertise to generate the task model.

Usability is a fundamental concern for interface design. Once it is known who will use the system and what they're going to do, targets need to be set for the levels of support that the system will provide to the user.

Another consideration is system constraints that may affect the choice of solution. A successful design for an interactive system requires the recognition that any system may involve many layers of technology including application software, operating system, networked resources and hardware. Often it is these systems that constrain the design.

The construction of the envisioned task model allows the designer to identify the new tasks to be supported in the system under design. To determine the focus of the

level	task
lexical	description of shape of icons, objects and keys in the UI
syntactic	description of order and structure of methods/services used to satisfy a main goal
semantic	description of dialogue in terms of its effect on the functional application core

Table 2.1: Level of dialogue modeling

system and how it fits within the current work organization, a clear understanding is needed of the distinction between the tasks performed by the computer and the tasks that are performed manually. This process is called task allocation.

After applying system constraints and usability targets to the task model, the model is then transferred from task based to object based. Given the task model, for each task an initial set of objects and their corresponding actions are identified. The most frequently occurring objects form the basis of the envisioned task model, and the relationships between the objects are defined by their roles in the tasks. As the analysis proceeds, the model is expanded to map all tasks.

To enable early discovery of errors, scenarios are produced and utilized as sample tasks to make sure all functionality required in the system is covered and various features of the system work seamlessly together to accomplish real tasks.

## Interface design

All tasks have to be performed through a friendly user interface, hence it is very important to design the visual presentation of the interface to effectively support the user in performing these tasks.

In Ecopro, an explicit dialogue model is used to describe the user computer conversation. There are three levels within the model[Elwert, 1995]. The lexical level describes low-level tokens. For the input language, these tokens include keystrokes,

mouse clicks, or mouse motion. For the output language, these tokens include the components that can appear on an user's display, their layout characteristics, and the visual dependencies among them. Normally the lexical level incorporates principles of graphic design[Shneiderman, 1998] in order to give a comprehensive support to the dialogue designer. The syntactic level defines the structure of the dialogue between the user and the application. It controls the execution of the logic of the dialogue by describing the sequencing between different presentation units called dialogue views. Semantics refers to the meaning or intentions of the end user. The semantic level of the dialogue model provides an interface between the dialogue and the task model. It deals with the functional description of the dialogue within a dialogue view including the realization of state changes of user interface objects.

Five steps are followed to generate the dialog model.

1. Based on the envisioned task model, define the syntactic level of the dialogue model.
2. Define the semantic level of the dialogue model.
3. Verify the model for completeness and reachability.
4. Select an appropriate interaction style of the interface.
5. Define the lexical level of the dialogue model

During the syntactic level of dialogue modeling, a view is attached to each object. It can be considered as an abstract representation of an object in the user interface. Afterwards, the structure of views and the sequence of interactions within each view is identified.

Then interactions within each view are mapped to the functions residing in the task model. In this way it is assured that the task fits into the view.

Once the views have been defined in terms of the tasks they support, the model can then be checked for completeness and reachability.

The design can then be further defined in terms of the mechanism that the user employs for interaction with the computer system. An appropriate interaction style depends on the specific tasks and the usability issues.

Finally details of visual aspects of the interface are defined which transform the dialogue model from an abstract level to a concrete level and smoothes out the development process.

### **Prototype and design evaluation**

In Ecopro, low-fi techniques - paper sketches, are used for prototyping. The sketches are then shown to the user with an explanation on structure, navigation path and the details of visual design.

Cognitive walk-throughs are used for design evaluation. This is a formalized way of depicting people's thoughts and actions when they use an interface for the first time. Sample tasks are selected to specify what a user would see and how they react step-by-step in performing the task. The feedback is used for redesign.

### **Implementation**

At this moment there are objects, their corresponding views, and the set of interactions by which users and system communicate with each other. Next, the Model-View-Controller (MVC) is used to generate the implementation model. MVC was introduced by smalltalk developers [Goldberg, 1983] and has been used as a framework for interactive system implementation for quite some time.

The standard interaction cycle in the MVC begins with the user taking some input action and the active controller notifying the model to change itself accordingly. The model carries out the prescribed operations, possibly changing its state, and broadcasts to its dependents (views) that it has changed. Views can then update their display if necessary. This message-sending is shown diagrammatically in Figure 2.5.

Since there already exist objects, views, and interactions in the task and dialogue models, little effort is required to integrate them and generate the implementation model. The system is described using the set of notations in Unified Modeling Language (UML). Thus the final implementation model is a set of class diagrams and interaction diagrams.

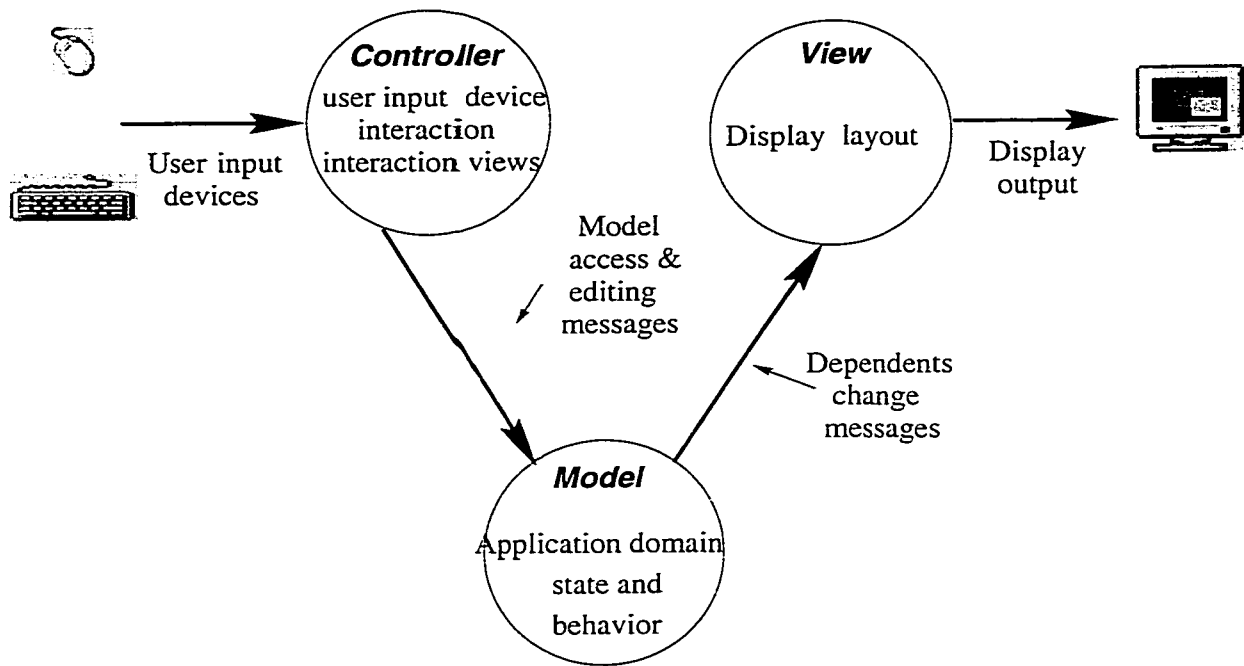


Figure 2.5: Model-View-Controller state and message sending

### Usability testing

The point of testing is to anticipate what will happen when real users start using the system. Thinking aloud [Nielsen, 1993] and user survey are used in Ecopro for usability testing.

To use thinking aloud, the user is asked to perform some tasks, and talk to the designer at the same time. They are asked to tell what they are thinking; what they are trying to do, questions that arise as they work, etc. The designer makes a recording of their comments. Process data is collected after applying think aloud to the user. Process data are observations of what the test users are doing and thinking as they work through the tasks. These observations tell us what is happening step by step, and suggests why it is happening.

A user survey is used to collect bottom-line data which give us a summary of what happened. For examples, how long did users take, were they successful, how many errors did they make. The survey form may cover the topics such as overall reaction to the system, satisfaction on system capabilities, terminology and information, screen representation etc.

After all the tests, the data is analyzed and the results are used for further im-



provement of the initial design.

# Chapter 3

## User Requirements and Task Analysis

In this chapter, we start with the analysis of user requirements and their profiles. Then the task model is generated and finally a scenario test is used to check the model.

### 3.1 Requirements definition

As the first step of our design, we define the main problems that we are going to solve, identify users, understand their usability demands, and consider alternative functional forms that may enable these requirements to be met more easily.

The initial problem can be viewed as “design an interactive system on top of existing software applications and provide the user with a high level of interaction support”. The main users of this system are people working in the ecology field. In order to have a clear understanding of the problem, the problem statement needs to be expanded to include a level of detail which defines the major interactive functions and the initial usability targets that the system is to provide. The result is shown in Table 3.1.

### 3.2 User profile analysis

The Ecopro user profile analysis focuses on four aspects:

- psychological characteristics includes users’ motivational level, their attitudes, and their cognitive style.

---

## 1 General

- 1.1 The system should provide a mechanism to help the user enter all necessary information to run an existing model
- 1.2 The system should provide a mechanism to help the user update existing information which was entered before and rerun the model
- 1.3 The system should provide a mechanism to help the user visualize the output data

## 2 Support

- 2.1 The system should hide all details of "ecosys" and the computer system
  - 2.2 The system should provide convenience for user's work
  - 2.3 Manual should provide the user with initial training and information on its use
- 

Table 3.1: Initial user requirement of Ecopro

- knowledge and experience includes users' education level, typing skill, experience of similar applications, and computer literacy.
- task characteristics includes users' job category, task experiences, frequency of usage, any training on working field, and task structure.
- physical characteristics includes users' gender, handedness, and whether they are color-blind.

Users of Ecopro have high motivation and positive attitudes. They have solid background on task knowledge, but moderate level of experience with computer systems. They do not have experience with similar applications in the past. Frequency of usage won't be high for Ecopro, which means users won't use the system everyday. On the other hand, due to the heavy input load, they may have to spend a long time to get the output result every time they use the system.

Based on the considerations above, decisions are made on usability and visual design which are illustrated in Table 3.2. The results can be used in the later design stages such as task analysis, usability performance set up, dialog modeling and usability testing.

We notice that sometimes the usability targets may conflict with each other, such as the trade off between ease of use and ease of learning . As a general rule, the more flexible a system is, the easier it is to use, but the harder to learn and control. Since

---

<i>User Definitions</i>	
psychological characteristics	high motivation positive attitude
knowledge and experience	moderate level of computer literacy no experience with similar applications
task characteristics	sound background knowledge low frequency of usage highly structured tasks

---

<i>Design Choices</i>	
usability	easy to learn easy to use
visual Design	familiarity: use existing framework for visual design consistency
others	system controlled interaction instead of user controlled interaction (less flexibility) user-participative design

---

Table 3.2: Applying user profile on design

the frequency of usage in Ecopro is low, users will not be able to learn and remember unless the system is designed for ease of learning. This factor will be one determinant that guides us in the ease of use versus ease of learning trade off. Another point is that usage of Ecopro is discretionary. Hence first impressions are very important to create motivation. Again ease of learning has a higher priority which implies that the system should provide simple and structured interactions with more instructions and feedback.

## **3.3 Task model generation**

### **3.3.1 User task model development**

After the initial user requirements are defined, more details are needed on overall functionality that the system is required to provide. By interviewing users, an understanding of the users and the tasks they perform can be obtained.

A clear structure is necessary for the interview to make sure all the relevant topics that are of interest are covered. Table 3.3 lists a few questions which are used in the interview.

The outcome of interviews with users is a set of written transcripts describing the tasks in users' minds and the means to perform those tasks. These transcripts are used to generate the user task model and user view. The user view will be elaborated upon in Chapter 4.

By arranging tasks in a hierarchical structure based on the goals they achieve and the steps they involve, the user task model is generated. Various subtasks must be accomplished in order to perform the main task and they are further subdivided into dependent tasks. Hence the user's task model is a hierarchy of tasks and subtasks and plans describing the order in which they are performed.

A graphical notation is used to represent the tasks. Figure 3.1 is an example of the first level of task analysis in Ecopro, and figure 3.2 is a further task breakdown of the tasks shown in figure 3.1.

Issue	Specific questions
Purpose	what is the purpose of this system from your point of view? why? what are the main differences between the new system and the old one, why?
Enumerating Activities	from the functionality perspective, what do you expect the system will provide for you, why? can you show me some specific examples?
Working Methods	in the given example, is there any other way that you may try to get the result? why you choose yours rather than this one? what would you like to do if this develops a fault? any specific tools you want to use in the system?
Support Issues	which level of help information do you think is appropriate? why? which is more important for you, ease of use or ease of learning? why?

Table 3.3: Sample questions in user interview

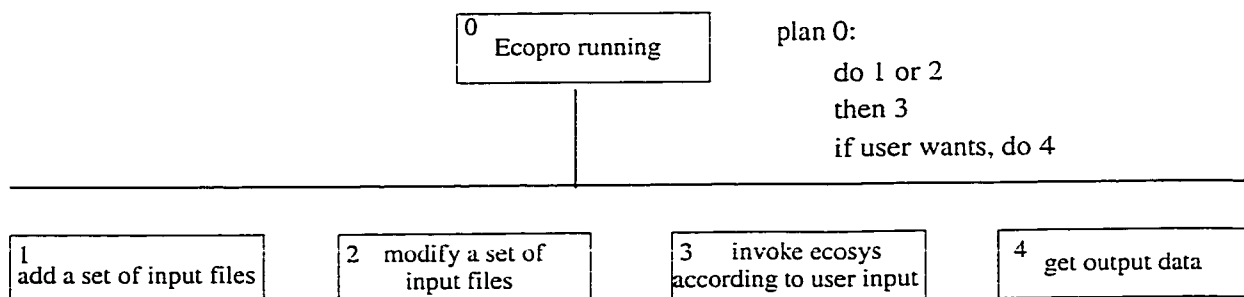


Figure 3.1: First level of task analysis in Ecopro

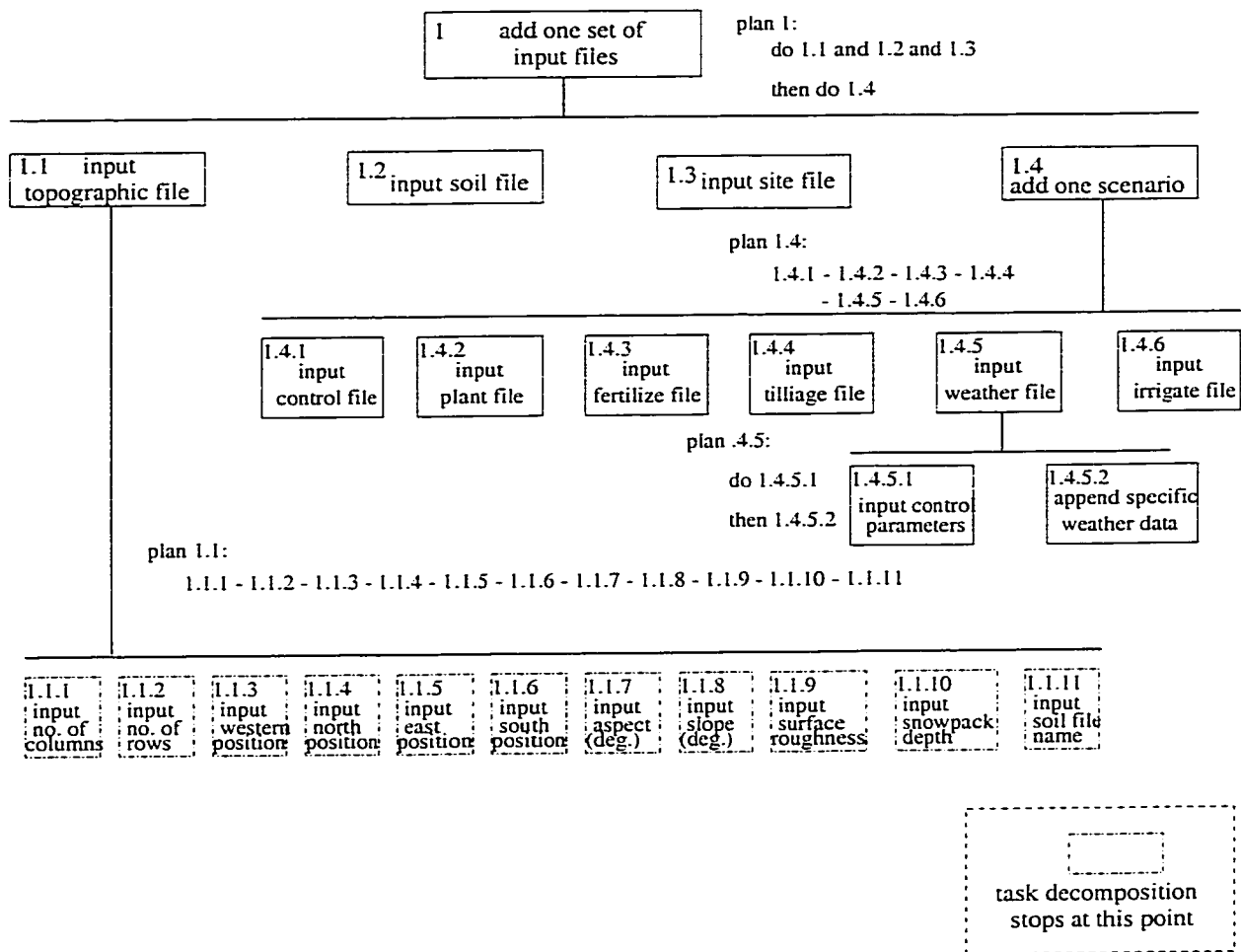


Figure 3.2: More levels of task analysis in Ecopro

### 3.3.2 Initial task model generation

The model showed in figure 3.1 and figure 3.2 provides the current designer with a general idea about user requirements. More detailed information is needed for further design. For example, is there any parallelism or iteration of the tasks depicted in the figures; are all the tasks necessary; are they completely covered by all the scenarios of usage; is the order of tasks correct. Hence there is an need to analyze the model together with domain expertise and refine it.

After refinement, the model is presented in a more formal descriptive way which contains the following information:

- the structuring of tasks into subtasks by using graphical notations.
- temporal relations between tasks by using plans.
- possible user decisions.

And each task in the model contains the following items:

- name of the task.
- pre- and post conditions for the task.
- objects used within a task context.

Figure 3.3 depicts the first level of task analysis of Ecopro after refinement. There are several changes compared to figure 3.1. Each evaluation process is assigned a name because there could be many such processes in the system. As the user may want to run the process with different sets of input files for multiple times, repetitions need to be allowed in steps 1 to 4. In order to perform step 2, there must be files which need to be pre-loaded. Thus, if step 1 does not occur, there should be no step 2. All these considerations are added to the model in figure 3.1 to generate the model in figure 3.3.

Similar to the refinement of the model depicted in figure 3.1, the model in figure 3.2 is refined to generate a detailed model for the task of adding input files (figure 3.4 -3.7 ). Other main tasks in Ecopro are illustrated in figure 3.8-3.10.



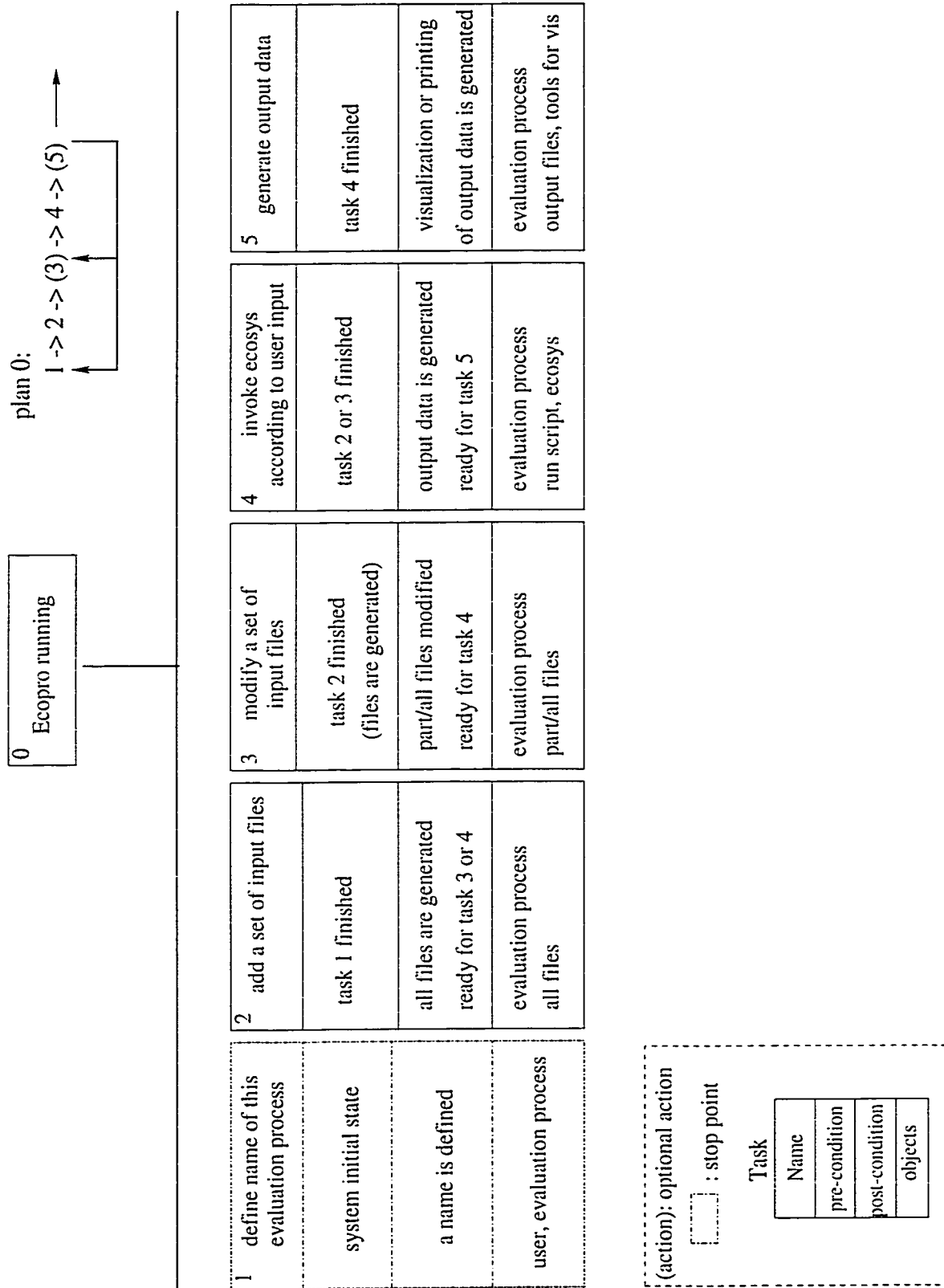


Figure 3.3: First level of task analysis after refinement

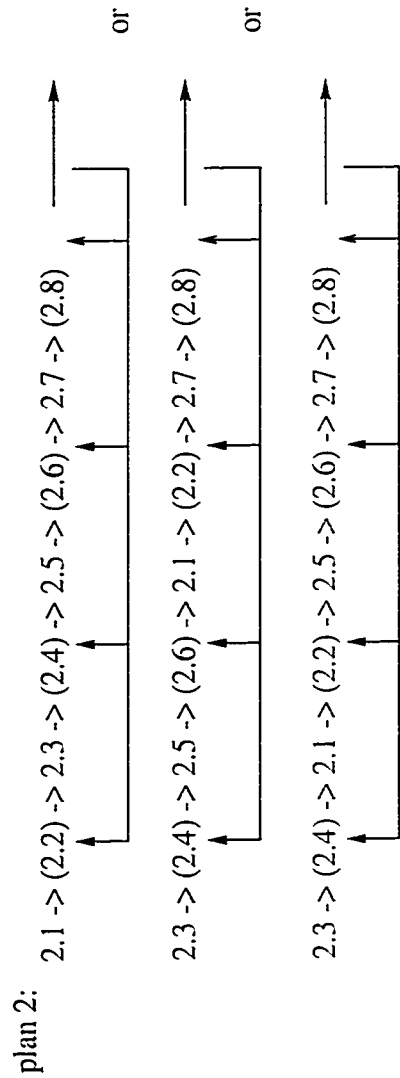
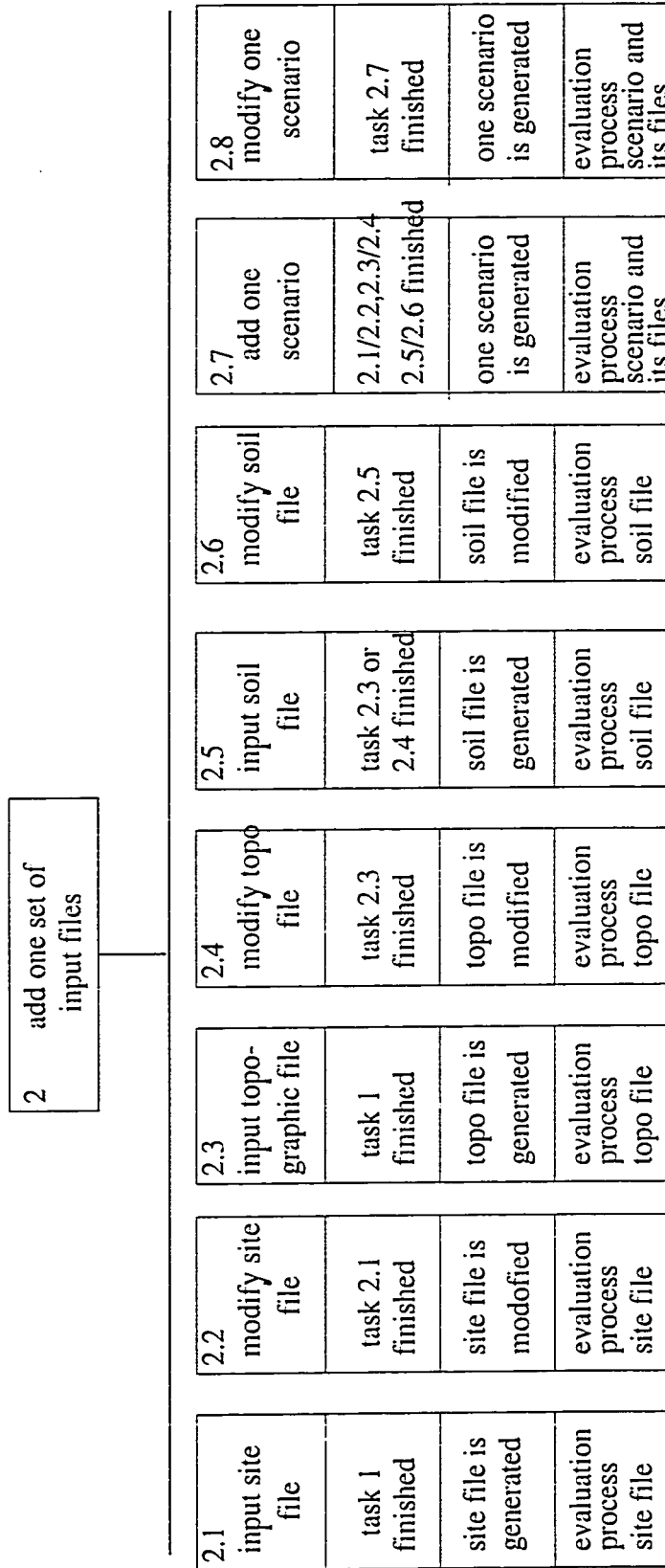


Figure 3.4: Task model of adding input files

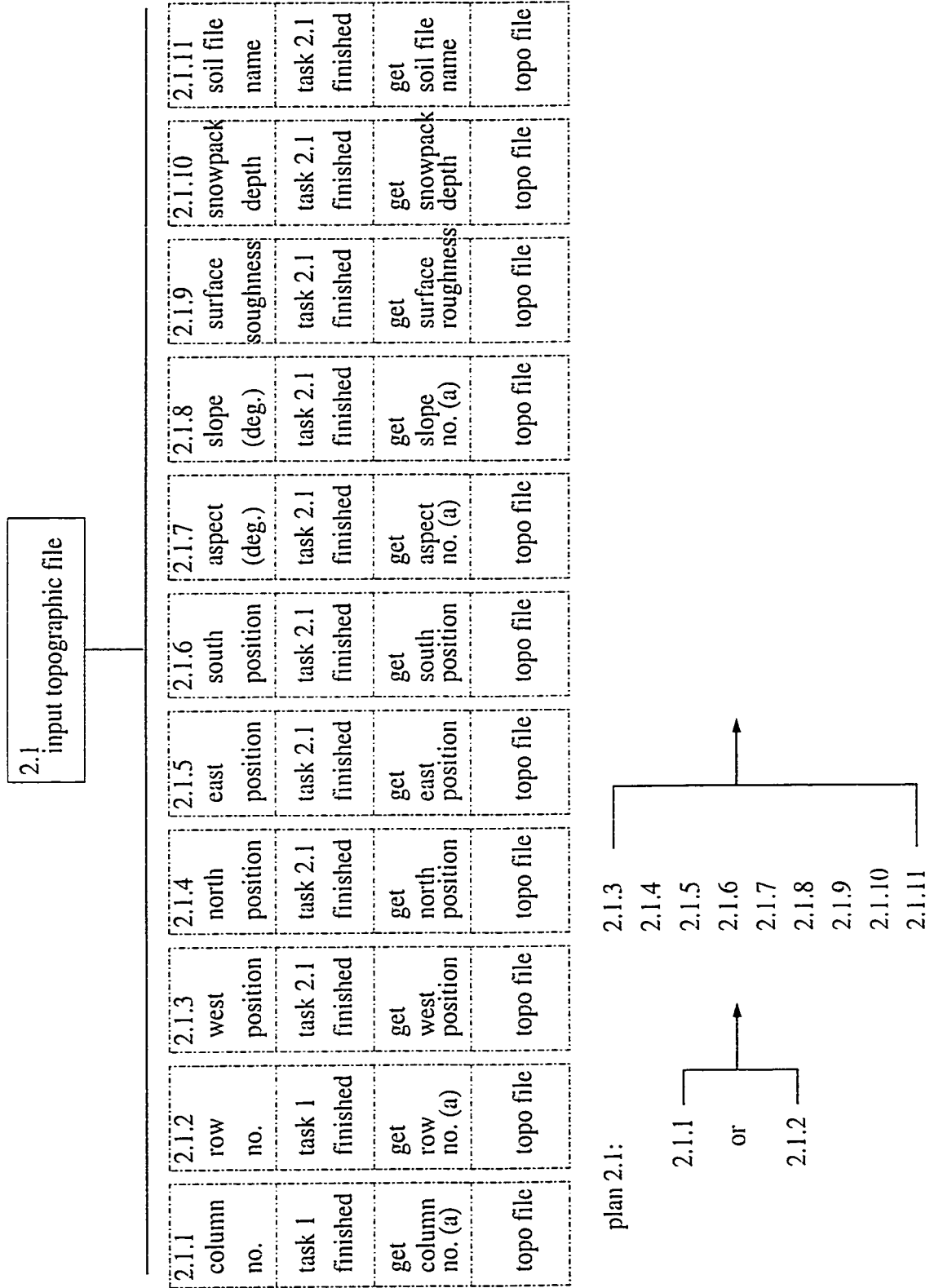


Figure 3.5: Task model of inputting topographic file

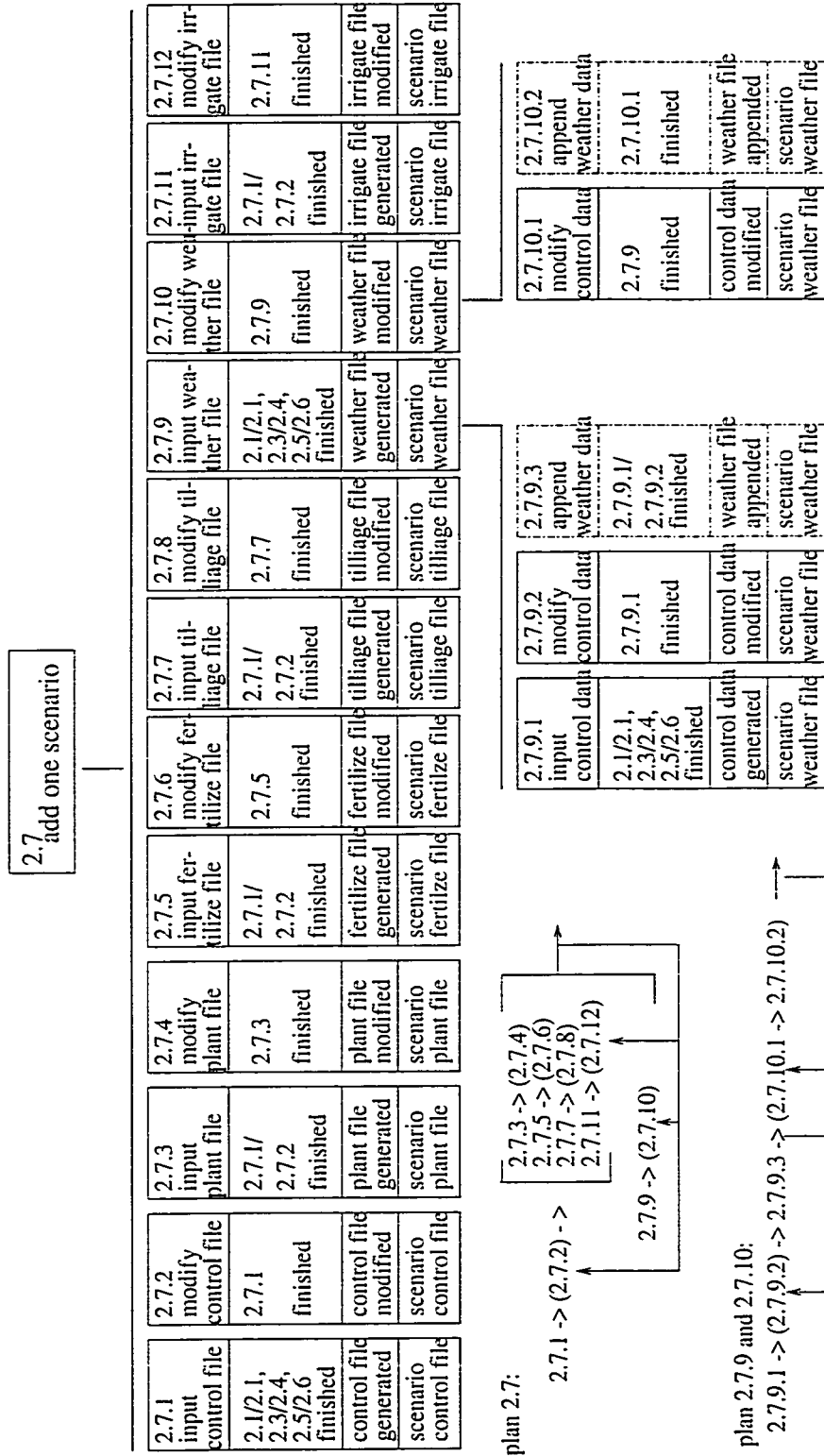


Figure 3.6: Task model of inputting one scenario

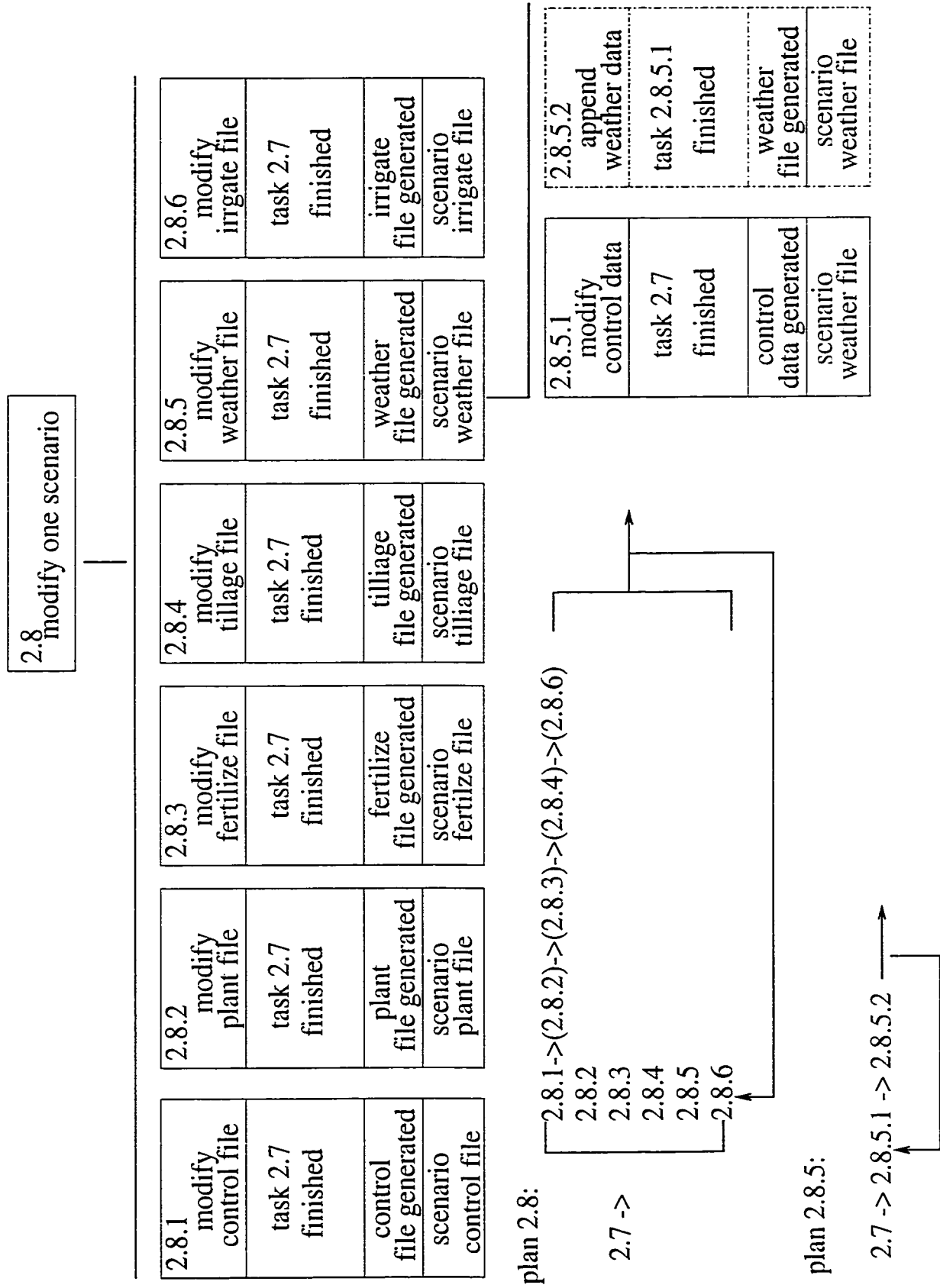


Figure 3.7: Task model of modifying one scenario

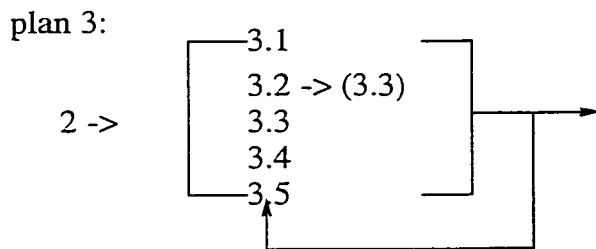
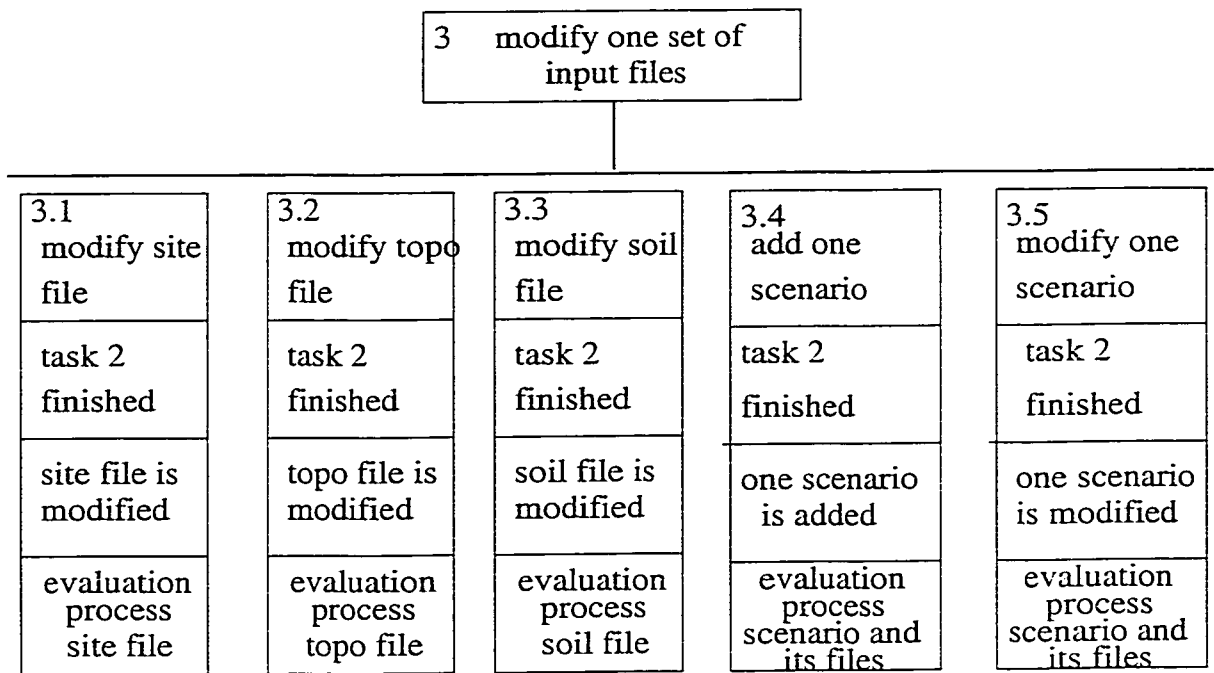


Figure 3.8: Task model of modifying input files

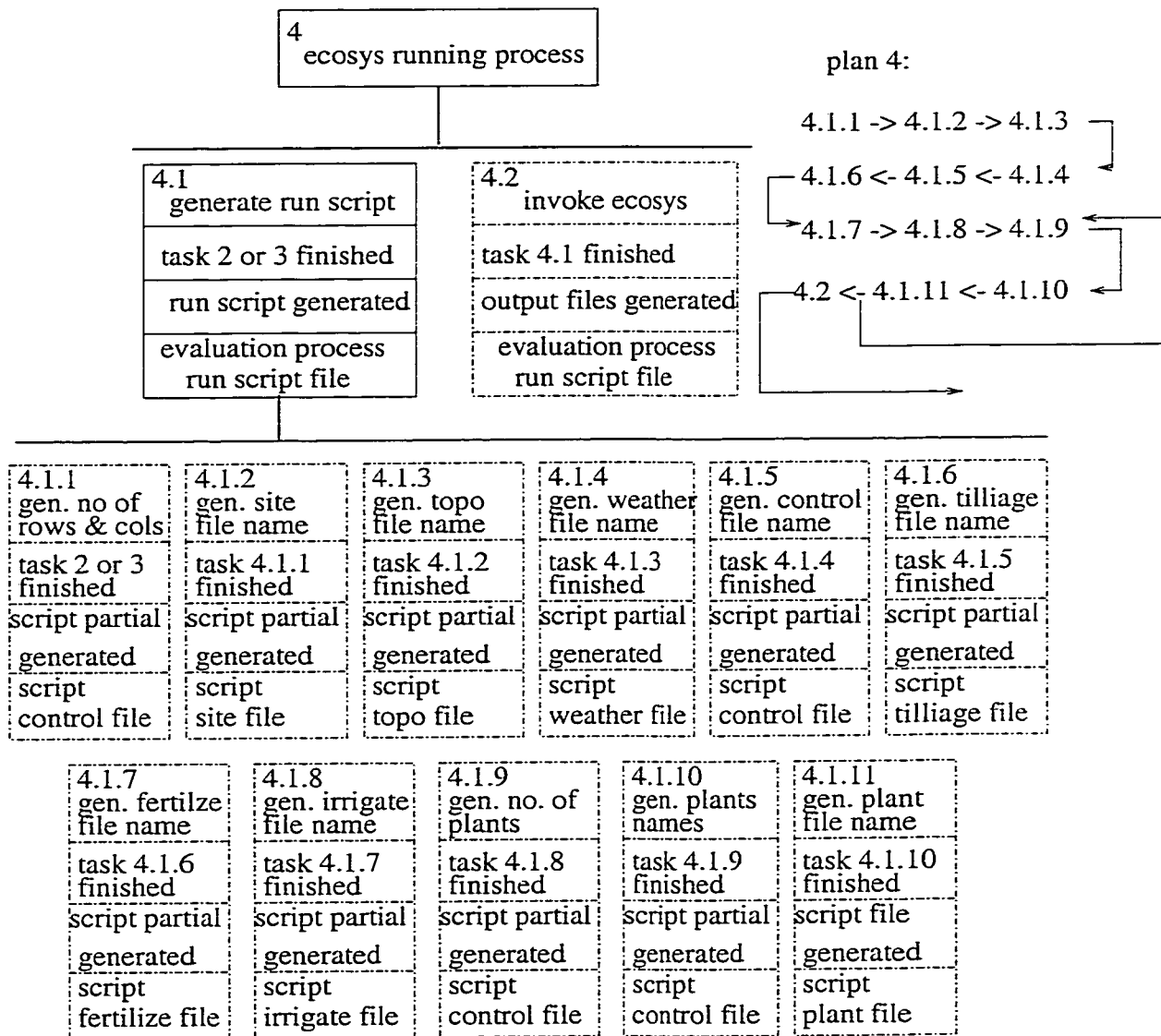
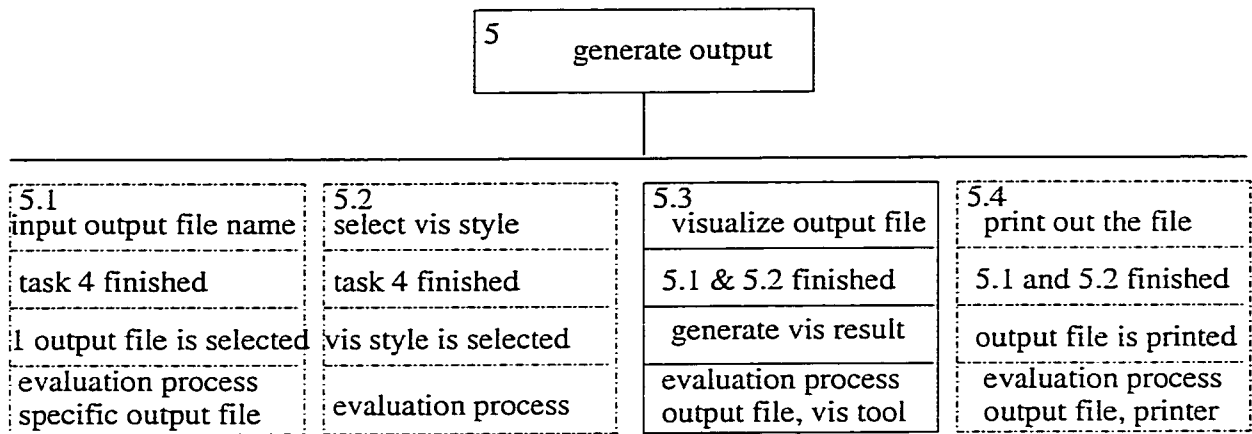


Figure 3.9: Task model of invoking ecosys



plan 5:

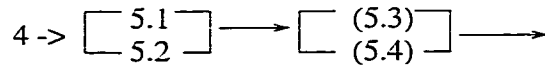


Figure 3.10: Task model of visualizing output data

An important issue is identifying when the tasks are basic enough and their sub-division can be stopped. The stop point criteria used for this purpose is when the task involves user response such as input operations or internal decision making. In the first case, further breakdown would not be productive, because explaining how such actions are performed is unlikely to be either accurate or useful. In the second case, the task sub-division would be done only if any decision making is related to external actions, such as reading help documents, but not when the activity is purely cognitive.

### 3.3.3 Envisioned task model generation

The construction of the envisioned task model allows the identification of the new tasks to be supported in the system under development. For this purpose, the current designer considers system constraints and usability performance, and applies them to the initial task model. Then the designer determines how it fits within the current work organization.



## **System constraints**

A successful interface design is based on the recognition that interactive systems involve many strands of technology. They include application software, operating system, networked resources and hardware. The designer may need to understand requirements for any or all of these other strands. And the new system will need to be integrated with these other systems.

Currently, Ecopro is running on UNIX, the designer choose to translate it to the personal computer environment so it can be easily accessed by most users. Java is employed as the interface development tool, and visualization toolkit (VTK) is used as the tool for output data visualization. One reason of choosing these tools is that they are platform independent. Another reason is that they are powerful high level tools, which means it is faster and easier for implementation.

## **Usability targets**

In an interactive systems, usability requirements are of prime importance, because they define the means to design a system that supports the user's activity properly. To determine the level of support provided to the user, basic usability goals are set for Ecopro. Some of them result from the user study and others are general usability considerations [Shneiderman, 1998]. Priority is given to each usability issue in the following order:

1. ease of learning
2. recovery from errors
3. speed and performance
4. ease of use

Ease of learning can be defined in terms of the amount of time that users need for making their decision while interacting with the system. If the appearance of the interface and its operations are familiar to users, the amount of decision time will be significantly reduced. Thus for the visual design of the interface, the existing framework and style is adopted so as not to surprise the user. An easily accessed

---

Functional part	Non-functional part
easily accessible help documents	consistency
feed back of operations	familiarity: use existing framework
undo and redo functions	disable interface parts which are not accessible
multiple threads	warning of mistakes
default value for data entry	keep user memory load to minimum

---

Table 3.4: Usability considerations in Ecopro

help function and documentation for the system is also provided. The change in the system's status as a result of user interaction should be visible as a way of feedback to the user. Consistency is another important issue while considering support for ease of use. The current design tries to be consistent on input expressions, output response, interaction style, and visual appearance of the interface.

Predictability helps reduce user errors. Users should be able to anticipate when and which operations can be performed. This user anticipation is used to disable parts of the interface when operations associated with those parts cannot be performed in the given scenario. In case of user error, there should be a warning to tell the user about the kind of error he or she just made. Undo and redo functions are provided.

To determine meaningful speed of operation requirements, the current designer focused on those activities that can be speeded up significantly by the system. To speed up Ecopro as compared to ecosys, multi threading is implemented so that several tasks can be executed concurrently.

Ease of use is another usability factor. If too much power of control is given to the user, such as more flexibility to customize the interface, and reorganize activities, the system will become more complicated and difficult to use. As ease of learning has higher priority than ease of use, only the basic support is provided which is enough for task completion, such as giving a default value for input data.

Table 3.4 shows the result of the current usability performance analysis.

---

	file storage
System:	source of data as default input values file and system administration (save, undo...)
User:	make decision on sequence of tasks input data

---

Table 3.5: General rules for task allocation in Ecopro

### Envisioned task model generation

Based on the above considerations related to system constraints and usability targets, an envisioned task model is defined. The current designer starts with task allocation which shows the division of labor between the user and the system, so that interactive functions of the system are identified and details of how tasks fits into the system are understood.

Within each task, actions are allocated to either the computer or the user, or both the user and the computer. Normally heuristic or associative tasks as well as tasks requiring initiative or judgment are allocated to the user while repetitive tasks, calculations or high volume data handling are allocated to the system. Table 3.5 shows the result of task allocation in Ecopro.

Afterwards the task flow in the original task model is used to generate an object-oriented envisioned task model. There are three steps associated with the model development.

1. Apply the results of system constraints analysis and usability targets analysis. Functions such as undo, redo, help access, etc. are added.
2. Identify important objects. For each object, identify its attributes (status etc.) and apply actions to it. The object state diagram is used to describe the object-action relationship. Here the purpose is not to generate any machine representations of the objects, but to describe their participation in human and computer

tasks. Thus, few details related to each object's attributes are brought in here.

3. Identify relationship among objects by applying tasks to the objects. The object relationship diagram is used to describe the object relationship. The most frequently used objects form the basis of the initial diagram, and the initial relationships are indicated by the most frequently performed tasks. As the analysis proceeds, the model is expanded to map all tasks. In the initial stages of modeling, it is often helpful to define relationships loosely. That is, to simply indicate that objects are related without specifying further details. In the object relationship diagrams, this is indicated by a simple line connecting two objects. When relationships are clear, they can be further defined as 'contains', 'decides', etc.

In figure 3.11, the object state diagram for the user is shown to describe the relationship between the user object and its corresponding actions. The user may add one evaluation process at the initial state. He or she may also set up preferences or access help. However, if the user wants to modify or delete one evaluate process, the system will tell him or her that there is no evaluation process to be modified or deleted at this time. After one evaluation process is added, the user may add another, modify or delete it. Similarly, figure 3.12 describe the relationship between the evaluation process and its corresponding actions.

Object relationships in Ecopro are described in figure 3.13. Each user may have multiple evaluation processes to run. Each evaluation process contains one site file, one topographic file, one soil file, multiple scenarios etc. And each scenario contains one control file, one fertilize file, one tillage file, one irrigate file, one weather file and multiple plant management files. Since site and topographic files contain information which are required for soil data entry, a relationship "decides" is defined which means if the soil information in site and topographic file is modified, the soil file has to be modified accordingly.

### **3.3.4 Scenarios test**

The outcome of the task analysis step is used as a starting point for interface design, as it imposes requirements on the dialogue model. All decisions made when

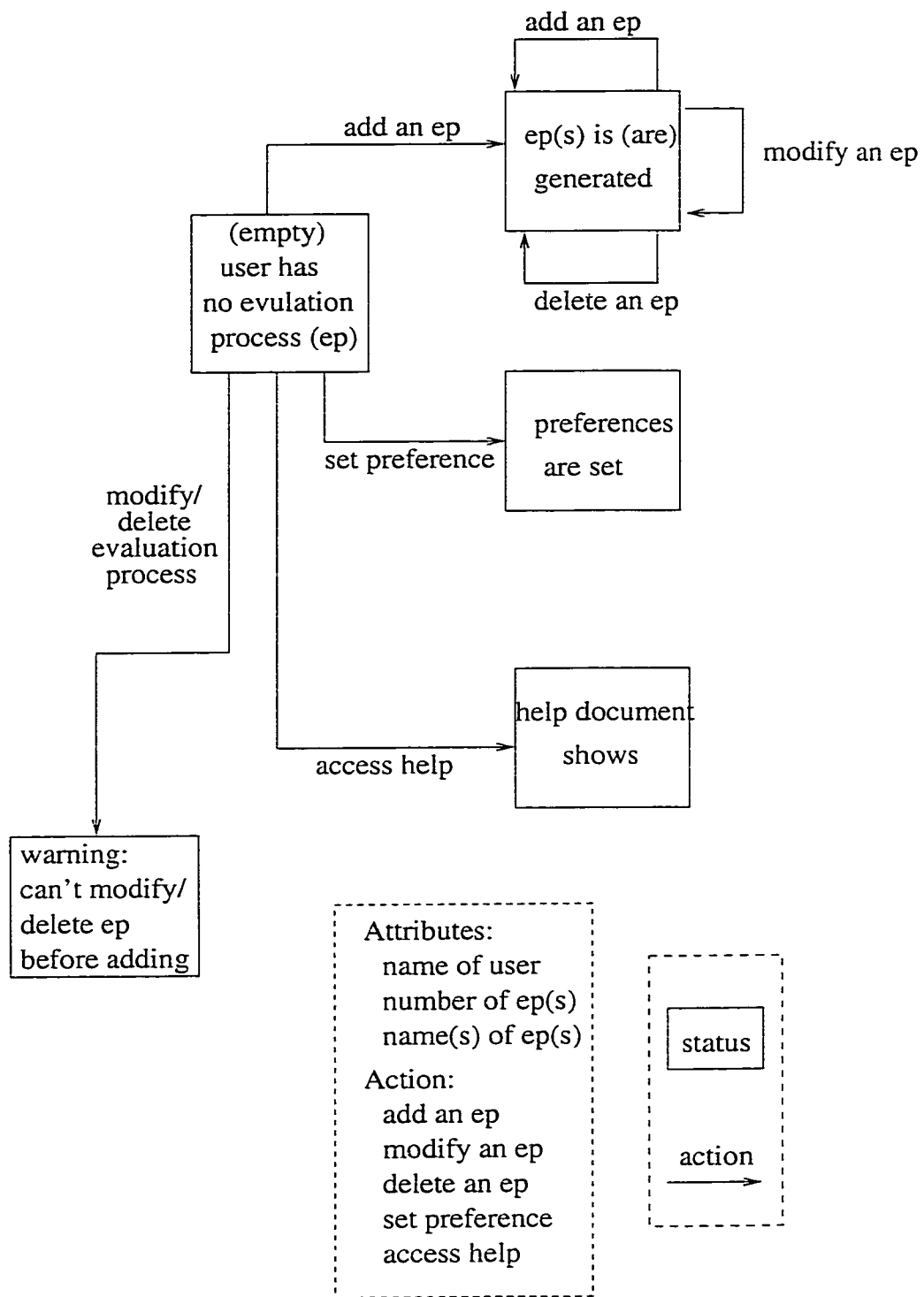


Figure 3.11: "User" object state diagram in envisioned task model

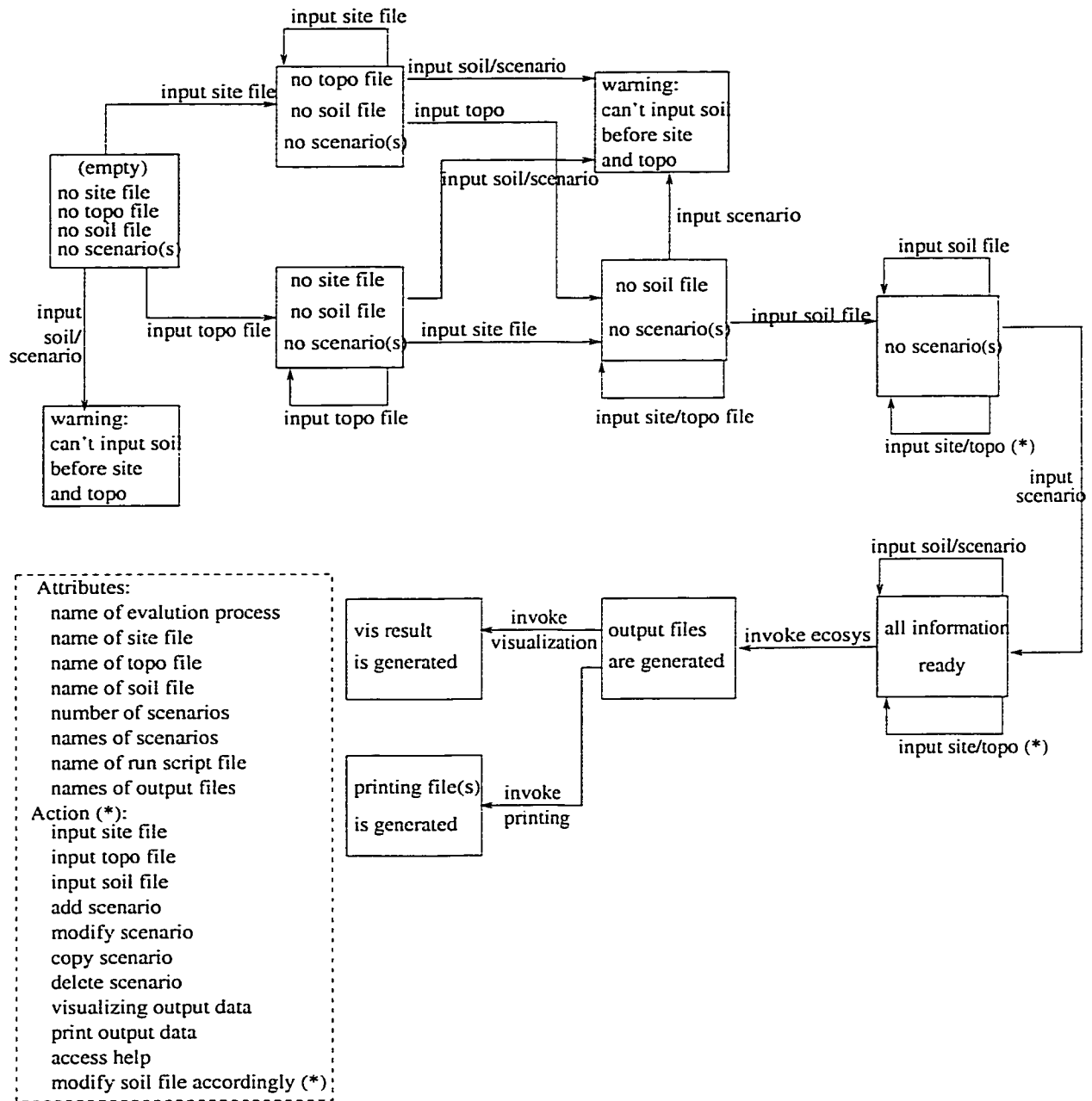


Figure 3.12: "Evaluation process" object state diagram in envisioned task model

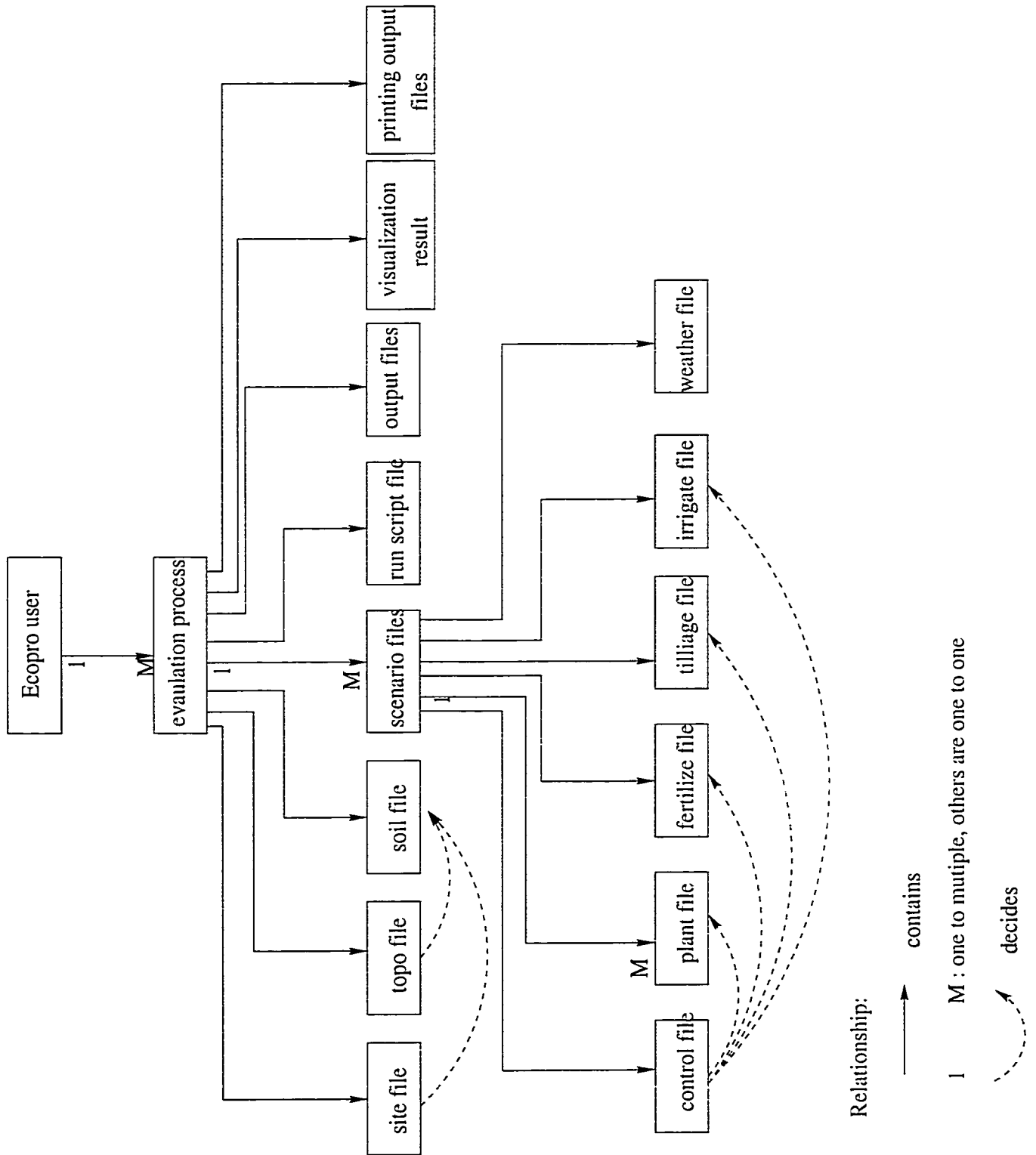


Figure 3.13: Object relationship diagram in envisioned task model

developing the task model are ultimately transferred to the user interface. Hence, all mistakes made during analysis or modeling normally remain undetected until the first prototype of the user interface is evaluated. To enable early discovery of mistakes, the task model itself has to be briefly tested.

In Ecopro, sample scenarios are used to test the task model. A key contribution of scenarios to interface design is that they provide a specific context for designers to analyze a task model. This is especially valuable in the present case because an object-oriented model is used as the envisioned task model. Since scenarios provide a view into the design's "flow of control" which is explicit in procedural designs, but is distributed across objects in object-oriented design. Sample scenarios are collected by the current designer imagining herself as the user trying to use the system in different situations.

When scenarios are used to go through the paths in the task model, thoroughness and preciseness are the main concerns. The current designer needs to find out whether the task structure in the model takes all possible aspects into consideration. For example, Does the task model show all possible tasks a user can perform in a given situation?, Is it sufficient and capable of supporting all the user tasks? For a specific task, does it clearly depict all the involved objects?

Once the scenario test is finished, the task model is assumed to be complete and specific enough for further design.



# Chapter 4

## Dialogue Design and Design Evaluation

The purpose of this chapter is to synthesize the visual representation of the system and its tasks. In the design of user interfaces, dialogue means the structure of the conversation between the user and the computer system. A dialogue model is used to describe the human-computer conversation. It describes when the end-user can invoke commands, select or specify input and when the computer can query the end-user and present information. Three levels of abstraction are employed to describe the present dialogue model. They are the lexical, syntactic and semantic levels.

In this chapter the user interface is designed first at the syntactic level by defining the structure of the interface. Then the design is linked to the task model. Afterwards, the physical screens of the user interface are organized and the dialogue modeling is completed. Finally prototyping techniques are used to present the interface to the user followed by design evaluation.

### 4.1 Syntactic level of dialogue modeling

During the syntactic level of dialogue modeling, the dialogue model represents the structure of the interface and the order of actions used to meet a main goal.

It is started by defining the necessary views for each object. A view can be considered as an abstract representation of an object in the user interface. The object state diagram and the object relationship diagrams from the envisioned task model can assist the designer in determining the necessary views of objects. Each

relationship identified for an object will result in a view, or part of a view of that object. The object corresponding to the view is called the viewed objects. Each dialogue view must provide attributes that control how they are presented to the end user, and methods that determine the high-level interactions that the end user can have with the view object.

In figure 4.1, it can be seen that there are views of the Ecopro user, evaluation process, scenario and each input file. They are called user view, evaluation process (ep) view, scenario view and file view. The diagram indicates that the user view shows the information displayed on the screen when the system is started. It includes one or several evaluation processes, system feedback, user preferences, timer and help documents. The evaluation process view shows information on scenarios, site, topographical and soil files, visualization results and help documents. The scenario view shows six input files, user preference and help documents. High level interactions are defined that may occur within each view. For example, within the user view, the user may add, modify or delete an evaluation process; he or she may also set preferences, read help documents and exit from the system.

As showed in this illustration, views are modeled as abstract objects themselves. The view of a file is not defined as a text area on the screen but simply as “the view of a file”. This allows the design methodology to be isolated from any specific visual or graphic design concerns and helps ensure accuracy and completeness of the view definition. It is known that an interface has to be maintained more often, probably more than any other parts of the system, therefore it is important that it is easy to change.

After defining the structure of the interface, the sequencing between different dialogue views is described. Augmented transition network (ATN)[Green, 1986] is used as part of the dialogue notation for syntactic dialogue modeling. It describes the progress of sequences of events within a system and has four basic components: a state is an object or entity that is represented as a circle; an event is something causing one state to finish and an object to change from one state into another that is represented as a arc; a register is a storage location that the transition network can set and test; a function refers to the system’s registers and the event that only occurs if the function is true. Translated into dialogue terms, a state will be the computer

awaiting a user's reply. The user's reply is an event that the interface has to deal with. It changes the interface from one state into another, if the user's interaction is allowed. The computer reacts to the user until it requires more human interaction. In this way the whole question and answer sequence in a dialogue can be described and planned.

Figure 4.2 is the ATN diagram for an initial view of Ecopro. As shown,  $s_1$  is the initial state. The events that may occur at  $s_1$  are  $e_1$ ,  $e_4$ ,  $e_6$  or  $e_8$ . If the user interacts with the interface by adding one evaluation process( $e_1$ ),  $f_1$  is first checked and the number of evaluation processes is increased. Then the corresponding action  $a_1$  is executed and the state of the view will change from  $s_1$  to  $s_2$  accordingly. The diagram describes the sequence of events that may happen and the corresponding state of views. Similarly figure 4.3 illustrates interactions which may occur in the evaluation process view including input files, adding scenarios, running ecosys, visualizing output data etc. The sequence of views is described by the arcs from one state to another. Object state diagrams from the envisioned task model are used again to identify the state of objects and the sequence of events.

ATNs are chosen instead of a grammar for dialogue modeling because a grammar is too complex for practical use and it is not good enough to give a clear illustration of event sequencing, though it has the advantage of making the structure of a dialogue clear and specifying permissible computer and human actions concisely. On the other hand, diagrams are good for sequences. They allow the designer to see at a glance the structure of the dialogue, though they often have trouble with more complex cases. The task structure of Ecopro is not very complicated. Therefore, using ATN diagrams is sufficient to generate a clear dialogue model. Another reason for the use of ATN diagrams is that it is easier to perform dialogue modeling because it matches object state diagrams in the envisioned task model very well.

## 4.2 Semantic level of dialogue modeling

The semantic level of dialogue modeling covers the extension of the dialogue model by determining the disposition of tasks across the views of objects. Actually this process is initiated in the syntactic level modeling. When the sequence of views is defined,

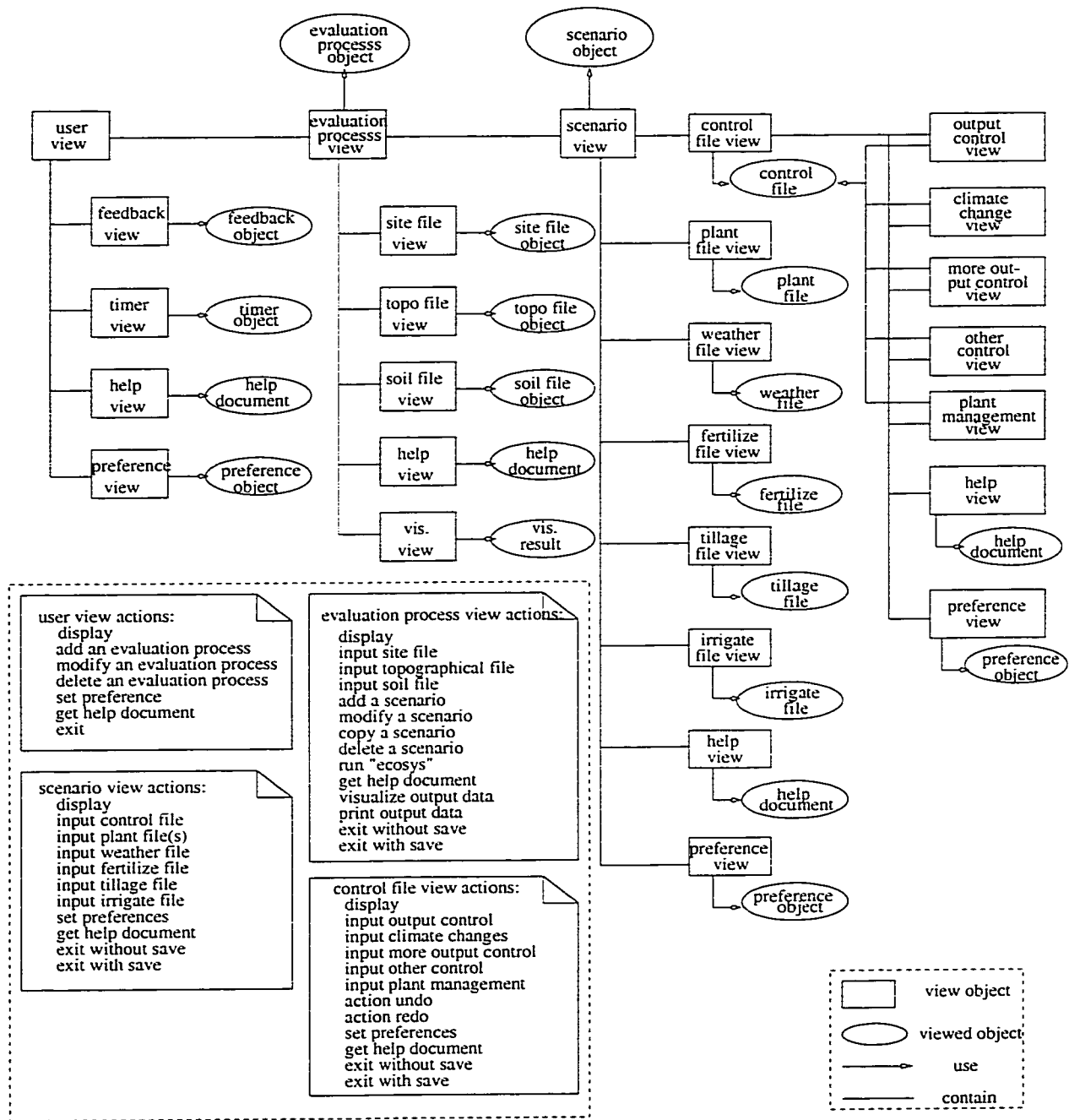
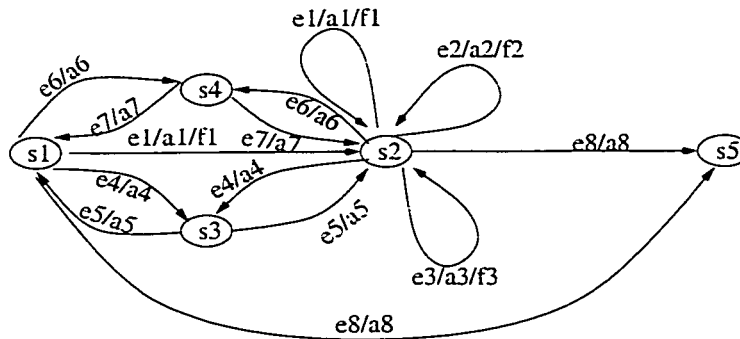


Figure 4.1: Object relationship diagram in dialogue modeling (I)



**register:**

r1: number of evaluation processes (ep)

**functions:**

f1: r1 = r1+1; return (true);

f2: if (r1>0) return (true);  
else return (false);

f3: if (r1>0) r1=r1+1; return (true);  
else return (false);

**events (interactions):**

e1: add one ep

e2: modify one ep

e3: delete one ep

e4: set preference

e5: close preference

e6: access help

e7: close help

e8: exit

**actions:**

a1: add a subview ep view, set its status=1;  
change feed back view accordingly

a2: add a subview ep view, set its status=2;  
change feed back view accordingly

a3: delete specific ep information  
change feed back view accordingly

a4: preference view open

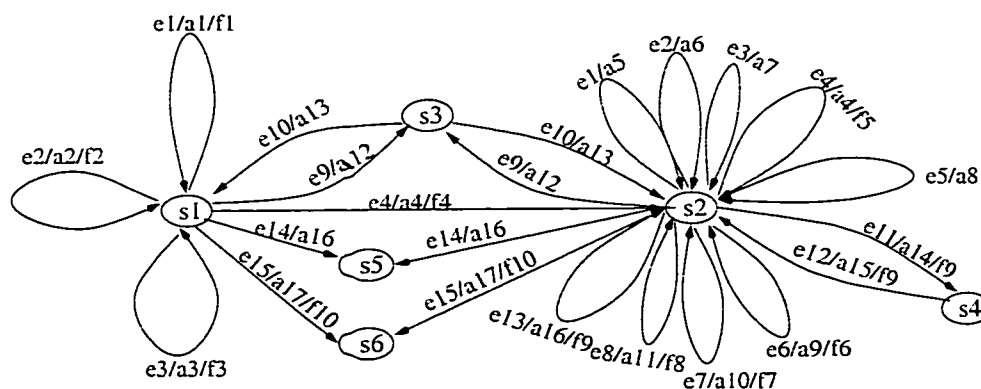
a5: preference view close

a6: help view open

a7: help view close

a8: Ecosys view close, exit

Figure 4.2: ATN diagram of initial "user" view



**register:**

- r1: site file finish input
- r2: topographic file finish input
- r3: soil file finish input
- r4: number of scenarios
- r5: output data is generated

**functions:**

- f1: r1=true; return (true);
- f2: r2=true; return (true);
- f3: if (r1 && r2) r3=true; return (true);  
else return (false);
- f4: if (r1 && r2 && r3) r4=1; return (true);  
else return (false);
- f5: r4=r4+1; return (true);
- f6: if (r4>0) r4=r4+1; return (true);  
else return (false);
- f7: if (r4>0) r4=r4-1; return (true);  
else return (false);
- f8: if (r4>0) r5=true; return (true);  
else return (false);
- f9: if (r5) return (true);  
else return (false);
- f10: if (r4>0) return (true);  
else return (false);

**events (interactions):**

- e1: input site file
- e2: input topographic file
- e3: input soil file
- e4: add a scenario
- e5: modify a scenario
- e6: copy a scenario
- e7: delete a scenario
- e8: invoke ecosys
- e9: access help
- e10: close help
- e11: visualize output
- e12: close visualization
- e13: print output
- e14: exit without saving
- e15: exit with saving

**actions:**

- a1: add a subview site view
- a2: add a subview topo view
- a3: add a subview soil view
- a4: add a subview scenario view
- a5: add a subview site view
- a6: add a subview topo view
- a7: add a subview soil view
- a8: add a subview scenario view
- a9: copy scenario information
- a10: delete scenario information
- a11: generate run script, call "ecosys"
- a12: help view open
- a13: help view close
- a14: visualization view open
- a15: visualization view close
- a16: ep view closes
- a17: ep view closes, save all information

Figure 4.3: ATN diagram of "evaluation process" view

each transition is linked to an action residing in the task model, since views and tasks are tightly coupled. At this stage only granularity not notation of the dialogue is changed.

Views are verified to check whether they support tasks in an appropriate way rather than merely representing the visual aspect of objects. No single view of an object should attempt to support too many tasks, because the view will be overly complex and those tasks will be difficult for the user to accomplish. If tasks, as identified in the task analysis, are grouped together for some reason, then such groupings should be maintained and incorporated in the view definition.

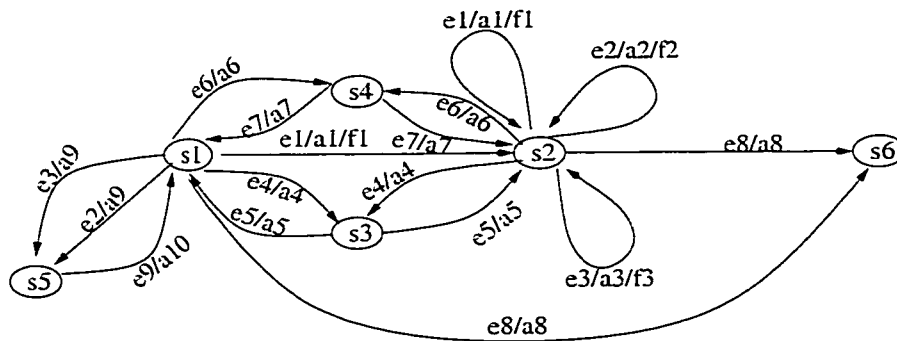
Once the views have been defined in terms of the tasks they support, the model can be checked in two steps. The first focuses on user interactions to see whether they are adequately specified and consistent. The second concerns the dialogue states, including those that the user wants to get to and those not.

At the first step, for each state of the dialogue, all events must be mentioned. This is called completeness. During this process, besides finding out that some states are missing certain events, it may also be discovered that some states have several arcs labeled with the same event. That is, the specifications are not exclusive for that action. At the second step, the user wants at least to be able to get to a desired dialogue state and ideally to be able to get there with ease. In general, characteristics like these are grouped under reachability. A basic check of reachability is whether there is full connection. That is, for any two states there is a sequence of actions which will take the user from the first state to the second.

After the completeness check, an error feedback view is added to the evaluation process view and the scenario view as an error pathway. This will ensure that the user will not stumble even if they don't understand the sequence of tasks within each view. Figure 4.4 is the improvement of figure 4.2 after dialogue model checking.

### **4.3 Interaction style**

After refinement of the semantic level of the dialogue model, the designer can create the metaphoric representations of the interface objects and actions. This begins with the selection of appropriate interaction styles.



**register:**

r1: number of evaluation processes (ep)

**functions:**

f1: r1 = r1+1; return (true);

f2: if (r1>0) return (true);  
else return (false);

f3: if (r1>0) r1=r1+1; return (true);  
else return (false);

**events (interactions):**

e1: add one ep

e2: modify one ep

e3: delete one ep

e4: set preference

e5: close preference

e6: access help

e7: close help

e8: exit

e9: close error feed back

**actions:**

a1: add a subview ep view, set its status=1;  
change feed back view accordingly

a2: add a subview ep view, set its status= 2;  
change feed back view accordingly

a3: delete specific ep information  
change feed back view accordingly

a4: preference view open

a5: preference view close

a6: help view open

a7: help view close

a8: Ecosys view close, exit

a9: error feed back view open

a10: error feed back view close

Figure 4.4: ATN diagram of "user view" after completeness and reachability check



An interaction style is the way in which the user addresses data or functions provided by the system. For any given task interaction, there may be several different techniques to perform it. Consequently, when choosing a particular interaction style, the designer needs to consider the merits and limitations of each particular type, the task on which it is applied, and the usability issues.

Direct manipulation (DM)[Shneiderman, 1987] is selected as the main interaction style for commands, while fill-in forms are selected for data entry, editing and display in the present design. DM is selected due to its advantage of ease to learn, which is a main concern in reaching usability targets. DM interfaces tap the user models that people already have (manipulating objects in two dimensional space), thus they are relatively easy to learn and remember. Another reason is the popularity that DM has as the interaction style in many other applications. It takes little time for the user to get familiar with it. The disadvantage of DM is that there is little or no prompting in a DM interface, thus it is not intuitive to the first-time user. Because of this, a high level of support needs to be provided by immediate feedback, concrete and realistic icons etc. Form filling is selected because of its suitability for data entry and display. It provides context and self-explanation, requires little memory and makes efficient use of screen space. Though it has the disadvantage of inflexibility, it is not a main concern in the present design as compared to other usability issues.

A high level visual presentation of the interface is generated after the interaction style is defined. Views which contain more actions (ep view, scenario view, file view) can be represented by windows, and within those views, icons and pop-up menus are provided representing objects and commands for user interaction.

## **4.4 Lexical level of dialogue modeling**

This is the process for completing the remainder of the concrete user interface Design, so that users can decompose their plan into a series of intermediate actions, all the way down to a series of detailed keystrokes and clicks.

Two pieces of information are used to accomplish this objective. The first is the object diagram, which indicates all views to be shown and their relationship. The other is the ATN diagrams of views that show how views may vary. User views

which may be generated during the user analysis stage could also be used if there are any. Besides the above, there are established interface design principles [shneiderman, 1998] such as guidelines for graphics design, use of color, data entry, and data display which are used as a basis for our final decision.

From the information mentioned above, an initial layout is designed for each view in each of its states. For example, an evaluation process view includes components such as scenario views, site file view, topographic file view, soil file view, and feedback view.

Control commands of each view in each of its states are also identified. Again, those controls should be defined abstractly as an object. That is to say, rather than defining a control as a “button”, such controls may be defined as something like a “triggering” control. This abstraction allows a complete and accurate interaction definition while still allowing for flexibility or variation in the final design. For example, the ‘triggering control’ may be represented by a button, a menu item, or an icon. When the design is further defined in terms of controls, it is important to update the object diagram. In the object model, such controls are objects; and for the model to be complete they must be reflected in the object diagrams such as in figure 4.5.

Components within each view are then represented by detailed graphical widgets. The graphical widgets should be designed so as to help the user to understand the interaction and correctly interpret the information displayed on the screen. They can take many different forms or can be hidden behind some metaphor which is familiar to the user. For example, for the evaluation process view in Ecopro, considering the one to many relationship between evaluation process view and scenario views, a list is used as a container for all scenario views within one evaluation process. Dialogue windows are used as error message views. And realistic icons are used to represent site, topographic and soil file views. When the icons are selected, the corresponding files are opened. All widgets are clearly defined. For example, the content within each widget, the relationships with other widgets, and the user actions they allow.

In Ecopro, more attention is paid to data entry as it is one of the major user tasks. The purpose of data entry is to minimize the user’s work load, and make entry error rate as low as possible. This is achieved by keeping the user’s memory load to a minimum, setting defaults for commonly entered items, giving clear explanations

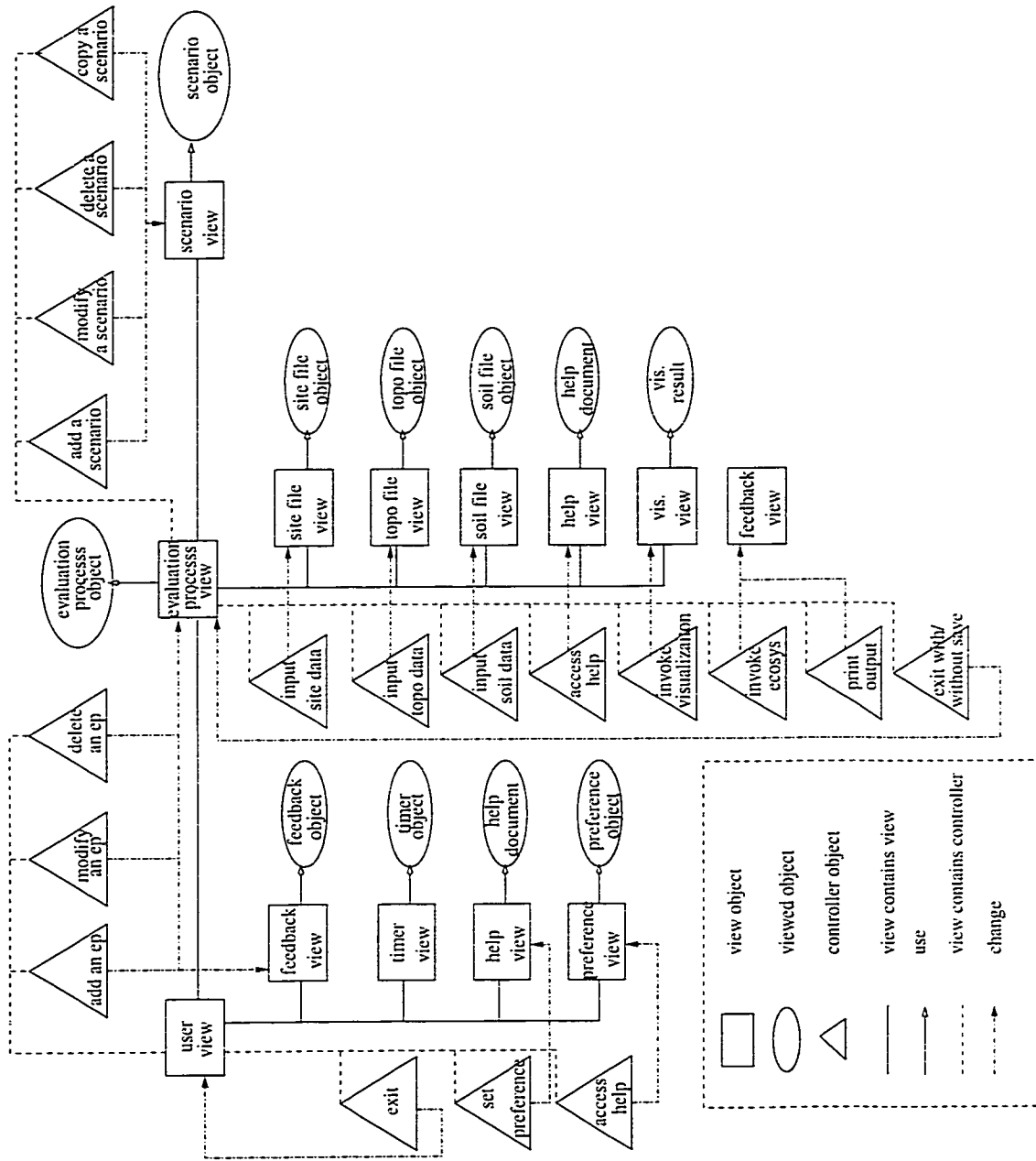


Figure 4.5: Object relationship diagram in dialogue modeling (II)

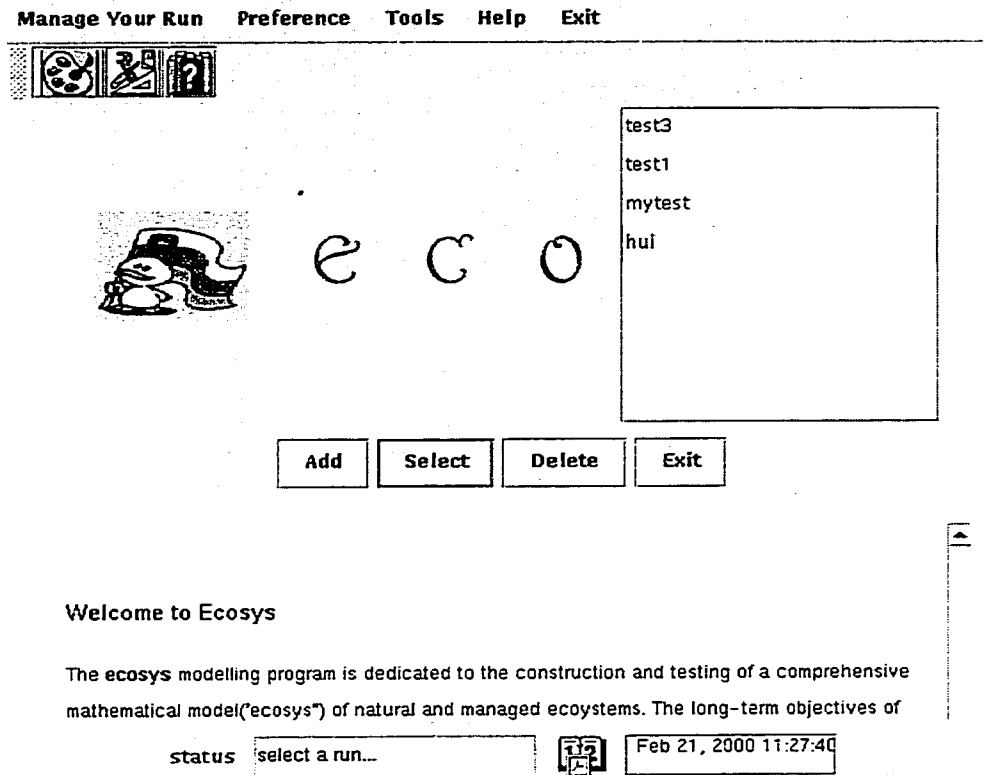


Figure 4.6: Prototyping of initial view of Ecopro (I)

of content of each item, providing data type checking, and providing undo and redo options. However, the ability such as skipping automatically from one data field to another is not provided. Instead the user is given the flexibility to control the sequence of communication. Consistency is provided by using similar display formats and control interactions among all views which are used for data entry.

Now we have a specific system image which could be used for further prototyping. A paper sketch technique is used for prototype development. All views in each state are drawn and shown to the user for walk-through evaluation. The following figures illustrate a few examples.

## 4.5 Design evaluation

A cognitive walk-through is used for design evaluation because it focuses on evaluating a design for ease of learning, which is the most important usability issue in our design. It is a usability inspection method that evaluates a design for ease of

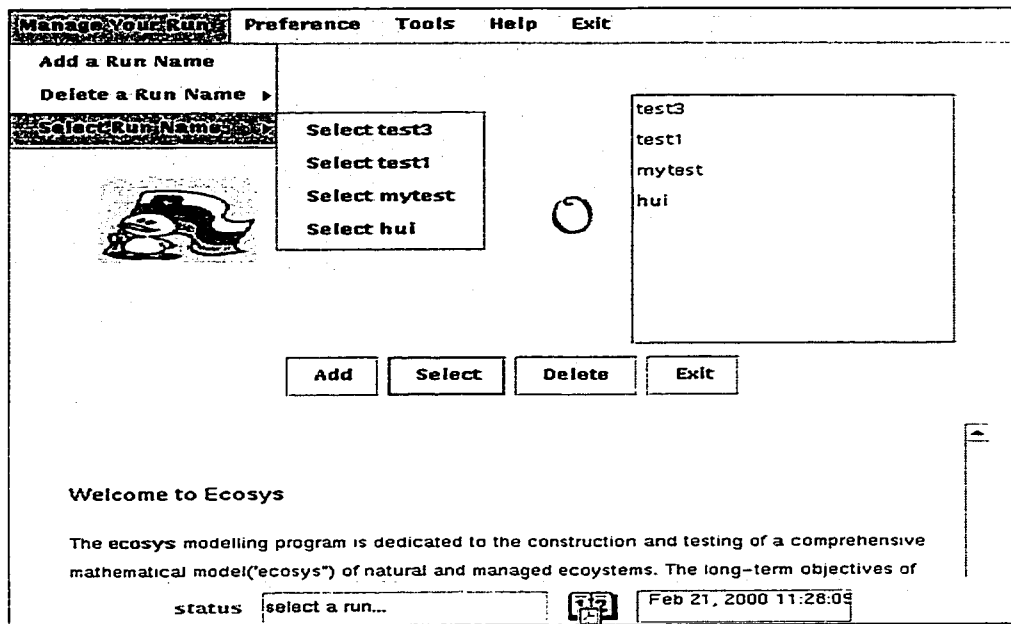


Figure 4.7: Prototyping of initial view of Ecopro (II)

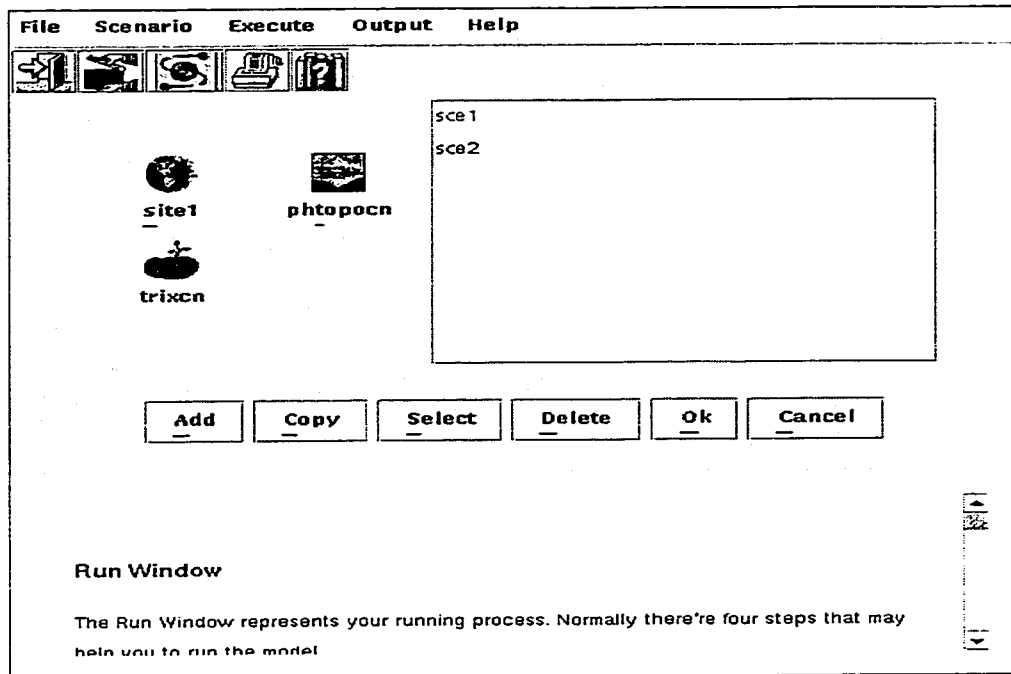


Figure 4.8: Prototyping of evaluation process view (I)

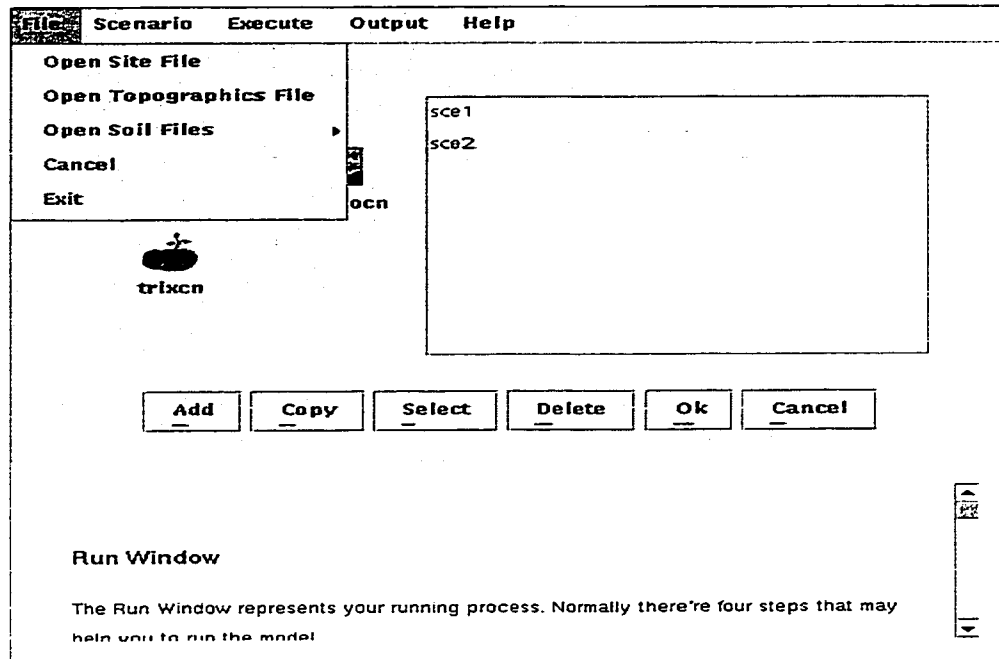


Figure 4.9: Prototyping of evaluation process view (II)

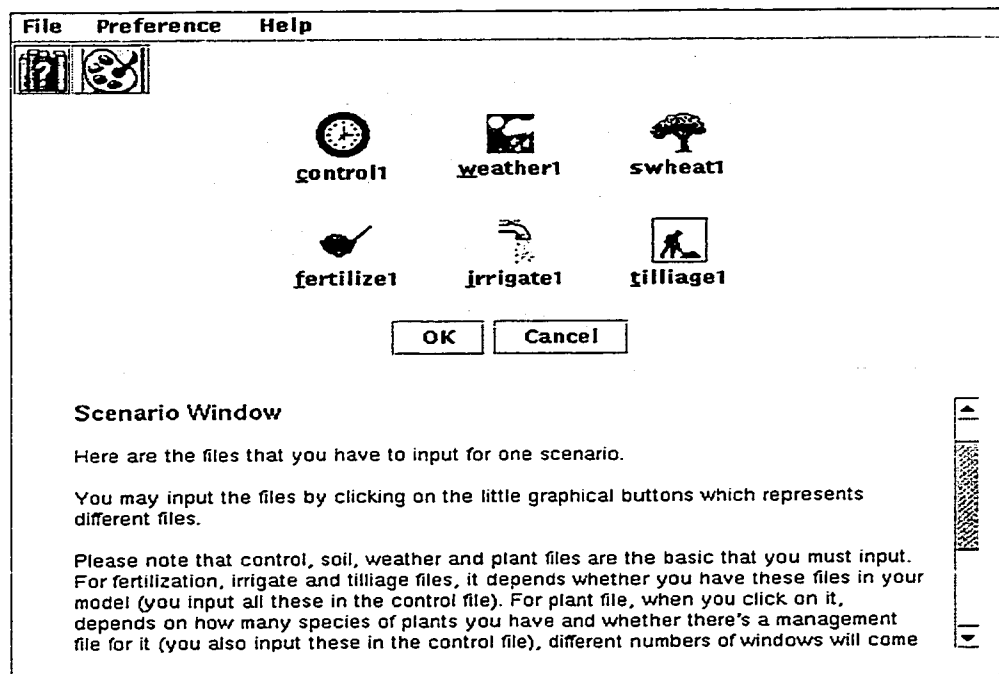


Figure 4.10: Prototyping of scenario view

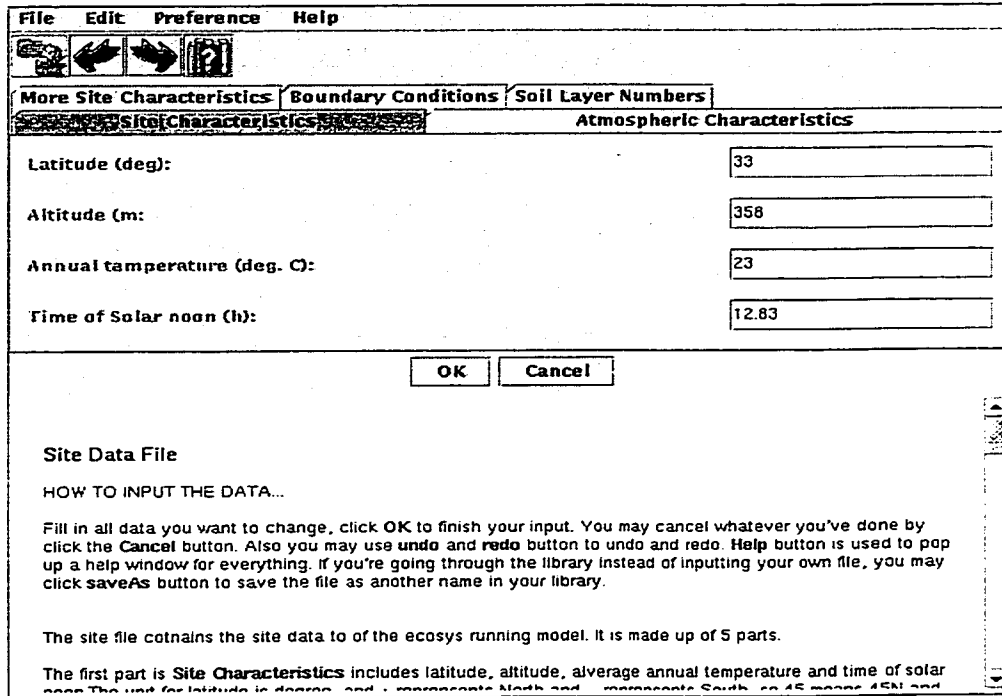


Figure 4.11: Prototyping of site file view (I)

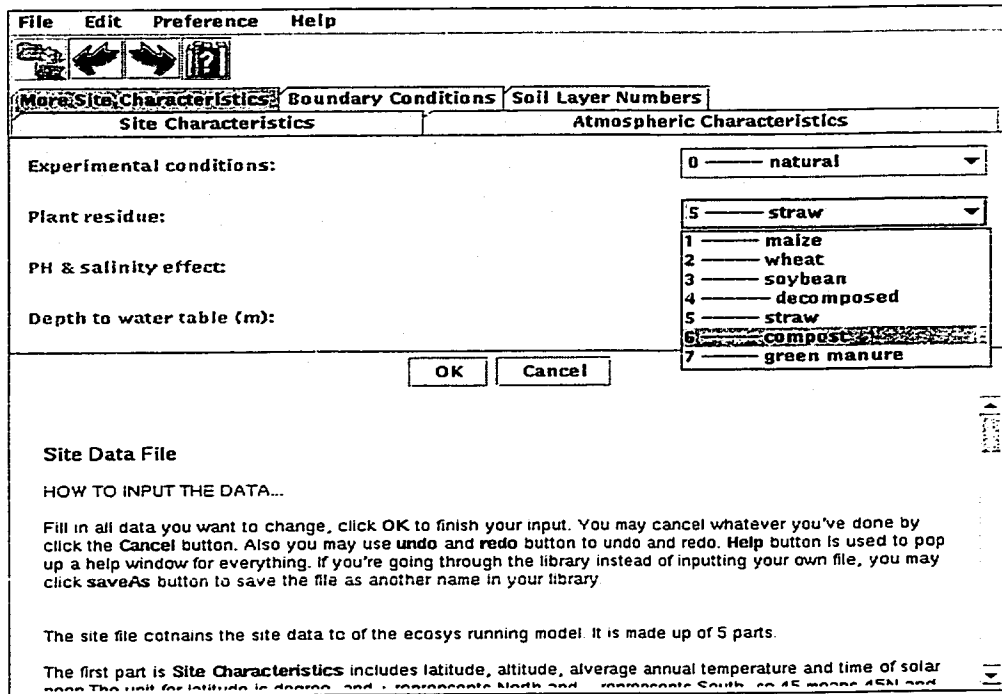


Figure 4.12: Prototyping of site file view (II)

learning by exploration. The method finds mismatches between user's and designers' conceptualization of a task, such as poor wording choices for menu titles and button labels, and inadequate feedback about the consequences of an action etc.

In the cognitive walk-through process, the reviewers evaluate a proposed interface in the context of one or more specific user tasks. The input to a walk-through session includes an interface's detailed design description, task scenarios, and a sequence of actions that a user should successfully perform to complete the designed tasks. The steps to perform a cognitive walk-through are as follows:

1. select sample tasks for evaluation.
2. define action sequence for completing the tasks.
3. walk-through the action sequences for each task.
4. record critical information, such as user knowledge requirements, side issues and design change.

A walk-through involves a detailed analysis of the tasks. In general, the analysis should be limited to a reasonable but representative collection of benchmark tasks. Task selection should be based on results of task analysis which means functionality and usability targets are the main concerns. Our sample tasks include basic functions such as data input or modification of evaluation processes and invoking ecosys. Situations in which users often make mistakes are also considered for task selection.

After task selection, for each task, a description of the sequence of actions that occur to accomplish the task within the current interface is generated. These actions may be simple movements, such as "move cursor to File menu", or they may be group of simple actions that a user could typically execute as a block. A sequence of actions is defined by describing the prompts preceding every action required to accomplish the task as well as the reactions of the interface to each of the actions. For example, if the user wants to input a scenario, a dialog box will pop up asking the name of scenario, then a scenario view will be opened and the name of that scenario will be added to the scenario list in the evaluation process view.

After sample task selection and action sequence definition, a walk-through begins with examining each action in the solution path and attempting to tell a credible



story as to why the expected users would choose that action. Credible stories are based on assumptions about the user's background knowledge and goals, and on an understanding of the problem-solving process that enables a user to guess the correct action. As the walk-through proceeds, we consider the following questions:

- Will the user try to achieve the right effect?
- Will the user notice that the correct actions are available?
- Will the user associate the correct action with the effect that the user is trying to achieve?
- If the correct action is performed, will the user see that progress is being made towards a solution for the task?

Notes are taken while performing the evaluation. The notes include both user information and design information. The first is related to what the user must know prior to performing the task and what the user should learn while performing the task. The second includes issues of inappropriate design with detailed context so that decisions may be made for reconstructing the interface at a later time. For example, inappropriate design related to improperly named menu item, or confusing action prediction. In the evaluation process for the Ecopro design, we find that providing defaults for data entry is not powerful enough for cases that need huge amounts of user input. Therefore another function is added into the task model that provide a library including the necessary information associated with different scenarios. A user may copy the data from his or her library so that the amount of input information that must be entered is reduced significantly.

After prototyping and design evaluation, the task model and the design model are refined accordingly and the system implementation may be started.

# Chapter 5

## System implementation

The previous four chapters detail a black box functional description of the system's desired behavior. In this chapter a high level description of the Ecopro system structure is presented using UML notations. The implementation model is used to describe how objects in the problem domain collaborate to provide the desired system behavior. Two aspects of the implementation model are considered in Ecopro. One is the system static model which describes classes and their associations. The other is the dynamic model which describes object interactions among classes.

### 5.1 Basic classes in Ecopro

The static model is developed by first defining the main classes, their associations and interactions. Subclasses are defined later to fulfill the implementation details. The dynamic model is developed later by identifying the interactions among these classes. UML class diagrams and collaboration diagrams are used to represent the static and dynamic domain models respectively.

#### 5.1.1 Identification of the main classes

Since MVC is utilized as the framework for implementing Ecopro, it is obvious to identify the top three main classes as, Model, View and Controller. These classes hold the generic behaviors and states of the three parts of MVC. Concrete subclasses are defined later to hold the specific states and behaviors for Ecopro functionality and user interface components.

The Model class defines the problem-domain that will be solved by the user interface. The View class displays output to the user (e.g., display boxes, colored symbols, visualizations, etc.). The Controller class receives input from the user (e.g., mouse movement, keyboard input, button clicks, key presses, etc.).

Java is used as the implementation language in Ecopro. It integrates the MVC paradigm into its structure by providing two components named Observable class and Observer interface. Class Observable represents a model object in MVC. Its states can be observed by an Observer which represents a view in MVC. Another component in Java which may be thought of as representing the Controller of MVC is the EventListener interface. Different types of listeners catch different interactions between user and interface.

Based on the above notation, Ecopro class “Model” is defined as a subclass of the Java Observable class and “View” is defined as a subclass of Observer. “Controller” is an abstract class and its concrete subclasses implement different types of EventListener interfaces depending on its corresponding view type.

### **5.1.2 Identification of the semantic component of each main class**

The MVC structure defines an “observer relationship”. The model contains a pointer to a list of its corresponding view objects. This pointer is only a base class pointer; the model knows nothing about the kind of view which it observes. By contrast, the view knows exactly what kind of model it is observing. For this purpose, it has a pointer to the model. In addition, the view also has a pointer to the controller. The controller has pointers to both the model and the view.

The attributes and operations in these three classes are described in Figure 5.1.

In the Model class, methods “addObserver ” and “deleteObserver” are provided to add or delete a view from the observer list. The method “NotifyObserver” is used by the model to notify all its corresponding views in the observer list that the status of the model has changed. The “setChanged” method gives the ability to change the status of the model. The primary responsibility of the view is keeping the MVC triad together as a family with consistent information. Two attributes are defined in the View class to represent its corresponding model and controller. A view

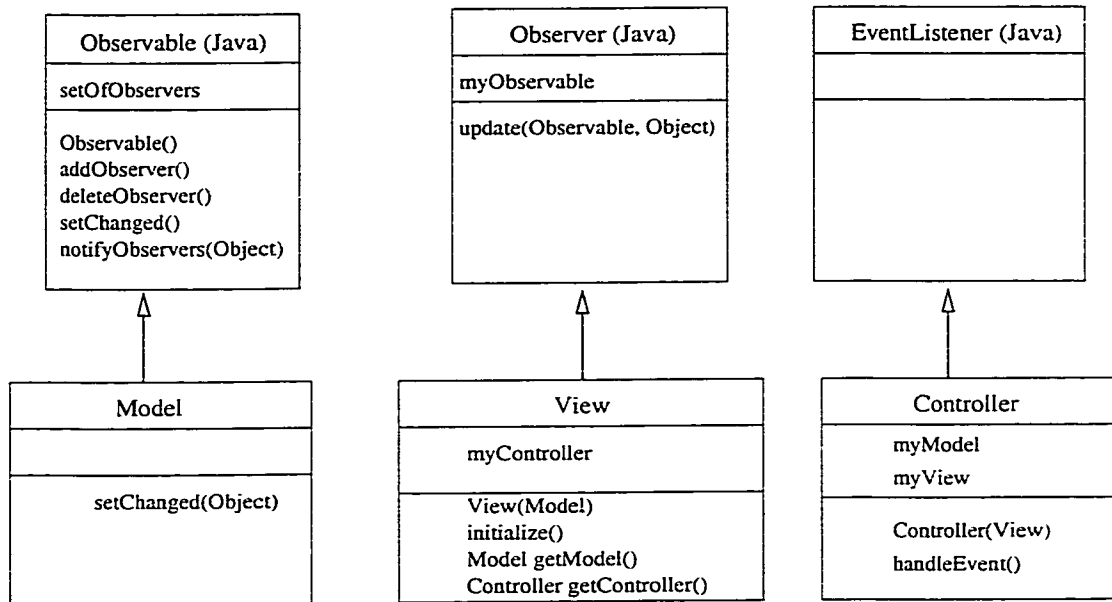


Figure 5.1: A high level class diagram of Model-View-Controller in Ecopro

sets up its model by invoking the “addObserver” method in the Model class when it is instantiated. Its controller is established as an instance of the corresponding controller class when the “initialize” method of the View class is invoked. Hence the view’s constructor together with its “initialize” method is sufficient for setting up the MVC structure. It is a controller’s job to handle the control of a model and a particular view. The class Controller contains two attributes “myModel” and “myView” pointing to its model and view. The method “handleEvent” is provided to detect user interactions. Concrete subclasses of Controller implement different types of Java EventListener interfaces and override the method according to their specific views.

### 5.1.3 Identification of interactions among main classes

To identify various interactions among classes, the dynamic model is explored to describe how the objects collaborate to provide the behavior described by user requirements. Basically two aspects are considered when identifying the interactions. The first is the set up process for the MVC triad, and the second is the work mechanism for each triad.

Figure 5.2 illustrates the set up process for a MVC triad. First a model is created

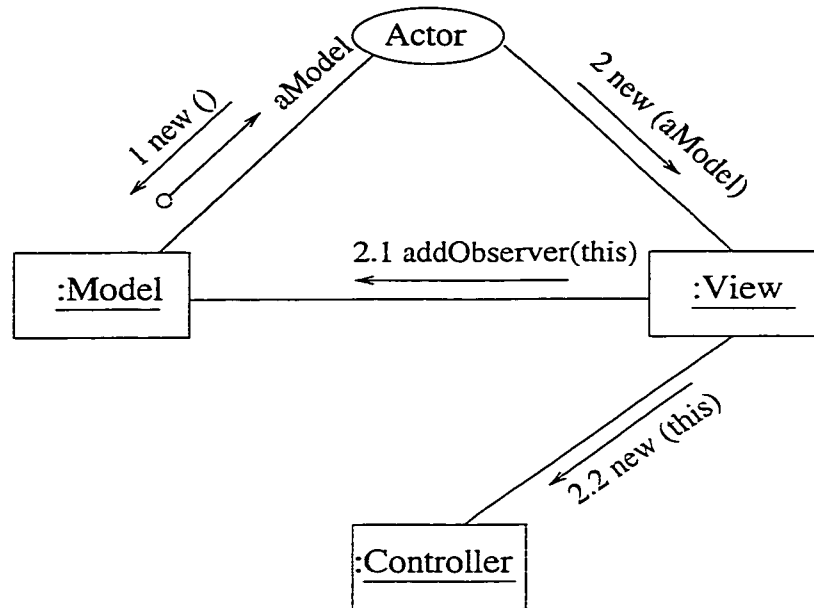


Figure 5.2: MVC triad set up process

by invoking the constructor of the Model class. Its status is set up by assigning the initial values of internal data and defining the observer list to be null. Then a view is created by setting its model to the newly constructed model and linking itself to the observer list of this model. Finally a view sets up its controller by invoking the constructor of the Controller class. The controller links the view to itself and defines the model of the view as its model.

After the MVC triad is set up, when the user interacts with the interface, the “handleEvent” method is triggered. The controller catches this event, interprets it and invokes a service procedure of the model by sending the message “setChanged” and passing the event. The model changes its status accordingly and notifies all its associated views about the change. The “update” method of each view is then invoked. This alerts the view that something has changed in the model and causes them to re-draw themselves. The details of the MVC work mechanism interaction are described in figure 5.3.

## 5.2 Subclasses

Subclasses of Model, View and Controller are the various classes that hold specific states and behaviors for Ecopro functionality and user interface components. With

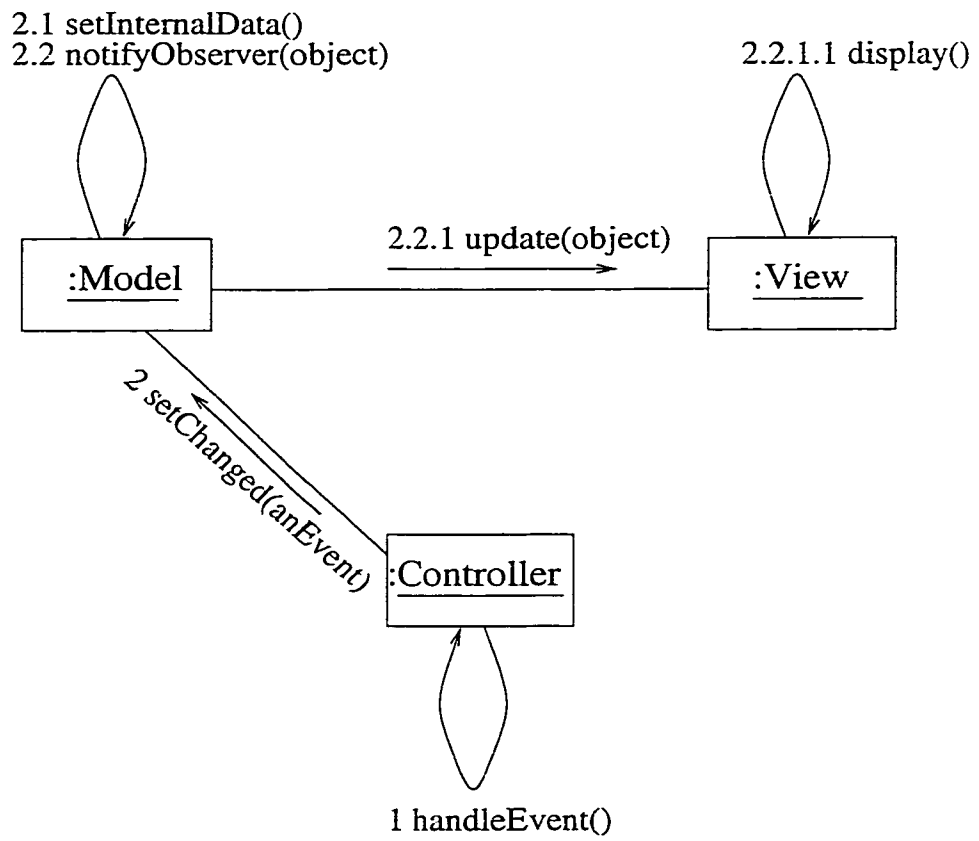


Figure 5.3: MVC triad work process

the help of the object relationship diagram generated in dialogue modeling (such as figure 4.5), it becomes intuitive to determine the different subclasses and their role in a MVC triad.

### 5.2.1 Subclasses of Model

The viewed objects in the object relationship diagrams depicted in the dialogue modeling stage (figure 4.5) can be considered as subclass of Model, since they represent the problem domain which holds the states and values of all objects required for Eco-pro functionality. The subclasses of Model include User, EProcess, Scenario, File, HelpDoc, Library etc. The aggregation relationship among these classes is shown in figure 5.4.

The User class represents all the objects used by a user who is running the system. These objects include a help document, a set of files and their statuses which are necessary for various evaluation processes running. They are represented by HelpDoc and EProcess classes respectively. Each user may have one or several instances of EProcess.

The Scenario class represents all the objects a user requires within a scenario. Each instance of the EProcess class contains one or several instances of scenario class. Each instance of EProcess also includes a library system and a set of output data which are represented by the classes Library and OutputData respectively. The library system contains the default value of all information which is necessary to run ecosys. The user may customize his/her library and copy required data from the library to his/her run processes instead of manually inputting it one by one.

The OutputData class is the output of the ecosys system run and may be used for future visualization. The File class in Eco-pro represents a file and its status. Each instance of Scenario contains several instances of File which represents different types of files. For example, SiteFile represents the site file and TopoFile represents the topographic file. Figure 5.5 illustrate the inheritance relationship among different classes.

After identifying the subclasses, attributes and methods may be defined accordingly for each subclass to provide the desired Eco-pro functionality. Object status diagram which are developed in the envision model generation stage are used for this

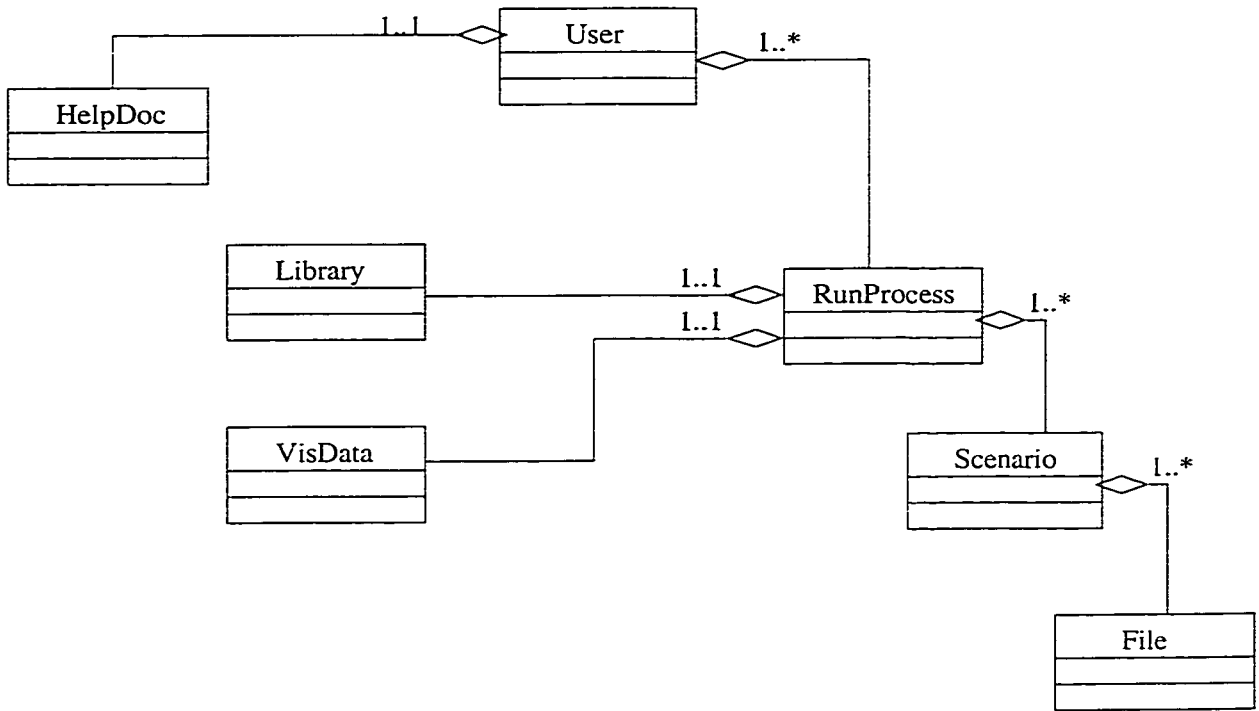


Figure 5.4: Class diagram of subclasses of Model in Ecopro (I)



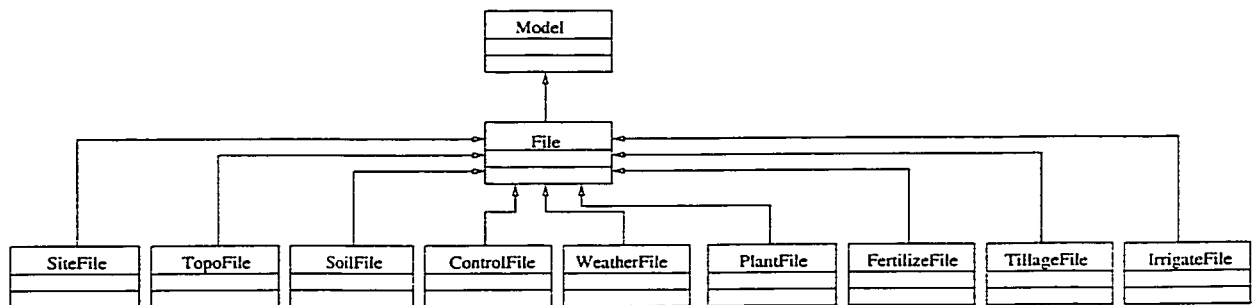
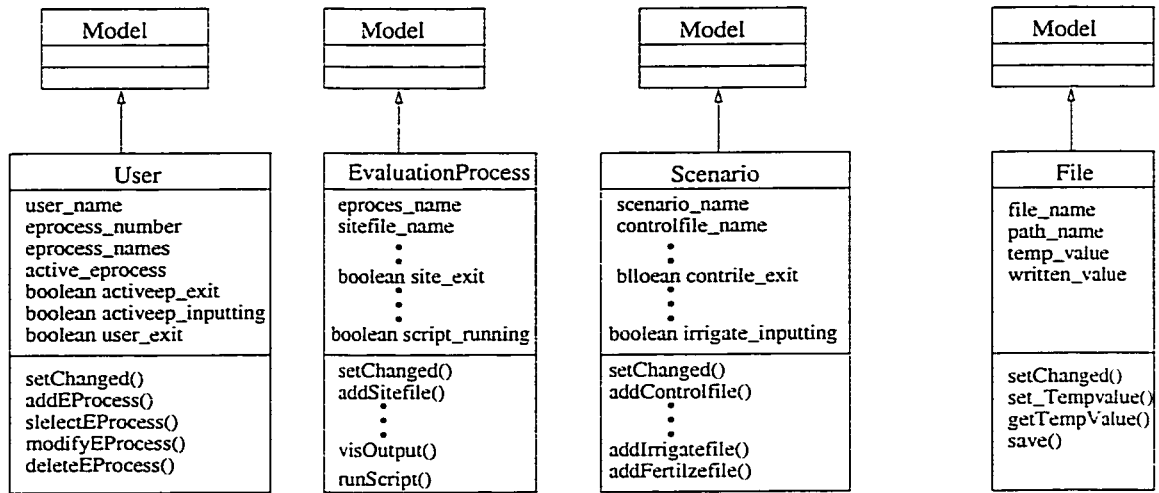


Figure 5.5: Class diagram of subclasses of Model in Ecopro (II)

purpose. For example, as shown in figure 3.11, the user class may have attributes such as name of the user, number of EProcess, name of each EProcess and currently active EProcess. Boolean attributes are added to distinguish different status of the model. For example, if `activeep_exit` is false, it implies that no evaluation process has been defined. If `activeep_inputting` is true, it implies that the user is inputting something and consequently the corresponding view of the active EProcess is plugged in at this moment. The method “SetChanged” is overridden to handle the internal status of each subclass. For example, in the User class, “setChanged” will invoke “addEProcess”, “selectEProcess”, “modifyEProcess” or “deleteEProcess” separately according to the event passed by its controller.

## 5.2.2 Subclasses of View

In object relationship diagrams (figure 4.5), all view objects together with the visual appearance of all controller objects within it are defined. They are considered to be subclasses of View. For example, in user view, the controller objects are add, modify, select and delete. The visual appearance of these objects is a set of menus and buttons. Hence the user view together with the set of menus and buttons are defined as the `UserView` class which inherits from View. Figure 5.6 illustrate subclasses of View and their aggregation relationship.

In `Ecopro`, `UserView` is the corresponding view of the User class. It is the initial window which appears when `Ecopro` is starting. Within this view, there is an `EProcessView` which displays the information for an evaluation process, a `HelpView` which displays the help documents, a `FeedBack View` which notifies the user about the status of `Ecopro`, and a `TimeView` which displays the date and time. Each instance of `EProcessView` has one or several instances of `ScenarioView` as its sub view. Each `ScenarioView` contains a set of menus, buttons and graphic icons to handle the different files (such as weather file, plant file etc )within a scenerio. In `Ecopro`, the `FileView` is responsible for showing the data values in a file. It contains a set of menus and buttons for user interaction control, a tab panel to display the data and a help view to show the help documents.

The main purpose of View is to display current status of its model. Attributes and operations are set for this purpose. Basically the “update” method is overridden

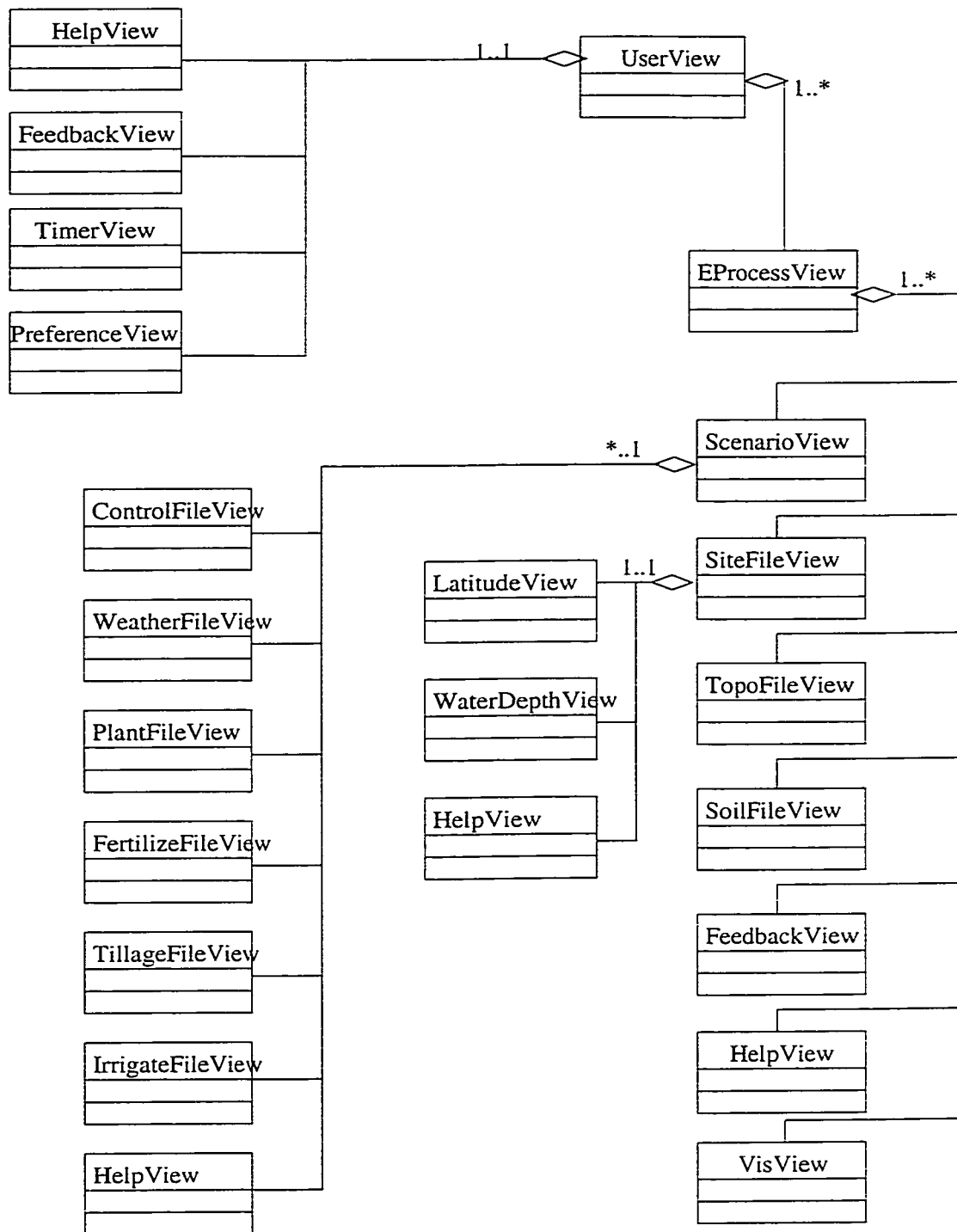


Figure 5.6: Class diagram of subclasses of View in Ecopro

in each subclass to draw itself according to the status of its model and the layout of the components within the view.

### **5.2.3 Subclasses of Controller**

In Ecopro, subclasses of Controller implement different types of Java EventListener interfaces according to its specific corresponding view.

As described earlier, each view contains the visual appearance of one or several controller objects whose visual appearance form a part of the view. For example, the set of menus, buttons and a list is part of the UserView. The corresponding type of EventListener for this part is defined as the controller of that particular view. Hence the controller of UserView (UserController) implements ActionListener and ListSelectionListener.

Since controller and view are tightly coupled, the relationships between controllers are the same as those between their views as shown in figure 5.6. Figure 5.7 illustrates subclasses of Controller and their relationships.

Subclasses of Controller contain the behavior for identifying the type of interaction between user and the system. Thus “handleEvent” is overridden in each subclass accordingly. Basically it catches user interactions within its view, identifies the type of event and invokes the “setChanged” method of its model. The details of interactions among the MVC triad in Ecopro are explained below.

## **5.3 Interactions among subclasses**

Interactions within each MVC triad are the same as those mentioned before for the MVC work mechanism. For example, in the case of user inputting or modifying a site file, the model is SiteFile; the view (SiteFileView) is a window which contains a set of text fields, menus and control buttons; and its controller is SiteFileController which implements ActionListener and DocumentListener.

When a user inputs or modifies a data value, the “handleEvent” operation is triggered. The SiteFileController catches the event and records the value that the user inputs. It then passes the message “setChanged” together with the new value to the model. SiteFile changes its internal data to the new value and notifies SiteFileView

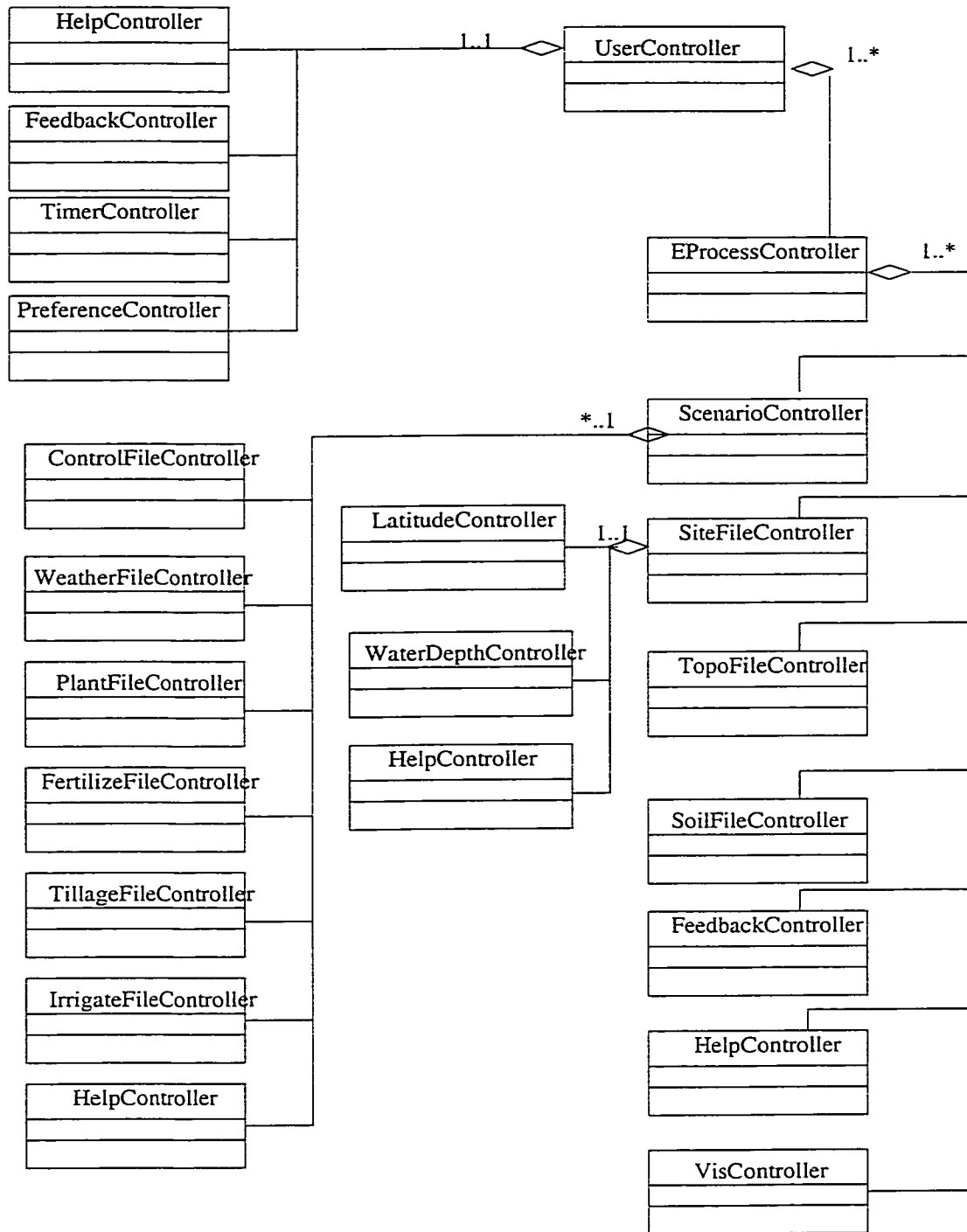


Figure 5.7: Class diagram of subclasses of Controller in Ecopro

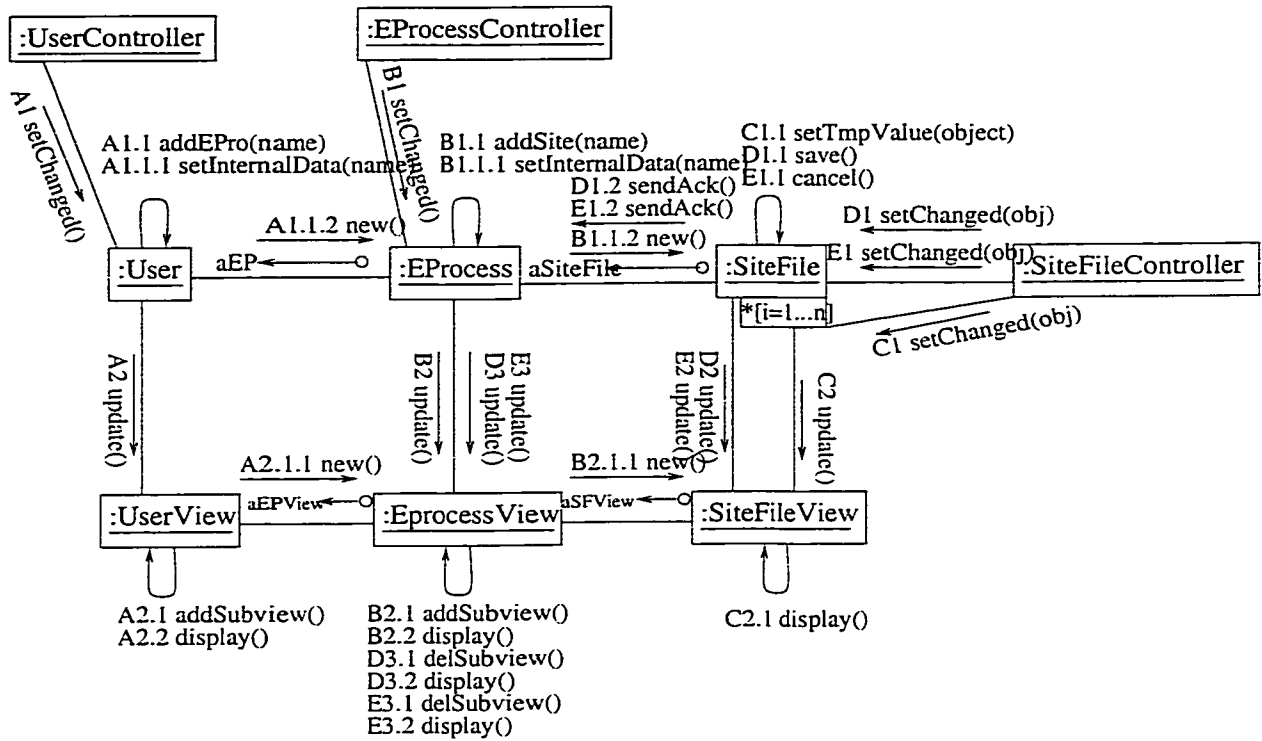


Figure 5.8: Collaboration diagram of “input file” scenario in Ecopro

that the value has changed. The SiteFileView then redisplay the text fields and redraws itself with the new values.

In Ecopro, MVC triads are tied together in a way that a model in one triad may change the status of a model in another triad when the user interacts with its interface. For example, if the user inputs a weather file into a scenario, the scenario model is changed to include the weather file. It also changes the status of the weather file model from “not available” to “available”. Thus consecutive interactions and status changes of the models make all MVC triads work together.

Figure 5.8 demonstrates a use case scenario where a user starts the system and inputs a site file. Initially a MVC triad (User-UserView-UserController) is available to fulfill the scenario. It gets launched when the user clicks the button “add EProcess” in the UserView. The UserController catches the event and passes the message “setChanged” together with the event to the model (User). This causes the User to understand that the user wants to add an evaluation process (EProcess), and the “addEProcess” method is invoked. The User first sets up its internal data, such as the name of the Eprocess and increases the number of EProcesses by one. It then

creates a new instance of EProcess. After this, the User notifies its corresponding view (UserView) by invoking the “update” method. The UserView gets informed that the status of User has changed and there is an EProcess object that has been added. The UserView then adds another sub view (EProcessView) to itself and redisplay itself. At this time, another MVC (EProcess-EProcessView-EProcessController) triad is created and its view is ready for further user interaction. Now, if the user adds a site file to the newly created EProcess, The EProcessController catches the event and asks the model (EProcess) to change its status by creating a new instance of SiteFile. The EProcess model then, notifies EProcessView to update itself by adding a sub view (SiteFileView). Thus a new instance of SiteFileView is created and the user may start to input the data. Within the SiteFileView when the user inputs any value, SiteFileController catches the event and passes it to the SiteFile. The SiteFile changes its internal data to the new value and invokes the “update” method. As a result the SiteFileView displays the new value according to user input. When the user clicks the “save file” or “cancel file” button, the SiteFileController catches the event and passes it to the SiteFile. The SiteFile then changes its status and sends an acknowledgement back to the model (EProcess), which tells the model whether the SiteFile has been successfully input or not. The EProcess changes its boolean instance variable according to the type of acknowledgment that it gets from the SiteFile and invokes the “update” method which causes the SiteFileView to disappear.

In this stage of software structure description, the UML collaboration diagram is used to describe the messages passed from class to class for some key scenarios which are generated in the task analysis stage. The key scenarios include user input and modification of a file, input and modification of a scenario, invocation of ecosys script, customization of the library etc.

With the help of the UML class diagrams and interaction diagrams, classes, their relationship and dynamic behaviors among all classes are defined in Ecopro. This information forms the basis for the coding stage. After the system has been implemented, an initial system is ready for system test.

# Chapter 6

## Usability Testing

Regardless of the extent of analysis that has been done in designing an interface, experience has shown that some problems tend to appear when the system is tested with end users. Thus a usability test is necessary to assess a system's ability to meet user satisfaction and performance objectives. Two kinds of techniques are used for usability testing in Ecopro, thinking aloud and user survey.

### 6.1 Thinking aloud

The thinking aloud method involves having one test user at a time using the system for a given set of tasks while being asked to talk to the observer. By verbalizing his/her thoughts, the user enables an observer to determine not just what the user is doing with the interface, but also why he/she is doing so. This additional insight into a user's thought process can help pinpoint concrete interface elements that cause misunderstandings, and help in the redesign to better suit the needs of the end user.

The advantage of thinking aloud is the wealth of qualitative data that can be collected from a fairly small number of users.

The main disadvantage of the thinking aloud method is that it is not well suited for most types of performance measurement. This is because, the need to verbalize can slow users down, thus making any performance measurement less representative. As performance is not the most important concern in Ecopro, thinking aloud is a good choice for usability testing.

Before the usability testing begins, a set of tasks is selected to decide what the user tries do in the test. These tasks should reflect real-life tasks, which would be



executing once the system has been delivered to the end-user. In Ecorpo, the same set of tasks that were selected for the design evaluation are used for the user test. No modification to the tasks is required, as the test users are the same people who took part in the design evaluation. It is assumed that these test users therefore have all the background that would be needed for the test.

During the test, the user is asked to express his/her feelings and reactions while performing the various tasks. The feelings and reactions of the user relate to issues such as what he/she is trying to do, questions and concerns that arise as the user works his way through the various tasks, decisions that the user makes, things that the user views and reads and so on. Help is provided when necessary and notes are taken to record observations related to what the user does, what the user says, and a list of all difficulties that the user encounters.

The purpose of the test is to get all possible information that can help improve the system. Thus after the test, analysis is done based on the collected data and decisions are made to determine what changes need to be made to improve the system.

There are two steps that are involved in the analysis. The first step involves checking whether the system works as the designer expected. For example, whether the test users took the expected approaches, or were they working in a different way? This step is useful as it enables the designer to rethink the design to make it better suited to the needs of the users. The second step involves creating a list of all difficulties that the test users encountered and, for each of the difficulties, attempt to find out why the problem occurred, how important the problem is, and how it would to be fixed.

## **6.2 User survey**

Many aspects of usability can best be studied by simply questioning the users. This is especially true for issues relating to the users' subjective satisfaction, which are hard to measure objectively. User survey is a useful method which give us a summary of user's level of satisfaction with the system.

User survey involves asking users a set of questions and recording their answers. It has the advantage that it finds subjective user preferences and is easy to repeat.

In Ecopro, rating scales are used to ask users how well they liked various aspects of the system or how useful they found different features. The survey form contains a checklist, which covers various topics about the system. A number of questions are taken directly from [Shneiderman, 1993]. There are six sections all together:

- Overall user reactions are used to test the user's satisfaction with the system. This includes questions such as: Is the interface easy to use; does the user feel helpless using the system; are adequate functionalities and features provided with the system.
- Visual clarity is used to test whether information displayed on the screen is clear, well organized and unambiguous. For example, does the sequence of screens confuse the user; is important information highlighted; is it easy to go back to previous screen.
- Since ease of learning is the first usability goal in Ecorpo, a learning section is used to test whether the system meets this objective. For example, how many steps are needed to perform a task; is there any requirement for the user to remember names and commands.
- Terminology and system information is used to test whether helpful information and feedback is provided by the system. For example, are the error messages helpful; does the user feel confused and wonders what is going on; is there sufficient amount of information on how to use the system.
- System capabilities test is used to determine whether the system meets the needs and requirements of end users. For example, does the system provide appropriate functionality; is the system reliable; is the response time for most operations durable.
- Other usability issues include error prevention and correction, consistency, and flexibility. For example, is there any undo action for the user to reverse an error situation; are there standard procedures for carrying out similar or related operations; can the user choose to name and organize information which may need to be recalled at a later stage.

The experience with usability testing conducted in Ecopro is positive and encouraging. A lot of usability problems and issues such as the contents of help documents, the size and position of the different windows and the way that the user edits a library file are identified. For each usability problem the possible solutions are considered. Most of the solutions consist of redesigning part of the interface to improve the usability, whilst some others consisted of designing new functionality which would be part of the future work.

# Chapter 7

## Conclusion and future work

### 7.1 Conclusion

This thesis describes an implementation of a model-based development process using a set of design techniques throughout the system life cycle. The development of the system is based on the integration of a task model, a dialogue model and an implementation model.

The entire process begins with task analysis, which takes user requirements and produces a set of tasks to be supported by the design. After the initial task model is defined, the views for the objects in the task model are identified for supporting mechanisms to achieve the various tasks. Once object views are identified to support the required tasks, implementation details are specified for system programming. As an iterative methodology, users participate in the design, changes in requirements are handled, and designs are refined through the entire design process.

The main contributions of this thesis are summarized below.

- By defining a complete and accurate task model, a clear description is presented to describe the user's world. In Ecopro, two levels of task modeling is provided. The first level is a scenario-based model describing an enumerated list of tasks that must be supported. The scenario-based structure for the initial task model has the advantage of making the user feel comfortable to describe their tasks in a sequence rather than in terms of objects. The second level is an object based task model that integrates the level of support and system constraints into the model. By transferring the model from scenario-based to object-oriented

based, we take the advantage of object-oriented design. For example, the system becomes easy to extend as user needs evolve over time.

- By defining a continuous and explicit dialogue model, the description of the user's world is well matched to the computer system. The dialogue model in Ecopro covers all three levels of dialogue modeling. By doing this, consistency between different levels of interface modeling is ensured, and a common ground for communicating between the end user and the system is provided. The diagrams used in dialogue modeling serve as excellent input to code design and programming, because they are explicit, and they use a notation familiar to those working in object-oriented programming.
- By using different techniques for design and usability evaluation, user satisfaction is achieved. In Ecopro, iteration takes place between any two adjacent stages of the design, as well as through the entire life cycle of the system. Different evaluation techniques are used according to their specific advantages and disadvantages. User participation at the various stages ensures that the design changes are driven by actual user feedback.

## 7.2 Future Work

As mentioned above, a complete and accurate design is not possible in a single iteration. Iterations need to take place throughout the entire life cycle of the system. Such iterations help ensure completeness and accuracy, and tend to lead to a more fully defined system. In Ecopro, three areas are identified for improvement.

- More work could be done on usability testing to evaluate the success of the system. Empirical evaluation can be done by experts on usability engineering. Videotaping or data logging could be used to keep track of user behavior and the time required to accomplish a task.
- There are some more functionalities that could be added to the system. For instance, the system could be enhanced to visualize more output files. Also the system can be transformed to a web based application so that the client

may invoke ecosys even their computers are not powerful enough to do all the calculations. When each new feature is added, prototypes of the design should be created and usability should be tested.

- System usability is always important for user interface design. This is an issue which requires continuous refinement and enhancements. More features may be considered for usability enhancements in Ecopro. For example, more flexibility could be provided to user, and error prevention and correction ability could be improved to make the system more usable.

# Bibliography

- [1] [Diaper89] D Diaper (ed.): "Task Analysis for Human-Computer Interaction. Chichester" Ellis HOrwood, 1989.
- [2] Dix A., Finlay J., Abowd G. and Beale R. Human-Computer Interaction. Prentice Hall, London, 1993
- [3] [Foley, 1982] Foley, J.D. and van Dam, A. (1982). "Fundamentals of Interactive Computer Graphics", Addison-Wesley, MA.
- [4] [Forbrig, 1997] Forbrig, Peter; Schlungbaum, Egbert: "Model-based approaches to the development of interactive systems", In seond multidisciplinary workshop on cognitive modeling and user interface development, Freiburg 16 - 18.12.1997
- [5] [Elwert, 1995] Elwert T. and Schlungbaum E. "Modelling and Generation of Graphical User Interfaces in the TADEUS Approach", in Palanque P. and Bastide R. Design, Specification and Verification of Interactive Systems'95. Springer, Wien, 1995, 193-208.
- [6] [Goldbert, 1983] Goldberg, Adele 1983. Smalltalk-80: "The Interactive Programming Environment". Addison-Wesley Publishers, Menio Park, 1983.
- [7] [Green, 1986] Green, Mark, 1986. "A Survey of Three Dialogue Models. ACM Transactions on Graphics", Vol. 5, No. 3, July 1986, Page 244-275.
- [8] [Newman, 1995] Newman, William M 1995. "Interactive System Design". Addison-Wesley Publishers, 1995.
- [9] [Norman, 1986] Norman, D. & Draper, S.W. (eds.), 1986. User Centered System design: New perspectives on Human-Computer Interaction Hillsdale NJ: Lawrence Erlbaum Associates.

- [10] [Rosson, 1988] Rosson, M.B., Maass, S. and Kellogg, W.A. "The designer as user: Building requirements for design tools from design practice". Commun. of the ACM, 31, 1988, 1288-1298.
- [11] [Rosson, 1990] Rosson, M. B. and Alpert, S. R. "Cognitive consequences of object-oriented design". Human-Computer Interaction 5, (1990), 345-379.
- [12] [Shneiderman, 1987] Shneiderman, B. (1987). "Designing the User Interface", Addison-Wesley, MA.
- [13] [Shneiderman, 1998] Shneiderman, Ben, 1998. "Designing the user interface: Strategies for effective human-computer interaction". Addison-Wesley Publisher.
- [14] [Nielsen, 1993] Nielsen, Jakob, 1993. "Usability Engineering". AP Pressional.