PRODUCT DEVELOPMENT PROCESS CAPTURE & DISPLAY USING WEB-BASED TECHNOLOGIES

by

NADER SABBAGHIAN

B.S. Electrical Engineering
Carleton University (Canada), 1993

Submitted to the Sloan School of Management and the School of Engineering
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING & MANAGEMENT

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999

Signature of Author.....
System Design & Management Program
December 15, 1999

Certified by.....
Earll Murman
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by.....
Steven Eppinger
Associate Professor of Management Science
Thesis Supervisor

Accepted by.....
Thomas Magnanti
Institute Professor
Codirector, System Design and Management Program

PRODUCT DEVELOPMENT PROCESS CAPTURE & DISPLAY USING WEB-BASED TECHNOLOGIES

by

NADER SABBAGHIAN

Submitted to the Sloan School of Management and the School of Engineering on December 15, 1999 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering and Management

ABSTRACT

The goal of this research is to define a distributed knowledge capture method used for modeling the product development process. A Web-based solution is proposed to enable rapid collection, continuous update and clear display of organizational and task interactions in large projects.

Modeling the product development process in large projects is a complex exercise requiring numerous participants and the coordination and clarification of vast amounts of collected information. Currently, this is performed through group meetings or an interview-base process, where participants attempt to integrate their fragmented knowledge of the overall development process. Web technology is used in the proposed approach to address the limitations of present process modeling practices.

A Web-based prototype system has been developed to validate the approach. The system is equipped with 'push' data capture and on-line multi-user issues resolution capabilities. It utilizes a multi-tiered, data-driven Design Structure Matrix (DSM) configuration to present collected information. The prototype system has been developed on the Windows NT platform using Java, Active Server Pages (ASP), MS SQL-Server RDBMS and JDBC middleware.

KEY WORDS

integrated product development, design for integration, re-engineering, concurrent engineering, product development, design structure matrix, web technology, process modeling, distributed knowledge collection, cooperative design, cooperative information sharing

ACKNOWLEDGEMENTS

This work was made possible through the guidance and support of many. First and foremost I would like to thank MIT's Center for Innovation in Product Development, its consortium of industrial partners and the National Science Foundation for providing me with the opportunity to conduct research in this area.

Infinite thanks to my academic advisors Professor Steven Eppinger and Professor Earll Murman for their valuable and continuous guidance over the last two years. I was also blessed with the precious advice of Dr. David Grose, a researcher with great experience at the Boeing Company. I would like to acknowledge his extensive contribution to my work and greatly thank him for openly sharing his tremendous knowledge and wisdom.

I am also enormously grateful to the Boeing Commercial Airplane Group, and especially to Mr. Waltt Gilette, Mr. Brian Jobes and members of the Configuration & Engineering Analysis group.

I would also like to thank my friends and colleagues at MIT for their contributions and guidance. Many thanks go to Tyson Browning, Maria Carrascosa, Shaun Abrahamson, Giammario Verona, Philipp Schierstaedt, Johannes Kuster, Stephen Donnelly and others.

Finally, a special thanks to my family and friends for their support and love. I am grateful to my father Nezam, my sister Negin, my best friend Alessandro, and especially to my lovely fiancé Valeria.

This work is dedicated to the loving memory of my mother who has been and will continue to be the greatest source of inspiration in my life.

BIOGRAPHICAL NOTE

The author has been a research assistant with M.I.T.'s Center for Innovation in Product Development since January 1997, working on the applications of Internet Technology in the area of product development process modeling. Mr. Sabbaghian is a native of Tehran, Iran and has lived in Italy, Canada and the United States. He has a Bachelor degree in Electrical Engineering conferred by Carleton University (Ottawa, Canada) in May 1993. Mr. Sabbaghian has four years of work experience in North America with Andersen Consulting and has worked as summer associate with the European operations of McKinsey & Company.

Address correspondence to:

Nader Sabbaghian
27 Danaher Drive
Nepean, Ontario
Canada K2J 3Y5

Phone: (613) 825-9802
Fax: (613) 825-7528
Internet: nad@alum.mit.edu

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 Problem Statement

There is mounting pressure in all industries to reduce the cost and time required to develop increasingly sophisticated products. Meanwhile, fast changing marketplaces, intense competition and rapid technological evolution have magnified the dynamic nature of product development, creating further need for flexibility and responsiveness in project management. Defining and coordinating development teams and activities under such circumstances is a real management challenge, especially in large engagements [3, 8].

Today's large product development programs can be characterized by the participation of thousands of designers, divided into hundreds of cross-functional teams, working on hundreds of thousands of tasks over a period of several years. Aircraft, satellite systems and automobiles are typical examples of products requiring development projects of such magnitude. The challenge in these programs is to overcome the tremendous complexity involved in planning and executing large numbers of interconnected and dynamic design [3, 6, 8] and development tasks. Success usually depends on management's ability to collect and process a considerable amount of continuously changing information for efficient decision making. Identifying instances of task iteration (planned or unplanned) is critical in reducing complexity and increasing program efficiency [7, 18, 21]. This important aspect of planning is most often neglected because of insufficient data collection and poor means of representation. Traditional project management tools provide a simplified view through the use of precedence network models and are unable to capture the iterative nature of the development process [1, 7].

This research attempts to address implementation issues related to the Design Structure Matrix modeling methodology. This matrix-based technique has proven to be an effective tool for planning and managing product development programs through information flow analysis [1, 7, 8, 15, 17, 18, and 21]. It is capable of intuitively representing complicated dependencies among numerous project entities and raising visibility on potential iterations in the development process.

1.2 Research Background

The *product development process* is defined as "the sequence of steps or activities that an enterprise employs to conceive, design and commercialize a product" [21]. Large product development programs involve hundreds or even thousands of activities performed by individuals with a variety of skills and intellectual capabilities. This type of organizational knowledge is considered by many researchers as the most strategic asset, source of economic value and key to competitive advantage [20, 22]. In analyzing the knowledge creation process, researchers Nonaka and Takeuchi [20] point out that their are two types of knowledge: explicit and tacit. The former relates to the "codified" knowledge, one that is quantifiable and transmittable in a systematic language. The latter refers to "personal" knowledge, one that is context specific and difficult to formalize and communicate. The researchers point out that it is through interactions among these two types knowledge (referred to as four modes of "knowledge conversion") that human knowledge is created and expanded.

Modeling the product development process entails tapping into the tacit and explicit knowledge of a vast population of experts in the organization. This is an enabling exercise in knowledge conversion according to the Nonanka-Takeauchi framework. Structuring activities and information flows into comprehensible representations provides key insights on the dynamics of product creation hence contributing to organizational knowledge in the critical area of product development.

Furthermore, numerous studies in the area of knowledge management have pointed out that flexible, quick and equal access to information throughout the organization is a necessary condition for the advancement of knowledge [5, 9, 13, 14]. Indiscriminate availability of information and the creation of technology infrastructures to enable information sharing is the central theme in many studies on project coordination mechanisms [4, 10, 14, 19].

One can therefore easily conclude that organizations can greatly benefit from a more detailed understanding of their product development process and better ways to share and promote this aspect of their knowledge.

1.2.1 Role of Information Technology

Recent advancements in information technology, especially the area of distributed computing have created entirely new opportunities in the area of knowledge management. The development of network technologies and the emergence of the

Internet as the leading collaborative tool have led to important steps towards the creation of a *seamless* communication environment. The word seamless alludes to the so-called "five-anys" signifying the creation of an environment that enables members to communicate *anything* in *anyway*, with *anyone* located *anywhere* in the world at *anytime* [14]. Prof. Halal [22] refers to such systems as "intelligent infrastructures" alluding to their ability to capture and distribute existing knowledge across organizations and facilitate learning. Examples of successful deployment of such intelligent infrastructures are presented in Figure 1-1 below.

Hewlett Packard
A corporate IS unit is putting management procedures and personnel practices onto a www site and Lotus Notes. A system called Knowledge Links supports product divisions with purchasing services, engineering data, market intelligence, and best practices. All this is unified by a "World Innovation Network" that allows employees to probe each other's experiences on what works.

Merrill Lynch
The world's largest security borker helps its eighteen thousand account managers operating in five hundred offices serve their millions of clients with a computer network that stores the firm's knowledge base about securities, financial forecasts, and the like.

Andersen Consulting
Andersen uses a global network called Knowledge Xchange to pool experiences and best practices of its worldwide consulting practice.

Figure 1-1 Exemplars of Knowledge Systems [22]

According to Prof. Halal: "*An intelligent infrastructure consists of a corporate wide information system and a web of close working relationships connecting entrepreneurial units to common pools of share knowledge. The result is a central nervous system that leverages ordinary learning to powerful new levels, forming an intelligent organization*"

However, current knowledge systems in the area of product development program planning, execution and process management fail to qualify as so-called forms of intelligent infrastructure. Factors contributing to their inability to gain widespread acceptance as catalysts of process-driven learning in organizations include [12]:

- single user, standalone implementations
- poor means of process representation
- steep learning curve combined with a lack of adequate performance support
- cumbersome entry of process information

A number of innovative approaches have been developed using Network technologies to address the above deficiencies of conventional process modeling tools [8 , 9, 16]. These approaches, despite taking advantage of the ease of information distribution and using creative process representation techniques, maintain the existing paradigm of centralized model creation and data management and therefore can not be considered fully distributed knowledge systems.

1.2.2 Data Collection Techniques

Typically process related information is obtained through extensive interviews with various experts in the organization. Cross-functional meetings are organized and facilitated by a small team of process modelers. These individuals are generally responsible for the coordination of data collection activities as well as entry, analysis and validation of gathered information.

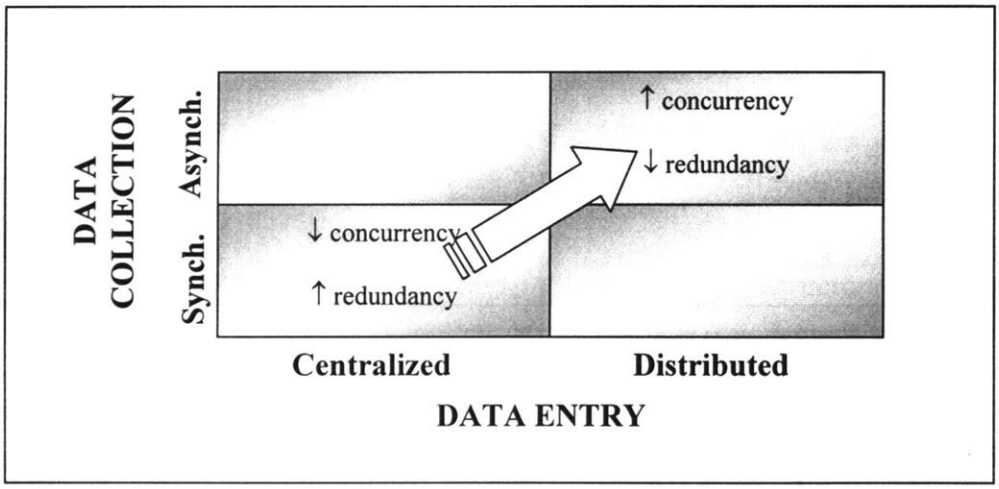

Figure 1-2 Model acquisition attributes

The process of model acquisition can be characterized by the two dimensions presented in Figure 1-2. This space-time matrix categorizes model acquisition approaches supporting either single or multiple point data entry and allowing users to interact either in real-time (synchronously) or in a time-independent fashion (asynchronously). The lower-left quadrant reflects the most common approach where data is collected

asynchronously and entered by a select group of individuals (referred to previously as "process modelers"). This data collection approach heavily relies on the process modeling team, with information providers playing mostly a passive role in the in model construction. This synchronous data acquisition approach is logistically difficult to carry out and is therefore time consuming. Significant effort is spent facilitating interactions among users in order to resolve data integration issues. There is little concurrency in data collection, since separate team meetings need to be scheduled and facilitated by the process modeling team. In addition, data is prepared twice. The first time by individual information providers in preparation for data collection meetings, and subsequently by process modelers for entry into the information system used for modeling.

The upper left quadrant of Figure 1-2 illustrates situations where process modeler's obtain information through individual contacts (through interviews) with the target audience or through the preparation of surveys that are individually and remotely compiled. Information continues to be centrally managed. Process related data is passed from target individuals to the modeling team who structures and compiles it for entry into appropriate information systems for further analysis. This scenario presents a higher degree of concurrency, due to the fact that data can be collected in waves by requesting data from a group of participants simultaneously (e.g. sending out an e-mail survey). However, due to the lack of distributed means of information entry, the degree of data entry redundancy remains the same as the previous case. Once again, participants must first compile data in surveys or other forms of documentation, which are then passed to the modeling team for entry into the appropriate tools for process analysis.

The top-right quadrant in Figure 1-2 refers to the modeling approach recommended in this research. In the asynchronized-decentralized scenario the role of data collection coordinators is minimized and model construction relies on the direct participation and interaction among the various experts in the organization. Data providers play a much more active role in model construction and a higher degree of concurrency is achieved through the availability of distributed data entry. Redundancy is reduced, with users providing information directly to an information system's central model repository. The three examples of knowledge systems reported in Figure 1-1 also utilize this approach for distributed knowledge management.

The above framework for the modeling process is drawn from Mariani and Roden's analysis of information models used by Computer Supported Cooperative Work (CSCW) information systems [11]. They present a groupware space-time matrix that characterizes cooperative systems according to the geographical location of the users and the form of

interaction supported. Figure 1-3 presents this framework highlighting the effectiveness of existing CSCW systems in addressing the two dimensions of cooperative information modeling. The concept of cooperative system developed in this research for the purposes of product development process capture is aligned with the recommended approach for the implementation of a remote-asynchronous CSCW relying on a combination of *messaging* and *web-based* conferencing.

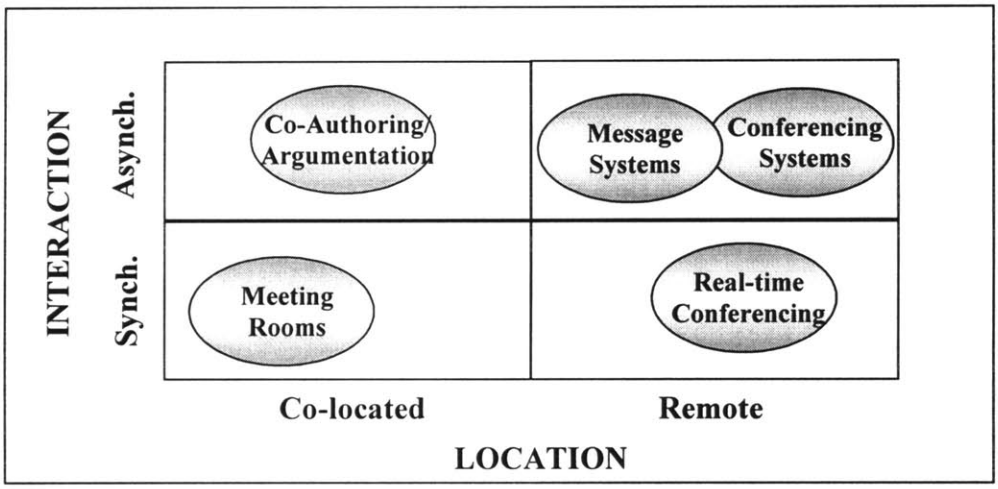

Figure 1-3 A Classification of CSCW systems [ref]

1.2.3 An Internet-based Distributed Approach

The product development modeling process can be characterized by the cycle presented in Figure 1-4. As stated previously, process modeling is an iterative mechanism by which the knowledge of a group of experts is collected and integrated to obtain a tangible representation of the activities and information flows at play during product development. Through the use of Web-based technologies this research attempts to accelerate this cycle while reducing the amount of resources required for the coordination and facilitation of the model creation process.

Data collection requests are initiated by the modeling group with the intent of gaining further detail on a series of high-level program activities. The web-based system enables every engaged participant to initiate requests to other experts in the organization in order to obtain further detail on various areas of the process. From this user-driven activity decomposition mechanism an increasingly detailed picture of the process emerges.

A distributed and cross-platform web-based interface provides the means for the collection of required data consisting of: task names, corresponding responsible

individuals/teams and, deliverables used and produced. Gathered information is stored in a central repository. During modeling's *Validation* phase collected process data is analyzed to ensure its accuracy and consistency. The system's *Notification* mechanism provides the means of introducing automation in the contact management component of modeling. A rule-based system is designed to diagnose situations requiring user intervention and construct targeted hyper-link enabled electronic messages to request participation in the issue resolution process.

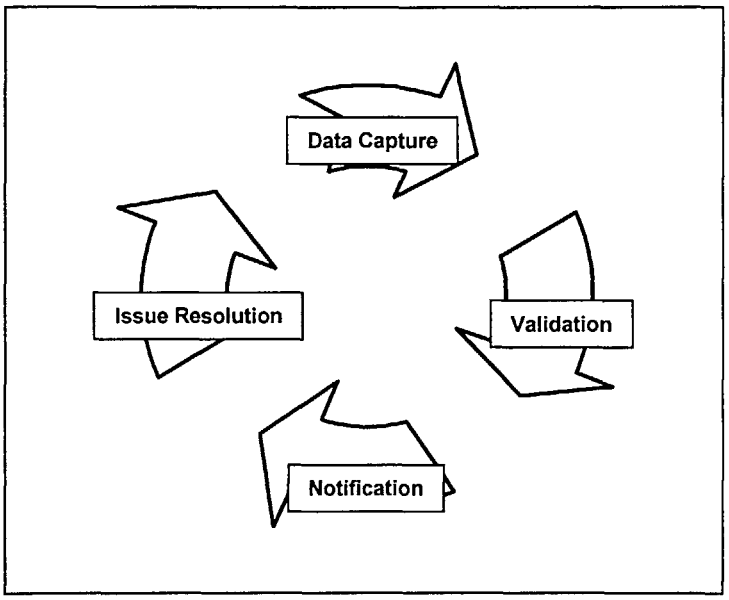

Figure 1-4 Distributed web-based modeling cycle

The aim of this framework is to reduce the time and effort required by the modeling team to pursue process-related data across the organization. Internet technologies are quite suitable for achieving this goal by easily providing the capabilities required to actively engage a large geographically dispersed group of information providers.

1.2.4 Process Modeling at the Boeing Commercial Airplane Group (BCAG)

BCAG's efforts in the area of product development process modeling over the last few years can be traced to the Cross Functional Integrated Design (CFID) group. This process improvement team headed by Dr. David Grose has strived to promote process thinking through the development of process mapping and analysis tools and their deployment in numerous pilot projects throughout the organization. Most notably Dr. Grose and his team have developed a software system based on the so-called "data driven" process modeling approach that requires the explicit definition of information flow among activities in the model. Each task is modeled together with its specific

information requirements and outputs. Dependencies are solely defined through the matching of "required" and "produced" deliverables.

The tool utilizes a multi-tiered Design Structured Matrix (DSM) representation to overcome challenges related to model size and complexity. The modeling exercise (as seen in Figure 1-5) begins with defining decision gates and presenting their dependencies in a tier 1 DSM. Rows within this matrix expand to the next level DSM representing program milestones and, the decomposition process creates as many levels of increasingly detailed DSMs until individual tasks are identified. In this approach, deliverables or data entities also follow a similar hierarchical pattern. The decomposition process for model construction occurs in a top-down fashion. The program schedule, on the other hand, is produced from the bottom-up with designers providing estimates on each identified task.

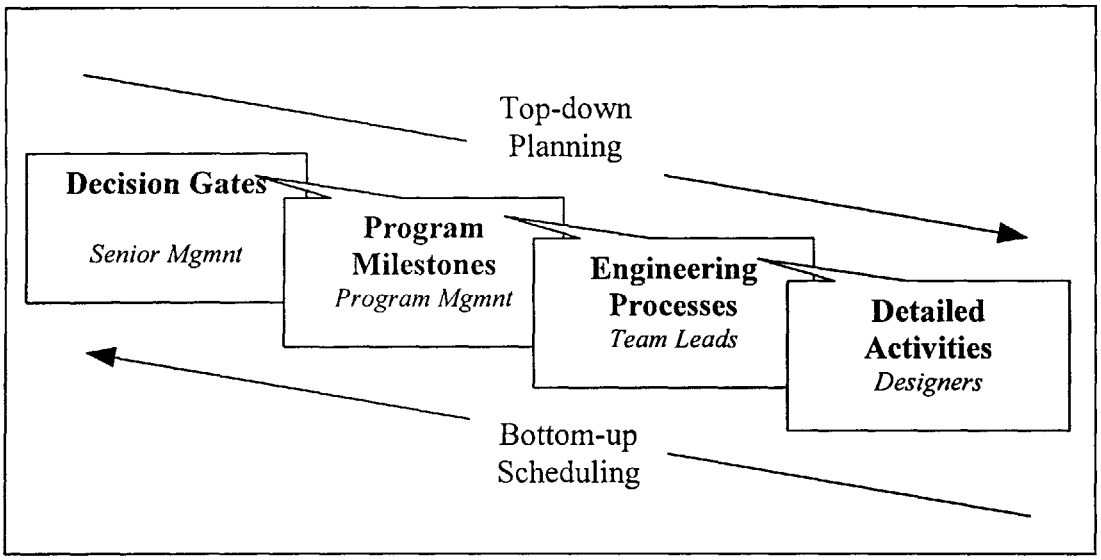

Figure 1-5 CFID's multi-tiered top-down/bottom-up modeling approach

Over the last few years CFID's modeling activities have been confined to a group of relatively small engagements. Interest in the above approach is growing as program managers and chief engineers become increasingly aware of its benefits in alleviating resource and schedule pressures. Despite management's enthusiasm, so far there has been no coordinated effort for a program-wide deployment of the tool.

Interviews with members of the CFID team have revealed some of the difficulties encountered in their modeling initiatives, specifically as they relate to the data collection issue. Dr. Grose's team maintains that product development teams are presently unable to readily provide the necessary information for data-driven model construction. The CFID

team spends a significant amount of time assisting each group in defining its processes and data requirements at various levels of abstraction in the model hierarchy. Frequently this turns into a coaching exercise in "process" and "system" thinking. Clarifications are needed on concepts such as "task", "data", "milestones" which are often used interchangeably. In addition, each participating team requires a certain level of background training on DSM theory and visualization techniques. Experience demonstrates that users find it difficult to adapt to matrix representations of process elements and dependencies.

The CFID team has observed a number of interesting trends over the course of their process modeling experience at BCAG. They point out the prevalent use of deadline driven scheduling (i.e. tell me "*when*" you want it for, and I'll tell you "*how long*" it will take) and ad-hoc sequencing of tasks (according to the way it has "*always*" been done) during the planning process. They also highlight a general tendency to ignore task iterations in the process. Iterations are acknowledged and understood but do not appear explicitly in any of the program plans. Consequently, the CFID team maintains that using existing tools and techniques it is difficult to assess the impact of iteration on schedule and cost.

During the summer of 1997 a series of interviews with Boeing senior management further demonstrated the need for alternative approaches and more sophisticated tools in the area of program planning. Figure 1-6 presents a series of insightful comments captured during the interviews. The quotes refer to the interviewees overall assessment of current planning practices at BCAG.

"I believe Boeing spends a lot of effort developing and executing SCHEDULES... not planning. Planning would account for resource constraints, evolving/changing requirements, morale, ... etc. Boeing does not do enough planning and lacks the skills and tools to do it effectively on all programs. "

"We need to look at how we define milestones. We spend a lot of effort doing the work, but we don't have the modeling tools to do the job well. We find Microsoft Project the answer to everything."

"We look at history, squeeze history into the current market requirements and assume we can accomplish with less resources and cost. Results are predictable and probably expected due to ever increasing regulatory requirements & complexity of products. We MANAGE!!"

Figure 1-6 Quotes from BCAG management on the topic of program planning

1.3 Thesis Overview

The goal of this thesis is to present an improved approach for the construction and visualization of large process models with the aid of web-based technologies. Chapter 2 of this work provides readers with sufficient theoretical background on the Design Structure Matrix (DSM) modeling methodology as well as details on the multi-tiered/data-driven variation used in this research. The web-enabled distributed and asynchronous modeling approach is presented in Chapter 3. The approach is synthesized into the three main topics of data collection, representation and integration. Techniques and concepts used to address each of these functional areas are presented in detail. The objective is to provide an implementation independent concept of the proposed cooperative process modeling system. For information on how this approach was translated into a working web-based prototype readers can refer to Chapter 4. Here, the prototype system development process is presented in detail starting from the outline of requirements and specifications to the tool's high-level design and software operating environment. Chapter 5 concludes by summarizing the outlined approach, presenting feedback received and lessons learned during prototype development and, discussing directions for further research on the role of the internet as the enabling technology for process analysis in product development.

2 DESIGN STRUCTURE MATRIX (DSM)

2.1 DSM Overview

Donald Steward introduced the Design Structure Matrix in 1981 as a generic matrix-based framework for information flow analysis [7]. It consists of an N-square diagram showing the interaction of each element with every other element in the model. By reading across a row, one can observe these interactions through the cell contents corresponding to each cross-referenced column. The matrix configuration serves as a powerful visualization tool for the analysis of very complicated dependencies. Various conventions are used to define the content of the DSM cells. These conventions usually depend on the model type and the nature of the problem being tackled. The most common uses of DSM and their applications are summarized in Table 2-1 below [1].

Approach	Application
Parameter-based modeling	System architecture analysis, product re-design
Team-based modeling	Organizational structure analysis, team design
Task-based modeling	Project planning, PD process analysis

Table 2-1 Common DSM classifications

This research focuses on issues related to the deployment of the task-based DSM modeling approach.

2.1.1 Parameter-based DSM

This type of modeling is used to analyze system architecture based on parameter interrelationships. A parameter-based DSM is constructed through explicit definition of a system's decomposed elements and their interactions. A systematic taxonomy and a quantification scheme assist in the analysis by categorizing types of interactions among system elements and associating an appropriate weight to each. Table 2-2 and Table 2-3 present the classification of interactions and an example of a quantification scheme proposed by Pimmler and Eppinger [15].

Interaction	Description
Spatial	Associations of physical space and alignment; need for adjacency or orientation between two elements
Energy	needs for energy transfer/exchange between two elements
Information	needs for data or signal exchange between two elements
Material	needs for material exchange between two elements

Table 2-2 Simple taxonomy of system element interactions

Type	Value	Description
Required	+2	Physical adjacency is required for functionality
Desired	+1	Physical adjacency is beneficial but not absolutely necessary for functionality
Indifferent	0	Physical adjacency does not affect functionality
Undesired	-1	Physical adjacency causes negative effects but does not prevent functionality
Detrimental	-2	Physical adjacency must be prevented to achieve functionality

Table 2-3 Example of spatial interaction quantification scheme

The parameter-based matrix can be manipulated to cluster elements into a set of sub-systems that reduce the overall system's coordination complexity. This can be accomplished by clustering highly interactive elements into subsystems while attempting to reduce inter subsystem interactions. Several clustering heuristics have been developed for parameter-based DSM analysis [15, 17, 18]. Further studies have provided links and insights to task allocation and Integrated Product Team (IPT) structures [2].

2.1.2 Team-based DSM

This approach is used for organizational analysis and design based on information flow among various organizational entities. Individuals and groups participating in a project are the elements being analyzed (rows and columns in the matrix). A Team-based DSM is constructed by identifying the required communication flows and representing them as

connections between organizational entities in the matrix. For the modeling exercise it is important to specify what is meant by information flow among teams. Table 2-4 presents several possible ways information flow can be characterized [1].

Flow Type	Possible Metrics
Level of detail	Sparse (documents, e-mail) to rich (models, face to face)
Frequency	Low (batch, on time) to high (on-line, real-time)
Direction	One-way (monologue) to two-way (dialogue)
Timing	Early (preliminary, triggers the process) to late (ends the process)

Table 2-4 Information flow classifications

Once again, the matrix can be manipulated in order to obtain clusters of highly interacting teams and individuals while attempting to minimize inter-cluster interactions. The obtained groupings represent a useful framework for organizational design by focusing on the predicted communication needs of different players.

2.1.3 Task-based DSM

This research will focus on the task-based use of the Design Structure Matrix. Figure 2-1 shows a sample task-based DSM. Tasks appear identically labeled in rows and columns of the matrix and are arranged top-down according to their sequence of execution. Each marked cell represents a task dependency. The convention adopted in this research regards row elements as information "providers" and column elements as information "dependents" or "receivers". For example, in Figure 2-1, the marked cell found at row 2, column 4 represents an information provided by Task 2 to Task 4.

Three types of task interactions can be observed from the matrix. In Figure 2-1, Tasks 1 and 2 are "independent" since no information is exchanged between them. These tasks can be executed simultaneously (in parallel). Tasks 3, 4, and 5 are engaged in a sequential information transfer and are considered "dependent". These tasks would typically be performed in series. Tasks 7 and 8, however, are mutually dependent on information. These are "interdependent" or "coupled" tasks often requiring multiple iterations for completion.

Marked cells below the diagonal represent iterations in the process. This occurs when an activity is dependent on information from a task scheduled for a later execution. Such scenarios often lead to rework and are undesirable. A number of algorithms have been developed [7, 8, 18, 20] to minimize such instances of iteration (below diagonal marked cells) by re-arranging the sequence of tasks in the process. Methods are also available on how to handle iterations in the process that cannot be eliminated through re-sequencing.

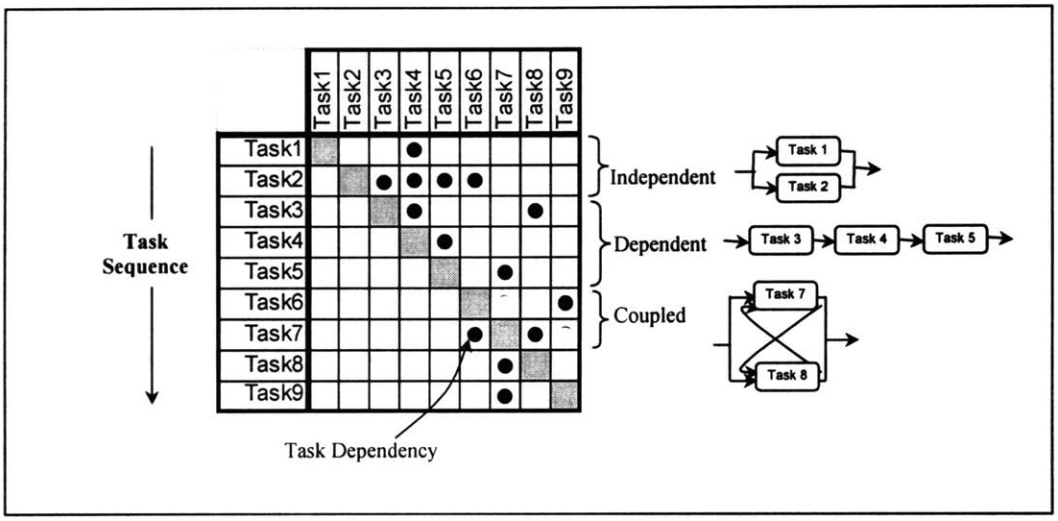

Figure 2-1 Sample Task-based Design Structure Matrix

DSM models using simple binary representations strictly display the existence of a dependency between two tasks without providing additional information on the nature of the interaction. Further studies have extended the basic DSM configuration by capturing additional facts on the development process. For example, the so-called numerical DSM adds task duration in the diagonal elements, and replaces marks with numbers in the off-diagonal cells each representing the degree of dependency between two tasks [7].

2.2 The Data-driven DSM

This research utilizes a DSM modeling technique pioneered by Dr. David Grose at Boeing called the *Data-Driven* approach. The method consists of creating process models through explicit capture of information exchange between project tasks. Deliverables/data produced and used by each activity are obtained in order to create a task-based DSM. A dependency (marked cell) is created once a task's output is defined as input to another task in the process. Figure 2-2 presents a sample DSM constructed using explicit information flow. As seen, the dependency between tasks 5 and 7 in the DSM results from task 5 producing deliverable β required by task 7.

The practice of explicitly defining information interfaces among tasks presents several benefits. The model enables management to identify inaccuracies in the constructed model and inefficiencies in the process prior to task sequencing analysis. From the modeling exercise certain tasks emerge as producing deliverables that are not required anywhere else in the process. These tasks are candidates for elimination since the model can clearly prove their lack of contribution to the project outcome. Similarly, other activities claim to utilize deliverables that are not produced anywhere in the process. In this scenario, since no activities are scheduled to produce the deliverables in question, required data becomes unavailable at the time of a task's execution introducing additional unplanned work, and therefore causing delays during project implementation. Redundancy is also easily identified anywhere a deliverable appears as the output of multiple tasks.

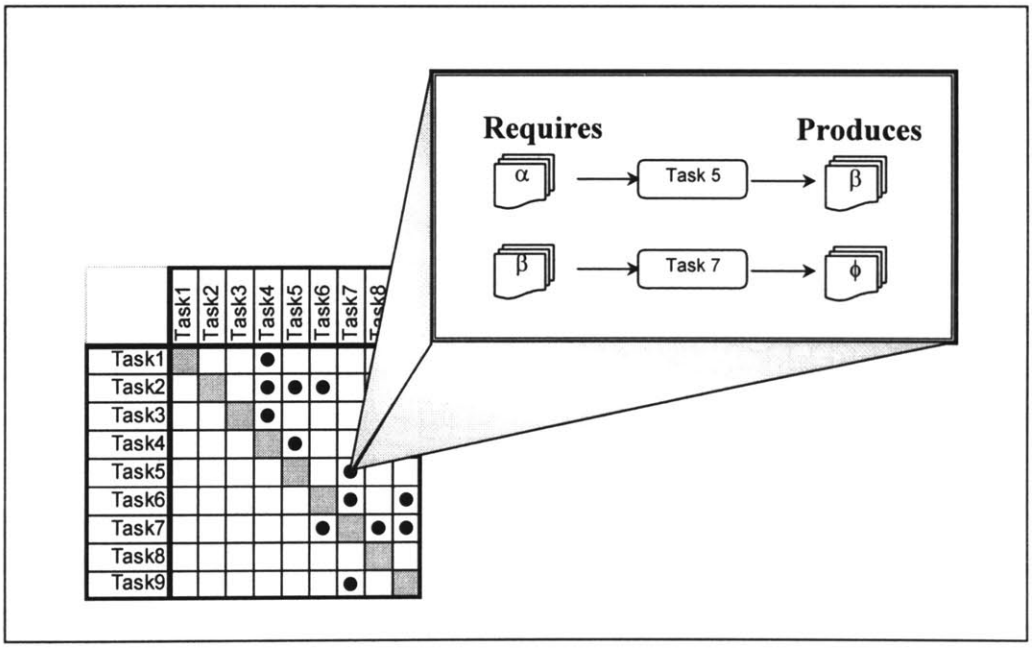

Figure 2-2 Sample Data-driven DSM displaying explicit information flow

Overall, this approach has demonstrated a remarkable ability to detect inconsistencies in projects by highlighting redundant and obsolete activities in the process. Management can obtain a detailed understanding of each project's information requirements at all levels and can therefore ensure that activities are appropriately planned to address internal project needs. Using the data-driven modeling approach managers can take a closer look at the planned activities prior to scheduling and resource allocation. The more analytically rigourous method also ensures that DSM resequencing analysis is

performed on an accurate model, one whose marked cells correspond to proven interactions among tasks.

2.3 Multi-tiered Configuration

Presenting very large models in a single matrix is challenging. When constructing models comprised of hundreds of tasks, the intuitiveness provided by the DSM representation diminishes. It becomes increasingly difficult to identify interfaces by observing the off-diagonal elements of a very large matrix. The method therefore loses its advantage of simplicity and becomes increasingly difficult to grasp.

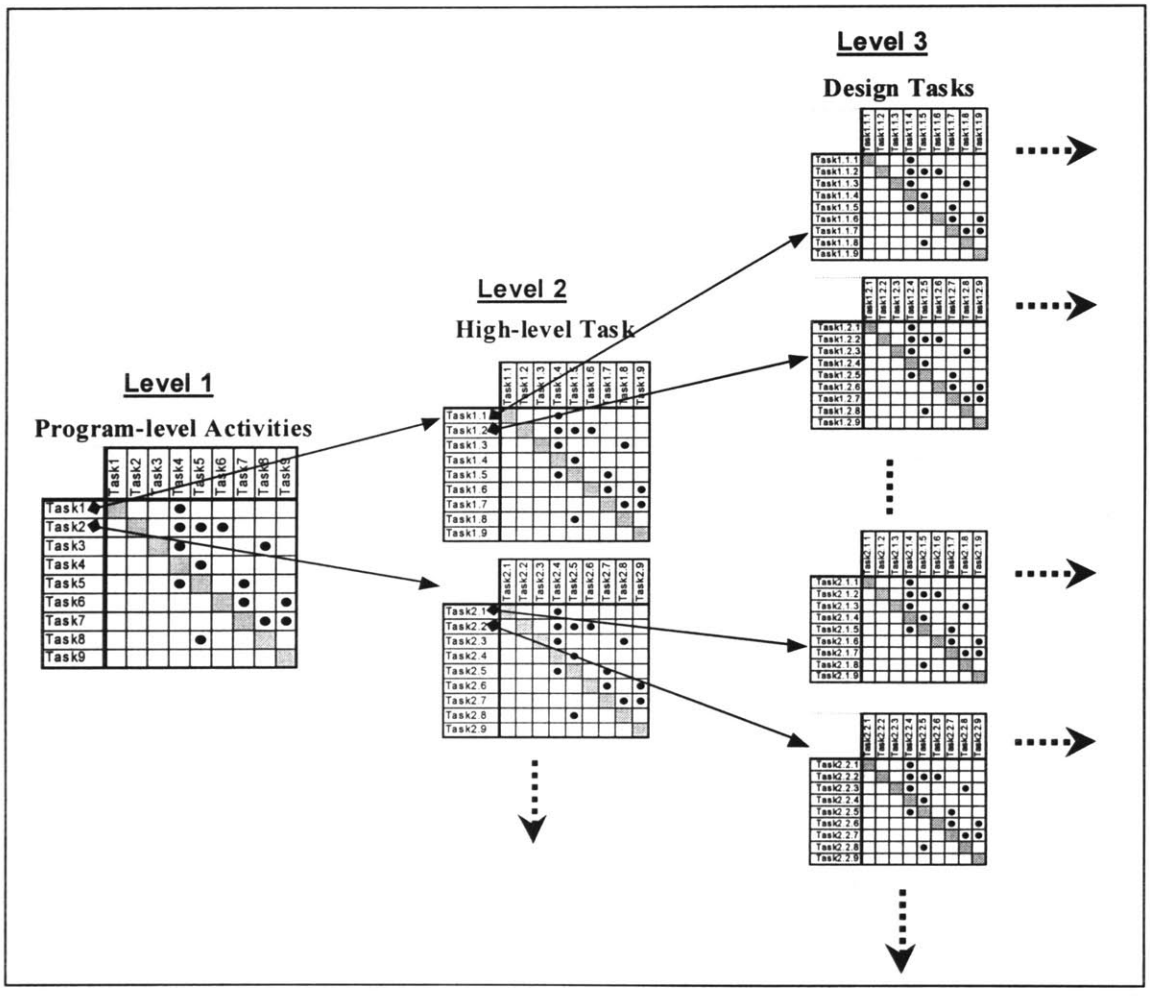

Figure 2-3 Sample DSM Multi-tiered configuration

A very large DSM can be effectively structured into a hierarchy of smaller DSMs. This configuration avoids problems related to presenting extremely large matrices by shifting the focus to smaller ones, obtained through hierarchical decomposition. It also provides

the flexibility to analyze the process at different levels of detail. This multi-tiered approach was developed by Dr. Grose at Boeing and has been adopted by this research as an effective strategy for both data capture and presentation. Figure 2-3 provides a sample view of this multi-tiered approach.

The modeling effort begins from the highest level activities and deliverables in a project (called Level 1). Next, each high-level task is further decomposed into a set of sub-tasks forming a series of level 2 DSMs. The decomposition process continues until the appropriate level of detail for analysis is obtained. A typical case may entail beginning from a set of overall program activities performed by major groups and performing multi-level decompositions all the way to a set of tasks performed by individual team-members in the project. In the example presented in Figure 2-3, the first two tasks in each matrix are decomposed to provide additional detail through the construction of lower level DSMs. The first two tasks of each of these Level 2 matrices are once again broken down into Level 3 DSMs leading to the creation of an increasingly refined model.

2.4 A Combined Approach

This research adopts both the data-driven and the multi-tiered methods for the construction and display of large-scale DSM models. The combined strength of the two techniques leads to increased model accuracy, better visualization and the creation of a structured approach to multi-user data collection. All matrices in the multi-tiered configuration are created through explicit definition of deliverables (or information) being exchanged among their tasks. These dependencies are carried through the decomposition process maintaining a consistent representation of information flow at higher and lower level matrices. There are three possible forms of information exchange in this structure: Internal, External and Boundary interactions.

2.4.1 Internal Interaction

This form of interaction simply refers to information being exchanged within a single matrix. Marked cells within individual matrices throughout the model represent task dependencies created by this type of interaction. For example, the exchange of deliverable α between tasks 5 and 7 in Figure 2-4 is *internal* as represented by the marked cell in position (5,7).

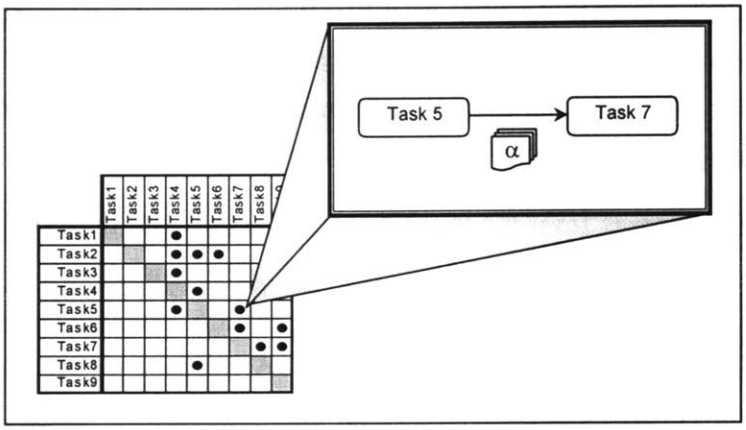

Figure 2-4 Sample Internal interaction

2.4.2 External Interaction

This consists of information being exchanged between two or more matrices. When high-level tasks are broken into lower-level matrices, the internal interactions between such tasks are represented by exchange of information between their corresponding decomposed matrix. In the case of the internal exchange of deliverable α in Figure 2-4 becomes an external exchange between two matrices, obtained from the decomposition of tasks 5 and 7 (see Figure 2-5).

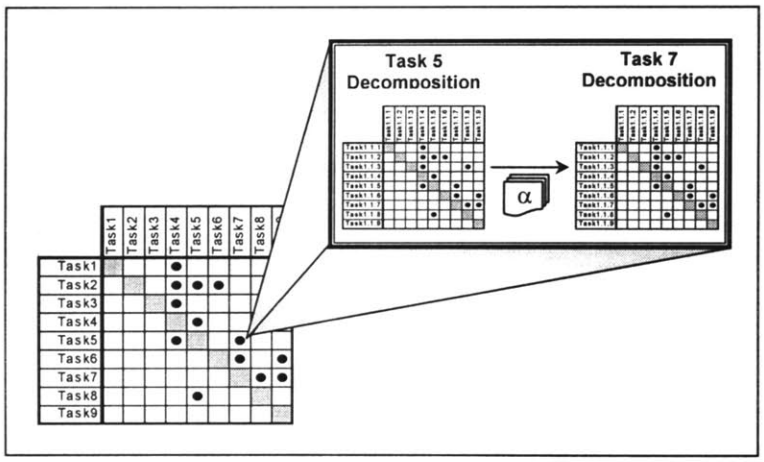

Figure 2-5 Sample External interaction

In lower-level matrices these interactions appear as deliverable exchanges between sub-tasks and are therefore not visible by looking at each decomposed matrix separately. In Figure 2-5 deliverable α is produced by a sub-task in the decomposed matrix of Task 5 and used by a sub-task in the decomposed matrix of Task 7. When looking at the overall model in its lowest level of detail, external interactions can be interpreted as instances of

dependencies among tasks that are positioned relatively far from each other. These are generally marked cells that are located far from the DSM's diagonal. The impact of such configuration on the visualization aspects of the approach is discussed later on in section 3.2.2.

It is important to clarify that throughout this research the labels *internal* and *external* are not used as attributes of deliverables in a model but strictly relate to interactions among tasks. The terminology referred to deliverables can become quite confusing since in certain cases a deliverable can be engaged in both an *internal* (link between adjacent sub-tasks) as well as *external* interactions (link between sub-tasks in different matrices).

2.4.3 Boundary Interaction

This information exchange occurs at the boundaries of the model interfacing with entities outside the project. In each engagement a number of dependencies relate to interactions with entities that are not explicitly defined in the model such as customers, suppliers, regulatory agencies, outside contractors etc. Information is provided and received from such entities and must be accounted for in the modeling exercise.

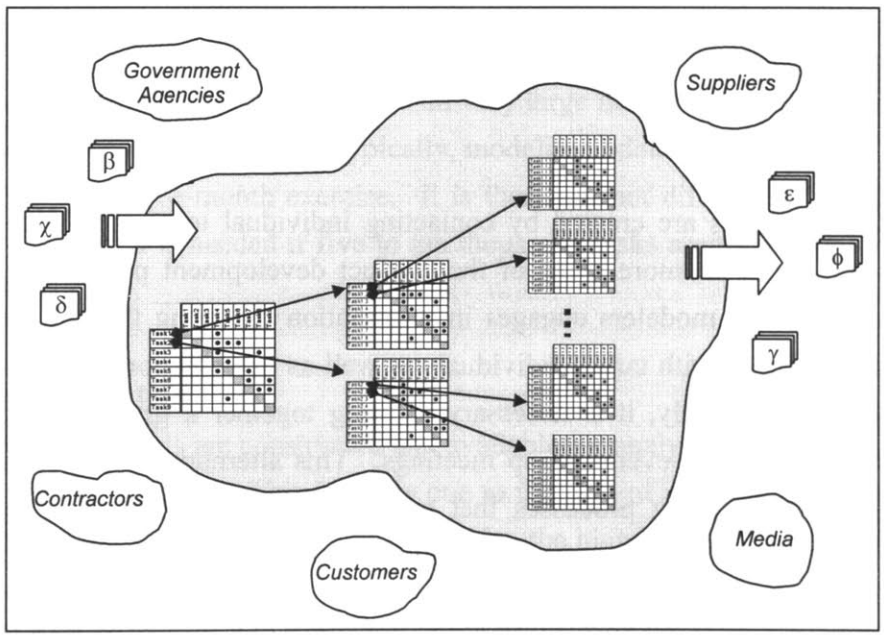

Figure 2-6 Sample Boundary interactions

In Figure 2-6, for example, deliverables β, χ, δ are inputs and ε, ϕ, γ are outputs to the overall process relating to interactions with outside groups. The origin or destination of these dependencies is not present within the model and they appear as

required deliverables not produced by any defined activity, or outputs for which there are no internally defined needs.

2.5 Challenges in Modeling Large Projects

Modeling large projects, in the order of thousands of tasks, requires the participation of many individuals from numerous functional areas [3, 8]. In these programs, it is clearly unrealistic to assume that any one individual is capable of outlining all required activities, organizational responsibilities and information dependencies in detail. As the size and complexity of a project grows so does the number of experts and team leaders that need to be consulted in order to complete the various pieces of the process modeling puzzle [4].

The modeling exercise requires a high degree of interaction among its many participants primarily for data integration purposes. Agreements must be reached on the role of tasks, the terminology and content for outlined deliverables as well as boundaries of responsibility. The intent is to assemble fragmented knowledge of an often-complex development process by tapping all available sources of know-how in the organization. Coordinating this participatory model creation process with a large group of information providers is challenging.

2.5.1 Data Collection

Currently DSM models are created by contacting individual team members as subject matter experts in one or more areas of the product development process. Typically, a small team of process modelers engages in information gathering through surveys and face to face interviews with target individuals as well as examination of existing project documentation. Frequently, it is necessary to bring together a group of experts from various domains through several group meetings. This alternative is especially needed when dealing with existing processes that are only partially understood by any one individual in the organization, or when attempting to model completely new processes.

Once data is collected, it is entered in a DSM modeling tool or simply in a spreadsheet program (such as MS Excel) for analysis. Integration of collected data requires further rounds of consultation and meetings with data collection participants. Members of the modeling team facilitate such meetings with the goal of bringing closure to integration issues created from the first round of data gathering. The data collection and issue

resolution continues through several iterations until a mutually agreeable process model is created.

The biggest challenge in DSM data collection is to reduce the time required to complete the process modeling exercise. This means reducing the total time duration from the beginning to the end of a large-scale modeling initiative. Currently, using the above technique, a significant amount of effort is required for a typical modeling team of 5 to 10 full-time professionals to construct models in the order of thousands of tasks. A burdening overhead is needed to organize an increasing number of team meetings for data collection and issue resolution. Larger groups of people must be brought together increasing the likelihood of schedule conflicts and therefore introducing significant delays in the model creation process. On one hand, not all parties invited to attend a modeling issue resolution meeting are needed or consulted during the meeting. On the other hand, the absence of one key individual could lead to a stall in addressing issues on the meeting's agenda. This is partly due to the fact that modelers, without a detailed knowledge of the process being modeled, must attempt to coordinate issue resolution meetings, and often fail to invite key individuals or request the presence of those having little to do with the issues being discussed.

Current data collection methods based primarily on a synchronous participatory approach clearly run into problems when dealing with very large models requiring the involvement of a large number of individuals. Typically, modeling a data driven DSM of around one hundred tasks is a two-month exercise. It is therefore not difficult to conclude that an alternative approach is needed if five to ten thousands tasks are to be modeled using the DSM methodology.

2.5.2 Representation

Currently DSM models are constructed and available for analysis only to a small group of experts in an organization. This is partly due to the lack of distributed and user friendly tools for the dissemination of DSM models. Also the high learning curve associated with this methodology is a significant barrier to its widespread adoption in organizations. The majority of professionals are quite familiar with traditional GANT or PERT view of processes promoted by increasingly popular tools such as Microsoft Project or Primavera. However, very few are familiar with the matrix-based process representation used by the DSM methodology. Introduction of tools aimed at supporting newcomers with features that partially translate the matrix to more familiar visualizations is one way of overcoming users' inertia.

Successful adoption of DSM means expanding its use beyond the current circle of DSM specialists and into the much larger group of managers, team leaders and functional experts. Increased awareness easily translates into a higher degree of user participation in data collection and most importantly a better understanding of the methodology's capabilities and benefits. There is also the critical issue of ownership in the analysis. Insights resulting from DSM modeling often point to a need for radical changes in the way projects are historically conducted. In this case, managers asked to undergo changes are the same ones that participated in the analysis which led to such changes and fully understand its underlying rational.

Providing adequate visualization for DSM models in very large programs is problematic. As explained in section 2.3 the intuitiveness of the DSM representation is lost when viewing matrices in the order of hundreds or perhaps thousands of tasks. The multi-tiered configuration provides a plausible solution to this problem by fragmenting the matrix into a series of smaller DSMs at different levels of detail. This maintains the user-friendliness of small DSMs while allowing users to easily navigate through the overall model and access increasing level of detail in a particular area. However, constructing the hierarchical task structure in a way that is familiar to users is challenging. One must decide what criteria to adopt for grouping tasks and how to ensure correct representation of inter-task information flow at various levels of abstraction. An appropriate method of inter-level navigation must be used, one that guarantees no loss of information during aggregation and decomposition operations.

2.5.3 Model Quality

Typically, DSM models are constructed by outlining all necessary activities and asking individuals responsible for each to identify inter-task dependencies. This is accomplished by specifying, for each task, all other activities that provide it information and all those that are dependent on its deliverables. In very large projects, however, it is sometimes difficult for individuals to accurately pinpoint the originator of information they require or the recipients of deliverables they produce. This leads to a situation where DSM models' accuracy becomes highly dependent on participants' knowledge of the origins and destination of deliverables. This knowledge becomes fuzzy with the growing number of participants and the increasing variety of functional disciplines involved in a project. Experience in large projects has demonstrated that the conventional origin/destination based approach to data collection leads to the compilation of partially accurate information resulting in deterioration of DSM model quality. Certain

dependencies appear where there are no tangible exchanges of information while other critical interactions are not represented at all. Sequencing and other forms of DSM analysis are therefore performed on models, which poorly reflect the dynamics of projects being examined.

The data-driven approach, presented in section 2.2, addresses this issue by ensuring that interactions in the DSM are represented only in cases where output deliverables match input requirements. Participants are asked to focus their attention on defining the needs and outcomes of their activities. A more accurate model can be obtained by explicitly collecting input and output information and using these to construct DSM task dependencies. Identifying instances of task redundancy and obsolescence in a project is another important aspect of the capabilities of a data-driven DSM. As explained in section 2.2, the method is able to detect unnecessarily planned tasks by highlighting their lack of contribution to the project's final outcome. These may relate to obsolete tasks whose presence in the schedule is justified solely through historic means and without a clear understanding of their role in the project's advancement. The data-driven method provides the capability to challenge past assumptions. It simply requires that each planned activity justify its presence by specifying its contribution to the program's outcome.

The dynamic nature of product development leads to frequent changes to the process Therefore, a static model could often be an inaccurate one. Factors such as unexpected change in project scope, customer requirements or market environment may lead to tasks or deliverables becoming obsolete or new ones being added. Accommodating such changes in order to maintain model accuracy is also requires laborious effort. In large projects, understanding the impact of changes and ensuring its timely communication to interested parties is quite challenging. Without a detailed map of how all the activities are connected in a project it becomes clearly difficult to perform quick re-planning and re-allocation of resources in response to a change.

2.6 Summary

This chapter presented an overview of the Design Structure Matrix modeling methodology by outlining its most common applications in the area of product development. The task-based variation, being the focus of this research, was further analyzed and its strengths and weaknesses explored. The modeling approach used in this thesis was presented in detail as a combination of the *data-driven* and *multi-tiered*

configurations. The various forms of interactions and dependencies were described to provide readers with the necessary understanding of topics later discussed in this thesis. Finally the methodology was analyzed within the context of modeling large-scale projects. Issues and obstacles related to the scale-up of DSM modeling were explored to provide a clear picture of the problems tackled in this research.

3 MODELING APPROACH

This chapter presents a distributed and asynchronous modeling approach to address the previously outlined limitations of DSM use in large projects. The method is implemented through a Web-based prototype system with the following overall objectives:

- To reduce data collection effort by efficiently engaging a large number of participants in the modeling exercise.
- To promote DSM adoption in project planning and management by providing distributed and user-friendly access to very large DSM models.

Internet technology was chosen as the most suitable infrastructure for the proposed system due its ease of deployment, cross platform capabilities and flexibility in data capture.

3.1 Distributed Data Collection

The most distinctive characteristic of this approach is that process modeling is no longer solely performed by a team of specialists. The responsibility for model construction and update is delegated to various individuals throughout the organization. The intent is to enable a relatively small team of model coordinators to effectively carry out overall administration and coordination functions required for modeling activities in a very large project.

3.1.1 Task Decomposition

The coordination team kick-starts the modeling exercise by constructing the first level matrix through information collected from a small group, usually comprised of high-level management possessing a broad view of the development process and its goals. The top-level DSM would typically contain roughly 10 to 20 major activities or overall program milestones. In addition to deliverables, outcome of decision gates or various types of authorizations may be included in the list of input and output information elements for each entity in this matrix. After collecting data through group meetings or interviews, the model coordination team enters the first DSM in the web-based system.

From this point onward the modeling activity can be delegated to a larger group of professionals. This is accomplished by assigning the decomposition of each task in the top-level matrix to suitable individuals in the organization. The Level 2 series of matrices

would typically cover around 200-300 tasks collected from ten to twenty mid to high-level managers in the organization. Each manager is responsible for the entry and update of required tasks and deliverables in the web-based system.

The process of decomposition through delegation may continue until the desired level of detail is reached in all areas of the model. At each level, participants can break down any task in their DSM, either on their own or by appointing the decomposition to others in the organization. Each sub-level DSM can easily be assigned to the person most familiar with the dynamics of the process in question. Table 3-1 presents a summary of estimates on the number of tasks and individuals involved in constructing a 4-Level model.

	Level 1	Level 2	Level 3	Level 4
Approx. number of tasks being modeled	10 - 20	200 - 300	3,000 - 4,000	40,000+
Approx. number of participants involved in data collection	5 - 7	10 - 20	200 - 300	1,000+
Approx. Modeling Scope	Overall Project	Major groups/divisions	Small teams	Individuals

Table 3-1 Estimated[1] magnitude and scope of modeling effort at each level

In order to ensure that a consistent and adequate level of detail is provided, participants are instructed to decompose each activity into a matrix of between 10 to 20 sub-tasks. Modeling experience at Boeing [ref. notes] has shown that this is an ideal policy. DSM's of 10x10 magnitude or less tend not provide the adequate level of detail and their activities should either be incorporated in a higher level DSM or further refined. Matrices having more than 20 tasks attempt to provide too much detail for the level of abstraction in question. Their "surplus" activities should be further clustered or moved to the next level of decomposition.

The top-down approach is used solely to initiate model creation at each level. A multi-tiered DSM is continuously refined through changes triggered in either direction. Assumptions made at a higher level can be challenged once further detail is available through task decomposition. Lower level matrices may reveal interactions not accounted for when developing higher tier processes. Two steps, as seen in Figure 3-1, characterize

[1] calculated assuming an average of 15 tasks per matrix at each level and a range of ±20% rounded

the creation of each new level. First, during task decomposition, the higher-level matrix ("Level n" in Figure 3-1) drives the creation of lower-level matrices ("Level n+1" in Figure 3-1). This is accomplished by specifying which tasks must be broken-down by which individual in the organization and, what input/output deliverables must be inherited by the new sub-tasks. Subsequently, during model refinement, information compiled in the newly created lower-level matrices may trigger changes in the parent DSM that could in turn require revisions in the decomposed sub-tasks. Therefore while model decomposition is top-down, model refinement is an iterative process between the parent DSM and its group of lower-level DSMs.

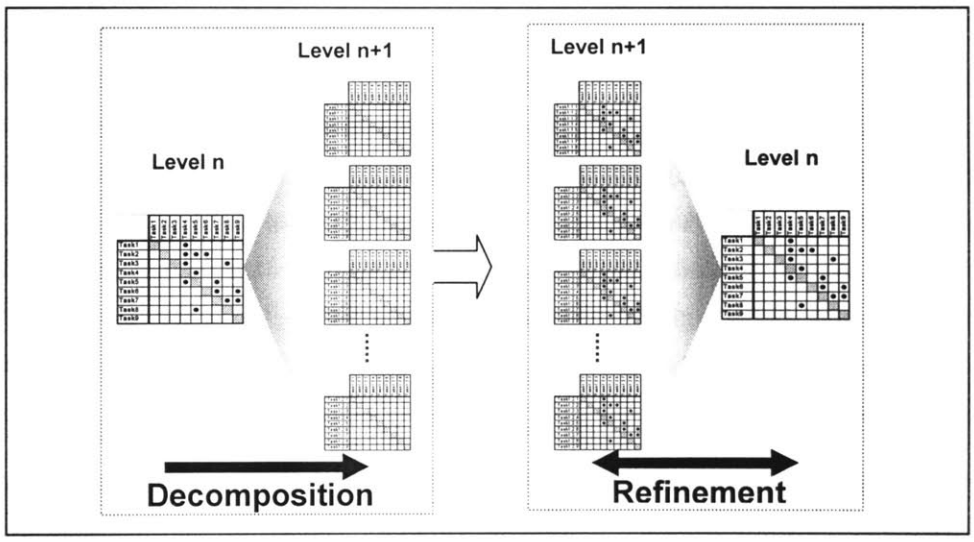

Figure 3-1 Change flow during DSM breakdown

A simple example can demonstrate this point. In a certain DSM, task 1 and task 3 interact by exchanging deliverable α. This deliverable is produced by task 1 and uniquely used by task 3. Suppose both tasks are decomposed therefore requiring sub-tasks 1.x (as output) and 3.x (as input) of the lower-level DSMs to inherit α. In this case, the individual responsible for the construction of the 3.x DSM concludes that deliverable α is not required. Consequently, the owner of the parent DSM not only must remove α from task 3 but also from task 1, the activity that produces α. Finally, this last change must also be reflected in the DSM breakdown of task 1 requiring the removal of deliverable α as an output of a sub-task 1.x.

In the above scenario, change was initiated by a lower-level DSM (3.x) leading to modifications to its parent DSM that in turn forced alterations to another lower-level-

DSM (1.x). As witnessed, the situation required interaction and intervention by three modeling participants at two levels, each responsible for separate but related DSMs.

All areas of the model are accessible to the user community in a read-only mode. The modeling team has editing rights to the entire model, while each participant is capable of modifications to matrices he/she originally created. This ensures needed discipline and control over potential changes to the model.

3.1.2 Automatic Notification

Internet technology is used in this approach to proactively engage users in the data collection process. The system's web interface allows users to initiate the decomposition process. Participants can navigate to their DSM (where they have editing capabilities) and select a task for breakdown. Once the delegation option is chosen, the user is asked to select a candidate that will be responsible for the construction the activity's sub-matrix. The system then automatically generates an e-mail to the target individual requesting his/her cooperation in the modeling activity. The e-mail contains information such as the user ID and password needed to logon the system as well as a hyperlink to the web-application's home page. Upon accessing the system, the user is greeted with information aimed at ensuring a full understanding of process modeling and its benefits.

A web page, acting as a personalized message board, is dynamically created for each participant in the modeling process. This page is utilized for communication purposes relating requests and displaying issues and announcements to users in the DSM modeling community. Upon login, the user is presented with details of the task decomposition request in his/her web page. When choosing to create the requested DSM model, the participant navigates through a series of screens that prompt him/her to enter all tasks and corresponding responsible individuals/teams in the process. Upon completion of task entry for the model, the user proceeds to the definition of existing interfaces by specifying input and output information elements for each outlined activity.

The e-mail notification mechanism is also used to remind participants of any follow-up adjustments to their DSMs or any additional information required to resolve model integration issues (discussed later in section 3.3). Details of all these requests are posted on the participant's web page with appropriate links to areas of the system designed to assist in the issue resolution process.

3.2 Usability

This section presents some of the visualization techniques used in the web-based tool to present the multi-tiered DSM model and to support its previously outlined data collection functions. To successfully engage users in the modeling exercise and promote the use of DSM methodology, the tool is equipped with an easy to operate user interface. A number of key features have been introduced to achieve this goal. These are summarized as follows:

3.2.1 DSM Layout

Users are able to quickly pinpoint the dependency represented by each cell in the matrix. As seen in Figure 3-2, when positioning the mouse over a particular cell, the two corresponding tasks are highlighted. In the illustrated example, the two tasks "Define High-Level Specs" and "Perform High-Level Design" are highlighted when pointing to the cell depicting their interaction. With this interface design, users no longer have to visually trace the cell's row and column coordinates in order to read a task dependency, an activity that becomes increasingly problematic with the growing size of the matrix being analyzed. Moreover, this approach eliminates the need to place coordinate labeling in the first row and column of the matrix.

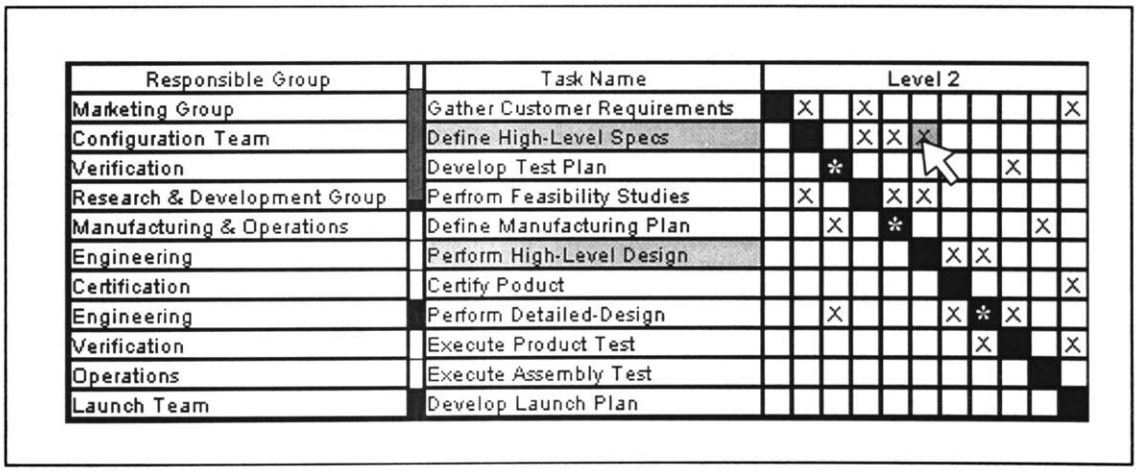

Figure 3-2 Sample DSM in Web-based tool

A visual distinction is made between feed-forward and feedback dependencies. When positioning the mouse over an iterative cell (one below the diagonal) the corresponding tasks are highlighted in red, while the color green is used for to highlight selected feed-forward dependencies (cells above the diagonal). The neutral color gray is used to when hovering over the "Task Name" column or diagonal elements in the matrix.

The individual or group responsible for the execution of each activity is presented in its corresponding row. For example, according to Figure 3-2 the activity "Gather Customer Requirements" is performed by the "Marketing Group", while "Perform Feasibility Studies" is the responsibility of the "Research & Development Group".

Activities are reported in the matrix from top to bottom according to the user-specified or computed order of execution. Clustered black and white stripes in the second (narrow) column indicate which tasks the user has planned for parallel execution. In Figure 3-2, for example, the user has indicated his/her intention of simultaneously executing tasks "Execute Product Test" and "Execute Assembly Test".

3.2.2 Inter-level Navigation

Marked elements in the diagonal of each matrix indicate that the corresponding task has been decomposed and therefore the existence of a lower level DSM. For instance, in Figure 3-2,"Develop Test Plan" and "Define Manufacturing Plan" have been further decomposed into detailed subtasks. By simply clicking on the marked diagonal element for each activity users are able to access its lower level matrix representation. Navigation through various levels of a model is restricted to the task decomposition scheme defined during data collection (see section 3.1.1). Accordingly, users are capable of viewing one matrix of approximately 10 to 20 tasks at a time.

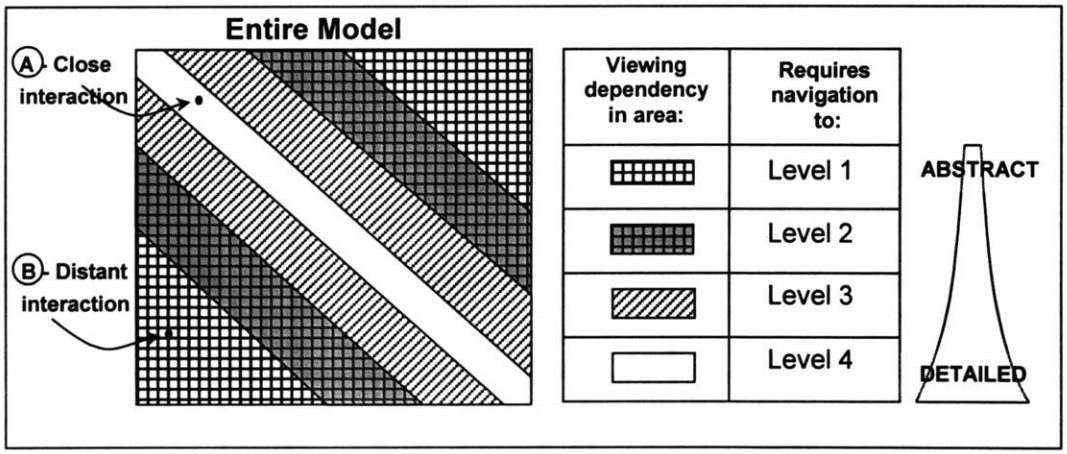

Figure 3-3 Sample 4-level dependency visualization

For every DSM, graphical representation of dependencies is limited to *internal* interactions (see section 2.4.1). At each level, exchanges between tasks in two separate matrices can be viewed by examining the relationship between their parent tasks at a higher level DSM. *External* interactions are therefore visible by climbing through the model's hierarchy of increasingly abstract activities. Figure 3-3 illustrates the visibility

range for dependencies found in different areas of a sample 4-level model. As seen in this figure, dependencies among closely scheduled tasks are easily visible at the most detailed set of DSMs (level-4). On the other hand, in order to view an interaction between two very distant tasks in the model one must navigate to the top level DSM (level-1). For example, point B in the model in Figure 3-3 represents an e*xternal* interaction between two distant level-4 matrices. By navigating to level-3 or level-2, the dependency continues to be *external* since it remains between tasks in two separate matrices at each level. However, at the top level DSM, the deliverable responsible for this information exchange is represented as an internal interaction and can therefore be seen by users.

In order to provide visibility to all dependencies among detailed tasks one must either display the entire matrix or allow users to zoom-in to any off diagonal location in the model. Displaying a DSM composed of thousands of tasks at a scale that is readable on a typical computer screen (800x600 resolution) is clearly not feasible. On the other hand, providing a window to off-diagonal locations in the model requires the presentation of different sets of tasks in the row and column positions. This would mean a departure from the consistent and intuitive n-square representation of matrices in the system. The dependency visualization topic is further discussed in Chapter 5 where alternative approaches are explained and analyzed in detail.

3.2.3 Auxiliary Screens

Two pop-up windows have been designed to provide a graphical representation of the information flow and assist users in editing the matrix content. These are especially useful in supporting the large portion of DSM-novice users by translating matrix dependencies into the more familiar graph flow representation. The screen in Figure 3-4 is used to display information requirements and output for a given activity. This window appears each time an activity in the "Task Name" column of a matrix is selected (see Figure 3-2). The task is presented in a box, with arrows indicating its defined information flow. Below each arrow a list box includes all required deliverables under the title "Inputs" and all outcomes under the title "Outputs". Authorized users (see Section 3.1.1) are able to add and remove deliverables or information elements from these lists. Arrows are only displayed when the list of input or output deliverables involved in the task information flow is populated.

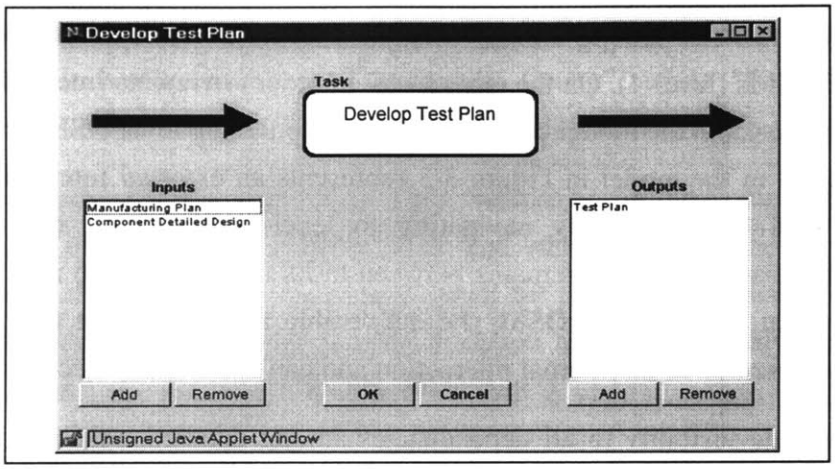

Figure 3-4 Sample auxiliary screen depicting task information flows

The screen in Figure 3-5, displays the task interaction pop-up window. This window is accessible by selecting any off-diagonal element in the matrix and is used to present a potential interaction between two tasks. An arrow indicates the existence and direction of an information transfer between the two activities. A list box labeled "Data Exchange" contains the names of specific information elements being exchanged. Information entities can be added or removed from this list (by authorized users only). To create a dependency a user can simply click on its corresponding off-diagonal element in the matrix and use this graphical interface to specify the data exchange between the two activities in question.

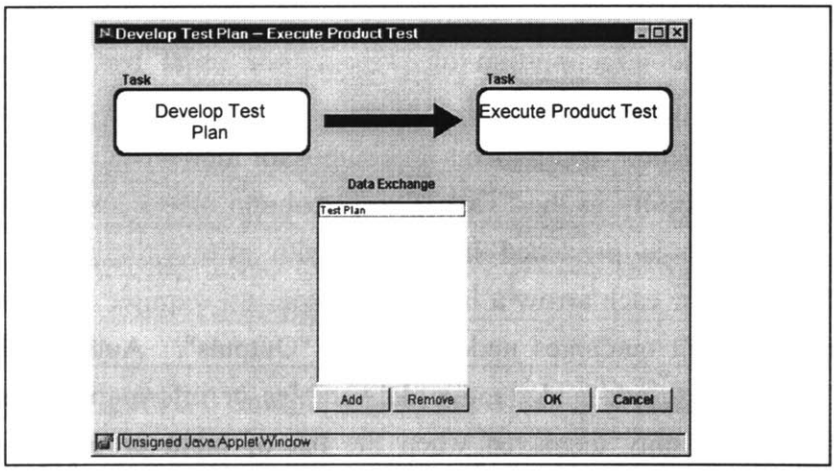

Figure 3-5 Sample auxiliary screen depicting task interactions

A pop-up window is provided to add, edit or delete deliverables or information elements in the model. This screen is displayed each time the "Add" option is selected in the "Input", "Output" or "Data Exchange" list boxes of the auxiliary windows. All

deliverables in the system's repository are listed alphabetically and, users are able to search for a particular information element in order to view its description for additional information.

3.2.4 Task Hierarchy View

The web-based system is capable of presenting the overall task decomposition scheme in an easy to operate interface. As seen in Figure 3-6, using this task-only view, users can navigate through the model by scrolling through the list of activities and obtaining further detail on their corresponding sub-task configurations. This screen provides a macro representation of the model clearly identifying the relationships between low and high level tasks. Its function is complementary to the DSM inter-level navigation (see section 3.2.2) in that it acts as a map for quickly pinpointing and tracing the location of a task in the model.

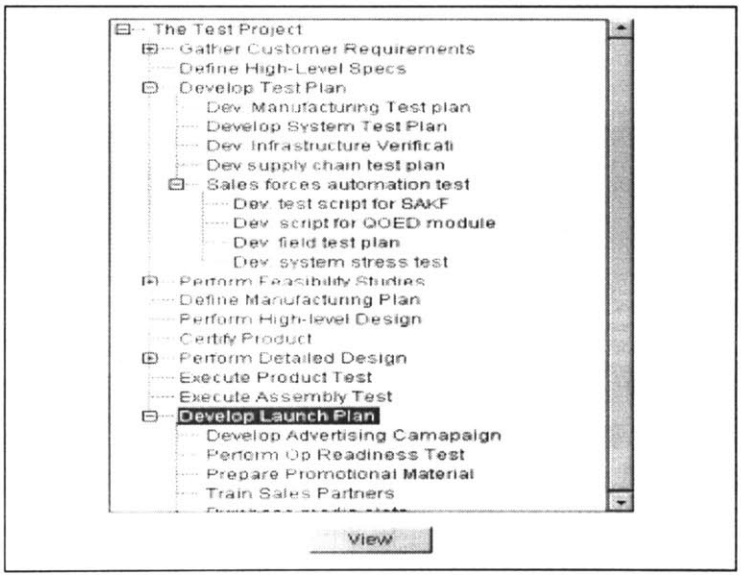

Figure 3-6 Sample task hierarchy view

Modeling participants can easily switch between the task hierarchy and the DSM view. Once a task is selected (for example, "Develop Launch Plan" in Figure 3-6) its corresponding DSM can be viewed by simply selecting the "View" option on the window. Similarly, users engaged in a DSM view can access the task-only representation with a single click.

3.2.5 Task Entry Interface

The task entry interface addresses the first step in DSM construction as outlined in section 3.1.1. It is comprised of a main screen and two auxiliary pop-up windows. Users access the main task entry screen by following the link in the model creation request posted to their web page. As shown in Figure 3-7, activities are added by clicking on the "Add" button and providing the task name and responsible individual information in the appropriate entry fields on the "Task Information Entry" auxiliary screen. Each added task is placed in the task list box of the main screen and given the sequential order by default. Users can easily modify the default sequence of execution by entering a new number in the "Sequence" column for each task. The same number must be given to activities that are planned for parallel execution in the model. The information entered for each task can be edited using another auxiliary pop-up screen. Activities can also be completely removed from the list. When task entry is complete, users advance to the next stage in data entry for information exchange definition by selecting "Save" in the main screen.

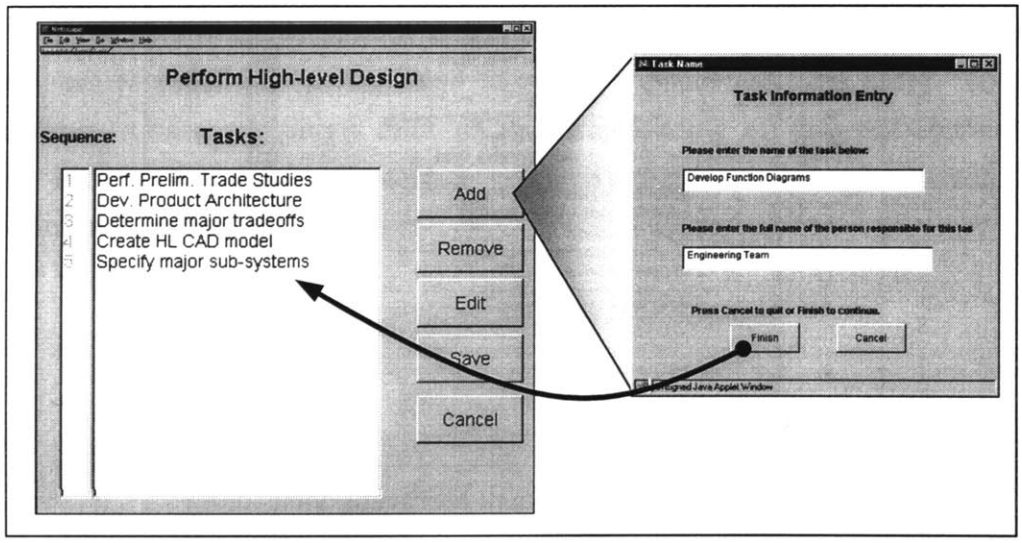

Figure 3-7 Sample task information entry sequence

This interface is modeled after the "wizard" approach frequently used in popular Windows applications as an effective way to engage novice users in data collection. In this case a concise view of the DSM tasks and their sequence in the process is presented in the main screen while supporting functions such as task addition, editing deleting are accomplished through auxiliary pop-up screens.

3.2.6 Personalized Message Board

This is an important interface as it addresses the core theme of collaborative problem solving during the modeling process. This page is created for each user engaged in the modeling exercise with the goal of presenting issues and requests that are received from various participants and the system's automated verification process. Communication is structured and presented through hyper-linked text for access to supporting screens and functions. Figure 3-8 displays a sample page containing both issues and requests.

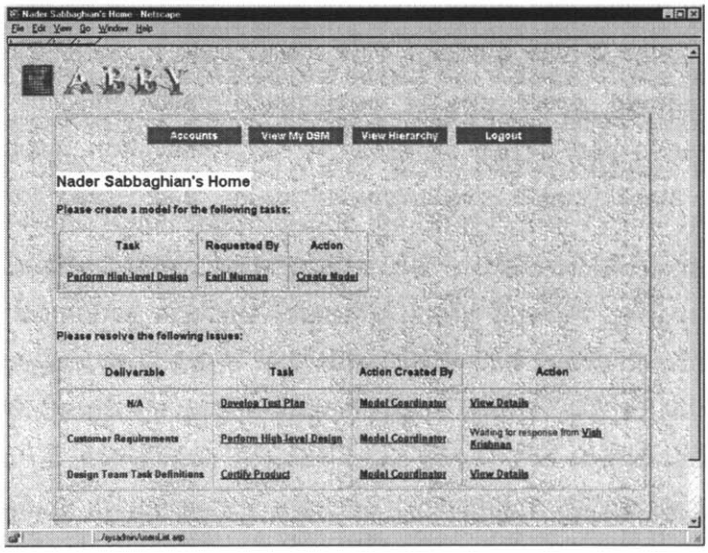

Figure 3-8 Sample user web page

Model creation requests are presented separately from the issue resolution records. The model creation request contains the name of the task being decomposed and the requestor's name. The task name is linked to its corresponding DSM allowing users to analyze the activity in question and its interactions at the higher-level matrix before engaging in data entry. The user name is linked to the requestor's e-mail address and, if selected, will launch the participant's default mail browser automatically placing "Model Creation" in the *Subject* and the requestor's e-mail in the *To* fields. This feature allows users to quickly contact the requestor for possible clarifications. The participant can proceed to the task entry pages (see section 3.2.5) by selecting the hyper-link "Create Model".

Issue resolution records follow a similar scheme with all displayed tasks linked to their corresponding DSM and all participant names e-mail enabled. Issues delegated to other members of the organization are presented with a message indicating their pending status (see task "Perform High-Level Design" in Figure 3-8).

For those having no pending issues or requests, the message board page is simply skipped during login. If users choose to access the page from other areas of the system, a brief message is displayed informing them that there are no outstanding issues or requests.

3.3 Model Integration

Integrating information received at different times from a large group of dispersed individuals is a major challenge in this approach. One of the biggest issues is the general lack of a commonly understood terminology for teams, deliverables and tasks, the basic ingredients of the *Data-Driven* model. This often leads to inconsistencies requiring further interaction among participants for a negotiated resolution. The system's goal is identify such scenarios and devise an effective communication mechanism for such negotiations to take place through exclusive involvement of necessary participants.

The proposed prototype addresses two major integration issues labeled *Data Disconnect* and *Inter-level Disparity*. Several batch processes perform periodical analysis of the collected data in order to detect such inconsistencies. A set of procedures and interfaces are then responsible for facilitating issue resolution through direct user involvement. Each scenario is explained as follows:

3.3.1 Data Disconnects

This occurs when a task's output is not an input to any other task in the process. In essence, there are no customers for the information being generated. Similarly, a Data Disconnect occurs when a task's input is not generated by any other activity in the model. In this case information is requested that is not being produced anywhere in the defined process. This clearly does not apply to boundary interactions (see section 2.3) where the input and output information exchange takes place with entities outside the model. The above situations can be attributed to one of the following:

3.3.1.1 Nomenclature

Interaction has been defined but not detected because of the difference in terminology used for the same information element. For example, Task A defines "Center of Gravity" as an information requirement while Task B indicates " C.G." as it's deliverable. Clearly, this nomenclature difference needs to be resolved for this interaction to appear in the matrix (from Task B to Task A).

3.3.1.2 Timing

Disconnect is present because of other participants' delay in providing data for the modeling exercise. It is unlikely to expect a coordinated response from all participants once data collection requests have been e-mailed. Clearly, users assigned to processes that are poorly understood or never before documented require additional time for data entry. Others may require more time to familiarize themselves with the DSM methodology and the web-based system's interface and functions.

3.3.1.3 Information obsolescence

For output data disconnects the scenario marks the existence of a process inefficiency. The output information element is deemed obsolete since there is no evidence of its use by another task in the process. This may be caused by changes to certain activities' required deliverables. It may also indicate the presence of a historically relevant, but at that time unnecessary deliverable.

3.3.1.4 Information omission

For input data disconnects the scenario relates to the absence of an activity responsible for the creation of information deemed necessary. This also flags the existence of a process inefficiency. If undetected, required information will not be available at the time of task execution leading to delays in the process.

3.3.1.5 Incomplete model

When modeling a process for the first time or during task decomposition it is difficult to fully define all activities and their scope. Therefore tasks and deliverables are sometimes left out of the model by mistake.

3.3.2 Online Issue Resolution

Upon detecting a Data Disconnect, the prototype system creates an "issue" record for the affected participant. Users with outstanding issues are periodically notified via e-mail to request their participation in its resolution. The personalized message board, accessed by each user upon login, presents each participant with his/her data disconnect issues. By selecting the appropriate link on this page the data disconnect screen is presented detailing in words and graphics the missing dependency as seen by the example in Figure 3-9.

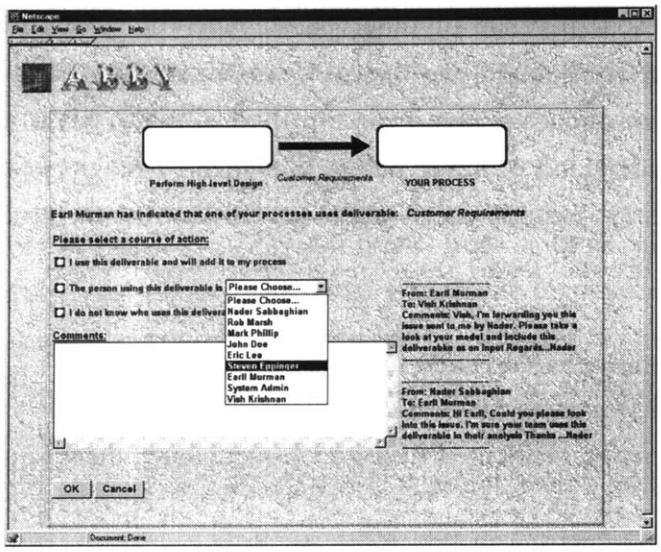

Figure 3-9 Sample data disconnect resolution page

The user is asked to engage in issue resolution through one of the following courses of action:

3.3.2.1 Online discussion

The user identifies the participant likely to be the recipient or provider of the information element in question. This option triggers the creation of an "issue" for the specified participant who is then drawn into the on-line discussion through the same notification mechanism and web interface. Users are required to enter a comment each time this option is chosen. All comments are stored in the "issues" repository and are presented to users each time an issue is evoked. The history of the issue includes all comments as well as the e-mail hyper-linked name of individuals involved in various stages of the issue resolution process.

3.3.2.2 Adjustment to existing model

The data disconnect can be addressed by the participant without the need for other users' intervention. By choosing this option, the participant is transferred to the affected matrix where he/she can proceed with modifications necessary to resolve the issue.

3.3.2.3 Delegation to modeling team

When the first two options are not applicable, the user can pass the issue to the model coordination team. The resolution of the issue in question will then be delegated to one of the model coordinators. Providing comments is also mandatory if this option is

chosen. Comments provided by the user are once again attached to the issue for future reference during issue resolution.

3.3.3 Inter-level Disparity

Interactions reported in the multi-tiers of matrices must be consistent across different levels. Once a task is decomposed, the lower level matrix must inherit as external interactions (see section 2.3) the input and output information flow of the parent task. In essence, the lower level matrix provides additional details on the parent task and is therefore required to be consistent with its information interactions. Figure 3-10 presents a sample decomposition of a task with its input and output information requirements. As seen in this figure, information elements **ß** and **ø** used and produced by Task 5 must also be present as inputs and outputs in the decomposed (lower level) matrix.

Upon detecting an inter-level inconsistency the system generates an "issue" for the participant responsible for the lower level matrix. The participant is then notified via e-mail and presented with the issue upon logon through his/her user page. The problem is clearly explained and the user is asked to either add the missing information element(s) to the lower level matrix, or resolve the inconsistency by contacting the owner of the parent task. In fact, discussions may lead to the resolution of the issue by simply removing the data element(s) in question from the parent task.

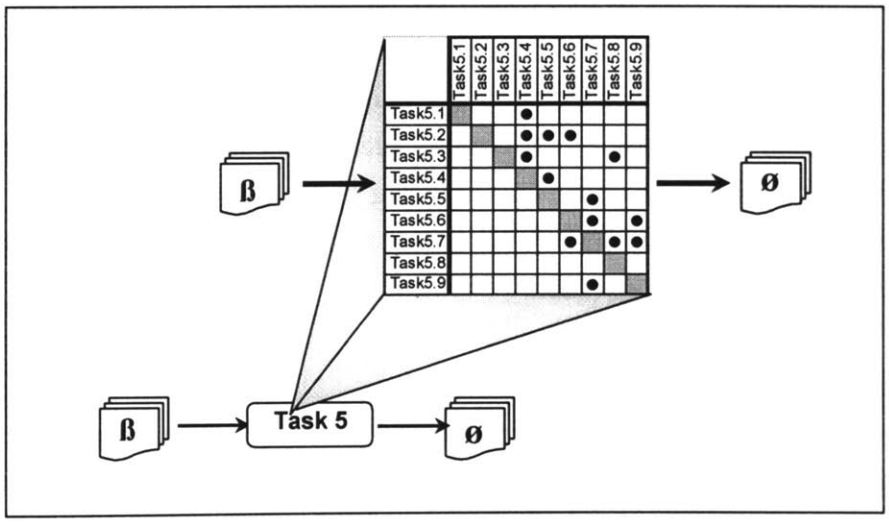

Figure 3-10 Deliverable inheritance during decomposition

There are also cases where task decomposition leads to the discovery of new external interactions. These interactions are initially unknown to the process modeler for the higher level matrix and are exposed as a result of a more detailed analysis through

decomposition. The system identifies such instances and proceeds with its resolution by simply adding the newly found information elements to the list of inputs and outputs of the parent task. The owner of the parent matrix is notified of the change in the model through his/her user page upon system logon.

3.3.4 Data Entry Validation

To ensure that a number of basic modeling guidelines are followed, the data collection interface performs a first pass, on-line validation on entered information. Examples of such data entry guidelines are presented as follows:

3.3.4.1 DSM dimension

Users are required to limit the number of tasks presented in a matrix to 20. This measure is critical to ensure that each layer in the decomposition represents a reasonably comparable level of abstraction thus preventing users from combining detailed and high-level tasks.

3.3.4.2 Redundant output deliverable

Two tasks are prevented from generating the same deliverable. Users are notified when attempting to add as output a deliverable being already produced by another task in the model. In this case, additional information is provided to assist a participant in resolving the potential conflict. This includes facts on the source of output and the individual currently claiming ownership of the data/deliverable in question. It must be noted, however, that not all scenarios point to the existence of redundancy in the process. Certain information is sometimes deliberately created by multiple sources in order to capture different opinions or obtain results using distinct approaches. Such cases can easily be incorporated in the model by simply assigning different names or version numbers to the deliverable in question.

3.3.4.3 Field size

In order to present captured data in a consistent and readable format users are invited to provide short-names for fields exceeding limits set by the system's graphical display. For example, a limit of 30 characters is placed on "Task Names" in order to adequately fit a 20x20 matrix in the lowest acceptable screen resolution (800x600 pixels).

3.4 Summary

This chapter presented a distributed and asynchronous knowledge collection and presentation approach through the use of Web-based technologies and a multi-tiered Design Structure Matrix (DSM) configuration. The method relies on the direct involvement of a large group of geographically dispersed participants for the creation and continuous update of the matrix-based model. The section outlined various user interface features developed to provide an improved process analysis framework to project managers, team leaders and other product development users engaged in the modeling exercise. Functions designed to resolve complex integration issues during data collection were discussed in detail, and their role in facilitating on-line interactions among DSM data collection participants were highlighted. This chapter is a useful synthesis of the major ideas and concepts explored in this research to address the obstacles outlined in chapter 2.

4 WEB-BASED PROTOTYPE

This chapter outlines the steps leading to the development of the web-based software concept. It attempts to provide a clear picture of the prototyped solution, its functionality and technical architecture. The software development environment is also briefly discussed.

4.1 Requirements Analysis

Several sources have been used in this research to gain an understanding of user needs for a large-scale distributed process modeling and project-planning tool. Middle to top-level managers primarily involved in project planning and scheduling are considered the target users for the system. A group of such users were interviewed at the Boeing Commercial Airplane Group during the summer of 1997. The discussions led to significant insight to existing planning practices as well as challenges faced in managing new and derivative commercial airplane programs. These are typically multi-billion dollar engagements requiring the participation of thousands of professionals over a period of several years. The commercial airplane programs therefore provide an ideal setting for the study of large-scale, complex product development projects requiring a notable amount of coordination and planning.

Discussions with Dr. David Grose and members of his CFID (Cross Functional Integrated Design) team were another very useful source for requirements analysis. The CFID team has had extensive experience in DSM-based process modeling at Boeing and has developed DSM-based tools for process analysis. Through this interaction the author was able to grasp the major challenges faced by DSM process modelers as well as the strengths of their existing approach. The author was able to observe the CFID team in action by attending process modeling team meetings and interviewing users previously involved in DSM-based modeling.

4.1.1 Requirements Definition

All the above information led to the development of eleven categories of high level requirements, each explored in further detail. These categories with their corresponding set of detailed needs are presented in Appendix A. Each high-level requirement is also briefly described as follows:

I Access

The data collection effort requires the participation of a large number of individuals. All users provide data directly to the system and will therefore require access. Data is collected and presented to individuals in different geographical locations. All features of the tool are available to remote access users. Users are able to access the same version of the tool with all its features from all PC computing platforms. System access is hardware independent requiring only access to the company Intranet and the capability to run specified Web browsers.

II Data Capture

The system is capable of collecting Task related information as well as information required to establish dependencies among tasks. The data capture functionality is scaleable and is able to support very large development models (in the order of thousands of tasks) as well as small ones (under 100 tasks). The system eliminates the need for cross-functional team meetings currently required for DSM data collection. Data collection is performed in a distributed and asynchronous mode. The system is able to identify users that need to be consulted on a particular issue and contact them to collect the necessary information. The system is equipped with an intuitive and user-friendly graphical interface for data collection. No formal (instructor) training is required to become proficient with the system for data entry. A hierarchical scheme is adopted to integrate the vast amount of tasks being collected and the required level of abstraction (whether high-level tasks or detailed tasks) is clarified to users during data collection. Data syntax and type validation is performed and the system is able to ensure that only recognized data elements in the correct format are entered. The system provides an acceptable performance (response time) during remote data collection.

III Interfaces

The system is capable of interfacing with common scheduling software as well as existing DSM task based modeling tools.

IV Presentation

The system can clearly present large matrices (on the order of hundreds of thousands of tasks) and provide users with maximum viewing flexibility. The system is capable of intuitively presenting inter-task information flow as well as

feed-forward and feedback scenarios. Hierarchical task roll-up and alternative views of the process are provided to assist users in the analysis.

V Reporting

Users are able to print all available DSM views as well as all information displayed through the system's interface. Matrices can be graphically exported for integration into presentations and reports.

VI Validation

Users benefit from a dictionary of existing project deliverables when entering information. They could also add new deliverables and attributes to the dictionary. The system is capable of detecting inconsistencies in the model and reporting the nature of the problem to the appropriate process stakeholder. First level validation (syntax, field length, field type etc.) is performed upon data entry.

VII Security

The system is capable of providing an environment for secure transactions over the Internet and is adequately protected against break-ins.

VIII Search/Retrieve

Users are able to perform a variety of data inquiries such as deliverable usage, list of tasks for a specific group or individual, history of modifications to the data etc. The system provides an acceptable response time during user search/retrieve operations.

IX Analysis

Sequencing can be performed on captured model in order to minimize iterations in the process. Users are able to adopt the resulting sequencing scheme by saving the new (re-sequenced) model. The system is able to easily accommodate existing sequencing algorithms. Impact of changes to the process are automatically analyzed and presented to users ("what if" capability)

X System Maintenance

The system is equipped with administration functions that provide access to all reference data as well as user management.

XI Performance Support

Users are able to utilize on-line support features and help functions to become familiar with the system's functionality and interface.

4.2 System Metrics and Specifications

Due to this research project's time and resource constraints the above outlined list of requirements (also see Appendix A) was further reduced to a subset of high priority needs to be addressed through the development of a prototype system. Each requirement was ranked according to its importance in addressing the research question primarily focusing on the data collection coordination and DSM-based visualization aspects. A subset of 30 requirements (out of a total of 54) was chosen to define the scope of the prototyping exercise. These requirements were subsequently analyzed in detail in order to develop specific set of specifications for the software development process. First, a list of metrics was drafted and the relationship of each metric to the requirements was determined. The resulting needs-metrics analysis can be found in the matrix in Figure 4-1. The rows of the matrix correspond to needs considered within scope and the columns of the matrix correspond to metrics used to determine the degree to which the system satisfies such needs. This exercise served to ensure that all requirements be addressed and provided the team with an intuitive visual representation of the impact of each metric on requirements during the design phase.

The development of system specifications was the last step in the process. A set of assumptions was made regarding the system's operating environment, users' computing environment and process modeling dynamics. All available information was then synthesized to come-up with target values for each metric (results are presented in Table 4-1). For example, it was assumed that users employ a graphical e-mail package (such as Eudora or MS Mail) which enables the system's notification mechanism to present a WWW hyperlink in the message body. Other examples include assumptions regarding the users' minimum screen resolution, reasonable upper and lower limits on the size of each DSM and number of users simultaneously accessing the system. The target specifications presented in Table 4-1 are the first point of reference in the development of a feasible concept for the distributed web-based system.

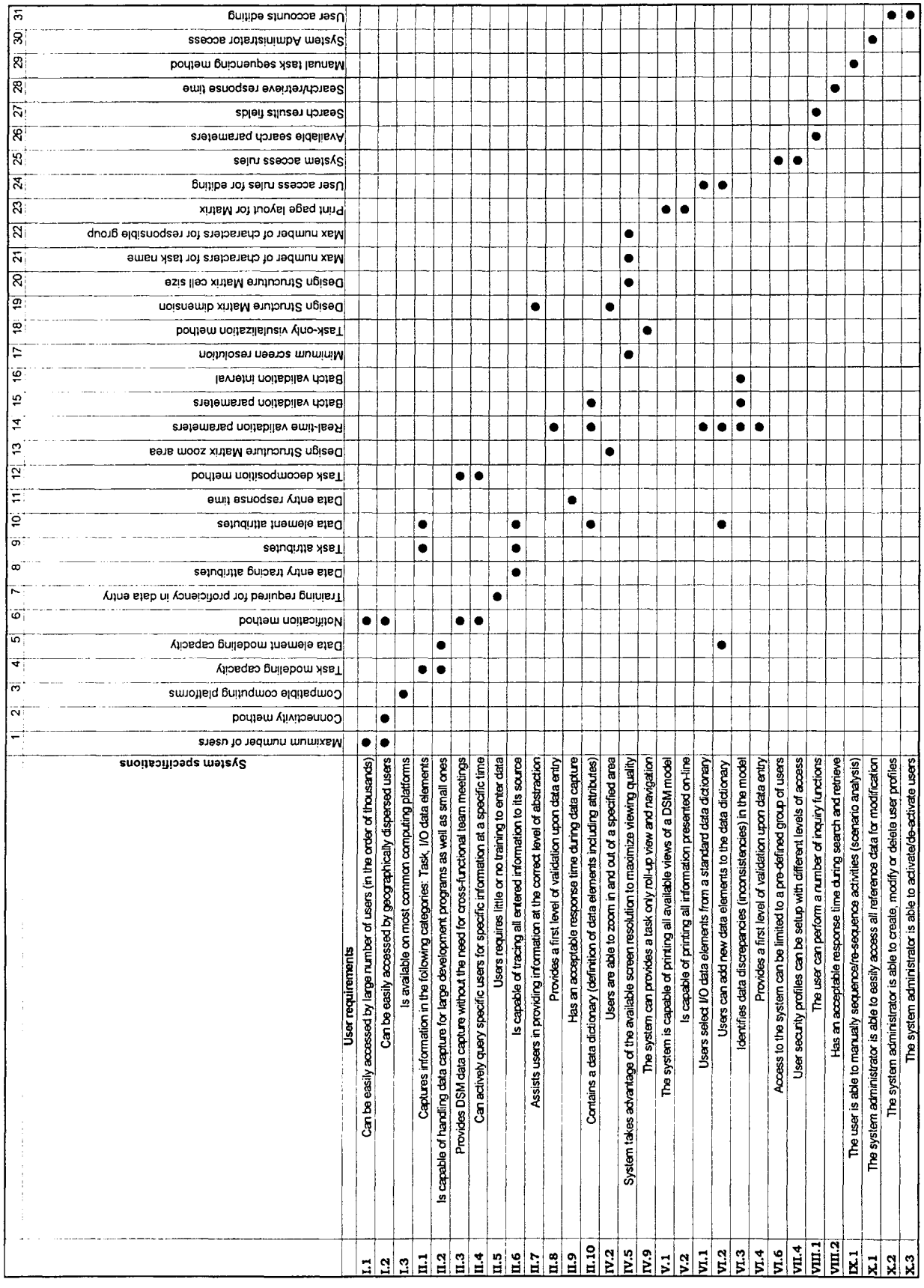

Figure 4-1 Requirements-Specifications matrix for Web-based system

	System specifications	Requirements	Target
1	Maximum number of users	I.1, I.2	Less than 2,000 users
2	Connectivity method	I.2	Internet/World Wide Web - Network, dial-up connectivity
3	Compatible computing platforms	I.3	All major platforms: Windows, Macintosh, Unix, Sun OS
4	Task modeling capacity	II.1, II.2	Less than 100,000 tasks
5	Data element modeling capacity	II.2, VI.2	Less than 100,000 data elements
6	Notification method	I.1, I.2, III.3, III.4	Electronic mail with hyperlink to Web-system
7	Training required for proficiency in data entry	II.5	Less than 15 minutes of self-training
8	Data entry tracing attributes	II.6	User ID, time and date of transaction
9	Task attributes	II.1, II.6	Name, description, task executor (team or individual), task abstraction, task sequence of execution
10	Data element attributes	II.1, II.6, II.10, VI.2	Name, description, creator's user ID
11	Data entry response time	II.9	Less than 5 seconds
12	Task decomposition method	II.3, II.4	Performed by task owner or delegated to another user for decomposition
13	Design Strucuture Matrix zoom area	IV.2	Zoom areas centered around DSM's diagonal
14	Real-time validation parameters	II.8, III0, VI.1-VI.4	Duplicate data element outputs, field length & type validation for data elements & tasks, matrix edit ownership, comments entry for issues, login-ID and password verification
15	Batch validation parameters	II.10, VI.3	Data disconnects (input and output), inter-level disparity(top-down,bottom-up)
16	Batch validation interval	VI.3	Every 2 business days
17	Minimum screen resolution	IV.5	800x600 pixels or higher
18	Task-only visualization method	IV.9	Using MS file explorer graphical model
19	Design Structure Matrix dimension	II.7, IV.2	Each matrix is comprised of 10-20 tasks
20	Design Strucuture Matrix cell size	IV.5	12x12 pixels
21	Max number of characters for task name	IV.5	Shortname 30 characters; Longname 50
22	Max number of characters for responsible group	IV.5	Shortname 15 characters; Longname 40
23	Print page layout for Matrix	V.1, V.2	Web browser-enabled printing
24	User access rules for editing	VI.1, VI.2	Each Design Structure Matrix can be edited by no more than one user; data elements are universally accessible for editing
25	System access rules	VI.6, VII.4	Only users with valid ID's and password are allowed access
26	Available search parameters	VIII.1	Data element name in the data dictionary
27	Search results fields	VIII.1	List-box, alphabetical order
28	Search/retrieve response time	VIII.2	Less than 5 seconds
29	Manual task sequencing method	IX.1	User defined by assigning ascending numerical values to indicate sequence
30	System Administrator access	X.1	System administrator has univeral access to all data in the system
31	User accounts editing	X.2, X.3	System administrator is the only user to manage accounts

Table 4-1 Target specifications for Web-based system

4.3 High-level Design Concept

Requirements-specifications analysis led to the development of a prototype concept, which can be characterized by three top-level components as shown in Figure 4-2. The user interface sub-system includes all the necessary modules designed to collect modeling data from participants and display information in structured formats. Activities such as integration of data received from multiple participants, user notification and routine data maintenance are performed through a set of periodically scheduled batch processes. The Data Repository interfaces with the other two components and is designed to efficiently store all data necessary to support the modeling activity.

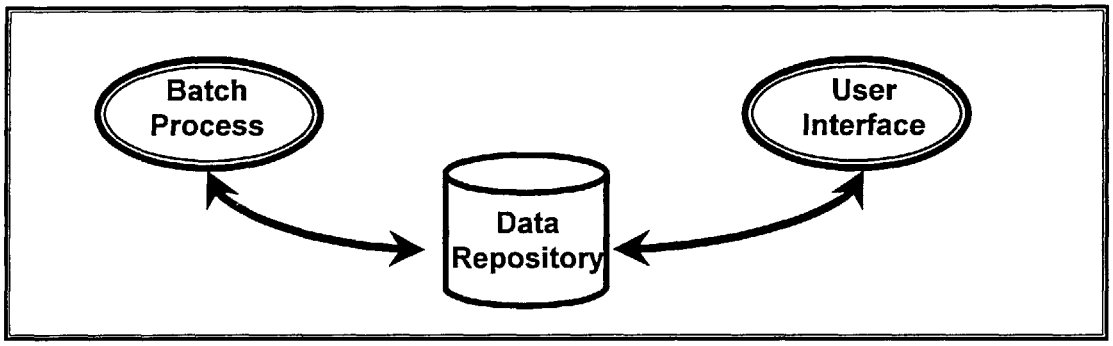

Figure 4-2 Top-level view of prototype system

An overview of each of each of the above sub-systems is provided in the following sections.

4.3.1 User Interface

Comprised of a series of interactive web pages with embedded and stand-alone Java components for advanced user interface functionality (see Figure 4-3). This set of linked screens provide a graphical interface for users to intuitively engage in the following activities:

- login system and manage user accounts
- view, add and edit DSMs
- provide information aimed at resolving integration issues
- obtain introductory information on the prototype and DSM-based process modeling

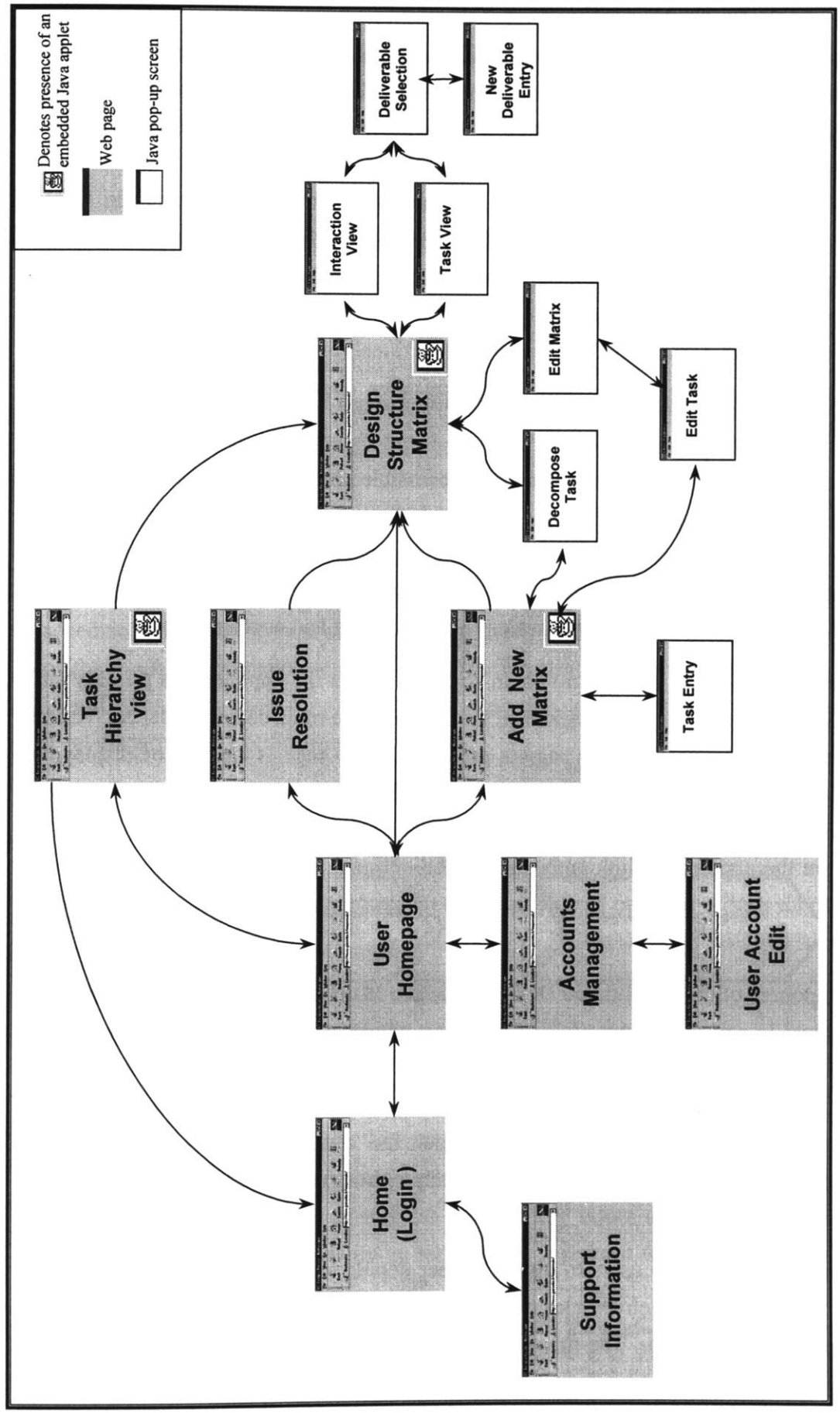

Figure 4-3 User interface screen flow for Web prototype

Figure 4-3 displays the system's overall user interface design. Gray panels represent interactive web pages developed with or without embedded Java applets. The embedded applets are active components delivering additional interactive functionality within the same web page. Certain parts of the system require a higher degree of user interactivity and thus are prime candidates for the implementation of embedded applets. This approach eliminates the need for re-loading a fresh web page each time a new view (data representation) is requested by the user. In addition, a series of Java applet windows (white panels in Figure 4-3) are designed to support the embedded elements. These standalone (pop-up) screens further extend the functionality of each web page and deliver a familiar look and feel of a typical graphical user interface (GUI) windows-based application.

Introductory information on the system is accessible through its home page. This material is aimed at explaining the system's overall functionality and providing background information on the process modeling technique used. The goal is ensure that first time users become acquainted with the basics of DSM-based modeling and its potential benefits in the area of large-scale project planning and coordination.

Users access the system through the login screen. This is the application's primary site and a link to its address is provided in all automated e-mail correspondence. Upon successful login, the user's home page is presented. This page is capable of displaying hyperlink-enabled messages specific to each user. Requests, issues and action items are posted to this page and participants are asked to take action or explore further details by clicking on the appropriate link embedded in the outlined message. Users with System Administrator profile will also be able to access the account management screens in order to add, edit, or delete active users.

Each component of the multi-tiered Design Structure Matrix is presented as an embedded applet linked to series of supporting Java pop-up windows. Matrices are constructed by isolating all tasks that share the same parent task (or no parent task in the case of the top-level matrix) and analyzing their information exchange in order to place the appropriate marks in the DSM. The system matches input and output deliverables among task groupings and translates each interaction into its corresponding matrix coordinates for DSM visualization.

The pop-up screens allow users to obtain further details on inter-task information flow, edit the matrix, add new deliverables and delegate task decomposition, all from the same web page (see Figure 4-3). The functionality of the matrix and its supporting Java pop-

ups is explained in detail in Chapter 3.2. The task hierarchy view (see Figure 3-6) provides a faster alternative for navigating through the model. The entire model's task decomposition scheme is presented as a tree structure through an embedded applet without any additional information on each task (such as deliverable I/O, ownership etc.). Data retrieval is therefore limited to task names and relative location in the model hierarchy. Users can quickly move through the tree structure, select a particular task and easily access its corresponding DSM for further detail and analysis.

A series of web pages are designed to guide the participant through the issue resolution process. These interfaces are designed to explain the issue at hand, provide participants with a number of options and capture their feedback for further analysis.

4.3.2 Batch Process

The system's batch process is comprised of five steps as outlined in Figure 4-4. The components are all written in Java and built into a Java application capable of running on any computing platform. The process can be executed manually or as a regularly scheduled job on the system's host server. Access to this process is restricted to members of the model coordination team. Several data collection and validation functions were identified as being most suitable for batch execution, among these data-disconnect, inter-level disparity and notification mechanisms.

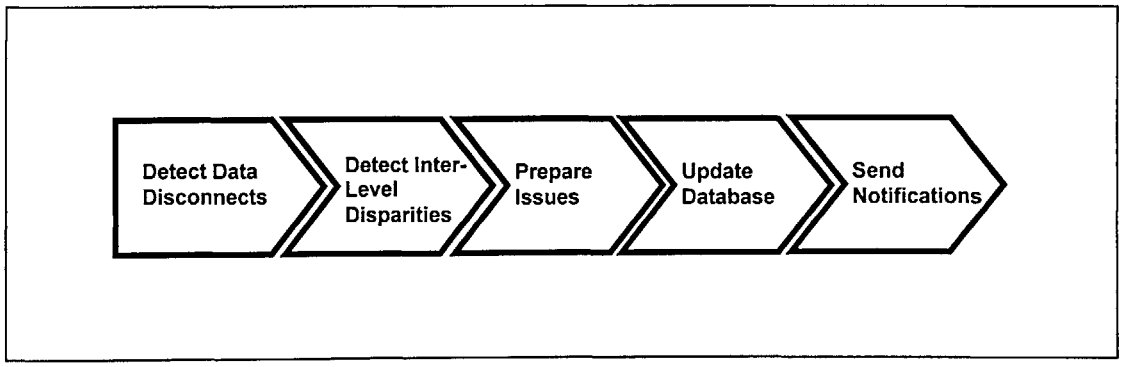

Figure 4-4 Batch Process Flow Diagram

It is logical to assume that users will not provide the requested data simultaneously. The challenge in integrating such time-fragmented arrival of information is clear, especially when the validation process attempts to identify key issues and facilitate direct contact between parties capable of resolving them. Once a request for data collection is made, different time delays are expected with some users responding immediately, others in a few hours or perhaps days. Since data validation for integration purposes depends on the

timing of the arrival of information, it can not be performed in a real-time. Users must be given enough time to respond to a data collection request before the system attempts to validate the overall model for data-disconnect identification. It would be best to minimize the data disconnect scenario outlined in part (b) of section 3.3.1.2 (related to timing) by ensuring that users have been given a reasonable chance to respond before initiating follow-up e-mail inquiries for validation purposes.

A similar argument applies to the Inter-Level Disparity case (see section 3.3.3). One can not assume that users will provide all the modeling information required to construct a DSM in a single data entry session. Participants may initially only provide partial information about their process and require further consultation with their peers to complete the full picture. Above issues together with a lack of real-time criticality in a process modeling tool clearly point-out that multi-user validation functions such as those discussed in sections 3.3.1 and 3.3.3 are best performed periodically.

The five steps in the batch process are explained in their sequential order of execution as follows:

4.3.2.1 Detect Data Disconnect

The system scans through output deliverables of every task in the model attempting to find a matching input deliverable to complete a dependency. A straight string comparison is performed in this search. Once the entire repository is traversed, task-deliverable combinations left without a companion are marked as input or output data disconnects (see section 3.3.1). The process also determines the nature of the dependency by examining the relative position of the origin and destination tasks in the model. Dependencies among tasks with the same parent task are marked as *internal* while all others are tagged as *external* (see section 2.4).

4.3.2.2 Detect Inter-Level Disparities

Inter-level disparities were discussed earlier in section 3.3.3. Task decomposition requires consistency of dependencies among activities at various levels. The batch process checks the information flow of each decomposed task to ensure that an instance of each deliverable exists at the lower-level DSM. Instances of high-level deliverables not inherited by lower-level DSMs are flagged as top-down inter-level disparities. Adjusting this inaccuracy requires the addition of missing deliverables to the suitable sub-task at the lower-level matrix. It therefore requires the involvement of the participant responsible for modeling the DSM in question.

The same inheritance consistency check is performed for parent DSMs. *External* interactions among lower level DSMs are analyzed to ensure that an instance of the information exchange exists at their corresponding parent tasks. Adjusting the model for this bottom-up inter-level disparity does not require user intervention because the task with missing information is clearly identifiable by the batch process. Therefore, deliverables engaged in *external* interactions but not recognized by their parent tasks are automatically added to the parent activity's list of inputs or outputs.

Instances of interactions between high-level and low-level tasks are also addressed in this process. These are situations where a deliverable is exchanged exclusively between tasks residing in two separate levels in the model. Deliverable β in Figure 4-5 is an example of such scenario. During the first step of the batch process (section 4.3.2.1) such interactions are flagged as *external* since they are simply diagnosed as connections between tasks not belonging to the same DSM. Once this is done, the bottom-up inter-level disparity is detected and the parent task is updated with the missing deliverable.

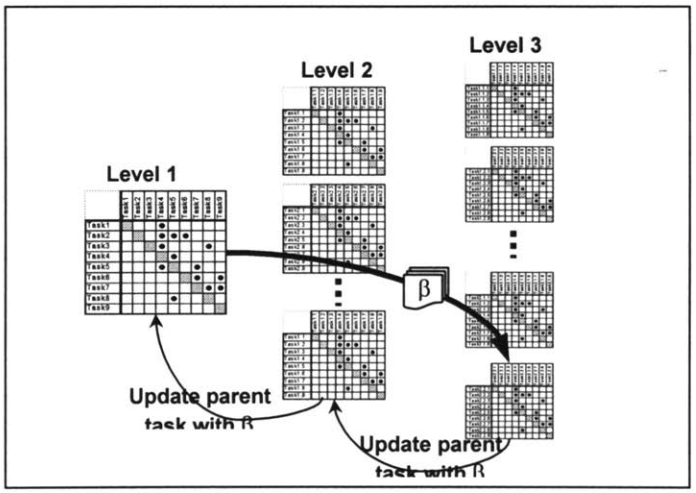

Figure 4-5 Addressing inter-level disparity caused by inter-level interaction

The process may entail multiple updates across the model hierarchy as shown in Figure 4-5. In this example, deliverable β being exchanged between tasks in a level-1 and level-3 DSM causes the batch process to record the presence of an *external* interaction. Next, since β is not found among the inputs and outputs of the parent task to the level-3 DSM, it is added to the level-2 activity. The process repeats itself because of the newly created inter-level interaction between levels 2 and 1 causing another β update, this time to list of deliverables of the parent task to the level-2 matrix.

Such interactions among high-level and low-level tasks typically occur when an unacknowledged higher level dependency is surfaced at the lower-level matrices through model decomposition.

4.3.2.3 Prepare Issues

For each issue detected by the first two phases of the batch process a record is prepared for submission to the data repository. This record contains information required for the issue resolution process such as the target user and data on the deliverable and task in question. Each issue is assigned a unique issue ID for tracing purposes. Anytime an issue is delegated to another individual in the organization a new record is created and is easily traceable to its parent through the issue ID.

Code	Description	Created by
01	Input data disconnect (input deliverable not produced in the model)	Batch process
02	Output data disconnect (output deliverable not used in the model)	Batch process
03	Top-down inter-level disparity (decomposed DSM missing inherited deliverables)	Batch process
04	Bottom-up inter-level disparity (parent DSM missing lower-level deliverable)	Batch process
06	Delegated issue	User delegation via issue resolution interface
09	Model creation request	User delegation via task decomposition interface

Table 4-2 Recognized issue types

The web-based system utilizes a set of six codes to identify the issue type. As seen in Table 4-2, four of these are created during this stage of the batch process and their corresponding codes incorporated in the issue record.

4.3.2.4 Update database

The previous steps in the batch process affect a number of tables in the data repository. All necessary database maintenance activities are performed at this stage. Deliverables are added to list of input/outputs of decomposed tasks as indicated by the bottom-up inter-level discrepancy analysis. Task dependencies are updated to reflect the correct

interaction type (whether *internal* or *external*). Existing issue records are revised and their status updated. Newly created issues are added to the repository.

4.3.2.5 Send Notification

The last step in the web system's batch process is the notification mechanism. The system scans the list of existing issues and obtains information on the contact person for each. An e-mail is compiled (as seen in Figure 4-6) for each target participant containing a link to the system's web page together with the user's login ID and password information. Newly created issues and requests resulting from the previous steps of the batch process are posted to each participant's message board and are available upon login (see section 3.2.6). The system's anti-Spam checking ensures that at every batch run no more than one message is sent to each affected user.

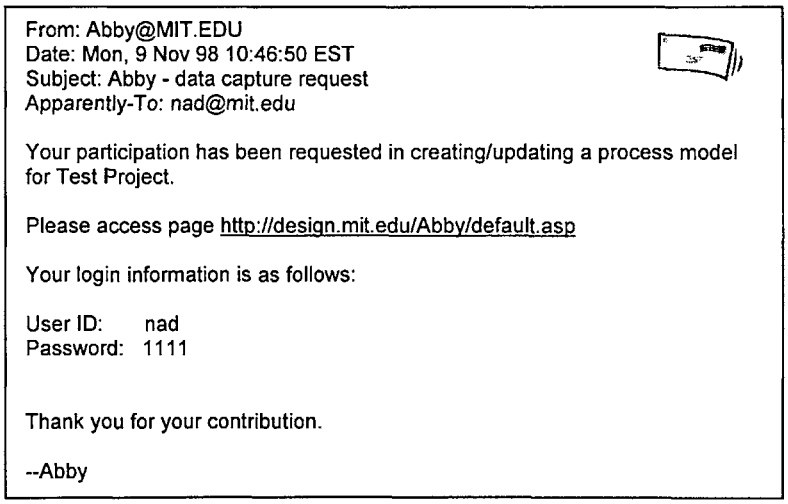

Figure 4-6 Sample e-mail Notification Message

The creation of customized E-mails for every type of issue or request was considered during the design phase. This approach although feasible, presented two major weaknesses. First of all, with each batch run, users whose involvement was necessary in multiple instances would have either received several different messages or, a very long e-mail detailing each issue. Secondly, the option would have been redundant with the system's user page (see section 3.2.6) presenting details, which would be once again displayed upon system login. The personalized message board could not be eliminated from the application in favor of customized e-mails since it meant providing access to system's on-line issue resolution solely through user's e-mail interface. It was considered necessary to provide users with the flexibility of accessing the system directly through

the browser (by typing the URL) or through the link provided in the message content of their notification e-mails.

Finally, a separate interface is developed for the stand-alone execution of the notification mechanism. The model coordination team has the flexibility of excluding this stage of the batch process in case frequent validation runs are deemed necessary. By disabling the system's automated notification module the team is also able to observe the results of the batch process before requesting user involvement.

4.3.3 Data Repository

The web-based prototype utilizes a relational database configuration for information storage and retrieval. Figure 4-7 outlines the details and relationships among the tables used in the system. Scalability is the most important factor considered during database design. For this purpose, four tables are designed to store information required for the construction of a large scale multi-tiered DSM. Task data is stored separately from information flow data. For each activity in the system one record is entered in the *DSMMaster* table (see Figure 4-7) containing all unique attributes while multiple records are entered in the *IO* table denoting the activity's defined information flow. Table 4-3 presents the list of fields and descriptions for the tables involved in DSM information storage.

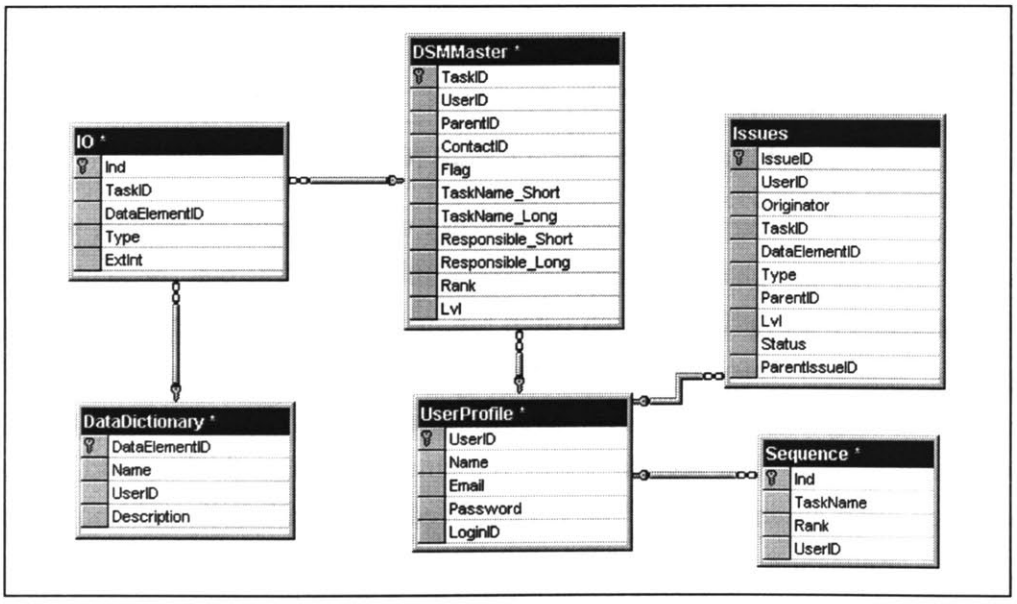

Figure 4-7 Data Repository Diagram

To better understand the dynamics of how a DSM is stored in the repository consider a task defined with 3 input and 2 output deliverables. This scenario will be represented by one entry in the *DSMMaster* table and five records in the *IO* table (three for the input deliverables and two for the output deliverables). By joining the two tables the list of inputs and outputs for each task is easily obtained. This design places no limits on the number of deliverables assigned to each task and is easily scaleable. The IO table is designed to be thin since it is expected to be far larger than any other table in the database (approximately one order of magnitude larger than *DSMMaster* table which represents the total number of tasks in the model). It is designed to contain strictly numerical field which are linked to appropriate reference tables during program execution.

Field	Description
DSM Master table	
TaskID	Unique system generated sequential key for each task
UserID	User ID for the task owner (linked field to UserProfile table)
ParentID	Task ID of the parent task (set to zero for level 1 tasks)
Flag	Indicates the existence of a decomposed DSM (if set to 1)
TaskName_Short	Short name of the task
TaskName_Long	Long name of the task
Responsible_Short	Short name of the team/individual responsible for task execution
Responsible_Long	Long name of the team/individual responsible for task execution
Rank	Task's ascending order of execution
Lvl	Task's decomposition level (Top level = 1)
IO table	
Ind	Unique system generated sequential key for each data flow
Task ID	Task ID (linked field to the DSM Master table)
DataElementID	ID for the I/O data element (linked field to DataDictionary table)
Type	Indicates whether the data element is being produced (1) or used (0)
ExtInt	Indicates whether record relates to external (1) or internal (0) interaction
DataDictionary table	
DataElementID	Unique system generated sequential key for each data element
Name	Given name for data element (deliverable)
UserID	User ID for the data element owner (linked field to UserProfile table)
Description	Description provided for each data element
UserProfile table	
UserID	Unique system generated sequential key for each user
Name	User's full name (first and last separated by a space)
Email	User's e-mail address
Password	Password used to access the system
LoginID	ID used to log into the system

Table 4-3 Field descriptions for DSM related tables

4.4 Software Architecture

The prototype system was developed by a group of three developers at the laboratories of MIT's Center for Innovation in Product Development (CIPD). The team utilized a combination of development tools and services. In the overall two environments were configured and explained in detail as follows:

4.4.1 Run-time environment

The prototype system is configured on a server computer connected to the World Wide Web (WWW). The production environment required for deployment is presented in Figure 4-8. The web server utilizes Microsoft Windows NT Server (version 4.0) as its operating system and Microsoft Internet Information Server (version 1.1) for Internet connectivity services such as WWW and Active Server Pages (ASP). Microsoft SQL Server (version 6.5) handles all database management functions on the server.

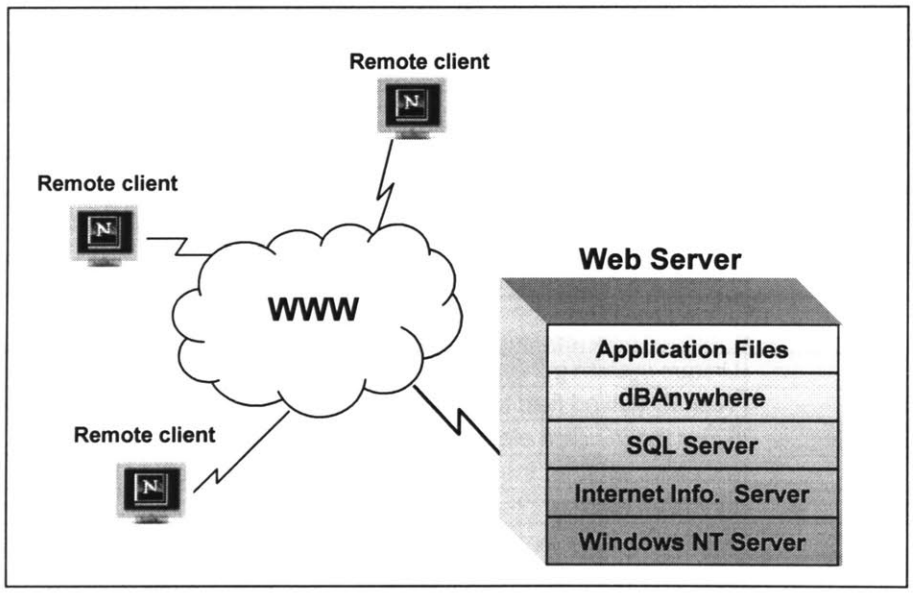

Figure 4-8 Web server configuration

Table 4-4 displays the types of application files residing on the system. Connectivity from Java files (applets and applications) to the SQL server database management system is accomplished through the Java DataBase Connectivity (JDBC) protocol using Symantec's middleware tool dBAnywhere. Database connectivity from ASP files was accomplished using standard SQL Server drivers provided by Visual Interdev.

Type	Description/Use
HTML files	display of static web pages
ASP files	creation of interactive web pages
Java applet class files	used for browser embedded execution
Java application class files	used for standlone execution using the Java Development Kit (JDK)
Java library class files	all vendor developed support libraries (Sun, Symantec etc.)

Table 4-4 File types utilized in the prototype system

The system can easily be replicated on any other web-server using the five steps outlined in Appendix B of this thesis. The instructions include the setup of the operating environment as well as required modifications to ASP and Java files. Modifications to the code relate to server specific (database or www) references such as domain name and IP addresses, which are unique to each operating environment.

4.4.2 Development Environment

A set of software development tools were used to construct the prototype web system. A multi-user integrated development environment was configured using Microsoft Visual Interdev (version 1.0). The tool managed a single web project containing database files, HTML, ASP, Java class files and other multi-media component used for the application. Using Visual Interdev each developer was able to work on files residing the development web server from any connected computer. Visual Interdev also provided the team with version control services enabling multi-developer work on the same collection of files.

Symantec Visual Cafe` (version 2.5) was used for the development of the system's Java components. The system's HTML pages were developed using Microsoft FrontPage (version 98) and the database management system was configured using Microsoft's SQL Enterprise Manager. All these tools were accessible through Visual Interdev's project workspace, which presented developers with an overall view of the system and a seamless interface for enhancements or new component design.

4.5 Summary

This chapter presented the various steps taken in this research to transform concepts and ideas presented in chapter 3 into a working prototype system. Areas of work leading to the development of the web-based software were reported in detail, including

requirement analysis, conversion of requirements to system specifications and the conception of target metrics for the system. The tool's high-level design concept was structured in the main areas of User Interface, Data Repository, and Batch Process. Each area was thoroughly discussed by focusing on the critical aspects of the prototype's design. Finally, the set of software development tools and sub-systems required for the design and execution of the system were presented. The overall goal of this chapter is to provide readers with relevant information on the design process used in this research as well as the web-based prototype's technical characteristics and architecture.

5 CONCLUSION

5.1 Summary

This thesis presented an approach for the deployment of Internet technologies to facilitate the collection and dissemination of knowledge on the product development process. The research highlighted inefficiencies in existing data collection techniques and the immense coordination challenges faced when attempting to apply these techniques to tap into the knowledge of a large group of geographically dispersed individuals. It has also emphasized the inadequacy of commonly adopted project planning tools in providing management with an adequate picture of the product development process for effective decision making.

A web-based process modeling approach was developed to overcome current data collection barriers. It attempts to engage a large number of geographically dispersed users in a virtual forum for the construction of a model representing their activities and interactions. The approach is presented as a transition from the conventional synchronous data gathering and centralized data management to an asynchronous-decentralized configuration through the use of web-based distributed data collection and on-line discussion methods.

Combinations of existing concepts from the Design Structure Matrix DSM modeling methodology were adopted. In particular, the "data-driven" approach to DSM modeling was drawn from Dr. Grose's many years of process modeling experience at the Boeing Commercial Airplane Group (BCAG). The benefits of this analytically rigorous approach to process modeling were discussed by focusing on the method's ability to pinpoint "black holes" (information generated that is never used) and "miracles" (required information that is never produced) [12].

A user defined task hierarchical scheme was proposed to introduce structure in the data collection process and resolve issues related to the visualization of very large matrices. The method entails gradual involvement of individuals in the organization's chain of command starting from top-management, in the definition project milestones, to designers outlining detailed activities performed in the product development process.

Enhanced web-based user interfaces were proposed to facilitate DSM adoption among the vast majority of novice users. Among these the introduction of familiar graph-based process representations in conjunction to the DSM matrix-based process view. The

concept of a personalized message board was also presented as a means to structure known user interactions for the purposes of model integration.

A web-based prototype system was successfully developed to further explore the concepts presented previously. The system development exercise was presented in detail starting from requirements and specifications analysis to high-level design and software architecture.

5.2 Directions for Future Work

The web-based prototype was demonstrated numerous times to CIPD industrial partner delegations, MIT students and faculty as well as members of other academic and research institutions. The modeling approach was also presented and discussed at CIPD, LAI and the IEEE SMC '98 seminars. Feedback received from these sessions and the lessons learned from the prototyping exercise have led to definition of the following areas of future work:

5.2.1 Pilot deployment

There is value in determining the effectiveness of the web-based approach in reducing the time and effort required to construct a task-based DSM model of the product development process. The approach presented in this thesis attempts to eliminate the bottlenecks in data collection which make large-scale DSM modeling unattainable. Major areas of overhead are addressed through a combination of automation techniques and distributed computing. The benefits are clear but not compelling unless a quantitative analysis is done through real life pilot implementations of the web-based system.

Scaled-down models comprised of approximately 200-400 activities requiring the participation of 20-30 professionals at a sponsoring company would be the recommended setting. Ideally, such experiments would involve two sample areas of the sponsoring organization's product development process. These two areas must be comparable in terms of size (no. of activities) and the number of experts required for data collection. One project must be modeled using the current centralized and asynchronous technique while the other using the Web-based prototype. Equal number of resources must also be allocated to each modeling initiative (typically 2 model coordinators would suffice). The modeling exercise is complete for each project once a target number of modeled activities is reached with a certain percentage of outstanding integration issues (e.g. 300 tasks, 5%

of which continue to be associated with data collection issues). The time to completion of each modeling exercise can therefore be measured and comparisons between the two approaches made.

Conducting several such experiments could provide enough empirical evidence on the potential efficiencies gained from the proposed web-based data collection mechanism. Extrapolating from data obtained in these experiments one may also be able to predict with increasing accuracy the time and resources required to construct large-scale DSMs using either technique.

5.2.2 Structural analysis

This research has presented a systematic approach to task decomposition, one that begins with users' preconceived notion of how all the thousands of project tasks are aggregated in a group of top-level activities. This approach is adopted at each level of decomposition with modelers subjectively present the next stage of task aggregation. This scheme however useful in introducing structure in data collection may not reflect the optimal clustering of tasks for visualization. The ideal representation of a task-hierarchical model should emerge from a sequencing followed by a clustering analysis to ensure that highly interactive group of ten or fifteen tasks are aggregated at each level in a bottom-up fashion.

Follow-up studies are needed to perform a comparative analysis between the subjective top-down task decomposition structure used for model creation and the optimal task aggregation scheme emerging from clustering and sequencing analysis on the lowest level set of DSMs in the model. This work may lead to interesting insights on the differences between the task decomposition scheme emerging from the modelers perspective and the one based on a systematic approach of minimizing interfaces among task clusters.

5.2.3 Behavioral analysis

Assumptions on users' willingness to participate in the scheme outlined in this research remain to be tested. It is important to explore whether the proposed approach provides the correct human computer interface for the target audience and type of information collection in question. Users' response to the virtual forum and automated aspects of issue resolution must be analyzed. In particular as this relates to the negotiation process for convergence on a common terminology for deliverables and other forms of information exchange among project participants. It is also interesting to observe users'

rate of response to the periodic e-mail solicitations generated by the system to validate the effectiveness of the "push" technique used in this research.

The type of insights gained from data-driven DSM-based modeling are likely to point towards radical changes in the design process and question existing organizational structures. This could potentially be the largest barrier towards a successful modeling exercise. Organizational change often leads to conflict and participants may stall the process once there is a realization that information they provide could lead to undesired side-effects. Further studies could explore potential conflicts of interest arising among modeling participants and the impact of these scenarios on the successful implementation of the DSM modeling exercise.

5.2.4 Software Enhancements

The following features were identified as potential areas of improvement to the system's functionality:

- Task dependencies are currently determined through a process of input and output deliverable matching. This mechanism is performed through a simple string comparison and could be further refined with pattern matching algorithms.
- The deliverable search mechanism could be enhanced to provide additional search features. This provides much needed assistance to participants wanting to locate a deliverable based on parameters such as the teams or individuals producing or using the deliverable, or a particular notation in the deliverable's description. Currently the list of deliverables is simply presented alphabetically. Alternatives must be explored to provide users with a categorization of such information and further facilitate the search process. These features can reduce the number of data disconnects stemming from terminology issues.
- Improved coordination features can be introduced by providing additional contact information and contact management functionality throughout the system. This may also include integration of modeling information with existing groupware where actual deliverables could be stored. Users can not only see tasks and deliverable exchanges but also access the content of the deliverable through especially provided links.
- Current Internet infrastructure hinders seamless worldwide deployment of the system. Firewalls present in practically every organization prevent the system's Java classes

from communicating with a remote database database. Suggestions to overcome this aggravation include: collecting data through ASP calls only, deploying the server behind a company firewall (problems with suppliers and other outside entities will persist), introducing features that would allow users to grant applets permission to exit firewall etc.

- Lack of standards in Web browsers leads to compatibility issues. Currently, the tool can only be deployed on Netscape Navigator version 4.06 or above (current release of Navigator is 4.5). There are several issues when using Microsoft Internet Explorer. These relate to the look and feel of web pages and the deployment of embedded Java applets. Standards are gradually solidifying in the browser market, however the effort can be placed to ensure that the application has the same look and feel across existing combinations of browser-operating system.

- Security must also be considered. Information being modeled with this system is quite sensitive in nature since it deals with the organization's biggest asset: it's knowledge. Secure transactions for database access beyond firewall and more sophisticated authentication mechanism for login and the use of encrypted e-mail are some alternatives that should be further explored.

- Providing users with maximum flexibility in the visualization of the multi-tiered DSM is another important area of work. Enabling high-level or detailed views of activities in very large models may entail a change of resolution of up to 1000 times (assuming a 4-level model). Navigation is currently structured according to a user specified task decomposition scheme emerged during model construction. This inter-level navigation method, as discussed in section 3.2.2, limits the ability of users to explore off-diagonal interactions to their desired level of detail. Alternative approaches could be introduced to provide enhanced visualization functionality aimed at more sophisticated DSM users. One possible approach is currently being explored by Shaun Abrahamson, another graduate student at MIT's Center for Innovation in Product Development (CIPD). As a member of the DOME project, Shaun has developed a *dynamic range* DSM visualization technique which provides users with a multi-view interface for navigation. A sample of this scheme can be seen in Figure 5-1. There are three views presented to the users. The first (top right corner) provides a map of the entire matrix using a 200x200 pixel display. Although the resolution can not accommodate a 10,000 node square matrix, it is sufficient to provide users with means of identifying potential areas of interest based on the

displayed patterns of interaction. The second view (left of the screen) is closest to the conventional views other than that it allows for the representation of any element with any other element in the model. This is a departure from DSM's N-square paradigm since rows and columns could represent different areas of the model. The third and final view (bottom right) is simply a list of all dependent activities for the selected element, providing the highest level of resolution.

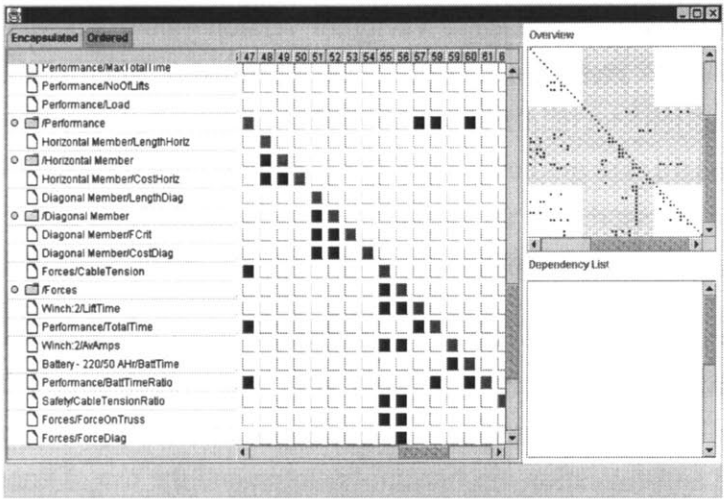

Figure 5-1 Example of an alternative DSM visualization technique

6 BIBLIOGRAPHY

[1] T. R. Browning. "An Introduction to the Use of Design Structure Matrices for Systems Engineering, Project Management, and Organization Planning", *Working Paper, #WP97-001-18, Feb. 98*, M.I.T. Lean Aircraft Initiative, Cambridge, MA

[2] T. R. Browning."Systematic IPT Integration in Lean Development Programs", *Master of Science in Technology Policy* and *Master of Science in Aeronautics and Austronautics,* June 1996, Chapter 4

[3] A. H. Bond and R. J. Ricci. "Cooperation in Aircraft Design", *Research in Engineering Design.* 1992 Vol. 4, pp. 115-130.

[4] R. A. Carbtree, M. S. Fox and N. K. Baid. "Case Studies of Coordination Activities and Problems in Collaborative Design". *Research in Engineering Design.* (1997) 9: pp. 70-84

[5] M. Case and S. C-Y Lu. "Discourse Model for Collaborative Design". *Computer Aided Design.* Vol. 28, No. 5 pp. 333-345, 1996

[6] G. A. Dirks. "Aircraft Configuration Optimization Considering Structural Flexibility". *American Institute of Aeronautics and Astronautics.* AIAA-96-4107-CP. pp. 1085-1087

[7] S. D. Eppinger et al. "A Model-Based Method for Organizing Tasks in Product Development", *Journal of Engineering Design.* Vol. 6, pp. 1-13, 1994.

[8] D. L. Grose, The Boeing Company. "Reengineering The Aircraft Design Process", *American Institute of Aeronautics and Astronautics*, 1994.

[9] C. C. Madni and A. M. Madni. "Web-enabled Collaborative Design Process Management: Application to Multichip Module Design". *IEEE International Conference on Systems Management & Cybernetics.* 1998 pp. 2625-2530

[10] T. W. Malone and K. Crowston, (1994). "The interdisciplinary study of coordination". *Computing Surveys*, *26*(1), 87-119.

[11] J. A. Mariani and T. Rodden. "Cooperative Information Sharing: Developing a Shared Object Service", *The Computer Journal.* Vol. 39, No. 6 1996 pp 455-470

[12] W. L. McCoy and A. M. Madni. "Process Support for IPPD-Enabled Systems Engineering. *IEEE International Conference on Systems Management & Cybernetics.* 1998 pp. 2585-2590

[13] G. R. Olsena, M. Cutkosky, J. M. Tenenbaum, T. R. Gruber. "Collaborative Engineering based on Knowledge Sharing Agreements". *ASME Database Symposium*, Sept 11-14, 1994

[14] B. Prasad, F. Wang and J. Deng. "Towards a Computer Supported Cooperative Environment for Concurrent Engineering". *Concurrent Engineering: Research and Applications.* Vol. 5, No. 3 , Sept. 1997

[15] T. U. Pimmler and S. D. Eppinger. "Integration Analysis of Product Decompositions", *ASME Conference on Design Theory and Methodology*, 1994 DE-Vol 68, pp 343-351

[16] A. O. Salas and J. L. Rogers. "A Web-based System for Monitoring and Controlling Multidisciplinary Design Projects". *NASA Technical Memorandum TM-97-206287.* December 1997

[17] R. P. Smith and S. Eppinger. "A Predictive Model of Sequential Iteration in Engineering Design", MIT Sloan School of Management Working Paper 1360, March. 1996.

[18] D. V. Steward. "The Design Structure System: A Method for Managing the Design of Complex Systems", *IEEE Transactions on Engineering Management.* August 1981, pp. 71-74.

[19] C. Stogdill, A. M. Madni and C. C. Madni. "ProcessWeb™: Web-enabled Process Support for Planning and Formation of a Virtual Enterprise". *IEEE International Conference on Systems Management & Cybernetics.* 1998 pp. 2591-2596

[20] H. Takeuchi and I. Nonaka. *The Knowledge Creating Company,* Oxford University Press, NewYork, 1995, Chapter 3 , pp. 56-95

[21] K. T. Ulrich and S. D. Eppinger. *Product Design and Development,* McGraw-Hill, NewYork, 1995, Chapter 12, pp. 260-282

[22] J. Halal. *The InfiniteResource,* John Wiley & Sons, Inc., NewYork, 1993

[23] E. Rechtin. *System Architechting,* Prentice-Hall Inc., NewJersy, 1991, Chapter 3, pp. 52-72

APPENDIX A - REQUIREMENTS DEFINITION

System Description

A web-based software tool for supporting planning operations on large-scale product development projects. The tool is used to collect information on scheduled tasks and inter-task interactions from a large audience of managers and team representatives . The information is used to create an accurate and timely map of the development process. It helps organizations identify iterations in the process and reduce unnecessary rework due to inadequate sequencing of activities in the schedule.

I Access

1 The system can be easily accessed by large number of users (in the order of thousands)
2 The system can be easily accessed by geographically dispersed users
3 The system is available on most common computing platforms (PC, MAC, UNIX etc.)

II Data Capture

1 The system captures information in the following categories:
-Task information
-Input / Output data elements
-Team or individual responsible
2 The system is capable of handling data capture for large development programs (on the order of hundred thousands tasks) as well as small ones
3 The system provides DSM data capture without the need for cross-functional team meetings
4 The system can actively query specific users for specific information at a specific time
5 The user requires little or no training to enter data
6 The system is capable of tracing all entered information to its source
7 The system assists users in providing information at the correct level of abstraction
8 The system provides a first level of validation upon data entry
9 The system has an acceptable response time during data capture
10 The system contains a data dictionary (definition of data elements including attributes)

III Interfaces

1 The system is able to obtain task information from most common scheduling tools (e.g. Microsoft Project)
2 The system is able to produce output for most common scheduling tools (e.g. Microsoft Project)

IV Presentation

1 The system provides full flexibility for viewing DSM models
2 The user is able to zoom in and out of a specified area
3 The system is capable of presenting very large matrices (on the order of hundred thousands tasks)
4 The system can automatically aggregate tasks for maximum viewing quality
5 The system takes advantage of the available screen resolution to maximize viewing quality
6 The user can specify a hierarchical task roll-up scheme for viewing
7 The system has a default hierarchical task roll-up scheme for presentation
8 The system can present task completion status through a color coding scheme
9 The system can provides a task only roll-up view and navigation

V Reporting

1 The system is capable of printing all available views of a DSM model
2 The system is capable of printing all information displayed on-line

3 The user can copy, paste or save a DSM model in a graphical format for integration into a presentation or report etc.
4 The system is capable of formatting DSM models for large printers/plotters

VI Validation
1 The user selects I/O data elements from a standard data dictionary
2 The user can add new data elements to the data dictionary
3 The system identifies data discrepancies (inconsistencies) in the model
4 The system provides a first level of validation upon data entry
5 The system ensures that only trained (certified) users be able to enter data

VII Security
1 The system is capable of securely collecting information through the Internet
2 Data collected in the system is adequately protected against break-in
3 Access to the system can be limited to a pre-defined group of users
4 Several user security profiles can be setup with different levels of access

VIII Search/Retrieve
1 The user can inquire on the following:
2 Data element usage
3 Data element backward chaining
4 Task location
5 List of tasks for a responsible group/individual
6 History of modifications to the model
7 The system has an acceptable response time during inquiries

IX Analysis
1 The user is able to manually sequence/re-sequence activities (scenario analysis)
2 The system automatically re-sequences activities to minimize iterations
3 The user is able to save each sequencing scheme
4 Future (more sophisticated) sequencing algorithms can be easily added to the system
5 The user can analyze the impact of potential upstream changes on downstream tasks in the model ("what if" capability)

X System Maintenance
1 The system administrator is able to easily access all reference data for modification
2 The system administrator is able to create, modify or delete user profiles
3 The system administrator is able to activate/de-activate users

XI Performance Support
1 The system is equipped with on-line help
2 The system is equipped with on-line tutorial

APPENDIX B - CONFIGURATION INSTRUCTIONS

Step 1) Cofiguration Web-server

The following software components must be installed on the web-server:

- Miscrosoft Windows NT 4.0 (including service pack 3)
- Microsoft Internet Information Server
- Visual Cafe` dBanywhere 1.1
- Microsoft SQL Server 6.5

Step 2) Move application files

All application files are compressed in the file "abby.zip" (approximately an 8.5 M-byte file). This file must be unzipped in the root web directory of the server (typically //InetPug\wwwroot\ directory). At this point the folder "abby" and all its sub-folders must appear and be accessible via a browser on the www.

Step 3) Install database files

The database file for this application has been backed-up and compressed in a file named "abbydatabase.zip" in the folder: "//abby\database backup". The following steps must be followed for the applications's database installation:

1.unzip the file abbydatabase.zip

2. copy the unzipped file "abbydatabase.dat" into the "backup" directory of SQL server (generally found in (//MSSQL\Backup

1. launch SQL Enterprise manager

2.create a new back-up device by right clicking on the "back-up devices" folder and selecting "create new"

3. specify the directory and filename for the "abbydatabase.dat" file found in the bacup directory and press OK

4. restore the databases in the newly created backup device by right-clicking on the device and selecting the "restore" option

At this point, two databases should be added to SQL server. These can be viewed in the "databases" folder as "abby" and "MainAbbyDB". The table "MainAbbyDB contains information used to connect to various databases each representing a different project being modeled. The field "SystemName" must be changed if the database name "abby" is modified or if another replica of the "abby" database is made.

Step 4) Perform code changes to accomodate new environment

The ASP file "default.asp" must be modified to reflect the new location of the database management system. The change affects a series of session variables defined in this file relating to data connectivity. First of all, all six references to the current database server "design" must be changed to the new database server name (this can be done through a simple string search and replace function). Secondly, the new user names and passwords must be entered in the three instances of session dataconnection variables.

There are also changes to the Java files to reflect the new server environment. The table below outlines the changes required to appropriate lines within each affected file. Once changes are made to the Java file these must be recompiled to class files. These newly compiled class files must replace the old ones in their corresponding folder.

File	Line	Code to be modified
abby\DataCapture\FrontEnd\AbbyDBConnection.java	11	String url="jdbc:dbaw://design:8889/SQL_Server/design/" + Project;
abby\DataCapture\FrontEnd\AbbyDBConnection.java	16	con=DriverManager.getConnection(url,"sa","gromit70");
abby\DataCapture\FrontEnd\AbbyPanel2.java	277	app.getAppletContext().showDocument(new URL("http://design.mit.edu/Abby/user/home.asp"));
abby\DataCapture\FrontEnd\AbbyPanel2.java	399	app.getAppletContext().showDocument(new URL("http://design.mit.edu/Abby/dsm/dsm.asp?Lvl="+lev+"&ParentID="+parentId));
abby\DataCapture\MailServer\AbbyDBChecker.java	11	String url="jdbc:dbaw://design:8889/SQL_Server/design/Abby";
abby\DataCapture\MailServer\AbbyDBChecker.java	41	String message=checkString(rs.getString(3))+",\n"+" Please go to the Abby Website: http://design.mit.edu\n"+
Abby\DataCapture\MailServer\AbbyMailServer.java	67	"Please access page http://design.mit.edu/Abby/default.asp \n\n\n"+
abby\Presentation\AbbyDBConnection.java	10	String url="jdbc:dbaw://design:8889/SQL_Server/design/" + Project;
abby\Presentation\AbbyDBConnection.java	16	con=DriverManager.getConnection(url,"sa","gromit70");
abby\Presentation\Task1PopUp.java	102	imgArrow1.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/arrow.gif"));
abby\Presentation\Task1PopUp.java	111	imgArrow2.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/arrow.gif"));
abby\Presentation\Task1PopUp.java	120	imgTask1.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/oval.gif"));
abby\Presentation\Task2PopUp.java	123	imgArrow1.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/arrow.gif"));
abby\Presentation\Task2PopUp.java	132	imgTask1.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/oval.gif"));
abby\Presentation\Task2PopUp.java	161	imgTask2.setImageURL(new java.net.URL("http://design.mit.edu/abby/Presentation/oval.gif"));

Step 5) Start your engines

At this point the application is ready for deployment. Accessing the file "default.asp" through a browser shoud display the system's login screen. User ID and password information are contained in the "UserProfiles" table of the "abby" database.

APPENDIX C - PROGRAM MODULES

D.1 Active Server Pages

The following modules are written in a combination of HTML and VB-Script and are used to create interactive web pages.

Module: /default.asp

Description: Home page of the application. Performs session initialization and accesses the main database to present available projects for user access. Includes standard login script. and links to introductory HTML pages. User permission is established and navigation is directed to user's home page if any issues are found in the issues database. The user home page is by-passed if there are no issues and the user's DSM is presented.

```
<% Response.Expires = 0 %>
<%
     Session.Timeout = 120
     '==Visual InterDev Generated - DataConnection startspan==
          Session("Personal_ConnectionString") = "DRIVER=SQL Server;SERVER=design;UID=sa;APP=Microsoft(R) Windows NT(TM) Operating
System;WSID=design;LANGUAGE=us_english"
          Session("Personal_ConnectionTimeout") = 15
          Session("Personal_CommandTimeout") = 30
          Session("Personal_RuntimeUserName") = "sa"
          Session("Personal_RuntimePassword") = "gromit70"
     '--Project Data Connection
          Session("MainAbbyDB_ConnectionString") = "DRIVER=SQL Server;SERVER=design;UID=sa;APP=Microsoft (R) Developer
Studio;WSID=INFORMATION;DATABASE=MainAbbyDB;LANGUAGE=us_english"
          Session("MainAbbyDB_ConnectionTimeout") = 15
          Session("MainAbbyDB_CommandTimeout") = 30
          Session("MainAbbyDB_RuntimeUserName") = "sa"
          Session("MainAbbyDB_RuntimePassword") = "gromit70"
     '--Project Data Connection
          Session("Data_Conn_ConnectionString") = "DRIVER=SQL Server;SERVER=design;UID=sa;APP=Microsoft(R) Windows NT(TM) Operating
System;WSID=design;LANGUAGE=us_english"
          Session("Data_Conn_ConnectionTimeout") = 15
          Session("Data_Conn_CommandTimeout") = 30
          Session("Data_Conn_RuntimeUserName") = "sa"
          Session("Data_Conn_RuntimePassword") = "gromit70"
     '==Visual InterDev Generated - DataConnection endspan==
     ' Set up initial DSM display variables
        Session("CellHeight") = 12
        Session("CellWidth") = 160
        Session("Padding") = 3
        Session("FontSize") = 10
        Session("Offset") = 2
        Session("MenuSpace") = 35
If Request("Action") = "LOGIN" Then
     Response.Cookies("LoginID") = Request.Form("LoginID")
     Response.Cookies("LoginID").Expires = Date + 14
     Response.Cookies("Project") = Request.Form("Project")
     Response.Cookies("Project").Expires = Date + 14
     SQLQuery = "SELECT * FROM MasterTable WHERE SystemName = '" + Request("Project") + "'"
     Set MainAbbyDB = Server.CreateObject("ADODB.Connection")
     MainAbbyDB.Open Session("MainAbbyDB_ConnectionString"), Session("MainAbbyDB_RuntimeUserName"), Session("MainAbbyDB_RuntimePassword")
     Set RSSysAdminCheck = MainAbbyDB.Execute(SQLQuery)
     Session("Data_Conn_ConnectionString") = "DRIVER={SQL Server};SERVER=design;UID=abby;APP=Microsoft (R) Developer
Studio;WSID=INFORMATION;DATABASE=" + Request("Project")
     Session("ProjectName") = Trim(RSSysAdminCheck("ProjectName"))
     Session("Project") = Request("Project")
     Session("isSysAdmin") = "false"
     Set Data_Conn = Server.CreateObject("ADODB.Connection")
     Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
     SQLQuery = "SELECT * FROM UserProfile WHERE LoginID='" + Request("LoginID") + "' AND Password='" + Request("Password") + "'"
     Set RSLogin = Data_Conn.Execute(SQLQuery)
     If (not RSLogin.EOF) Then
          Session("UserID") = RSLogin("UserID")
          Session("LoginID") = RSLogin("LoginID")
          Session("Name") = Trim(RSLogin("Name"))
          Session("Password") = RSLogin("Password")
          If (Trim(RSSysAdminCheck("LoginID")) = Request("LoginID")) Then
               Session("isSysAdmin") = "true"
          End If
          RSLogin.Close
          RsSysAdminCheck.Close
          MainAbbyDB.Close
          'SET UP INITIAL DSM AND CHECK IF USER HAS ISSUES.
          SQLQuery = "SELECT * FROM DSMMaster WHERE UserID=" + FormatNumber(Session("UserID"),0) + " ORDER BY Lvl, TaskID"
          Set RSDSM = Data_Conn.Execute(SQLQuery)
          If (not RSDSM.EOF) Then
               Session("InitialLvl") = FormatNumber(RSDSM("Lvl"),0)
               Session("InitialParentID") = FormatNumber(RSDSM("ParentID"),0)
          Else
               Session("InitialLvl") = 1
               Session("InitialParentID") = 0
          End If
```

```
          RSDSM.Close
          SQLQuery = "SELECT * FROM Issues WHERE UserID=" + FormatNumber(Session("UserID"),0)
          Set RSIssues = Data_Conn.Execute(SQLQuery)
          If (not RSIssues.EOF) Then
               RSIssues.Close
               Data_Conn.Close
               Response.Redirect "user/home.asp"
          Else
               RSIssues.Close
               Data_Conn.Close
               Response.Redirect "dsm/dsm.asp?Lvl=" + FormatNumber(Session("InitialLvl"),0) + "&ParentID=" +
FormatNumber(Session("InitialParentID"),0)
          End If
     End If
     RSLogin.Close
     RSSysAdminCheck.Close
     Data_Conn.Close
End If
%>
<%Response.Expires = 0%>
<%
Warn=""
If (Request("Action") = "LOGIN") Then
     Warn="Invalid Login! <br><br><br>  Please try again..."
end if%>
<html>
<head>
<link REL="stylesheet" HREF="StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>
<body background="dgrey042.jpg">
<p><img src="WB01355_1.gif" width="43" height="43">   <img src="a_clr.gif"
width="43" height="42" alt="a_clr.gif (3367 bytes)"><img src="b_clr.gif" width="43"
height="42" alt="b_clr.gif (3670 bytes)"><img src="b_clr.gif" width="43" height="42"
alt="b_clr.gif (3670 bytes)"><img src="y_clr.gif" width="43" height="41"
alt="y_clr.gif (3259 bytes)"></p>
<div align="center"><center>
<table border="3" width="70%">
  <tr>
    <td width="100%"><p align="center"><br>
    <applet code="fphover.class" codebase="_fpclass/" width="130" height="24">
      <param name="text" value="About Abby">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="aboutabby.htm" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="_fpclass/" width="130" height="24">
      <param name="text" value="The Team">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="team.htm" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="_fpclass/" width="130" height="24">
      <param name="text" value="The Project">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="project.htm" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
    </p>
    <div align="center"><center><table border="0" width="534" height="179" cellpadding="15">
      <tr>
        <td width="293" height="1" align="center"><form method="POST" action="default.asp">
          <table border="0" width="102%" cellpadding="30" cellspacing="1">
            <tr>
              <td width="94%"><div align="left"><h5><font color="#FFFFFF"><strong>Enter Your User ID</strong><br>
              </font><input type="text" name="LoginID" size="12" value="<%=Request.Cookies("LoginID")%>">
              </h5>
              </div><div align="left"><h5>Enter Your Password<br>
              <input type="password" name="Password" size="12"> </h5>
              </div><div align="left"><h5>Select Your Project<br>
              <select name="Project" size="1">
<%   'Get Info for Combo Box
Set MainAbbyDB = Server.CreateObject("ADODB.Connection")
MainAbbyDB.Open Session("MainAbbyDB_ConnectionString"), Session("MainAbbyDB_RuntimeUserName"), Session("MainAbbyDB_RuntimePassword")
SQLQuery = "SELECT * FROM MasterTable"
Set RS = MainAbbyDB.Execute(SQLQuery)
While (not RS.EOF)
     If ((Request("Project") = Trim(RS("SystemName"))) OR (Request.Cookies("Project") = Trim(RS("SystemName")))) Then
          Response.Write ("<option selected value='" + Trim(RS("SystemName")) + "'>" + Trim(RS("ProjectName")) + "</option>")
     Else
          Response.Write ("<option value='" + Trim(RS("SystemName")) + "'>" + Trim(RS("ProjectName")) + "</option>")
     End If
     RS.MoveNext
Wend
RS.Close
MainAbbyDB.Close
%>              </select></h5>
            </div></td>
```

```
            <td width="100%"><div align="center"><center><h5><font face="Arial" color="#FFFFFF"><big><%=Warn%></big></font></h5>
            </center></div></td>
          </tr>
        </table>
        <div align="center"><center><p><input type="submit" value="LOGIN" name="Action"
        width="100"></p>
        </center></div>
      </form>
      </td>
    </tr>
  </table>
  </center></div></td>
 </tr>
</table>
</center></div><font face="Arial" color="#000000">
<p align="center"></font><font color="#FFFFFF"><strong><small>This tool requires Netscape
Navigator 4.06  - Click here    </small></strong><a
href="http://www.netscape.com/download/index.html"><img src="netscape_now_anim.gif"
alt="netscape_now_anim.gif (1194 bytes)" WIDTH="90" HEIGHT="30">   </a> <strong><small>to
get the latest version</small></strong><br>
</font><font face="Arial" color="#000000"></p>
</font> </body> </html>
```

Module: /user/home.asp

Description: User home page where issues and requests are posted. All outstanding issue records for the user are analyzed, formatted and presented in a tabular format. Various fields in the table are hyperlinked to provide users with additional information on each issue. Users can navigate to their DSM, the model's task hierarchy view. The system administrator is given access to the user accounts page.

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Response.Expires = 0 %>
<script  LANGUAGE="VBScript" RUNAT="Server">
<html>
<head>
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title><%=Session("Name")%>'s Home</title>
<meta name="Microsoft Border" content="none">
<meta name="Microsoft Theme" content="none">
</head>
<body background="../dgrey042.jpg">
<script Language="JavaScript">
function openAssign(s) {
     window.open(s, 'Assign', 'width=400,height=400')
}
function openRequest(s) {
     window.open(s, 'Request', 'width=400,height=240')
}
function openDeny(s) {
     window.open(s, 'Deny', 'width=400,height=400')
}
</script>
<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="42" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>
<table border="3" width="90%">
  <tr>
    <td width="100%"><p align="center"><br>
    <font face="Arial" color="#000000"><%If Session("isSysAdmin")="true" Then
          Response.Write "<applet code="+"fphover.class"+" codebase="+"../_fpclass/"+" width="+"130"+ " height="+"24"+">"
          Response.Write "<param name="+"text"+" value="+"Accounts "+">"
          Response.Write "<param name="+"hovercolor"+" value="+"#9BECEA"+">"
          Response.Write "<param name="+"textcolor"+" value="+"#FFFFFF"+">"
          Response.Write "<param name="+"effect"+" value="+"glow"+">"
          Response.Write "<param name="+"url"+" value="+"../sysadmin/usersList.asp"+" valuetype="+"ref"+">"
          Response.Write "<param name="+"font"+" value="+"Dialog"+">"
          Response.Write "<param name="+"fontstyle"+" value="+"bold"+">"
          Response.Write "<param name="+"fontsize"+" value="+"14"+">"
          Response.Write "<param name="+"bgcolor"+" value="+"#9BECEA"+">"
          Response.Write "<param name="+"color"+" value="+"#366D6D"+">"
          Response.Write "</applet>   "
     End If%>
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="View My DSM">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url"
      value="../dsm/dsm.asp?Lvl=<%=Session("InitialLvl")%>&amp;ParentID=<%=Session("InitialParentID")%>"
      valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="View Hierarchy">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="heirarchy.asp" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
```

```
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="Logout">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="../default.asp" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
    </font></p>
    <p><br>
    </p>
    <h1><font face="Arial" color="#FFFFFF"><strong><%=Session("Name")%>'s Home<br>
    <br>
    </strong></font></h1>
    <p><%
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
SQLQuery = "SELECT * FROM ISSUES WHERE UserID=" + FormatNumber(Session("UserID"),0) + " ORDER BY Type DESC, TaskID DESC"
Set RSIssues = Data_Conn.Execute(SQLQuery)
MoveOn=0
If (RSIssues.EOF) Then
     Response.Write("<h5><br><br><center>You currently have no issues.</center><br><br><br><br></h5>")
Else
     While (not RSIssues.EOF) AND (MoveOn = 0)
          If (RSIssues("Type") = "9") Then
               Response.Write("<H5>Please create a model for the following tasks:</H5>")
               Response.Write(chr(10) + chr(13))
               Response.Write("<TABLE BORDER=1 CELLPADDING=10>")
               Response.Write("<TR><TH><H5>Task</TH><TH><H5>Requested By</TH><TH><H5>Action</TH></TR>")
               Do While (RSIssues("Type") = "9")
                    Response.Write(chr(13))
                    SQLQuery = "SELECT * FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
                    Set RSTaskInfo = Data_Conn.Execute(SQLQuery)
                    Response.Write("</h5><TR><TD>")
                    Response.Write("<A HREF=../dsm/dsm.asp?Lvl=")
                    Response.Write(FormatNumber(RSTaskInfo("Lvl"),0))
                    Response.Write("&ParentID=" + FormatNumber(RSTaskInfo("ParentID"),0)+"><u>")
                    Response.Write(Trim(RSTaskInfo("TaskName_Long")) + "</A></TD>")
                    SQLQuery = "SELECT * FROM UserProfile WHERE UserID=" + FormatNumber(RSIssues("Originator"), 0)
                    Set RSOriginator= Data_Conn.Execute(SQLQuery)
                    Response.Write("<TD><a HREF='mailto:")
                    Response.Write(Trim(RSOriginator("Email")) + "?subject=Model Creation'><u>" + Trim(RSOriginator("Name")) + "</a></TD>")
                    Response.Write("<TD>")
                    Response.Write("<a HREF=datacapture.asp?Lvl=")
                    Response.Write(FormatNumber(RSTaskInfo("Lvl"),0) + 1)
                    Response.Write("&ParentID=" + FormatNumber(RSIssues("TaskID"),0) + "><u>")
                    Response.Write("Create Model</A></TD>")
                    Response.Write("</TR>")
                    RSIssues.MoveNext
                    If RSIssues.EOF Then
                         Exit Do
                    End If
                    RSOriginator.Close
                    RSTaskInfo.Close
               Loop
               Response.Write("</TABLE>")
               Response.Write("<BR><br><br>")
          Else
               MoveOn = 1
          End If
     Wend
End If
LastTask = ""
If (not RSIssues.EOF) Then
     Response.Write("<H5>Please resolve the following issues:</H5>")
     Response.Write(chr(10) + chr(13))
     Response.Write("<TABLE BORDER=1 CELLPADDING=10>")
     Response.Write("<TR><TH><h5>Deliverable</TH><TH><h5>Task</TH><TH><h5>Action Created By</TH><TH><h5>Action</TH></TR>")
     While (not RSIssues.EOF)
          SELECT CASE RSIssues("Type")

               CASE "1", "2", "3"
          Do While ((FormatNumber(RSIssues("Type"),0) = "1") OR (FormatNumber(RSIssues("Type"),0) = 2) or (FormatNumber(RSIssues("Type"),0) = "3"))
                    If ((RSIssues("Type") = "3") and (FormatNumber(RSIssues("TaskID"),0) = LastTask)) Then
                    Else
                         Response.Write(chr(10) + chr(13))
                         SQLQuery = "SELECT * FROM DataDictionary WHERE DataElementID=" + FormatNumber(RSIssues("DataElementID"),0)
                         Set RSDE = Data_Conn.Execute(SQLQuery)
                         SQLQuery = "SELECT * FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
                         Set RSDSM = Data_Conn.Execute(SQLQuery)
                         SQLQuery = "SELECT * FROM UserProfile WHERE UserID=" + FormatNumber(RSIssues("Originator"),0)
                         Set RSOriginator = Data_Conn.Execute(SQLQuery)
                         Response.Write("<TR><TD><H2>")
                         If (RSIssues("Type") = "3") then
                              Response.Write("<center>N/A</center>")
                         Else
                              Response.Write(Trim(RSDE("Name")))
                         End If
                         Response.Write("</TD>")
                         Response.Write("<TD><u><A HREF=../dsm/dsm.asp?Lvl=" + FormatNumber(RSDSM("Lvl"),0) + "&ParentID=" +
FormatNumber(RSDSM("ParentID"),0) + "><u>" + Trim(RSDSM("TaskName_Long")) + "</u></A></TD>")
                         Response.Write("<TD><a href="+"mailto:nad@mit.edu"+"><u>Model Coordinator</a></TD>")
                         Response.Write("<TD>")
                         If (RSIssues("Type")= "3") Then
                              ChildLevel=RSDSM("Lvl")+1
                              Response.Write("<A HREF=../dsm/dsm.asp?Lvl=" + FormatNumber(ChildLevel,0) + "&ParentID=" +
FormatNumber(RSIssues("TaskID"),0) + ">")
                              Response.Write("<u>View Details</u></A>")
                              LastTask = FormatNumber(RSIssues("TaskID"),0)
                         Else
```

```
                              SELECT CASE RSIssues("Status")
                              Case 0,1
                                   Response.Write("<A HREF=assign.asp?IssueID=" + FormatNumber(RSIssues("IssueID"),0) + ">")
                                   Response.Write("<u>View Details</u></A>")
                              Case 2
                                   Response.Write("Waiting for response from <A HREF=mailto:" + Trim(RSOriginator("Email")) + "><u>" +
Trim(RSOriginator("Name")) + "</u></A>")
                              Case 3
                                   Response.Write("<A HREF=assign.asp?IssueID=" + FormatNumber(RSIssues("IssueID"),0) + ">")
                                   Response.Write("<u>View Details</u></A>")
                                   Response.Write(" (Request Denied by <A HREF='mailto:" + Trim(RSOriginator("Email")) + "?subject=Request Denial'>"
+ Trim(RSOriginator("Name")) + "</A>)")
                              Case 4
                                   Response.Write("<h5>Adressed by you")
                              END SELECT
                         End If
                         Response.Write("</TD></TR>")
                         RSDE.Close
                         RSDSM.Close
                         RSOriginator.Close
                    End If
                         RSIssues.MoveNext
                         If RSIssues.EOF Then
                              Exit Do
                         End If
               Loop
          CASE "5", "6"
               Do While ((FormatNumber(RSIssues("Type"),0) = "5") OR (FormatNumber(RSIssues("Type"),0) = "6"))
                    SQLQuery = "SELECT * FROM DataDictionary WHERE DataElementID=" + FormatNumber(RSIssues("DataElementID"),0)
                    Set RSDE = Data_Conn.Execute(SQLQuery)
                    SQLQuery = "SELECT * FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
                    Set RSTask = Data_Conn.Execute(SQLQuery)
                    Response.Write("<TR><TD><H2>")
                    Response.Write(Trim(RSDE("Name")) + "</TD>")
                    Response.Write("<TD><H2><A HREF=../dsm/dsm.asp?Lvl=" + FormatNumber(RSTask("Lvl"),0) + "&ParentID=" +
FormatNumber(RSTask("ParentID"),0) + "><u>" + Trim(RSTask("TaskName_Long")) + "</u></A></TD>")
                    Response.Write("<TD><H2>")
                    SQLQuery = "SELECT * FROM UserProfile WHERE UserID=" + FormatNumber(RSIssues("Originator"),0)
                    Set RSOriginator = Data_Conn.Execute(SQLQuery)
                    Response.Write("<A HREF=mailto:" + Trim(RSOriginator("Email")) + "><u>" + Trim(RSOriginator("Name")) + "</u></a></TD>")
                    RSOriginator.Close
                    Response.Write("<TD><H2>")
                    SELECT CASE RSIssues("Status")
                    Case 0,1
                         Response.Write("<A HREF=request.asp?IssueID=" + FormatNumber(RSIssues("IssueID"),0) + ">")
                         Response.Write("<u>View Details</u></A>")
                    Case 2
                         Response.Write("<h5>Waiting for response from <A HREF=mailto:" + Trim(RSOriginator("Email")) + "><u>" +
Trim(RSOriginator("Name")) + "</u></A>")
                    Case 3
                         Response.Write("<A HREF=request.asp?IssueID=" + FormatNumber(RSIssues("IssueID"),0) + ">")
                         Response.Write("<u>View Details</u></A>")
                         Response.Write("<h2>(Request Denied by <A HREF='mailto:" + Trim(RSOriginator("Email")) + "?subject=Request Denial'><u>"
+ Trim(RSOriginator("Name")) + "</u></A>)")
                    Case 4
                         Response.Write("<h2>Adressed by you")
                    END SELECT
                    Response.Write("</TD>")
                    Response.Write("</TR>")
                    RSIssues.MoveNext
                    If RSIssues.EOF Then
                         Exit Do
                    End If
                    RSDE.Close
                    RSTask.Close
               Loop
     End SELECT
Wend
Response.Write("</TABLE><br><br>")
End If
RSIssues.Close
Data_Conn.Close
%> </td>
  </tr>
</table>
</center></div>
<p align="center">  </p> </body></html>
```

Module: /user/hierarchy.asp

Description: Presents model's task hierarchy through the embedded applet"presentation.Applet1.class". Users can navigate to the DSM view or home page or logout the system.

```
<%
numTasks = 0
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")

SQLQuery = "SELECT * FROM DSMMaster"
Set RSCount = Data_Conn.Execute(SQLQuery)
While (not RSCount.EOF)
     RSCount.MoveNext
     numTasks = numTasks + 1
Wend
RSCount.Close
Data_Conn.Close
%>
<html>

<head>
```

```
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>

<body background="../dgrey042.jpg">

<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="41" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>

<table border="3" width="70%">
  <tr>
    <td width="100%"><p align="center"><br>
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="View My DSM">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url"
      value="../dsm/dsm.asp?Lvl=<%=Session("InitialLvl")%>&amp;ParentID=<%=Session("InitialParentID")%>"
      valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="My Home">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="home.asp" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="Logout">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="../default.asp" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
    </p>
    <h1><%=Session("ProjectName")%> Task Hierarchy</h1>
    <p align="center">
    <applet code="Presentation.Applet1.class" codebase="../" width="350" height="470">
      <param name="Project" value="<%=Session("Project")%>">
      <param name="numTasks" value="<%=numTasks%>">
      <param name="ProjectName" value="<%=Session("ProjectName")%>">
      <param name="UserID" value="<%=Session("UserID")%>">
    </applet>    <br>
    <br>    </td>  </tr></table></center></div></body></html>
```

Module: /user/assign.asp

Description: Page created to describe a data disconnect scenario for both input and output dangling dependencies. Three option buttons are provided with a comment box. Validation is performed to ensure comments are provided for options 2 and 3. A combo box containing all users in the database is provided for issue delegation.

```
<%
Msg = "Please select a course of action:"
Warn =""
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")

SQLQuery = "SELECT * FROM Issues WHERE IssueID=" + Request.QueryString("IssueID")
Set RSIssues = Data_Conn.Execute(SQLQuery)
If (Request("ActionChoice") = "") Then
     Warn = ""
Else
     If (Request("ActionChoice") = "C2") Then
          If (Request("UserChoice") = "-42") Then
               Warn = "Please select a valid user."
          End If
          If ( (Request("Comments") = "")  and (Warn="")) Then
               Warn = "Please enter a comment."
          End If
     ElseIf (Request("ActionChoice") = "C3") Then
          If (Request("Comments") = "") Then
               Warn = chr(13) + "Please enter a comment."
          End If
     End If
End If
If Request("Action")=" OK " Then
     If Warn = "" Then
          SELECT CASE Request("ActionChoice")
```

```
                CASE "C1"
                     SQLQuery = "UPDATE Issues SET Status=4 WHERE IssueID=" + Request("IssueID")
                     Set RSUpdate = Data_Conn.Execute(SQLQuery)
                     SQLQuery = "SELECT ParentID, Lvl FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
                     RSDSM = Data_Conn.Execute(SQLQuery)
                     Target = "../dsm/dsm.asp?ParentID=" + FormatNumber(RSDSM("ParentID"),0) + "&Lvl=" + FormatNumber(RSDSM("Lvl"),0)
                     Response.Redirect Target
                CASE "C2"
                     If (RSIssues("Type") = "1") Then
                          NewType = "5"
                     Else
                          NewType = "6"
                     End If
                     SQLQuery = "UPDATE Issues SET Status=2, Originator =" + FormatNumber(Request("UserChoice"),0)
                     SQLQuery = SQLQuery + " WHERE UserID =" + FormatNumber(Session("UserID"),0)
                     SQLQuery = SQLQuery + " AND TaskID =" + FormatNumber(RSIssues("TaskID"),0)
                     SQLQuery = SQLQuery + " AND DataElementID =" + FormatNumber(RsIssues("DataElementID"),0)
                     Set RSUpdate = Data_Conn.Execute(SQLQuery)

                     SQLQuery = "INSERT INTO Issues VALUES(" + FormatNumber(Request("UserChoice"),0) + ", "
                     SQLQuery = SQLQuery + FormatNumber(Session("UserID"),0) + ", "
                     SQLQuery = SQLQuery + FormatNumber(RSIssues("TaskID"),0) + ", "
                     SQLQuery = SQLQuery + FormatNumber(RSIssues("DataElementID"),0) + ", " + NewType + ", 0, 0, 0, "
                     SQLQuery = SQLQuery + FormatNumber(RSIssues("IssueID"),0) + ")"
                     Set RSInsert = Data_Conn.Execute(SQLQuery)

                     SQLQuery = "SELECT IssueID FROM Issues WHERE ParentIssueID=" + FormatNumber(Request.QueryString("IssueID"),0)
                     Set RSNewIssue = Data_Conn.Execute(SQLQuery)

                     Set cmdTemp = Server.CreateObject("ADODB.Command")
                     Set cmdTemp.ActiveConnection = Data_Conn
                     Set rsTemp = Server.CreateObject("ADODB.Recordset")
                     cmdTemp.CommandText = "SELECT * FROM Comments"
                     cmdTemp.CommandType =1
                     rsTemp.Open cmdTemp, ,1,3
                     rsTemp.AddNew
                     rsTemp("IssueID") = FormatNumber(RSNewIssue("IssueID"),0)
                     rsTemp("ParentIssueID") = FormatNumber(Request.QueryString("IssueID"),0)
                     rsTemp("FromID")= FormatNumber(Session("UserID"),0)
                     rsTemp("ToID")= FormatNumber(Request("UserChoice"),0)
                     rsTemp("Comments")= Request("Comments")
                     rsTemp.Update
                     rsTemp.Close
                     Response.Redirect "home.asp"
                CASE "C3"
                     SQLQuery = "UPDATE Issues SET Status=4 WHERE IssueID=" + Request("IssueID")
                     Set RSUpdate = Data_Conn.Execute(SQLQuery)
                     SQLQuery = "INSERT INTO Comments VALUES(" + FormatNumber(Request.QueryString("IssueID"),0) + ", "
                     SQLQuery = SQLQuery + FormatNumber(Request.QueryString("IssueID"),0) + ", "
                     SQLQuery = SQLQuery + FormatNumber(Session("UserID"),0) + ", "
                     SQLQuery = SQLQuery + "-1, '"
                     SQLQuery = SQLQuery + Request("Comments") + "')"
                     Set RSInsertComment = Data_Conn.Execute(SQLQuery)
                     Response.Redirect "home.asp"
          END SELECT
     End If
ElseIf Request("Action")="Cancel" Then
     RSIssues.Close
     Response.Redirect "home.asp"
End If
SQLQuery = "SELECT Name FROM DataDictionary WHERE DataElementID=" + FormatNumber(RSIssues("DataElementID"),0)
Set RSDE = Data_Conn.Execute(SQLQuery)
SQLQuery = "SELECT TaskName_Long, TaskName_Short FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
Set RSTask = Data_Conn.Execute(SQLQuery)
If (RSIssues("Type") = "1") Then
     IO = "input"
     PUpres = "produces"
     PUing = "producing"
     PUpast = "produce"
Else
     IO = "ouput"
     PUpres = "uses"
     PUing = "using"
     PUpast = "use"
End If
%>
<html>
<head>
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>
<body background="../dgrey042.jpg">
<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="41" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>

<table border="3" width="80%">
  <tr>
    <td width="100%"><br>
    <table border="0" ALIGN="CENTER">
      <tr>
        <td align="CENTER"><img src="../Presentation/Oval.gif" alt="Oval.gif (533 bytes)"
        WIDTH="196" HEIGHT="67"></td>
        <td align="CENTER"><img src="../images/arrow.GIF" alt="<%=Trim(RSDE("Name"))%>"
        WIDTH="138" HEIGHT="42"></td>
        <td align="CENTER"><img src="../Presentation/Oval.gif" alt="Oval.gif (533 bytes)"
        WIDTH="196" HEIGHT="67"></td>
      </tr>
      <tr>
        <td ALIGN="CENTER"><h2><%If (RSIssues("Type") = "1") Then
          Response.Write("?????")
     Else
          Response.Write(Trim(RSTask("TaskName_Short")))
     End If
     %> </h2>
```

```
        </td>
        <td align="CENTER"><i><%=Trim(RSDE("Name"))%></i> </td>
        <td align="CENTER"><%If (RSIssues("Type") = "2") Then
          Response.Write("<h2>?????")
      Else
          Response.Write(Trim(RSTask("TaskName_Short")))
      End If
      %>
</td>
     </tr>
    </table>
    <h5>There are no tasks in the model that <%=PUpast%> deliverable:      <i><%=Trim(RSDE("Name"))%></i>
    <br>
    </h5>
    <form method="POST" onsubmit="return FrontPage_Form1_Validator(this)"
    name="FrontPage_Form1">
      <table border="0" width="100%">
        <tr>
          <td width="60%"><h5><u><%=Msg%></u></h5>
          <h2><input type="radio" value="C1" name="ActionChoice"> I will fix this </h2>
          <h2><input type="radio" name="ActionChoice"
          <%If (Request("ActionChoice") = "C2") Then
      Response.Write("checked ")
      End If%>value="C2">
          The person <%Response.Write(PUing + " ")%> this deliverable is <select name="UserChoice" size="1">
            <option value="-42">Please Choose...</option>
<%
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")

SQLQuery = "SELECT * FROM UserProfile"
Set RSUsers = Data_Conn.Execute(SQLQuery)
While (not RSUsers.EOF)
      If (FormatNumber(RSUsers("UserID"),0) = FormatNumber(Request("UserChoice"),0)) Then
          Response.Write ("<option selected value='" + FormatNumber(RSUsers("UserID"),0) + "'>" + Trim(RSUsers("Name")) + "</option>")
      Else
          Response.Write ("<option value='" + FormatNumber(RSUsers("UserID"),0) + "'>" + Trim(RSUsers("Name")) + "</option>")
      End If
      RSUsers.MoveNext
Wend
RSUsers.Close
%>          </select></h2>
          <p><br>
          <input type="radio" name="ActionChoice"
          <%If (Request("ActionChoice") = "C3") Then
      Response.Write("checked ")
      End If%>
          value="C3"> I do not know who <%Response.Write (PUpres + " ")%> this deliverable </p>
          <h5><u>Comments:</u><br>
          <!--webbot bot="Validation" startspan S-Display-Name="Comments" I-Maximum-Length="250" --><!--webbot
          bot="Validation" endspan --><textarea rows="7" name="Comments" cols="40"><%=Request("Comments")%></textarea></h5>
          <h5><font face="Arial" color="#000000"><input type="submit" value=" OK " name="Action"> <input
          type="submit" value="Cancel" name="Action"></font></h5>
          </td>
          <td width="40%"><div align="center"><center><h5><%=Warn%></h5>
          </center></div></td>
        </tr>   </table>    </form>    </td>  </tr></table></center></div></body></html>
```

Module: /user/request.asp

Description: Page created to describe a data disconnect scenario that has been assigned by another user for both input and output dangling dependencies. Three option buttons are provided with a comment box. Previous comments are presented separately with hyper-linked origin and destination information Validation is performed to ensure comments are provided for options 2 and 3. A combo box containing all users in the database is provided for issue delegation.

```
<%
Msg = "Please select a course of action:"
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")

SQLQuery = "SELECT * FROM Issues WHERE IssueID=" + Request.QueryString("IssueID")
Set RSIssues = Data_Conn.Execute(SQLQuery)

If (Request("ActionChoice") = "") Then
     Warn = ""
Else
     If (Request("ActionChoice") = "C2") Then
          If (Request("UserChoice") = "-42") Then
               Warn = "Please select a valid user."
          End If
          If ( (Request("Comments") = "")  and (Warn="")) Then
               Warn = "Please enter a comment."
          End If
     ElseIf (Request("ActionChoice") = "C3") Then
          If (Request("Comments") = "") Then
               Warn = chr(13) + "Please enter a comment."
          End If
     End If
End If
If Request("Action")="  OK  " Then
     If Warn = "" then
          SELECT CASE Request("ActionChoice")
               CASE "C1"
                    SQLQuery = "UPDATE Issues SET Status=4 WHERE IssueID=" + Request("IssueID")
                    Set RSUpdate = Data_Conn.Execute(SQLQuery)
                    SQLQuery = "SELECT ParentID, Lvl FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
```

```
                    RSDSM = Data_Conn.Execute(SQLQuery)
                    Target = "../dsm/dsm.asp?ParentID=" + FormatNumber(Session("InitialParentID"),0) + "&Lvl=" +
FormatNumber(Session("InitialLvl"),0)
                    Response.Redirect Target
               CASE "C2"
                    SQLQuery = "UPDATE Issues SET Originator =" + FormatNumber(Request("UserChoice"),0)
                    SQLQuery = SQLQuery + " WHERE IssueID =" + FormatNumber(RSIssues("ParentIssueID"),0)
                    Set RSUpdate = Data_Conn.Execute(SQLQuery)

                    SQLQuery = "UPDATE Issues SET Originator = " + FormatNumber(Session("UserID"),0)
                    SQLQuery = SQLQuery + ", UserID = " + FormatNumber(Request("UserChoice"),0)
                    SQLQuery = SQLQuery + " WHERE IssueID = " + FormatNumber(Request.QueryString("IssueID"),0)
                    Set RSUpdate2 = Data_Conn.Execute(SQLQuery)

                    Set cmdTemp = Server.CreateObject("ADODB.Command")
                    Set cmdTemp.ActiveConnection = Data_Conn
                    Set rsTemp = Server.CreateObject("ADODB.Recordset")
                    cmdTemp.CommandText = "SELECT * FROM Comments"
                    cmdTemp.CommandType =1
                    rsTemp.Open cmdTemp, ,1,3
                    rsTemp.AddNew
                    rsTemp("IssueID") = FormatNumber(Request.QueryString("IssueID"),0)
                    rsTemp("ParentIssueID") = FormatNumber(RSIssues("ParentIssueID"),0)
                    rsTemp("FromID")= FormatNumber(Session("UserID"),0)
                    rsTemp("ToID")= FormatNumber(Request("UserChoice"),0)
                    rsTemp("Comments")= Request("Comments")
                    rsTemp.Update
                    rsTemp.Close
                    Response.Redirect "home.asp"
               CASE "C3"
                    SQLQuery = "UPDATE Issues SET Status=4 WHERE IssueID=" + Request("IssueID")
                    Set RSUpdate = Data_Conn.Execute(SQLQuery)
                    SQLQuery = "INSERT INTO Comments VALUES(" + FormatNumber(Request.QueryString("IssueID"),0) + ", "
                    SQLQuery = SQLQuery + FormatNumber(RSIssues("ParentIssueID"),0) + ", "
                    SQLQuery = SQLQuery + FormatNumber(Session("UserID"),0) + ", "
                    SQLQuery = SQLQuery + "-1, '"
                    SQLQuery = SQLQuery + Request("Comments") + "')"
                    Set RSInsertComment = Data_Conn.Execute(SQLQuery)
                    Response.Redirect "home.asp"
          END SELECT
     End If
ElseIf Request("Action")="Cancel" Then
     RSIssues.Close
     Response.Redirect "home.asp"
End If
SQLQuery = "SELECT Name FROM DataDictionary WHERE DataElementID=" + FormatNumber(RSIssues("DataElementID"),0)
Set RSDE = Data_Conn.Execute(SQLQuery)
If (RSIssues("Type") = "5") Then
     PU = "produce"
     PUpres = "produces "
     PUing = "producing"
     PUpast = "produced"
ElseIf (RSIssues("Type") = "6") Then
     PU = "use"
     PUpres = "uses "
     PUing = "using"
     PUpast = "used"
End If
SQLQuery = "SELECT TaskName_Long, TaskName_Short FROM DSMMaster WHERE TaskID=" + FormatNumber(RSIssues("TaskID"),0)
Set RSTask = Data_Conn.Execute(SQLQuery)
%>
<html>
<head>
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>
<body background="../dgrey042.jpg">
<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="41" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>
<table border="3" width="90%">
  <tr>
    <td width="100%"><br>
    <table border="0" ALIGN="CENTER">
      <tr>
        <td align="CENTER"><img src="../Presentation/Oval.gif" alt="Oval.gif (533 bytes)"
        WIDTH="196" HEIGHT="67"></td>
        <td align="CENTER"><img src="../images/arrow.GIF" alt="<%=Trim(RSDE("Name"))%>"
        WIDTH="138" HEIGHT="42"></td>
        <td align="CENTER"><img src="../Presentation/Oval.gif" alt="Oval.gif (533 bytes)"
        WIDTH="196" HEIGHT="67"></td>
      </tr>
      <tr>
        <td ALIGN="CENTER"><h2><%If (RSIssues("Type") = "5") Then
          Response.Write("YOUR PROCESS")
     Else
          Response.Write(Trim(RSTask("TaskName_Short")))
     End If
     %> </h2>
        </td>
        <td align="CENTER"><i><%=Trim(RSDE("Name"))%></i> </td>
        <td align="CENTER"><%If (RSIssues("Type") = "6") Then
          Response.Write("<h2>YOUR PROCESS")
     Else
          Response.Write(Trim(RSTask("TaskName_Short")))
     End If
     %>
</td>
        </tr>
    </table>
<%
SQLQuery = "SELECT Name FROM UserProfile WHERE UserID=" + FormatNumber(RSIssues("Originator"),0)
Set RSUsers = Data_Conn.Execute(SQLQuery)
%>
```

```
    <h5><%=RSUsers("Name")%> has indicated that one of your processes <%Response.Write(PUpres + " deliverable:" + "   <i>" +
Trim(RSDE("Name"))+ "</i></font>")%><br>
    <br>
    </h5>
    <form method="POST" onsubmit="return FrontPage_Form1_Validator(this)"
    name="FrontPage_Form1">
      <table border="0" width="100%">
        <tr>
          <td width="60%"><h5><u><%=Msg%></u></h5>
          <h2><input type="radio" value="C1" name="ActionChoice"> I <%Response.Write(PU + " " + "this deliverable")%> and will add it to my process
</h2>
          <h2><input type="radio" name="ActionChoice" value="C2"> The person <%Response.Write(PUing + " " + "this deliverable ")%> is <select
          name="UserChoice" size="1">
            <option selected value="-42">Please Choose...</option>
<%
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")

SQLQuery = "SELECT * FROM UserProfile"
Set RSUsers = Data_Conn.Execute(SQLQuery)
While (not RSUsers.EOF)
     Response.Write ("<option value='" + FormatNumber(RSUsers("UserID"),0) + "'>" + Trim(RSUsers("Name")) + "</option>")
     RSUsers.MoveNext
Wend
RSUsers.Close
%>          </select> </h2>
          <h2><input type="radio" name="ActionChoice" value="C3"> I do not know who <%Response.Write (PUpres + "this deliverable")%> </h2>
          <h5><u>Comments:</u><br>
          <!--webbot bot="Validation" startspan S-Display-Name="Comments" I-Maximum-Length="250" --><!--webbot
          bot="Validation" endspan --><textarea rows="7" name="Comments" cols="40"><%=Request("Comments")%></textarea></h5>
          <p> </p>
          <h5><font face="Arial" color="#000000"><input type="submit" value="  OK  " name="Action"> <input
          type="submit" value="Cancel" name="Action"></font></h5>
          <h5><%=Warn%></h5>
          </td>
          <td width="40%" align="top" align="left"><%

SQLQuery = "SELECT ParentIssueID FROM Comments WHERE IssueID=" + Request("IssueID")
Set RSComment1 = Data_Conn.Execute(SQLQuery)

SQLQuery = "SELECT * FROM Comments WHERE ParentIssueID=" + FormatNumber(RSComment1("ParentIssueID"),0) + "ORDER BY CommentID DESC"
Set RSComment = Data_Conn.Execute(SQLQuery)

While (not RSComment.EOF)
     SQLQuery = "SELECT Name, Email FROM UserProfile WHERE UserID=" + FormatNumber(RSComment("FromID"),0)
     Set RSFrom = Data_Conn.Execute(SQLQuery)
     Response.Write("--------------------------------<BR>")
     Response.Write ("From: <A HREF=mailto:" + Trim(RSFrom("Email")) + ">" + Trim(RSFrom("Name")) + "</A><BR>")
     SQLQuery = "SELECT Name, Email FROM UserProfile WHERE UserID=" + FormatNumber(RSComment("ToID"),0)
     Set RSTo = Data_Conn.Execute(SQLQuery)
     Response.Write ("To: <A HREF=mailto:" + Trim(RSTo("Email")) + ">" + Trim(RSTo("Name")) + "</A><BR>")
     Response.Write ("Comments: " + RSComment("Comments")+"<BR>")
     Response.Write("--------------------------------")
     Response.Write("<BR><BR>")
     RSComment.MoveNext
Wend
%>
</td>    </tr>   </table>   </form>   </td> </tr> </table> </center></div> </body> </html>
```

Module: /user/datacapture.asp

Description: Presents the data capture main embedded applet "DataCapture.FrontEnd.AbbyApplet.class" used to present entered tasks for addition, deletion, editing and sequence definition.

```
<%
Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
SQLQuery = "SELECT TaskName_Long FROM DSMMaster WHERE TaskID=" + Request.QueryString("ParentID")
Set RSDSM = Data_Conn.Execute(SQLQuery)
TaskName = Trim(RSDSM("TaskName_Long"))
RSDSM.Close
Data_Conn.Close
Set RSDSM = Nothing
%>
<html>
<head>
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>
<body background="../dgrey042.jpg">
<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="41" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>
<table border="3" width="70%">
  <tr>
    <td width="100%"><p align="center"><br>
    </p>
    <p align="center">
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="View My DSM">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url"
      value="../dsm/dsm.asp?Lvl=<%=Session("InitialLvl")%>&amp;ParentID=<%=Session("InitialParentID")%>"
      valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
```

```
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
      
    <applet code="fphover.class" codebase="../_fpclass/" width="130" height="24">
      <param name="text" value="My Home">
      <param name="hovercolor" value="#9BECEA">
      <param name="textcolor" value="#FFFFFF">
      <param name="effect" value="glow">
      <param name="url" value="home.asp" valuetype="ref">
      <param name="font" value="Dialog">
      <param name="fontstyle" value="bold">
      <param name="fontsize" value="14">
      <param name="bgcolor" value="#9BECEA">
      <param name="color" value="#366D6D">
    </applet>
    </p>
    <h1>Model Creation for <%=TaskName%></h1>
    <p align="center">
    <applet code="DataCapture.FrontEnd.AbbyApplet.class" codebase="../" width="399"
    height="552">
      <param name="ParentID" value="<%=FormatNumber(Request.QueryString("ParentID"), 0)%>">
      <param name="Lvl" value="<%=FormatNumber(Request.QueryString("Lvl"), 0)%>">
      <param name="UserID" value="<%=FormatNumber(Session("UserID"), 0)%>">
      <param name="Project" value="<%=Session("Project")%>">
      <param name="TaskName" value="<%=TaskName%>">
    </applet>
    </td> </tr> </table> </center></div> </body></html>
```

Module: /dsm/dsm.asp

Description: Presents the data capture main embedded applet "DataCapture.FrontEnd.AbbyApplet.class" used to present entered tasks for addition, deletion, editing and sequence definition.

```
<%
Response.Expires = 0
If (Session("NewModel") = "true") Then
     SQLQuery = "UPDATE MasterTable SET NewModel=0 WHERE SystemName = '" + Session("Project") + "'"
     Set MainAbbyDB = Server.CreateObject("ADODB.Connection")
     MainAbbyDB.Open Session("MainAbbyDB_ConnectionString"), Session("MainAbbyDB_RuntimeUserName"), Session("MainAbbyDB_RuntimePassword")
     Set RSSetOldModel = MainAbbyDB.Execute(SQLQuery)
     MainAbbyDB.Close
End If
Lvl = Request.QueryString("Lvl")
ParentID = Request.QueryString("ParentID")

Set Data_Conn = Server.CreateObject("ADODB.Connection")
Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
SQLQuery = "SELECT * FROM DSMMaster WHERE ParentID=" + ParentID + " AND Lvl=" + Lvl + " ORDER BY Rank"
Set RSDSM = Data_Conn.Execute(SQLQuery)

If ((FormatNumber(RSDSM("UserID"),0) = FormatNumber(Session ("UserID"),0)) OR (Session("isSysAdmin") = "true")) Then
     Session("AllowChange") = 1
Else
     Session("AllowChange") = 0
End If
numTasks = 0
While (not RSDSM.EOF)
     numTasks = numTasks + 1
     RSDSM.MoveNext
Wend
Width = (2*Session("Padding")) + ((((Session("CellHeight") + Session("Offset"))*numTasks))-Session("Offset")) + ((2 *
(Session("CellWidth"))+Session("Offset"))) + ((Session("CellHeight")/2) + Session("Offset"))
Height = (2*Session("Padding")) + (((Session("CellHeight") + Session("Offset"))*numTasks)-Session("Offset")) +
(Session("CellHeight")+Session("Offset"))+ Session("MenuSpace")
RSDSM.Close
Data_Conn.Close

%>
<%
IF Lvl=1 then
     ParentTaskName= "Top Level decompostion"
Else

     Set Data_Conn = Server.CreateObject("ADODB.Connection")
     Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
     SQLQuery = "SELECT TaskName_Long FROM DSMMaster WHERE TaskID=" + ParentID
     Set RSDSM = Data_Conn.Execute(SQLQuery)
     ParentTaskName= RSDSM("TaskName_Long")
     RSDSM.Close
     Data_Conn.Close
End If
%>
<%
     Messages=""
     MsgTitle=""
     MsgEnd=""
     TableWidth="0"
     If (Session("AllowChange") = 1) then
          Set Data_Conn = Server.CreateObject("ADODB.Connection")
          Data_Conn.Open Session("Data_Conn_ConnectionString"), Session("Data_Conn_RuntimeUserName"), Session("Data_Conn_RuntimePassword")
          SQLQuery = "SELECT DataElementID FROM Issues WHERE TaskID=" + ParentID + "AND Type= 3"
          Set RSDSM = Data_Conn.Execute(SQLQuery)
          While (not RSDSM.EOF)
               DataID = RSDSM("DataElementID")
               SQLQuery = "SELECT Name FROM DataDictionary WHERE DataElementID=" + FormatNumber(DataID,0)
               Set RSDSM2 = Data_Conn.Execute(SQLQuery)
               SQLQuery = "SELECT Type FROM IO WHERE DataElementID=" + FormatNumber(DataID,0) +" AND TaskID=" + FormatNumber(ParentID,0)
               Set RSDSM3 = Data_Conn.Execute(SQLQuery)
               if (RSDSM3("Type")=1) then
```

```
                depType="(Output)   "
            else
                depType="(Input)     "
            end if
            Messages = Messages + depType + "<i> " + RSDSM2("Name") + "</i><br>"
            RSDSM2.close
            RSDSM3.close
            RSDSM.MoveNext
        Wend
        RSDSM.Close
        Data_Conn.Close
        If Messages<>"" then
            MsgTitle="<h5>The following deliverables are required in your model:</h5>"
            MsgEnd="Please add them to this matrix if you haven't done so already"
            TableWidth="1"
        End If
    End If
%>
<html>
<head>
<link REL="stylesheet" HREF="../StyleSheets/mystyles.css" TYPE="text/css">
<title></title>
</head>
<body background="../dgrey042.jpg">
<p><img src="../WB01355_1.gif" width="43" height="43">    <img
src="../a_clr.gif" width="43" height="42" alt="a_clr.gif (3367 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../b_clr.gif" width="43" height="42" alt="b_clr.gif (3670 bytes)"><img
src="../y_clr.gif" width="43" height="41" alt="y_clr.gif (3259 bytes)"> </p>
<div align="center"><center>

<table border="3" width="70%">
  <tr>
    <td width="100%"><p align="center"><br>
    </p>
    <h1>Level <%=Lvl%> - <%=ParentTaskName%> </h1>
    <p align="center">
    <applet code="Presentation.DSMStart.class" codebase="../" width="<%=Width%>"
    height="<%=Height%>">
      <param name="ParentID" value="<%=FormatNumber(ParentID, 0)%>">
      <param name="Lvl" value="<%=FormatNumber(Lvl, 0)%>">
      <param name="numTasks" value="<%=FormatNumber(numTasks,0)%>">
      <param name="CellHeight" value="<%=FormatNumber(Session("CellHeight"), 0)%>">
      <param name="CellWidth" value="<%=FormatNumber(Session("CellWidth"), 0)%>">
      <param name="Padding" value="<%=FormatNumber(Session("Padding"), 0)%>">
      <param name="FontSize" value="<%=FormatNumber(Session("FontSize"), 0)%>">
      <param name="Offset" value="<%=FormatNumber(Session("Offset"), 0)%>">
      <param name="MenuSpace" value="<%=FormatNumber(Session("MenuSpace"), 0)%>">
      <param name="UserID" value="<%=FormatNumber(Session("UserID"), 0)%>">
      <param name="AllowChange" value="<%=FormatNumber(Session("AllowChange"),0)%>">
      <param name="Project" value="<%=Session("Project")%>">
    </applet>
    </p>
    <dd align="center"><br>
      <br>
      <br>
      <table border="<%=TableWidth%>" width="70%">
        <tr>
          <td width="100%"><br>
<%=MsgTitle%>          <blockquote>
            <blockquote>
<%=Messages%>
            </blockquote>
          </blockquote>
          <p><br>
          <br>
          </p>
          <h5><%=MsgEnd%>
</h5> </td> </tr>  </table>  <p><br>  <br>  </p>  </dd>  </td> </tr></table></center></div></body></html>
```

D.2 Java Applets

Object DataCapture.FrontEnd.AbbyDBConnection.class

Description: This is a critical object used througout the system to provide database connectivity. It establishes connectivity with the dbAnywhere middleware service. It returns con to calling object as a live connection to the middlware.

```
package DataCapture.FrontEnd;

import java.sql.*;

public class AbbyDBConnection{

    Connection con;

    public AbbyDBConnection(String Project) {

        String url="jdbc:dbaw://design:8889/SQL_Server/design/" + Project;
        System.out.println(url);
        try {
            String driverName = "symantec.itools.db.jdbc.Driver";
            Driver driver = (Driver)Class.forName(driverName).newInstance();
            con=DriverManager.getConnection(url,"sa","gromit70");
        }
        catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }

    public Connection getConnection() {
        return con;
    }
}
```

Object DataCapture.FrontEnd.AbbyPanel2.class

Description: Task entry and edit object. Used throughout the sysetm as a pop-up or embedded applet. Performs series of functions such as calls to supporting screens, storing entered data in the appropriate tables, editing and removing of exisiting tasks etc.

```
package DataCapture.FrontEnd;

/*
        A basic extension of the java.applet.Applet class
 */

import java.awt.*;
import java.applet.*;
import java.sql.*;
import java.util.*;
import Presentation.TaskRecord;
import Presentation.DERecord;
import java.net.*;
import symantec.itools.awt.ScrollingPanel;

public class AbbyPanel2 extends Panel
{
    public int userId=1;
    public int parentId=1;
    public int level=1;
    public AbbyApplet app;
    public AbbyDialog ad;
    public boolean submit=true;
    public Vector newTasks=new Vector();
    public Vector removedTasks=new Vector();
    public String Project;    //Mark added Project code
    public boolean ContinueWithDelete=true; //Mark added DeleteWarning
    public AbbyPanel2(AbbyApplet a) {
        this();
        app=a;
        userId=app.userId;
        parentId=app.parentId;
        level=app.level;
        Project=app.Project;
        label2.setText(app.TaskName);
        submit=true;
    }
    public AbbyPanel2(AbbyDialog d) {
        this();
        ad=d;
        userId=ad.userId;
        parentId=ad.parentId;
        Project=ad.Project;
        level=ad.level;
        int numTsks=ad.numTsks;
        submit=false;
        for (int i=0;i<numTsks;i++) {
            Task t=new Task();
            t.de=new Vector();
            t.shortName=ad.tsks[i].TaskNameShort;
            t.longName=ad.tsks[i].TaskNameLong;
            t.responsiblePersonShort=ad.tsks[i].ResponsibleShort;
            t.responsiblePersonLong=ad.tsks[i].ResponsibleLong;
            t.rank=ad.tsks[i].Rank;
    //Nader 09/18/98
            t.TaskID=ad.tsks[i].TaskID;
            Vector vi=ad.des[i][0];
            Vector vo=ad.des[i][1];
            for(int j=0;j<vi.size();j++) {
                DERecord dr=(DERecord)(vi.elementAt(j));
                DataElement de=new DataElement(dr.Name,dr.Description,true);
                t.de.addElement(de);
            }
            for(int j=0;j<vo.size();j++) {
                DERecord dr=(DERecord)(vo.elementAt(j));
                DataElement de=new DataElement(dr.Name,dr.Description,false);
                t.de.addElement(de);
            }
            tsks.addElement(t);
            list1.addItem(ad.tsks[i].TaskNameShort);
            list2.addItem(String.valueOf(ad.tsks[i].Rank));
        }
    }

      public AbbyPanel2()
      {

              // This code is automatically generated by Visual Cafe when you add
              // components to the visual environment. It instantiates and initializes
              // the components. To modify the code, only use code syntax that matches
              // what Visual Cafe can generate, or Visual Cafe may be unable to back
              // parse your Java file into its visual environment.
              //{{INIT_CONTROLS
              setLayout(null);
              setSize(411,552);
              setFont(new Font("Dialog", Font.PLAIN, 14));
              setBackground(new Color(12632256));
              addButton = new java.awt.Button();
              addButton.setActionCommand("button");
```

```
                addButton.setLabel("Add");
                addButton.setBounds(304,110,84,38);
                addButton.setBackground(new Color(12632256));
                add(addButton);
                removeButton = new java.awt.Button();
                removeButton.setActionCommand("button");
                removeButton.setLabel("Remove");
                removeButton.setBounds(304,153,84,38);
                removeButton.setBackground(new Color(12632256));
                add(removeButton);
                editButton = new java.awt.Button();
                editButton.setActionCommand("button");
                editButton.setLabel("Edit");
                editButton.setBounds(304,196,84,38);
                editButton.setBackground(new Color(12632256));
                add(editButton);
                label1 = new java.awt.Label("Tasks:",Label.CENTER);
                label1.setBounds(108,72,100,26);
                label1.setFont(new Font("Dialog", Font.BOLD, 16));
                add(label1);
                label2 = new java.awt.Label("Abby Task Entry System",Label.CENTER);
                label2.setBounds(1,12,408,51);
                label2.setFont(new Font("Dialog", Font.BOLD, 16));
                label2.setForeground(new Color(0));
                add(label2);
                submitButton = new java.awt.Button();
                submitButton.setActionCommand("button");
                submitButton.setLabel("Save");
                submitButton.setBounds(304,240,84,38);
                submitButton.setBackground(new Color(12632256));
                add(submitButton);
                quitButton = new java.awt.Button();
                quitButton.setActionCommand("button");
                quitButton.setLabel("Cancel");
                quitButton.setBounds(304,288,84,38);
                quitButton.setBackground(new Color(12632256));
                add(quitButton);
                list1 = new java.awt.List(4);
                add(list1);
                list1.setBounds(48,108,228,384);
                list1.setBackground(new Color(16777215));
                list2 = new java.awt.List(4);
                add(list2);
                list2.setBounds(22,108,24,384);
                list2.setBackground(new Color(16777215));
                list2.setEnabled(false);
                label3 = new java.awt.Label("Sequence:",Label.CENTER);
                label3.setBounds(0,72,72,26);
                label3.setFont(new Font("Dialog", Font.BOLD, 12));
                add(label3);
                //}}
                System.out.println(1);
    //DataCache.initialize(new AbbyDBConnection());
          System.out.println(2);
                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                addButton.addActionListener(lSymAction);
                removeButton.addActionListener(lSymAction);
                editButton.addActionListener(lSymAction);
                SymItem lSymItem = new SymItem();
                list1.addItemListener(lSymItem);
                quitButton.addActionListener(lSymAction);
                submitButton.addActionListener(lSymAction);
                //}}
        }
        //{{DECLARE_CONTROLS
        java.awt.Button addButton;
        java.awt.Button removeButton;
        java.awt.Button editButton;
        java.awt.Label label1;
        java.awt.Label label2;
        java.awt.Button submitButton;
        java.awt.Button quitButton;
        java.awt.List list1;
        java.awt.List list2;
        java.awt.Label label3;
        //}}
  public Vector tsks=new Vector();
        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent event)
                {
                        Object object = event.getSource();
                        ew.hide();
                        if (object == addButton)
                                button1_Action(event);
                        else if (object == removeButton)
                                button2_Action(event);
                        else if (object == editButton)
                                button3_Action(event);
                        else if (object == quitButton)
                                quitButton_Action(event);
                        else if (object == submitButton)
                                submitButton_Action(event);
                }
        }
        void button1_Action(java.awt.event.ActionEvent event)
        {
            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
            System.out.println("Mark 1");
                TaskWizard tw=new TaskWizard(this);
                System.out.println("Mark 2");
                tw.start();
            System.out.println("Mark 3");
            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        }
        void button2_Action(java.awt.event.ActionEvent event)
        {
```

```
    setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
    int pos=list1.getSelectedIndex();
    //Mark added If statement
    if (pos > -1){
    list1.delItem(pos);
          int g=Integer.parseInt(list2.getItem(pos));
          list2.delItem(pos);
          Task tk=(Task)(tsks.elementAt(pos));
          if (newTasks.contains(tk)) {
              newTasks.removeElement(tk);
          }
          else {
              removedTasks.addElement(tk);
          }
          tsks.removeElementAt(pos);
          for(int i=0;i<tsks.size();i++) {
              if (((Task)(tsks.elementAt(i))).rank>g) {
                  Task t=(Task)(tsks.elementAt(i));
                  t.rank=t.rank-1;
                  tsks.removeElementAt(i);
                  tsks.insertElementAt(t,i);
                  list2.replaceItem(Integer.toString(t.rank),i);
              }
          }
}
          setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
}
void button3_Action(java.awt.event.ActionEvent event)
{
    setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
    //Mark added If statement
    if (list1.getSelectedIndex() > -1)
              (new TaskInfoDialog(this, true, (Task)(tsks.elementAt(list1.getSelectedIndex())))).show();
    setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
}
class SymItem implements java.awt.event.ItemListener
{
          public void itemStateChanged(java.awt.event.ItemEvent event)
          {
                    Object object = event.getSource();
                    if (object == list1)
                              list1_ItemStateChanged(event);
          }
}
void list1_ItemStateChanged(java.awt.event.ItemEvent event)
{
          // to do: code goes here.
          ew.hide();
          String in="";
          try {
              in=list2.getItem(list1.getSelectedIndex());
          }
          catch(Exception e) {
              in="";
          }
          ew=new EditWindow(this,in,list1.getSelectedIndex());
          ew.show(getLocationOnScreen().x+24,getLocationOnScreen().y+108+list1.getSelectedIndex()*15);
}
EditWindow ew=new EditWindow(this, "",0);

void quitButton_Action(java.awt.event.ActionEvent event)
{
    setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
          setVisible(false);
          try{
          if (submit) {
              app.getAppletContext().showDocument(new URL("http://design.mit.edu/Abby/user/home.asp"));
          }
          else {
              ad.hide();
          }
          }
          catch(Exception e) {
              System.out.println(e.toString());
          }
}
void submitButton_Action(java.awt.event.ActionEvent event)
{
    AbbyDBConnection aDBc=new AbbyDBConnection(Project);
    Connection con=aDBc.getConnection();
    /*try {
        PreparedStatement pst2=con.prepareStatement("INSERT INTO DataDictionary " +
                                  "VALUES (?,"+userId+",?)");
        Vector des=DataCache.getNewElements();
        System.out.println("HERE "+des.size());
        for(int y=0;y<des.size();y++) {
            DataElement de=(DataElement)(des.elementAt(y));
            System.out.println("DENAME: "+de.name);
            pst2.setString(1,de.name);
            pst2.setString(2,de.description);
            pst2.executeUpdate();
        }
        pst2.close();
    }
    catch (Exception e) {
        System.out.println(e.toString());
    }*/
    if (submit) {
        PreparedStatement pst=null;
        try {
            int lev=level;
            String s="INSERT INTO DSMMASTER " + "VALUES ("+userId+","+parentId+
                                              ",0,0,?,?,?,?,?,"+lev+")";
            pst=con.prepareStatement(s);
            PreparedStatement st=con.prepareStatement("SELECT TaskID FROM DSMMASTER "+
                                                 "WHERE TaskName_Short LIKE ?");
            int x=0;
            while (x<tsks.size()) {
```

```
            Task t=(Task)(tsks.elementAt(x));
            pst.setString(1,t.shortName);
            pst.setString(2,t.longName);
            pst.setString(3,t.responsiblePersonShort);
            pst.setString(4,t.responsiblePersonLong);
            pst.setInt(5,t.rank);
            pst.executeUpdate();
            x++;
        }
        System.out.println("Done DSMMaster");
            //con.commit();
        /*x=0;
        Vector tIDs=new Vector();
        while (x<tsks.size()) {
            Task t=(Task)(tsks.elementAt(x));
            st.setString(1,t.shortName);
            ResultSet rs=st.executeQuery();
            //con.commit();
            rs.next();
            tIDs.addElement(new Integer(rs.getInt(1)));
            x++;
        }
        x=0;
        while (x<tsks.size()){
            Task t=(Task)(tsks.elementAt(x));
            int y=0;
            Vector rts=new Vector();
            PreparedStatement st2=con.prepareStatement("SELECT DataElementID FROM DATADICTIONARY "+
                                                       "WHERE Name LIKE ?");
            while(y<t.de.size()) {
                DataElement de=(DataElement)(t.de.elementAt(y));
                st2.setString(1,de.name);
                ResultSet rs2=st2.executeQuery();
                System.out.println(de.name);
                rs2.next();
                System.out.println(de.name);
                ResultType2 rt2=new ResultType2();
                rt2.dID=rs2.getInt(1);
                rt2.tID=((Integer)(tIDs.elementAt(x))).intValue();
            if (de.input) rt2.input=0;
            else rt2.input=1;
            rts.addElement(rt2);
                y++;
                //rs2.close();
            }
            con.commit();
            st2.close();
            PreparedStatement pst3=con.prepareStatement("INSERT INTO IO VALUES (?,?,?,9)");
            for(int i=0;i<rts.size();i++) {
                ResultType2 rt2=(ResultType2)(rts.elementAt(i));
                pst3.setInt(1,rt2.tID);
                pst3.setInt(2,rt2.dID);
                pst3.setInt(3,rt2.input);
                boolean comd=false;
                while (!comd) {
                try {
                    pst3.executeUpdate();
                    comd=true;
                    System.out.println("Hit!");
                }
                catch (Exception e) {
                    System.out.println("Missed!");
                }
                }
            }
            //pst3.close();
            x++;
            //con.commit();
        }
        //pst.close();
        //pst3.close();
        //st.close();
        //st2.close();
        con.commit();*/
        lev=lev;


        app.getAppletContext().showDocument(new URL("http://design.mit.edu/Abby/dsm/dsm.asp?Lvl="+lev+"&ParentID="+parentId));
        PreparedStatement sm=con.prepareStatement("UPDATE DSMMaster SET Flag=1 WHERE TaskID=?");
        //Mark addition
        PreparedStatement sm2=con.prepareStatement("DELETE FROM Issues WHERE Type=9 AND TaskID=?");
        sm.setInt(1,parentId);
        sm.executeUpdate();
        sm2.setInt(1,parentId);
        sm2.executeUpdate();
    }
    catch (Exception e) {
        System.out.println(e.toString());
    }
}
else {
    System.out.println(removedTasks.size());
    if (!removedTasks.isEmpty()){
        (new DeleteWarning(this, true)).show();
    }
    System.out.println("ContinueWithDeleteqq--" + ContinueWithDelete);
    if (ContinueWithDelete)
    {
        System.out.println("CWDSTART");
    try {
        setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        //PreparedStatement npst1=con.prepareStatement("INSERT INTO NewTasks VALUES ("+
        //                    userId+","+parentId+",?,?,?,?,"+level+")");
        PreparedStatement npst1=con.prepareStatement("INSERT INTO DSMMaster VALUES ("+
                            userId+","+parentId+",0,0,?,?,?,?,?," + level + ")");
        //PreparedStatement npst2=con.prepareStatement("INSERT INTO RemovedTasks VALUES (?,"+
        //                    userId+")");
        PreparedStatement npst2=con.prepareStatement("DELETE FROM DSMMaster WHERE TaskID=?");
```

```
                    PreparedStatement npst2b=con.prepareStatement("DELETE FROM IO WHERE TaskID=?");
                    PreparedStatement npst2c=con.prepareStatement("DELETE FROM Issues WHERE TaskID=?");

                    PreparedStatement npst3=con.prepareStatement("INSERT INTO NewDEs VALUES (?,?,"+
                                        userId+")");
                    PreparedStatement npst4=con.prepareStatement("INSERT INTO RemovedDEs VALUES (?,"+
                                        userId+")");
                   // PreparedStatement npst5=con.prepareStatement("UPDATE DSMMaster SET Rank=? WHERE TaskName_Long LIKE ?");
                   //Nader 09/18/98
                    PreparedStatement npstN=con.prepareStatement("UPDATE DSMMaster SET Rank=?,
TaskName_Short=?,TaskName_Long=?,Responsible_Short=?,Responsible_Long=? WHERE TaskID=?");

                    PreparedStatement nst1=con.prepareStatement("SELECT TaskID FROM DSMMaster "+
                                        "WHERE TaskName_Short LIKE ?");
                    PreparedStatement nst2=con.prepareStatement("SELECT DataElementID FROM DataDictionary "+
                                        "WHERE Name LIKE ?");
                    System.out.println("test0");
                    for(int i=0;i<removedTasks.size();i++) {
                        //Mark put the two following statements here to re-init for loop
                        System.out.println("test1");
                        nst1=con.prepareStatement("SELECT TaskID FROM DSMMaster WHERE TaskName_Short LIKE ?");
                        npst2=con.prepareStatement("DELETE FROM DSMMaster WHERE TaskID=?");
                        npst2b=con.prepareStatement("DELETE FROM IO WHERE TaskID=?");
                        npst2c=con.prepareStatement("DELETE FROM Issues WHERE TaskID=?");
                        System.out.println("test2");
                        Task rt=(Task)(removedTasks.elementAt(i));
                        System.out.println("this is the tname -- word -- " + rt.shortName);
                        nst1.setString(1,rt.shortName);
                        ResultSet rs=nst1.executeQuery();
                        rs.next();
                        System.out.println("this is the tid --" + rs.getInt(1));
                        int tID=rs.getInt(1);
                        npst2.setInt(1,tID);
                        npst2.executeUpdate();
                        npst2b.setInt(1,tID);
                        npst2b.executeUpdate();
                        npst2c.setInt(1,tID);
                        npst2c.executeUpdate();
                        for (int j=0;j<rt.de.size();j++) {
                            DataElement rd=(DataElement)(rt.de.elementAt(j));
                            nst2.setString(1,rd.name);
                            ResultSet rs2=nst2.executeQuery();
                            rs2.next();
                            int dID=rs2.getInt(1);
                            npst4.setInt(1,dID);
                        }
                    }
                    for(int i=0;i<newTasks.size();i++) {
                        System.out.println("test5");
                        Task nt=(Task)(newTasks.elementAt(i));
                        npst1.setString(1,nt.shortName);
                        npst1.setString(2,nt.longName);
                        npst1.setString(3,nt.responsiblePersonShort);
                        npst1.setString(4,nt.responsiblePersonLong);
                        npst1.setInt(5,nt.rank);
                        npst1.executeUpdate();
                        nst1.setString(1,nt.shortName);
/*                      for (int j=0;j<nt.de.size();j++) {
                            System.out.println("test8");
                            DataElement nd=(DataElement)(nt.de.elementAt(j));
                            nst2.setString(1,nd.name);
                            ResultSet rs2=nst2.executeQuery();
                            rs2.next();
                            int dID=rs2.getInt(1);
                            npst3.setInt(1,dID);
                            if (nd.input) {
                                npst3.setInt(2,0);
                            }
                            else {
                                npst3.setInt(2,1);
                            }
                            npst3.executeUpdate();
                        }
                        System.out.println("test9");*/
                    }

                    for(int i=0;i<tsks.size();i++) {

                        Task t=(Task)(tsks.elementAt(i));

                        npstN.setInt(1,t.rank);
                        npstN.setString(2,t.shortName);
                        npstN.setString(3,t.longName);
                        npstN.setString(4,t.responsiblePersonShort);
                        npstN.setString(5,t.responsiblePersonLong);
                        npstN.setInt(6,t.TaskID);

                        npstN.executeUpdate();
                    }
                    npst1.close();
                    npst2.close();
                    npst3.close();
                    npst4.close();
                    npstN.close();
                    nst1.close();
                    nst2.close();
                }
                catch (Exception e) {
                    System.out.println(e.toString());
                }

                    setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                ad.hide();
            }
            }
        }
}
```

Object DataCapture.Presentation.Loopz.class

Description: Main applet for displaying the DSM. Includes all user interface functions designed for the matrix such as, color coordinate tracking, pop-up envokes, matrix layout, dependency populatoin, matrix editing calls, saving matrix data, decomposition etc.

```
/*
        A basic extension of the java.awt.Frame class
 */
package Presentation;
import DataCapture.FrontEnd.AbbyDialog;
import java.awt.*;
import java.applet.*;
import java.sql.*;
import java.util.*;
import java.net.*;
import java.lang.*;

public class Loopz extends Panel
{
    int ParentID;
    int Lvl;
    int numTasks;
    int CellHeight;
    int CellWidth;
    int Padding;
    int FontSize;
    int Offset;
    int MenuSpace;          //Should almost always be above 30--definitely always above 22.
    int UserID;
    boolean AllowChange;
    String IntersectionChar;
    String BreakdownChar ;
    int maxTasks = 25;
    boolean DataChanged=false;
    DSMStart pFrame;
    String Project;
    Label [] aResponsible = new Label[maxTasks];
    Label [] aSerialParallel = new Label[maxTasks];
          Label [] aTaskLabels = new Label[maxTasks];
          TaskRecord [] aTaskRecords = new TaskRecord[maxTasks];
          Vector [][] aDataElements = new Vector[maxTasks][maxTasks];
    Vector [][] aDiagonal = new Vector[maxTasks][2];
    Label [][] aTasks = new Label[maxTasks][maxTasks];
          public Loopz(DSMStart parent)
          {
              int i=0;
                    int j=0;
                    int k=0;
                pFrame = parent;
                    //{{INIT_CONTROLS
                    for (i=0; i<maxTasks; i++)
                        {
                            aDiagonal[i][0] = new Vector();
                            aDiagonal[i][1] = new Vector();
                            aTaskRecords[i] = new TaskRecord();
                            for (j=0; j<maxTasks; j++)
                                aDataElements[i][j]= new Vector();
                        }
        ParentID = pFrame.ParentID;
        Lvl = pFrame.Lvl;
        numTasks = pFrame.numTasks;
        CellHeight = pFrame.CellHeight;
        CellWidth = pFrame.CellWidth;
        Padding = pFrame.Padding;
        FontSize = pFrame.FontSize;
        Offset = pFrame.Offset;
        UserID = pFrame.UserID;
        MenuSpace = pFrame.MenuSpace;
        AllowChange = pFrame.AllowChange;
        IntersectionChar = pFrame.IntersectionChar;
        BreakdownChar = pFrame.BreakdownChar;
        //maxTasks = pFrame.maxTasks;
        aTaskRecords = pFrame.aTaskRecords;
        aDataElements = pFrame.aDataElements;
        aDiagonal = pFrame.aDiagonal;
        Project = pFrame.Project;
        System.out.println("INLOOPZ -- " + Project);
                    setLayout(null);
                    //setVisible(false);
                    setSize((2*Padding) + (((CellHeight + Offset)*numTasks)-Offset) + (2 * (CellWidth+Offset)) + ((CellHeight/2) + Offset),
(2*Padding) + (((CellHeight + Offset) * numTasks)-Offset) + (CellHeight+Offset)+ MenuSpace);
                    resize((2*Padding) + (((CellHeight + Offset)*numTasks)-Offset) + (2 * (CellWidth+Offset)) + ((CellHeight/2) + Offset),
(2*Padding) + (((CellHeight + Offset) * numTasks)-Offset) + (CellHeight+Offset)+ MenuSpace);
                    setBackground(Color.white);
                    MatrixPanel = new java.awt.Panel();
                    MatrixPanel.setLayout(null);
                    MatrixPanel.setBounds(0,0, (2*Padding) + (((CellHeight + Offset)*numTasks)-Offset) + (2 * (CellWidth+Offset)) + ((CellHeight/2) +
Offset), (2*Padding) + (((CellHeight + Offset) * numTasks)-Offset) + (CellHeight+Offset));
                    MatrixPanel.setBackground(Color.black);
                    add(MatrixPanel);
                    //Write Responsible Header
                    lblResponsibleHeader = new java.awt.Label("Responsible Group",Label.CENTER);
                    lblResponsibleHeader.setBounds(Padding, Padding, CellWidth, CellHeight);
                    lblResponsibleHeader.setFont(new Font("Dialog", Font.PLAIN, FontSize));
                    lblResponsibleHeader.setBackground(Color.white);
```

```
                MatrixPanel.add(lblResponsibleHeader);
                //Write Serial/Parallel Header
            lblSerialParallel = new java.awt.Label("",Label.CENTER);
                lblSerialParallel.setBounds(Padding + (CellWidth +Offset), Padding , CellHeight / 2, CellHeight);
                lblSerialParallel.setFont(new Font("Dialog", Font.PLAIN, FontSize));
                lblSerialParallel.setBackground(Color.white);
                MatrixPanel.add(lblSerialParallel);
                //Write Task Name Header
                lblTaskNameHeader = new java.awt.Label("Task Name",Label.CENTER);
                lblTaskNameHeader.setBounds(Padding + (CellWidth +Offset) + ((CellHeight/2) + Offset), Padding , CellWidth, CellHeight);
                lblTaskNameHeader.setFont(new Font("Dialog", Font.PLAIN, FontSize));
                lblTaskNameHeader.setBackground(Color.white);
                MatrixPanel.add(lblTaskNameHeader);
                //Write Matrix Header
                lblMatrixHeader = new java.awt.Label("Level " + Lvl ,Label.CENTER);
                lblMatrixHeader.setBounds(Padding + (2*(CellWidth +Offset)+ ((CellHeight/2) + Offset)), Padding , (numTasks * (CellHeight
+Offset))-Offset, CellHeight);
                lblMatrixHeader.setFont(new Font("Dialog", Font.PLAIN, FontSize));
                lblMatrixHeader.setBackground(Color.white);
                MatrixPanel.add(lblMatrixHeader);
                //Write Responsible Groups
                for (i=0; i<numTasks; i++)
                {
                    aResponsible[i] = new java.awt.Label(aTaskRecords[i].ResponsibleLong);
                    aResponsible[i].setBackground(Color.white);
                    aResponsible[i].setFont(new Font("Dialog", Font.PLAIN, FontSize));
                    aResponsible[i].setBounds(Padding , Padding + ( i* (CellHeight+Offset)) + (CellHeight+Offset),CellWidth , CellHeight);
                    MatrixPanel.add(aResponsible[i]);
                }
                //Write Task Names
                for (i=0; i<numTasks; i++)
                {
                    aTaskLabels[i] = new java.awt.Label(aTaskRecords[i].TaskNameShort);
                    aTaskLabels[i].setBackground(Color.white);
                    aTaskLabels[i].setFont(new Font("Dialog", Font.PLAIN, FontSize));
                    aTaskLabels[i].setBounds(Padding + (CellWidth+Offset) + ((CellHeight/2) + Offset), Padding + (i *(CellHeight+Offset))+
(CellHeight + Offset),CellWidth ,CellHeight);
                    MatrixPanel.add(aTaskLabels[i]);
                }
                //Write Serial/Parallel Strip
                int LastRank = aTaskRecords[0].Rank;
                boolean BlackSquare = true;
                for (i=0; i<numTasks; i++)
                {
                    aSerialParallel[i] = new java.awt.Label("");
                    aSerialParallel[i].setFont(new Font("Dialog", Font.PLAIN, FontSize));
                    aSerialParallel[i].setBounds(Padding + (CellWidth+Offset), Padding + (i *(CellHeight+Offset))+ (CellHeight +
Offset),CellHeight/2 ,CellHeight + (Offset - 1));
                    if (LastRank == aTaskRecords[i].Rank)
              BlackSquare = BlackSquare;
                else
                    BlackSquare = !BlackSquare;
                if (BlackSquare)
                aSerialParallel[i].setBackground(Color.black);
            else
              aSerialParallel[i].setBackground(Color.white);
                    MatrixPanel.add(aSerialParallel[i]);
                    LastRank = aTaskRecords[i].Rank;
                }
            //Draw Matrix from aDataElements and aDiagonal Info

                for (i=0; i<numTasks; i++)
                    for (j=0; j<numTasks; j++)
                    {
                       if (aDataElements[i][j].isEmpty())
                            aTasks[i][j] = new java.awt.Label("", 1);
                       else
                            aTasks[i][j] = new java.awt.Label(IntersectionChar, 1);
                       if (i == j)
                       {
                        aTasks[i][j].setForeground(Color.white);
                        aTasks[i][j].setBackground(Color.blue);
                        if (aTaskRecords[i].Flag == 1)
                           aTasks[i][j].setText(BreakdownChar);
                      }
                      else
                      aTasks[i][j].setBackground(Color.white);
                          aTasks[i][j].setFont(new Font("Dialog", Font.PLAIN, FontSize));
                          aTasks[i][j].setBounds(Padding + ((2*CellWidth)+Offset+Offset) + ((CellHeight/2) + Offset) + (j*(CellHeight+Offset)),
Padding + (i *(CellHeight+Offset)) + (CellHeight + Offset),CellHeight ,CellHeight);
                          MatrixPanel.add(aTasks[i][j]);
                      }
                  for (i=0; i<numTasks; i++)
                      for(j=0; j<numTasks; j++)
                         aTasks[i][j].repaint();
          validate();
          MatrixPanel.repaint();
            //Add Menu Buttons
            cmdBack = new java.awt.Button();
                cmdBack.setLabel(new String ("Up"));
                cmdBack.setBounds (0 * ((MatrixPanel.getBounds().width - 74)/4),MatrixPanel.getBounds().height + (MenuSpace-22),74,22);
                cmdBack.setBackground(new Color(12632256));
                if (ParentID == 0)
                    cmdBack.setEnabled(false);
                else
                    cmdBack.setEnabled(true);
                add(cmdBack);
                cmdSave = new java.awt.Button();
                cmdSave.setLabel("Save");
                cmdSave.setBounds(3 * ((MatrixPanel.getBounds().width - 74)/4),MatrixPanel.getBounds().height + (MenuSpace-22),74,22);
                cmdSave.setBackground(new Color(12632256));
                cmdSave.setEnabled(false);
                add(cmdSave);
                cmdOptimize = new java.awt.Button();
                cmdOptimize.setLabel("Sequence");
                cmdOptimize.setBounds (2 *((MatrixPanel.getBounds().width - 74)/4),MatrixPanel.getBounds().height + (MenuSpace-22),74,22);
                cmdOptimize.setBackground(new Color(12632256));
                //cmdOptimize.setEnabled(false);
```

```
                add(cmdOptimize);
                cmdHome = new java.awt.Button();
                cmdHome.setLabel("My Home");
                cmdHome.setBounds (4 *((MatrixPanel.getBounds().width - 74)/4),MatrixPanel.getBounds().height + (MenuSpace-22),74,22);
                cmdHome.setBackground(new Color(12632256));
                cmdHome.setEnabled(true);
                add(cmdHome);
                cmdEdit = new java.awt.Button();
                cmdEdit.setLabel("Edit Tasks");
                cmdEdit.setBounds(1 *((MatrixPanel.getBounds().width - 74)/4),MatrixPanel.getBounds().height + (MenuSpace-22),74,22);
    cmdEdit.setBackground(new Color(12632256));
    cmdEdit.setEnabled(AllowChange);
    add(cmdEdit);
                //}}
                //{{REGISTER_LISTENERS
                SymWindow aSymWindow = new SymWindow();
                SymMouse aSymMouse = new SymMouse();
                //Listeners for Connections
                for (i=0; i<numTasks; i++)
                {
                    aTaskLabels[i].addMouseListener(aSymMouse);
                    aResponsible[i].addMouseListener(aSymMouse);
                    for (j=0; j<numTasks; j++)
                        aTasks[i][j].addMouseListener(aSymMouse);
            }
            cmdSave.addMouseListener(aSymMouse);
            cmdBack.addMouseListener(aSymMouse);
            cmdOptimize.addMouseListener(aSymMouse);
            cmdHome.addMouseListener(aSymMouse);
            cmdEdit.addMouseListener(aSymMouse);
            //}}
        }
public void DeleteDE_aDiagonal (int TaskNum, DERecord DE)
{
 int j, k;

  for (j=0; j<2; j++)
    for (k=0; k<aDiagonal[TaskNum][j].size(); k++)
        if (((DERecord)aDiagonal[TaskNum][j].elementAt(k)).ID == DE.ID)
            {
                aDiagonal[TaskNum][j].removeElementAt(k);
                break;
            }
}
public DERecord FindDE_Database (String DEName)
{
    DERecord FoundDE = new DERecord();
    AbbyDBConnection adbc=new AbbyDBConnection(Project);
    Connection con=adbc.getConnection();
      ResultSet rs=null;
    Statement stm=null;
    try
        {
            stm=con.createStatement();
            String FindDE="SELECT *"+
            " FROM DataDictionary\n"+
            " WHERE Name LIKE '" + DEName + "'\n";
            System.out.println(FindDE);
            rs=stm.executeQuery(FindDE);
            rs.next();
            FoundDE.ID = rs.getInt("DataElementID");
            FoundDE.Name = rs.getString("Name").trim();
            FoundDE.UserID = rs.getInt("UserID");
            FoundDE.Description = rs.getString("Description");
            rs.close();
        }
    catch (Exception e)
        {
            System.out.println(e.toString());
        }
      return FoundDE;
    }
public int FindTaskID_aTaskRecords (int Index)
{
    int i;
    for (i=0; i<numTasks; i++)
    {
        if (aTaskRecords[i].TaskID == Index)
        break;
    }
    return i;
}
public void RestoreConnections()
{
    int i, j, k;
    int ii, jj, kk;
    //Clear all current connections
    for (i=0; i<numTasks; i++)
        for (j=0; j<numTasks; j++)
            aDataElements[i][j].removeAllElements();
    //Cycle through all outputs, match them up with inputs
    for (i=0; i<numTasks; i++)
    {
        for (k=0; k<aDiagonal[i][1].size(); k++)
        {
            for (ii=0; ii<numTasks; ii++)
            {
                for (kk=0; kk<aDiagonal[ii][0].size(); kk++)
                {
                    if (((DERecord)aDiagonal[i][1].elementAt(k)).ID == ((DERecord)aDiagonal[ii][0].elementAt(kk)).ID)
                    {
                      //  ((DERecord)aDiagonal[ii][0].elementAt(kk)).OutputFrom = i;
                        aDataElements[i][ii].addElement(aDiagonal[ii][0].elementAt(kk));
                    }
                }
            }
        }
    }
```

```
    }
    public void RefreshMatrix()
    {
        int i;
        int j;

    //ReDraw Matrix from aDataElements and aDiagonal Info
                    for (i=0; i<numTasks; i++)
                        for (j=0; j<numTasks; j++)
                        {
                            if (i != j)             //Don't touch diagonals
                            {
                           if (aDataElements[i][j].isEmpty())
                                aTasks[i][j].setText("");
                           else
                                aTasks[i][j].setText(IntersectionChar);
                        }
                 }
    }
    public void UpdateDatabase()
    {
        if (DataChanged)
        {
            int i;
            int j;
            int k;
            AbbyDBConnection adbc=new AbbyDBConnection(Project);
            Connection con=adbc.getConnection();
          ResultSet rs=null;
            Statement stm=null;
            try
                {
                    stm=con.createStatement();
                    String DeleteCall="DELETE FROM IO\n"+
                    " FROM DSMMaster\n"+
                    " WHERE IO.TaskID = DSMMaster.TaskID\n"+
                    " AND DSMMaster.ParentID=" + ParentID + " AND DSMMaster.Lvl=" + Lvl;
                    System.out.println(DeleteCall);
                    rs=stm.executeQuery(DeleteCall);
                    stm=con.createStatement();
                    String InsertCall=" ";
                    for (i=0; i<numTasks; i++)
                    {
                        for (k=0; k<aDiagonal[i][0].size(); k++)        //Loop to write back inputs
                        {
                            InsertCall += " INSERT INTO IO VALUES (" + ((DERecord)aDiagonal[i][0].elementAt(k)).InputTo + "," +
((DERecord)aDiagonal[i][0].elementAt(k)).ID + ",0,9)\n";
                        }
                    }
                    for (i=0; i<numTasks; i++)
                    {
                        for (k=0; k<aDiagonal[i][1].size(); k++)        //Loop to write back outputs
                        {
                            InsertCall += " INSERT INTO IO VALUES (" + ((DERecord)aDiagonal[i][1].elementAt(k)).OutputFrom + "," +
((DERecord)aDiagonal[i][1].elementAt(k)).ID + ",1,9)\n";
                        }
                    }
                    System.out.println(InsertCall);
                    rs=stm.executeQuery(InsertCall);
                    rs.close();
                }
                catch (Exception e)
                {
                    System.out.println(e.toString());
                }
        }

    }
          public synchronized void show()
          {
                    move(50, 50);
                    super.show();
          }
          public void addNotify()
          {
              // Record the size of the window prior to calling parents addNotify.
              Dimension d = getSize();
                    super.addNotify();
                    if (fComponentsAdjusted)
                              return;
                    // Adjust components according to the insets
                    setSize(insets().left + insets().right + d.width, insets().top + insets().bottom + d.height);
                    Component components[] = getComponents();
                    for (int i = 0; i < components.length; i++)
                    {
                              Point p = components[i].getLocation();
                              p.translate(insets().left, insets().top);
                              components[i].setLocation(p);
                    }
                    fComponentsAdjusted = true;
          }
    // Used for addNotify check.
          boolean fComponentsAdjusted = false;
                    //{{DECLARE_CONTROLS
                    java.awt.Label lblTaskNameHeader;
                    java.awt.Label lblResponsibleHeader;
                    java.awt.Label lblMatrixHeader;
                    java.awt.Label lblSerialParallel;
                    java.awt.Panel MatrixPanel;
                    java.awt.Button cmdSave;
                    java.awt.Button cmdBack;
              java.awt.Button cmdOptimize;
              java.awt.Button cmdHome;
              java.awt.Button cmdEdit;
          //}}
          //{{DECLARE_MENUS
          //}}
          class SymWindow extends java.awt.event.WindowAdapter
```

```
{
        public void windowClosing(java.awt.event.WindowEvent event)
        {
                Object object = event.getSource();
                if (object == Loopz.this)
                        Frame1_WindowClosing(event);
        }
}
void Frame1_WindowClosing(java.awt.event.WindowEvent event)
{
}
class SymMouse extends java.awt.event.MouseAdapter
{
        public void mouseClicked(java.awt.event.MouseEvent event)
        {
            Object object = event.getSource();
            if (object == cmdSave)
                cmdSave_Clicked();
            else if (object == cmdBack)
                cmdBack_Clicked();
            else if (object == cmdEdit)
                cmdEdit_Clicked();
            else if (object == cmdHome)
                cmdHome_Clicked();
            else if (object == cmdOptimize)
                optimize();
            else
                MatrixClicked(object);
    }
        public void mouseEntered(java.awt.event.MouseEvent event)
        {
                Object object = event.getSource();
                for(int i=0; i<numTasks; i++)
                {
                    if ((object == aTaskLabels[i]) || (object == aResponsible[i]))
                    {
                        aTaskLabels[i].setBackground(Color.lightGray);
                        //aResponsible[i].setBackground(Color.lightGray);
                        //aTaskLabels[i].setFont(new Font("Dialog", Font.BOLD, FontSize));
                    }
                    else
                    {
                    for (int j=0; j<numTasks; j++)
                        if (object == aTasks[i][j])
                        {
                            if (i < j) {
                  aTasks[i][j].setBackground(Color.green);
                                aTaskLabels[i].setBackground(Color.green);
                            aTaskLabels[j].setBackground(Color.green.brighter());
                          }
                          else
                          {
                              if (i==j){
                                        aTasks[i][j].setBackground(Color.lightGray);
                    aTaskLabels[i].setBackground(Color.lightGray);
                                    aTaskLabels[j].setBackground(Color.lightGray);
                                }
                                else {
                                        aTasks[i][j].setBackground(Color.red);
                    aTaskLabels[i].setBackground(Color.red);
                                    aTaskLabels[j].setBackground(Color.red);
                                }
                          }
                          //aResponsible[i].setBackground(Color.lightGray);
                          //aResponsible[j].setBackground(Color.lightGray);
                                //aTaskLabels[i].setFont(new Font("Dialog", Font.BOLD, FontSize));

                          }
                        }
      }
    }
    public void mouseExited(java.awt.event.MouseEvent event)
        {
                Object object = event.getSource();
                for(int i=0; i<numTasks; i++)
  {
      if ((object == aTaskLabels[i]) || (object == aResponsible[i]))
                    {
                        aTaskLabels[i].setBackground(Color.white);
                        aResponsible[i].setBackground(Color.white);
                    }
                    else
                    {
      for (int j=0; j<numTasks; j++)
                        if (object == aTasks[i][j])
                        {
                            if (i == j)
                                aTasks[i][j].setBackground(new Color(255));
                            else
                                aTasks[i][j].setBackground(Color.white);
                            aTaskLabels[i].setBackground(Color.white);
                      aTaskLabels[j].setBackground(Color.white);
                      //aResponsible[i].setBackground(Color.white);
                      //aResponsible[j].setBackground(Color.white);
                      //aTaskLabels[i].setFont(new Font("Dialog", Font.PLAIN, FontSize));
                        }
                    }
                }
    }
}
void MatrixClicked (Object object)
{
            for (int i=0; i<numTasks; i++)
                {
                    if (object == aTaskLabels[i])
                    {
                        setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
                        Task1PopUp Pop = new Task1PopUp((aTaskRecords[i].TaskNameLong).trim(), Project);
```

```
                                Pop.setParents(this, aTaskRecords[i]);
                                Pop.WriteLabels((Vector)aDiagonal[i][0].clone(), (aTaskRecords[i].TaskNameShort).trim(),
(Vector)aDiagonal[i][1].clone());
                                Pop.show();
                                setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                            }
                            else
                            {
                                for (int j=0; j<numTasks; j++)
                                if (object == aTasks[i][j])
                                {
                                    if (i != j)
                                        {
                                            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        Task2PopUp Pop = new Task2PopUp((aTaskRecords[i].TaskNameShort).trim() + " -- " + (aTaskRecords[j].TaskNameShort).trim(), Project);
                                                Pop.setParents(this, aTaskRecords[i], aTaskRecords[j]);
        Pop.WriteLabels((aTaskRecords[i].TaskNameShort).trim(), (Vector)aDataElements[i][j].clone(), (aTaskRecords[j].TaskNameShort).trim(),
(Vector)aDiagonal[i][1].clone(), (Vector)aDiagonal[j][0].clone());
                                                Pop.show();
                                            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                                        }
                                    else
                                        {
                                            if ((aTaskRecords[i].Flag == 0) && (AllowChange))      //If no breakdown
                                                {
                                                    (new BreakDownTask(this, true, aTaskRecords[i].TaskID, aTaskRecords[i].TaskNameLong)).show();
                                                }
                                            else if (aTaskRecords[i].Flag == 1)          //must be a breakdown
                                            {
                                            try {
                                                    if (DataChanged){
                                                    setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));

                                                        String nxtPage2="dsm.asp?Lvl=" + (Lvl+1) + "&ParentID=" + aTaskRecords[i].TaskID;
                                                          (new SaveChanges(this, true, nxtPage2)).show();
                                                    setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                                                }
                                                else{
                                                                setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
                                            URL url = new URL (pFrame.getDocumentBase(), "dsm.asp?Lvl=" + (Lvl+1) + "&ParentID=" + aTaskRecords[i].TaskID);
                                                    pFrame.getAppletContext().showDocument(url);
                                                    setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
                                                }
                                            }
                                             catch (Exception e) {
                                                e.printStackTrace();
                                            }
                                        }
                                        }

                                 }
                            }
                        }
        }
        void cmdSave_Clicked ()
        {
            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
            UpdateDatabase();
            DataChanged = false;
            cmdSave.setEnabled(false);
            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        }
    void cmdEdit_Clicked ()
        {
            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
            DataCapture.FrontEnd.AbbyDialog ad = new DataCapture.FrontEnd.AbbyDialog (UserID, ParentID, Lvl, aTaskRecords, aDiagonal, numTasks,
Project);
            ad.show();
            validate();
            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
       try
       {
           URL url = new URL (pFrame.getDocumentBase(), "dsm.asp?Lvl=" + Lvl + "&ParentID=" + ParentID);
           pFrame.getAppletContext().showDocument(url);
       }
       catch (Exception e)
       {
           e.printStackTrace();
       }
        }
    void cmdHome_Clicked ()
        {
            System.out.println("data" + DataChanged);
            if (DataChanged)
                (new SaveChanges(this, true, "../user/home.asp")).show();
            else
            {
                try
                {
                    URL url = new URL (pFrame.getDocumentBase(), "../user/home.asp");
              pFrame.getAppletContext().showDocument(url);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
    }
    void cmdBack_Clicked ()
        {
                AbbyDBConnection adbc=new AbbyDBConnection(Project);
          Connection con=adbc.getConnection();
         ResultSet rs=null;
           Statement stm=null;
           try
               {
                   stm=con.createStatement();
                   String ParentCall="SELECT * FROM DSMMaster\n"+
```

```
                    " WHERE DSMMaster.TaskID = " + ParentID + "\n";
                    System.out.println(ParentCall);
                    rs=stm.executeQuery(ParentCall);
                    rs.next();
                        if (DataChanged){
                            String nxtPage="dsm.asp?Lvl=" + (Lvl-1) + "&ParentID=" + rs.getInt("ParentID");
                              (new SaveChanges(this, true, nxtPage)).show();
                    }
                    else{
                        URL url = new URL (pFrame.getDocumentBase(), "dsm.asp?Lvl=" + (Lvl-1) + "&ParentID=" + rs.getInt("ParentID"));
                        pFrame.getAppletContext().showDocument(url);
                        rs.close();
                    }
                }
                catch (Exception e)
                {
                    }

    }
    void optimize() {
        Vector TaskIds=new Vector();
        Vector outsideInputs=new Vector();
        Vector validInputs=new Vector();
        Vector newOrder=new Vector();
        Vector OutIds=new Vector();
        for (int i=0; i<numTasks; i++) {
            TaskIds.addElement(new Integer(aTaskRecords[i].TaskID));
        }
        for (int i=0; i<numTasks; i++) {
            Vector temp=aDiagonal[i][1];
            for (int j=0; j<temp.size(); j++) {
                OutIds.addElement(new Integer(((DERecord)(temp.elementAt(j))).ID));
            }
        }
        System.out.println("Num: "+numTasks);
        for (int i=0; i<numTasks; i++) {
            System.out.println(((Integer)(TaskIds.elementAt(i))).intValue());
        }
        for (int i=0; i<numTasks; i++) {
            Vector temp=aDiagonal[i][0];
            for (int j=0; j<temp.size(); j++) {
                Integer deid=new Integer(((DERecord)(temp.elementAt(j))).ID);

                if (!(OutIds.contains(deid))) {
                    outsideInputs.addElement(deid);
                }
            }
        }
        for (int i=0; i<numTasks; i++) {
            Vector temp=aDiagonal[i][0];
            boolean test=true;
            for (int j=0; j<temp.size(); j++) {
                DERecord der=(DERecord)(temp.elementAt(j));
                if (!(outsideInputs.contains(new Integer(der.ID)))) {
                    test=false;
                }
            }
            if (test) {
                newOrder.addElement((Integer)(TaskIds.elementAt(i)));
                Vector temp2=aDiagonal[i][1];
                for (int k=0; k<temp2.size(); k++) {
                    System.out.println(((DERecord)(temp2.elementAt(k))).ID);
                    validInputs.addElement(new Integer(((DERecord)(temp2.elementAt(k))).ID));
                }

            }
        }
        System.out.println("Entering while.");
        for (int i=0; i<newOrder.size(); i++) {
}
        while (newOrder.size()<numTasks) {
            for (int i=0; i<numTasks; i++) {
                Vector temp=aDiagonal[i][0];
                boolean test=true;
                for (int j=0; j<temp.size(); j++) {
                    DERecord der=(DERecord)(temp.elementAt(j));
                    boolean test1=true;
                    //System.out.println(der.OutputFrom + " - " + ((Integer)(TaskIds.elementAt(i))).intValue());
                    if (!(outsideInputs.contains(new Integer(der.ID)))) {
                        test1=false;
                    }
                    boolean test2=true;
                    if (!(validInputs.contains(new Integer(der.ID)))) {
                        test2=false;
                    }
                    if (!(test1 || test2)) {
                        test=false;
                    }
                }
                if (test) {
                    if (!(newOrder.contains((Integer)(TaskIds.elementAt(i))))) {
                        newOrder.addElement((Integer)(TaskIds.elementAt(i)));
                        Vector temp2=aDiagonal[i][1];
                        for (int k=0; k<temp2.size(); k++) {
                            System.out.println(((DERecord)(temp2.elementAt(k))).ID);
                            validInputs.addElement(new Integer(((DERecord)(temp2.elementAt(k))).ID));
                        }

                    }
                }
            }
        }

        for (int i=0; i<numTasks; i++) {
            System.out.println(((Integer)(newOrder.elementAt(i))).intValue());
        }
    }
}
```

D.3 Java Applications

Object DataCapture.MailServer.AbbyMailServer.class

Description: This object checks the issues database, perpares a list of users to contact and customizes a message for each person. Messages are then sent to the users. The module ensures that multiple messages are not sent to users.

```
package DataCapture.MailServer;
import java.net.*;
import java.util.*;
import java.io.*;
import DataCapture.FrontEnd.*;
import java.sql.*;
public class AbbyMailServer {
    String srvName;
    String project = "abby";
    Socket sock;
    PrintStream ps;

    public AbbyMailServer(String name) {
        srvName=name;
    }

    public void connect() {
        try {
            sock=new Socket(srvName,25);
        }
        catch (Exception e) {
            System.out.println("Connection to "+srvName+" failed.");
        }
        ps=null;
        try{
            ps=new PrintStream(sock.getOutputStream());
        }
        catch (Exception e) {
            System.out.println("Unable to open socket for output.");
        }
    }

    public void sendMessage(String rec, String mess) {
        connect();
        System.out.println(rec+"\n"+mess);
        String message="HELO \n"+"MAIL FROM: Abby\n"+"RCPT TO: "+rec+"\n"+"DATA\n"+
            "Subject: Abby - data capture request\n" + mess + "\n.\nQUIT\n";
        ps.print(message);
    }

    public void sendMessages() {
        try {
        Connection con=(new AbbyDBConnection(project)).getConnection();
        PreparedStatement ps=con.prepareStatement("SELECT UserID FROM Issues WHERE Type=? or Type<?");
        ps.setInt(1,9);
        ps.setInt(2,5);
        ResultSet rs=ps.executeQuery();
        Vector users=new Vector();
        while (rs.next()) {
            Integer nw=new Integer(rs.getInt(1));
            if (!(users.contains(nw))){
              users.addElement(nw);
            }
        }
        rs.close();
        ps.close();
        for(int x=0;x<users.size();x++) {
            PreparedStatement ps2=con.prepareStatement("SELECT Email,LoginID, Password FROM UserProfile WHERE UserId=?");
            ps2.setInt(1,((Integer)(users.elementAt(x))).intValue());
            ResultSet rs2=ps2.executeQuery();
            rs2.next();
            sendMessage(rs2.getString(1),"\nYour participation has been requested in creating/updating a process model for Test Project.\n\n"+
                "Please access page http://design.mit.edu/Abby/default.asp \n\n\n"+
                "Your login information is as follows: \n\n"+
                " User ID:   " + rs2.getString(2) + "\n"+
                "Password:   " + rs2.getString(3) + "\n\n\n"+
                "Thank you for your contribution.\n\n--Abby");
            rs2.close();
            ps2.close();
        }
        con.close();
        }
        catch (Exception e) {
            System.out.println(e.toString());
        }
    }

}
```

Application DataCapture.BackEnd.Checker.class

Description: Performs the necessary data setup and maintenance activities before and after checking the model for inconsistencies (through DataElementCheck.class). This entails preparing issues and updating the issues database after each validation batch run.

```
//  Checker uses the data from DataElementCheck to write
//  Issues to the Issues table in the DB.

package DataCapture.BackEnd;

import DataCapture.FrontEnd.*;
import DataCapture.MailServer.*;
import java.util.*;
import java.sql.*;

public class Checker {

    DataCapture.FrontEnd.AbbyDBConnection adbc;
    Vector issues;
    Connection con;

    public Checker() {
        adbc=new AbbyDBConnection("abby");
        con=adbc.getConnection();
        issues=new Vector();
    }

    public void checkDE() {
        //Checks for hanging data elements and
        //assigns internal/external values to all elements

        try {
        Vector iss4Pars=new Vector();
        PreparedStatement pdst=con.prepareStatement("DELETE FROM ISSUES WHERE Type<? AND Status<2");
        pdst.setInt(1,4);
        pdst.executeUpdate();
        pdst.close();
        PreparedStatement psst=con.prepareStatement("SELECT DataElementID,TaskID,Type FROM Issues");
        ResultSet psrs=psst.executeQuery();
        Vector issueIDs=new Vector();
        while (psrs.next()) {
            ResultType2 rt2=new ResultType2();
            rt2.dID=psrs.getInt(1);
            rt2.tID=psrs.getInt(2);
            rt2.input=psrs.getInt(3);
            System.out.println("In: "+rt2.dID+", "+rt2.tID+", "+rt2.input);
            issueIDs.addElement(rt2);
        }
        psrs.close();
        psst.close();

        issues=(new DataElementCheck(adbc)).run();
        System.out.println(issues.size());
        //con=new AbbyDBConnection().getConnection();
        for (int i=0;i<issues.size(); i++) {
            Issue iss=(Issue)(issues.elementAt(i));
            System.out.println(iss.Type+", "+iss.TaskID+", "+iss.DataElementID);
            boolean contain=false;
            for (int j=0; j<issueIDs.size(); j++) {
                ResultType2 issID=(ResultType2)(issueIDs.elementAt(j));
                if ((issID.dID==iss.DataElementID) && (issID.tID==iss.TaskID) && (issID.input==iss.Type)) {
                    contain=true;
                }
            }
            if (!contain) {

            PreparedStatement pst1=con.prepareStatement("SELECT TaskName_Short,UserID FROM DSMMaster WHERE "+
                                        "TaskID = ?");

            pst1.setInt(1,iss.TaskID);
            ResultSet rs=pst1.executeQuery();
            rs.next();
            int rsi=rs.getInt(2);
            String tn=rs.getString(1);
            pst1.close();
            rs.close();
            PreparedStatement pst2=con.prepareStatement("SELECT Name FROM DataDictionary WHERE "+
                                        "DataElementID = ?");
            pst2.setInt(1,iss.DataElementID);
            ResultSet rs2=pst2.executeQuery();
            rs2.next();
            String dn=rs2.getString(1);
            rs2.close();
            pst2.close();
            PreparedStatement upst=con.prepareStatement("SELECT Name,Email FROM UserProfile WHERE UserID = ?");
            upst.setInt(1,rsi);
            ResultSet rs3=upst.executeQuery();
            rs3.next();
            String em=rs3.getString(2);
            String un=rs3.getString(1);
            rs3.close();
            upst.close();
            //System.out.println(iss.Type+", "+iss.TaskID+", "+iss.DataElementID);
            if (iss.Type<4) {

            if (iss.Type ==3) {
                PreparedStatement pstN=con.prepareStatement("SELECT UserID FROM DSMMaster WHERE "+
```

```
                                                "ParentID = ?");
                pstN.setInt(1,iss.TaskID);
                ResultSet rsN=pstN.executeQuery();
                rsN.next();
                rsi=rsN.getInt(1);
                rsN.close();
                pstN.close();
            }
            PreparedStatement ipst=con.prepareStatement("INSERT INTO Issues VALUES (?,?,?,?,?,?,?,?,?)");
            ipst.setInt(1,rsi);
            ipst.setInt(2,0);
            ipst.setInt(3,iss.TaskID);
            ipst.setInt(4,iss.DataElementID);
            ipst.setInt(5,iss.Type);
            ipst.setInt(6,0);
            ipst.setInt(7,0);
            ipst.setInt(8,1);
            ipst.setInt(9,-1);
            ipst.executeUpdate();
            ipst.close();
            }
            if (iss.Type==4) {
                PreparedStatement psm=con.prepareStatement("SELECT ParentID FROM DSMMaster WHERE TaskID=?");
                psm.setInt(1,iss.TaskID);
                ResultSet psmrs=psm.executeQuery();
                psmrs.next();
                int pid=psmrs.getInt(1);
                ResultType2 rs22=new ResultType2();
                rs22.dID=iss.DataElementID;
                rs22.tID=pid;
                rs22.input=iss.DEType;
                psmrs.close();
                psm.close();
                iss4Pars.addElement(rs22);
            }

            /*Message m=new Message(iss);
            AbbyMailServer ams=new AbbyMailServer("mit.edu");
            ams.sendMessage(m.rec, m.mess); */
            }
        }
        boolean no=true;
        for (int k=0; k<iss4Pars.size(); k++) {
            ResultType2 pID=(ResultType2)(iss4Pars.elementAt(k));
            PreparedStatement psm2=con.prepareStatement("INSERT INTO IO VALUES (?,?,?,?)");
            psm2.setInt(1,pID.tID);
            psm2.setInt(2,pID.dID);
            psm2.setInt(3,pID.input);
            psm2.setInt(4,9);
            psm2.executeUpdate();
            psm2.close();
            no=false;
        }
        //delete resolved issues
        for (int j=0; j<issueIDs.size(); j++) {
            ResultType2 issID=(ResultType2)(issueIDs.elementAt(j));
            if ((issID.input==1) || (issID.input==2) ||(issID.input==5) || (issID.input==6)){
                boolean contain=false;
                for (int i=0; i<issues.size();i++){
                    Issue iss=(Issue)(issues.elementAt(i));
                    boolean dangling=true;
                    if ((iss.Type==3) || (iss.Type==4)){
                        dangling=false;
                    }
                    if ((issID.dID==iss.DataElementID) && (issID.tID==iss.TaskID) && (dangling)){
                        contain=true;
                    }
                }
                if (!contain){
                    PreparedStatement pdstD=con.prepareStatement("DELETE FROM ISSUES WHERE TaskID=? AND Type=? AND DataElementID=?");
                    pdstD.setInt(1,issID.tID);
                    pdstD.setInt(2,issID.input);
                    pdstD.setInt(3,issID.dID);
                    pdstD.executeUpdate();
                    pdstD.close();

                }

            }
        }
        //if (!no) {
        //    this.checkDE();
        //}

        }
        catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

Application DataCapture.BackEnd.DataElementCheck.class

Description: Scans the entire model and checks for data inconsistencies such as input and output deliverable disconnects and inter-level disparities. Marked issues are returned to "Checher.class" for storage in the repository.

```
// This file contains the database search engine that looks
// problematic data.
package DataCapture.BackEnd;

import DataCapture.FrontEnd.*;
import java.sql.*;
import java.util.*;
public class DataElementCheck {
    Connection con;
    public Vector issues=new Vector();
    public DataElementCheck(AbbyDBConnection aDBC) {
        con=aDBC.getConnection();
    }
    public Vector run() {
        //System.out.println("Enter run");
        try {
            Vector in=new Vector();
            Vector out=new Vector();
            PreparedStatement istmt=con.prepareStatement("SELECT DataElementID,TaskID FROM IO WHERE Type=?");
            PreparedStatement pst=con.prepareStatement("SELECT ParentID, Lvl FROM DSMMaster WHERE TaskID = ?");
PreparedStatement pst2=con.prepareStatement("UPDATE IO SET ExtInt=? WHERE "+ "DataElementID = ? AND TaskID = ?");
            //System.out.println("Enter irs");
            istmt.setInt(1,0);
            ResultSet irs=istmt.executeQuery();
            Vector irt=new Vector();
            while (irs.next()) {
                ResultType1 r1=new ResultType1();
                r1.DEID=irs.getInt(1);
                r1.TID=irs.getInt(2);
                irt.addElement(r1);
            }
            irs.close();
            istmt.setInt(1,1);
            ResultSet ors=istmt.executeQuery();
            Vector ort=new Vector();
            while (ors.next()) {
                ResultType1 r1=new ResultType1();
                r1.DEID=ors.getInt(1);
                r1.TID=ors.getInt(2);
                ort.addElement(r1);
            }
            //Close unneeded connections
            ors.close();
            istmt.close();
            boolean found;
            boolean internalDE=false;
            boolean update_parent=false;
            int idID=0;
            //System.out.println("Entering Loop");
            int y=0;
            int z=0;
            while (y<irt.size()) {
                System.out.println("Entered Loop");
                idID=((ResultType1)(irt.elementAt(y))).DEID;
                found=false;
                internalDE=false;
                update_parent=false;
                z=0;
                while ((z<ort.size()) && (!internalDE)) {
                    int odID=((ResultType1)(ort.elementAt(z))).DEID;
                    if (idID==odID) {
                        System.out.println("In Here.");
                        found=true;
                        pst.setInt(1,((ResultType1)(irt.elementAt(y))).TID);
                        ResultSet rs1=pst.executeQuery();
                        rs1.next();
                        System.out.println("Here1");
                        int ipID=rs1.getInt(1);
                        int iLvl= rs1.getInt(2);
                        System.out.println("Here2");
                        pst.setInt(1,((ResultType1)(ort.elementAt(z))).TID);
                        ResultSet rs2=pst.executeQuery();
                        rs2.next();
                        System.out.println("Here3:"+((ResultType1)(ort.elementAt(z))).TID);
                        int opID=rs2.getInt(1);
                        int oLvl= rs2.getInt(2);
                        System.out.println("Here4");
                        rs1.close();
                        rs2.close();
                        if (ipID==opID) {
                            ResultType1 r1=new ResultType1();
                            r1.DEID=idID;
                            r1.TID=((ResultType1)(irt.elementAt(y))).TID;
                            in.addElement(r1);
                            internalDE=true;
                        }
                        if ((oLvl > iLvl) | (iLvl == oLvl)) {
                            update_parent=true;
                        }
                    }
                    z++;
                }
                //found connection but it is not an internal link
                if ((found) && (!internalDE)) {
                   if (update_parent) {
```

```
                    ResultType1 r1=new ResultType1();
                    r1.DEID=idID;
                    r1.TID=((ResultType1)(irt.elementAt(y))).TID;
                    out.addElement(r1);
                   }

                }
                if (!found) {
                    issues.addElement(new Issue(1,idID,0,((ResultType1)(irt.elementAt(y))).TID));
                }
                y++;
            }
            System.out.println("Exit Loop");
            y=0;
            z=0;
            while (y<ort.size()) {
                int odID=((ResultType1)(ort.elementAt(y))).DEID;
                found=false;
                internalDE=false;
                update_parent=false;
                z=0;
                while ((z<irt.size()) && (!internalDE)) {
                    idID=((ResultType1)(irt.elementAt(z))).DEID;
                    if (odID==idID) {
                        found=true;
                        pst.setInt(1,((ResultType1)(ort.elementAt(y))).TID);
                        ResultSet rs1=pst.executeQuery();
                        rs1.next();
                        int opID=rs1.getInt(1);
                        int oLvl= rs1.getInt(2);
                        pst.setInt(1,((ResultType1)(irt.elementAt(z))).TID);
                        ResultSet rs2=pst.executeQuery();
                        rs2.next();
                        int ipID=rs2.getInt(1);
                        int iLvl= rs2.getInt(2);
                        rs1.close();
                        rs2.close();
                        if (ipID==opID) {
                            ResultType1 r1=new ResultType1();
                            r1.DEID=idID;
                            r1.TID=((ResultType1)(ort.elementAt(y))).TID;
                            in.addElement(r1);
                            internalDE=true;
                        }
                        if ((iLvl > oLvl) | (oLvl==iLvl)) {
                            update_parent=true;
                        }
                    }
                    z++;
                }
                //found connection but it is not an internal link
                if ((found) && (!internalDE)) {
                    ResultType1 r1=new ResultType1();
                    r1.DEID=idID;
                    r1.TID=((ResultType1)(ort.elementAt(y))).TID;
                    out.addElement(r1);
                }
                //no connection found at all => dangling dependency
                if (!found) {
                    issues.addElement(new Issue(2,odID,1,((ResultType1)(ort.elementAt(y))).TID));
                    //pst2.setInt(1,3);
                    //pst2.setInt(2,odID);
                    //pst2.setInt(3,((ResultType1)(ort.elementAt(y))).TID);
                }
                y++;
            }
//Prepare IO table for Internal/External flag update
//Clear all flags
            PreparedStatement clrIO=con.prepareStatement("UPDATE IO SET ExtInt=9");
            clrIO.executeUpdate();
//Set all external flags
            for(int x=0;x<in.size();x++) {
                ResultType1 r1=(ResultType1)(in.elementAt(x));
                pst2.setInt(1,0);
                pst2.setInt(2,r1.DEID);
                pst2.setInt(3,r1.TID);
                pst2.executeUpdate();
            }
//Set all internal flags
            for(int x=0;x<out.size();x++) {
                ResultType1 r1=(ResultType1)(out.elementAt(x));
                pst2.setInt(1,1);
                pst2.setInt(2,r1.DEID);
                pst2.setInt(3,r1.TID);
                pst2.executeUpdate();
            }
        pst.close();
        pst2.close();
        }
        catch (SQLException e) {
            System.out.println(e.toString());
        }
          //Parent-Child Check
          //Checking for the Child to Parent Consistency
        Vector exts=new Vector();
        try {
        PreparedStatement pstmt=con.prepareStatement("SELECT DataElementID,TaskID,Type FROM IO WHERE "+
                                                     "ExtInt=1");
        ResultSet rslt=pstmt.executeQuery();
        while (rslt.next()) {
            ResultType2 rt2=new ResultType2();
            rt2.dID=rslt.getInt(1);
            rt2.tID=rslt.getInt(2);
            rt2.input=rslt.getInt(3);
            exts.addElement(rt2);
        }
        rslt.close();
        pstmt.close();
```

```
        for (int x=0;x<exts.size();x++) {
            //System.out.println("Child check");
            PreparedStatement pstm1=con.prepareStatement("SELECT ParentID FROM DSMMaster WHERE "+
                                                "TaskID=?");
            pstm1.setInt(1,((ResultType2)(exts.elementAt(x))).tID);
            ResultSet rst=pstm1.executeQuery();
            rst.next();
            int pID=rst.getInt(1);
            PreparedStatement pstm2=con.prepareStatement("SELECT DataElementID FROM IO WHERE "+
                                                "TaskID=? AND Type=?");
            pstm2.setInt(1,pID);
            pstm2.setInt(2,((ResultType2)(exts.elementAt(x))).input);
            ResultSet rst2=pstm2.executeQuery();
            boolean child=false;
            while (rst2.next()) {
                if (rst2.getInt(1)==((ResultType2)(exts.elementAt(x))).dID) {
                    child=true;
                }
            }
            if (pID==0) {
                child=true;
            }
            if (!child) {
                Issue is=new Issue(4,((ResultType2)(exts.elementAt(x))).dID,((ResultType2)(exts.elementAt(x))).input,
                                   ((ResultType2)(exts.elementAt(x))).tID);
                issues.addElement(is);
            }
            rst.close();
            rst2.close();
            pstm1.close();
            pstm2.close();
        }

        //Checking for the parent to child cosistency
        Vector allDeps=new Vector();
        PreparedStatement depnds=con.prepareStatement("SELECT DataElementID,TaskID,Type FROM IO WHERE "+ "ExtInt < 3");
        ResultSet rslt2=depnds.executeQuery();
        while (rslt2.next()) {
            ResultType2 rt3=new ResultType2();
            rt3.dID=rslt2.getInt(1);
            rt3.tID=rslt2.getInt(2);
            rt3.input=rslt2.getInt(3);
            allDeps.addElement(rt3);
        }
        rslt2.close();
        depnds.close();
        for (int x=0;x<allDeps.size();x++) {
            PreparedStatement pstm5=con.prepareStatement("SELECT Flag FROM DSMMaster WHERE "+
                                                "TaskID=?");
            pstm5.setInt(1,((ResultType2)(allDeps.elementAt(x))).tID);
            ResultSet rst5=pstm5.executeQuery();
            rst5.next();
            int flg=rst5.getInt(1);
            rst5.close();
            // System.out.println("Parent check");
            if (flg==1) {
                //System.out.println("Parent check2");
                boolean parent=false;
                PreparedStatement pstm3=con.prepareStatement("SELECT TaskID FROM DSMMaster WHERE ParentID=?");
                pstm3.setInt(1,((ResultType2)(allDeps.elementAt(x))).tID);
                ResultSet crs=pstm3.executeQuery();
                Vector children=new Vector();
                while (crs.next()) {
                    children.addElement(new Integer(crs.getInt(1)));
                }
                crs.close();
                pstm3.close();
                for(int i=0;i<children.size();i++) {
                    PreparedStatement pstm4=con.prepareStatement("SELECT DataElementID FROM IO WHERE TaskID=? AND Type=?");
                    System.out.println(((Integer)(children.elementAt(i))).intValue());
                    pstm4.setInt(1,((Integer)(children.elementAt(i))).intValue());
                    pstm4.setInt(2,(((ResultType2)(allDeps.elementAt(x))).input));
                    ResultSet crs2=pstm4.executeQuery();
                    while (crs2.next()) {
                        if (crs2.getInt(1)==((ResultType2)(allDeps.elementAt(x))).dID) {
                            parent=true;
                        }
                    }
                    crs2.close();
                    pstm4.close();

                }
                if (!parent) {
            Issue is=new
Issue(3,((ResultType2)(allDeps.elementAt(x))).dID,((ResultType2)(allDeps.elementAt(x))).input,(ResultType2)(allDeps.elementAt(x))).tID);
                    issues.addElement(is);
                }
            }
        }
        //con.close();
        }
        catch (SQLException e) {
            System.out.println(e.toString());
        }
        System.out.println("Exit run");
        return issues;
    }
}
```

4378-82