Requirements Analysis for Information Web (IWEB): An Issue-based, Information Modeling and Management Environment for Software Engineering

Robert Coyne, Allen Dutoit, James Uzmack, Kevin OToole

EDRC 05-87-94

Requirements Analysis for Information Web (IWEB):[1]

An Issue-based, Information Modeling and Management Environment for Software Engineering

Prepared by the IWEB/n-dim Use Case team:
Robert Coyne, Allen Dutoit, Jim Uzmack and Kevin O'Toole

Contributions By:
Bernd Bruegge, Ira Monarch, David Rothenberger and the n-dim Group[2]

Contact:
dutoit@edrc.emu.edu

Abstract

The IWEB project is focused on creating an improved information management environment for team-based software engineering. This includes support for information structuring and communication during a project as well as maintenance and (re)use of project histories across projects. With this in mind, we are experimenting primarily within the context of the project based software engineering courses 15-413 (the basic course offered each Fall semester) and 15-499 (the advanced course offered each Spring) in the School of Computer Science at Carnegie Mellon University. These courses reflect much that is typical of the "state of the art" and outstanding issues in the current practice of software engineering. They are large (60 students, teaching assistants and instructors in F '93), require the delivery of a running prototype to a real client, and generally attempt to provide, as realistic as possible, software engineering experience to the participants.

This document summarizes the requirements for IWEB based on general case studies of engineering work conducted at the EDRC (and elsewhere) and integrates the experience of the authors as students, teaching assistants and instructors in the courses. We accumulated insights and studied information problems and communication breakdowns from three different perspectives: direct observation, student surveys, and quantitative analysis of project communication records and documents. IWEB incorporates ideas from previous issue-based support systems, such as gIBIS (graphical issue-based information system), and intends to build on top of a more general information modeling perspective by adapting a prototype of the w-dim (n-dimensional Information modeling) environment. To capture the functional requirements for IWEB we employ a development method structured around the viewpoint of prospective users of the system *(actors)* and that produces descriptions of all of the transactions these actors will engage in with the system as *use cases.* Though we are focusing on software engineering for the initial prototype of IWEB, we expect that it will have broader application within any team-based or collaborative work context.

---

1 This work has been supported in part by the Engineering Design Research Center, a NSF Engineering Research Center.

2 Robert Coyne, Doug Cunningham, Joe Davis, Allen Dutoit, Eric Gardner, Suresh Konda, Sean Levy, Ira Monarch. Robert Patrick. Yoram Reich. Eswaran Subrahmanian, Mark Thomas, James Uzmack, and Arthur Westerberg.

1 Introduction

The goal of the Information Web i IWEB) project is to develop an information modeling and management tool specifically designed to meet the needs of software engineers working in a group environment on large projects. This includes support for information structuring and communication during a project as well as maintenance and (re)use of project histories within and across projects.

As more people work on a project, the requirements placed on the communication infrastructure which supports their work grows rapidly. The more people working on a project, the more difficult it is to maintain a shared understanding of a project's goals, status, design, and implementation. Simply keeping track of responsibilities and deadlines can become very cumbersome. This problem is heightened in a part-time and /or distributed development environment where both the managers and the developers involved in a project often have many other projects or unrelated activities for which they are responsible.

Current environments available to support collaborative software design and implementation are insufficient for the task at hand. Typically, for each project developers cobble together an ad hoc combination of communication means such as e-mail, electronic bulletin boards, word processing and document preparation tools and a set of disjointed CASE tools which they use to support their work and communications. The teams must spend time learning how to use these tools as well as putting in extra effort to make the tools interact in even the most rudimentary of ways. The IWEB project proposes to develop an integrated information support environment which allows developers to communicate and track progress through a single software system. IWEB would also provide a single control point for interacting with other external software tools. It is hoped that by providing a single, integrated information repository developers working on a software project would be able to spend less time trying to learn tools and find information, and more time using the tools and manipulating information.

With these requirements in mind, we are experimenting primarily within the context of the project based software engineering courses in the School of Computer Science at Carnegie Mellon University (15-413, the basic course offered each Fall semester, and 15-499, the advanced course offered each Spring). These courses reflect much that is typical of the "state of the art" and outstanding issues in the current practice of software engineering. They are large (60 students, teaching assistants and instructors in Fall [4]93), require the delivery of a running prototype to a real client, and generally attempt to provide, as realistic as possible, software engineering experience to the participants.

This document summarizes "work-in-progress" toward establishing the requirements for IWEB based on general case studies of engineering and software design work conducted at the EDRC (and elsewhere) and integrates the experience of the authors as students, teaching assistants and instructors in the courses. We accumulated insights and studied information problems and communication breakdowns from three different perspectives: direct observation, student surveys, and quantitative analysis of project communication records and documents. IWEB incorporates ideas from previous issue-based support systems, such as gIBIS (graphical issue-based information system), and intends to build on top of a more general information modeling perspective by adapting a prototype of the n-dim (iwlimensionai information modeling) environment. To capture the functional requirements

for IWEB we employ a development method structured around the viewpoint of prospective users of the system *(actors)* and that produces descriptions of all of the transactions these actors will engage in with the system as *use cases.* Though we are focusing on software engineering for the initial prototype of IWEB, we expect that it will have broader application within other team-based or collaborative work contexts.

The rest of this document is structured as follows (see Figure 1.):

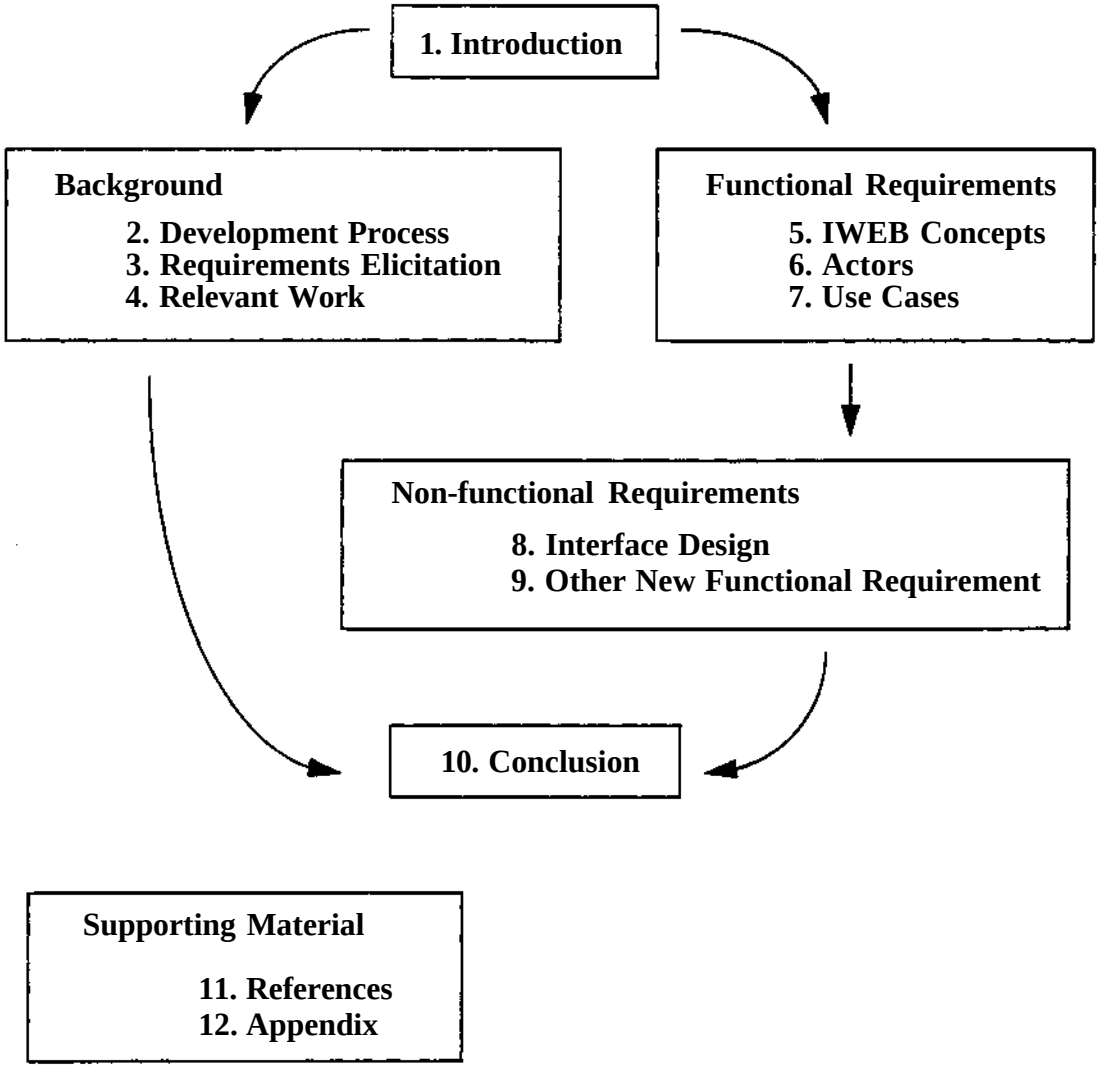


Figure 1. Document structure

Section 2 outlines our development process and the central activities for the requirements phase of development: requirements elicitation and requirements analysis in the form of defining important concepts, actors and use cases for IWEB.

Section 3 summarizes both the qualitative and quantitative aspects of the requirements elicitation process in which we have thus far engaged.

Section 4 reviews relevant work that IWEB is most directly motivated by and/or depends on. It describes common problems in information management for collaborative work in general and software design in particular; it then discusses current and envisioned information technologies that offer potential for improved support and gives an overview of gIBIS, an issue-based support system, and n-dim, a general information modeling approach and environment, two main contributors to the IWEB concept. The chapter concludes with a summary of how IWEB is envisioned to incorporate and integrate key ideas from both of these areas in order to combine dynamic issue-based communication, negotiation, structuring and notification on a project with the ability to interlink heterogeneous information both within and across projects in a comprehensive navigable web.

Section 5 lists terms and defines their meaning and interrelationships in describing the important concepts underlying IWEB, starting with the central terms and concepts from M-dim. We use these as a glossary of terms and a basic vocabulary for all the subsequent sections of the document.

Section 6 gives an overview of the Actors that we have modeled as prospective users of IWEB. Actors model everything external that communicates with the system, including human users and other systems. We show the relationship between IWEB actors and /z-dim and illustrate the mapping between IWEB actors and the typical roles for developers in our software engineering course projects.

Section 7 is a comprehensive description of all of the use cases of IWEB (that we have thus far modeled; we fully expect that there will be additions and modifications in subsequent development). A use case is a complete course of events in the system from the user's perspective. For each actor defined in Chapter 6, a set of use cases describe the associated functionality available to that actor. The use case descriptions, through multiple references, also make heavy use of the terms and concepts defined in Chapter 5. An understanding and development of IWEB requires a parallel and iterative reading and understanding of the Actors, Use Cases and Concepts.

Section 8 presents interface ideas and sketches for using IWEB and executing each of the use cases. Interface details are not included in the use case description in order to avoid premature commitment to a specific implementation and so that the flow of events in the use case is stated at a level of abstraction where it need not change with every modification in the interface. This approach supports a process where alternative interfaces can be explored and evaluated for each use case while the use case descriptions themselves stabilize. We have not progressed far yet but this chapter will eventually contain a section for the "visual presentation of each use case" and will become the basis for a first order user's manual.

In the rest of the document we give a very brief overview of non-functional requirements (Section 9) that we know must be met in a useful and usable support environment such as IWEB, an outline of future plans (Section 10) with respect to proposed experiments and evaluations of IWEB, references cited (Section 11), and finally an appendix (Section 12) with scenarios envisioning the use of IWEB within a software engineering project course.

2 Development Process

We are employing a structured, user-centered development method whose requirements analysis phase produces a functional specification of the proposed system as "use cases". Here, we discuss our motivation for doing so and describe the phases of the method.

2.1 Motivation

The scope and functionality of the IWEB system are conceived with broad capabilities. But our resources are limited. Thus, the development process will be extended in time, and it will involve various researchers and programmers, who, at any given time, will be able to concentrate on only a small portion of the system. It is therefore crucial that these distributed efforts be compatible with each other and allow future extensions and modifications without compromising the stability of the overall system. We therefore decided to start work on IWEB with a broadly conceived specification of its functionalities, which can then be broken down into subtasks and handled by small development teams, each of which understands its role in the overall effort and the interaction of the piece of software on which it concentrates with the other (existing or envisioned) pieces. These specifications provide also a means of communication with our clients and sponsors and form an important portion of the system documentation.

To assure the desired functionality through the later development phases, it is desirable to continue with this effort into the following development stages. We intend to do this to the extent allowed by the resources available to us. That is, we intend to employ software engineering techniques in the development of IWEB whenever possible. The methods employed by us are introduced in the following section.

2.2 Development Phases and Methods

Given the scope and the originality of the problem IWEB is attempting to address, we use a variety of methods to collect and organize information to specify the problem, and to synthesis and refine a solution. The development process leading to a complete definition of IWEB includes the following phases:

- *requirements elicitation* during which we collect and analyze information about the current state of the problem domain and outline broad classes of functionality; the main outcome of the requirements elicitation phase is a problem statement summarizing the issues discovered during this phase;
- *requirements analysis* during which we formalize and detail the classes of functionality found during the previous phase; the main outcome of this phase is a requirements analysis document, describing completely the system being built;
- *early user evaluation* during which we prototype subsets of the functionality defined previously and evaluate them; the main outcome of this phase is a number of mock-ups and prototypes of IWEB, leading to refinements in the problem statement and the requirements analysis document.

Although these three phases are initiated sequentially, they continue and coexist for the duration of the project. The development process we are following is iterative in nature, i.e.

results from later phases may lead to refinements and correction of documents produced in earlier phases.

Requirements elicitation includes studying the problem domain and outlining new classes of needed functionality. We approached this problem from a specific point of view: although we are interested in communication support and information modeling in software engineering in general, we initially studied the narrower problem of providing an infrastructure for the software engineering course we teach to senior undergraduates in computer science (15-413). We argue that we cannot develop a general solution for software engineering before we acquire the understanding of a specific instance of the problem. To this end, we collected and studied data from the Fall '93 software engineering course. We studied communication breakdowns from three perspectives:

1. We observed the students' work progress and problems during the semester from a manager's perspective. Observations of the instructors and the teaching assistants were collected in a series of papers [1][2][9]. These observations are summarized in Section 3.2.
2. At the end of the semester, students were surveyed (specifically about information management, communication and process issues). A summary of the survey is provided in Section 3.3.
3. Metrics on bboard communication and documents were evaluated (Section 3.4).

At the same time, the instructors and a subset of the teaching assistants of the Fall '93 course met on a weekly basis to brainstorm on the infrastructure requirements. A scenario-based method was used: members of the requirements group wrote scenarios for common tasks performed by the students during the course. Scenarios were written to describe both tasks as they were performed during the Fall '93 course ("the old way") and the same tasks as they would be performed using IWEB ("the newer way"). These scenarios envision IWEB as first order simulations of its "use" where we project ourselves as "end users" (based on our knowledge of them and the situations they find themselves in) in order to "walk-through" potential interactions with the system. This gives us a type of feedback on what is needed in the system. We can then proceed to "higher order" simulations in which use cases, mock-ups and actual prototypes (like using the existing Ai-dim) are all tools that can be used in some appropriate order. Examples of scenarios are provided in Section 3.5 and in an Appendix in Section 12.

Requirement analysis includes the formalization and the detailing of the classes of functionality discovered during requirements elicitation. We approached this phase using the OOSE[15] method. This approach asserts that the most essential property of a system is a stable structure (architecture) during its lifetime. It asserts furthermore that the best way to insure this is to model the system functionality on the basis of the organization in which it will be used and its users, and to expect change - by viewing new system development as only a special case of further system evolution, development, and maintenance.

OOSE belongs to a growing number of methods that consider behavior modeling, or a high-level description of the system's functionality from the point of view of its users, as central to driving system development. In OOSE, this behavioral description is based on *actors* and *use cases.* These concepts are a simple aid in defining what exists outside the system - the actors - and what should be performed by the system - the use cases. In

developing a system in this way, the whoie system architecture will be controlled from what the users wish to do with the system. The use of object-oriented methods to model systems ensures also that there will be traceability through all models from requirements to code and maintenance. In combination with the u^ e case approach, this enables us to modify the system through changing and newly emerging 1 ..uirements. Use cases for IWEB are described in Section 7.

Actors model prospective users - everything external that is to communicate with the system, including other systems. Actors represent a certain role, or class description of behavior, rather than an actual person who uses the system. They are the main tool for finding use cases, and together actors and use cases define the complete functionality of the system. Each use case is a complete course of events in the system from the user's perspective. Use cases are the central thread running through the whole of OOSE. Actors for IWEB are described in Section 6.

In addition to developing descriptions of actors and use cases for IWEB, we collected the main terms and concepts needed for understanding those descriptions in a glossary (see Section 5). The terms and concepts will then be used as a starting point for identifying objects and their associations.

The use cases and the actors form the *requirements model,* which aims at delimiting the system and defining what functionality the system should offer in terms of actors and use cases. It serves as a means of communication between the developers and orderer of the system. It thus forms the developers view of what the customer wants and should be readable for non-OOSE practitioners.

The terms and concepts description, later formalized and documented in an object catalog form the *analysis model* This model structures the system independently of the actual implementation environment. It is derived from the requirements model and forms the basis of the system architecture.

Early user evaluation is necessary given that the problem we are solving is relatively ill dchned. We plan to develop mock-ups and prototypes of IWEB for the purpose of evaluation by ourselves and extern users. The results of the evaluations (not covered in this document) will be used to refine the problem statement and the requirements analysis document.

3 Requirements Elicitation

This section documents the IWEB requirements elicitation effort, during which we collected and analyzed information about the current state of the problem domain, and defined broad classes of functionality for IWEB. We chose the software engineering classes taught to senior undergraduates at Carnegie Mellon University as the focus of our case study.

This section is organized as follows: Section 3.1 provides background on the organization of the Fall '93 software engineering class; Section 3.2 summarizes our observations; Section 3.3 summarizes feedback from a student survey that we conducted at the end of the Fall '93 class; Section 3.4 is a qualitative analysis of the bboard traffic generated during that class; Section 3.5 discusses scenarios that we constructed to describe everyday tasks accomplished by the students during the class and to envision how they would be accomplished with IWEB; and finally, Section 3.6 presents a problem statement for IWEB, that summarizes the issues we describe in Sections 3.1 through 3.5.

3.1 15-413 Background

The SE courses in the School of Computer Science have been taught following the same format since Fall '89. In an attempt to provide the students with a realistic software engineering experience, the system the students (~50 in Fall, ~20 in Spring) are asked to build must solve an actual problem specified by a real client. At the end of the semester, the students present a prototype system to the client during a client acceptance test which has a significant impact on the students' grade.

In parallel to the development of the system, students are trained to use a specific software engineering methodology (e.g. Fall '92 & Spring '93: OMT[26]; Fall '93 & Spring '94: OOSE[15]), and are taught basic organizational skills (how to conduct a meeting, write meeting agendas and minutes, assign roles and coordinate with other teams through liaisons). A bboard per team is provided for the purpose of inter- and intra-team communication, and method and documentation support tools are provided. In addition, some project wide bboards for announcements by management and to facilitate project-wide discussion are also provided.

The development process followed by the course starts with the instructors and teaching assistants (hereafter referred to as "project management") pre-engineering a set of requirements based on the client's input, and then generating an initial system decomposition. The students are then responsible for detailing the requirements, designing, implementing and testing the system. The project management initially provides the students with a structured process including phases, deadlines and required deliverables. However, after the course starts, there is little intervention from the project management as to the direction of the design or the evolution of the process. Students are even encouraged to take initiatives with respect to the artifact and the process perspectives.

Figure 1 is an illustration of a typical organization structure for the software engineering course. The course students are initially divided into teams, usually one team per subsystem. In addition to the subsystem teams, a user interface team and a documentation team are formed. Each team also has additional project support tasks, such as selecting revision control tools, user interface development environment, databases, etc. The

integration of the separate subsystems is usually managed by an integration team, or by one of the subsystem teams which has strong dependencies to other subsystems.

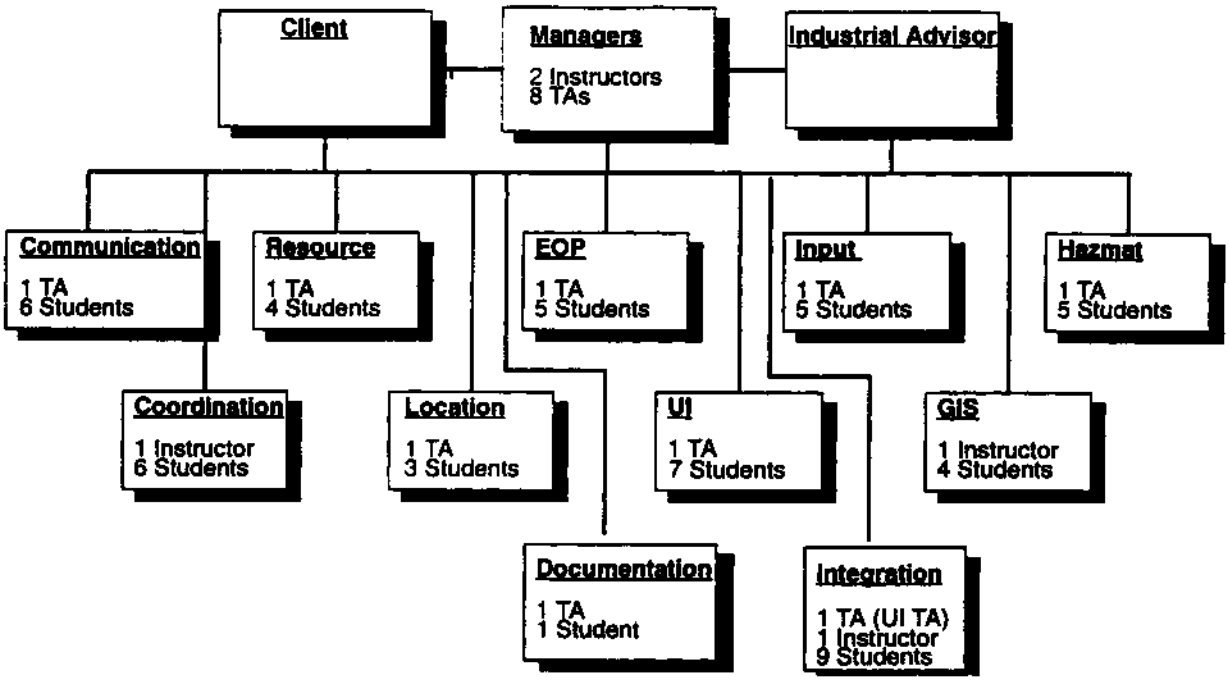


Figure 1 Organizational Structure for 15-413 Fall '93

Team members are also assigned roles:

- the *team leader* or *primary facilitator* is assigned the task to organize meetings (write agendas) and monitor the work of the team;
- the *minute taker* writes minutes for every scheduled meeting;
- the *time keeper* ensures that all the issues of the agenda were raised during the meeting;
- the *UI liaison* represents the team at the UI meetings;
- the *liaison* represents the team at the integration meetings.

The communication within and across teams occurs in a variety of modes. Each team is required to meet on a weekly basis with an assigned TA[1] to report on their status and make plans for the following week. Each team is also allocated a bboard for intra- and inter-team communication. Table 1 lists the typical content, the size of the targeted audience and the latency of reply for each communication mode.

Table 1 Communication Modes

Mode	Context	Audience Size	Latency of Reply
Zephyr[a]	Low level detailed issue requiring **immediate attention.**	1 -50	seconds

1 **The integration team was managed by the user interface TA and an instructor**

Table 1 Communication Modes

Mode	Context	Audience Size	Latency of Reply
Personal e-mail	Low level detailed issue requiring short term reply; questions hidden from the general public	1-10	minutes - days
Bboard (intra-team)	Team agendas and minutes; higher level issues; issues relevant to entire team.	3-7	1-2 days
Bboard (inter-team)	Class announcements; inter-team issues;	7-50	1 day - 1 week
Weekly meeting	Team meetings; Integration meetings	3-7	immediate
Ad hoc meeting	Specific hard-to-solve inter-team issue	3-20	immediate

a. Zephyr is a "notice transport and delivery system" developed at MIT. A Notice Transport and Delivery system is a method of getting small quantities of time sensitive information efficiently from one client to another.

The inter-team communication was facilitated by the project wide bboards (such as discuss for project level issues), and by teams cross posting messages on one another's bboards. However, there are problems with both methods - as we observed and report in the next section - ineffectively supporting inter-team communication.

3.2 Direct Observations

The instructors' and TAs' observations of the course were captured in a series of papers [1][2][9]. Although these papers focus mostly on the educational aspects of the course, the authors report on a number of software engineering issues and related information structuring and communication issues which were raised during the course.

The instructors observed that the static nature of the communication structure[1] makes the organization slow to respond to unexpected problems. This proves to be a disadvantage when it becomes clear that the system decomposition should change (sometimes drastically) after the start of the course.

The instructors also observed that although communication mechanisms are in place from the beginning of the class, students usually need two to three weeks to learn to use those mechanisms effectively for intra-team communication. For inter-team communication, the ramp-up time is even longer (4-6 weeks). The longer learning curve for inter-team communication is due to the problem of learning the mapping between students, roles and teams. During that period, messages are frequently misdirected. This is compounded by the fact that students go through other learning curves as week during the same time (new tools, new methodology, project support functions, etc.).

1 **For example: difficulty in creating new bboards, weekly nature of meetings, etc.**

Instructors found that the models and documentation submitted by the students during the course of the class are not effective in mediating and evaluating the status of the project: they become obsolete fairly quickly. This is potentially due to the lack of integration between the artifact modeling method and tool, and the larger project information and communication structures. This is also potentially due to the poor performance of the modeling tool and the students' lack of experience in using it. For the same reason, they observed that inter-team coordination through models and documents is not a feasible idea. The most reliable measure of progress is the weekly TA meeting during which each TA summarized the progress of their team, but an hour meeting cannot substitute for the volume of continuous communication that is typically required on a large project.

Finally, they observed that the integration of the system (from a documentation, design and code point of view) started too late. Although an integration team was formed at mid semester, the time associated with its setup was such that it was not able to detect and resolve project wide issues. The user interface team, which was formed at the beginning of the project and composed of one representative of each team, had more success in maintaining some consistency during the project.

3.3 Student Observations

Towards the end of the Fall '93 course, a written survey of the students was conducted. The survey was structured as a homework on the information modeling lecture. Students were asked to critique the communication infrastructure, to express their opinion on whether or not they thought a project wide issue base would have made their task easier, and to raise any other issues they thought were important in improving the course.

Most of the students answered that inter-team communication was the major bottleneck of their process. They attributed the problems encountered during the integration phase to a lack of communication between teams during the requirements phase, problems of terminology, and important decisions made arbitrarily or not communicated in a timely fashion.

In general, students thought that a bboard communication structure was adequate for inter-team communication, but also reported that it could be greatly improved. Students reported that bboards (and e-mail to some extent) helped overcome the distributed aspect of the process (i.e. during some period of development, lots of students of the same team were unable to meet due to different course and exam schedules, different working locations, etc.). Reported problems with bboards included difficulty in targeting a message (which bboard to send it to; who should reply to a request), volume of information (e.g. one student suggested to require messages to be shorter than three lines), difficulty in maintaining the thread of a discussion, and difficulty of searching for related posts by only using captions (i.e. inaccessible history).

In general, students mentioned the difficulty in targeting a message as a separate issue, related to the evolving nature and the scale of the organization. Some important roles, such as team leader and integration liaison were rotated on a two week basis. Some students had several e-mail addresses and did not configure their e-mail forwarding mechanisms correctly. A simple listing of the personnel of the project including current e-mail addresses, phone numbers, roles and team assignment was thought to be a significant improvement. Although

the course had such a listing, it was not maintained as the organization evolved to adapt to the changing system decomposition.

Students found documentation and the exchange of object models an ineffective communication medium for inter-team coordination. The main reason listed was the insufficient turnaround time of documents and models (i.e. by the time a document or a model was made available to the class, it was obsolete). The difficulty in searching and keeping a history of the evolution of the document/model was also listed as a contributing factor. In general, documentation and models were viewed as a necessary evil to be produced for the consumption of the client, and not as an inter-team coordination mechanism.

Students found weekly meetings extremely useful for exchanging design and status information. Their periodical nature significantly reduced the overhead associated with organizing such a meeting. Moreover, students judged weekly meetings as the only acceptable medium for negotiation and decision making. Although bboards were adequate for exchanging large volumes of information, they rarely witnessed a successful negotiation process. On the other hand, ad hoc meetings called to raise and solve specific issues were judged a failure, primarily due to their large overhead (for example: setting up an adequate time and location for such a meeting usually took several days), and their large attendance (that is more people than necessary were involved, thus slowing down the decision process). In fact, these ad hoc groups did not have the authority or influence to make and implement unequivocal solutions for the teams they represented.

Finally, students found that the most effective inter-team communication mechanism was the role of liaison. A fixed representative of the team attending integration or user interface meetings proved to be the most reliable and fastest communication medium. The role of liaison became valuable when raising and solving inter-team issues involving a limited number of teams. In contrast, the user interface and the integration team were judged appropriate for solving project wide issues, but ineffective in solving issues involving only two or three teams. The negative side of the role of liaison was that it created a small number of specialists who maintained most of the history by memory or in other personal forms, thus rendering the organization very sensitive to changes in personnel (e.g. during the Fall '93 course, a user interface liaison dropped the class at midterm; the user interface team and the other team spent about two weeks to find and train somebody else to fill the role, generating a significant amount of miscommunication and misunderstanding).

3.4 Quantitative Analysis

3.4.1 Global Traffic

During the course of the semester, a total of 2502 messages were posted on the bboards by students, TAs and instructors. We observed that most bboards followed a weekly pattern: bboard activity increased immediately before and after the weekly meeting of the associated team, and decreased at the beginning of the week (Sunday night and Monday morning). This observation motivated us to sample our metrics on a weekly basis.

Figure 2 shows the communication activity on the bboards on a weekly basis. The bottom curve (black) shows the total number of messages sent per week. The top curve (light gray) displays the total number of unique noun phrases[1] used in those messages for each

week. The middle curve (dark gray) displays the number of new noun phrases used in those messages for each week (i.e. noun phrases which were not used in any previous week). The horizontal axis represents time in weeks where the first week is numbered 0. The vertical axis represents the number of messages for the black curve, and the number of terms divided by 10 for the gray curves.

The number of unique noun phrases were counted using a set of NLP (natural language processing) tools ([18] and [23]). Messages were stripped of their headers, signatures, special characters and other encoded enclosures before being submitted to the NLP tools.

The number of messages posted peaked during requirements (week #3) and the start of final integration (week #11). The drop at weeks #6 and #7 indicates a drop in the activity of the class waiting for the integration team to release the first merged database of models (delay occurred due to problems with the modeling tool and with the set up time of the integration team).

An examination of the gray curves lead to similar observations. However, the peak during integration is more pronounced than the requirement peaks, indicating a richer vocabulary, and possibly a broader range of issues being discussed.

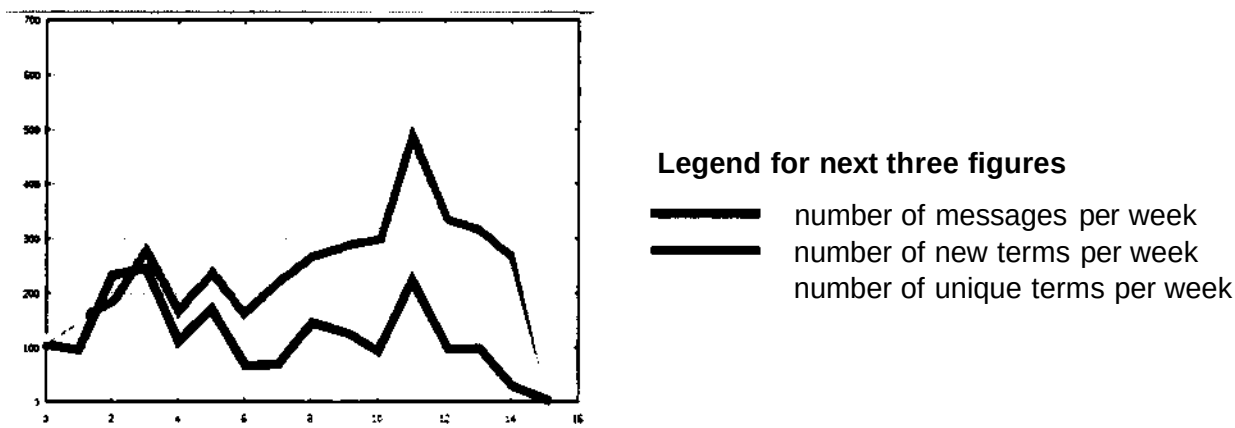


Figure 2 Global Bboard Traffic

3.4.2 Intra-team Communication

Figure 3 displays the same curves as Figure 2 for the messages posted on single team bboards. The left graph represents the number of messages, the total number of unique terms and the number of new terms per week for the user interface team. The right graph represents the same information for the Hazmat team (whose assignment was to build the database and query system for hazardous materials).

The user intert^e team shows peaks in the number of messages similar to the global metric. However, the gray curves peak during the requirements phase, thus indicating a richer vocabulary during problem definition. This is due to the fact that the user interface team was composed of representatives of all the teams, and had a better shared understanding of the overall system. This pattern was also observed for other teams who performed an integration function, and were aware of it during requirements (e.g. the subsystem team charged with

1 We interpret unique noun phrases to represent concepts that are being formulated and exchanged.

code integration, not displayed here, aiso demonstrated a richer vocabulary during requirements). In contrast, the Hazmat team displayed on the right, who did not have much understanding of the rest of the system before integration, shows a profile similar to the global profile (lesser volume during requirements and peaks in later phases near integration). This profile is representative of teams which were not assigned an integration function.

These observations were correlated with the fact that the teams displaying a better understanding of the overall system performed better according to the standards set in the course (less deadlines missed, functionally rich subsystem demonstrated, integration with the whole system achieved early).

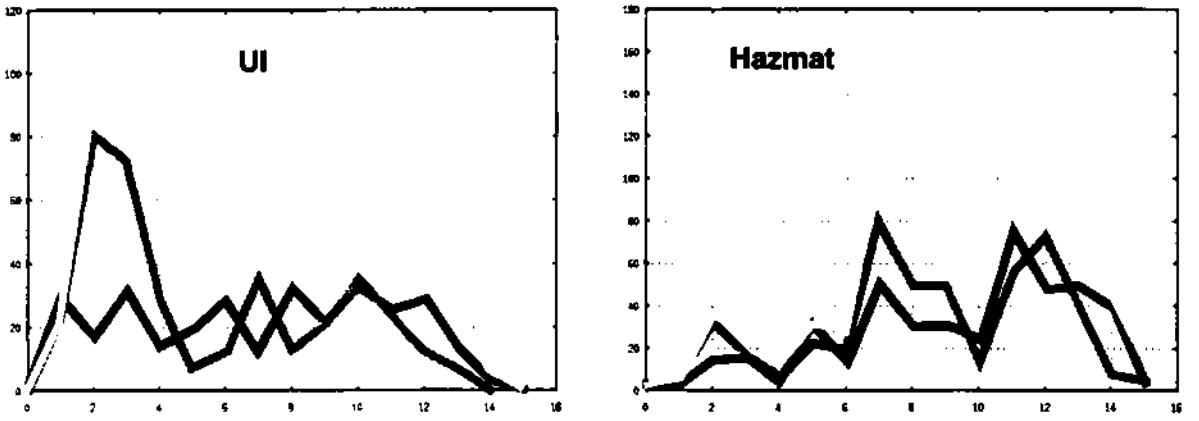


Figure 3 Bboard Traffic for UI and Hazmat teams

The gray curves in Figure 4 displays the ratio of new terms to the total number of terms per week[1] for each of the user interface and Hazmat teams. The black curve displays the same ratio for the overall number of new terms and unique terms. In both cases, the ratio is generally decreasing, indicating fewer new terms are introduced as the project progresses. However, the Hazmat team displays higher rates of new terms than the user interface team. The fraction of new terms also increases during the integration phase, indicating a crisis (there was a fairly substantial debate about whether the public interface of the Hazmat subsystem should be available to other subsystems as a C++ object or only to the user

1 The number unique terms for week X and team Y are the number of unique noun phrases that could be found in the messages posted on bboard Y during week X. The number of new terms for week X and team Y are the number of unique terms which nave not yet been used in bboard Y since the beginning of the project. The gray curve is the fraction of new terms vs. unique terms.

interface; the user interface team was not part of the debate since it already resolved interface issues previously through its UI liaison).

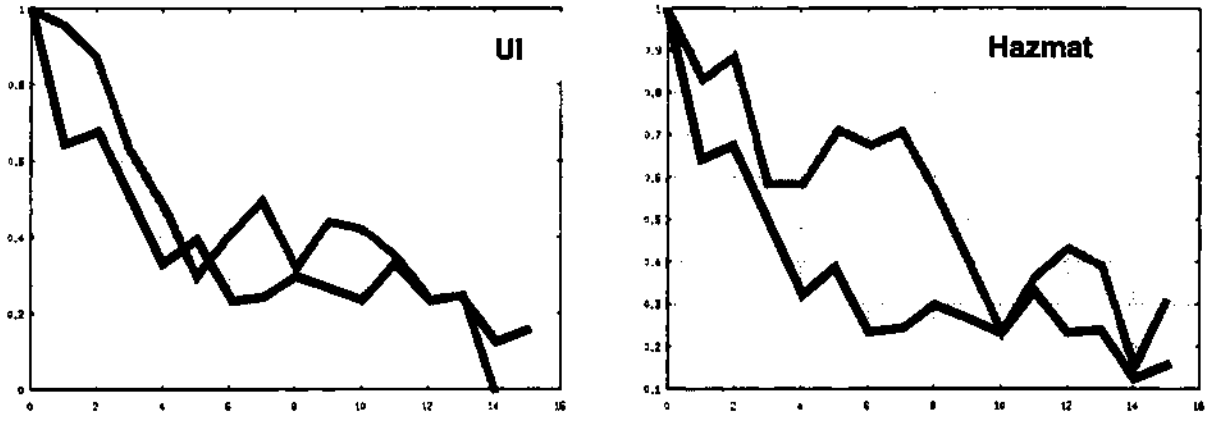


Figure 4 Ratio of new terms to total terms for UI and Hazmat teams

3.4.3 Inter-team Communication

The observations in the previous sections are further confirmed by the examination of the inter-team communication for the user interface and Hazmat team. In Figure 5, the horizontal axis (x) represents time in weeks. The y axis represents the teams of the project (the front team being project management). In the left graph, the vertical axis (z) represents the number of messages sent from the user interface team to a given team, added with the number of messages from the given team to the user interface team. The right graph displays the same information for the Hazmat team.

The user interface team displays a fairly constant interaction with most of the teams, with moderate peaks during integration. In the case of Hazmat, the inter-team interaction is weaker, with more significant peaks during integration. Hazmat also had a stronger interaction with management (Hazmat relied more heavily on their TA for inter-team coordination and did not attempt very often to directly synchronize with other teams).

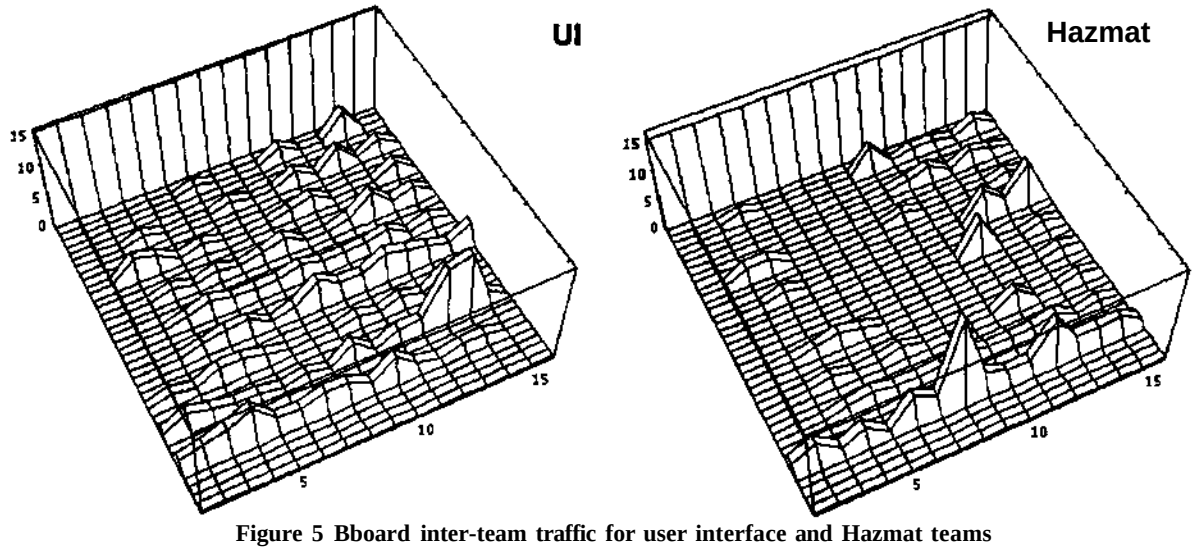


Figure 5 Bboard inter-team traffic for user interface and Hazmat teams

3.5 Scenarios

Ideally, in order to design better information management support for team-based SE, we could engage in a type of extended cooperative design process with end users. In this process we would envision future work situations by simulation of possible computer support using mock-ups and prototypes. Thus, end users would get a hands-on experience and give valuable feedback.

Scenarios are an intermediate step towards a cooperative design with end users. A scenario (in the context of this document) is a sequence of events which describes how a specific task is presently accomplished, and a sequence of events describing how the same task might be accomplished using IWEB. By using scenarios, we identify common tasks and analyze how those tasks could be improved with IWEB. This scenario based approach can be thought of a first order simulation in which we project ourselves in the role of the user. Using this approach, we hope to focus on the most important and frequent information exchange tasks accomplished by participants in a software engineering project (here the specific context is an instance of one of our project courses) and to whatever extend possible develop and refine the requirements for IWEB to support them before we conduct any experiments with real users.

Below is an example of scenario describing how an ad hoc meeting for a group would be organized. The current sequence of events required to perform this task is labeled O (i.e. Old way). The envisioned sequence of events required to perform the same task with IWEB is labeled N (i.e. New way). Scenarios are labeled with times to illustrate the chronology and duration of the task. Other scenarios are provided in an Section 12.

10:25-10:30 am

O« A subset of the group realize that they need to have a meeting to discuss some aspect of the system. They agree to select a time and try to get the rest of the group to attend. The meeting time and place is posted on the group's bboard.

N. After lecture, he, along with a couple of other team members decide to call for a special team meeting (other than their normally scheduled weekly meeting. He volunteers to send an IWEB message to the other team members and they agree that everyone will check the IWEB a couple of times through the day to see the status of the meeting. They agree that by 4:00pm the meeting will have either been confirmed or postponed.

3:00-3:10 pm

O. Read bboard, check meeting messages.

N. Checks IWEB for message(s) about the meeting. Meeting is confirmed for 7:30pm.

4:30-5:00 pm

O. Reads bboards, posts some internal messages. If needed, reads another group's bboard and the Discuss board to carry on inter-group arguments. Is annoyed by duplicate posts, inability to find old messages, and textual drawings.

0. Poses a message saying that an Objectory Model exists relating to a message that was posted. Gives a description of how to find the model. Becomes concerned no one will ever look at the Model because Objectory is so cumbersome to start, further if they do start it, they will have to navigate to the appropriate use case/model to check the information.

N. Reads the IWEB issue-base; posts some sub-issues and responses; notices that a new task assignment has been proposed for her team.

N. Brings up Objectory from within IWEB and does some modeling, then annotates a new version of a specific Objectory model and links it to a discussion node in the issue base.

7:30-8:30 pm, Attends the special team meeting

0. At the meeting takes action items to be discussed with other teams. Meeting minutes are taken on a Duo, and then later posted to the team bboard. Further, those minutes are later copied by the Tech Writer into a Frame document for hypertext access that is never used because Frame is cumbersome to start and difficult to learn to use effectively.

N« At the meeting the team decides to "open a dialogue" within IWEB with another team regarding a particular object model.

3.6 Problem Statement

The Software Engineering class 15-413 at Carnegie Mellon has been in existence for some years. The class is unique in its approach to teaching software engineering to undergraduate CS students. Where many classes just talk about software engineering and perhaps have their students work on "toy" systems, 15-413 students work on a large scale project. The project always involves a client with a complex, real problem who is in search of a computer based solution. The class often has upwards of 50 people in it, which leads to a total project membership of approximately 60 people once management (professors and TAs) is added to the group.

During the Fall '92 and Fall '93 classes, the class has worked on the FRIEND project. FRIEND is a system for distributed accident management involving many levels of government, multiple cutting edge technologies, and considerable original software development. The project work has led both management and the students to encounter many of the classic problems found in software engineering. The academic setting in which this project is being undertaken heightens the effects of these problems, however, and in many cases hinders the both the education of the students and the progress of FRIEND system. We hope to develop a system to greatly reduce the effects of these problems through the use of informati* : modeling.

The first problem to be encountered by the students, and one which endures the length of the project, is simple intra-group communication. The 50 students in the class are broken into groups along, each group being responsible for a functional aspect of the system (UI, Communication, Database, etc.). Each group has approximately 5-9 people. The problem first arises when the groups are attempting to find a convenient meeting time. Since there is

no readily available central repository for this type of information, the students must post e-mail messages to electronic bulletin boards in an attempt to find a time to meet. This often leads to people not making any real project progress for the first two weeks simply because they cannot find a meeting time to overcome other organizational issues.

This intra-group problem continues and grows more complicated as people begin to work on the project and need to acquire some type of shared understanding of the system as it evolves around them. While modelling is supposed to overcome this problem (i.e. someone should be able to look at a model and more quickly understand the system), often models and textual descriptions quickly become too voluminous to be scanned each day to find what other people have changed. A facility for tracking and alerting a developer to changes relevant to his/her area of responsibility would be invaluable to increasing shared understanding within a group.

A larger problem that becomes increasing visible when the project groups begin to integrate their designs and especially their code, is the problem of inter-group communication. Most systems under development are simply too complicated for a single person to understand. This leads to decomposition and modularization of project components and work assignments. Conversely, every developer also needs to have some level of a global view of the system. This view must be more precise in its understanding the closer the other project components are to the developer's area of responsibility. There is currently no readily accessible means for a developer to at once obtain a global view of the project and also obtain a precise view of his/her area of responsibility.

A further problem encountered in inter-group communication is the discovery, analysis, negotiation, resolution, and dissemination of problems and fixes in the design of the system. This is especially problematic in places where multiple groups are responsible for similar activities or for communicating with one another. A facility for tracking arguments and negotiation would provide a more moderated, trackable way of deciding how to resolve problem within the project.

A problem that is especially plaguing in the academic environment is a lack of good model usage and referral. This has two motivators. First the students are learning modeling and modeling techniques, so one would expect them to be a little slow to grasp the need and benefits of modeling. The long run problem, however, is convenient model access. Many of the tools that are used are cumbersome to start and difficult to become familiar with. Finally, the tools do not organize the material very well and it is difficult to find the state of a document from within most tools. It would help greatly if the CASE tools and other software in the project were nearly inspection usable.

Another problem is the explosion of client interaction. The client that we work with is often accessible to the entire class via email and interviews. The client becomes deluged with questions (some of which are ill formed) and in some cases returns inconsistent answer simply because he thinks he's been asked a different question. A tool for monitoring/ cataloging client interaction would be greatly helpful in guiding design and requirements decisions which are often made late at night by tired students too overwhelmed to go hunting through old bulletin board messages looking for snippets of relevant information.

The previously mentioned problems with communication and information management leave a lasting impression on the project. This lasting impression is visible later in the

project's life cycle as developers (especially new developers) try to understand the rationale of the design and implementation of the system. Often a developer will be able to look at a design and say "This is bad." Unfortunately, people don't just build things poorly due to laziness (usually). There is usually a reason that a particular bad design transformed itself into existence. Without the ability to look at the relevant factors which influenced a design, the developers later in the project are forced to re-engineer the requirements which led to the existing design. Most likely they will miss some of these requirements (especially the subtle requirements generally discovered late a night by a single person). This lack of ready information may lead to an even worse redesign than the one currently implemented. This lack of long term "project memory" can greatly increase code entropy, code drift, and lower productivity of the project members.

A final communication problem is that there are multiple forms of communication being used in the project. Project members communicate through:

- Personal Electronic Mail
- Electronic Bulletin Boards
- Zephyr
- Objectory and project documents
- Meetings using whiteboards, chalk boards, overheads, and paper
- UNIX Talk Sessions
- Telephone conversations
- Group wide project meetings and presentations
- Model and code Comments

This multi-modal communication makes it difficult if not impossible for a developer to go back through the various project documentation in search of a design decision. It is very likely the design decision or the rationale which drove the decision do not even exist in any formal project documentation.

A tool which combined the above communication channels into a single, tracked environment would be extremely useful to project developers in their attempts to communicate and then look back on what they have done and why.

4 Relevant Work

In this section we briefly review relevant work that most directly motivated IWEB.[1] In Section 4.1, we describe a set of common observations, insights and understandings of problems in information management and integration that have emerged from studies of a variety of group work settings in engineering and software design. In Section 4.2, we further articulate software engineering as a collaborative, social (as well as technical) design process and summarize requirements for an improved information support infrastructure. Section 4.3 discusses gIBIS, a clear predecessor of the issue-based approach that is central to IWEB and which has been shown to be an effective means of structuring the communication of and access to ideas and decisions in software development contexts. In Section 4.4 we give an overview of n-dim, a general information modeling approach and prototype support environment that IWEB will be embedded in and will adapt for software engineering project work. Finally, Section 4.5 summarizes how IWEB is planned to incorporate the various requirements discussed for collaborative work and software design to provide an improved information support environment for team-based software engineering projects. It will integrate and extend proven information management ideas and technologies such as issue-base discussion support with the comprehensive information modeling and project history capabilities envisioned in *n-dim.*

4.1 Case Studies of Engineering and Software Design Work

The focus of the requirements analysis effort documented in this report, toward an improved support enviroment for communication and integration of information in team-based software design, reflects our belief that information modeling and management is a key area impacting productivity and quality in software engineering. This belief grows out of experience, observations of recurrent problems and subsequent case studies which were initially targeted at engineering design, then broadened to include software design (software engineering) and ultimately encompass and are supplemented by similar insights emerging from collaborative, team-based work settings in general.

At the Engineering Design Research Center at Carnegie Mellon University we have observed many different design projects, for example, a new control system for power generation with Westinghouse, a multi-university electrical connector design with three other Engineering Research Centers [37], and a Database design with Alcoa. In all of these we noticed the importance of managing the information that the different groups created and the difficulty all the project participants had in sharing it and establishing a shared understanding of it. Subsequent projects, one with ABB Power Transformer Design [12], one with a Union Switch and Signal in software engineering [10] and a general study of Japanese compaanies (Sony, Mitsubishi Kaisei, Toyo Engineering) [32] have provided us with more empirical evidence of the importance of handling information within a design activity. There are also several other projects reported in the literature that report similar conclusions including studies of computer-supported cooperative work at the University of Arhus [14], and the European Esprit project case studies [5] [28]. Another is a joint Bellcore-Rutgers University study of the design of communication systems [33].

1 **The space of related work and literature on information modeling and management approaches and tools is vast and we make no claim to covering that in a comprenensive way here.**

Common observations and findings taken collectively from these experience reports and empirical studies is summarized in the following points. Many of the problems/issues detected concern information management. The significance of this is amplified by the fact that as much as 50%-60% of daily engineering work consists of information gathering, communication and documentation. The kinds of recurrent problems experienced in a wide variety of contexts include:

- Information loss
- Information access problems.
- Information coordination problems.

These are primarily caused by or aggravated by the fact that today's large, monolithic, vertical information systems do not support daily work well. In response to this, companies often glue multiple vertical systems together by expensive, ad hoc and cumbersome procedures. These problems result in:

- costly overhead in information management
- breakdowns in communication and understanding

The implications of these problems taken collectively from the experience reports snd empirical studies mentioned and others indicate an urgent need for improved support for project-wide and organization-wide information integration to handle heterogeniety of:

1. expertise:for example, in engineering design electrical, mechanical, and control engineers in software engineering managers, analysts, human factors experts, designers, programmers, etc.
2. perspectivexustomer, marketing, design, manufacturing
3. source: documents, sketches, drawings, messges, notes, minutes, and so on
4. scope: individual vs. group information (the need for its seamless integration)
5. modeling: product (e.g., a software artifact), process, and organization
6. tools: a polyglot of access and interfaces (the need for more uniform and seamless interconnection)
7. history: design process and product, both within and across projects

4.2 Information Management Requirements for Software Engineering

Software engineering has been recognized as a collaborative, social, work (design) process requiring the adaptation of methods and tools to the needs of cooperation and incremental teamwork. Important aspects of this collaborative view of software production are the creation of a "theory" or shared understanding of the software artifact under development [22], and the dynamic coordination of the development process [13]. Therefore, a comprehensive approach to software engineering must address the integration of support technologies with software development processes and practice encompassing social, organizational and communication and information issues and structures [27]. These requirements are accentuated by the current trends toward a global marketplace, multinational corporations and an increasingly distributed but interdependent workforce. Multiple teams of people are typically required for most projects of any significant scale and therefore productive teamwork and effective collaborative work processes are central

concerns. Additionally, object-oriented development, which is becoming increasingly widespread, promotes exploratory modeling and iterative prototyping approaches that are more conducive to group processes involving the free exchange of ideas and the negotiation of diverse and distinct viewpoints [7][24], These approaches are reinforced by general trends toward iterative process models for software development. These trends indicate our need to better understand and support the dynamic communication and negotiation processes that lead to shared meaning and decision making in software development.

Problematic situations regularly arise in software engineering due to changes in a multitude of factors; this is so because software engineering occurs within a dense web of social, technical and economic factors, for example [27]:

> System design generally entails articulation of a set of modules, and this system configuration must be mapped onto the staff in terms of division of labor. When any of these get out of sync, they must be renegotiated - the infrastructure must support negotiation of these during development.

These kinds of problematic situations are unavoidable, and cannot be viewed as negative phenomena but as an inherent part of the dynamic decision making processes (including evaluation of progress, additions / changes of personnel, re-decomposition of the system, teams and task assignments, reassessment and reallocation of resources, etc.) involved in iterative development. The resolution of these situations depends on the ability of the participants to regularly negotiate issues and agreements and to maintain a mutual shared understanding or shared (project) memory of the state of the project [17][27]. If the participants get too far or remain to long out of sync in their shared understanding, wasted work, frustration and loss of commitment will result and the chances for success in a project are compromised.

Constantine [7] notes the fundamental importance of maintaining a shared memory to support teamwork in object-oriented system development. Among several critical roles he identifies for team members to play, that of "information manager" is deemed most important and central. Constantine and others advocate a primarily organizational approach to support the dynamic negotiation and decision making processes. However, organizational approaches alone suffer certain drawbacks in maintaining a shared memory within a project or across projects *{corporate memory)* — for instance, key personnel, who are the carriers of shared memory may leave a project or an organization. Therefore, a complementary approach to supporting shared memory is needed that integrates organizational approaches with technical capabilities and infrastructure to facilitate the creation of an explicit external record of shared project memory composed of structured information from multiple media [17][34][35].

Currently, there are a variety of research efforts, growing out of concerns with teamwork in design activities, that are aiming to support collaborative teamwork through the integration of organizational approaches with computational support for shared memory [11][36][20]. At a general level there is common agreement regarding the following kinds of requirements:

- Coordination and management of group activities and work-flow requires that information be made available in a meaningful form at the appropriate time. The creation of and access to meaningful information along with the seamless

integration of the varieties of information created and used during the design of the product is called *information modeling and management.*

- A design support environment should enable project work and management to be carried out within the same uniform information modeling environment. Facilities should be built into the environment that will enable designers to create shared structures of information (text, models, source code, sketches, pictures).
- The environment should permit asynchronous (different time, different place) group activity.
- Facilities are required to enable retrieval of information by participants with diverse views of that information.

4.3 gIBIS

gEBIS ([6]) is a hypertext system designed to facilitate the capture of early design deliberations. It supports a method and rhetorical model called IBIS ([19]), which has been developed for structuring discussions on large, complex, design problems. IBIS structures information as a graph with three types of nodes: *issues, positions* and *arguments.* Nodes refer to other nodes via links of different types: for example a position has a *responds-to* link to an issue. An argument must be linked to a position via either a *supports* or *objects-to* link. Issues may be organized by linking them to other issues via *specializes, generalizes, replaces* or *questions* links. Finally, gIBIS introduces a catch all node type and link type *{other)* which allows users to structure information in ways unanticipated by IBIS The set of legal rhetorical relationships of the nodes and links in IBIS is summarized in Figure 2..

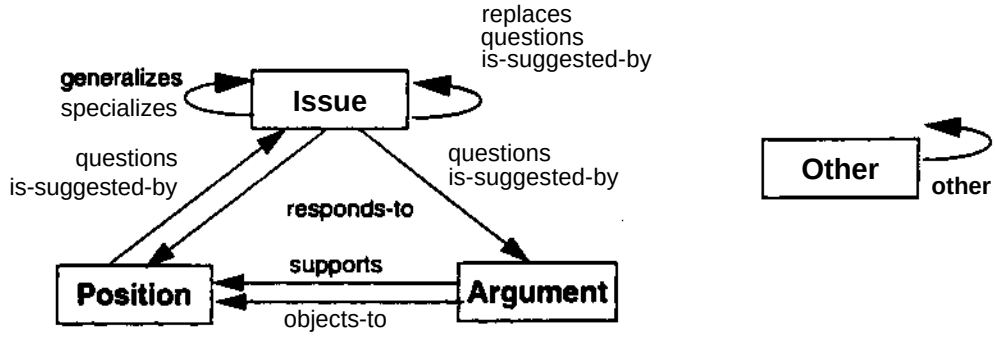


Figure 2. The set of legal rhetorical relationships in IBIS

The gIBIS tool allows multiple users to concurrently build and access IBIS networks. It provides two main indices into the issue base: a graphical browser, which displays the graph as nodes and links, and a node index, which allows the user to examine the nodes in the order they were posted. Issue subgraphs can be clustered, thus allowing the users to organize the graph in a hierarchical manner. gIBIS also provides a search mechanism allowing users to search nodes by title, author and node content.

gIBIS showed the IBIS method to be powerful for design deliberation and capturing design rationale [38]. The Issue-Position-Argument framework helped designers to focus their thinking on the critical parts of problems and to more readily detect incompleteness and

inconsistencies. The IBIS model provided a canonical representation for capturing design decisions, thus providing an information base which was understandable by all participants in the design. Moreover, not only decisions were captured, but their rationale as well, thus allowing decisions to be revised later in the design without fear of hidden dependencies. Overall, the use of gIBIS "paid for itself, that is, the additional time spent recording information in gIBIS was paid off by early detection and correction of design problems.

However, Conklin and Yakemovic noted a few areas of possible improvements. In general, gIBIS lacks mechanisms for dealing with evolving networks of issues which can lead to certain problems in its use, particularly when extended in duration and scope: difficulty of maintaining and accessing appropriate history; difficulty in coping with evolving terminology when searching through the issue base; lag time between when an issue is effectively resolved and when the resolution is entered in the issue base. When studying gIBIS users, they also noticed a trend from the users to use geometrical information to create indices (e.g. by convention, some users would always position annotations on the upper right corner of an issue context, critical issues in the upper left corner, etc.). Such information was not accessible to the search mechanism.

4.4 /*-Dimensional Information Modeling (/i-dim)

> **... comparative advantage *[in product development]* will be in sensitive integration of hardware, software and humanware [sic] into an effective system of designing[4].**

n-dim [8][10][20][21][25][34][35] has been conceived primarily as a system to support humans in the process of design, (and we believe the basic approach to be broadly applicable to any team-based work context). The n-dim group views design as a social process, as opposed to a set of algorithms or a search strategy. This view has been developed and reinforced over the years by a number of empirical studies (as cited in Section 4.1) conducted both by the *n-dim* group and other researchers.

In our view design consists of an extensive set of negotiations between design participants over both the product and the processes used during design and manufacture, while design shares many features of other social processes, it imposes special requirements on information systems due to its own specific attributes. Chief amongst these attributes is the need for collaboration among participants who often come from a wide range of backgrounds requiring the negotiation and reconciliation of different perspectives, terms, and knowledge bases. Hence, design takes place in a highly heterogenous environment wherein cross-disciplinary actors confront and must be responsive to varied social and economic factors that govern the entire life cycle of the artifact being designed. Finally designs take place in larger organizational contexts which increasingly impose cost-effectiveness requirements on their investments in design, designers, and design tools. An organization's investment in design knowledge is expensive, and needs to be recouped by maintaining the knowledge for subsequent reuse by designers with divergent perspectives: the designer, the new design group, or a different organizational entity.

Since our goal was the creation of useful tools for supporting design, the implications of its social nature and its specific characteristics needed to be operationalized. These operational characteristics were the basis for creating several increasingly sophisticated versions of a design support environment we call n-dim.

4.4.1 Current Information Modeling and Management Technologies

Information modeling and management involve the discovering and/or creating, the abstracting, the interpreting, the organizing, the using, and the sharing of information. The computer, and very specifically the networking of computers, is at the heart and will continue to be at the heart of all new information modeling and management technology. Important considerations are supporting ways to discover information located elsewhere, to organize information for the purpose of exposing relationships unong it and for sharing it both directly and by making it available for others to discover.

A simple form of managing information is to organize files containing it under a directory/subdirectory tree as is done in all computer operating systems. One typically organizes such a directory system to put things which are related into the same subdirectory. Through the use of "links" (a special pointer to a file), a file can appear to be in more than a single subdirectory. If one browses a subdirectory, the system lists a link exactly as it lists a file name. Clicking on a link will move one to the subdirectory containing the file and will open the file. The system will now be in the subdirectory where the file is located with the disadvantage that the user now views the file in the context in which it is located, not in the context in which it referred to. Thus, this simple form of multiple indexing has drawbacks, not the least of which is that in these circumstances, the system (and often the user) will not know where they are nor remember how they got there. Files can contain any of a variety of things: text, drawings, audio recordings, film strips, and so forth.

A second way to organize information is to place it in a database, and a related third way are document management systems. A fourth approach is to organize information into a hypertext A *hypertext* is a text or graphics document with buttons attached to it. Clicking a mouse on one of these buttons causes the system to open up another page of text that the author has *linked* with the current document by associating it with the button. On-line help systems today often are hypertexts. The document author places buttons which are often invisible under a piece of text (or within a drawing) so one gets the impression that one is clicking on the text. Doing so will bring up a page relevant to that piece of text. A hypertext is, therefore, a complex interlinked document which one can read by following several different paths through it. The hypertext concept is an important ingredient of the world wide web (WWW).

Placing meaningful labels on the links which connect two documents in a form the system recognizes further extends the capability to relate information and is part of the semantic network modeling one often uses to create data models for databases. Such labeling is also an important ingredient of the n-dim information modeling environment.

In response to the implications and suggestions gleaned from the above mentioned case studies (Section 4.1) and the possibilities that new technologies offer, there are many properties we hypothesize we want to have in information management systems. We want to keep and link information on people, on their activities, on the formal and informal organizational structures relating those people, on the numerical and symbolic data describing physical and software artifacts, on the group development

of arguments such as described by the rhetorical model of gIBIS (see Section 4.3), on drawings, on correspondence and e-mail files, and so forth. We want to be able to store tremendous amount of information. We want to be able to find things in this information by browsing, by formal searching and by being pointed to it because someone else or the system concludes it might be interesting to us. We usually want very fast responses from such a system. We will often want the system to record enough history of the data creation process that we can study the process and/or reproduce the arguments behind the *why* of the data and not just its values. The argument that the blueprint for a design is not enough supports such capture. Often, especially for a new activity, deciding what information to gather, how to structure it and how to share it is a major part of the effort one makes in managing it. We want these systems to support exploration among alternatives for organizing and interrelating it.

With respect to this extensive list of potential requirements, there are certain advantages and disadvantages of each of the above mentioned information technologies depending on their context of use and the goals and expectations of the processes they are intended to support. What is clear is that greatly expanded information management is now possible but it will take time to discover all the attributes support systems must have for information management as no one has yet really tested these systems. A more detailed description of the WWW and n-dim are presented in [25] along with a comparison along key issues in three archetypal application contexts, Generating Standards, Supporting Collaboration and Reusing and improving work processes; the key issues identified are structuring information, allowing ad hoc views, finding information, maintaining history, speed of access and controlling who can join (the information environment).

n-dim represents a hypothesis for the more comprehensive type of support needed (than provided by the WWW for instance) for collaboration, reuse and the continuous creation of multiple and incremental classifications leading toward standards.

4.4.2 Information Modeling in n-dim

> **... my first premise, based on what I have observed, is that designing is a social process.... [the design] exists only in a collective sense. Its state is not in the possession of any one individual to describe or completely define, although participants have their own individual views, their own images and thoughts, their own sketches, lists, diagrams, analyses, precedents, pieces of hardware, and now spreadsheets which they construe as the design [3].**

n-dim provides a generalized framework for modeling information, and has been designed to be used either by itself, or as a foundation for building domain- and organization-specific applications and environments.

n-dim is based on three fundamental mechanisms: *aflat space* of objects, the ability to build *modeling languages,* and a notion of persistence based on the metaphor of the library, called *publishing.* In our experience, the whole of these mechanisms is greater than the sum of the pans — together, they achieve a "critical mass" in that no subset of them yields the desired functional attributes.

Flat Space of Objects

The space of objects managed by n-dim is conceptually flat: there is no single, system-imposed hierarchy. Rather, many different views, hierarchical and otherwise, of this space can exist simultaneously.

The primary information-structuring primitive is the *model.* Models consist of a set of *references* to other objects and a set of *links* which relate these references to one another. *Atoms* are objects which contain single *values,* such as strings, integers, etc. One atom which deserves special mention is the *operation,* which is used to encapsulate a piece of executable code. Operations can be thought of as being *methods* in an object-oriented programming environment, n-dim models may have many associated operations.

As any given n-dim object can be referred to by many models,[1] n-dim objects[2] do not have any single, intrinsic name. Rather, names are a property of references in models. More formally, references are tuples of the form *<model target, title>,* where *model* is the model from which the reference emanates, *target* is the object being referred to, and *title* is the name given to *target* in the context *of model.* So, for instance, an object might be referred to twice from a given model, with the first reference having a name in Swedish and the second reference a name in English, or simultaneously from several different models in which it plays different roles. In this sense, n-dim models can be thought of as *contexts.*

The links in a model connect its references; they are tuples of the form *<source, target, type>,* where both *source* and *target* are references in the containing model. Links are purely intra-model; there is no link whose source or target are not referred to by the model containing the link.[3] The *type* attribute of a link does not carry any intrinsic meaning in itself — Ai-dim will treat it as a label. The modeling language mechanism provides a way for giving links and types of links richer semantics in specific contexts.

Models as **Languages**

All models have a reference to another model which serves as their *modeling language.* The modeling language of a model constrains the kinds of parts and links which can be included in the model. Any model can be used as a modeling language. Ai-dim provides a number of built-in models which can serve as modeling languages, e.g. Universal is a modeling language which allows any part and any link to be included in a model. The property of being or not being a ML has to do with the *use* of a model as a language, as opposed to some intrinsic attribute. Models are used as languages by invoking the *Use As Language* primitive on them, which results in a new model being created. The model used as a language is interpreted by n-dim as a *grammar* which constrains the contents of the newly created model.

---

1 **Or. in fact, multiple times from the same model.**

2 **We will use the term "*n*-dim objsource.targeuypeecf to cover both models and atoms in what follows.**

3 **There is actually nothing preventing inter-model links in principle, as links are actually dealt with internally as first-class objects, but it was felt that it introduced more problems than it solved in the current round of implementation.**

The relationship of the new model to the model used as a language is similar in nature to that of an instance to us class in certain object-oriented systems, but the similarity is distant enough that it should only be thought of metaphorically. The strongest resemblance lies in the fact that models inherit any associated *operations* from their language at the time of creation. It is also possible to refine or even override system-defined primitives with user-defined operations, allowing for the imposition of more stringent semantics on references and links than currently is possible with the ML mechanism.

Given these facilities, the potential structure and behavior of a set of models can be specified and grouped together in a model which can be used as a language, and all models created using it as their language will inherit these basic features. However, unless specific action is taken to limit the structure of instances, they are free to evolve independently, within the grammatical constraints of the language.[1] For instance, individual models may have operations defined on them which supercede or subsume those defined by their language.

The rationale behind the modeling language mechanism is that users should be able to define their own kinds of modeling structures, instead of having to rely on system administrators or programmers to do it for them. It then can easily be seen how information modeling structures such as the IBIS-Issue-Position-Argument model can be accommodated in such a system.

Private, Public and Published Models

All n-dim objects have an associated *sharing status,* one of *private, public* or *published.* The sharing status can only change in one direction: private —> public—> published. All n-dim objects start out in the private state.

Private objects exist in a user's private Workspace, and are not visible to any other user of the system. The only way to share an object is to move it to one of the other two states. Such a change will force a strict legality check that ensures the model is a valid instance of its language.

Published objects are analogous to published papers in a library. Publication simultaneously makes an object immutable and visible to others.[2] In order for a model to be published, the transitive closure of all models to which it refers must be published, and the model which acts as its language must also be published. Once published, it is truly immutable, even to its original author; if a revision to a published model is needed, the only option is to *copy* the published model and change the copy. The new copy will start out as a private model, which may be freely modified. Copying a model also updates a system-maintained *pedigree model* which collects the revision history of a model from its initial creation.

Public models can be thought of as being halfway between private and published models. They are visible to all users, but can be modified, to a limited degree,

1 This is enforced all of the time by default, but can be temporarily relaxed while users works on a model or set of models in their private workspace.

2 There is an access control mechanism, implemented using Al-dim models, but its operation is more or less independent of the published/private distinction.

concurrently by multiple users. The limitation is that references and links may only be *added* to public models, not removed or modified.[1]

/i-dim: Support For Collaborative Activities

We believe that these three main features (together with other capabilities which can be built with them such as *access control revisions,* etc.) will support the following essential teamwork activities:

Structuring of information: Incremental creation and/or reuse of models

Communication, coordination & negotiation of information: **sharing of models** with others in a shared information space

Preserving information within a shared (corporate) memory: **recording of all** project information models for future reference as a *side effect* of current project activities.

4.4.3 Current Status

n-dim has progress through three generations of development, from a Macintosh hypertext version to a demonstration multi-machine, multi-user system. The current version is a complete rewrite, comprising the **BOS** (Basic Object System) in C and supported by an interactive language called STITCH. The system (refered to as n-diml.l) is currently maintained by four graduate students and two staff programmers. A public release of n-dim 1.1 is scheduled for the end of 1995.

The current system is stable enough that we can quickly implement new ideas (such as IWEB), test them with real users, and refine them as we take into account user feedback. The current system can maintain and store without performance degradation as many as several hundred thousands of n-dim objects.

4.5 How IWEB Incorporates These Streams

As will be described in detail through the rest of this document, the central information structuring concept of IWEB - the "Issue Forest" - is based squarely the type of rhetorical model and shared issue-based support environment pioneered by the **IBIS** and gIBIS work. However, we believe that IWEB adapts and extends this work in important ways.

First, IWEB embedds an issue-based support environment within a more comprehensive and integrated information modeling environment such that much more of the overall project information can be interrelated, cross-indexed and captured in an on-line navigable web to support on-going communication and decision making as well as for archival purposes. In its own terms, gIBIS is a limited stand-alone argument support system. It was primarily developed to capture design rationale on a project and to handle some part of the large amount of "unstructured" information on a project where lack of structure in infomation is appropriately characterized by the statement "that the information generated in the course of ongoing activitiy is heterogenous, ad hoc and imbedded in the activity of its

1 **Certain user-interface realizations of models may implement the illusion of removing references or links from public models by adding special links, called *tags,* but the underlying representation maintains the old information.**

production and use" [38]. However, much of the information generated on a project, in its structure and use, is not appropriate to put directly into an issue base. For example, organization charts or the various types of information produced in modeling a software artifact through requirements, analysis, design and implementation [1]. Yet it is important that these and all types of related project information are able to be cross-referenced from an issue base and vice versa. For instance, we believe it will be common to want to refer to an object model (such as those typically produced in using the OMT method and tool [26]) from an Issue or other node in the issue base; and it would be useful to be able to refer to a node from the issue base in another context such as in a Task assignment model.

While gIBIS provided a limited means to refer to other information that is not in the issue-base, it did not envision a means to comprehensively structure all project information nor to flexibly interrelate this information using a generic linking device from which the types of links in the issue base itself could be specialized. **IWEB** accomplishes this by sitting on top of n-dim: the Issue-forest template is actually just another modeling language embedded within rt-dim (one of potentially many others such as organization, process, document models, etc.). In addition to the specialized components and interfaces of **IWEB,** its users will be able to use all of the power of n-dim (to which they have access privileges) to create information models and modeling structures as needed and to cross-reference these as needed to and from the Issue Forest. This will specifically include certain model templates (modeling languages) which will be created for software engineering project support. An illustration of the use of n-dim to explore a large set of useful information structures for a software engineering project is provided in [10]. This was a case study using actual project information, but it did not include an issue base. In **IWEB,** we plan to augment these types of information models with an issue base and to make them mutually complementary by providing uniform and transparent cross-referencing.

Second, IWEB will provide several types of redundant access and notification mechanisms to enable users to become aware of information that has been added to Issue Forests, to be reminded of when and where they viewed the information, and to notify others of the existence of specific issue forest information and optionally annotate it with comments or elaborate it with additional cross-references. The specific concepts and mechanisms to support these capabilities. What's New Models, Notices and Notification Boxes, are described in Section 5.2.25 through Section 5.2.31 of the IWEB concepts and throughout Section 7 on IWEB use cases. These mechanisms are provided in addition to the types of flexible search and filtering capabilities described for gIBIS. These and others are supported in n-dim also and thus will be provided in IWEB.

It is our expectation that the number of nodes in the Issue Forests in a deployment of IWEB on a project may grow very rapidly with the size, scope and duration of a project, (this is dependent, or course, on the number of participants and their decomposition into roles and teams and so on). The What's New Model will enable users to view only those nodes and links that have been added since they last consulted IWEB, and then to be directed, if desired, to the overall context where these were added within an Issue Forest. Notices enable users to direct the attention of other individuals, groups, teams or the overall project participants to a

[1] **Or, whatever development phases are elaborated ana supported in one of the numerous software development methods and CASE tools that have been proliferated, especially in object-oriented approaches.**

specific Issue Forest entry (node, or nodes and links) that may be especially important or requires timely response. Notices are similar to e-mail messages which may have attachments. Issue Forest nodes and/or links. They support a very informal and unstructured communication process which smoothily extends the more formal and structured discussion and negotiation process captured in the issue forests. Again, the informal and formal formation is easily linked and the history of the exchange of information is captured. Going iyond gIBIS, this interlinks a rhetoncal model and issue base with the broader, situated and conversational information exchange that takes piace within projects.

We believe that these notification and reminding mechanisms planned for IWEB are required to keep an issue base active and alive and effectively utilized by a critical mass of a project's participants. These mechanisms will extend the on-line support of an issue base by enabling its users to view, remember, recover and notify others about information in specific contexts of its usage with respect to their sessions with IWEB, times of posting, viewing or reviewing and their organizational roles and assignments.

Third, based on our experience wth an earlier direct implementation of the EBIS rhetorical model in M-dim (a subsystem we called Negotiate) and our experience of managing meetings, communication and negotiation in the software project course, we changed and slightly extended the rhetorical model for IWEB. The specific nodes and links supported are described in the Section 5, IWEB concepts (for a synposis see Figure 5.) and in Section 7 in the IWEB use cases. It is our experience and that of others [38] that there is a trade-off between the *expressability* of a rhetoncal model (that is, the number, type and complexity of the nodes and links) and its *usability* (that is, the model itself cannot be too elaborate or users will have trouble learning and using it and may reject it). With this in mind, in IWEB we are experimenting with an modest elaboration of the basic IBIS model that we view as a hypothecal design that must be tested and may evolve through use. [1]

In summary, IWEB is planned to be a dynamic issue-based communication, negotiation, structuring and notification system which in conjunction with n-dim will greatly expand the capabilities of previous issue-based systems. It will provide these capabilities in a project support infrastructure with the ability to interlink heterogeneous information both within and across projects in a comprehensive navigable web. IWEB and n-dim represent our hypothesis for the type of support needed for collaboration. Our next step is to prove this hypothesis by actual case studies.

1 As of the the time of final editing of this document, we have already provisionally included an additional node type for Action Items based on emerging requirements from our introduction of IWEB into use for documenting meetings in a software project course in the form of Indented Text IWEB (itIWEB, see Section 12).

5 IWEB Concepts

This section describes the key terms and concepts that underlie IWEB and its modules and are used in our communications about them. First, we give a brief overview of a few general terms and concepts for *information modeling* and *n-dim* that are needed to understand our approach and certain dependencies that IWEB has on n-dim. These concepts and many others related to n-dim are described in detail elsewhere and available from the authors of this document [8]. Then, the concepts specific to IWEB are presented in the following informal groupings: General, Issue Forests, Issue Forest Nodes and Links, Nodes and Links Related Concepts, Notice and Notification, IWEB Interface Components, User Information, Organizational Information and Software Project Course. Concepts, as well as actors and use cases, appear in Helvetica type.

The central concept of IWEB is that of the Issue Forest. An Issue Forest is a method and computer-based model for structuring discussions and communication between participants in a collaborative work setting. It is composed of a variety of nodes and link-types (Issues, Proposals, Arguments and Resolutions) that support an IBIS-like rhetorical model, and enables a bulletin-board style interaction among its users [38]. The Issue Forest and other components of IWEB, taken together with the general information modeling and management environment that it sits on top of (n-dim), are intended to broadly support a style of asynchronous computer-supported cooperative work.

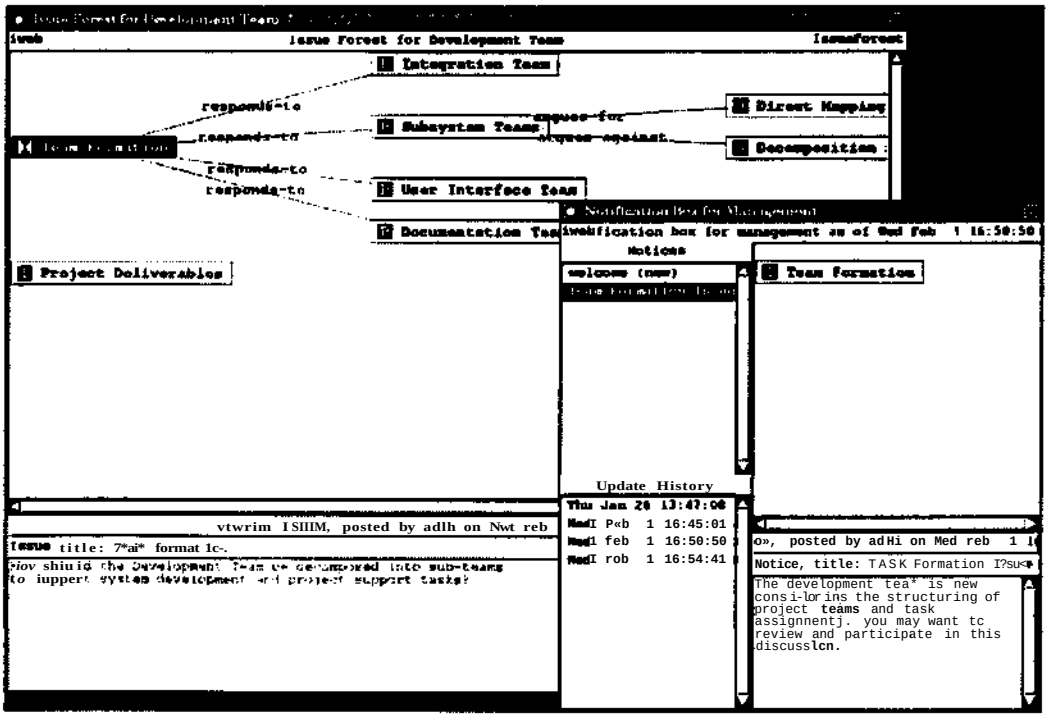


Figure 3. An Overview of IWEB

Figure 3. illustrates the use of issue forests and notification boxes. The leftmost window shows an issue forest used by the development team of a project for discussion of development related issues. The rightmost window shows a notification box used for

notifying management. The issue forest of the development team contains two issues: *Team Formation* and *Project Deliverabies.* The description of the *Team Formation* issue appears at the bottom of the issue forest window, together with its author and the date when is was posted. The nodes displayed at the right of the issue are proposals and arguments related to the issue that other members of the development team have posted in response to the issue. The management notification box displays a notice entitled *Team Formation Issue* which was sent by one of the members of the development team. The notice includes the issue being discussed (displayed in the upper right hand corner of the notification box) and includes a description (displayed in the lower right corner of the notification box).

5.1 Information modeling and n-dim Concepts

This is a partial listing of terms and concepts associated with n-dim that are essential to understand IWEB (which is built on n-dim). For a more complete description see [8].

5.1.1 n-dim

n-dim is an acronym for n-Dimensional Information Modeling. It is an approach and prototype computational infrastructure for comprehensive information modeling and management to support team-based work.

5.1.2 n-dim Object

An n-dim Object is an information object that is referenced from n-dim.

5.1.3 Flat Information Space

The space of n-dim objects *is flat:* n-dim objects are not instrinsically organized hierarchically, in the sense that objects contain objects (e.g. such as in the case of a unix file system). Any n-dim object may refer to, or be refered to by any number of objects.

5.1.4 n-dim Controlled Object

an n-dim Controlled Objects is an n-dim object whose existence is controlled by n-dim because they are stored in n-dim. All n-dim controlled objects have at least three attributes: a unique id, the time created and the time last modified. Figure 4

gives a classification of Ai-dim controlled objects.

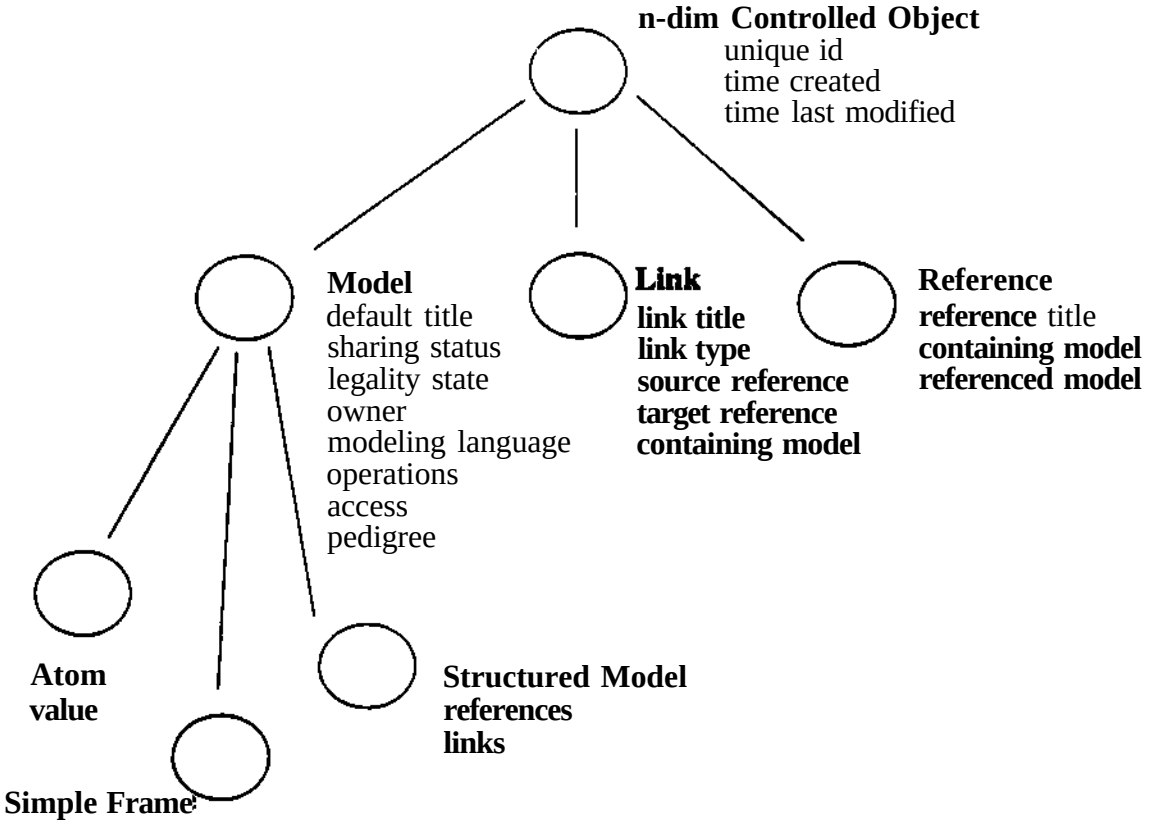


Figure 4 Classification of n-dim controlled objects and their attributes

5.1.5 Ai-dim Non-Controlled Object

An n-dim non-Controlled Objects is an n-dim object whose existence is not controlled by n-dim because it is stored external to n-dim.

5.1.6 Model

A model is a description of information created within n-dim that is either an atom, frame, or structured model. Every model has the following attributes: owner, sharing status, legality status, access, default title, operations, and language.

5.1.7 Reference

A reference to a model within a structured model.

5.1.8 Link

A link is a user defined relationship between two parts contained in the same model. A link has a source, target, title, and type. Its source and target are parts that

the link s are relating, and the title and type are symbols.

5.1.9 Structured Model

A structured model is a collection of References and links.

5.1.10 Atom

An atom is an unstructured model with a single value.

5.1.11 Owner

Owner is a reference to the person model of the user who created the model.

5.1.12 Access

Access is a reference to a model describing which users can read and modify that model.
For operations, the access also describes the users who can invoke the operations. In order for users to modify a model, they must have both access to the model and the operation they have to invoke in order to modify the model.

5.1.13 Sharing Status

The sharing status of a model determines whether a model can be shared by multiple users, and if it can be shared, whether the users can modify the model. The sharing status of a model can take one of three values: *private, public* or *published.*

5.1.14 Private

A *private* sharing status indicates that only the owner of the model may access and modify it. Any modification to the model is allowed.

5.1.15 Public

A *public* sharing status indicates that any user whose person model is referred to by the access model can view the model. However, users who can access a *public* model must also have access to the various operations of the model in order to modify it (e.g. only users who have access to the add reference operation of a public model may add references to the given model; only users who have access to the remove reference operation of a public model my remove references from it). Any modifications to a public model are seen immediately by all users who are viewing the model.

5.1.16 Published

A *published* sharing status indicates that any user whose person model is referred to by the access can view the model. However, nobody may modify or remove the model, regardless of their access to the operations of the model.

5.1.17 Legality State

The legality state of a model provides an indication to whether or not a given model satisfies the constraints of its modeling language. The legality state of a model can take one of two values: *legal* and *under construction*..

5.1.18 Legal

Legal is one of the possible states of a model that has Private status; the other state is Under-Construction. A Legal model is necessarily completeand consistent with respect to required and/or allowed nodes and links according to its modeling language. When we say that a model is Legal, it is by implication a model with the status Private and that can be changed to become under construction by modification or by an explicit action of the user to Place Model Under Construction; Public and Published models are always enforced to be in a Legal state.

5.1.19 Under-Construction

Under-Construction is one of the possible states of a model that has Private status; the other state is Legal. A model that is in the under-construction state is a model that can exist only in a users private workspace and that is not necessarily complete or consistent with respect to required and/or allowed nodes and links according to the modeling language that was selected in which to build the model. When we say that a model is Under-Construction, it is by implication a model with the status Private. There is some visible (UI) indication on both the model's Open and Closed Views that it is under construction.

5.1.20 Operations

The operations of a model is a set of references to a number of operations atoms. An operation atom is an n-dim model which may be executed and perform an action on the model on which the operation is invoked on.
A model inherits the operations of its language, that is, the set of operations which can be invoked on a model is the union of its set of operations and the set of operations which can be invoked on its modeling language.

5.1.21 Modeling Language

The modeling language of a model is a reference to another model that constrains the content of the given model.

5.1.22 Pedigree Model

A pedigree model is a structured model describing the history of a model. Every time a model is copied, a link is created in its pedigree model between the original model and its copy.

5.1.23 Person Model

A person model is an Ai-dim model representing the identity of an n-dim User. Person models are used to identify the owner of a model and to define the access

rights to the model.

5.2 IWEB Terms and Concepts

General

5.2.1 IWEB Object

An IWEB Object is any of the Issue Forest Nodes or any the other information objects created within a Modeling Language specific to IWEB, e.g., Project, Organization, Team, group, and User models. An IWEB Object is a subclass of n-dim Object.

Note: It is envisioned that IWEB Objects will eventually include models created in many other specialized modeling languages for software engineering that support Project, process and artifact modeling and their interrelationships. For examples of these types of models see the case study of the integration Objectory models (an Object-oriented Analysis and Design method and tool) with Project and process models created in n-dim [10].

Issue Forests

5.2.2 Issue Forest

An Issue Forest is a computer-based, structured information base for collaborative design that supports communication and discussion and provides a history of design issues and decisions. Multiple Issue Forests can be created for a Project and specialized according to use and access depending on project requirements and organizational context (see 5.3.1,5.3.2 and 5.3.3 for examples of specific instances of Issue Forests that we believe will be useful in the context of a software project course.) Typically, an Issue Forest will be accessed and modified concurrently by multiple project Members acting in their various Roles as IWEB actors. An Issue Forest serves as a communication medium for developers and clients to raise, discuss and potentially resolve issues related to the Project. It is incrementally structured and populated using Issue Forest nodes connected by Issue Forest links and may recursively reference other Issue Forests.

Issue Forests are instances of and examples of the use of n-dim model that are made *public.* Therefore, anytime a part or a link is added to an Issue Forest, any IWEB actor who has that Issue Forest opened will see the new node without interaction from their side. As with public models" in n-dim, the broadest classes of users, as specified in IWEB by the actor Developer, may only add information to an Issue Forest. A restricted set of IWEB users, /s spec' *ed by the IWEB actors IWEB Manager and Issue Forest Manager, may create issue Forests (in the n-dim sense this means to make a model public which is restricted by access to the operation Make Public) and drop parts from an issue Forest for the purpose of maintenance.

5.2.3 **Root Issue Forest**

A Root issue Forest is a top-ievei Issue Forest that only an IWEB Manager can create. A Root Issue Forest can be referenced as a Generic Node from other Issue Forests (root or otherwise). However, a Root Issue Forest cannot be nested within another Issue Forest. (Examples of top-level Issue Forests that might be created for a specific project context are the General Issue Forest, Client Issue Forest and Management Issue Forests described under Software Project Concepts group (see 5.3.1, 5.3.2 and 5.3.3).

Issue Forest Nodes and Links

5.2.4 Issue Forest Node

The Issue Forest Node describes the common attributes of all of the Issue Forest node types. In particular, it specifies that each node will provide a space in which to optionally put a brief textual comment on the content and purpose of the node. Issue Forest Node is the superclass of all nodes in an Issue Forest. Only the specialized subclasses of Issue Forest Node described below can actually be created in an Issue Forest, not a Issue Forest Node itself.

An Issue Forest Node is a type of n-dim model, and as such, each node contains information about who created it, the time and date it was created, and a unique symbolic identifier within the IWEB/n-dim.

5.2.5 Issue Node

The Issue Node contains text which describes an issue that needs resolution. It can be connected to another Issue Node with a Sub-lssue-of Link or an Abstracts Link. An Issue Node can form the root of a new discussion in which the subject of the Issue Node is not directly related to another Issue Node. Typically, an Issue Node will be created with the intention that it will be resolved (have an associated Resolution Node) in some time frame. [1] An *active* Issue Node is an issue Issue that does not yet have an associated resolution. Issue Node is a subclass of Issue Forest Node.

5.2.6 **Proposal** Node

A Proposal Node contains text which describes a possible proposal to resolve an issue. It must be attached to one or more Issue Nodes with a Responds-to Link. Proposal Node is a subclass of Issue Forest Node.

5.2.7 Argument Node

An Argument Node contains text which states an argument for or against a proposal. It must be attached to a Proposal Node with either an Argues-for Link or an Argues-against Link. An Argument Node may be attached to more than one Proposal.

[1] **This interpretation of the expected usage of Issue nodes, and IWEB nodes and links in general, is based on its intended structure which is very similar to an iBIS-like rnetoncal model for discussions ana decision making. It must be evaluated by use and is subject to adaptation according to the policy and procedures of the organization and project in which IWEB is used.**

Argument Node is a subclass of Issue Forest Node.

5.2.8 Resolution Node

A Resolution Node contains text and possibly other references describing how one or more issues are resoived and may be connected with a Based-on Link to proposals attached to those issues (issues resolved by the same Resolution Node must be related by a sequence of sub-issue-of or Abstracts Links, though issues related in these ways are not all automatically resolved when a resolution is entered for one of them; see 5.2.24). Therefore, a resolution resolves a discussion structured by one or more issues and any associated proposal and Argument Nodes. A resolution may be attached to one or more Issue Nodes with a Resolves-link and to zero or more Proposal Nodes with a Based-on Link. A Resolution includes an Executive Summary that must be completed by the author of the resolution. There can only be one Resolution Node for each Issue Node, and an issue that has an attached Resolution Node is a Resolved Issue. A **Resolution Node** is a subclass of Issue Forest Node.

5.2.9 Generic Node

A Generic Node is a place holder for information in the Issue Forest. It can contain references to other Issue Forest nodes or external information objects (IWEB Objects or n-dim Objects). A Generic Node can be linked from any other node in the Issue Forest, but must be linked with a References Link. Generic Node is a subclass of Issue Forest Node.

5.2.10 Issue Forest Link

The Issue Forest Link describes the common attributes of all of the Issue Forest Link types. In particular, it specifies that each link will have a source and target node each of which must be one of the types of Issue Forest Nodes. Issue Forest Link is the superclass of all links in an Issue Forest. Only the specialized subclasses of Issue Forest Link described below can actually be created in an Issue Forest, not an Issue Forest Link itself.

An Issue Forest Link is a type of /i-dim link, and as such, each link contains the following information: a name, a type, a source and a target.

5.2.11 Sub-Issue-of Link

Sub-lssue-of Link denotes that the source node is a sub-issue of the target node. Both nodes must be Issue Nodes.

5.2.12 **Abstracts** Link

The Abstracts Link denotes that the source node is a more specific version of the Issue stated in the target node. Read in the reverse, the target node is a more general version of the issue presented in the source node. This link connects two Issues nodes. The same Issue Node may abstract or be abstracted by more than one other Issue Node.

5.2.13 Responds-to Link

The Responds-to LinK denotes thai the source node, which must be a Proposal Node, is a proposal for addressing the target node, which must be an Issue Node.

5.2.14 Argues-for Link

The Argues-for Link denotes that the source node, which must be an Argument Node, supports the discussion or idea presented in the target node, which must be a Proposal Node.

5.2.15 Argues-against Link

The Argues-against Link denotes that the source node, which must be an Argument Node, questions or refutes the discussion or idea presented in the target node, which must be a Proposal Node.

5.2.16 Based-onLink

The Based-on Link denotes that the source node, which must be a Resolution Node, is based on the proposal presented in the target node, which must be a Proposal Node.

5.2.17 Resolves Link

The Resolves Link denotes that the source node, which must be a Resolution Node, resolves the issue presented in the target node, which must be an Issue Node.

5.2.18 Reference Link

The Reference Link denotes that the target node, which must be a Generic Node, contains information which is related to the source node. The source node can be any type n-dim Object.

Connectivity of Issue Forest Nodes and Links

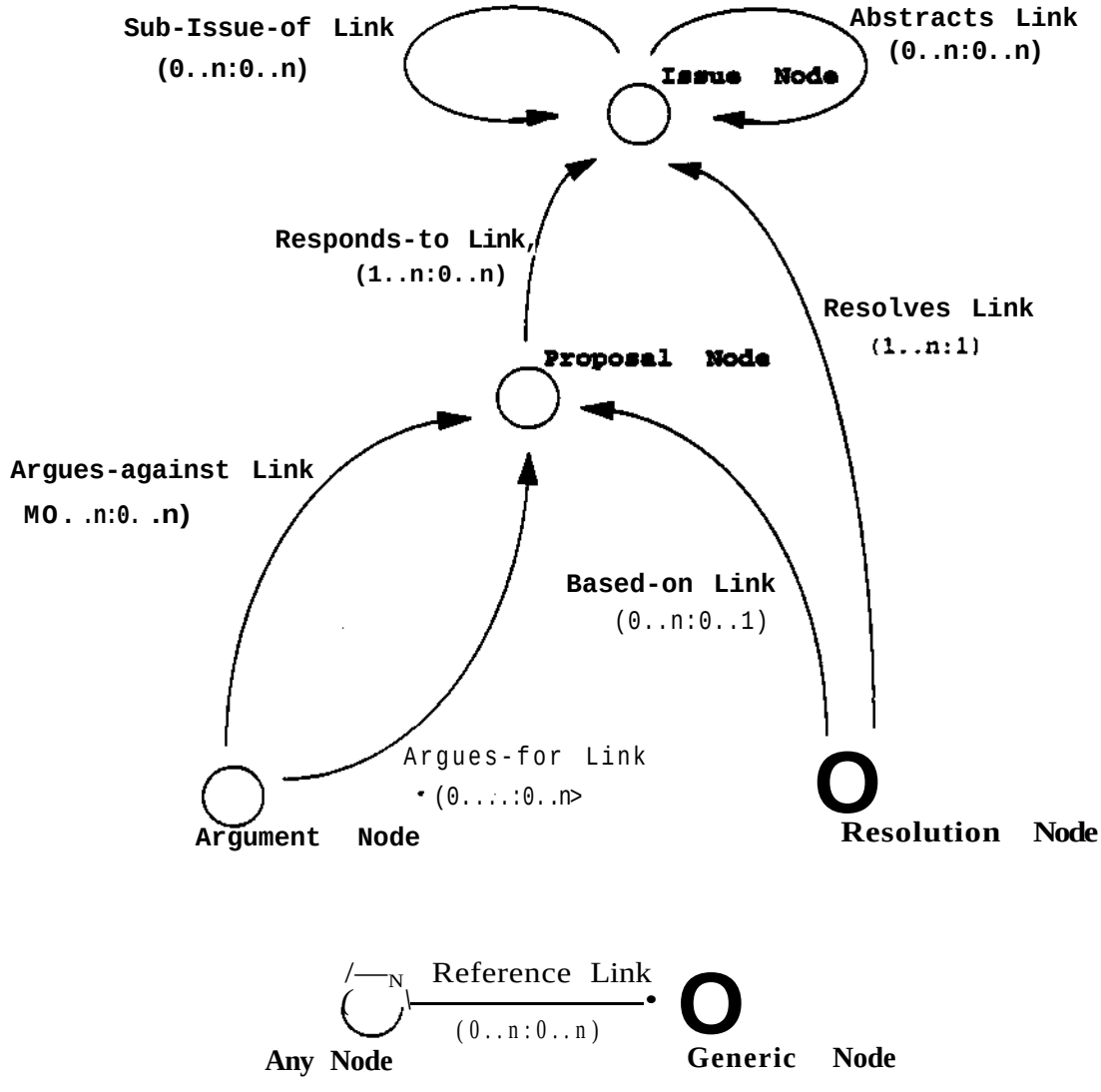
The following table and graph (Figure 5.) summarize the supported connections between nodes and the respective link-types.

Table 2: Connectivity Between Source Node and Target Node

Source Node	Link Type	Target Node
Any Node	**Reference Link**	**Generic Node**
Argument Node	**Argues-for Link**	**Proposal Node**
Argument Node	**Argues-against Link**	**Proposal Node**
Issue Node	**Abstracts Link**	**Issue Node**
Proposal Node	**Responas-to Link**	**Issue Node**
Issue Node	**Sub-issue-of Link**	**Issue Node**

Table 2: Connectivity Between Source Node and Target Node

Source Node	Link Type	Target Node
Resolution Node	Based-on Link	Proposal Node
Resolution Node	Resolves Link	Issue Node



* An Argument Node must be connected to at least one Proposal Node, however, this may be done using either an Argues-tor link or an Argues-against link: therefore the node (Argument) to node (Proposal) is really 1 .. n. but the cardinality ot the relationship specified in terms of either link type must be 0 .. n. because neither is mandatory, either one or the other is required.

Figure 5. Visual Representation of Node and Link Connectivity

5.2.19 **Root** Issue Node

A Root issue Node is an issue Node that currently has no parent Issue Node (that is. is not a sub-issue of any other issue). When a new discussion is started in the Issue Forest, a new Issue Node will be created that is a root issue. A new root issue may also be entered as a super-issue to which an existing issue or issues are connected with a Sub-lssue-of Link. Subsequently, however, the new root issue may become a subissue to other issues by being linked to them through a Sub-lssue-of Link; in that case it ceases to be a root issue. See the use case 7.2.7

5.2.20 Executive Summary

An Executive Summary is a brief overview of an entire related discussion that encompasses a combination of an Issue Node (or Issue Nodes), possible Proposal Nodes and a Resolution Node. An IWEB Author must enter an Executive Summary when they create a Resolution Node. This will typically summarize the discussion represented by a tree or network of IWEB nodes emanating from an Issue (not necessarily a root issue) that is resolved by the new Resolution.

5.2.21 Can-be-resolved-by List

This is the list of people who are authorized to and are responsible for subsequently creating / linking a Resolution Node to the Issue Node. The Can-be-resolved-by List consists of the creator of an issue, any other names that the creator wants to add and the default list of people who may resolve issues as established by the project.

5.2.22 Issue Forest Access

Issue Forest Access is a model which contains two lists of users: 1. those who can view the Issue Forest (acting in the role of IWEB Users); 2. those who can edit (create and post nodes and drop notices in Notification Boxes) the Issue Forest (acting in the role of Authors.)

5.2.23 Node-and-Link Subassembly

A Node-and-Link Subassembly is subset of connected nodes and links from a single Issue Forest that may be referenced together elsewhere (for instance in a Notification Box or What's New Model) - any links that are included in the subassembly must be accompanied by both their source and target nodes.

5.2.24 Resolved Issue

A Resolved Issue is an Issue Node that has a corresponding Resolution Node attached with a resolve-link. This link may be created directly by an Author actor, or may be created automatically when a Resolution Node is attached with a Based-on Link to a Proposal Node that is connected to its parent issue: see (Create Resolution Node, 7.2.4). A Resolved Issue is displayed differently than issues that have not been resolved. If an issue is resolved, all of its attached sub-issues (if any) are

automatically resolved, i.e., resolution links are automatically created between the same Resolution Node and all of the sub-issues. However, if ail of the sub-issues of an Issue Node are resolved, this does not necessarily completely resolve the (super) issue (there is not a strict composition relationship between issues and sub-issues); therefore, no Resolves Link is automi: 'cally created to the (super)issue. Resolution links will not be automatically addea . issues which another issue abstracts, i.e., abstracts links do not play a role in automatically propagating resolutions. See 5.2.8and 7.2.4.

Notice and Notification

The following are concepts to facilitate the easy noticing by users of new entries into the Issue Forest (What's New Model), and their direct notification about something in the Issue Forest by another User who wants to draw their attention to new or existing entries (Notification Box).

5.2.25 What's New Model

A What's New Model provides a User an index of everything that has been added to a specific Issue Forest since the last time they consulted it. A What's New Model is created for each User / Root Issue Forest pair and is automatically maintained by the system. On opening or update by the User, the What's New Model contains references to all of the new nodes and links that have been created in the Issue Forest since the last time that the User updated their view of What's New.

5.2.26 Notification Box

Notification Boxes may be created at the personal, group, Team and Project level. They provide a means for an IWEB Author to directly notify an individual, Team or Ad Hoc Group about a specific node or Node-and-Link Subassembly in the Issue Forest. A Notification Box is automatically created for each (sub)lssue Forest, for example, one for each Team corresponding to each Teams' Issue Forest within a Project wide Issue Forest (see 5.3.1), or one for each Ad Hoc Group Issue Forest[1] that may be created during a Project. When a Notice is "dropped" into a Notification Box all of the Members on the Notification Box distribution list of the box are alerted that they have a new Notice in the box. All IWEB Authors[2] (project Members) can see all of the project's Notification Boxes and the distribution lists for the boxes. Users can see only the contents of only those Notification Boxes for which they are Members of the distribution list. (See the note under 5.3.4 which discusses the possible configuration of Notification Boxes to match the Organization and specific Issue Forests

1 A Notification Box is not automatically created for an Ad Hoc Group unless a corresponding Issue Forest is created for that group. In that case, however, a Notification Box may still be manually created for the Ad Hoc Group.

2 IWEB Viewer actors can see only the Issue Forests and corresponding What's New boxes. This implies that viewers cannot have distribution boxes nor be on the distribution lists for group or Team boxes. This is an arbitrary decision for simplicity in the first prototype of IWEB. It is based on the assumption that having Notification Boxes implies a level of involvement and interaction in a project that goes beyond just observation or viewing and that people involved at that level will be given privileges of an IWEB author anyway. This assumption must be tested.

that are envisioned for the software project course that motivated the development of IWEB, as discussed in Section 3.)

5.2.27 Notice

A Notice is a model that is created by IWEB for inclusion in Notification Box whenever an IWEB User drops a reference to an Issue Forest Node, an Issue Forest Link or a Node-and-Link Subassembly in the box, or for inclusion in the What's New model whenever a new Issue Forest Link is created
(in which case the Notice will contain the subassembly consisting of a copy of the new link and references to the source and target models.) The Notice contains information about the time and date of the creation and, in the case of a notification, the person who dropped it in the box.

5.2.28 View Access

View Access to a Notification Box means that a User can see the contents of the box. The User will be included on the Notification Box Distribution List.

5.2.29 Drop Access

Drop Access to a Notification Box means that a User can drop notices in the box. All Authors have Drop Access to all Notification Boxes. IWEB Users do not have Drop Access to Notification Boxes. Having Drop Access implies that the User can see all of the Notification Boxes, but a User can only open (see the contents) of those boxes to which they also have View Access.

5.2.30 Notification Box Distribution List

The Notification Box Distribution List is the list of Users who have View Access. Anyone can see the distribution list for a box in order that they can determine whether to notify others by using that box. [1]

5.2.31 Update of a What's New Model or a Notification Box

Update is a User applied operation to the What's New Model and/or a Notification Box. Upon update: all of the items in the respective model that have been viewed (opened, touched - not necessarily read or understood!) by the User and marked for removal are removed from the view of the model; if the update is applied to a What's New Model, any new entries in the Issue Forest will appear in the view of the What's New Model; if the notification is applied to a Notification Box, any new notifications directed to that box will appear there.

IWEB Interface Objects

5.2.32 Issue Forest Directory

An Issue Forest Directory is a list of all Issue Forests that have been created in

1 In this respect, a distribution list and the access list for an Sl-dim model are necessarily distinct and different because in rt-dim a User can not see the access list for a model to which they don't have access.

the IWEB sytern. The User may filter their view of the Issue Forest Directory so that it includes only the Root Issue Forests.

5.2.33 **What's** New **History**

A view of the What's New Model that allows the User to see what appeared in the What's New Model in specified time frames, marked as intervals between one update to the next update.

5.2.34 Notification Box History

A view of the Notification Box model that allows the User to see what appeared in the Notification Box in specified time frames, marked as intervals between one update to the next update.

5.2.35 Notification Box Closed View

The closed view of a Notification Box (or its icon) may be seen by those who do not have access to it. This is necessary so that all IWEB users can notify other users about entries in the Issue Forest. The view of a Notification Box includes the ability to display its distribution list. Only those who are Members of the distribution list may see the contents of a Notification Box (its opened view).

5.2.36 Notification Box Opened View

The opened view of a Notification Box displays its contents to the User (this is equivalent to viewing the contents of an n-dim model). The contents are not structured by any groupings of where the notices in the box came from. This is not needed because when a User opens a Notice (a referenced Issue Forest Node or Node-and-Link Subassembly) the Issue Forest in which it was created or most recently referenced is brought into view and the node or subassembly is highlighted. The User may customize the viewable area and layout of notices within the Notification Box.

User Information

5.2.37 User

A User is a model in IWEB that represents a person who uses IWEB in the role of an IWEB User actor. Each IWEB User will have a corresponding User model that will be created when they are added to list of permitted users of IWEB by the System Administrator. A User model references the following other models with information related to an IWEB User: Contact, Schedule, Roles, Notification Box, and an associated *n-6\rr* Person model (by which the IWEB user's access to various models in IWEB and /z-dim is managed). Each of these referenced models has information specific to the IWEB User represented by the User model.

5.2.38 Contact

A Contact model is associated with a User. It is a text or frame model with information similar to a Unix plan file that describes how to get in touch with a

person through phone, e-mail, address, and so on. It may include both home, office and other sources for these Contact points. It may also include a brief description of the person's interests, projects, task assignments and any other information that the person may want others to know about.

5.2.39 Schedule

A Schedule model is associated with a User. It is a text object that summarizes a person's daily, weekly and/or monthly scheduled such that other IWEB Users, who may be working with that person on a common Project, may infer when the person may be available for meetings or other working Tasks that must be **scheduled.** ***(In the future, this could be structured "calendar" model with operations that enable it to act as an active assistant or agent in arranging meetings with other IWEB Users by interacting with their Schedules.)***

5.2.40 Roles

A Roles model is associated with a User. It is text model that describe what Roles the person is assigned to or plays in a Project that is using IWEB. The model may also contain references to Tasks or one or more models that describe a specific Role.

Organizational Information

5.2.41 Team

A Project Team represents a designated group of Users in the Project that is using IWEB. A Team model references the following other models with information related to Team: associated Issue Forest, Members, Tasks, associated Ad Hoc Groups, and the corresponding Notification Box for the Team.
Teams can only be formed by IWEB Managers and will typically be created at the beginning of a Project; whenever a Team is created, a corresponding Issue Forest and Notification Box are automatically created.

5.2.42 Members

A Members model is a collection of references to Users models. It represents the Members of Team or Organization models (associated with levels or contexts of decomposition) within projects, associated with the same Organization structures, that are using IWEB.

5.2.43 Ad Hoc Group

An Ad Hoc Group represents a group of Users who are working on the same problem or sub-task. Ad Hoc Group models are created in the same Modeling Language as Team models and therefore may include the same referenced models that a Team includes. The Users in a Ad Hoc Group can be from the same or different Teams and the Ad Hoc Group can be referenced by each of these Teams. Ad Hoc Groups can be formed by Issue Forest Managers, who may also be Developers (students), at any time during the course of a Project that is using IWEB. They will

typically be created dynamically to address sub Tasks of Teams or the types of "unforeseen" issues and problems that arise regularly in software engineering project work. When a Team is created, a corresponding Issue Forest and Notification Box are NOT automatically created.

5.2.44 Organization

An Organization model represents the global organizational information for a Project that is using IWEB. An Organization model references the following other models with information related to the Organization: Teams, Ad Hoc Groups, Members and Notification Boxes.

Project Information

5.2.45 Project

A Project model refers to any information relevant to the project such as artifact models and documents, organization structures, process models, tools, and so on. An illustration of the possible content and use of Project models in adapting *n*-dim to support software engineering project work is given in [10].

5.2.46 Task

Tasks are models which include descriptions of discrete pieces of work to be assigned to project Members. Tasks include information about start and end conditions, requirements for completion, the relationship of the Tasks to other Tasks and to the overall project goals, expected time and resources requirements, and so on. Depending on their complexity and scope, Tasks may be assigned to Teams or Users in a Project.

5.3 Software Project Course

In this section we describe concepts for specific Issue Forests and related notification components that illustrate the adaptation of the general concepts of IWEB to the context of the software project course that we are using as a test-bed for the requirements elicitation and evaluation of IWEB (see Section 3). In this context, the set of Root Issue Forest reflects the organizational divisions of the Project into management (in an academic setting this is the instructors and teaching assistants), development and client groups and related issue, discussion and communication needs.

5.3.1 General Issue Forest

This Issue Forest contains issues that can be viewed by all Members of a Project. These issues include design and implementation questions, and any other issues that arise from within the project Teams that have to do with the creation and construction of the software being deveioDed.

5.3.2 Management Issue Forest

This Issue Forest contains issues in the management of the software project. These issues include scheduling, Team decomposition, and other issues which are

decided by management. Only Project Managers can read/post to the Management Issue Forest.

5.3.3 Client Issue Forest

This Issue Forest contains issues that involve interaction with the client. These issues include trade-offs (time vs. cost, time vs. functionality, functionality vs. cost, etc.), interface design, and other client driven discussions. Only Clients and Client-Liaisons can post to the Client Issue Forest.

5.3.4 Project Course Notification Boxes

Assuming that the Teams and persons for an instance of our software project course are the same across the General Issue Forest, Client Issue Forest and Management Issue Forest, only one set of project course Notification Boxes will exist to serve them all. Ad Hoc Group Notification Boxes will be classified according to the Root Issue Forest from which they derive - for instance, groups involved in the General Issue Forest, Client Issue Forest, Management Issue Forest - unless the group applies to all Issue Forest classes. Therefore, general developers will only see the General Issue Forest but will be able to see all of the above mentioned Notification Boxes. A Manager may be able to see the Client and Management Issue Forests as well as the General Issue Forest, but will see the same set of Notification Boxes as the general developer. In terms of the views of Notification Boxes, this implies that there is only one "expanding" Notice box area in the UI, not separate ones per Issue Forest class.

6 Overview of Actors

6.1 IWEB Actors

This section describes the different users of IWEB. Actors model prospective users ~ everything external communicating with the system, including other systems. Actors represent a certain role, or class descnption of behavior, rather than an actual person who uses the system. Actors are the main tool for finding use cases. Figure 6. displays all the IWEB actors as well as their inheritance relationships.

IWEB sits within the larger n-dim environment and specializes n-dim for the software engineering domain. The actors and use cases described in the following sections are, unless specified otherwise, an addition to n-dim.

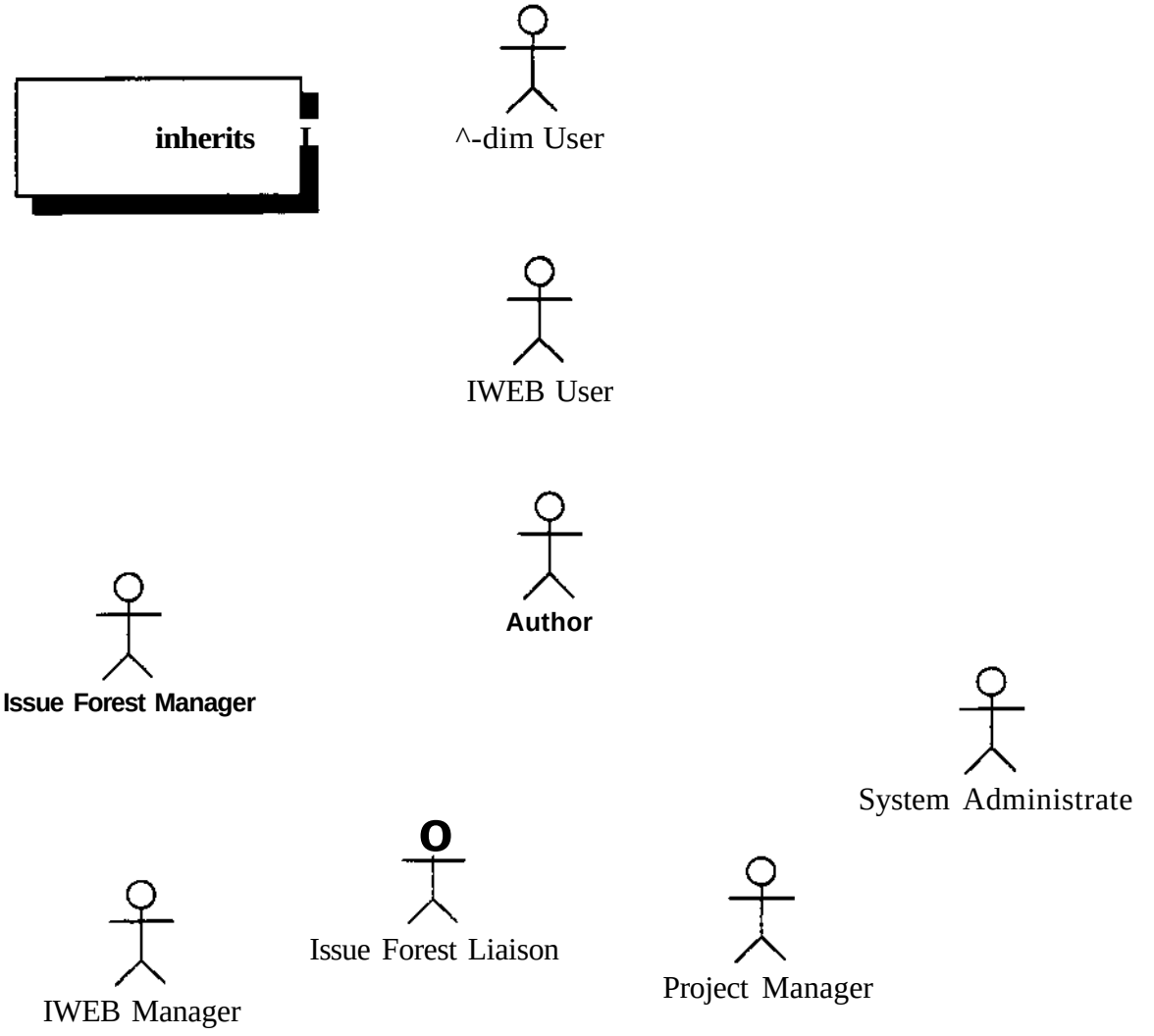


Figure 6. IWEB Actors

6.1.1 /i-dim User

/l-dim User is an actor who accesses and modifies models in n-dim. Each n-dim model has an access permissions enabling or denying an n-dim User the access of a given model and/or any of its operations. The use cases associated with the *n-dim* actor are described in the n-dim use case document [ref].

6.1.2 IWEB User

An IWEB User initiates and terminates IWEB sessions, examines information he/she has access to in IWEB, such as Issue Forests, Notification Boxes and What's New models. An IWEB User can also control the visual display of information, perform searches on data stored in IWEB and examine other information as well. All IWEB Users known to IWEB have a User Model associated with them. Such model contains telephone and e-mail information which allows other IWEB Users to communicate with a given IWEB User. Each IWEB User has the ability to modify its own User modellWEB User inherits from the n-dim User actor.

6.1.3 Author

The role of Author is to make changes to the IWEB information base. This actor sends notices, manipulates ad hoc groups, posts Issue Nodes in Issue Forests. Author inherits from IWEB User.

6.1.4 Issue Forest Manager

The role of Issue Forest Manager is to maintain a set of Issue Forests. Such maintenance work includes removing obsolete Issue Nodes, restructuring an Issue Forest into a number of sub Issue Forests and specifying access rights of other users to a given set of Issue Forests. Issue Forest Manager inherits from Author

6.1.5 IWEB Manager

The role of the IWEB Manager is to maintain a set of Root Issue Forests and other global level information associated with a project. This maintenance work includes creating and removing Root Issue Forests from the Issue Forest Directory, adding and removing users from the access list of Root Issue Forest, and restructuring Root Issue Forests into sub Issue Forests. The IWEB Manager inherits from the Issue Forest Manager.

6.1.6 Issue Forest Liaison

The role of Issue Forest Liaison is to forward Issue Nodes nodes from one Issue Forest to another. The Issue Forest Liaison can also post Issue Nodes to multiple Issue Forests. In a project, use cases associated with the Issue Forest Liaison are typically used by a person whose responsibility is to bridge between organizational divisions (e.g. management vs. development). Issue Forest Liaison inherits from Author.

6.1.7 Project Manager

The roie of the Project Manager is to edit a set of organizational models provided in IWEB. The Project Manager creates and updates Roie. Team and Task models, and maps them to User models. Project Manager inherits from Author.

6.1.8 System Administrator

The role of the System Administrator is to maintain information associated with a given installation of IWEB. This maintenance work includes adding and removing User models from IWEB, restart database servers the IWEB System is not able to restart, repairing corrupted databases if needed, and moving IWEB databases from and to disk partitions.

6.2 Mapping between IWEB actors and the software engineering course roles

This section describes how the IWEB Actors would be mapped to typical roles in a software engineering course. As described in the terms and concept sections, we assume that three root issue forests are created by an IWEB Manager (typically an Instructor or a TA): one for the use of management (Instructors and TAs), one for Developers (students) and one for the Client (interaction between the external Client and the requirements group).

6.2.1 Client

The role of the Client in IWEB is based on the role of the Client in the overall development process: to decide on the functional and interface requirements of the project. A Client answers questions from Developers which are presented by a Client Liaison. A Client has the ability to post in the Client Issue Forest. The Client cannot read or post to the Management Issue Forest, or the General Issue Forest. Client is an Author who only has access to the Client Issue Forest.

6.2.2 Developer

A Developer is a student which is responsible for specifying, designing and building a subsystem or a part of a subsystem for the Client. A Developer is an Author which has only access to the Developer Root Issue Forest.

6.2.3 Team Information Manager

A Team Information Manager is an Issue Forest Manager whose is responsible for maintaining Issues Forests which belong to a team. A Team Information Manager is also a Project Manager whose is responsible for the organizational models associated with a team.

6.2.4 Client Liaison

A Client Liaison is a mediator between Clients and Developers. This purpose of the role is to create the ability to post to both the Developer Root Issue Forest and the Client Root Issue Forest. Additionally, Client Liaisons have the ability to do "cross-posts'* 10 both Issue Forests, thereoy serving as a channel and control point for Client-Developer communication. Client Liaison is an Issue Forest Liaison who has

access to both the Client Issue Forest and General Issue Forest.

6.2.5 Instructor

An Instructor is an Author who has access to all issue forests and organizational models.

6.2.6 Project Information Manager

A Project Information Manager is an IWEB Manager responsible for setting up and maintaining the root issue forests. A Project Information Manager is also a Project Manager responsible for the project organizational models.

6.2.7 Experimenter

The role of the Experimenter is to study the use of IWEB. The Experimenter devises hypotheses about how people use the system, sets up an experimental context, makes observations, and decides whether observations confirm hypotheses. The Experimenter needs support from the system for making quantitative observations. For example, the system might record a replayable transcript of time-stamped user actions including keystrokes, mouse movements, and menu picks. Analysis of this transcript would provide quantitative observations such as typical command sequences and frequencies. This information can be used to justify proposed changes to the system.

7 IWEB Use Cases

This section describes the different use cases of IWEB. For each actor defined in the previous section, a set of use cases describe the functionality associated available to that actor. A use case is a complete course of events in the system from the user's perspective. A use case is composed of a brief description, summarizing the purpose of the use case and which actor invokes it, and a flow of events, completely describing all the user visible events occurring during the interaction of the actor and the system.

This section is organized as follows. Each subsection represents all the use cases associated with a specific actor. When the number of use cases associated with a single actor is large, the actor subsection is further divided into functional groups (e.g. Issue Forest use cases, Notification Box use cases, etc.). For each functional group, a figure summarizes the associations between the actor and all the use cases presented in that group. Subsections are in order: IWEB User (Section 7.1), Author (Section 7.2), Issue Forest Manager (Section 7.3), IWEB Manager (Section 7.4), Issue Forest Liaison (Section 7.5), Project Manager (Section 7.6) and System Administrator (Section 7.7).

7.1 Use Cases for IWEB User

Session Use Cases

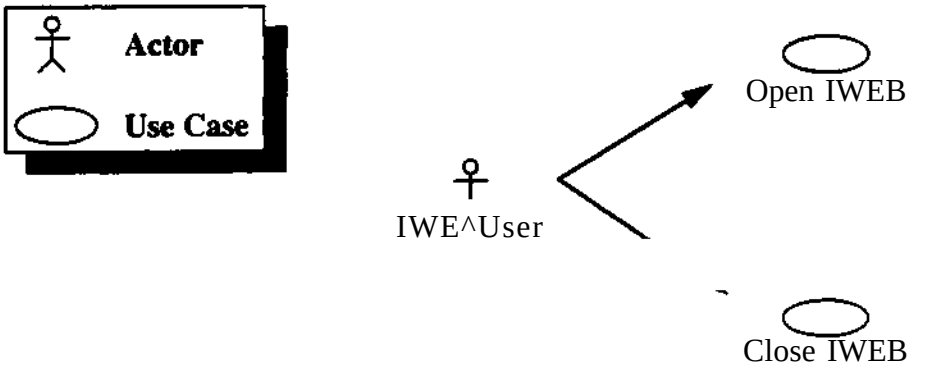


Figure 7. IWEB User - Session Use Cases Communication Associations

7.1.1 Open IWEB

Brief Description: An IWEB session is opened by an IWEB User before any other use case can be invoked. IWEB connects with the database servers associated with the current installation of IWEB. The models (e.g. Issue Forest, What's New and Notification Boxes) which were marked by the IWEB User (or the IWEB Manager) as "open on start-up" are opened.

Flow of Events: When IWEB starts, it attempts to connect with the central IWEB server. In case of failure, IWEB notifies the user that the central services are temporarily unavailable and falls back on the users' local database.

Any model that was marked by the IWEB User as "open on start up" is also opened. If this is the first time the IWEB User invokes this use case, only the default models marked as "open on start-up" by the IWEB Manager are displayed. This use case

ends when the entry port is displayed.

See Visual Presentation in Section on page 79.

7.1.2 Close IWEB

Brief Description: An IWEB session is terminated when an IWEB User selects the quit function from the entry port. IWEB terminates.

Flow of Events: When the user initiates the termination of a session, IWEB checks for any Issue Forest Nodes drafted by the user but not yet posted. The IWEB User has the choice to either save the drafts or post the Issue Forest Nodes as they are. At any point when IWEB queries the IWEB User about posting/saving draft issues, the user has the option of interrupting the termination process. This use case ends when IWEB terminates

Issue Forest Viewing Use Cases

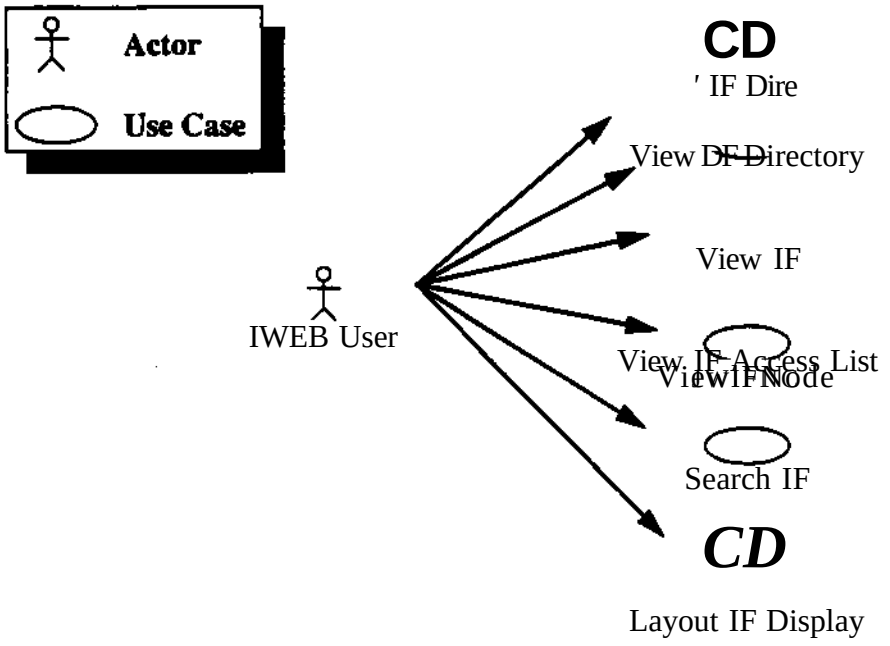


Figure 8. IWEB User - Viewing Use Cases Communication Associations

7.1.3 View Issue Forest **Directory**

Brief Description: The View Issue Forest Directory Use Case is used by an IWEB User to view a list of all existing Issue Forests in the system.

Flow of Events: The IWEB User issues a "view Issue Forests" command from the IWEB entry port. IWEB displays a list of all the Root Issue Forests accessible in the system. The IWEB User has the option of requesting IWEB to list all the Issue Forests accessible by the IWEB User (not only the root forests), as well as their relationships (which Issue Forests are nested in which Issue Forests). This use case ends when the Issue Forest Directory is displayed.

7.1.4 View Issue Forest

Brief Description: The View Issue Forest Use Case is used by an IWEB User to browse through an Issue Forest.

Flow of Events: The IWEB User selects an Issue Forest in the Issue Forest Directory (or in any other open model containing a reference to an Issue Forest, like a parent Issue Forest) and issues a "view Issue Forest" command. This use case ends when IWEB opens the selected Issue Forest.

7.1.5 View Issue Forest Access List

Brief Description: The View Issue Forest Access List Use Case is used by an IWEB User to view the access list of a given Issue Forest.

Flow of Events: The IWEB User selects an Issue Forest in the Issue Forest Directory (or in any other open model containing a reference to the Issue Forest) and issues a "view Issue Forest Access List" command. The use case ends when IWEB opens a dialog listing all the IWEB Users which currently have access to the selected Issue Forest. It is to be noted that an IWEB User is able to see an Issue Forest (open or closed) only if he/she has access to it.

7.1.6 View Issue Forest Node

Brief Description: The View Issue Forest Node Use Case is used by an IWEB User to inspect the content of an Issue Forest Node.

Flow of Events: When an Issue Forest Node is selected for viewing anywhere in IWEB (i.e. directly from an Issue Forest, from What's New, or from a Notification Box), the node information is displayed. Node information will vary depending on the type of node being viewed. All references to the node being viewed (in the Issue Forest, What's New and any Notification Boxes referring it) are subsequently displayed differently to indicate the fact that the IWEB User viewed the node. On subsequent updates, the reference to the viewed node will be removed from What's New or the respective Notification Box. It is to be noted that nodes from the same Issue Forest may be displayed differently in different IWEB Users workspaces, given that they may not have read the same set of nodes. The use case ends when the Issue Forest Node content is displayed.

7.1.7 Search Issue Forest

Brief Description: The Search Issue Forest Use Case is used by an IWEB User to search an Issue Forest. The IWEB User can query for Issue Forest Nodes by various criteria including title, by creation time, and the user who posted the node. The IWEB User can also query for issues by resolution status (e.g. search all resolved issues: search all unresolved issues).

Flow of Events: The IWEB User selects an Issue Forest and issues a search command. IWEB displays a search dialog:

The IWEB User can search issues in the selected Issue Forest by title, in which case they type a partial title. IWEB will then collect all the nodes whose title match the

partial title.

The IWEB User can search the Issue Forest for unresolved issues. IWEB will then collect all the issues which are not ihe target of a resolves iink. and which are not subissues of an issue which is the target of a resolves link.

The IWEB User can search the Issue Forest by user, in which case the IWEB User chooses a set of users he/she is interested in. IWEB will then collect all the nodes posted by any of the users included in the set.

The IWEB User may select any of the above, in which case IWEB will collect the nodes satisfying all of the search criteria.

The IWEB User may restrict the scope of the search to the selected Issue Forest or to include nested Issue Forests in the search.

In all cases, IWEB will return a model that collects the results of the search. The IWEB User may choose to include the search model in his/her workspace or discard it. This use case ends when IWEB displays the search model.

7.1.8 Layout Issue Forest Display

Brief Description: The Layout Issue Forest Display Use Case is used by an IWEB User to personalize the layout of an Issue Forest. Layouts of Issue Forests are stored on a per user basis, allowing different users to view the same Issue Forest differently.

Flow of Events: The IWEB User selects one or more nodes in an Issue Forest and drags them with the mouse. The use case ends when IWEB moves all the selected nodes and updates the shape of any connected link. The IWEB User may also select a single link and modify its shape by dragging it. The layout of an Issue Forest is stored on a per user basis, allowing each user to personalize the layout of the same Issue Forest.

"What's New" Use Cases

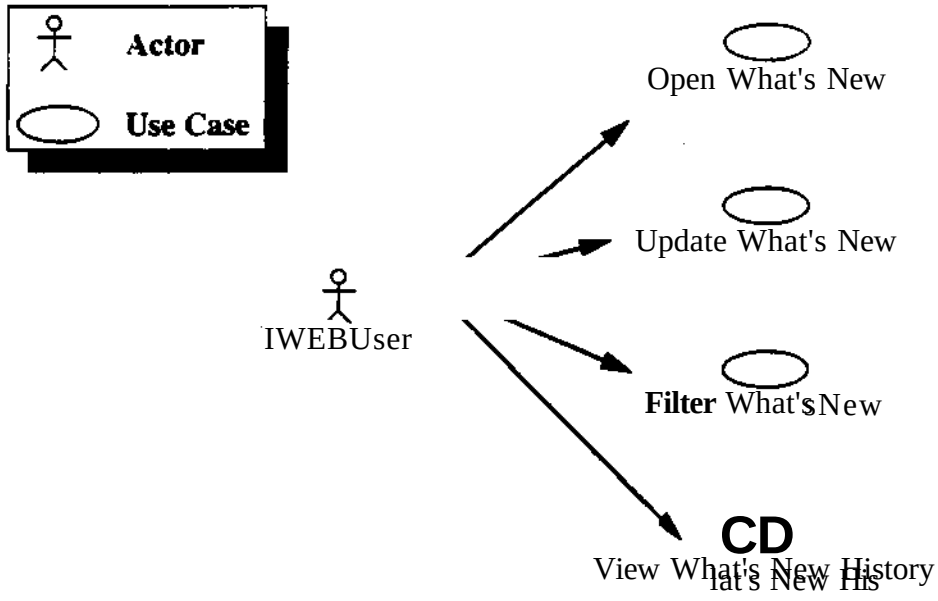


Figure 9. IWEB User - "What's New" Use Cases Communication Associations

7.1.9 Open What's New

Brief Description: The Open What's New Use Case is used by an IWEB User to view the nodes and links which were recently created.

Flow of Events: The IWEB User selects "What's New" from the IWEB entry port, and the What's New Window opens. IWEB opens the What's New Window and includes all the nodes and links which were created since last time the What's New window was either opened or updated by this IWEB User. The What's New window also includes all the nodes and links which were previously created but not yet viewed by the IWEB User. The use case ends when the What's New Window is displayed.

See Visual Presentation in Section on page 79.

7.1.10 Update What's New

Brief Description: The Update What's New Use Case is used by an IWEB User to update the state of the What's New window to include nodes and links which were created since the last time they updated/opened the What's New window.

Flow of Events: The IWEB User selects "Update" in the What's New window. IWEB modifies the content of the What's New window to include all the nodes and links which were created since the last time the What's New window was updated or opened. The nodes and links which were previously created but not viewed by the IWEB User continue to be included in the What's New window. Any node which was viewed by the IWEB User is dropped from the What's New model. Any node which was ever referenced by the What's New model can be recovered by an IWEB Viewer using the View What's New history. The use case ends when the new contents of the What's New Window is displayed.

7.1.11 Filter What's New

Brief Description: The Filter What's New Use Case is used by an IWEB User to restrict the nodes displayed in the What's New window based on their type, title. Issue Forest or author.

Flow of Events: The IWEB User issues a "filter" command in the What's New Window. IWEB displays a filtering control panel. The IWEB User can then select which nodes to display in the What's New Window by setting filters in the filter window. Filters include date, keyword, author, topic, and Issue Forest name. After specifying a filter the user issues an "apply filter" command in the filter control panel. The IWEB User has the option to apply the filter to the What's New Window only for this session, or for all subsequent updates of the What's New Window. The use case ends when the filtered nodes are displayed and the filter panel is closed.

It should be noted that the filter affects only what is viewed in the What's New Window. Nodes that have been filtered out continue to be referenced in the What's New Model and can be subsequently viewed.

See Visual Presentation in Section on page 80.

7.1.12 View What's New History

Brief Description: The View What's New History Use Case is used by an IWEB User to restore the What's New Window to a previous state.

Flow of Events: When an IWEB User issues a "view history" command in the What's New Window, IWEB displays a list of all dates/times at which the What's New model was updated or opened by this IWEB User. The IWEB User can then select one of the date/times in the history. IWEB then displays the What's New Window as it was at the selected time. Nodes which the IWEB user has already read are displayed differently.

An alternate way for an IWEB User to view a previous state of the What's New model is to issue a "view model as at last update" command. IWEB then displays the What's New Window as it was before the last update. An IWEB User can repeatedly issue "view model as at last update" commands to browse backward through the What's New history. Similarly the IWEB User can issue *kview model as at next update" commands to browse the What's New history forward. Issuing a "view model as at next update" command on the most recent version of the What's New Window is equivalent to issuing an "update" command (see Update What's New Use Case).

Thi use case ends when the IWEB User has completed browsing through What's New History.

See Visual Presentation in Section on page 80.

Notification Boxes Use Cases

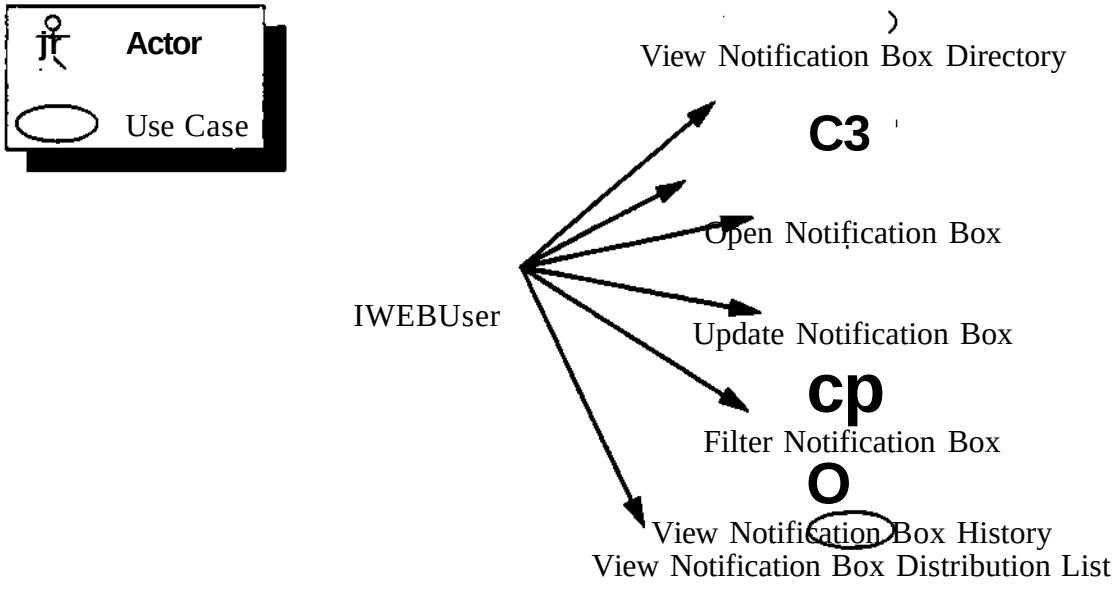


Figure 10. IWEB User - Notification **Boxes** Use Cases **Communication** Associations

7.1.13 View Notification Box Directory

Brief Description: The View Notification Box Directory Use Case is used by an IWEB User to view the Notification Boxes in the system.

Flow of Events: The IWEB User issues a "view notification boxes" command in the IWEB entry port. IWEB displays a list of the Notification Boxes in the system. The IWEB User has also the option of displaying only the Notification Boxes they have access to. This use case ends when the Notification Box Directory is displayed.

7.1.14 Open Notification Box

Brief Description: The Open Notification Box Use Case is used by an IWEB User to open and inspect the contents of a Notification Box.

Flow of Events: The IWEB User selects a Notification Box in the Notification Box Directory and issues an "open notification box" command. IWEB displays a View of the Notification Box, containing all the notices which were dropped in the selected Notification Box since last time the IWEB User updated or opened the Notification Box. The Notification Box View will also continue to include any notices from before which have not yet been viewed by the IWEB User. This use case ends when the Notification Box is displayed.

7.1.15 Update Notification Box View

Brief Description: The Update Notification Box View Use Case is used by an IWEB

User to update the contents of a Notification Box View.

Flow of **Events:** The IWEB User selects a Notification Box View and issues an "update" command. IWEB updates the content of the Notification Box View to include any notices which were dropped in the Notification Box since it was last opened or viewed. The Notification Box View will also continue to include any notices from before which have not yet been viewed by the IWEB User. The use case ends when the new contents of the Notfication Box is displayed.

7.1.16 View Notice

Brief Description: The View Notice Use Case is used by an IWEB User to view the contents of a notice in a Notification Box.

Flow of Events: The IWEB User selects a Notice in a Notification Box View and issues a "view" command. IWEB opens the Issue Forest which is associated with the notice and highlights in the Issue Forest the subassembly referred to by the Notice. If needed, IWEB scrolls the Issue Forest View to allow the user to view the subassembly. This use case ends when the Issue Forest Nodes of the Notice are highlighted.

7.1.17 Filter Notification Box View

Brief Description: The Filter Notification Box View Use Case is used by an IWEB **User** to restrict the types of Notices displayed in the Notification Box View. Notices can be restricted by Author, date of notice, keyword, topic and Issue Forest Name.

Flow of Events: The IWEB User issues a "filter" command in the Notification Box View. IWEB displays a filtering control panel. The IWEB User can then select which Notices to display in the Notification Box by setting filters in the filter window. Filters include date, keyword, author, topic, and Issue Forest name. After specifying a filter the user issues an "apply filter" command in the filter control panel. The IWEB User has the option to apply the filter to the Notification Box View only for this session, or for all subsequent updates of the Notification Box View. The use case ends when the filtered nodes are displayed and the filter panel is closed.

It should be noted that the filter affects only what is viewed in the Notification Box View. Nodes that have been filtered out continue to be referenced in the Notification Box View and can be subsequently viewed.

See Visual Presentation in Section on page 80.

7.1.18 View Notification Box History

Brief Description: The Notification Box History Use Case is used by an IWEB User to restore the state of the Notification Box View to a previous state.

Flow of Events: The IWEB User selects an open Notification Box and issues a "view history" command. IWEB displays a list of all date/times at which the Notification Box was opened or updated. The IWEB User selects one of those times and IWEB restores the Notification Box View to the same state as it was at the selected time. Notices which have already been viewed are displayed differently.

An alternate way for an IWEB User to view a previous state of the Notification Box View is to issue a "view model as at last update" command. IWEB then displays the Notification Box View as it was before the last update. An IWEB User can repeatedly issue "view model as at last update" commands to browse backward through the Notification Box history. Similarly the IWEB User can issue "view model as at next update" commands to browse forward through the Notification Box history. Issuing a "view model as at next update" command on the most recent version of the Notification Box View is equivalent to issuing an "update" command (see Update Notification Box Use Case).

This use case ends when the IWEB User is done browsing the Notification Box history.

See Visual Presentation in Section on page 80.

7.1.19 View Notification Box Distribution List

Brief Description: The View Notification Box Distribution List Use Case is used by an IWEB User to view the set of users who have access to view the notices of a given Notification Box.

Flow of Events: The IWEB User selects a Notification Box and issues a "view notification box distribution list" command. IWEB displays a model referring to the user models associated with all the IWEB Users who have access to the Notification Box. Any IWEB User who has drop access to a Notification Box may view its distribution list. They may not themselves be on the distribution list per se. In case the Notification Box is associated with an Issue Forest, the distribution list of the Notification Box is identical to the access list of the Issue Forest. This use case ends when the Notification Box Distribution List is displayed.

7.1.20 View Notification Box Usage List

Brief Description: The View Notification Box Usage List Use Case is used by an IWEB User to view the sub set of IWEB Users in the distribution list of a Notification Box who have actually accessed the Notification Box.

Flow of Events: The IWEB User selects a Notification Box and issues a "view notification box usage list" command. IWEB displays a model referring to the user models associated with the IWEB Users who have accessed the Notification Box. Any IWEB User who has drop access to a Notification Box may view its usage list. They may not themselves be on the distribution list. This use case ends when the Notification Box Usage List is displayed.

User Model Use Cases

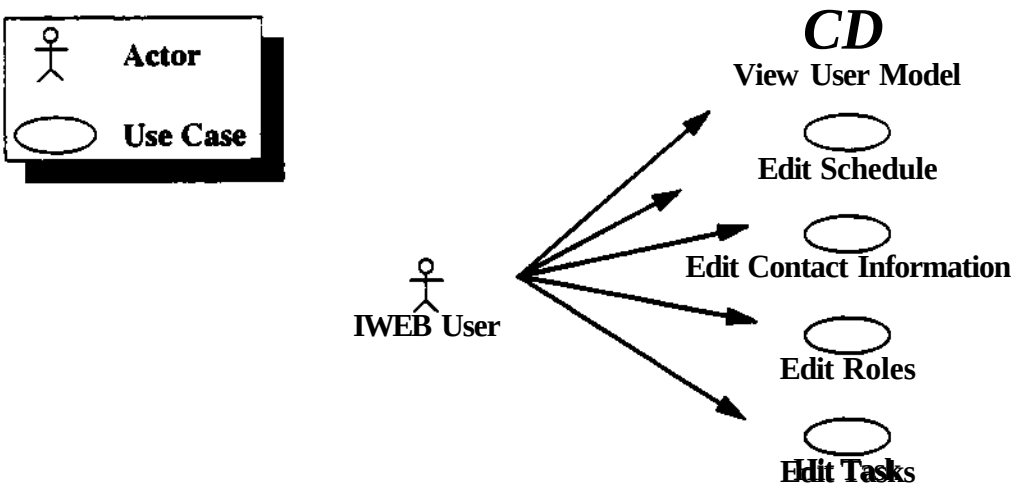


Figure 11. IWEB User - User Model Use Cases Communication Associations

7.1.21 View User Model

Brief Description: The **View User Model use case is used by a IWEB User to refer to the contact,** schedule **information, roles and tasks associated with an IWEB User.**

7.1.22 Edit Schedule

Brief Description: The **Edit Schedule use case is used by an IWEB User to modify his/her** available times.

7.1.23 Edit Contact Information

Brief Description: The Edit Contact Information use case is used by an **IWEB User** to modify his e-mail address, **phone number, and** so **on,** as **stored by IWEB.**

7.1.24 Edit Roles

Brief Description: The Edit Roles use case is used by **an IWEB** User to add or remove Roles from his/her User Model.

7.1.25 Edit Tasks

Brief Description: The Edit Tasks use case is used by an **IWEB** User to add or remove Tasks from his/her User Model.

Organizational Model Use **Cases**

7.1.26 View Organizational Model

Brief Description: The View Organizational Model use case is used by a **IWEB** User to browse through project organization, user, role, group and team models stored by

IWEB.

7.2 Use Cases **for Author**

Edit Issue Forest Use Cases

Figure 12. illustrates the relation ship between IWEB Issue Forest use cases and n-dim use cases. The use cases displayed in gray are n-dim use cases; the use cases displayed in white are IWEB use cases. A *uses* relationship between the Add Issue Forest Node use case and the Publish Model use case means that the Add Issue Forest Node use case "reuses" part of the flow of events of the Publish Model use case.

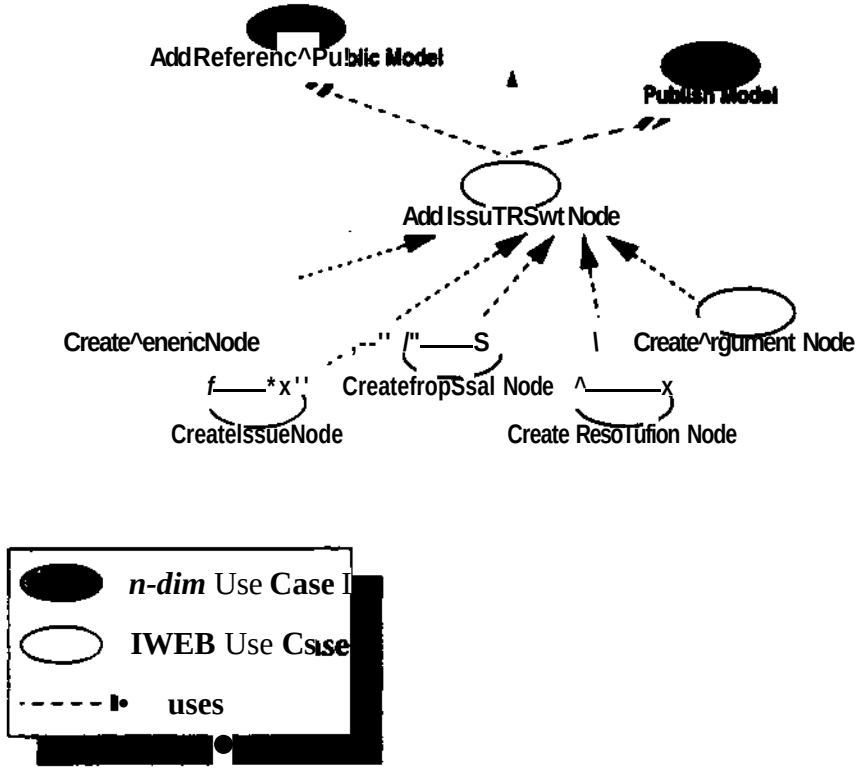


Figure 12. Issue Forest *Use* Case **Dependencies**

7.2.1 Add Issue Forest Node

Brief Description: The Add Issue Forest Node is an abstract use case used by some Issue Forest use cases to add Issue Forest Nodes to an Issue Forest.

Flow of Events: The Add Issue Forest Node use case uses parts of the n-dim use case Publish Model to publish an existing Issue Node and uses parts of the n-dim use case Add Reference To Public Model to add the node in the Issue Forest. This use case ends when the Issue Forest Node is displayed in the Issue Forest.

Edit Issue Forest Coupled Node-Link[1] Use Cases

The following use cases are used by an Author to add nodes to an Issue Forest. IWEB ensures that the issue models are syntactically correct by creating the required links for each issue node which is created.

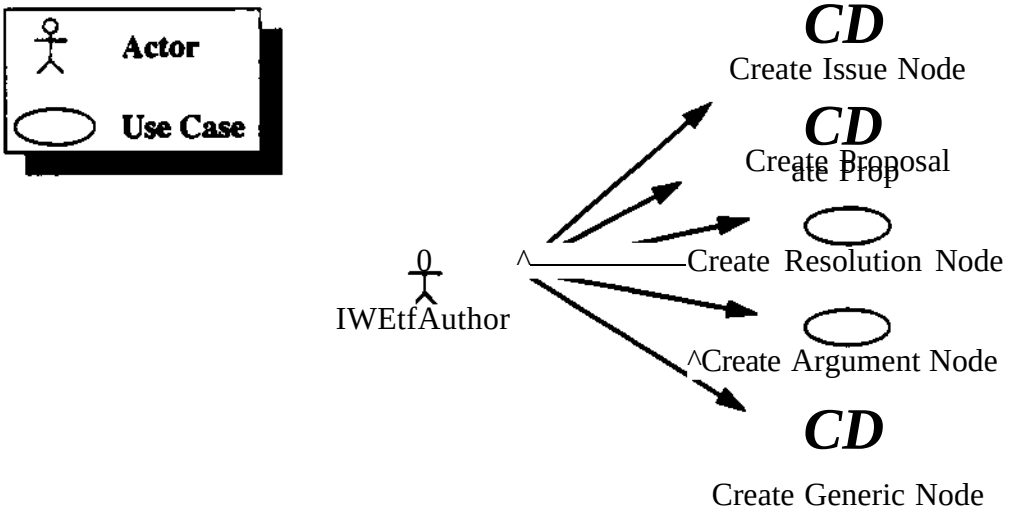


Figure 13. IWEB Author - Issue Forest Node-Link Use Cases Communication Associations

7.2.2 Create Issue Node

Brief Description: The Create Issue Node Use Case is used by an Author to create a new Issue node in IWEB. The Author is automatically included in the *can-be-resolved-by list* for the Issue node and may also enter additional names to the list. This use case uses the Add Issue Forest Node Use Case.

Flow of Events: The Create Issue Use Case starts when the Author issues the "create issue" command. The Author is given the opportunity to enter new information about an issue into the system. The Author enters a textual description of the issue. This use case ends when the new Issue is displayed in the Issue Forest.

7.2.3 Create Proposal Node

Brief Description: The Create Proposal Node Use Case is used by the Author to create a new Proposal node in IWEB that will be connected to an existing Issue node by a responds-to link. This use case uses the Add Issue Forest Node Use Case.

Flow of Events: The Create Proposal Use Case starts when the Author selects an Issue node to which the Proposal is to be addressed and issues the [kt]create proposal" command. The Author is then given an interface to enter a new proposal node for resolving the issue. The Author enters a description of the Proposal, and indicates when they are done. The new proposal node is added to the Issue Forest and a

1 In this group ot use cases, the user will create one or more new nodes, and any required links will automatically be created by the system between two designated nodes - the new node and another user selected node of the appropriate type.

"responds-tcT link is automatically created between the currently selected issue and the new node. The use case ends when the new node and link are added to and displayed in the Issue Forest.

7.2.4 Create Resolution Node

Brief Description: The Create Resolution Node Use Case is used by the Author to create a new Resolution node in IWEB that will be connected to an existing Proposal node by a based-on link and/or connected to other existing Issue nodes by resolves links. This use case uses the Create IWEB Node Use Case.

Flow of Events: The Create Resolution Use Case usually starts when the Author selects an existing Proposal node and issues the command "Create Resolution." The system checks if the Author is authorized to post a resolution to the Issue that the Proposal is attached to by determining if the Author is a member of the *can-be-resolved'by list* for that Issue. If this is not the case, IWEB abons the command and displays an error message to be acknowledged by the Author. The Author is then given an interface to enter the contents of a new Resolution node. As part of the Resolution the Author must enter an Executive Summary of the entire related discussion that the Issue / Proposal / Resolution encompasses. When the Resolution node is completed the following links are automatically created: a "based-on" link is created between the new Resolution and the Proposal and a "resolves" link is created between the new Resolution and the parent Issue node of the Proposal.

An alternative course of events starts when the Author selects an existing Issue and issues the command "Create Resolution." Similar to above, the system checks if the Author is authorized to post a resolution to the Issue, and if not, the command is aborted. When the Resolution node is completed a resolves link is automatically created between the new Resolution and the selected Issue node.

In either of the above cases, the status of the affected Issue node is changed to that of a Resolved Issue and it is displayed differently than issues that have not been resolved. The use case ends when the new Resolution node and the appropriate new link are added to and displayed in the Issue Forest.

7.2.5 Create Argument Node

Brief Description: The Create Argument Node Use Case is used by the Author to create a new Argument node in IWEB that will be connected to an existing Proposal node by an argument-for or an argument-against node. This use case uses the Create IWEB Node Use Case.

Flow of Events: The Create Argument Node Use Case starts when the Author selects a a Proposal node in IWEB and issues the "add argument node" command. The Author then enters a textual commentary as the body of the Argument Node. The Author must select whether this is an argument for or against the parent node. The node is added to the IF and a link of the selected type is created between the new node and the Proposal node being commented on. The use case ends when the node and the new link are added to and displayed in the Issue Forest.

7.2.6 Create Generic Node

Brief Description: The Create Generic Node Use Case is used by the Author to create a new generic IWEB node of any n-dim type and link it to any other existing IF node with a reference link. This use case uses the Create IWEB Node Use Case.

Flow of Events: The Create Generic Node Use Case starts when an Author selects an IWEB node to which they want to attach a generic node containing any other system supported information object construct. (For example, the generic node may contain related information for annotation or reference; the information may often be textual (annotation) in which case a text editor will be brought up.) The Author then issues the "create generic node" command. This use case ends when the creation of the generic node is completed and when a reference link is created from the selected Issue Forest node to the new generic node.

Issue Forest Link Use Cases

The following use cases are used by an Author to add links to an Issue Forest. It should be noted that all Issue Forest links (except reference links) have their title set to their type. Therefore, the IWEB Author never specifies explicitely the title of an Issue Forest link.

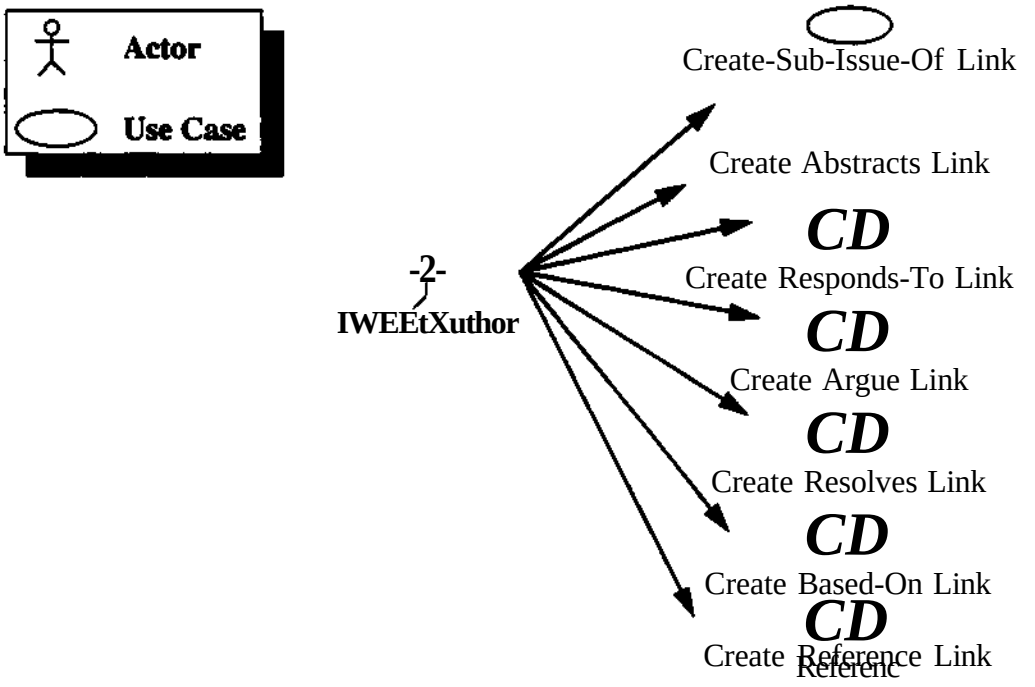


Figure 14. IWEB Author - Issue Forest Link Use Cases Communication Associations

7.2.7 Create Sub-Issue-Of Link

Brief Description: The Create Sub-Issue-Of Link Use Case is used by an Author to create a sub-issue-of link between two existing issues of the same Issue Forest.

Flow of Events: The Create Sub-Issue-Of Link Use Case starts when the Author selects two issue nodes (source, then target) and issues the *create sub-issue-of link* command. IWEB checks if the issue nodes are already connected by a path of sub-

issue-of links. If that is the case, IWEB aborts the command and reports an error to the Author. Otherwise. IWEB creates a sub-issue-of link between the source and target issue and sets the title of the link to *sub-issue-of.* This use case ends when the link is displayed.

7.2.8 Create Abstracts Link

Brief Description: The Create Abstracts Link Use Case is used by an Author to create an abstracts link between two existing issues of the same Issue Forest.

Flow of Events: The Create Abstracts Link Use Case starts when the Author selects two issue nodes (source, then target) in an Issue Forest and issues the *create abstracts link* command. IWEB checks if the issue nodes are already connected by a path of abstracts links. If it is the case, IWEB aborts the command and reports an error to the Author. Otherwise, IWEB creates an abstracts link between the source and target issue and sets the title of the link to *abstracts.* This use case ends when the link is displayed.

7.2.9 Create Responds-to Link

Brief Description: The Create Responds-to Link use Case is used by an Author to create a responds-to link between an existing proposal and an existing issue. This use case is typically used when an Author creates a Proposal which responds to more than one Issue.

Flow of Events: The Author selects a Proposal and an Issue node in an Issue Forest and issue the *create responds-to link* command. IWEB checks if the pair of nodes is already linked with a responds-to link, in which case, IWEB aborts the operation and reports an error to the Author. Otherwise, IWEB creates a new *responds-to* link between the selected nodes and sets its title to *responds-to.* This use case ends when the link is displayed.

7.2.10 Create Argue Link

Brief Description: The Create Argue Link Use Case is used by an Author to create an argue-for or an argue-against link between an existing argument and proposal node in the same Issue Forest.

Flow of Events: The Create Argue Link Use Case starts when the Author selects an argument node and a proposal node in an Issue Forest, and issues the *create argue link* command. IWEB checks if the argument node already has an outgoing argue-for or argue-against link to the selected proposal. If it is the case, IWEB aborts the command and reports an error to the Author. Otherwise, the Author selects whether an argue-for or an argue-against link is to be created. IWEB then creates an argue link from the argument to the proposal and sets the title of the link to *argues-for* or *argues-against* respectively. The use case ends when the link is displayed.

7.2.11 Create Resolves Link

Brief Description: The Create Resolves Link Use Case is used by an Author to create a resolves link from a resolution node to an issue node. Although the

resolution node may already have an outgoing resolves link to another issue, the Author may want to add another resolves iink to an issue abstracting or containing the first issue.

Flow of Events: The Create Resolves Link Use Case starts when the Author selects a resolution node and an issue node in an Issue Forest, and issues the *create resolves link* command. IWEB first checks if the following is true:

- the Author belongs to the *can-be-resolved-by* list of the selected issue,

- the resolution node does not always have a resolves link to the selected issue,

- there is a path of sub-issue-of or abstracts links from an issue referred to by the selected resolution node to the selected issue node.

If any of the above checks fail, IWEB aborts the command and reports an error to the Author. Otherwise, IWEB creates a reference link between the resolution and the issue and sets the title of the link to *resolves.* This use case ends when the link is displayed.

7.2.12 Create Based-on Link

Brief Description: The Create Based-on Link Use Case is used by an Author to create a based-on link from an existing resolution node to a proposal node. Although the resolution node may already have a based-on link to a proposal, the Author may want to add another based-on link to a different proposal to indicate that the resolution was synthesized from a number of proposals.

Flow of Events: The Author selects a resolution node and a proposal node in an Issue Forest, and issues the *create based-on* link command. IWEB first checks if the pair of nodes are already connected by a based-on link, in which case, IWEB aborts the operation and reports an error to the Author. Otherwise, IWEB creates a based-on link between the selected nodes and sets its title to *based-on.* This use case ends when the link is displayed.

7.2.13 Create Reference **Link**

Brief Description: The Create Reference Link Use Case is used by an Author to create a reference link from an Issue Forest Node to an arbitrary IWEB node in the same Issue Forest. If the node the Author wants to create a reference link to does not belong to the same Issue Forest, the Author may use the Add Node Use Case prior to this use case to add the node to the Issue Forest.

Flow of Events: The Create Reference Link Use Case starts when the Author selects an Issue Forest Node (source), an arbitrary IWEB node (target) in the same Issue Forest and issues the *create reference link* command. IWEB checks if there is already a reference link between the selected nodes. If it is the case, IWEB aborts the command and reports an error to the Author. Otherwise, IWEB creates a reference link between the source and the target. This use case ends when the link is displayed.

Notification Boxes Use Cases

Figure 15. IWEB Author - Notification Box *Use* Cases Communication Associations

7.2.14 Notify via Notification Box

Brief Description: The Notify via Notification Box use case is used by an Author to notify an individual or team about a specific node or subassembly in an Issue Forest.

Flow of Events: The Author selects one or more Issue Forest Nodes and/or Links in an Issue Forest and a Notification Box he/she has drop access to. The Author then issues a "notify" command. IWEB creates a notices, adds references in the notice to the selected subassembly and adds the notice in the Notification Box. IWEB then broadcasts the fact that a new notice has been added to the notification box to all the IWEB Users who have that Notification Box open for viewing. This use case ends when the Notice is availble to the other IWEB Users.

7.3 Use Cases for Issue Forest Manager

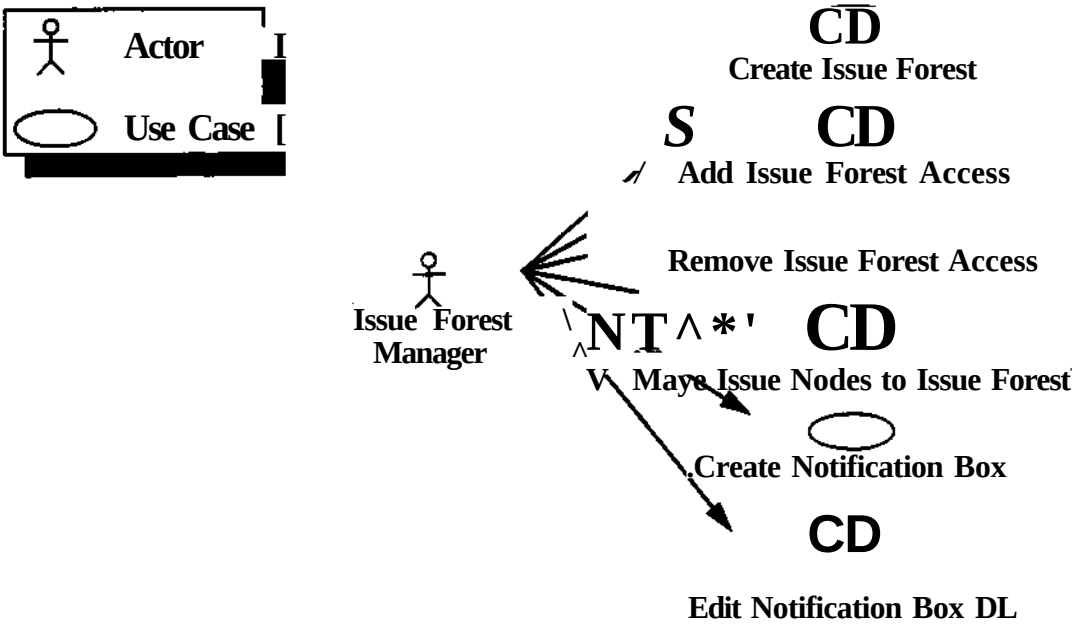


Figure 16. Issue Forest Manager - Use Cases Communication Associations

7.3.1 Create Issue Forest

Brief Description: The Create Issue Forest use case is used by an Issue Forest

Manager to create new Issue Forests within an Issue Forest he/she is managing.

Flow of Events: The Issue Forest Manager issues an "create Issue Forest" in an open Issue Forest. The Issue Forest Manager specifies the name of the new Issue Forest, a group of users who can post to it, and a group of users who can view it. The users specified in the access groups have to already have equivalent access to the parent Issue Forest.

IWEB adds the new Issue Forest to the selected Issue Forest, and creates a Notification Box for it. The creation of the new Issue Forest is broadcast under the form of a notice in the Notification Box of the parent Issue Forest. The use case ends when the Notice is available to the other IWEB Users.

7.3.2 Add Issue Forest Access for User

Brief Description: The Add Issue Forest Access for User use case is used by an Issue Forest Manager to enable an IWEB User to view or an IWEB Author to modify an Issue Forest. Before an IWEB User can be added to an Issue Forest, they first has to have access to the parent Issue Forest.

7.3.3 Remove Issue Forest Access

Brief Description: The Remove Issue Forest Access use case is used by an Issue Forest Manager to deny an IWEB User the view or modification of an Issue Forest and its containing Issue Forests.

7.3.4 Move Issue Forest Nodes To Issue Forest

Brief Description: The Move Issue Forest Nodes To Issue Forest use case is used by an Issue Forest Manager to move an aggregate of Issue Forest Nodes to another Issue Forest the Issue Forest Manager has access to.

Flow of Events: The Issue Forest Manager selects an issue or a set of issue nodes in an Issue Forest and issues a "move issue nodes" command. IWEB then selects all the Issue Forest Nodes reachable through Issue Forest links in the same Issue Forest. IWEB then prompts the user for a target Issue Forest. Once the Issue Forest Manager selects an Issue Forest he/she manages, IWEB moves all of the selected Issue Forest Nodes and Links to the new Issue Forest. Notices are posted by IWEB in the Notification Boxes associated with both Issue Forests. The use case ends when the Notices are available to the other IWEB Users.

7.3.5 Create Notification Box

Brief Description: The Create Notification Box use case is used by the Issue Forest Manager to create any additional Notification Boxes that are desired beyond those that are automatically created corresponding to Issue Forests. The Issue Forest manager also adds one or more IWEB Users to the distribution list for the box - any users whose names are added must have privileges of an Author in order to be able

to see the contents of the box.

7.3.6 Edit Notification Box Distribution List

Brief Description: The Edit Notification Box Distribution List use case is used by an Issue Forest Manager to add or remove IWEB Users from the distribution list of a Notification Box. If the Notification Box corresponds to an Issue Forest, the Notification Box Distribution List has to be a superset of the users who can access the Issue Forest.

7.4 Use Cases **for** IWEB Manager

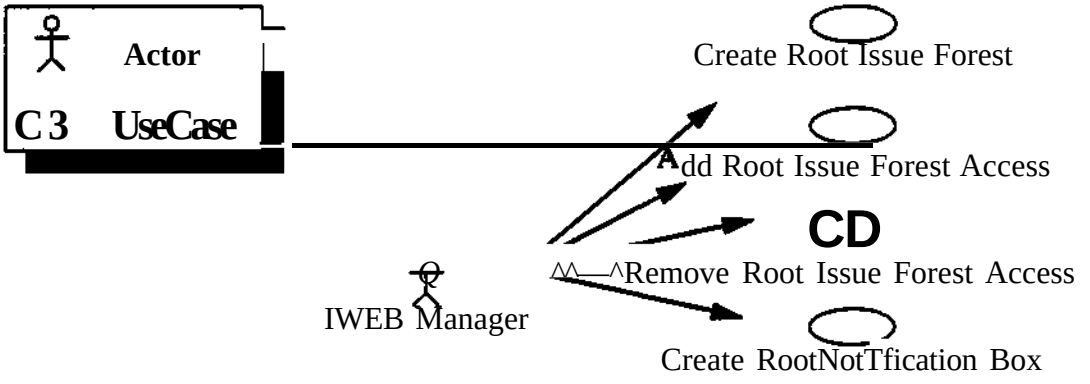


Figure 17. IWEB Manager - Use Cases Communication Associations

7.4.1 Create Root Issue Forest

Brief Description: The Create Root Issue Forest use case is used by an IWEB Manager to create a new Root Issue Forest.

Flow of Events: The IWEB Manager issues a "create root Issue Forest" command in the Issue Forest directory, specifies the name of the new Issue Forest, a group of users who can post issues on it, and a group of users who read issues in it. IWEB creates a new Issue Forest, adds its name to the Issue Forest directory, creates a Notification Box for it which is also added in the Notification Box directory. The creation of the new Issue Forest is broadcast under the form of a Notice posted in the project Notification Box. This use case ends when the Notice is available to the other IWEB Users.

7.4.2 Add Root Issue Forest Access for User

Brief Description: The Add Root Issue Forest Access for User use case is used by an IWEB Manager to enable an IWEB User to view and/or modify a root Issue Forest. In order for the IWEB User to be able to modify Issue Forests nested in the Root Issue Forest, the Issue Forest Manager of the corresponding Issue Forests (or the IWEB

Manager) has to add the new IWEB User to these Issue Forests as well.

7.4.3 Remove Root Issue Forest Access

Brief Description: The Remove Root Issue Forest Access is used by a IWEB Manager to deny an IWEB User the view or modification of a Root Issue Forest (and all of its containing Issue Forests).

7.4.4 Create Root Notification Box

Brief Description: The Create Root Notification Box use case is used by the IWEB Manager to create any additional Notification Boxes at the global level that are desired beyond those that are automatically created corresponding to Root Issue Forests. The IWEB Manager also adds one or more IWEB Users to the distribution list for the box - any users whose names are added must have privileges of an Author in order to be able to see the contents of the box.

7.5 Use Cases for Issue Forest Liaison

Dual Issue Forest Manipulation Use Cases

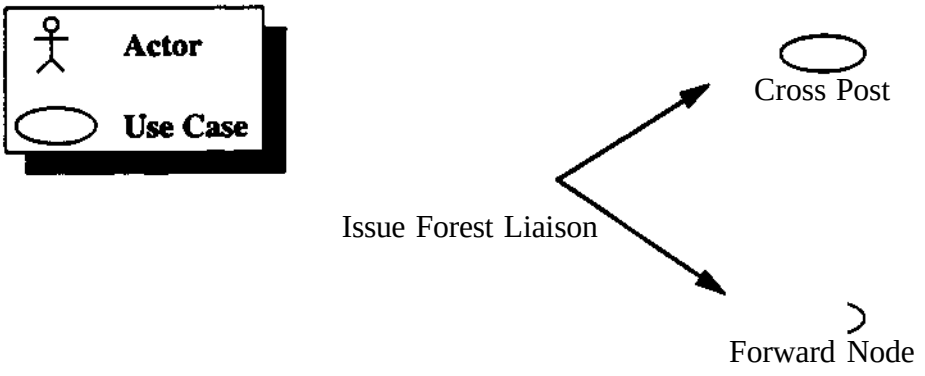


Figure 18. Issue Forest Liaison -Dual Issue Forest Communication Associations

7.5.1 Cross Post

Brief Description: The Cross Post use case is used by the Issue Forest Liaison actor to post Issue Forest Nodes on two different Issue Forests at the same time. An Issue Forest Liaison will typically use this use case to post issues on different Issue Forests which are not accessible by the same set of users.

7.5.2 Forward Node

Brief Description: The Forward Node use case is used by the Issue Forest Liaison to add existing Issue Forest Nodes to a different Issue Forest. An Issue Forest Liaison will typically use this use case to post existing issues on an Issue Forest which can be accessed by different users, thus broadening the scope of the

discussion for that specific issue.

7.6 Use Cases **for** Project Manager

Edit Organizational Model Use Cases

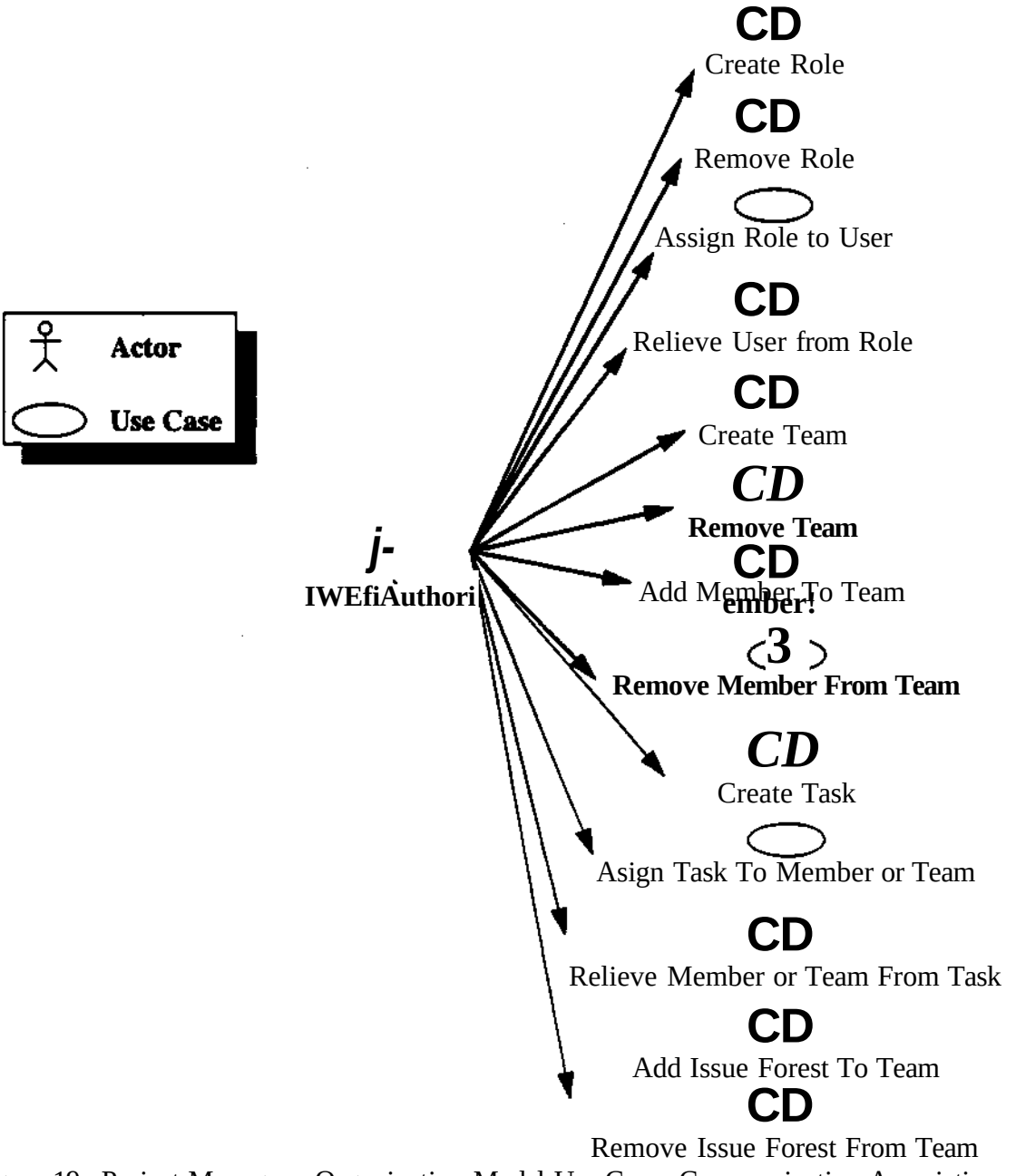


Figure 19. Project Manager - Organization Model Use Cases Communication Associations

7.6.1 Create Role

Brief Description: The Create Role use case is used by a Project Manager to create new role models in IWEB. Roles created in this use case can then be used in the

Assign Role to User use case.

7.6.2 Remove Role

Brief Description: The Remove Role use case is used by a Project Manager to remove a role model in IWEB. A role can be removed only if no more users are assigned this role.

7.63 Assign Role to User

Brief Description: The Assign Role to User use case is used by a Project Manager to assign a role to an IWEB User. A role can be assigned to more than one user.

7.6.4 Relieve User from **Role**

Brief Description: The Relieve User from Role use case is used by an Project Manager to remove a role from a user's model.

7.6.5 Create Team

Brief Description: The Create Team use case is used by a Project Manager to create a new Team model and add it to the Organizational Model.

7.6.6 Remove Team

Brief Description: The Remove Team use case is used by a Project Manager to remove a Team model from the Organizational Model.

7.6.7 **Add** Member To Team

Brief Description: The Add Member To Team use case is used by a Project Manager to add a User Model to a Team Model. A User Model may be included in more than one Team Model.

7.6.8 Remove Member From Team

Brief Description: The Remove Member from Team use case is used by a Project Manager to remove a User Model from a Team Model. Removing a User Model from a Team Model does not destroy or otherwise affect the selected User Model.

7.6.9 Create Task

Brief Description: The Create Task use case is used by a Project Manager to create a new Task Model.

7.6.10 Assign Task To Team Or User

Brief Description: The Assign Task To Team Or User is used by a Project Manager to add a Task to a Team or User Model.

7.6.11 Relieve Team Or User From Task

Brief Description: The Relieve Team Or User From Task is used by a Project

Manager to remove a Task from a Team or User Model.

7.6.12 Add Issue Forest To Team

Brief Description: T.. : Add Issue Forest To Team use case is used by a Project Manager to add an Issue Forest to a Team **Model.**

7.6.13 Remove Issue Forest From Team

Brief Description: The Remove Issue Forest From Team use case is used by a **Project Manager** to **remove an Issue Forest from a Team Model. The Issue Forest is not destroyed** or otherwise **affected.**

7.7 Use Cases for System Administrator

Edit User Configuration Use Cases

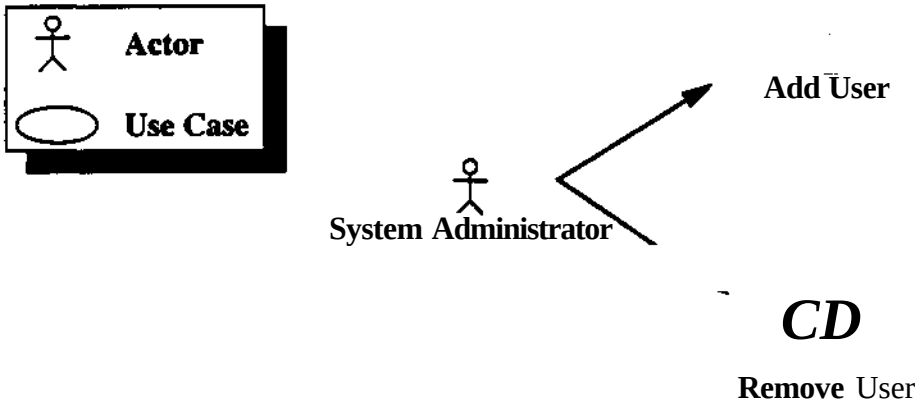


Figure 20. System Administrator - Configuration Use Cases Communication Associations

7.7.1 Add User

Brief Description: The Add User use case is used by a System Administrator to add a user to the system.

7.7.2 Remove User

Brief Description: The Remove User use case is used by a System Administrator

to remove a user from the system.

Data Management Use Cases

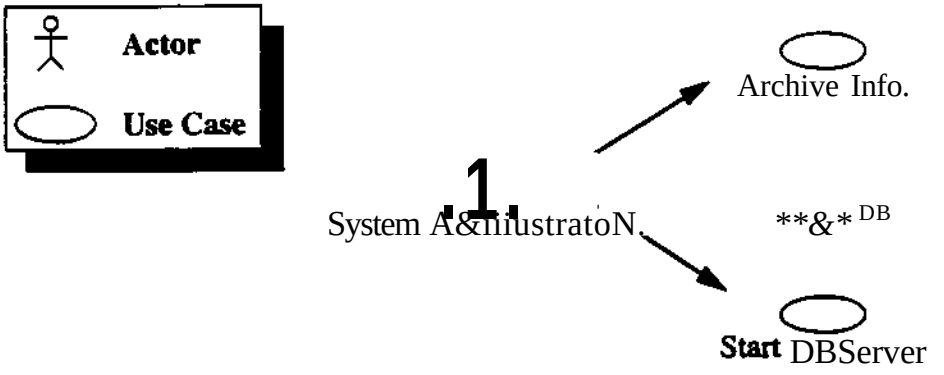


Figure 21. System Administrator - Data Management *Use* Cases Communication Associations

7.7.3 Archive Information

Brief Description: The Archive Information use case is used by the System Administrator to remove seldom accessed nodes from the active Issue Forest and place them into a more stable (but not easily accessible) storage medium.

7.7.4 Repair Database

Brief Description: The Repair Database use case is used by the System Administrator to repair inconsistencies resulting from crashes or errors caused by IWEB.

7.7.5 Start Database Server

Brief Description: The Stan Database Server use case is used by the System Administrator to restart a database server which crashed.

7.8 Use Cases **for IWEB** System

Once IWEB has been installed and started, it will continuously manage the Issue Forests and other models created by the user. Even when no users are logged in, IWEB can initiate use cases and do work (e.g. build terminology dictionaries, restart crashed database servers, etc.).

7.8.1 Create Notification Box Automatically

Brief Description: The Create Notification Box Automatically use case is used by the IWEB System to create corresponding Notification Boxes whenever an Issue Forest is created. The distribution list of the Notification Box will have the same persons as associated with the IF that spawned its creation - for instance, the team members associated with a team that has an IF will be members of the distribution list for the corresponding Notification Box.

8 Visual Presentation of the Use Cases

This section presents conceptual sketches for the user interface of IWEB. Given that use cases are a functional description of the system and do not include any reference to any particular user interface, the purpose of this section is to give the reader a more concrete view of how the user would interact with IWEB.

At the time of publication of this report, a prototype of IWEB was used to reconstruct the information base of the JEWEL project (Software Engineering classes of Fall'94 and Spring'95). We included screen dumps from that reconstruction for the purpose of comparison with the initial requirements sketches.

Section 8.1 includes the requirements sketches. Section 8.1.1 describes the main windows of IWEB and their relationship. Section 8.1.2 presents the windows associated with browsing and searching the Issue Forests. Section 8.1.3 presents the Entry Port, which is the first window displayed when the user starts IWEB. Section 8.1.4 presents the windows associated with the What's New functionality.

Section 8.2 includes screen dumps taken from the JEWEL information base. Section 8.2.1 presents a typical IWEB screen as seen by a user. Section 8.2.2 presents a view of the Project Model. Section 8.2.3 presents a view of the Team Model. Section 8.2.4 shows the Issue Forest Viewer displaying a team Issue Forest. Section 8.2.5 shows the Notification Box viewer displaying a team's Notification Box. It is to be noted that Project and Team Models are *n-dim* models. At the time of publication of this report, none of the windows associated with the What's New functionality was implemented.

8.1 Conceptual Sketches

8.1.1 General Vision for a screen in IWEB

A general vision for the display of the IWEB screen is shown in Figure 22. As currently defined, the Issue Forest and Notification Boxes are the primary tools offered by the system. It is hoped that visual space can be managed in such a way as to allow the display of as much information as possible. We hope to make available through the user interface all the information the user knows he wants access to, while keeping other information organized and easily accessible.

Issue Nodes and Notices are displayed in such a way to reflect whether the user has ever viewed them or not. For example, an Issue Node will initially appear in black and white colors when it is initially posted and then grayed out once the user views the node. The same Issue Forest Node may appear in many places at any one time (e.g. in an Issue Forest, in a Notice and in the What's New Window). The user may issue node viewing commands from any of these places. Once the node is viewed, the

representation of the node in each place will be grayed out.

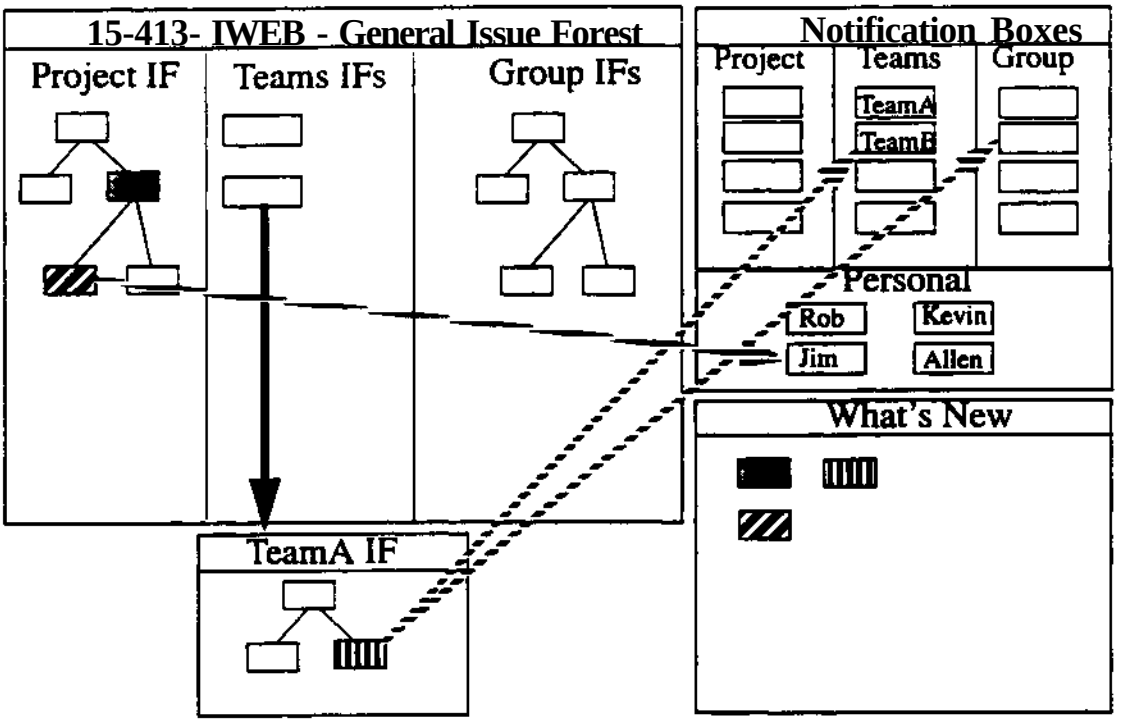


Figure 22. IWEB Screen

8.1.2 Issue Forest Windows

There are two main alternative windowing strategies for the Issue Forest Portion of the IWEB. The strategy presented in Figure 23. shows the use of multiple windows, one window for each issue forest (or sub-forest). This has the benefit of helping to compartmentalize information, but may lead to a cluttered screen. Figure 24. shows the use of a single window for all issue forests. In this case, the user would use scroll bars to move among sub-forests that exist within each main Issue Forest, for example UI, Comm., and so on as shown in the figure.

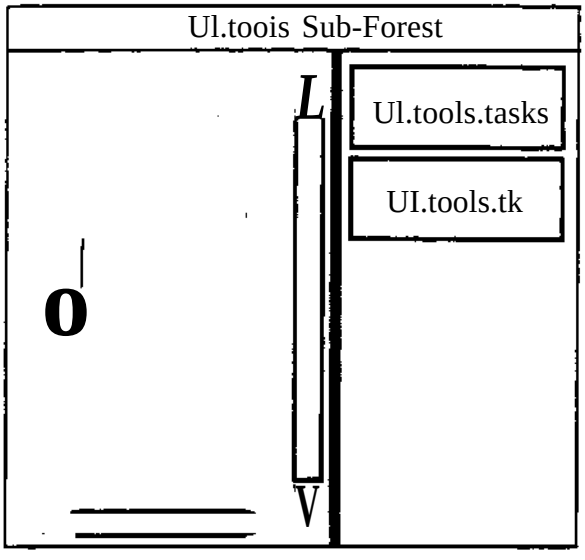


Figure 23. Recursive Forest Concept

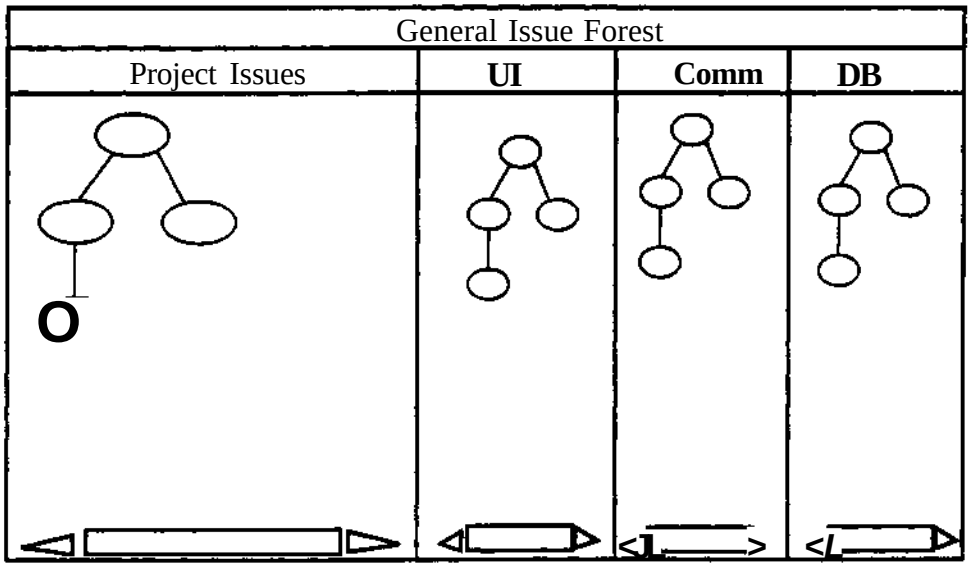


Figure 24. Flat Space Concept

8.1.3 Entry Port

The entry port of IWEB is the first window which appears on the screen when the user starts IWEB. The entry port gives the user access to global functionality such as the issue forest directory, the notification box directory or the what's new window.

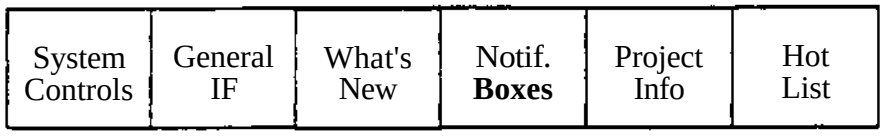


Figure 25. IWEB Entry Port

8.1.4 What's New

What's New Window

The What's New window displays all the Issue Forest Nodes which have been posted in a given Root Issue Forest or any of its sub Issue Forest since the last time the user updated or opened the What's New window. The What's New window also provides access to functionality to filter the content of a What's New window, display its history or update its content.

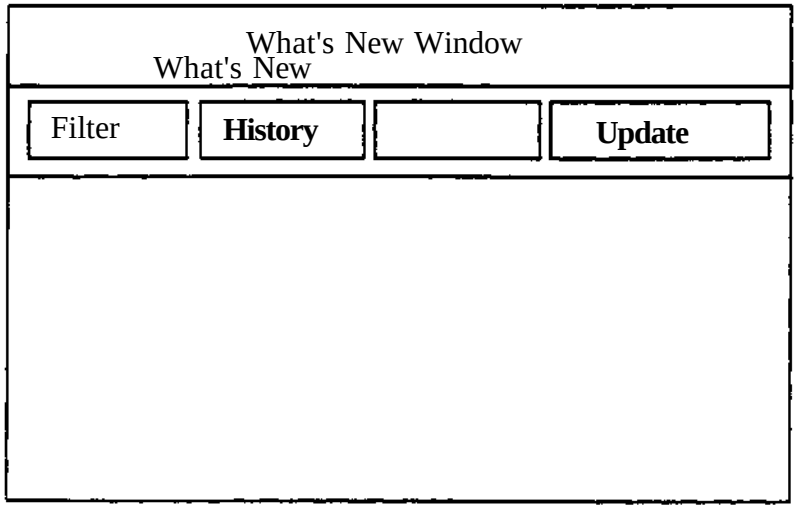


Figure 26. What's New Window

Filter What's New (or Notification Box) Window

The Filter window is used by an IWEB User to specify new criteria for filtering a What's New window or a Notification Box window. The user can either apply the filter once or keep the filter as the default filter for the given window.

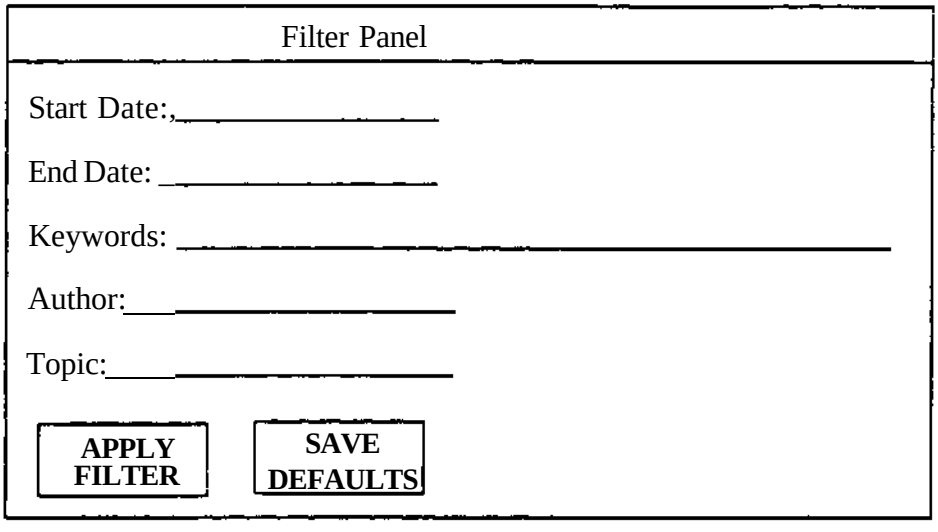


Figure 27. Filter Panel

View What's New (or Notification Box) History Window

The History window is used by an IWEB User to view the state of the What's New window or a notification box as of a previous date. The History window lists all the times at which the What's New window was either opened or updated.

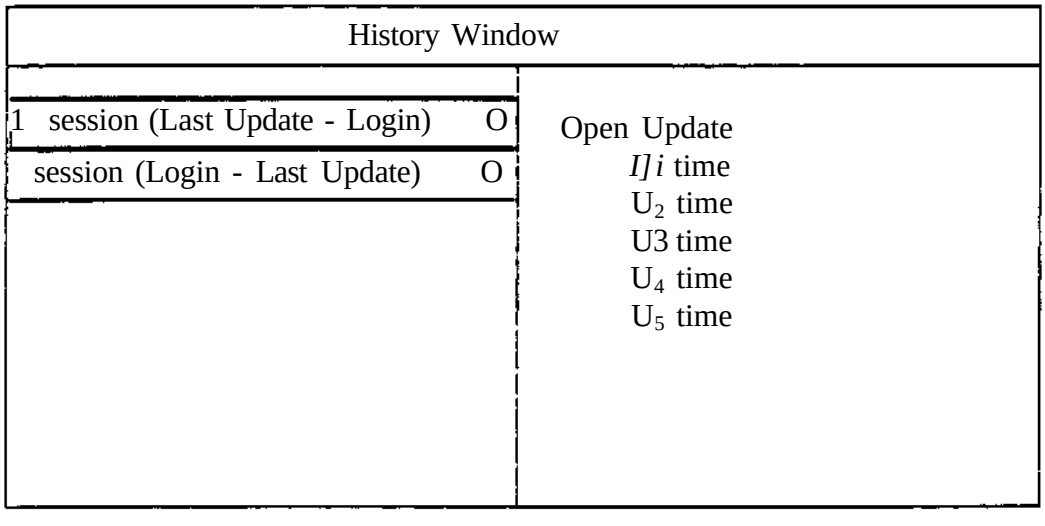


Figure 28. History Window

8.2 IWEB Screen Dumps

8.2.1 Overview

Figure 29. presents a typical IWEB screen. The window on the left hand side is an Issue Forest Viewer displaying the Issue Forest for the development team. On the right, a Notification Box Viewer displays the Notification Box associated with that Issue Forest. Various pans of both viewers are further detailed in Section 8.2.4 and in Section 8.2.5.

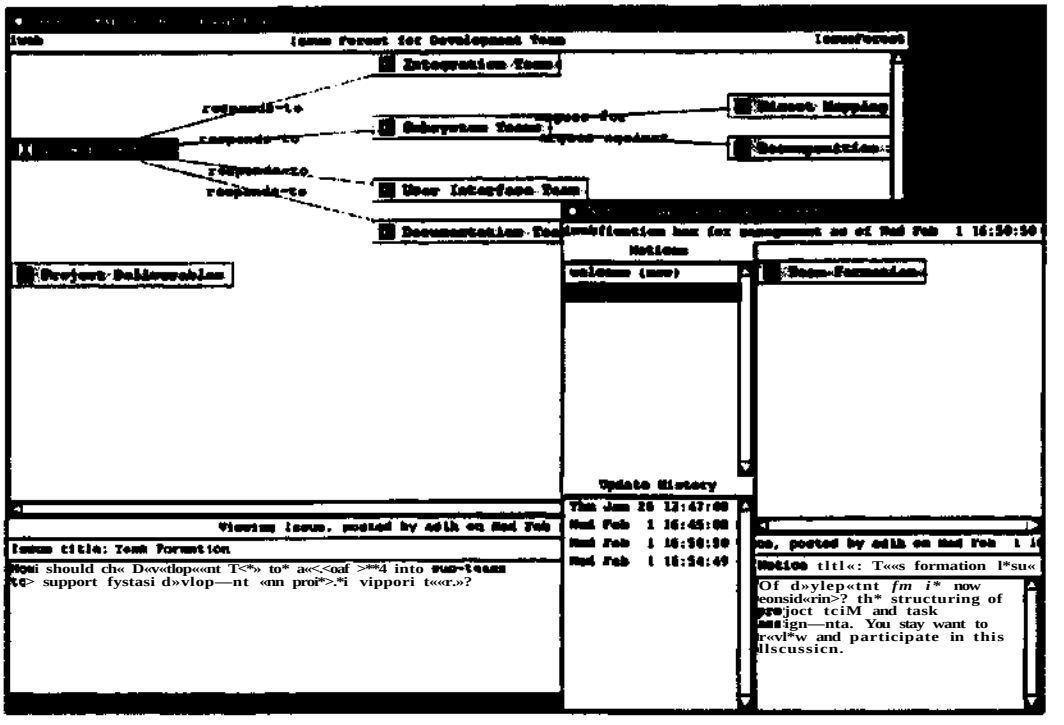


Figure 29. Typical IWEB Screen

8.2.2 Project Model

Figure 30. shows the Project Model for the JEWEL project. A Project Model is an rt-dim model which typically refers to (including but not limited to) models referring to, respectively, all the Issue Forests, Member models, Notification Boxes and Team models associated with the project. In the case of the JEWEL project, the Project Model also contains a reference to the WWW page documenting the project.

Figure 30. Project Model and WWW Page for JEWEL

8.2.3 Team Model

Figure 31. displays the Team Model for the Emission team of the JEWEL project. A Team Model is an n-dim model which typically refers to (including but not limited to) models referring to, respectively, all the Member models of the team, ail the meeting minutes and agendas, all the Issue Forests associated with the team, and all of its Notification Boxes. In the case of the Emission team, the object models associated with the Emissions subsystem are also referred to by the Team Model. Figure 31. also pictures an open OMT model displayed by OMTool.

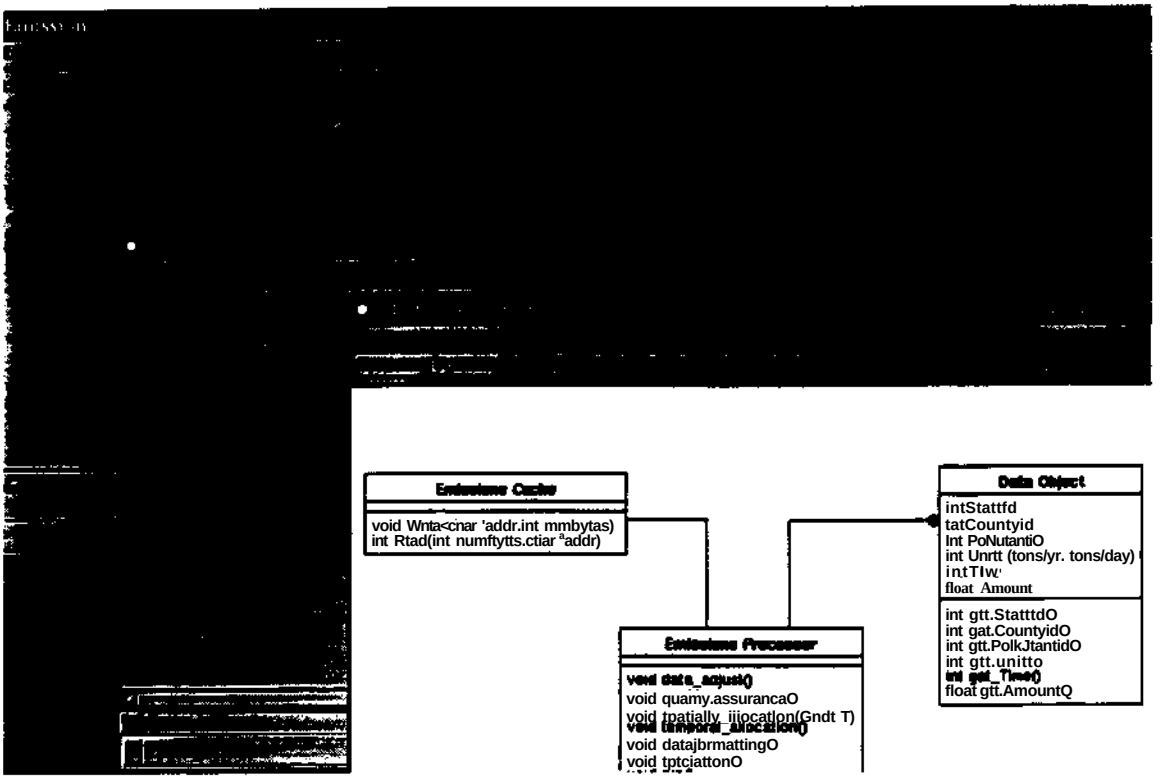


Figure 31. Team Model and OMT Model for Emission Team (JEWEL Project)

8.2.4 Issue Forest Viewer

The Issue Forest Viewer is the main access the user has to view an Issue Forest. Figure 32.pictures the Issue Forest for the Emission Team. The upper part of the Viewer displays the structure and title of the Issue Nodes. The bottom part displays the content of the selected node (such as its author, date of post, description and references). In this case a Proposal referring to the OMT model Emission is displayed.

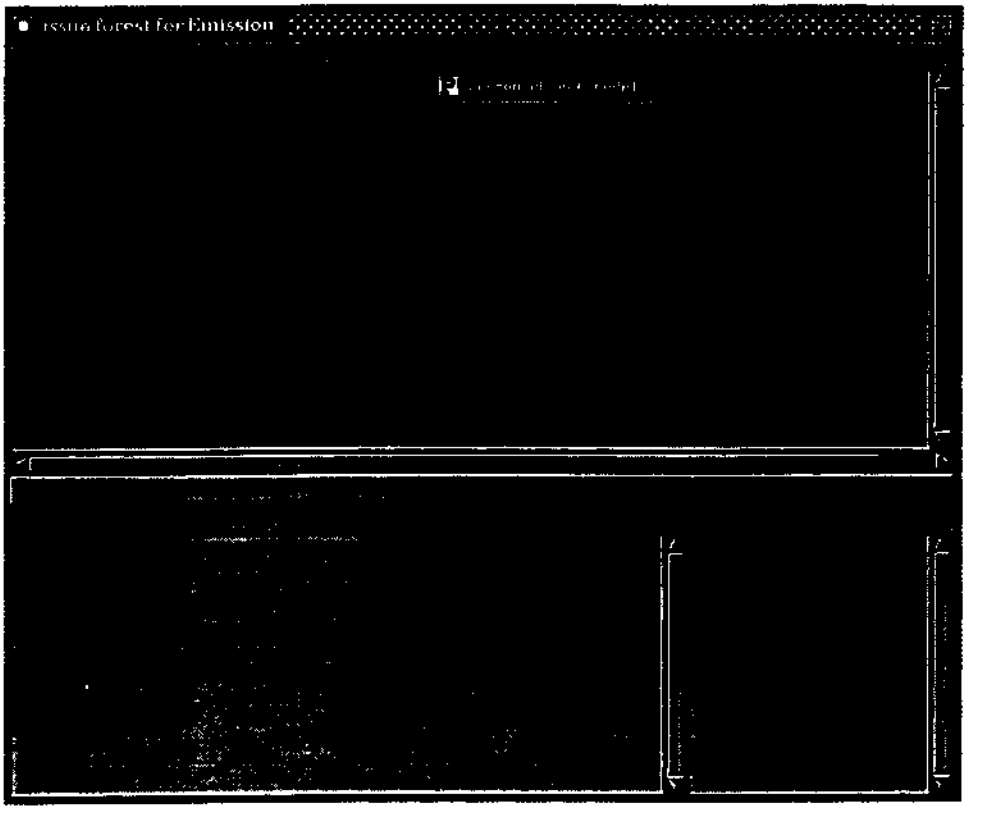


Figure 32. Issue Forest Viewer for Emission Team (JEWEL Project)

8.2.5 Notification Box Viewer

The Notification Box Viewer is the main access the user uses to view a Notification Box. Figure 33. displays the Notification Box of the Emission Team. The bottom left pan of the Viewer displays the dates and times the Notification Box was opened or updated by the user. The upper right part of the Viewer displays the Notices unread as of the time selected in the lower left. The top right part displays the nodes associated with the selected Notice. The bottom right part displays the content of the Notice, such as its author, its description and any associated references.

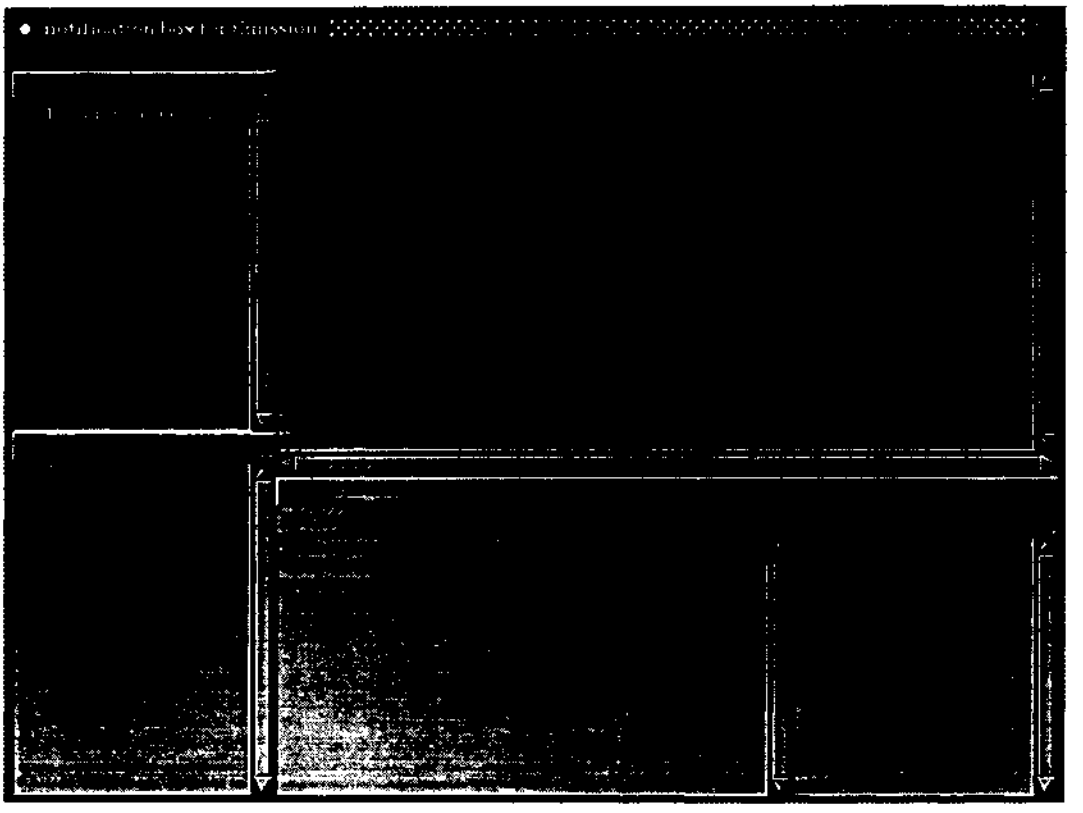


Figure 33. Notification Box Viewer for Emission Team (JEWEL Project)

9 Non-Functional Requirements

9.1 Student Acceptance

Our experience <' ith Objectory during Fall'93 gave us a deerer understanding of how non-functional issue . a factor in the acceptance c - ol. Until then, we considered Objectory as an effect! \ ^ tool for supponing **the** OOSE. nod, and had some successes with using it. **The** student survey and our **direct observations** indicated **that the students did** not **share that opinion. Although Objectory is relatively complete in functionality, the course environment (e.g. smaller machines, large number of users, non standard platform) in which it was installed degraded the performance of the tool, both in terms of speed and reliability. Some defects in the software which were not encountered during our previous experience became major bottlenecks in the use of the tool. Moreover, constraining students to use the tool (i.e. dictating a project standard) did not improve the students' opinion of the tool. Although a subset of students acknowledged the usefulness of the tool, most of them generated models for the sake of deliverables and not for the sake of documentation or inter-team coordination.**

In order for IWEB to be useful in the context of the course, its implementation will have to meet the standards set by other tools that the students use on a daily basis. We will have to take special care with availability and response time. Similarly, the user interface of the tool will have to meet usability standards set by tools with similar functionality.

Specifically, we will take the bboard and e-mail reader that students use (AMS messages) as a standard for **performance and reliability. IWEB start-up time should be less than the start-up time of messages. Its response time should be comparable. The default interface to an issue forest should be very similar to the folder interface of messages, although alternate, more n-dim-like interfaces may be provided as well.**

9.2 Organizational Assumptions

We do not assume that **IWEB** by itself will improve project performance in general or inter-team communication in particular. We assume that **the** organization using **IWEB** will have to adapt to the communication infrastructure in order to use it effectively. We foresee that at least one user per team should carry the role of information manager. His/her task would be to maintain the issue forests owned by a team (drop obsolete issues, create sub issue forests, update dictionaries, etc.). We also assume that the need for early integration and inter-team communication should be encouraged, through various assignments and incentives from the beginning of the project.

We also assume that the accessibility of the communication medium can significantly influence its use. We observe that students preferred to use personal e-mail to discuss some issues because the messages posted on the bboard could be accessed by everybody. For that purpose, IWEB will provide access control mechanisms for the students to enable or deny the access of some models to other users (including management).

10 Future Plans

As described in Section 2, an important part of our requirements and development process is early user evaluation. We are interested in evaluating and improving IWEB by observing its use in real life context. This section describes our short term and mid term plans for deploying and evaluating IWEB.

10.1 Test Case

In order to maximize the chance of user accepting the tool, we will first conduct an in-house test of IWEB. A subset of the n-dim group (6 developers and 3 non-developers) will initially use an alpha version of IWEB in the Fall of 94. We will use this test to refine the user interface, improve the speed and response time of the system and detect and solve any reliability problems.

While conducting this test, we will have to be aware that the n-dim group will be a more tolerant audience that the course students, given that they developed n-dim (and therefore have a deeper understanding of its fundamental concepts) and that they are seasoned developers.

10.2 Proposed Experiments

During the in-house tests of IWEB, as a transitional step, we will introduce the IWEB rhetorical model by using indented-text IWEB (itIWEB) in two courses: the advanced software engineering course of Spring *95 and the Navigator course of Fall 94-Spring '95. itIWEB is a low-tech textual version of IWEB containing a subset of links and nodes. It follows specific conventions that can be easily used on paper or whiteboards, as well as with text editors and document publishing tools (see Section 12.5).

IWEB, itIWEB, and student training, will be introduced during the INI Software Engineering project during the Summer 95 term. At that stage, we expect to be able to use the metrics part of IWEB developed during the Spring for early detection of problems and inter-team communication evaluation.

10.3 Metrics Collection and Postmortem Evaluation

In addition to evaluating similar metrics to those we used for the Fall'93 case study (see Section 3), we plan to collect metrics on the usage of IWEB. The data available for Fall'93 only allowed us to evaluate metrics on which information was sent, however, we have little idea of how much of that information reached its intended receiver. In IWEB, we plan to collect frequency and time information on the nodes accessed by each user in order to evaluate such metrics.

Given that IWEB will provide different user interfaces to the same models (i.e. bboard-like interface, gIBIS-like graph display) and multiple indices into the same information (notification boxes, what's new, issue forest browser), we will collect data on interface usage, in order to help us refine the user interface and evaluate the usefulness of different indices. We are also interested in the frequency of use of the issue forest dictionaries.

In order to assess the impact of IWEB on the overall process, we will measure whether the integration phase of the project starts earlier than in previous instances of the same course, and whether the on-line communication traffic increases. We will also measure the relative size of the respective teams' vocabulary and see if the size of their shared vocabulary increases earlier.

11 References

f 1] B. Bruegge and R. Coyne.¨ Model-Based Software Engineering in Larger Scale Project Courses," Proceedings of the IFIP Working Conference on Software Engineering Education, Hong Kong (September 1993).

[2] B. Bruegge and R. Coyne. ' Teaching Iterative and Collaborative Design: Lessons and Directions," Software Engineenng Education: Lecture Notes in Computer Science, J. L. Diaz-Herrera (ed.), 7th SEI Conference on Software Engineering, San Antonio, Texas (Janurary 1994).

[3] L. Bucciarelli, "An ethnographic perspective on engineering design", Design Studies, Vol 9, p. 160, 1988.

[4] K. Clark and T. Fujimoto. *Product Development Performance.* Harvard Business Press, Cambridge, MA, 1991.

[S] COMIC Project Deliverable. Parts 1.1 and 3.1, Computing Department, Lancaster University, Lancaster U.K, 1993.

[6] J. Conklin and M. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," ACM Transactions on Office Information Systems, vol. 6, no. 4 (October 1988): pp. 303 - 331.

[7] L. Constantine, 'Teamwork Paradigms and the Structured Open Team," Proceedings: Embedded Systems Conference. San Francisco (1989).

[8] R. Coyne et al.. "Requirements Analysis for n-dim," (work in progress) Engineering Design Research Center, CMU (1994).

[9] R. Coyne, A. Dutoit, B. Bruegge and D. Rothenberger., 'Teaching More Comprehensive Model-Based Software Engineering: Experience With Objectory's Use Case Approach," Proceedings of the Conference on Software Engineering Education, New Orleans (Janurary 199S). Also available as Tech. Report, EDRC-05-86-94, Engineering Design Research Center, CMU (1994).

[10]R. Coyne and M. Ehrenberger, "Information Modeling for Software Engineering: An Illustrative Project History," Tech. Report, EDRC-05-88-94, Engineering Design Research Center, CMU (1994).

[11]J.C. Ferrans et al., "Hyperweb: A Framework for Hypermedia-Based Environments," Software Engineering Notes, vol. 17 (December 1992): pp. 1-10.

[12] S. Finger, E. Gardner, and E. Subrahmanian, "Design Support Systems for Concurrent Engineering: A Case Study in Large Power Transformers Design," Proceedings of the International Conference on Engineering Design. Heuritsa. Zurich (1993): pp. 1433-1440.

[13] F. Floyd, F. Feisin and G. Schmidt, "STEPS to Software Development with Users," 2nd European Software Engineering Conference, Warwick, Coventry, UK, (1989): pp. 48-64.

[14]K. Gronbaek. M. Kyng, and P. Mogensen. "CSCW Challenges: Cooperative Design in Engineering Projects," Communications of the ACM. vol. 36,no. 6 (1993): pp. 67-77.

[15] I. Jacobson et al.. *Object-Oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley Publishing Company. NY. 1992.

[16] J. Jacquot, J. Guyard. and L. Boidot. "Modeling Teamwork in an Academic Environment," Software Engineering Education: Lecture Notes in Computer Science, J.E. Tomayko (ed.) (1990): pp. 110-122.

(17] S. Konda et al., "Shared Memory in Design: A Unifying Theme for Research and Practice, " Research in Engineenng Design, vol. 4 (1992): pp 23-42.

(18]S. Konda and I. Monarch. "An Interactive Computational Approach for Building a Software Risk Taxonomy," 3rd Annual Conference on Software Risk. Pittsburgh (April 1994).

[19] W. Kunz and H. Rittel. "Issues as Elements of Information Systems," Working Paper No. 131, Inst. of Urban and Regional Development, University of California. Berkeley (1980).

[20] S. Levy et al., "An Overview of the n-dim Environment," Tech. Report. EDRC-05-65-93, Engineering Design Research Center, CMU (1993).

[21] S. Levy, E. Gardener, and E. Subrahmanian, "The n-dim White Papers," Tech. Report, EDRC-xx-94 (in progress). Engineering Design Research Center, CMU (1994).

[22] P. Naur, *Computing: A Human Activity.* ACM Press, Addison-Wesley, NY. 1992.

[231Y. Reich et aL "New Roles tor Machine Learning in Design,'* Artificial Intelligence in Engineering, vol. 8 (1993): pp. 165-181.

(24] M. Rettig, The Practical Programmer: Software Teams." Communications of the ACM. vol. 33 (October 1990).

(251J- Robertson. E. Subrahmanian. M. Thomas and A. Westerberg, "Management of the design process: the impact of information modeling." In Proceedings of FOCAPD, Snowmass. CO, July 10-15, 1994.

[26] J. Rumbaugh, et al.. *Object Modeling and Design,* Prentice-Hall, NJ, 1991.

[27] W. Scacchi. "Managing Software Engineering Projects: A Social Analysis." IEEE Transactions on Software Engineering, vol. 10. (January 1984): pp. 45-59.

[28] K. Schmidt. "Modes and Mechanisms of Interaction in Cooperative Work," Computational Mechanisms of Interaction for CSCW, C. Simone and K. Schmidt (ed.). Computing Department, Lancaster University, 1993, pp. 21-104. [COMIC Deliverable 3.1].

[29] Ben Schneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction,* Addison-Wesley Publishing Company, NY, 1992: pp. 474.

[30] D. Siewiorek et al. "An Interdisciplinary Concurrent Design Methodology as Applied to The Navigator Wearable Computer System," Journal of Computer and Software Engineering, vol. 2, no. 2 (1993).

[31] A. Smailagic and D. Siewiorek. "A Case Study in Embedded Systems Design: The Vuman 2 Wearable Computer," IEEE Design and Test of Computers, vol. 10, no. 4 (1993).

[32] E. Subrahmanian, S. Finger, and T. Mitchell, "A Report on the NSF-EDRC Study of Product Design Processes in Selected Japanese Companies," Tech. Report. EDRC-05-75-93, Engineering Design Research Center, CMU (1993).

[33] D. Sonnenwald, "Constested Collaboration: Case Studies in Design," Ph.D. Thesis, Department of Sociology, Rutgers University (1993).

[34] E. Subrahmanian et al.. "Computational Support for Shared Memory in Design," Automation-Based Creative Design: Issues in Computers and Architectures. I. White and A. Tzonis (ed). Elsevier (1993).

[35] E. Subrahmanian et al.. "Support System for Different-Time Different Place Collaboration for Concurrent Engineering," Workshop on Enabling Technologies In Concurrent Engineering, West Virginia (1993).

[36] G. Toye et al., "SHARE: a Methodology and Environment for Collaborative Product Development,** Tech. Report 0420. Center tor Design Research, Stanford University (1993).

[37] D. Wilkins et al., "CINERG: a Design Discovery Experiment," Preprints of the NSF Engineering Design Conference, College or Engineering, University of Massachusetts. Amherst (June 1989).

[38] B. Yakemovic and J. Conklin. 'Report on a Development Project *Use* of an Issue-based Information System." Proceedings or the Conference on Computer-Supported Cooperative Work. Los Angeles (October 1990): pp. 105-118.

12 Appendix

This appendix provides more examples of scenarios developed during the requirements elicitation phase. Section 12.1 describes the conventions used to describe conventions for writing scenarios. Section 12.2 describes general tasks which are engaged in throughout a project (typically on a daily basís). Section 12.3 describes requirements tasks, generally occuring at the beginning of the project. Section 12.4 describes organizational tasks.

12.1 Conventions

Scenarios presented here are composed of two sequences of events. The first one described how a given task is currently accomplished (labeled O. for old way). The second one descirbes how the same task would be accomplished using IWEB (labeled N. for new way). Each scenario is also labeled with actual times to illustrate the typical duration of sequences of events.

12.2 General tasks

· Before lecture: stops at cluster to make a quick check of the IWEB for new announcements, issues, communications (Doesn't consult any in depth, just notes there existence for later.

9:00-10:20 Attends lecture and takes quiz:

O. Waits for quiz to come back. Answers are not generally returned.

N. Is reminded that answers to the quiz questions are always posted and available in IWEB by later in the day.

10:25-10:30

O. A subset of the group realize that they need to have a meeting to discuss some aspect of the system. They agree to select a time and try to get the rest of the group to attend. The meeting time and place is posted on the group's bboard.

N. After lecture, he, along with a couple of other team members decide to call for a special team meeting (other than their normally scheduled weekly meeting. He volunteers to send an IWEB message to the other team members and they agree that everyone will check the IWEB a couple of times through the day to see the status of the meeting. They agree that by 4:00pm the meeting will have either been confirmed or postponed.

3:00-3:10

O. Read bboard, check meeting messages.

N. Checks IWEB for message(s) about the meeting. Meeting is confirmed for 7:30pm.

4:30-5:00

O. Reads bboards, posts some internal messages. If needed, reads another group's bboard and discuss board to carry on inter-group arguments. Is annoyed by duplicate posts, inability to find old

messages, and textual drawings.

O. Posts a message saying that an Objectory Model exists relating to a message that was posted. Gives a description of how to find the model. Becomes concerned no one will ever look at the Model because Objectory is so cumbersome to start, further if they do start it, they will have to navigate to the appropriate use case/ model to check the information.

N. Reads the IWEB issue-base; posts some sub-issues and responses; notices that a new task assignment has been proposed for her team.

N. Brings up Objectory from within IWEB and does some modeling, then annotates a new version of a specific Objectory model and links it to a discussion node in the issue base.

7:30-8:30 Attends the special team meeting

O. At the meeting takes action items to be discussed with other teams. Meeting minutes are taken on a Duo, and then later posted to the team bboard. Further, those minutes are later copied by the Tech Writer into a Frame document for hypertext access that is never used because Frame is cumbersome to start and difficult to learn to use effectively.

N. At the meeting the team decides to "open a dialogue" within IWEB with another team regarding a particular object model.

8:30-8:45

O. See above

N. After the meeting, he updates the team's project meeting models in the IWEB with the minutes, notes and action items from the meeting. (Some of these he scans in from paper, others were taken on-line at the meeting and are merely linked in.)

12.3 Requirement tasks

9:00 PM

O. Unable to remember which use cases their group is responsible for (and what a use case is in the first place), students search bulletin boards, send each other zephyrs, and possibly go back to their dorm room in search of the appropriate use cases, and then the directory to find Objectory and the files in.

N. Unable to remember which use cases their group is responsible for, students check the IWEB for the slides from latest class where they were assigned. Alternatively, they select their own name from the IWEB and follow the link to the group they belong t Once at the group, the select the use cases node, which starts Objectory for them and loads the use files into the tool.

9:15

O. A student brings up Objectory and begins to edit one of the use cases. After editing, the student posts a message to the bboard saying that the use case has been edited, but that there is a question about its content. The student copies and pastes a section of the use case into the message to illustrate what the problem is.

N. A student starts IWEB and begins to edit one of the use cases for his group by clicking on the use case in IWEB which starts Objectory. After editing the use case, he enters a discussion node for his group. The node is linked to the use case so that other students can simply call the entire use case up with one click.

9:30

0. A student realizes that it is unclear whether his or another team should really be responsible for a use case.

0. The students posts to both teams[1] bulletin boards asking people to begin a discussion about the problem and resolve it.

N. The student creates a new issue node in the issue base and details the problem there. He attaches the use case in question to the node. Further .he issue node is attached to the groups announce nodes to ind^ ..te that each member of the group should begin to track this issue.

9:45 A student begins working on an interaction diagram and is unable to remember the proper commands for interacting with the tool.

0. The student pulls out the manual and begins to read the examples.

N. The student selects the Objectory Info Node and then Interaction Diagrams. The manual documentation appears on scree The student also brings up an animated example of how to create/edit Interaction Diagrams that was created by a TA earlier in the semester.

10:00 The student realizes that there is a new requirement but is unsure how to specify the requirement.

0. The student sends zephyrs to the a teammate who replies. After a series of zephyr messages, the new requirement is formalized and entered into Objectory.

N. The student opens a tracked zephyr from inside IWEB. After a series of zephyr message, the new requirement is formalized and entered into Objectory. The student then edits out the extraneous messages from the tracked zephyr and attaches these to the new requirement.

11:00 A Manager questions the new requirement.

O. The student receives a zephyr/email from a manager stating that they do not agree with the new requirement. The student and the manager spend 25 minutes as the student attempts to reconstruct the previous zephyr session Unfortunately, the student forgets a key idea that drove the new requirement (hey, it's late).

Eventually, the manager disconnects still unclear on why the new requirement is there.

N. The student receives a zephyr/email and directs the manager to the IWEB detailed zephyr node which recreates the conversation which was earlier created by the two students.The manager sees the key argument upon which the change hangs, smiles, disconnects, and goes home to a restful night of slumber sure that all is well is 15-413 land.

12.4 Organizational Tasks

Problem: New Temporary group needed to resolve intra-group subsystem problem. (Such as the query manager). Later, a team member from another group joins this new subgroup due to some cross team dependencies.

O. The students split off and form a subgroup, but all there message traffic regarding the subsystem is still sent to the main team bboard. When the new team member comes into the group, messages are duplicated and sent to both teams bboards. When the group reaches a conclusion, a **message is posted to both team bboards. Members** of both **teams skip the new message because they believe that it is intended just for the small subgroup.**

N. The students create a transient group in the IWEB. A sub-forest is created in the issue forest to handle discussion about the groups tasks and ideas. The other team members are not disrupted by the new group's activities. When the new team member joins from the other group, **she** is **merely added to the new transient group. When the group comes to a conclusion, dissemination nodes are added** and placed in **the Group mailboxes** of **both** the involved original groups. The **member takes note** of **the dissemination node and** read the announcement.

Problem: Management wants to know the status of the tool tasks and how the tasks are progression.

O. Management must search through each bulletin board and look for messages which discuss the tool task. When they do not find any, they each ask their group during the next meeting. The groups deliver status (incompletely) and the managers spend time at the next meeting discussing it.

N. Management logs into the IWEB and finds the root nodes of all issue forests which discuss tool tasks. Alternatively, they have the what's new window list new nodes from a certain date which contain the keywords "tool task".

12.5 itIWEB template

When and Where	Role
Date:	Primary Facilitator:
Start:	Secondary Facilitators:
End:	Timekeeper:
Building:	Minute Taker:
Room:	Scribe:

1. Agenda [Allocated Time:]

This section should contain a list of issues which are to be discussed during the meeting. They should be structured following the itIWEB method outlined in the discussion section below. Issues listed here may also include proposals and arguments from previous meetings or from previous bboard discussions.

2. Status (information sharing) [Allocated Time:]

This section should contain the action items of the previous meeting. The syntax for action items is given in section 4.

As the meeting progresses, these action items should be completed with the information provided by the persons who were assigned the action items. The additional information should be structured in issues, proposals, arguments and resolutions. Any new action item generated in the status report discussion, and any incomplete action item in this section should be added to the action item section of this meeting and referred to from a {ref} paragraph of the corresponding issue, proposal, argument or resolution.

3. Discussion (issue-based information processing) [Allocated Time:]

This section should include any new or continuing issues, proposals, arguments and resolutions; this includes the issues listed in the agenda. Action items generated during this discussion should be added to the action item section and referred to from the (ref} section of the relevant issue/proposal/argument/ resolution.

The following paragraphs represent an indented text information web (itIWEB). itIWEB is an issue-based method for structuring discussions. The order in which the paragraphs appear is very important, especially the Resolution Paragraphs and sub-Resolution Paragraphs.

Description of the itIWEB method: paragraph types and indentation conventions.

I: [1] The Issue Paragraph contains text that describes an issue that needs a resolution. Typically, it is created with the intention that it will be resolved, that is have an asso-

ciated Resolution Paragraph. An active Issue Paragraph is an issue that does not yet have an associated resolution.

{ref}: The Reference Paragraph is a place holder for information. It may reference another itIWEB Paragraph, including action items at the end of the minutes, or external information such as a file location or itIWEB paragraphs in other meeting minutes. They may be left empty or removed if not needed. Reference Paragraphs can follow any itIWEB Paragraph and it is assumed the information they contain is directly associated with the preceding itIWEB Paragraph (this Reference Paragraph is associated with the Issue Paragraph above).

P: The Proposal Paragraph contains text that describes a possible proposal intended to resolve the issue above. There may be several Proposal Paragraphs for each Issue Paragraph.

{ref}: (this Reference Paragraph is associated with the Proposal Paragraph above)

R: The Resolution Paragraph contains text describing how an issue will be resolved. If the Resolution Paragraph is placed directly after a Proposal Paragraph, as is the case here, the resolution is assumed to be based-on that proposal. If a resolution is not based on one of the listed proposals, the Resolution Paragraph should follow the Issue Paragraph (as an example, see the position of the next Resolution Paragraph).

{ref}:

+A: The Argue-for Paragraph contains text that states an argument in favor of a proposal.

{ref}:

-A: The Argue-against Paragraph contains text that states an argument against a proposal.

(ref}:

si: The sub-Issue Paragraph contains text that more specifically describes some aspect of the previous Issue Paragraph. Typically, it is created with the intention that it will be resolved, that is have an associated sub-Resolution Paragraph. An active sub-Issue Paragraph is a sub-issue that does not yet have an associated sub-resolution.

Iref}:

sP: The sub-Proposal Paragraph contains text that describes a possible proposal that is intended to resolve a sub-issue.

{ref}:

sR: The sub-Resolution Paragraph contains text describing how a sub-issue will be resoived. If the sub-Resolution Paragraph is placed directly after a sub-Proposal Paragraph (as is the case here), the sub-resolution is assumed to be based-on that sub-proposal. If a sub-resolution is not

based on a listed sub-proposal, the sub-Resolution Paragraph should follow a sub-Issue Paragraph. Resolution of all of the sub-Issues of an Issue does not imply that the Issue is necessarily resolved, however, the Resolution to the Issue must include, subsume or override any sub-Issue Resolutions.

{ref}:

+sA: The sub-Argue-for Paragraph contains text that states an argument in favor of a sub-proposal,

{ref}:

-sA: The sub-Argue-against Paragraph contains text that states an argument against a sub-proposal,

{ref}:

I: [2] This Issue Paragraph contains text that describes another issue that needs a resolution.

{ref}: remember, you may refer to an action item at the end of the minutes or from some other meeting. The following are examples: "see At [3]"; "see At [5] in the 12.5.94 Info-group meeting minutes. "

R: Since this Resolution Paragraph directly follows an Issue Paragraph, it is not based-on any of the proposals listed. It resolves the Issue and all sub-Issues (if any).

{ref}:

P: Since this Proposal Paragraph follows a Resolution Paragraph, it can be assumed it was not the basis of the resolution.

+A: {text}

{ref}:

-A: {text}

{ref}:

Schematic view of the ltlWEB paragraph types: sequence and hierarchy

I: [3] {text}.

{ref}:

P:

R:

+A:

-A:

si:

sP:

{ref}:

sR:

+sA:

-sA:

I: [4]

R:

P:

+A:

-A:

{ref}:

4. Action Items

This section contains the action items generated in this meeting. It may contain action items from a previous meeting which were not completed or new action items generated during the discussion of the issues listed in the agenda.

An action item is represented with an action item tag "AI:" followed by a number (e.g.[l]) which identifies it uniquely in this document. The action item is assigned to a specific person and is followed by a number tasks to be completed as part of the action item. The action item or the tasks may refer to issues, proposals or anything else in the discussion and status sections.

AI: [1] The Action Item Paragraph contains text that describes a specific action that must be taken. It may refer to an issue, proposal or any other paragraph in the discussion or status sections. Since this action item follows an issue, it refers to that issue. Action item paragraphs are followed by an associated person and associated tasks. The action items are also listed in the final section of the minutes.

For: The Person Paragraph assigns a specific person(s) to an action item.

Task: The Task Paragraph lists any number of tasks to be completed as part of the action item.

Example:

AI: [2] Homework 2 grading (this action item duplicates the first action item discussed)

For: Allen Dutoit

Task: Submit graded homework 2 in time for review by Prof. Bernd Bruegge and James Uzmack

AI: [3] TIGER files demo (this action item has a reference to an issue from another meeting.)

{ref}: see to issue I:[1291] in minutes for 1-15-1996 meeting

For: John Doe & Jane Smith

Task: Get TIGER files for the state of PA.

Task: Compile TIGER demo for pmax/ultrix and install it in the afs tools directory.

{ref}: /afs/cs/project/classes-bob/JEWEL/tools