MANUFACTURING INFRASTRUCTURE AND DESIGN AUTOMATION SYSTEM (MIDAS) WITH XML REPRESENTATION

By

Yonggang Qin

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2002

ABSTRACT

MANUFACTURING INFRASTRUCTURE AND DESIGN AUTOMATION SYSTEM (MIDAS) WITH XML REPRESENTATION

By

Yonggang Qin

Digital Manufacturing and Virtual Factory is revolutionizing product manufacturing and process design. MIDAS (Manufacturing Infrastructure and Design Automation System), a design and manufacturing framework, has the same goal as Digital Manufacturing systems to integrate process design, manufacturing and assembly to solve problems caused by isolation of manufacturing from designing in the traditional manufacturing industry. To meet the requirements of collaboration and distribution among different enterprises the MIDAS system architecture has been advanced significantly. In order to provide to the MIDAS system with information that is easy to manage and well-structured, XML files have been employed to store the process information and to exchange information among collaborators. A three level database design technology has been used to graphically design XML Schema through mapping with Unified Modeling Language (UML). XML technologies, such as DOM and Schema, have been utilized to develop the new system and to validate XML files. A new parallel multi-threading algorithm has replaced the previous linear scheduling algorithm in order to manage the process elaboration efficiently.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 Introduction

Product manufacturing and process design has been revolutionized by the advancement of computer technology. In the manufacturing world, manufacturing and design were considered separately. Most of product designers could not solve all of problems in the design phase because of the limited time available in order to stay competitive. They had to leave some problems to industrial engineers. Even though the problems were solved during production by the industrial engineers, the modifications made in production could not fix the design because they are never incorporated into the CAD models (Freedman, 1999). In addition, most of the issues, which were easily fixed in the design phase, are more complex from an industrial engineer's perspective (Donald, 1999). These problems caused by isolating manufacturing from design can be overcome by utilizing Digital Manufacturing and Virtual Factory, which is a 3D computer environment that can achieve seamless integration of design, manufacturing and assembly (Donald, 1999; Freedman, 1999; Zha, 2002).

Digital Manufacturing and Virtual factory has drawn more and more attentions due to its capability of integrating manufacturing process with design process (Geller et al., 1995; Donald, 998; Brugali et al., 1998). In addition, Digital Manufacturing and Virtual Factory has great potentials in that it can provide a safe environment where designers can experiment with different manufacturing techniques. This virtual environment can circumvent some of the problems of a real workshop, such as the inability to go backwards in time. A Virtual Factory also allows designers to easily tighten and relax the

applied constraints. By effectively integrating manufacturing process into designing process, a Virtual Factory can provide a lot more capabilities rather than being just a simulation of a real workshop.

Two of the ealier virtual manufacturing systems were developed by Bayliss and coworker (1994) and Gaines et al. (1995), respectively. Both of the systems support the integration of computer-based system design, production engineering and production sub-systems into a collaborative framework. They provide the capabilities of recording and tracing decisions made at each stage of the production process.

One of the digital manufacturing integration method has been outlined by Freedman (1999), which is referred as an integrated umbrella application. It has an expanded scope beyond the typical Computer Aided Process Planning tools. This application integrated all of the accumulated product data, processes, resources, and knowledge required to design, engineer, test, manufacture, and maintain the entire car, ship or aircraft.

In order to create this application, it has been suggested that following information be collected. (Freedman, 1999)

- Process knowledge or library.
- The product library, which includes the physical parts of the overall assembly, subassemblies, and components that are designed and purchased.
- The definition of the resources required to manufacture and assemble the products.

- The association of the products to the processes and the association of resources to the processes. These associations generate connectivity between the process steps, the CAD models, the simulation models, and the resources required.

Applications of Digital Manufacturing technologies have recently been developed as Computer Aided Process Planning, Computer Aided Production Engineering, and Manufacturing Database which contains product data, process information, manufacturing resources (PPR). These technologies make it possible to generate automation programs, workers' instructions, and the feedback of manufacturing performance data (Brown, 2000). In order to perform designing of manufacturing and assembly processes, Brown (2000) identified a series of simulation and monitoring tools to validate the process plan, resource allocation and the human/machine interfaces of the manual work stations as well as tools to reconfigure the design.

Product data and process information have been shown to be the fundamental resource in Digital Manufacturing and Virtual Factory. Once the process information are collected and presented following appropriate modeling technique, the system of Digital Manufacturing and Virtual Factory will execute the design process based on the process information presented. During the execution, the system continuously monitors the design process and captures the process data, which are used to decide whether or not to reconfigure the design process in order to obtain satisfactory product. Simulation can be employed to estimate and validate the current configured process.

The system that is discussed in this work is a design and manufacturing framework, called MIDAS (Manufacturing Infrastructure and Design Automation System), having the same goal as Digital Manufacturing and Virtual Factory. The original MIDAS (Keys, 1995) provides features of running tools in a distributed environment and multi-threading technologies to execute process. However, it does not have the collaborative feature, its process information is not distributed, and its multi-thread scheduling algorithm has obvious redundancy. Therefore, effort has been directed to advance the original system to accommodate the need of more recent process information. The current version of MIDAS still uses a grammar based process model, integrates design and manufacturing processes and coordinates activities through the Internet.

Process models are helpful to understand processes, realize our ideas, and develop process management systems. (Kueng, 1997) The commonly used process models in process management systems are IDEF series (Plaia and Carrie, 1995; Kusiak, Larson, and Wang, 1994), Role Activity Diagrams (Murdoch and McDermid, 2000; Ould, 1995), and Petri net (Rudas and Horvath, 1997). These models have a variety of strengths and weaknesses. Some of their weaknesses are the lack of alternative tasks, abstraction and process iteration or rollback. The process grammar based process model that is adopted for MIDAS in this work has incorporated abstraction, alternatives, and process execution, iteration and rollback. (Baldwin and Chung, 1995) This process model is based on process flow graphs and a design process grammar.

Process flow graphs (Figure 1.1), which consist of flow objects and directed lines between flow objects, describe the information flows of processes. The process flow graphs discussed in this document are acyclic directed graphs. Three flow objects, which represent logical tasks, atomic tasks and data specifications, were proposed to represent a flow graph initially. After Keyes (1996) added another two flow objects, which represent selectors and database specifications, in the representation of flow graphs, clients can create designs in a top-down fashion using five graphic elements and directed lines to represent flow graphs.

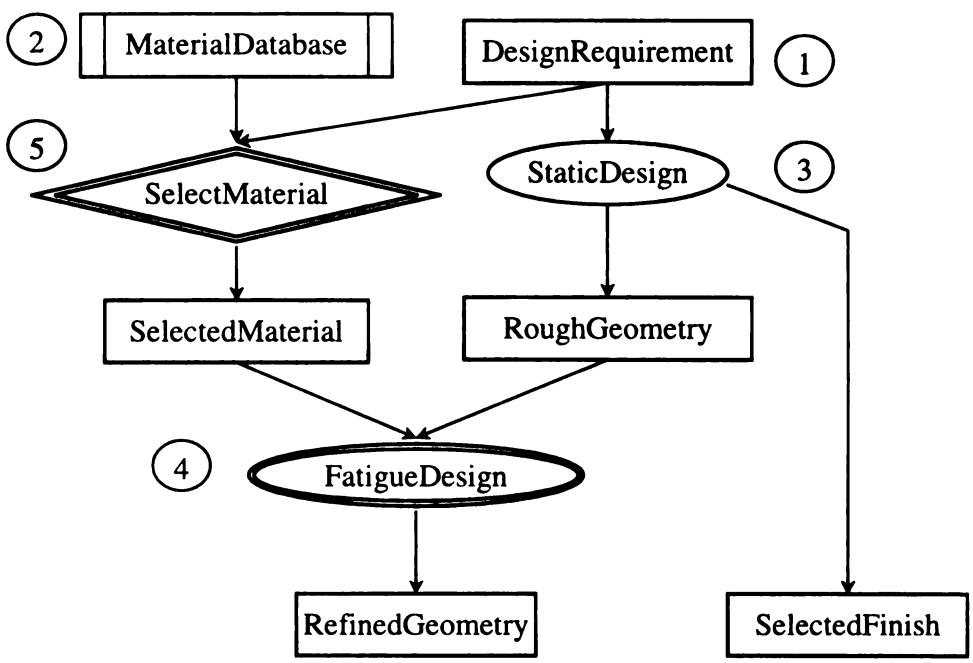

Figure 1.1 A process flow graph

These flow objects can be divided into two categories: task flow objects and specification flow objects. The flow objects representing atomic task, logical task and selector belong

to the categories of task flow objects; the flow objects representing data and database belong to specification flow objects.

- An atomic task, which invokes and monitors an external software tool when requested by a client, cannot be further decomposed because it represents an elemental event: execution of an application program.
- A logical task represents an abstract task, a higher-level task. A logical task could potentially be completed via one of many productions and performed by another process flow graph. Obviously, a logical task has at least one production.
- A selector represents a query of databases.
- Data either are specified by process designers or are created during the execution of processes. Therefore, data specification can be further categorized into input data specification and output data specification.
- Databases provide information for selectors; therefore, a selector has to have at least one database specification as an input.

A set of lines indicates the specifications used and produced by each task. Each line connects one task flow object to one specification flow object, or vice versa. Obviously, each specification must have at most one incoming edge. Data specification flow objects with no incoming lines in a process design, usually specified by process designers, are inputs of the entire process flow. Data specification flow objects with no outgoing lines are outputs of the entire process flow. In addition, designers can point intermediate data specification flow objects as outputs of the entire process flow.

An example of a process flow graph is given in Figure 1.1. Oval nodes (marked by 3) represent logical task flow objects. Double concentric oval nodes (marked by 4) represent atomic task flow objects. Double concentric diamond nodes (marked by 5) represent selector flow objects. Rectangular nodes (marked by 1) represent data specification flow objects, and rectangular nodes (marked by 2) with margins represent database specification flow objects. The directed lines represent the edges described in the previous paragraph.

A design process grammar is a concise and flexible mechanism for representing the set of methodologies that are supported by a methodology management system. The design process grammars proposed by Reid A. Baldwin and Moon Jung Chung (1995) is a means for transforming high-level process flow graphs into progressively more detailed flow graphs. Among the five graphic elements presented previously, only logical tasks can be replaced by choosing one of its productions to transform into more detailed flow graphs with the help of the pre-evaluation functions.

A production in a graph grammar allows the substitution of one sub-graph by another. Generally speaking, productions reflect routine design decisions and a company's practice style.

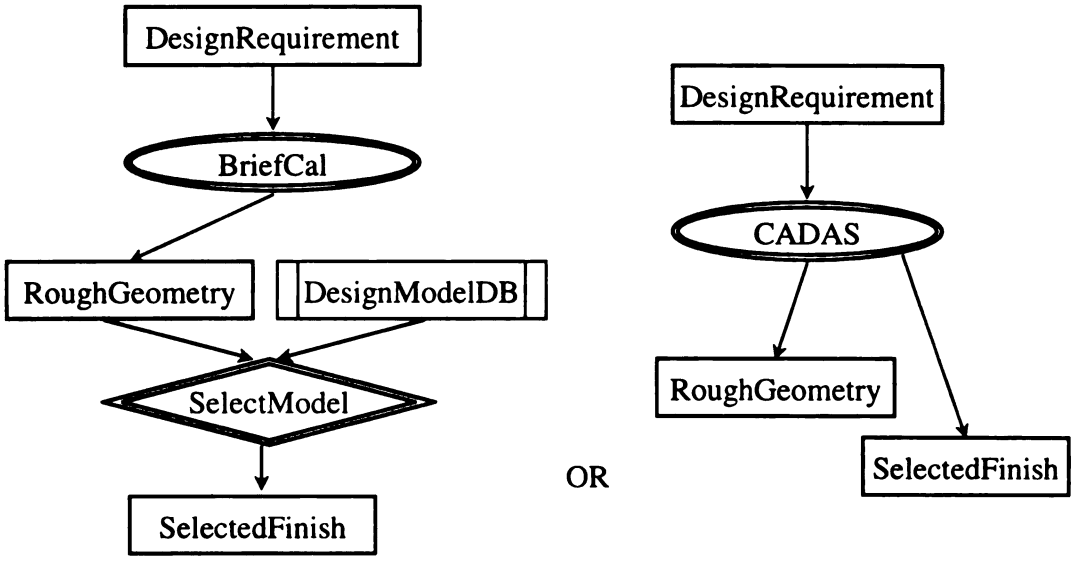

Figure 1.2 Alternatives of logical task "StaticDesign"

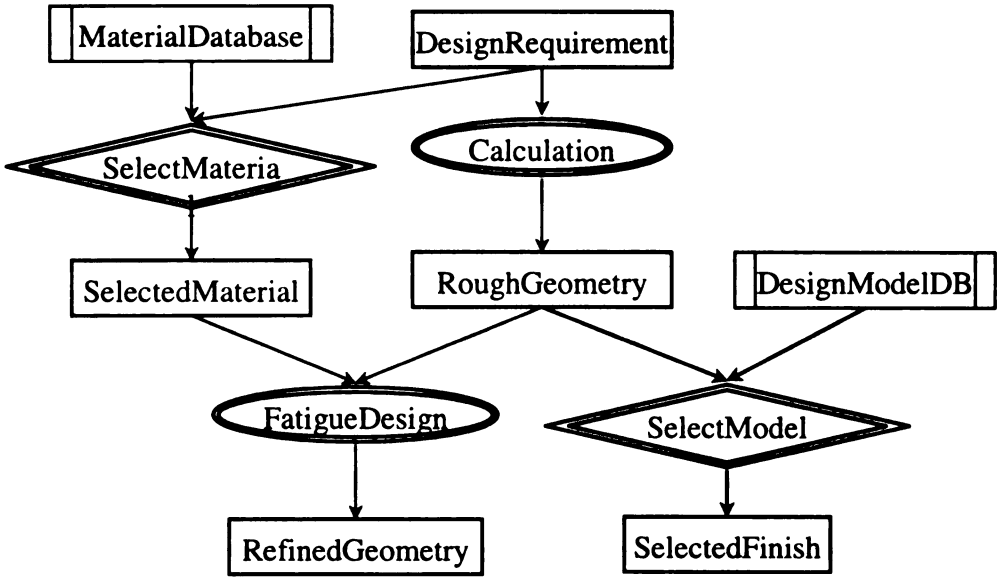

Figure 1.3 The Process flow graph after applying production "ManualDesign" for logical task "StaticDesign"

For example, a logical task flow object "StaticDesign" in Figure 1.1 and its productions are shown in Figure 1.2. By means of the design process grammar, Figure 1.1 will be transformed into Figure 1.3 using the production flow graph ManualDesign to replace the logical task StaticDesign. This capability is critical to maintaining the usability and effectiveness of the overall framework.

Obviously, the productions of a logical task have to have same type or subtype of inputs and outputs as this logical task does in order to replace this logical task. In this thesis, the names of specifications are considered same as the type of specification. Productions are allowed to have more database specifications as its inputs if the database specifications are the initial inputs.

The concept of applying productions to logical tasks is analogous to the productions in formal language grammars, under which logical tasks correspond to non-terminal elements in a grammar, and atomic tasks correspond to terminal elements.

To design a process, designers need to provide a more abstractive initial process flow with design requirements. The abstractive initial process flow is created according to the designers' domain knowledge. If the designers have more domain knowledge, they can provide an initial process flow with more details. The designers can create the initial process flow using MIDAS editor or use a system built-in process flow as an initial process flow. MIDAS keeps track of the state of the process flow, generates a detailed

process flow, and executes tasks and tools. Given a process flow, a more detail flow can be obtained by applying a production to a logical task. To select a production to apply, a pre-evaluation function of the logical task is performed to return the possible productions to apply and the corresponding probability to be successful based on the input specification of the logical task. Usually, the production with highest possibility of success is applied for the logical task. For example, Figure 1.1 shows two productions for the logical task StaticDesign: ManualDesign and AutoDesign. The pre-evaluation function of the logical task StaticDesign returns both productions, and the production ManualDesign has higher probability to succeed than the production AutoDesign. Figure 1.2 shows the expanded process flow that is obtained after applying the production ManualDesign for the logical task, as shown in Figure 1.1. This elaboration may continue until all tasks become atomic tasks. Each logical task or atomic task may have pre-condition functions or post-condition functions, or have both kinds of condition functions. The pre-condition functions of a task are performed to check whether the inputs of the task satisfy its constraints before the task is performed. Similar to pre-condition functions, the post-condition functions are performed to check whether the outputs of the task satisfy the global constraints after a task is executed. Pre-condition and post-condition functions provide checkpoints in the execution of the process. When an input or an output specification does not satisfy the constraints of tasks, rollback must occur.

Rollback is the basic mechanism for handling iteration of a process flow execution. Whenever a task is failed or a constraint is not satisfied, rollback takes place. When

rollback is performed, the current process flow is changed back to the previous process flow. The failure of a task happens differently according to its type. An atomic task is failed when it fails to execute a tool. A logical task is failed when there are not any available productions for it. A select task is failed when the database query returns null result or the query results are consumed. The previous example can be used to illustrate the rollback mechanism. When the query result of selector SelectModel is null because the specification RoughGeometry generated by the handy task BriefCal is out of the data range, rollback happens in selector SelectModel. The final result is that the expanded process flow (Figure 1.3) changes back to the previous process flow (Figure 1.1). Thereafter, the production AutoDesign can be selected to apply for the logical task StaticDesign.

The capability of generating an alternative process in mid-stream is a critical aspect of the functionality provided by a grammar-based approach. As with any design process, iteration continues until the final output specifications meet the design objectives. By doing this, the process grammar naturally captures the hierarchical design methodology and allows systematic exploration of design space. It should be noted that the iterative nature of the design process is not explicitly represented in the grammar. Instead, pre- and post-conditions are embedded within each task to provide the necessary feedback loop.

The process information was previously stored in a monolithic text file, which prevented MIDAS from advancing to a distributed system. This file is also not in a standardized

format, which makes it to difficult to integrate with other systems. In this work, XML technologies have been utilized to replace this monolithic file in order to overcome those problems.

XML stands for eXtensible Markup Language, which is a simple, very flexible text format derived from SGML (ISO 8879). Since its elevation to W3C Recommendation in February 1998, XML is quickly becoming the standard method for sending information across the Internet.

XML has the following advantageous features:

- XML is independent of hardware, software and application.
- XML is content-oriented. XML only addresses content, while presentations are handled separately by XSL (Extensible Stylesheet Language), XHTML and etc. This supports multiple views of the same content for different user groups and media.
- XML is extensible. Unlike HTML, XML users can define their own tags as needed for particular application. This feature promotes interoperability among organizations by defining shared sets of tags. Many groups have already defined XML formats for information exchange inside and outside W3C (http://www.w3.org/XML/Activity). The Petri Net Markup Language (PNML), a preliminary proposal of an XML-based interchange format for Petri nets, is proposed by a research team at Humboldt-Universität zu Berlin (Jngel, Kindler,

and Weber, 2000). Lubell and Schlenoff (1999) described how the PSL semantic concepts are mapped to the Extensible Markup Language (XML).

- XML is well structured and is able to support data validation. XML documents are associated with a Document Type Description (DTD) or an XML Schema, both of which can be used to define a structure for conforming applications. However, due to the lack of richer set of structures, types and constraints for describing the structure of XML documents, DTD is eventually replaced by XML Schema. XML Schema is a language with XML syntax that defines the structure of a specific type of XML documents. XML Schemas provide strong data typing, modularization and reuse mechanisms that are not available in XML DTDs. Based on these advantageous features, XML and related tools can be employed to facilitate data sharing at different levels. . (Seligman, Rosenthal, 2001)

Based on the grammar-based process model, the development of the new MIDAS system architecture is presented in Chapter 2 in order to meet the requirements of collaboration and distribution among different enterprises. In order to provide to the MIDAS system with information that is easy to manage, well-structured XML files have been employed to store the process information. The use of XML Schema and XML files to represent the grammar based process model is described in detail in Chapter 3. Multi-threading technology, which is explained in Chapter 4, is used to manage the process elaboration. and to exchange information among collaborators. Related XML technologies, such as DOM and Schema, are employed to develop the new system and validate XML files, how

to use XML files in execution model is represented in Chapter 5. Finally, an example is given in Chapter 6 to illustrate its application to a sample design situation.

Chapter 2 The Architecture of MIDAS

The following functional requirements of process management for collaboration have been identified from current industrial collaborative activities (Kumar, 2001; Fu et al., 2000; Gan et al., 2000; Donald, 1998; Geller et al., 1995; Brugali et al., 1998):

- Collaboration for entire process with other companies in the supply chain because exchanging product catalog is not enough
- Distribution of process database between companies with protection for proprietary information
- Execution of individual company processes (i.e. its own part of the entire process) concurrently with its collaborators
- Screen sharing during process execution to explore process alternatives or synchronize the progress of their process
- Scalability of large-scale process as new processes and tools should be added in a modular way

To satisfy the requirements described above, a new version of MIDAS framework has been created, which is a distributed and collaborative manufacturing process management system.

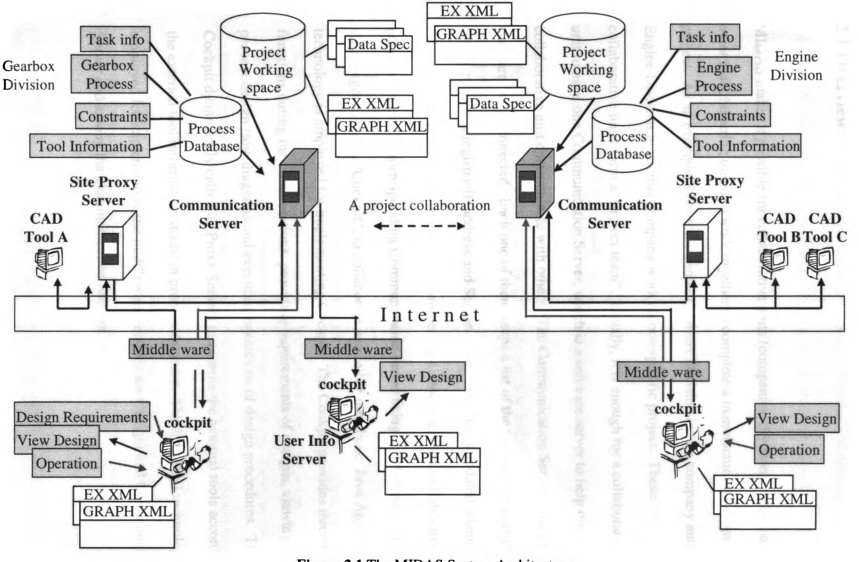

Figure 2.1 The MIDAS System Architecture

2.1 Overview

MIDAS makes it possible for collaborative units (companies, divisions, teams, or other manufacturing units) to work together to complete a manufacturing design project. As an example, Figure 2.1 shows Gearbox Division in one company and Engine Division in another company work on one specific project. These collaborative units form a "project team". Usually, it is enough for a collaborative unit to have one Communication Server, which is a software server to help one collaborative unit communicate with others. The Communication Servers in a project team are fully connected. Each one of them keeps a list of the servers' connection information (i.e. registry IP address and Server name). The list is updated whenever a Communication Server joins or leaves a project team. Members in a collaborative unit can access the corresponding Communication server using a web-based client side program, called "Cockpit", to collaborate with other members. Java Applet technology is employed for implementing Cockpit. The Cockpit provides three major functions during the collaborations: providing requirements of designs, viewing process design flow diagrams and executing operations of design procedures. The Cockpit dynamically calls Site Proxy Servers to execute the external tools according the external tool information stored in process database. The Java RMI technology is employed for implementing the middleware between the Cockpit and Communication Server and among the Communication Servers.

Two kinds of XML files are kept in client sides EX XML and GRAPH XML files. The EX XML files record the execution information of processes, and GRAPH XML files record the local view information of processes.

2.2 The Components of MIDAS

2.2.1 Communication Servers

The Communication Servers are the key components of MIDAS to communicate among the collaborative units. They provide services to distribute the process design execution snapshot, access process database and communicate with Cockpits. Also, the Communication Servers provide project working space. A project working space consists of a public working directory and individual working directories. The public working directory stores the EX XML files. The individual working directories store the GRAPH XML files. All of the Communication Servers in a project team are fully connected. Each one of them keeps a list of the servers' connection information (i.e. registry IP address and Server name). The list is updated whenever a Communication Server joins or leaves a project team. The communications among the Communication Servers are multi-casting according to the permissions of the processes and tasks to the project team, even though these servers are considered fully connected. The Communication Servers provide two main services: communication service, which stores and distributes the data specification and process execution

information among the collaborative unit; and operation service, which extracts process information from the process database.

2.2.2 Cockpit

Cockpit is a web-based client side program, and users can download instances of Cockpit from a Web Server. An instance of Cockpit automatically hooks up a Communication Server via the user information and a project team name selected by the user. A Cockpit provides two major modes, Author Mode and Execution Mode. The Execution Mode is divided to Manual Mode and Auto Mode. Users can edit process flow in Author mode and execute process design in both Manual Mode and Auto Mode.

Currently the Cockpit in MIDAS provides the following main functionalities:

- **Process flow editing**

  Users may create and edit process flow graphs and their execution information on the Cockpit flow editor's canvas using the flow editor module in the Author Mode. Users are able to save them to XML files. Administrators can save these XML representations to the process database. None administrative users have to save these XML representations to their own working directories in Communication Server.

- **Display and maintenance of process information archive**

The Cockpit provides a browser function to allow users to search the tasks, productions, and specifications archives. Users can access process information based on their permissions. The users may also edit process flow by simply dragging the appropriate object from the browser and dropping it onto the Cockpit flow editor's canvas in the Author Mode. As mentioned in process flow editing, administrators are able to modify the tasks, productions, and specifications archives.

- **Operations of process design**

  The Execution mode of Cockpit is divided into two sub-modes: Manual mode and Auto mode. In the Manual mode, the Cockpit provides four core operations for process design: Apply, Rollback, Expand and Shrink. In the Auto mode the Cockpits automatically explores the feasible alternative productions for logical tasks. In both modes the users need to load the initial process flow diagram and bind the design requirements into it.

2.2.3 External Tools

David S. Keyes (1996) well defined the concept of external tool in his master project.

External tools are defined as the concrete entities, to which two types of flow objects, atomic tasks and selectors, are bound. The execution of an atomic task or

a selector invokes the corresponding external tool. The external tool accepts a series of inputs and generates a series of outputs. Both of inputs and outputs are bound to their corresponding flow objects in a process flow. Outputs from one tool are typically used as inputs for another. Keyes (1996) made it possible for the input and outputs files from external tools to be transferred among different remote sites through site proxy servers, in conjunction with remote file server.

The implementation of external tools may be performed using any language, and on any platform that is capable of running a site proxy server. The requirements for accessing an external tool are as follows,

- The tool must be executable from a shell prompt or a DOS prompt.
- The tool must be capable of parsing the following command line format: *toolname <commandlineoptions> INPUTS i1 i2 i3 ... iN OUTPUTS o1 o2 o3 ... oM*
- where i1, i2, i3, ... iN are the names of the N input files and o1, o2, o3, ..., oM are the names of the M output files.

The tool must be capable of identifying the input file types regardless of their order.

2.2.4 Site Proxy Server

Site proxy servers are needed to invoke remote external tools because the external tools to complete the design processes will be distributed across many sites.

David S. Key (1996) described in detail the implementations and functions of site proxy servers also.

Each site that hosts external tools must have a site proxy server. This server acts as a proxy between system Cockpits and the various external tools on the site. When a system Cockpit is ready to invoke an external tool, it sends a message to the appropriate proxy server at site where the tool situates.

Site proxy servers perform a variety of functions to facilitate the invocation of external tools. These tasks are outlined below:

- **Processing of System Cockpit Invocation Requests**

  Each site proxy server listens to invocation requests from Cockpits. When a request is received and verified, the site proxy server prepares a system call for the local operating environment.

- **File Transfer**

  Each site proxy server handles the transfer of necessary input and output files to and from the local operating environment of its tools, respectively.

- **Return of Results to Requesting Cockpits**

  The execution of external tools is monitored by proxy servers. The proxy server captures returns the result code and output to the Cockpit that sent the invocation request when a task is completed.

2.3 External Tool Integration

Integration of user-defined external tools into an abstract process flow is a fundamental functionality of MIDAS. MIDAS uses these tools to find a flow configuration that meets specific constraints.

In order to integrate external tools with a process flow, MIDAS needs the information that binds an external tool to an abstract flow item, such as an atomic task, and the external tool invocation system. The "binding" information is stored in the process flow database with the corresponding abstract flow item.

In this section we will describe how MIDAS integrates flow entities with external tools. We will also describe the low-level steps involved in executing external tools that are bound to abstract flow entities.

2.3.1 Binding information

Binding an external tool to a flow object involves simply defining certain properties in the flow object. The following properties must be defined in an object that is to be bound to an external tool:

- **SITE**

Because MIDAS can execute tools on remote systems, it is necessary to specify the remote system where the tool is located. Typically, a default site will be specified in the system defaults, therefore it may not be necessary to define the site property of every flow object, unless the default is to be overridden.

- **CMDLINE**

  The CMDLINE property specifies the command to be executed at the specified remote site. The CMDLINE property must include any arguments that will always be sent to the external tool. The CMDLINE argument must be in the same format that would be used if the command were executed from a shell/DOS prompt.

Both of inputs and outputs of an external tool are bound to their corresponding flow objects. Site and path properties must be defined in these objects. Default Site and path will be specified in the system defaults.

2.3.2 Execution of External Tools

Once flow objects have been bound to the appropriate external tools, MIDAS can be used to perform process management. Figure 3.2 illustrates the components of the tool invocation system. The Tool Monitor seems to be a new component we have invited into the system, however, it is nothing but a thread created by Site Proxy Server to monitor the execution of an external tool. The responsibility of

Tool Monitor is to wait on the tool, store its stdout and stderr, and move any input and output files to their appropriate sites, and notify the calling site proxy server when the tool has complete. The reason we need the Tool Monitor is to improve the efficiency of the Site Proxy Server. As a thread, the Tool Monitor allowed the Site Proxy Server to handle other requests instead of spending a large amount of time waiting for a tool to complete its work.

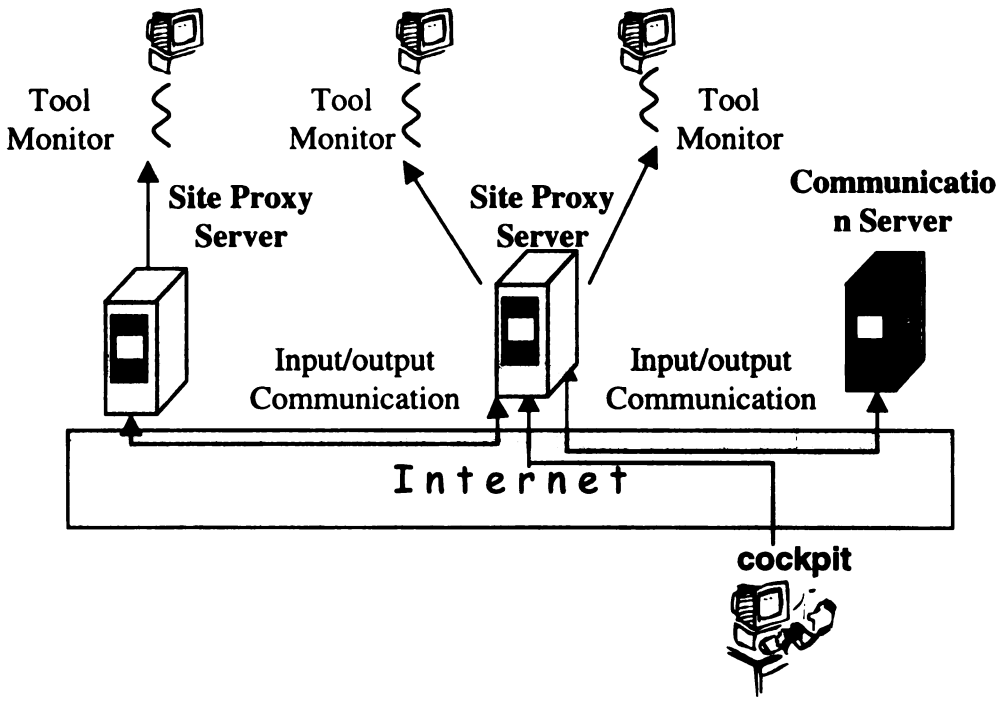

Figure 2.2 Executing external tools

When a task needs to be invoked during a process design, an instance of Cockpit sends an execution invocation request to the corresponding Site Proxy Server. A Tool Monitor is created by the proxy server to monitor the tool execution. The Tool Monitor transfers input and output files to their appropriate locations. If the

input and output files are not explicitly informed to be transferred to certain location, the default location are used.

2.4 Authorizations

In order to keep proprietary parts of productions and tasks and share the necessary information among collaborators at same time, user permissions need be granted and accessibilities of productions and tasks need be determined.

2.4.1 User permissions

In MIDAS, user permissions are employed to distinguish the users. There are three types of user permissions: administrator, member and guest.

- **Administrator**

  Administrators are the only ones having full permission to write and modify production information, read production information and execute tasks in their own collaborative unit. In addition, administrators manage the user accounts.

- **Member**

  Members do not have permissions to write or modify production information directly in production database. According to the permissions of tasks and productions, members have limited access to read production information and execute tasks in their own collaborative unit.

- **Guest**

  Guests do not have permissions to write or modify production information either. Usually, guests only have permission to view and execute some demonstrations of process designs. Guests may be invited to view a process design, but they don't have permission to execute any tasks in a process.

2.4.2 Accessibilities of tasks and processes/productions

In a project team across different collaborative units or companies, tasks and productions belonging to a collaborative unit have to have permissions to access other collaborative units. However, if a production in certain collaborative unit is proprietary, others collaborative units do not have any permission to read, to write or to execute it. If a production in the collaborative unit is not proprietary, other collaborative unit will be granted permission to access it. To simplify tasks and productions management, we only give the administrators the "write" permission in their own collaborative unit. Likewise permitted users in a collaborative unit can only execute tasks and productions belonging to their own collaborative unit. When "read" and "execute" permission is assigned a value of "any" in a collaborative unit, any users in the unit have this permission; If it is assigned a value of "administrator", only administrators have this permission; if it is assigned a permission value of "member", both administrators and members have the permission to "read" and "execute".

Chapter 3 XML representation of static processes information

A valid XML data representation of processes followings specifications in the grammar based process model. The XML representation conforms to the structure, content and semantics defined by XML schema, which needs to be compatible with the grammar based process model. It is more difficult for users to understand XML Schemas implementation than to focus on conceptual domain modeling. Compared to XML Schema, UML makes it easier to visualize the model, and to ensure that integrity constraints are defined.

Not only does UML help the architects, developers and programmers of software development specify, visualize, and document models of software systems, but also it is valuable for data modeling (Saldhana, Shatz, and Hu, 2001). Skogan (1999) uses UML as a schema language to define data interchange format based on the XML. He has developed an abstract representation model of schema and XML using UML and provided conversion rules to translate the UML representations to DTD and XML. Booch et al. (1999) and Carlson (2001) have done similar works using UML to provide a graphic representation of XML schema for better communication. However, in these applications, the designers are still exposed to some low-level implementation issues of XML Schema. In order to allow designers to graphically design XML Schemas without exposing them to low-level implementation issues, Routledge, Bird and Goodchild (2002) have developed a traditional three level database design approach to build a mapping between UML and XML Schema, the conceptual level is represented using

standard UML class notation, the logical level is represented in UML, using a set of UML stereotypes, and the XML Schema itself represents the physical level. This approach allows the data modeler to begin by focusing on conceptual domain modeling issues rather than implementation issues. They have developed an algorithm, which is used to generate a logical level representation based on a conceptual level UML class diagram. This algorithm can minimize redundancy in the XML-instances, while maximizing the connectivity of the XML data structures.

In this work, the three level design approach proposed by Routledge, Bird and Goodchild (2002) is employed to transform the conceptual level UML class diagrams of the grammar-based process modeling into XML Schema through successive steps. There is no standard UML representation of XML schema, but the conceptual level UML diagram employed by Routledge, Bird and Goodchild (2002) does not conform to the standard UML class diagram recommended by OMG. Therefore, The concise UML representation of XML schema used by Booch et al. (1999) and Carlson (2001) are adopted in our work since their UML diagram follows the recommended standard more closely. A box with three entries represents a type of element. Type and name of the element are in the first entry, attributes of the element are in the second entry, and functions the element has are in the third entry. Detailed description of association among different object classes is given in this work.

Process flow graphs and process grammar are summarized here according to the description in the introduction.

- A process flow has at least one task object (logical or atomic or selector).
- A logical task object has at least one production, and a production belongs to at most one logical task object. A production has one to one mapping relationship with a process flow.
- Database objects are the inputs only for selector objects. Also, database is not considered as an output of a task in this thesis.
- A data object can be an output only for one task (logical or atomic or selector).

3.1 Setting up conceptual level UML diagrams

The UML diagram of grammar-based process model we have created is shown in Figure 3.1.

A process flow has at least one task object, which can be one of three types, logical tasks, atomic tasks and selectors. Because a task object is a part of a process flow, the association between a task and a process is an aggregation. Although a process flow has at least one task object, but a task object may not belong to any process flow in MIDAS system. Therefore, the multiplicity is '0..*' on the process class side, and the multiplicity is '1..*' on the task class side. The logical tasks, atomic tasks and selectors are considered the inheritances of the task objects. Because data specification objects are used in both process flows and task objects, we create two classes, task data class and process data class.

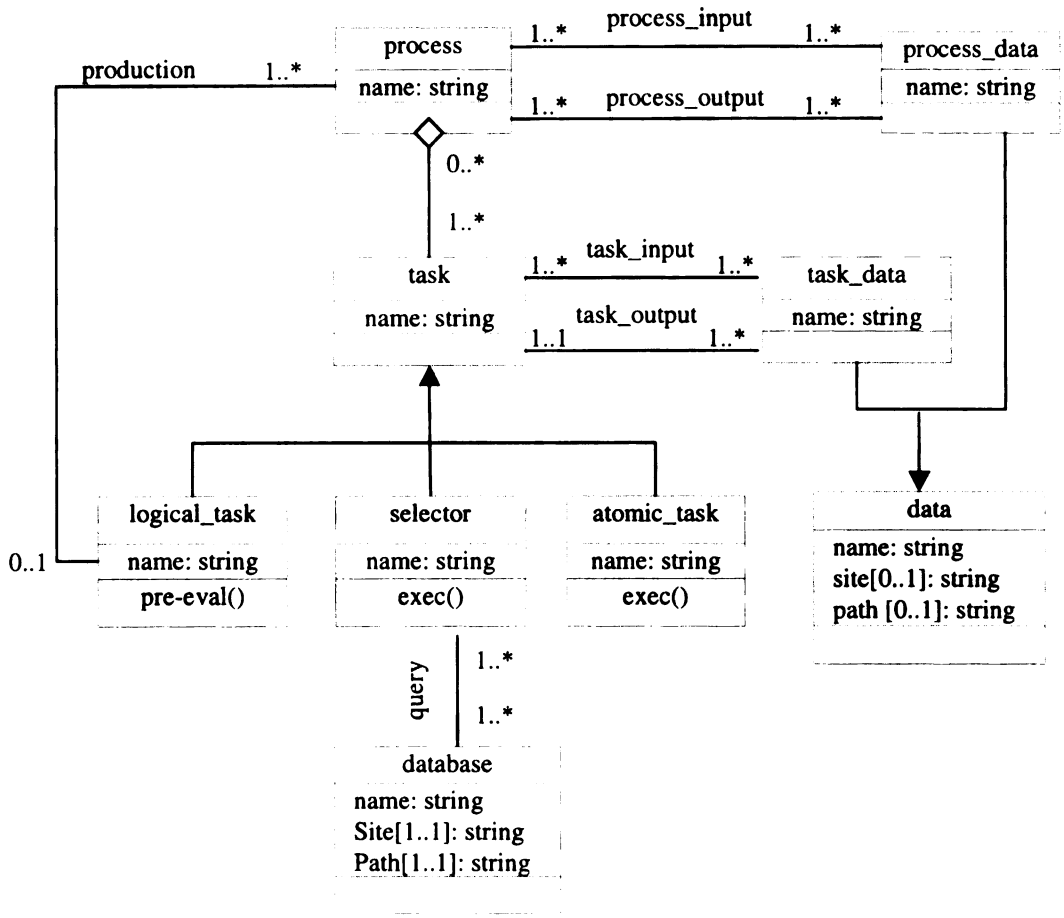

Figure 3.1 The conceptual level UML diagram of grammar-based process model

The multiplicities at both ends of production association between process flow and logical task classes are '1..*' on the process flow class side and '0..1' on the logical task class side. Process flow class has both input and output associations with process data class. Obviously, both ends of inputs and outputs associations between process flow class and process data class have multiplicities of '1..*'. Similarly, there are inputs and outputs associations between task class and task data class. However, both ends of the input association have multiplicities of '1..*', while the task class

end of the output association has a multiplicity '1..1', and the task data end has a multiplicity '1..*' because a data object can only be an output for one task object. There is only input association between selector and database because database cannot be an output of a task, and both ends of the input association have a multiplicity of '1..*'.

The pre-evaluation function, 'pre-eval()', in logical task class, and the execution function, 'exec()', in both of atomic task and selector classes are programming functions. To further map them into logical level UML diagrams, which only include the information in the form of data, we need to change the function to the execution information. To execute the functions, we need to know their locations and execution commands. Therefore, these UML diagrams are further transformed into the diagrams (Figure 3.2).

The new version of MIDAS not only has the distributive components, but also can use distributive information to accomplish a process design collaboratively. We consider that MIDAS supports a number of collaborative units. The members in different units work collaboratively. They not only share information, but maintain their own process flows, logical tasks, atomic tasks and selectors. The XML representations of processes are used as a means to integrate the shared information. For the benefit of process information distribution and easy maintenance of process information, our XML representations of process management information are

divided into process XML files, logical task XML files, atomic task XML files and selector XML files in MIDAS.

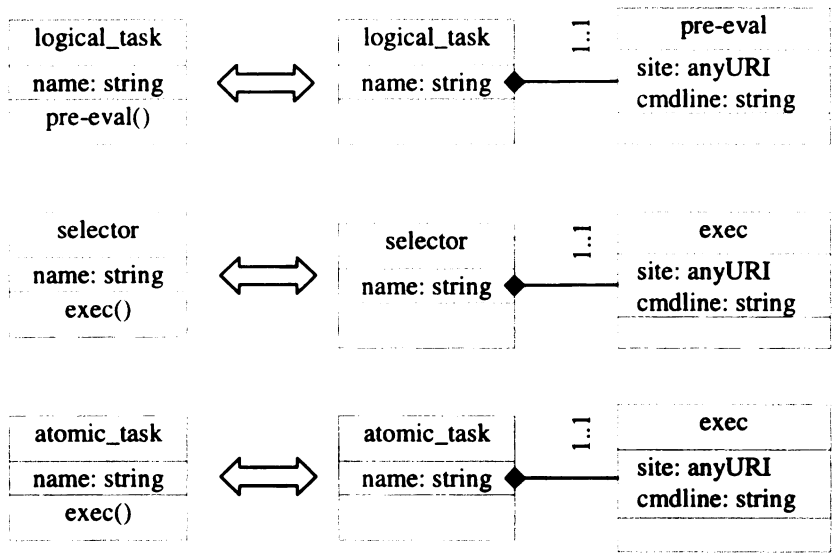

Figure 3.2 Transformation of logical task, atomic task and selector classes

Each process XML file represents a process flow. In order to have maintain process information well organized, we keep the logical task XML files, atomic task XML files and selector XML files separately. Therefore, we decompose the conceptual level UML diagram of the grammar-based process model in Figure 4 into the following diagrams: The conceptual level UML diagram of a process in Figure 3.3, the conceptual level UML diagram of logical tasks in Figure 3.4, the conceptual level UML diagram of atomic tasks in Figure 3.5, and the conceptual level UML diagram of selector in Figure 3.6.

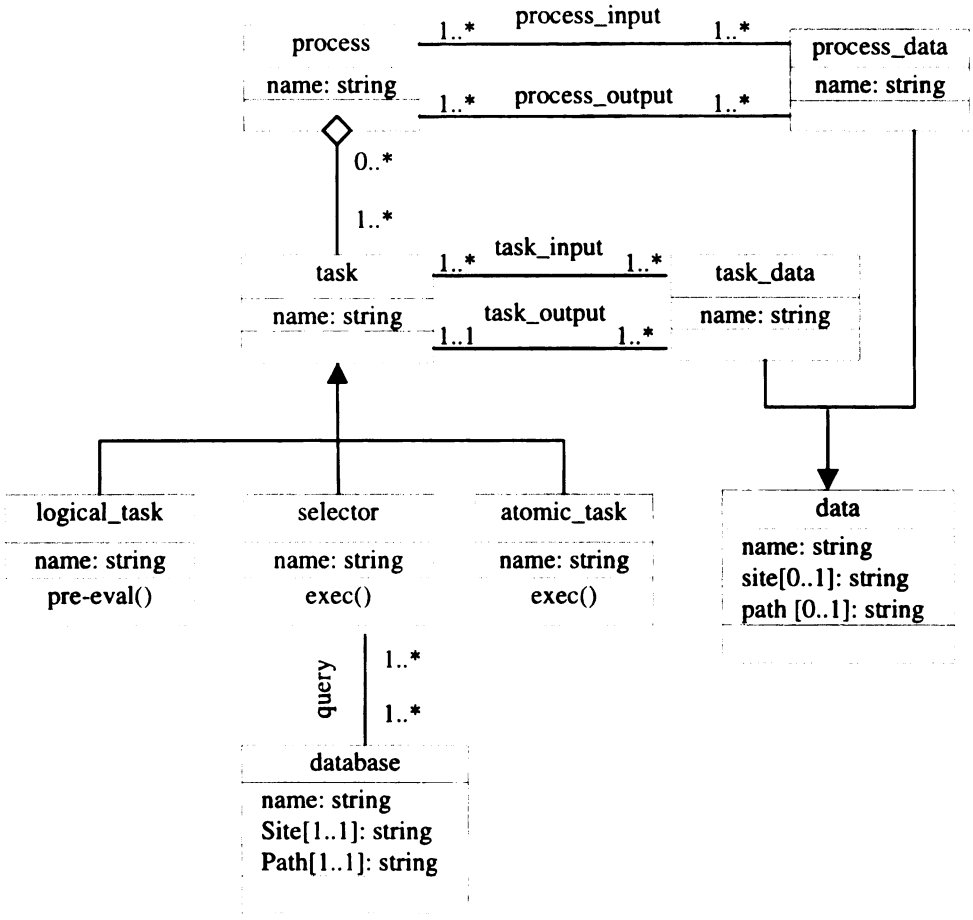

Figure 3.3 The conceptual level UML diagram of processes

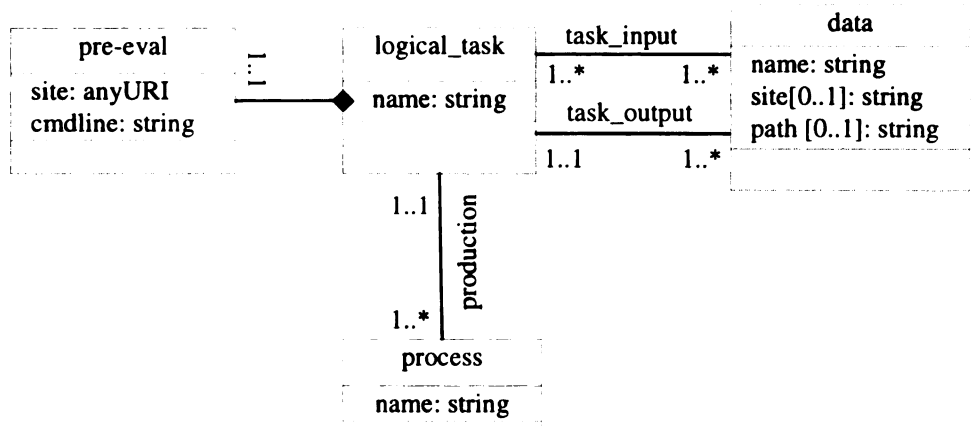

Figure 3.4 The conceptual level UML diagram of logical tasks

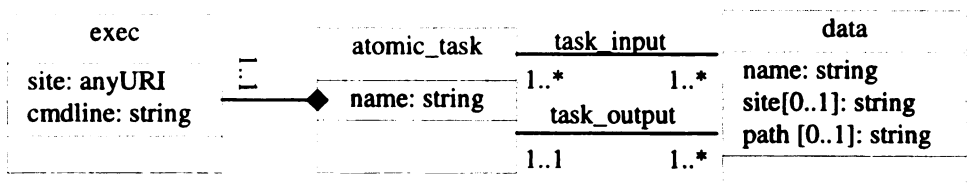

Figure 3.5 The conceptual level UML diagram of atomic tasks

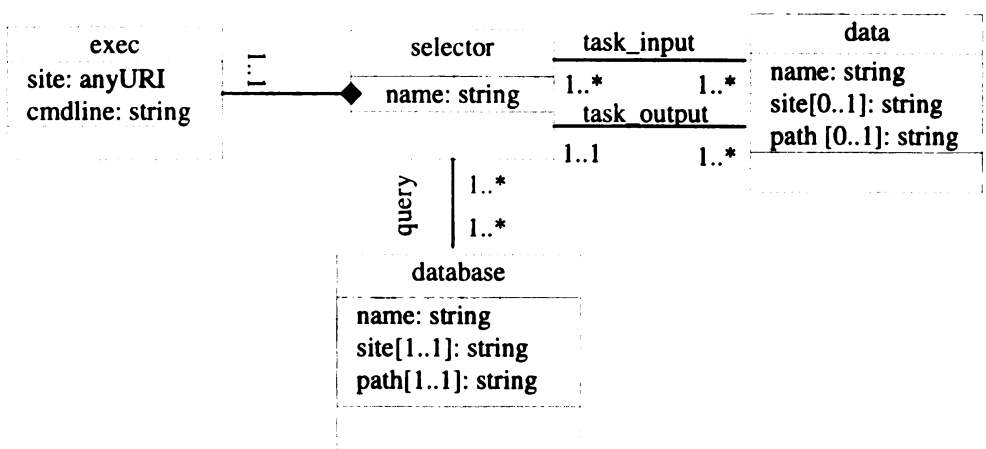

Figure 3.6 The conceptual level UML diagram of selectors

The decomposition of the conceptual level UML diagram of the grammar-based process model could be done in various ways according to their different applications. Some redundancy exists in our decomposed models due to the following reasons,

- If we keep the input and output information only in the task XML files, it is impossible to describe a process flow graph properly because it is not allowed for two tasks to have even one identical output in a process, but this does not mean that two tasks in different processes cannot have any identical output. If we keep the input and output information only in the process XML files, it is difficult to provide more functions for MIDAS, such as browsers.
- There are two kinds of users in MIDAS, designers and system managers. Designers usually create a process using the information existed in MIDAS system. System managers update the logical tasks, atomic tasks and selectors information. However, a designer is able to create a process, which includes tasks not existed in MIDAS, and the designer may not provide the detailed execution information. Therefore, it is necessary to use the process XML files to record the graphic information of the process. Tasks information redundancy cannot be avoided due to the need of system implementations.

- It is also convenient for those who are interested in the XML files as well as for system managers who manage the system information. However, compared to the reasons that are discussed earlier, this reason is not the essential reason.

3.2 Logical level UML class diagrams and physical level Schemas

The methodology proposed by Routledge, Bird and Goodchild (2002) and the UML diagram representation of XML schema used by Grady Booch, Magnus Christerson, Matthew Fuchs and Jari Koistinen (1999) are combined to generate the logical level UML diagrams from the conceptual level UML diagrams created in last section. However, Routledge, Bird and Goodchild (2002) did not use standard UML association concepts in conceptual level UML diagram. We have applied more UML notions in our conceptual level UML diagram, such as generalization, to accurately express the grammar-based process flow model. We have also added more guidelines to the mapping methodology.

Mapping methodology is utilized in four types of XML files, including process flow files, logical task files, atomic task files and selector files. In order to explain the mapping methodology from conceptual level UML to the logical level UML, the conceptual level UML diagram of process flow is used as an example to generate logical level UML diagram.

- **Type definition**:

Routledge, Bird and Goodchild (2002) pointed out that each conceptual class should be mapped into a complexType. Its key attributes are included in the complexType definition as XML Schema attributes. Its non-primary attributes are represented by child elements. Attributes, which have restriction to their contents, map into simpleTypes. Primitive types, used by an attribute, are mapped into XSD simpleTypes from the XML Schema namespace.

In order to reduce some redundancy, only necessary classes are mapped to complexType in this paper. The task data class and process data class are presented in Figure 3.3 in order to help users to understand the design at the conceptual level, although they are not really necessary at the logical level. Therefore, task data class and process data class are skipped during type mapping since their corresponding associations can be used to identify the type of the data class. On the other hand it is a different situation for the task class. Its child classes, logical_task, atomic task, and selector, do not have individual association to be used to distinguish themselves from each other, and these classes inherit all of the features from their parent class. Therefore, it is possible to skip the task class. The type definitions in the logical level UML diagram are shown in Figure 3.7.

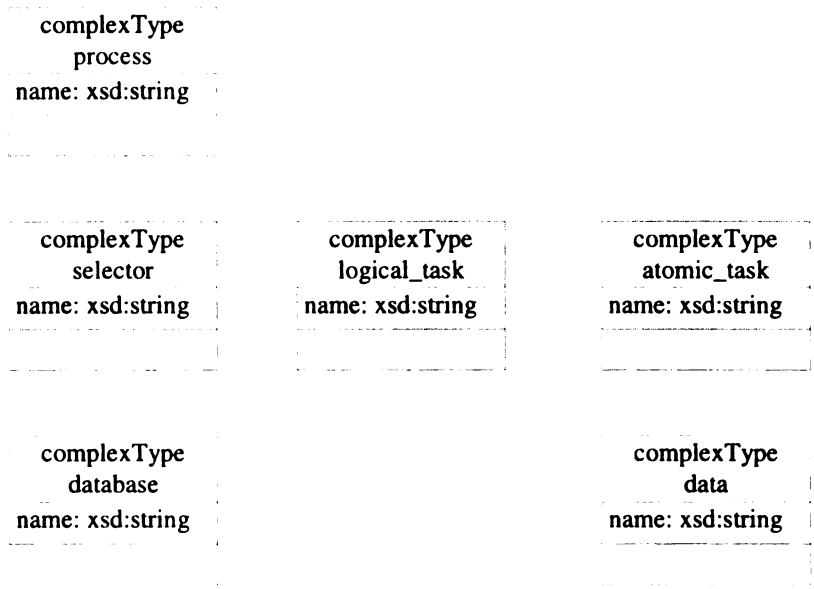

Figure 3.7 The type definitions of logical level UML diagram of processes

- **Determine Class Groupings and Build the Complex Type Nestings**

To determine an appropriate nesting for the schema based on the associations between them, navigations of associations needs to be determined first. It is straightforward to decide the directions of the navigations of associations using the approach summarized by Routledge, Bird and Goodchild (2002).

"a. If exactly one association end has a minimum multiplicity of 1 (i.e. (1..1) or (1..*)), then define the navigation in the direction of the opposite association end, or

b. If one association end has a smaller maximum multiplicity than the other, (e.g. '0..7' is smaller that '0..*' then navigation towards the end with the smaller maximum multiplicity, or

c. If exactly one class has only one attribute, then navigate towards it."

The following rules are used to determine nesting:

"1. If exactly one association end has a multiplicity of '(1,1)' (i.e. it is mandatory and functional), then nest the class at the other association end towards it.

2. If both association ends have a multiplicity of (1,1), then nest the classes in the opposite direction to the direction of navigation – i.e. nest the target of the navigation towards the source of the navigation"

According to the rules listed above, the data class is nested into the process class, the atomic_task, logical_task and selector classes. An example of the logical level mapping UML diagram of data class nesting into the process class is represented in Figure 3.8.

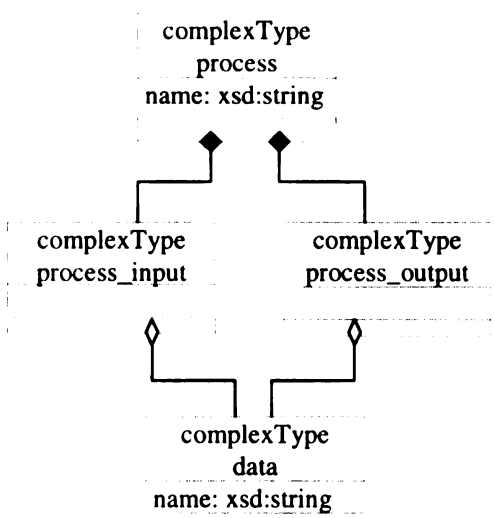

Figure 3.8 The logical level mapping UML diagram of data class nesting into the process class

A class called "tasks" is created for the aggregation association from task class to process class. Therefore, atomic_task, logical_task and selector classes are nested into tasks class, and tasks class is nested into process class.

To set up the 1..1 to 1..* task_output association between the task class and data class in the conceptual level UML diagram of processes, the key constraint of the attribute "name" of the data class has to be added to the tasks class in logical level UML diagram of processes. The aggregation association of atomic_task, logical_task and selector classes to tasks class indicates that tasks element contains at least one of atomic_task, logical_task and selector elements. We use the compositor "choice" and its cardinality to represent this aggregation association. Also, the key constraint of the attribute "name" of the data class is added to the task_input, process_input and process_output class to protect data from duplication.

The final logical level UML diagram of a process is shown in Figure 3.9. In addition, the logical level UML diagram of logical tasks is presented in Figure 3.10, the logical level UML diagram of selectors in Figure 3.11, and the logical level UML diagram of atomic task in Figure 3.12. Even though translating the logical level UML diagrams to the physical level XML schemas is straightforward, the extensive XML Schema knowledge is needed to build final XML schemas. These XML schemas are attached at the end of this paper (Appendix 1).

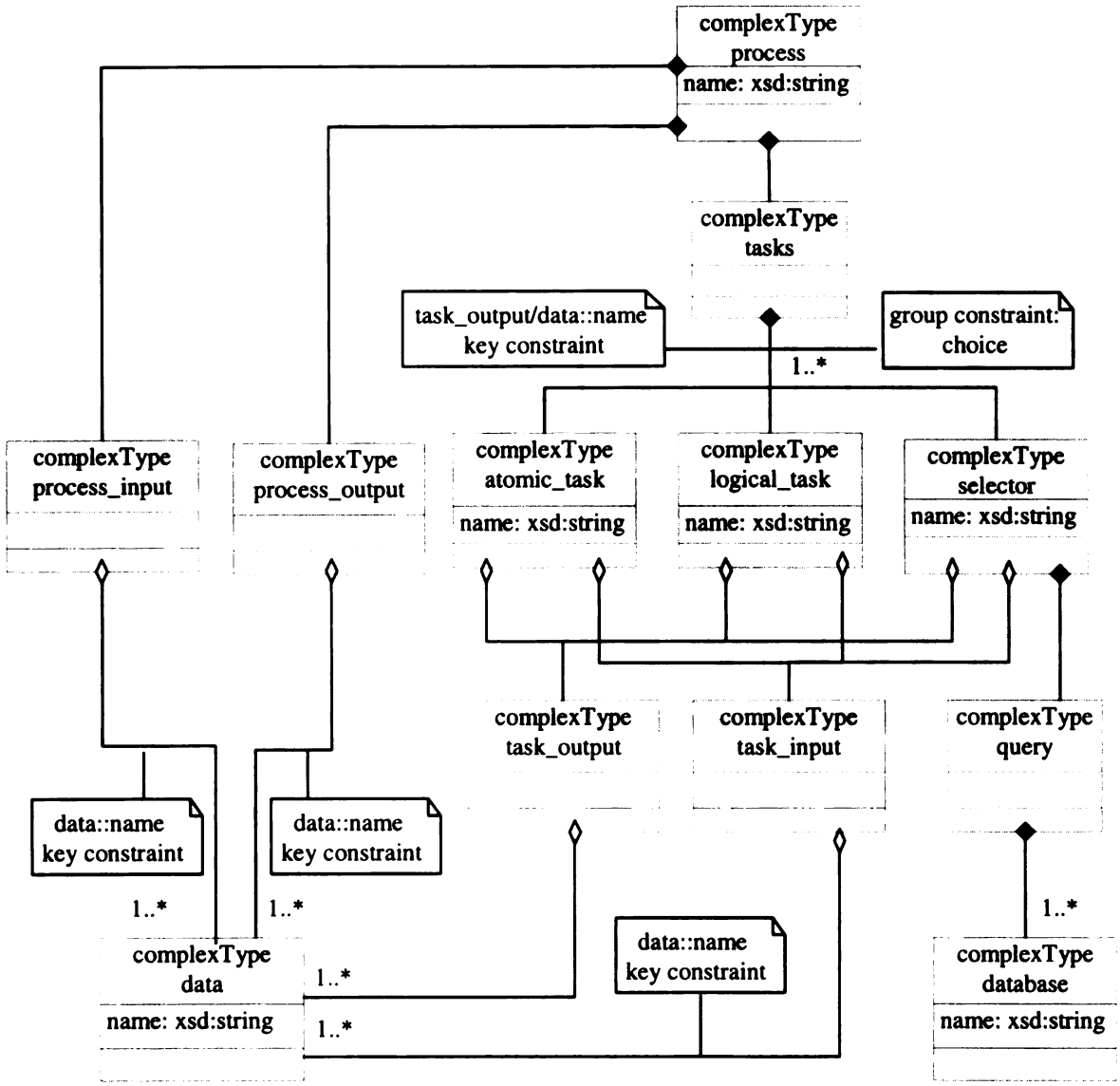

Figure 3.9 The logical level UML diagram of processes

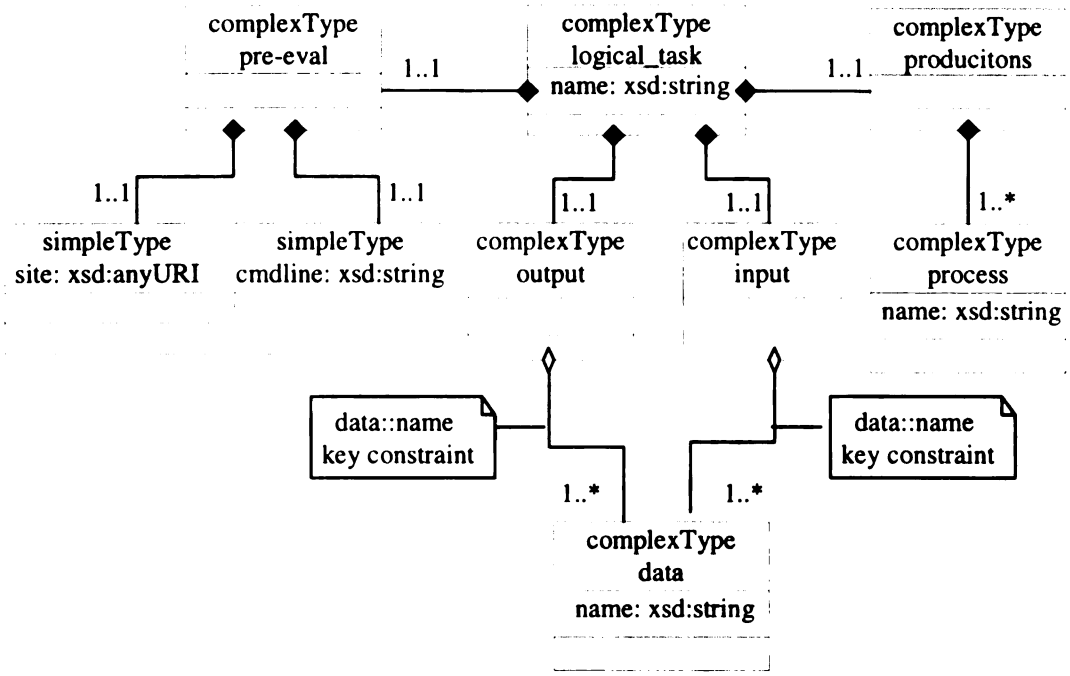

Figure 3.10 The logical level UML diagram of logical tasks

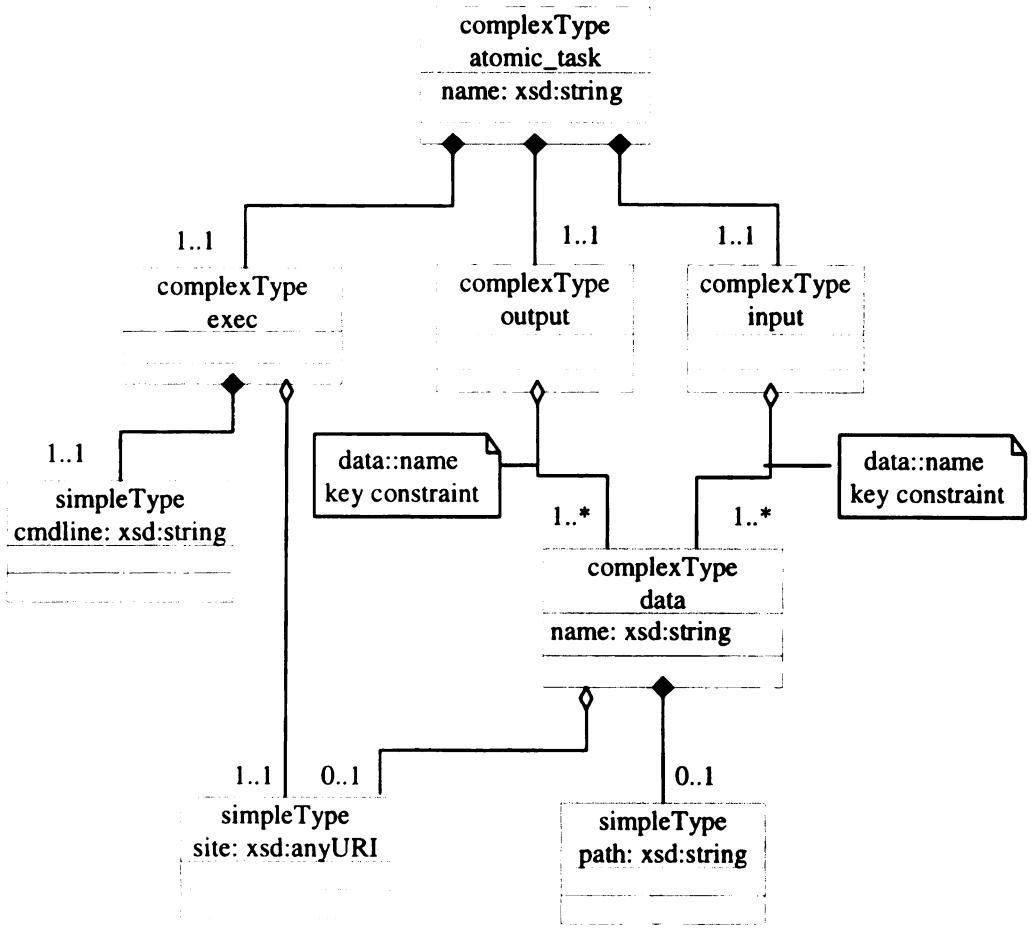

Figure 3.11 The logical level UML diagram of atomic tasks

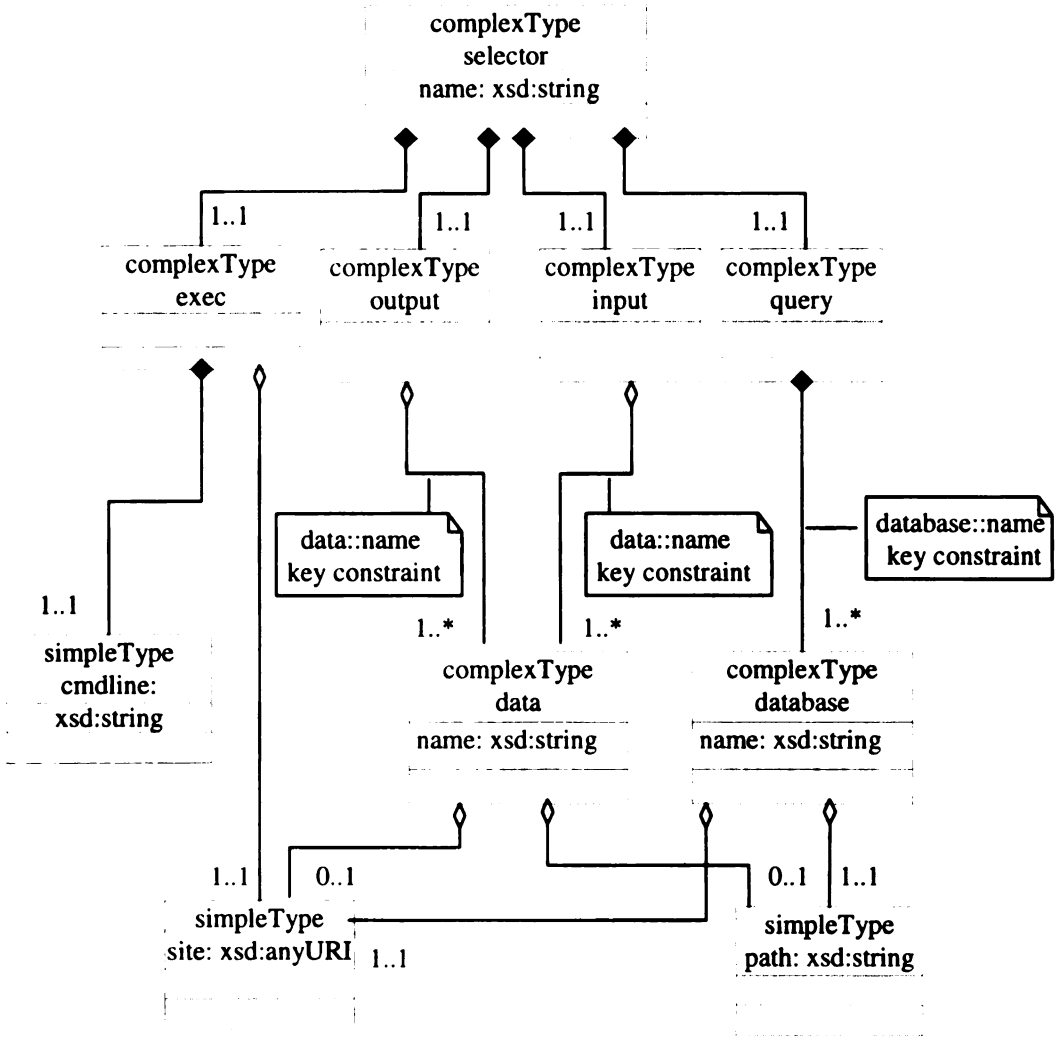

Figure 3.12 The logical level UML diagram of selectors

Chapter 4 Multiple-thread Scheduling

To run a process efficiently and execute tasks in a distributed environment, MIDAS creates a thread for each task. Therefore, multi-thread scheduling becomes a fundamental feature in MIDAS. The multiple-thread scheduling in MIDAS is about how to manipulate multiple threads to search all possible process configurations to discover the one that meets user specification requirement. Because the Proxy Server is responsible for executing atomic tasks and selectors, the threads discussed here are only for logical tasks.

4.1 Requirements of a successful scheduling approach

The requirements of a successful scheduling approach recommended by Keyes (1996) are adopted to verify our new scheduling approach. The requirements are listed as follows:

- **Identification of Successful Configuration**:

  A correct scheduling approach is able to determine whether certain configuration meets user specified requirement.

- **Exhaustive search**:

  A correct scheduling approach is able to search all possible process configurations to find a suitable one if it exists. Otherwise, a failure is reported.

- **Reasonable Performance**:

  The search of all possible process configurations grows exponentially (in the number of logical tasks). Therefore, a successful scheduling approach should be able to provide a more efficient way to search a desirable process configuration.

4.2 Scheduler and Scheduler Monitor

Whenever a process flow graph is loaded or a production is applied for a logical task, a thread is generated. The thread named "scheduler" is responsible for the elaboration of the task. Therefore, schedulers are related to each other through the task/productions relationship in MIDAS. The schedulers having the task/productions relationship are considered as parent/child schedulers. Because MIDAS is a distributed system, parent/child schedulers are able to execute at different collaborative sites. Therefore, to maintain schedulers at each collaborative site, a thread named "scheduler monitor" is necessary at each collaborative site.

A scheduler can be generated either by its parent scheduler or by a scheduler monitor. The first scheduler for the initial process is generated by a scheduler monitor. A scheduler can directly produce its child schedulers if these schedulers run at the same site as it does; otherwise, it will trigger the scheduler monitors, which are located at the same site as its child schedulers will be, to generate its child schedulers.

Parent/child schedulers keep the pointers of each other if they run at the same site; otherwise, they keep each other's site and name. . A scheduler monitor keeps pointers of all the schedulers at its site. Child schedulers inform its parents whenever their corresponding process flow is successful or failed, and parent schedulers can kill child schedulers by forcing tasks to fail. If parent/child schedulers are at the same site, they can achieve the communication with each other via pointers, otherwise, they need go through scheduler monitors to communicate with each other.

4.3 Algorithm

4.3.1 Data Structure

Each scheduler maintains a data structure, which records the tasks execution information and the relationships of tasks in the process or sub-process that the scheduler corresponds to.

In order to run parallel tasks, schedulers have to maintain more complex topological structures than the Linear Scheduler (Keyes, 1996) does. A Scheduler maintains a data structure, which has the same topological structure as its corresponding process flow or an alternative of a logical task, e.g. Figure 4.1. Each node corresponding to a task in the data structure is called "task node". To make sure all task nodes can be reached from a single node, a "Ghost" node is employed to lead to all "top" task nodes, which do not have any task nodes

leading to them (Figure 4.1). To check the completeness of a process, the schedulers have to check all of "bottom" tasks, which do not have any task following them. An "End" node is created to follow these task nodes (Figure 4.1). If the "End" node is reached, all of task nodes leading to it will be checked for completeness. If all of these task nodes are successful, the corresponding process is completed. Otherwise, the corresponding process is uncompleted.

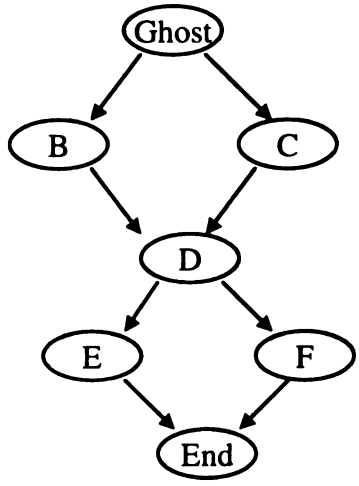

Figure 4.1 A topological structure maintained by a scheduler

To select a rollback route, each task node should keep its corresponding alternative production names and statuses. An alternative production could be assigned one of three statuses: status "available", which represents that the alternative production is available to execute; status "failed by requirement", which represents that the alternative production is failed by not satisfying design requirement; and status "failed by configuration", which represents that the alternative production is failed by inappropriate configurations.

task name	prod. name	status
	prod1	available
	prod2	failed by requirements
	prod3	failed by configuration
	prod4	available

Figure 4.2 The task/production structure

4.3.2 Searching a successful process configuration

Rollback is critical to explore a successful process configuration. Rollback of a node to its parent is explored here in order to find an efficient way to handle the multi-threading algorithm.

To cover all possible process configurations, the task nodes (e.g. node D, in Figure 4.1) with multiple parent nodes have to do rollback to their parents sequentially (e.g. node B first and node C next, in Figure 4.1). However, it is not necessary for the task nodes (e.g. node B, in Figure 4.1) with only one parent node to consider their siblings when rolling back to their parents. However, there is an issue worth considering, that is, for some nodes (e.g. node B, in Figure 4.1), they need to know when rolling back to their parents and when rolling back to their siblings. Therefore, analyzing what causes nodes rollback is necessary.

As mentioned at beginning of this chapter, threads discussed here are only for logical tasks. Therefore, rollbacks discussed here are for logical tasks. Rollback of a logical task is caused by fails of all of its alternative productions. An alternative production is failed by not satisfying the design requirement itself or by inappropriate configurations. For example, if all of the alternative productions of task B are failed by not satisfying design process requirements, it is not necessary to search valid alternative productions of task C. However, if some alternatives of logical task B are failed by inappropriate configuration when searching a valid configuration of an alternative production of logical task B and an alternative production of logical task C, it is necessary to search other available alternative productions of task C. Therefore, the data structure we represented above is needed to store alternative production status.

Rollback mechanisms are provided for different scenarios as follows:

- A task node rolls back to itself to apply an available alternative if some of it alternative production statuses are "available".
- A task node rolls back to its parent node if the statuses of all alternative production of the task are "failed by requirement". After rollback, the statuses of all alternative production of the task will change back to "available".
- A task node rolls back to the node that just before it in a topological order if the statuses of some alternative productions of the task are "failed by requirement", and others are "failed by configuration". If the node a task

node rollbacks to is the task node's parent node, its parent node will force all its following node in the logical sort change their production statuses to "available", otherwise, the task node will change its alternative productions with status "failed by configuration" to status "available".

With the knowledge of rollback rules discussed above, it is possible to efficiently search a successful process configuration.

Chapter 5 XML Execution models in MIDAS

In addition to storing the static process information, such as process flow graphs, tasks and productions, XML files also play important roles in process elaboration, collaboration and visualization. The XML files for process execution information are named EX XML files, and the XML files for process flow display information is named GRAPH XML files.

5.1 EX XML in MIDAS

Before discussing the role of EX XML in MIDAS, we need to emphasize two rules for the communications among the collaborators.

- If a collaborator does not have any authority to see the initial process flow, the collaborator cannot see any part of the elaboration of the process except for the tasks belonging to the collaborator, because it is possible to figure out the initial process flow if the collaborator obtains more information.
- If a company has proprietary processes, it will not share any information related to these processes with other companies, including process flow, because if the company shares some information with certain companies, these companies may share the information with the company's competitor. Eventually, the company will lose its proprietary rights.

Each EX XML file corresponds to a Scheduler. Whenever a Scheduler is changed,the change is recorded in its corresponding EX XML file. The EX XML file is named following the name of the Scheduler plus "_EX". After the EX XML file is modified, the Scheduler informs its parent Scheduler to obtain the up-to-date EX XML file if the process corresponding to the EX XML file is not proprietary; otherwise, the Scheduler only informs its parent Scheduler its status. The status can be running, success or failure. The parent Scheduler recursively informs its higher-level Scheduler to obtain the up-to-date EX XML file too if the process corresponding to the parent Scheduler is not proprietary; otherwise the parent Scheduler informs its higher-level Scheduler whenever its execution status changes. Eventually the notification stops. If the notification stops in the initial Scheduler (it does not have parent Scheduler), the initial Scheduler will inform the permitted sites to view the initial process.

To improve the efficiency of the notification procedures, a scheduler does not receive an up-to-data EX XML file if it is informed by a lower-level scheduler at the same site.

EX XML files are employed to record the process elaboration operations. In EX XML files, we add an attribute to tasks, called "status", in order to keep track of the statuses of the tasks. For an applied logical task, we nest the applied productions into it. If the logical task is failed, we keep the failed information for its productions and

apply another alternative production for it. The conceptual level UML diagram of EX XML model is shown in Figure 5.1.

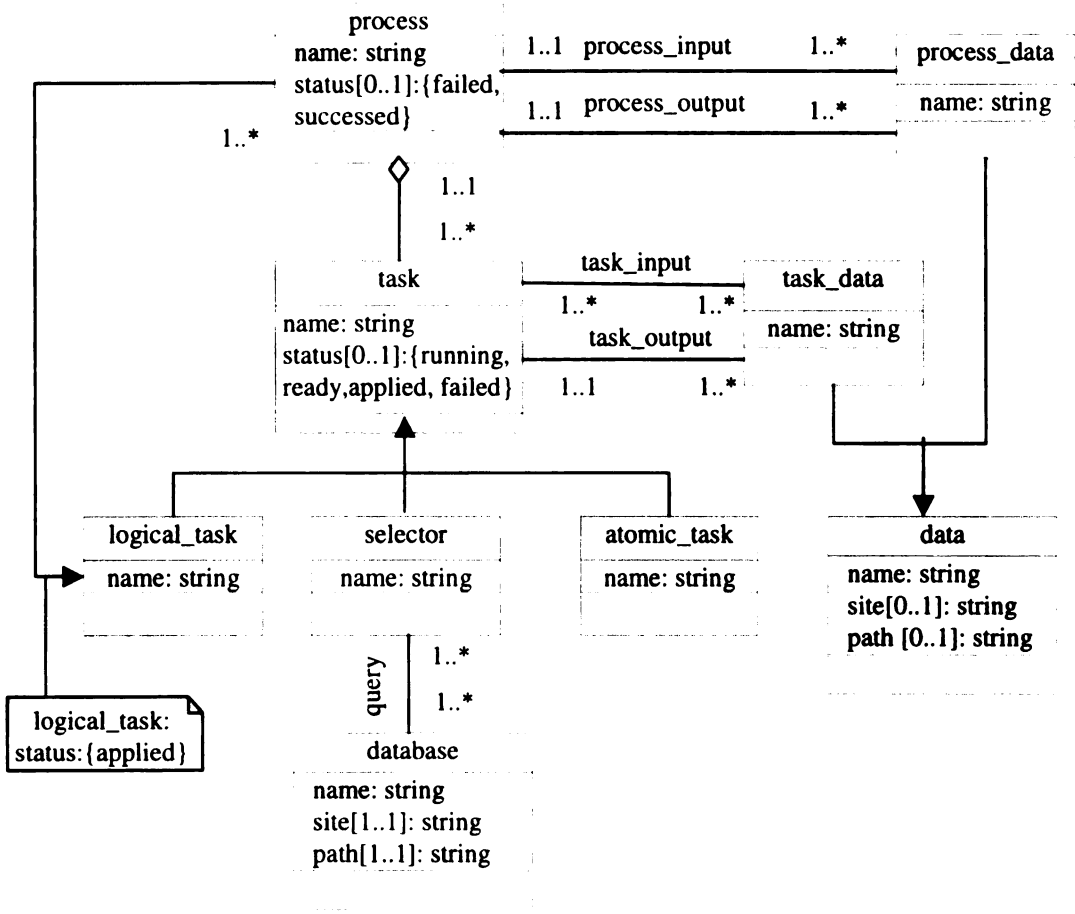

Figure 5.1 The conceptual level UML diagram of EX XML model

5.2 GRAPH XML in MIDAS

GRAPH XML files store the process flow display information. Different users may choose to see the process flow differently according to their needs. Therefore, each user has to have his/her own GRAPH XML files. A collaborator keeps only one GRAPH XML file. The first GRAPH XML file is created at the same time as the first EX XML file is generated. Thereafter, the GRAPH XML file is updated whenever an EX XML file is created or modified. The names of tasks and EX XML files are employed to locate the part of the GRAPH XML file needed to be modified. The GRAPH XML files record the operations "expand" and "shrink" as well.

GRAPH XML files are also employed to serve the visualization purpose of process flow graphs. The initial GRAPH XML file is created by copying the EX XML file directly because both of them keep the same information at the initial status. In GRAPH XML model, one more value "shrunken" is added for the attribute "status" of logical tasks. Productions are not recorded in GRAPH XML model, but the shrunken tasks have to be kept for expansion later on. The conceptual level UML diagram of GRAPH XML model is shown in Figure 5.2.

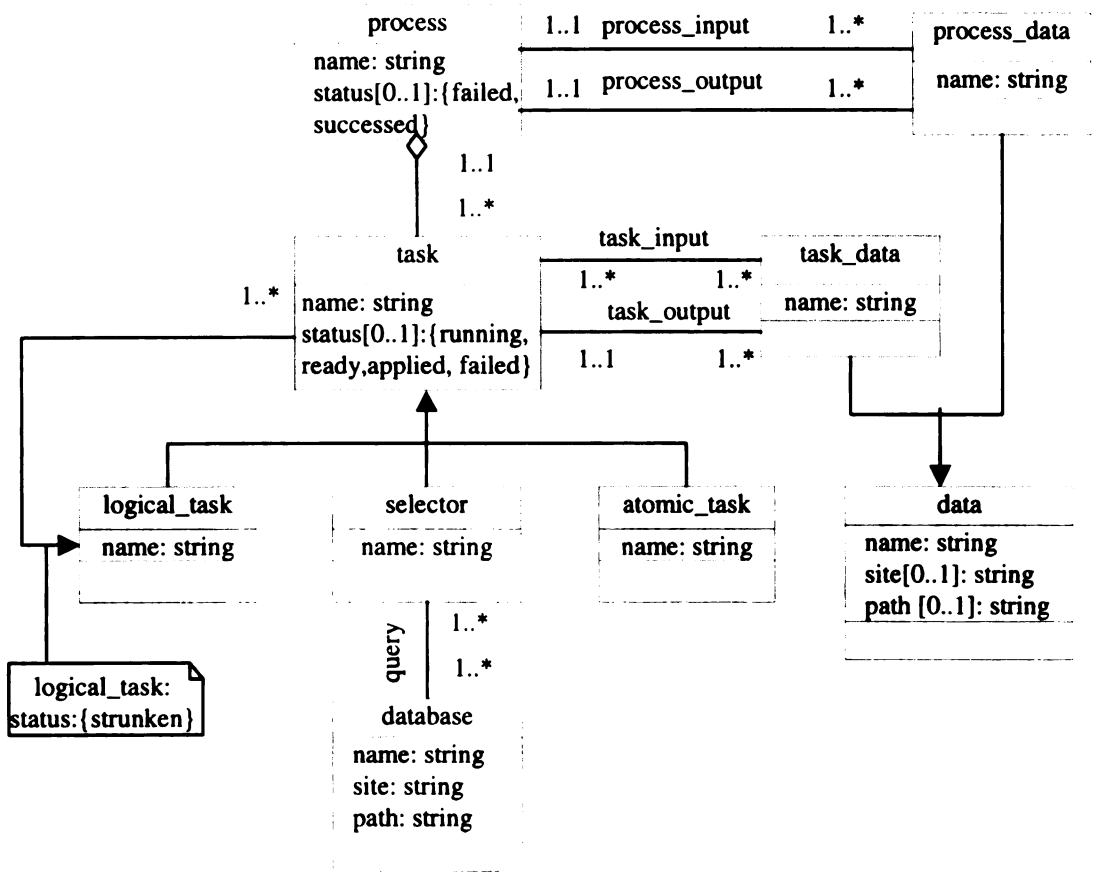

Figure 5.2 The conceptual level UML diagram of GRAPH XML model

Whenever an EX XML file is changed or a new EX XML file is generated at a site, the MIDAS will update the GRAPH XML file in this site. Also, GRAPH XML file will be changed when a user shrinks a logical task to hide the details of a process or expands a logical task to see the details of a process.

Chapter 6 A Scenario

In this section, an example is given to illustrate how the Schedulers and XML files work. Figure 6.1 shows the information of a process elaboration, and related XML files in Appendix 2.1~2.9.

Suppose that two collaborators work together. The initial process flow in collaborator1 has two logical tasks A1 and A2. Logical task A1 has one alternative, A1_P1, which is executed in collaborator 2. Logical task A2 has two alternatives, A2_P1 and A2_P2, which are executed in collaborator 1. Alternative, A1_P1 consists of two logical tasks, D1 and D2 and an atomic task, D3,. Logical task D1 has two alternatives, D1_P1 and D1_P2, which are executed in collaborator 1; logical task D2 has two alternatives, D2_P1 and D2_P2, which are executed in collaborator 2. In this example, task A1 and its alternatives are proprietary, and collaborator 2 is not authorized to see the initial process.

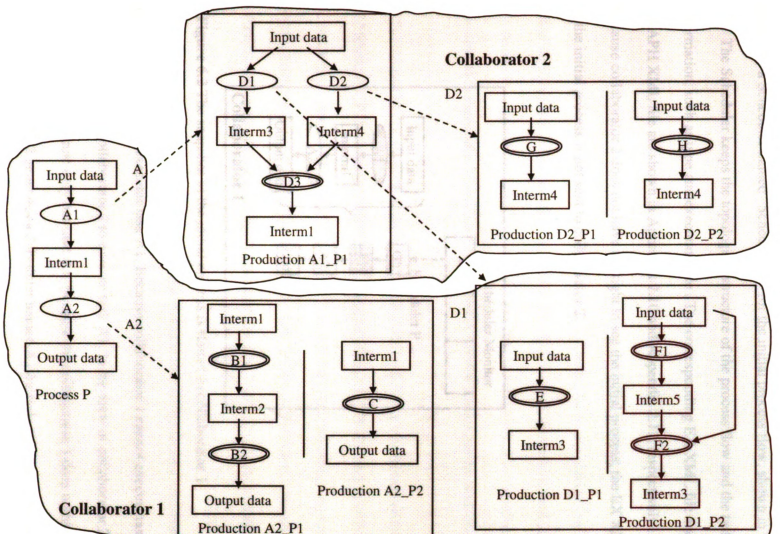

Figure 6.1. The information of a process elaboration

After collaborator 1 loads an initial process flow (root flow), the Scheduler Monitor generates a Scheduler named "Scheduler P" for the initial process flow, shown in Figure 6.2. The Scheduler keeps the topological structure of the process flow and the task information, such as type and execution site . The corresponding EX XML file and GRAPH XML file are shown in Appendix 2.10 and Appendix 2.11, respectively. Because collaborator 2 does not have the right to see the initial process, the EX XML file of the initial process is not sent to collaborator 2.

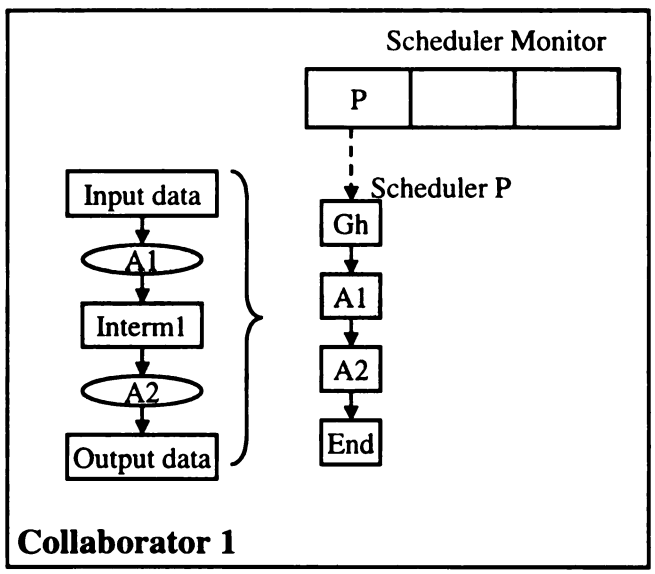

Figure 6.2 The snapshot as the initial process is started in collaborator 1

First, the Scheduler P executes task A1. Because collaborator 1 cannot execute task A1, collaborator 1 asks collaborator 2 to execute task A1. In the view of collaborator 2, the process going to execute only has task A1. Because the collaborator 1 does nothing about task A1, collaborator 2 generates a Scheduler named A1 for task A1 from its Scheduler Monitor first. Then, the Scheduler A1 generates the Scheduler A1_P1 for alternative

A1_P1 of task A2. A snapshot of generation of Scheduler A1 in collaborator 2 is shown in Figure 6.3.

The EX XML file of process P is in Appendix 2.12. Because task A1 and its alternatives are proprietary to collaborator 1, EX XML files of process A1 and A1_P1 are not sent to collaborator 1. Collaborator 1 only receives the status changing information of task A1. Therefore, the GRAPH XML file of collaborator 1 does not have any elaboration information of task A1 (Appendix 2.13). The EX XML files of process A1 and A1_P1 are shown in Appendix 2.14 and Appendix 2.15, and GRAPH XML file for collaborator 2 is shown in Appendix 2.16. Any changes from previous XML files are highlighted.

After Scheduler A1_P1 is created, the logical task D1 and D2 in the alternative the Scheduler A1_P1manages are ready to be executed. Logical task D1 runs in collaborator 1, and logical task D2 runs in collaborator 2. Through the same mechanism as mentioned above, Scheduler D1 and Scheduler D1_P1 are generated in collaborator 1 because collaborator 2 cannot execute task D1. Because alternative D2_P1 can be executed in collaborator 2, Scheduler D2_P1 is generated directly by Scheduler A1_P1. Therefore, the task D1 and D2 can be executed simultaneously at two different sites. A snapshot of this process is shown in Figure 6.4. New EX XML files of processes D1, D1_P1 and D2_P1 are created during execution(Appendix 2.17, 2.18 and 2.21). Because task A1 is a proprietary task, the EX XML file of D2_P1 will not be sent to collaborator 1. However, the EX XML files of processes D1 and D1_P1 are sent to collaborator 2 since D1 is not proprietary to collaborator 2. The XML files of process A1_P1 is updated in the EX

XML file A1_P1_EX.xml (Appendix 2.20). The GRAPH XML files for collaborator 1 and 2 are shown in Appendix 2.19 and 2.21, respectively. All changes from previous files are highlighted.

Suppose that task E and task G are successfully completed. The success of these two tasks leads the alternatives D1_P1 and D2_P1 to be successful, which shows their corresponding logical tasks, the D1 and D2, are successful completely. Suppose that the task D3 is successfully completed under this situation. Thereafter, alternative A2_P1 is applied for logical task A2, and scheduler P generates the scheduler A2_P1 to manage the execution of alternative A2_P1. At the same time, task B1 is ready to be executed (Figure 6.5). The updated EX XML files of process P, D1 and D1_P1 are shown in Appendices 2.23, 2.24 and 2.25, respectively. The new created EX XML file of process A2_P1 is shown in Appendix 2.26.

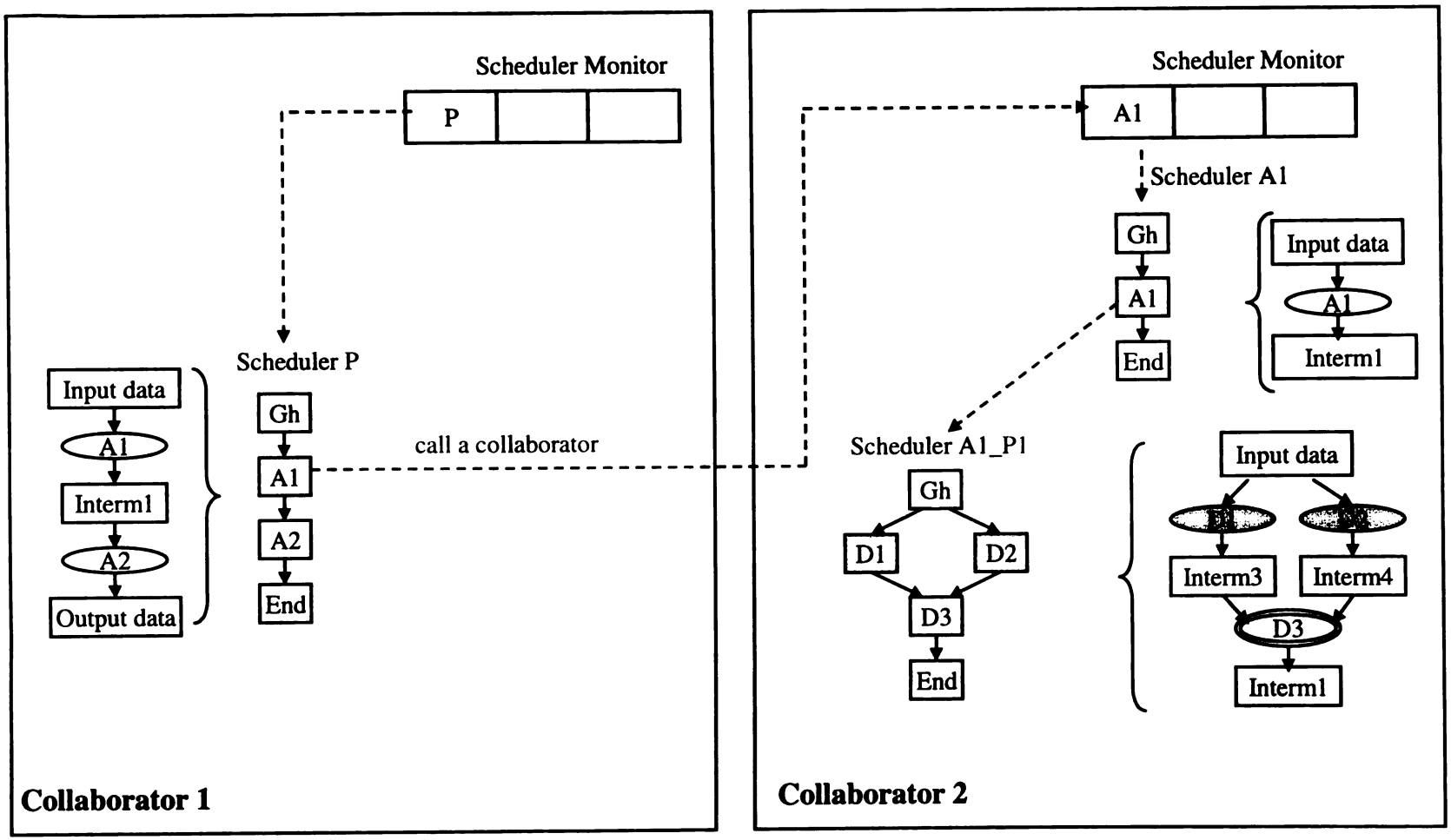

Figure 6.3 The snapshot as production A1_P1 of logical task A1 is applied in Collaborator 2

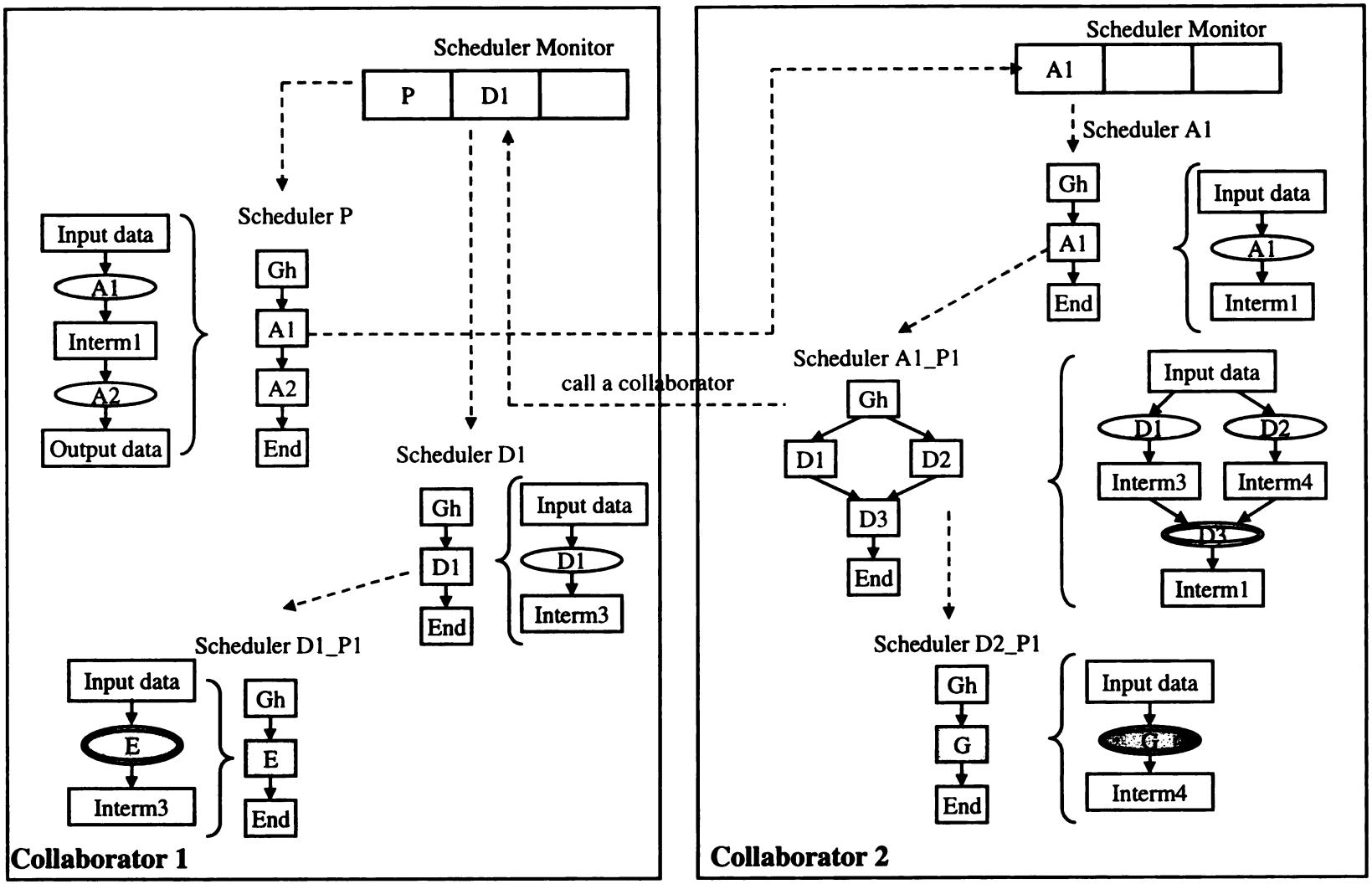

Figure 6.4 The snapshot as production D1_P1 of logical task D1 is applied in Collaborator 1, and production D2_P1 of logical task D2 is applied in Collaborator 2

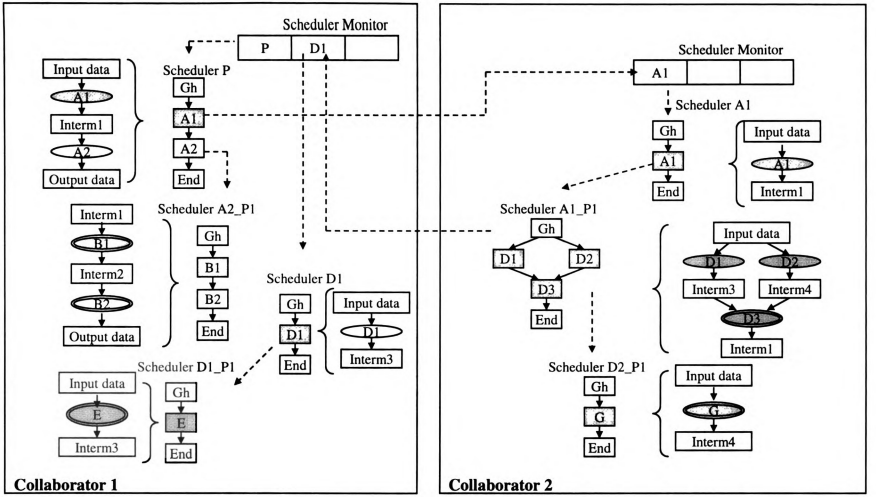

Figure 6.5 The snapshot as production A2_P1 is applied for task A2 after task A1 is finished
(A1 is finished by the finishing of tasks E, G and D3)

The updated EX XML files of processes A1, A1_P1 and D2_P1 are shown in Appendices 2.28, 2.29 and 2.30. The updated GRAPH XML files of collaborator 1 and collaborator 2 are shown in Appendices 2.27 and 2.31, respectively.

Suppose that task B1 is failed (Figure 6.6). Both of the EX XML of process A2_P1 (Appendix 2.32) and the GRAPH XML file in collaborator 1 (Appendix 2.33) are updated accordingly. Because there is no change in EX XML files in collaborator 2, nothing occurs in collaborator 2. Thereafter rollback happens. Because task B1 is an atomic task and first task in alternative A2_P1, it has to rollback to logical task A2 and to apply the other alternative A2_P2. Scheduler A2_P2 is generated (Figure 6.7). Its corresponding EX XML file (Appendix 2.34) is also created. Because alternative A2_P1 is failed by the failure of task B1, the scheduler is deleted after rollback happens to logical task A2. The EX XML file of process P is updated (Appendix 2.35). At this point, task C is ready to execute (Figure 6.7). The GRAPH XML file is updated in collaborator 1 (Appendix 2.36), but nothing happens in collaborator 2.

Suppose task C is failed too (Figure 6.8). The EX XML of process A2_P2 (Appendix 2.37) is updated, and GRAPH XML file in collaborator 1 (Appendix 2.38) is modified. There is no EX XML files in collaborator 2 created or modified, nothing happens in collaborator 2. Thereafter Rollback happens again. Because logical task A2 does not have any more alternatives available, it is failed. At this point, scheduler A2_P2 is deleted (Figure 6.9). The EX XML file of process P and GRAPH XML file are updated in collaborator 1 (Appendix 2.39 and 2.40).

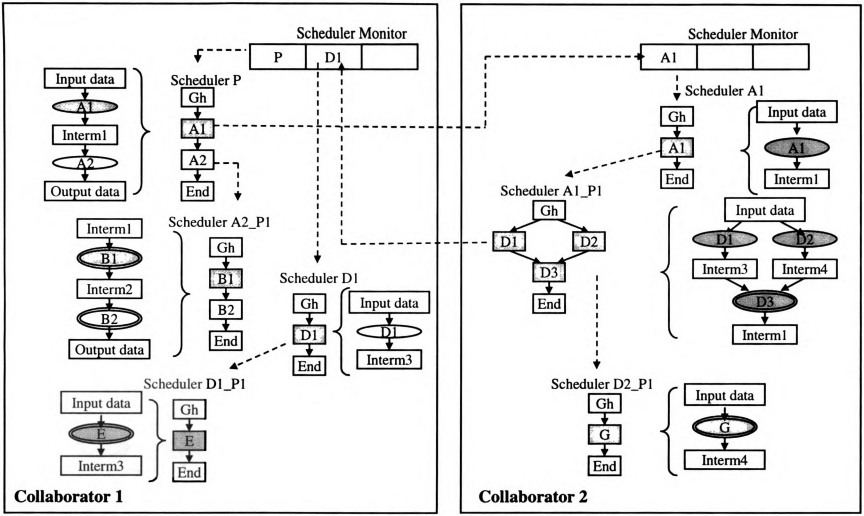

Figure 6.6 The snapshot as task B1 is failed in production A2_P1

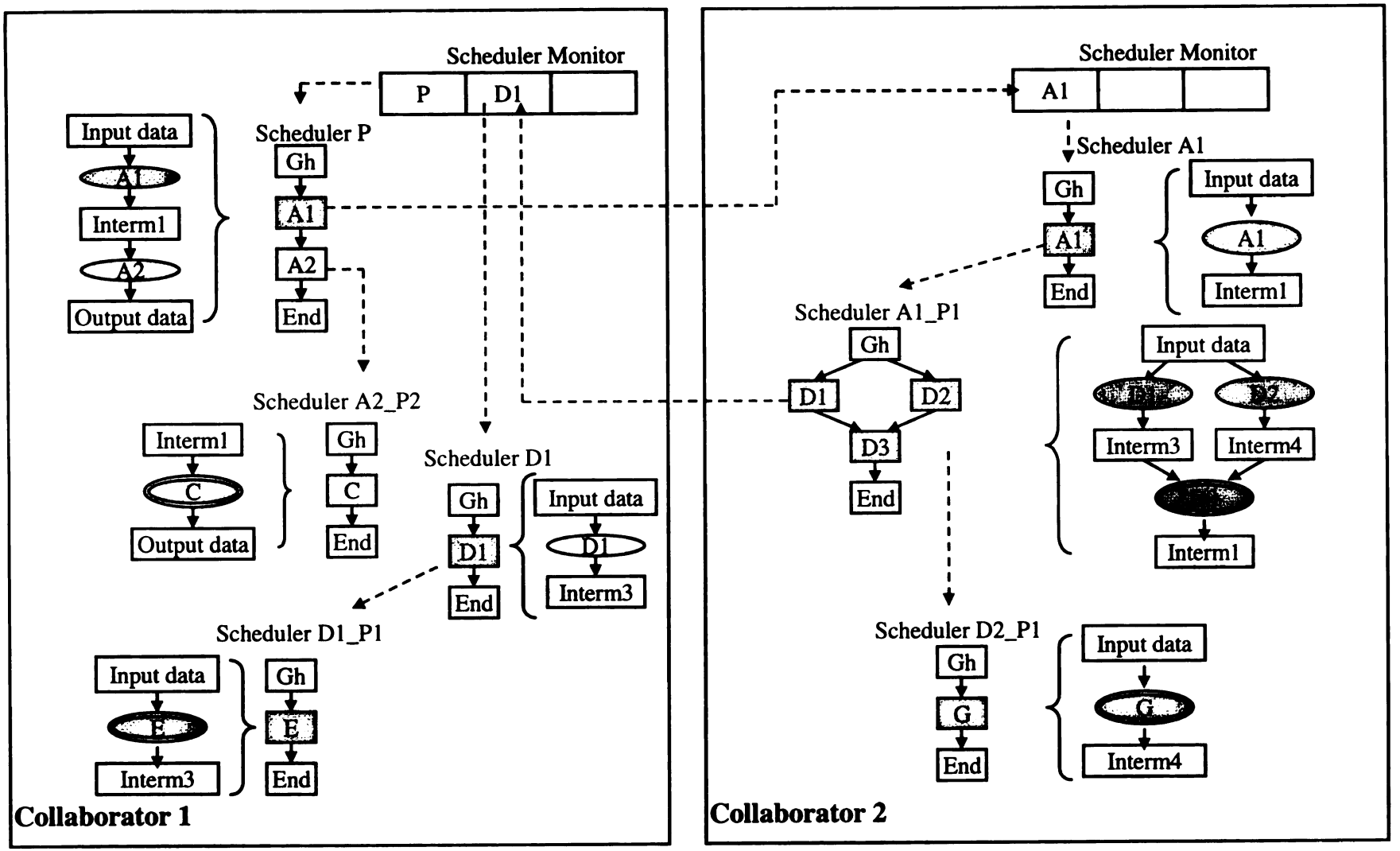

Figure 6.7 The snapshot as production A2_P2 is applied for logical task A2 after rollback happens to task A2

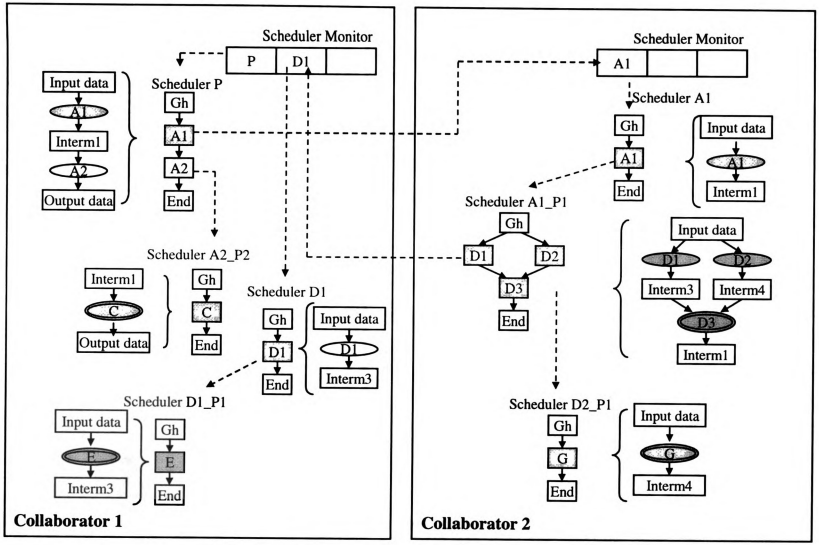

Figure 6.8 The snapshot as task C is failed in production A2_P2

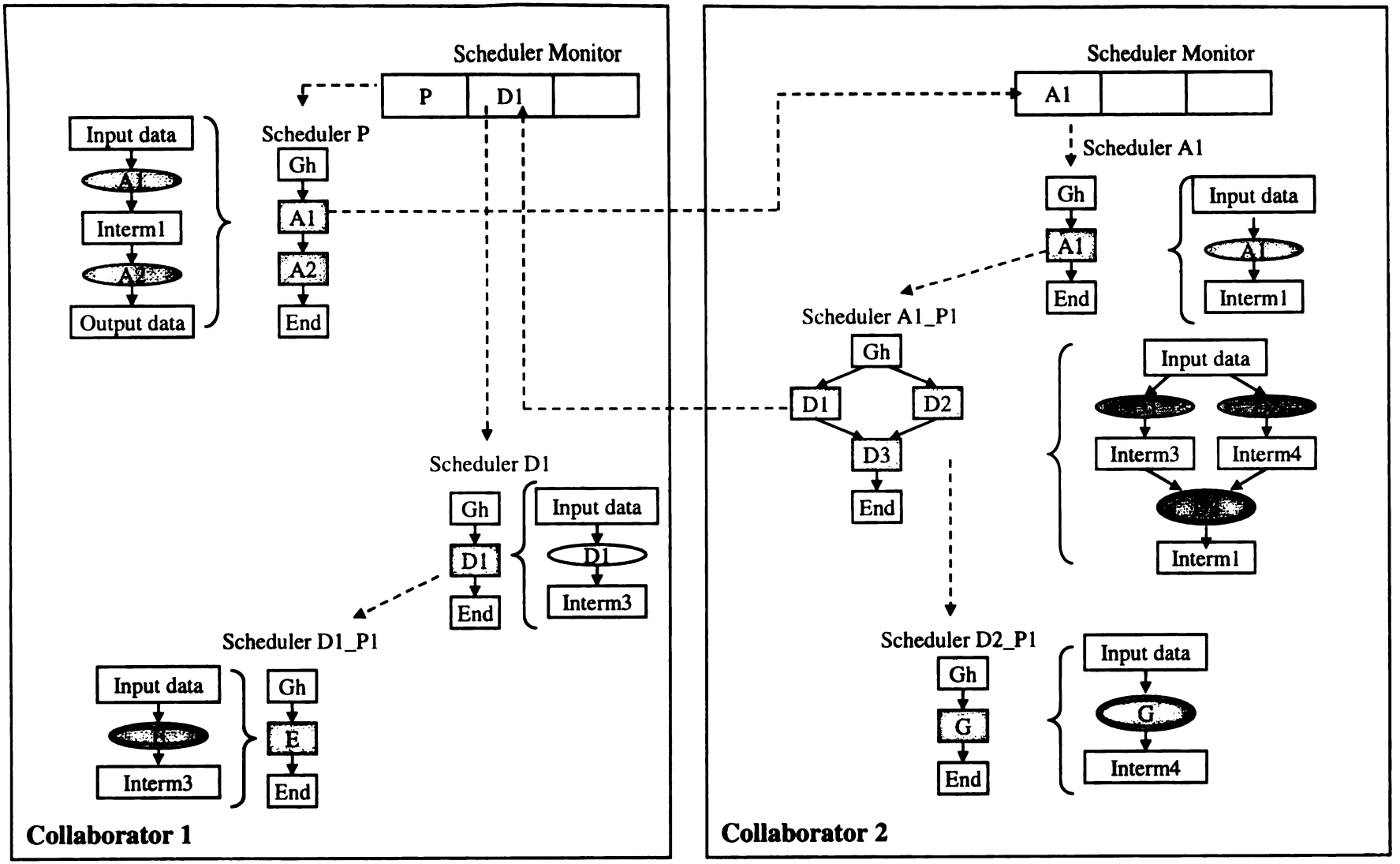

Figure 6.9 the snapshot as task A2 is failed

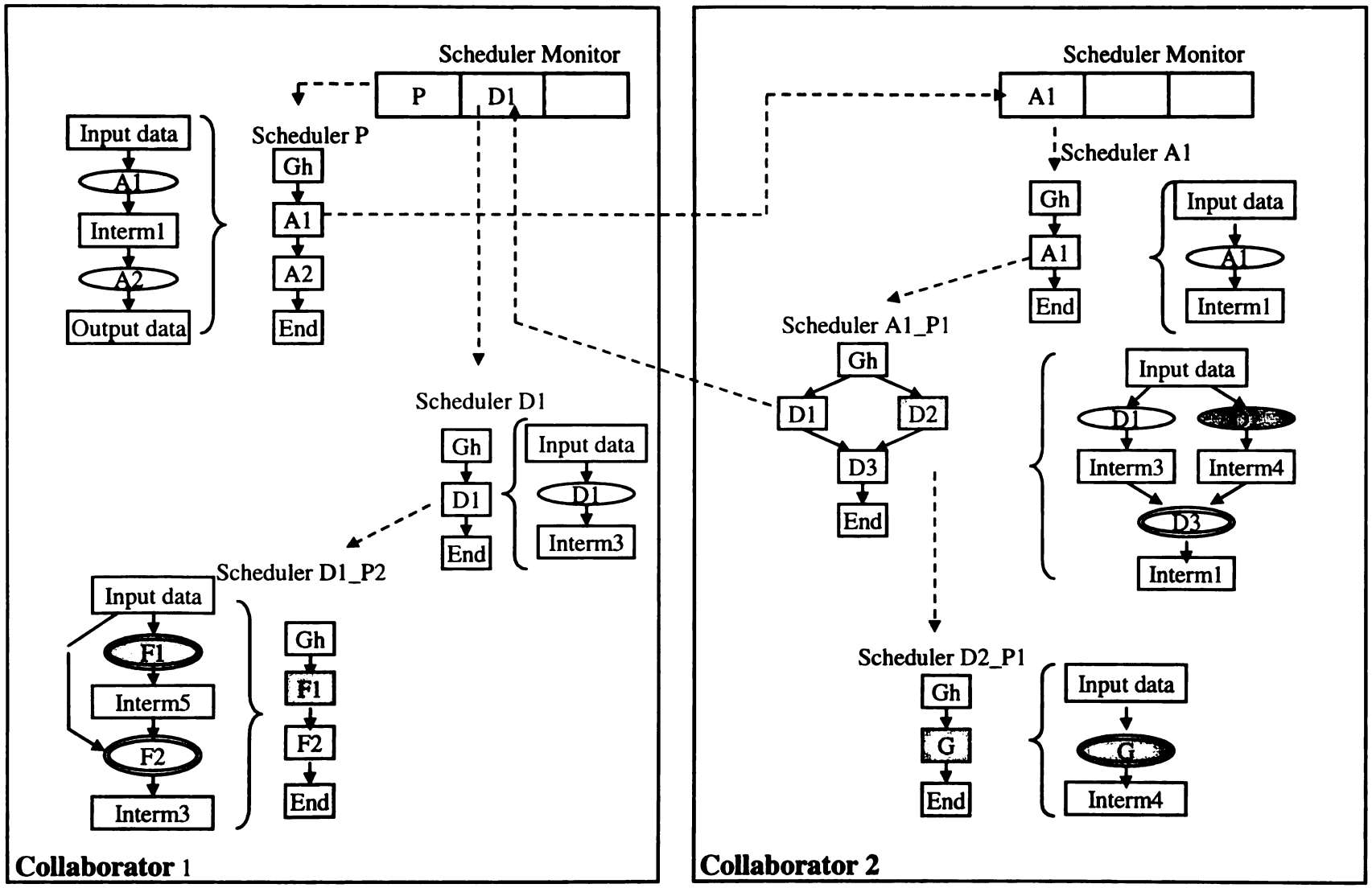

Figure 6.10 the snapshot as rollback happens to task D1

The rollback happens recursively until an alternative can be found to replace the previous one. In this example, alternative D1_P2 replaces alternative D1_P1. Scheduler D1_P1 is deleted, and scheduler D1_P2 is created (Figure 6.10). Therefore, a new EX XML file of process D1_P2 (Appendix 2.43) is created in collaborator 1 and is sent to collaborator 2. The EX XML files of processes P and D1 in collaborator 1 (Appendix 2.41 and 2.42) and processes A1 and A1_P1 in collaborator 2 (Appendix 2.45 and 2.46) are updated. The GRAPH XML files in both collaborator 1 (Appendix 2.44) and collaborator 2 (Appendix 2.47) are updated as well. In order to allow Rollback to continue, schedulers are kept active even after the corresponding executions are completely successfully. Therefore, Scheduler A1_P1 still remains active after successfully completing execution of the flow representing alternative A1_P1 of logical task A1 in this example.

Summary

Advancement of computer technologies makes it possible to revolutionize product manufacturing and process design. MIDAS (Manufacturing Infrastructure and Design Automation System) provides a way to integrate process design, manufacturing and assembly to solve problems caused by isolation of manufacturing from designing in the traditional manufacturing industry. To meet the requirements of collaboration and distribution among different enterprises the MIDAS system architecture has been advanced significantly. In order to provide to the MIDAS system with information that is easy to manage and well-structured, XML files have been employed to store the process information and to exchange information among collaborators. A three level database design technology has been used to graphically design XML Schema through mapping with Unified Modeling Language (UML). XML technologies, such as DOM and Schema, have been utilized to develop the new system and to validate XML files. A new parallel multi-threading algorithm has replaced the previous linear scheduling algorithm in order to manage the process elaboration efficiently.

References

Archibald, George, Karabakal, Nejat, and Karlsson, Paul, "Supply chain vs. supply chain: using simulation to compete beyond the four walls", Proc. Winter Simulation Conf., ACM, Phoenix, 1999, pp. 1207 - 1214

Baldwin, R., and Chung, M. J., 1995, "Design Methodology Management: A Formal Approach", Computer, 28(2), pp. 54–63.

Bayliss, G.M., Bowyer, A., Taylor, R.I., and Willis, P.J., 1994, "Virtual Manufacturing," Proceedings, CSG 94: Set-Theoretic Solid Modeling Techniques and Applications, pp. 353-365

Booch, G., Christerson, M., Fuchs, M. and Koistinen, J. "UML for XML Schema Mapping Specification". Rational Website, December 1999. Available at http://www.rational.com/media/uml/resources/media/uml_xmlschema33.pdf.

Brown, Robert G.,"Driving Digital Manufacturing to Reality", Proc. Winter Simulation Conf., ACM, Orlando, 2000, pp. 224-228

Brugali, D., Menga, G. and Galarraga, S. (1998). "Inter-Company Supply Chains Integration via Mobile Agents". In Jacucci, G. (ed.), Globalization of Manufacturing in the

Carlson, Dave, "Modeling XML Vocabularies with UML", http://www.xml.com/pub/q/all_schema, accessed on Aug. 22, 2001

Christiaanse, Ellen and Kumar, Kuldeep, "ICT-enabled coordination of dynamic supply webs", International Journal of Physical Distribution & Logistics Management, Vol.30, No. 6, 2000, pp 268-285

Curties, W., Kellner, M. I. & Over, J., "Process modeling", Communications of the ACM, 35, 9, pp75-90, September 1992.

Donald, Deidra L.,"A Tutorial on Ergonomic and Process Modeling Using QUEST and IGRIP", Proc. Winter Simulation Conf., ACM, Washington, D.C., 1998, pp. 297-302

Donald, Deidra L., Andreou, Nick, Abell, Jeffrey and Robert J. Schreiber, "The New Design: The Changing Role of Industrial Engineers in the Design Process Through the Use of Simulation", Proc. Winter Simulation Conf., ACM, Phoenix, 1999, pp. 281-285

Freedman, Scott,"An Overview of Fully Integrated Digital Manufacturing Technology", Proc. Winter Simulation Conf., ACM, Phoenix, 1999, pp. 281-285

Fu, Yonghui, Piplani, Rajesh, Souza, Robert de, and Wu, Jingru, "Multi-agent Enabled Modeling and Simulation towards Collaborative Inventory Management in Supply Chain", Proc. Winter Simulation Conf., ACM, Orlando, 2000, pp. 1763-1771

Gaines, B.R., Norrie, D.H. and Lapsley, A.Z., "Mediator: an Intelligent Information System Supporting the Virtual Manufacturing Enterprise", Proc. of IEEE International Conference on Systems, Man and Cybernetics, New York, 1995, pp. 964-969.

Gan, B. P., Turner, S. J. et al., "Distributed supply chain simulation across enterprise boundaries", Proc. Winter Simulation Conf., ACM, Orlando, 2000, pp. 1245-1251

Ganeshan, Ram and Harrison, Terry P., "An Introduction to Supply Chain Management", http://silmaril.smeal.psu.edu/misc/supply_chain_intro.html

Geller, Tracey L., Lammers, Suzanne E., and Mackulak, Gerald T., "Methodology for Simulation Application to Virtual Manufacturing Environments", Proc. Winter Simulation Conf., ACM, Arlington, Virginia, 1995, pp. 909-916

Graham, Gary and Hardaker, Glenn, "Supply-chain management across the Internet", International Journal of Physical Distribution & Logistics Management, Vol.30, No. 3,4, 2000, pp 286-295

Gupta, Anurag, Whitman, Larry and Agarwal, Ramesh K., "Supply Chain Agent Decision Aid System (SCADAS)", Proc. Winter Simulation Conf., ACM, Arlington, Virginia, 2001, pp. 553-559

Hasselbring, Wilhelm, "Information system integration", Communications of the ACM, Vol. 43, Issue 6, June 2000, pp 32-38.

Jngel, Matthias, Kindler, Ekkart, and Weber, Michael. "The Petri Net Markup Language". In Stephan Philippi, editor, 7. Workshop Algorithmen and Werkzeuge fr Petrinetze, pages 47-52, Universitt Koblenz-Landau, German, June 2000. AWPN. http://www.informatik.hu-berlin.de/top/pnml/

Kueng, Peter, "Process Models: a help or burden?", Proc. of the Americas Conference for Information Systems, AIS'97, Indianapolis, 15-17 August, 1997, pp. 676-678

Kumar, Kuldeep, "Technology for supporting supply chain management: Introduction", Communications of the ACM ,Vol. 44, Issue 6, June 2001, pp 58-61

Kusiak, A., Larson, T.N., Wang, J.R., "Reengineering of Design and Manufacturing Process", Computers and Industrial Engineering, Vol.26 No.3 521-536, July 1994
Lakos, Charles, From coloured Petri nets to object Petri nets, In proceedings of the "Application and Theory of Petri Nets 1995", volume 935 of LNCS, pages 278-297, Torino, Italy, June 1995

Lubell, Joshua and Schlenoff, Craig, "Process Representation Using Architectural Forms: Accentuating the Positive", In Proceedings of the Markup Technologies '99 Conference, 1999

Murdoch, John, McDermid, John A., "Modeling Engineering Design Process with Role Activity Diagrams", Transactions of the SDPS Journal of Design and Process Science, Vol.4, No. 2, pp.45-65 June 2000

Ould, M. Business Process: "Modeling and Analysis for Re-engineering and improvement", John Wiley & Sons, Chichester 1995

Plaia, A., Carrie, A., "Application and assessment of IDEF3 – process flow description capture method", International Journal of Operations and Production Management, Vol.15 No.1: 63-73, 1995

Routledge, N., Bird, L. and Goodchild, A., "UML and XML Schema", Proc. of the thirteenth Australasian conference on Database technologies, 2002, Melbourne, Victoria, Australia, vol. 5, pp 157-166

Rudas, I.J., Horvath, L., "Modeling of manufacturing processes using a petri-net representation", Engineering Applications of Artificial Intelligence,10 (3): 243-255 JUN 1997

Saldhana, J., Shatz, S. M., and Hu, Z., "Formalization of Object Behavior and Interactions From UML Models," International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Dec. 2001, Vol. 11, No. 6, pp. 643-673.

Schlenoff, C., Knutilla, A., Ray, S., "Requirments for modeling manufacturing process", http://www.mel.nist.gov/psl/pubs/semiotics/conf-pap.htm (accessed on 09/12/02)

Seligman, L., Rosenthal, A.: "The Impact of XML on Databases and Data Sharing". IEEE Computer, June 2001

Skogan, D., "UML as a schema language for XML based data interchange". In Proceedings of the 2nd International Conference on The Unified Modeling Language (UML'99), 1999. http://www.ifi.uio.no/~davids/papers/Uml2Xml.pdf.

Wang, Alf Inge. "Experience paper: Using XML to implement a workflow tool". In 3rd Annual IASTED International Conference Software Engineering and Applications, Scottsdale, Arizona, USA, 6-8 October 1999.

William E. Riddle, "Fundamental Process Modeling Concepts" http://lsdis.cs.uga.edu/activities/NSF-workflow/FundConc.html (accessed on 09/12/02)

Zha, X.F., "A knowledge intensive multi-agent framework for cooperative/collaborative design modeling and decision support of assemblies", Knowledge-based Systems, 15 (8): 493-506 NOV 2002

Appendix 1

XML Schema of Processes

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang (Michigan State) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="process">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="process_input">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="process_input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="process_output">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="process_output_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="tasks">
               <xs:complexType>
                  <xs:choice maxOccurs="unbounded">
                     <xs:element ref="atomic_task"/>
                     <xs:element ref="logical_task"/>
                     <xs:element ref="selector"/>
                  </xs:choice>
               </xs:complexType>
               <xs:unique name="output_constraint">
                  <xs:selector xpath="*/task_output/data"/>
```

```
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="data">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="atomic_task">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="task_input"/>
      <xs:element ref="task_output"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="logical_task">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="task_input"/>
      <xs:element ref="task_output"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="selector">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="task_input"/>
      <xs:element ref="task_output"/>
      <xs:element name="query">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="database" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

```
    <xs:element name="task_input">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="data" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:key name="input_constraint">
            <xs:selector xpath="data"/>
            <xs:field xpath="@name"/>
        </xs:key>
    </xs:element>
    <xs:element name="task_output">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="data" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="database">
        <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema of Logical Tasks

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang (Michigan State) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="logical_tasks">
      <xs:annotation>
         <xs:documentation>Comment describing your root
element</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="logical_task" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="logical_task">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="task_input">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="task_output">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element ref="pre-eval" minOccurs="0"/>
            <xs:element ref="productions"/>
         </xs:sequence>
```

```
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="data">
        <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="productions">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="process" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="name" type="xs:string" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="pre-eval">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="site" type="xs:anyURI"/>
                <xs:element name="cmdline" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema of Atomic Tasks

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang (Michigan State) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="atomic_tasks">
      <xs:annotation>
         <xs:documentation>Comment describing your root
element</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="atomic_task" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="atomic_task">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="task_input">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="task_output">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element ref="exec"/>
         </xs:sequence>
         <xs:attribute name="name" type="xs:string" use="required"/>
```

```
        </xs:complexType>
    </xs:element>
    <xs:element name="data">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="site" minOccurs="0"/>
                <xs:element ref="path" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="exec">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="site"/>
                <xs:element name="cmdline" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="site" type="xs:anyURI"/>
    <xs:element name="path" type="xs:anyURI"/>
</xs:schema>
```

XML Schema of Selectors

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang (Michigan State) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="selectors">
      <xs:annotation>
         <xs:documentation>Comment describing your root
element</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="selector" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="selector">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="task_input">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                     <xs:element ref="database" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="task_output">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="data" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
               <xs:key name="input_constraint">
                  <xs:selector xpath="data"/>
                  <xs:field xpath="@name"/>
               </xs:key>
            </xs:element>
            <xs:element name="query">
               <xs:complexType>
```

```
                    <xs:sequence>
                        <xs:element ref="database" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="exec"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="data">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="site" minOccurs="0"/>
            <xs:element ref="path" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="exec">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="site"/>
            <xs:element name="cmdline" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="database">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="site"/>
            <xs:element ref="path"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="site" type="xs:anyURI"/>
<xs:element name="path" type="xs:anyURI"/>
</xs:schema>
```

Appendix 2

Appendix 2.1 P.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="P">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm 1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2">
                  <task_input>
                        <data name="Interm 1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.2 A1_P1.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="A1_P1">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="D1">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </logical_task>
            <logical_task name="D2">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </logical_task>
            <atomic_task name="">
                  <task_input>
                        <data name="Interm3"/>
                        <data name="Interm4"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.3 A2_P1.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="A2_P1">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="B1">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Interm2"/>
                  </task_output>
            </atomic_task>
            <atomic_task name="B2">
                  <task_input>
                        <data name="Interm2"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.4 A2_P2.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="A2_P2">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="C">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.5 D1_P1.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="D1_P1">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <atomic_task name="E">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.6 D1_P2.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="D1_P2">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <atomic_task name="F1">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm5"/>
                  </task_output>
            </atomic_task>
            <atomic_task name="F2">
                  <task_input>
                        <data name="Input data"/>
                        <data name="Interm5"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.7 D2_P1.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="D2_P1">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm4"/>
      </process_output>
      <tasks>
            <atomic_task name="G">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.8 D2_P2.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\process.xsd" name="D2_P2">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm4"/>
      </process_output>
      <tasks>
            <atomic_task name="H">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.9 Logical_task.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<logical_tasks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\logical_task.xsd">
      <logical_task name="A1">
            <input>
                  <data name="Input data"/>
            </input>
            <output>
                  <data name="Interm1"/>
            </output>
            <productions>
                  <process name="A1_P1"/>
            </productions>
      </logical_task>
      <logical_task name="A2">
            <input>
                  <data name="Interm1"/>
            </input>
            <output>
                  <data name="Output data"/>
            </output>
            <productions>
                  <process name="A2_P1"/>
                  <process name="A2_P2"/>
            </productions>
      </logical_task>
      <logical_task name="D1">
            <input>
                  <data name="Input data"/>
            </input>
            <output>
                  <data name="Interm3"/>
            </output>
            <productions>
                  <process name="D1_P1"/>
                  <process name="D1_P2"/>
            </productions>
      </logical_task>
      <logical_task name="D2">
            <input>
                  <data name="Input data"/>
            </input>
            <output>
                  <data name="Interm4"/>
            </output>
            <productions>
                  <process name="D2_P1"/>
                  <process name="D2_P2"/>
            </productions>
      </logical_task>
</logical_tasks>
```

Appendix 2.10 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.11 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="A2">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Output data"/>
                        </task_output>
                  </logical_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.12 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.13 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="running">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="A2">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Output data"/>
                        </task_output>
                  </logical_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.14 A1_EX.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
                  <alternative name="A1_P1" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.15 A1_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </logical_task>
            <logical_task name="D2" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </logical_task>
            <atomic_task name="D3">
                  <task_input>
                        <data name="Interm3"/>
                        <data name="Interm4"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.16 GRAPH.XML (collaborator 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="A1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm1"/>
            </process_output>
            <tasks>
                  <logical_task name="D1" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="D2" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm4"/>
                        </task_output>
                  </logical_task>
                  <atomic_task name="D3">
                        <task_input>
                              <data name="Interm3"/>
                              <data name="Interm4"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.17 D1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
                  <alternative name="D1_P1" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.18 D1_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd" name="D1_P1">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <atomic_task name="E" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.19 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="running">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="A2">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Output data"/>
                        </task_output>
                  </logical_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.20 A1_P1_EX.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </logical_task>
            <logical_task name="D2" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
                  <alternative name="D2_P1" status="running"/>
            </logical_task>
            <atomic_task name="D3">
                  <task_input>
                        <data name="Interm3"/>
                        <data name="Interm4"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.21 D2_P1_EX.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D2_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm4"/>
      </process_output>
      <tasks>
            <atomic_task name="G" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.22 GRAPH.XML (collaborator 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="A1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm1"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="G" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm4"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="D3">
                        <task_input>
                              <data name="Interm3"/>
                              <data name="Interm4"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.23 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2" status="running">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
                  <alternative name="A2_P1" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.24 D1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
                  <alternative name="D1_P1" status="succeeded"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.25 D1_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd" name="D1_P1">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <atomic_task name="E" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.26 A2_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A2_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="B1" status="ready">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Interm2"/>
                  </task_output>
            </atomic_task>
            <atomic_task name="B2">
                  <task_input>
                        <data name="Interm2"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.27 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <atomic_task name="B1" status="ready">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="B2">
                        <task_input>
                              <data name="Interm2"/>
                        </task_input>
                        <task_output>
                              <data name="Output data"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
```

```
                </atomic_task>
            </tasks>
        </process>
    </processes>
```

Appendix 2.28 A1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
                  <alternative name="A1_P1" status="succeeded"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.29 A1_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </logical_task>
            <logical_task name="D2" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
                  <alternative name="D2_P1" status="succeeded"/>
            </logical_task>
            <atomic_task name="D3" status="succeeded">
                  <task_input>
                        <data name="Interm3"/>
                        <data name="Interm4"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.30 D2_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D2_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm4"/>
      </process_output>
      <tasks>
            <atomic_task name="G" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.31 GRAPH.XML (collaborator 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="A1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm1"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="G" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm4"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="D3" status="succeeded">
                        <task_input>
                              <data name="Interm3"/>
                              <data name="Interm4"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.32 A2_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A2_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="B1" status="failed">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Interm2"/>
                  </task_output>
            </atomic_task>
            <atomic_task name="B2">
                  <task_input>
                        <data name="Interm2"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.33 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <atomic_task name="B1" status="failed">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="B2">
                        <task_input>
                              <data name="Interm2"/>
                        </task_input>
                        <task_output>
                              <data name="Output data"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
```

```
                </atomic_task>
            </tasks>
        </process>
    </processes>
```

Appendix 2.34 A2_P2_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A2_P2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="C" status="ready">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.35 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2" status="running">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
                  <alternative name="A2_P1" status="failed"/>
                  <alternative name="A2_P2" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.36 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <atomic_task name="C" status="ready">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.37 A2_P2_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A2_P2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Interm1"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <atomic_task name="C" status="failed">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.38 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <atomic_task name="C" status="failed">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.39 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2" status="failed">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
                  <alternative name="A2_P1" status="failed"/>
                  <alternative name="A2_P2" status="failed"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.40 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="A2" status="failed">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </logical_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="E" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```

Appendix 2.41 P_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="P" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Output data"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </logical_task>
            <logical_task name="A2">
                  <task_input>
                        <data name="Interm1"/>
                  </task_input>
                  <task_output>
                        <data name="Output data"/>
                  </task_output>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.42 D1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
                  <alternative name="D1_P1" status="failed"/>
                  <alternative name="D1_P2" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.43 D1_P2_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="D1_P2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm3"/>
      </process_output>
      <tasks>
            <atomic_task name="F1" status="ready">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm5"/>
                  </task_output>
            </atomic_task>
            <atomic_task name="F2">
                  <task_input>
                        <data name="Input data"/>
                        <data name="Interm5"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.44 GRAPH.XML (collaborator 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="P">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Output data"/>
            </process_output>
            <tasks>
                  <logical_task name="A1" status="running">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </logical_task>
                  <logical_task name="A2">
                        <task_input>
                              <data name="Interm1"/>
                        </task_input>
                        <task_output>
                              <data name="Interm2"/>
                        </task_output>
                  </logical_task>
            </tasks>
      </process>
      <process name="D1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm3"/>
            </process_output>
            <tasks>
                  <atomic_task name="F1" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm5"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="F2">
                        <task_input>
                              <data name="Input data"/>
                              <data name="Interm5"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
```

```
                    </task_output>
                </atomic_task>
            </tasks>
        </process>
    </processes>
```

Appendix 2.45 A1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="A1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
                  <alternative name="A1_P1" status="running"/>
            </logical_task>
      </tasks>
</process>
```

Appendix 2.46 A1_P1_EX.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<process name="A1_P1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\EX_process.xsd">
      <process_input>
            <data name="Input data"/>
      </process_input>
      <process_output>
            <data name="Interm1"/>
      </process_output>
      <tasks>
            <logical_task name="D1" status="running">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm3"/>
                  </task_output>
            </logical_task>
            <logical_task name="D2" status="succeeded">
                  <task_input>
                        <data name="Input data"/>
                  </task_input>
                  <task_output>
                        <data name="Interm4"/>
                  </task_output>
                  <alternative name="D2_P1" status="succeeded"/>
            </logical_task>
            <atomic_task name="D3">
                  <task_input>
                        <data name="Interm3"/>
                        <data name="Interm4"/>
                  </task_input>
                  <task_output>
                        <data name="Interm1"/>
                  </task_output>
            </atomic_task>
      </tasks>
</process>
```

Appendix 2.47 GRAPH.XML (collaborator 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by fang
(Michigan State) -->
<processes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\qinyongg\thesis\midas_modeling\schema
\GRAPH_process.xsd">
      <process name="A1">
            <process_input>
                  <data name="Input data"/>
            </process_input>
            <process_output>
                  <data name="Interm1"/>
            </process_output>
            <tasks>
                  <atomic_task name="F1" status="ready">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm5"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="F2">
                        <task_input>
                              <data name="Interm5"/>
                        </task_input>
                        <task_output>
                              <data name="Interm3"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="G" status="succeeded">
                        <task_input>
                              <data name="Input data"/>
                        </task_input>
                        <task_output>
                              <data name="Interm4"/>
                        </task_output>
                  </atomic_task>
                  <atomic_task name="D3">
                        <task_input>
                              <data name="Interm3"/>
                              <data name="Interm4"/>
                        </task_input>
                        <task_output>
                              <data name="Interm1"/>
                        </task_output>
                  </atomic_task>
            </tasks>
      </process>
</processes>
```