MARIFlow
A Workflow Management System for Maritime Industry

Asuman Dogac, Catriel Beeri[*]*, Arif Tumer, Murat Ezbiderli, Nesime Tatbul, Cengiz Icdem, Guray Erus, Orhan Cetinkaya and Necip Hamali*

Software Research and Development Center, Faculty of Engineering
Middle East Technical University (METU), 06531 Ankara Turkiye
asuman@srdc.metu.edu.tr

[*] Institute of Computer Science, Hebrew University
91904 Jerusalem Israel
beeri@cs.huji.ac.il

Abstract

The aim of MARIFlow Project is to provide a prototype of an architecture for automating and monitoring the flow of control and data over the Internet among different organisations. This "electronic medium", capable of delivering value-added services to the participants, encompasses many different technological areas: from communication to security, databases, transaction support and agents. The project will make use of these technologies to produce a workflow management system. In particular, the goal of the project is to develop an adaptable workflow engine through which the activities of the different participants in the maritime industry can be harmonised, combined, and expanded through better tracking of functional dependencies and documents, improved data access and handling, and lower administrative overheads.

The MARIFlow system is based on *data-centric* approach, that is, execution of activities on a host machine is triggered by arrival of data, and generates further data that is sent to participants in the process, where upon arrival may trigger further activities. Nevertheless, the important characteristic of such processes is distribution, not just in terms of geography, but also in terms of ownership, responsibility and autonomy. This paper also addresses the important issues such as security of the documents as well as the tracking of data and documents, for monitoring purposes.

1 Introduction

Workflow systems, in general, provide for declarative means for specifying the control flow among activities and extensive research and development, both in the academia and in the industry have contributed to various aspects of the workflow system in making a mature technology - Alonso (1995), Georgakopoulos (1995), Miller (1997), Ming Shan (1998), Dayal (1998), Muth (1998), Sheth (1998), Vossen (1998), Alonso (1998), Cichocki and Rusinkiewicz (1998), and Dogac (1998). However in most of the systems, data flow is restricted to the parameters of the involved activities often disconnected from the description of the flow itself. In other words, the workflow management systems as used by industry today, use a process centric approach. Thus, they lack a mechanism with which it is possible to define the source of data, control its flow over the net, and identify and possibly invoke the activities that make use of it. The workflow engine to be developed as part of this project will address these crucial issues with a special emphasis on the ease of use, maintenance, and customization to the needs of maritime industry.

In maritime industry, materials used in shipbuilding or repairs need to be certified by a classification society. In current practice, the material is checked while it is at the production plant and the "quality data", related to this material, is delivered to the classification society. If the quality data fulfills the requirements, a paper certificate is issued and delivered to the production plant as well as to the customer. Once issued the certificate follows the material to the main shipyard or to one of the subcontractors from where it is eventually added to the ship's documentation file. The certificate is checked at every production stage as well as at ship's handover and at each survey during ship's life cycle. In an industry involving the flow of large amount of paper documents among different organizations, this is a slow, expensive, tedious, error-prone and very limiting process, which in some cases can hinder the ability to improve the service quality. In this work we describe an architecture that provides for automating and monitoring the flow of control and data over the Internet among different organizations and companies with special attention to maritime industry.

In the MARIFlow system, the higher order process is defined through a graphical user interface, which is then mapped to a textual language called FlowDL. FlowDL allows indicating the source of the documents, their control flow and the activities that make use of these documents. A process definition in FlowDL is executed through co-operating agents, called MARCAs (MARIFlow Co-operating Agents), that are automatically initialized at each site that the process executes. The initiation of agents and monitoring facilities are managed through Java programs which can be used by authorized users through Web. The main Web page through which the tools can be started is shown in Figure 1.

MARCAs are responsible for handling the activities at their sites, routing the documents in electronic form, according to the process description among other MARCAs, keeping track of process information by logging activities, and providing for the security and authentication of documents during communication.

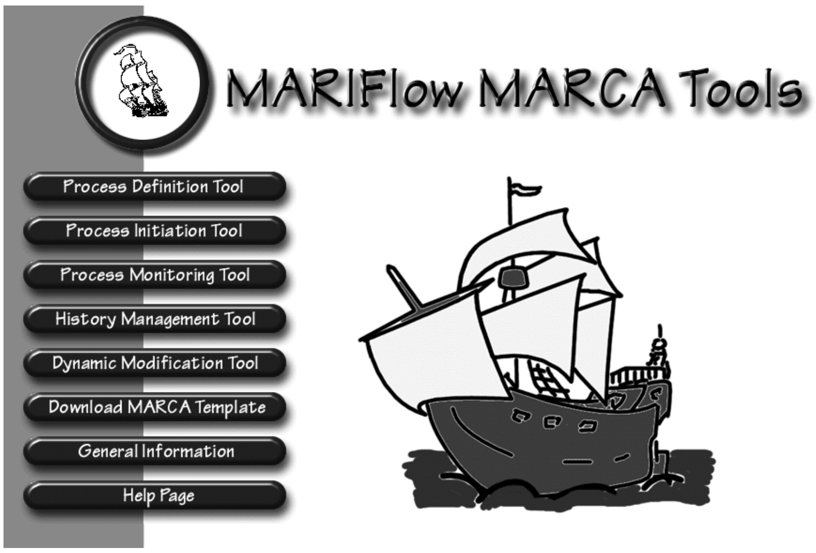


Figure 1. MARIFlow MARCA Tools Web Page

The overall system should be immune to failures, and the processing power should be distributed such that it will not generate a serious bottleneck built around a single machine, which is accessed extensively. The participants of the workflow system should decide and behave on their own rather than being invoked or commanded by other programs in a centralized fashion. These requirements necessitate an agent-based architecture for the workflow system, where independent entities capable of completing complex assignments without intervention are used rather than tools that must be manipulated by a user.

In the MARIFlow system, the responsibilities of MARCAs in order to support the requirements given above are defined as follows:

- A MARCA receives messages through a persistent queue and evaluates them to decide what action to take. The persistent queue is necessary so that the agent does not lose its state after a program crash, site failure etc.
- It persistently stores the documents it receives. If the organization that the MARCA resides on has a firewall mechanism, it is also MARCA's responsibility to pass the documents to the in-house system, get the

resulting documents from the system and forward them to the related agents as specified in the process definition.

- Process related information should be stored persistently for further monitoring purposes. Therefore it is MARCA's responsibility to direct related information to a database through its JDBC interface.
- A process definition is compiled at a host and through a special MARCA the information is distributed over the system to other MARCAs necessary for the given workflow definition at initialization phase. That special MARCA is also responsible for data warehousing for monitoring purposes.

This paper is organized as follows: In Section 2, the general architecture of the system is described. This section introduces the MARIFlow Cooperating Agents (MARCAs), the FlowDL workflow definition language, monitoring of the workflow processes along with an example definition for maritime industry to illustrate to details of the architecture. Section 3 describes how security and authentication of documents are handled in MARIFlow. The availability and scalability issues are discussed in Section 4. In Section 5, the work that remains to be done is described namely the persistency of the messages to recover from failures and compensation of activities. Finally, Section 6 concludes the paper.

2 The Architecture of the System

2.1 An Overview of the Architecture

Figure 2 gives an overview of the general architecture of the MARIFlow system. Each organization may have in-house applications inside a firewall protected from unauthorized access. MARCAs exist on a host machine outside the firewall relative to the site network. The MARIFlow agent informs in-house applications when necessary through internal process initiator and is responsible for sending and receiving documents and process related information through the firewall mechanism.

In MARIFlow an inter-enterprise workflow is defined graphically where the workflow designer specifies domains, tasks and process information which are used in building the process definition. The graphical representation of the process definition is mapped to FlowDL, the language used in MARIFlow, and from that definition the specifications for each MARCA is generated. The compilation of the process definition is done on the coordinating MARCA, which is installed in one of the sites. The behavioral structure, obtained from the process definition, for each MARCA is then transmitted to corresponding agent and MARCAs become available for distributed workflow management. It should be noted that this transmission along with the communication at instance level should be realized through persistent queues in order to survive through system crashes and other problems.

2.2 MARIFlow Cooperating Agents: MARCAs

A workflow instance in MARIFlow is executed by co-operating agents called MARCAs. There is exactly one MARCA at each site participating to the workflow execution and it handles all the activities running at its site. At initialization, once in their life time, the sites download the generic MARCA template from a given URL. At compilation time the guards of activities within the responsibility of a MARCA are determined according to the process definition. Guards are special mechanisms based on intertask dependencies - Attie (1993), Singh (1996). They are logical expressions for significant events of activities of a MARCA like "start" or "terminate". MARCAs evaluate these guards with the messages that they receive to decide on their actions. In other words, guards inform the MARCA when to execute a certain activity. A detailed formal description of obtaining guard expressions from a given workflow specification is given in Dogac (1998). All of the information about guard structures is obtained from the process definition at the initialization phase of the MARCA.

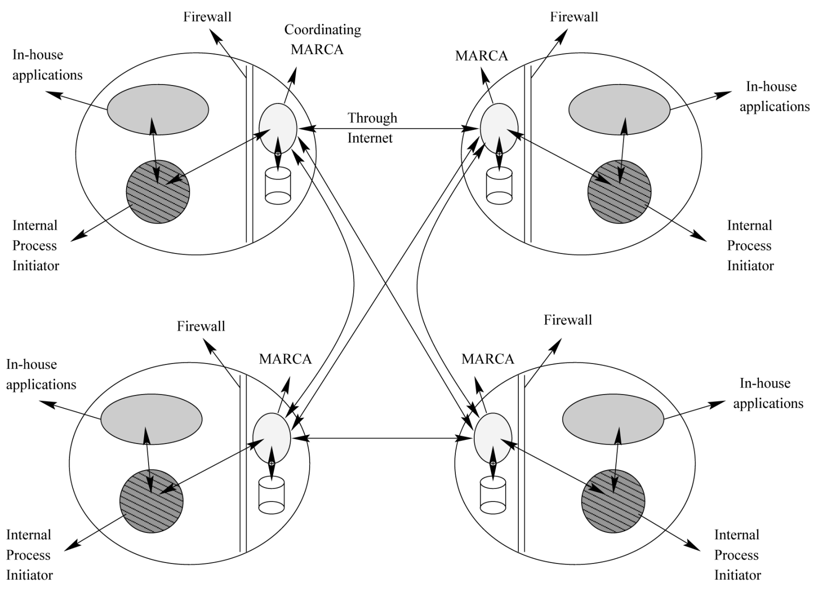
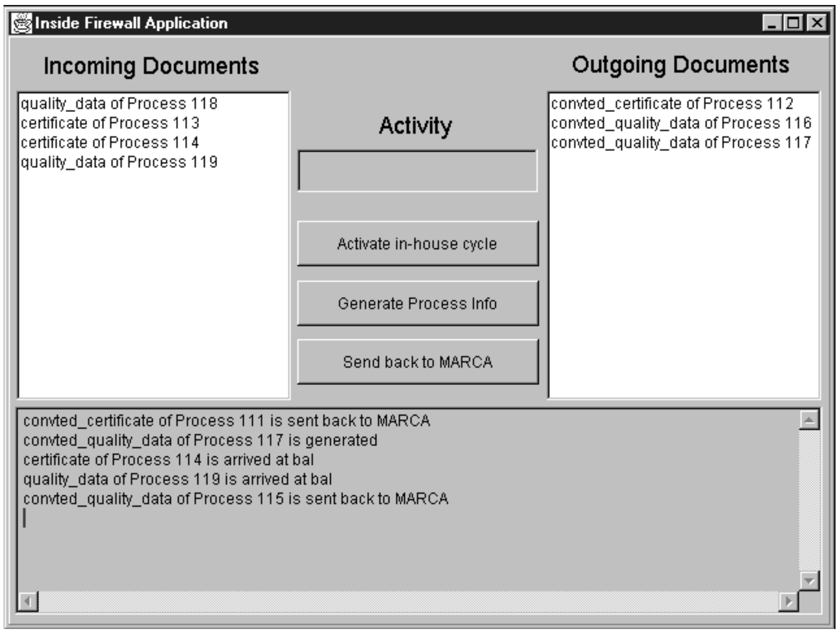


Figure 2. An Overview of the Architecture

There is a special MARCA in the workflow system called the coordinating MARCA. A workflow definition is realized through the coordinating

MARCA and compiled only once. During this compilation, guard structure for each MARCA participating the system along with the interactions with the other MARCAs are obtained and the MARCAs are initialized with this information by the coordinating MARCA. A single MARCA residing on a host is capable of handling multiple workflow definitions and multiple instances of a given workflow at a certain time. All messages are differentiated by unique workflow id obtained from the definition and instance ids that are automatically assigned by the system for each instance generated.

MARIFlow agents communicate with each other through TCP/IP over the Internet. Network concurrent accesses to a single port is handled by Java's Net Package by assigning dynamic ports for each request. Simple message buffering and queuing are also provided by this package. These queuing facilities have been extended to persistent queue implementation and transactional agent communication for safe and consistent transmission.



```
quality_data of Process 118
certificate of Process 113
certificate of Process 114
quality_data of Process 119
```

```
convted_certificate of Process 112
convted_quality_data of Process 116
convted_quality_data of Process 117
```

```
convted_certificate of Process 111 is sent back to MARCA
convted_quality_data of Process 117 is generated
certificate of Process 114 is arrived at bal
quality_data of Process 119 is arrived at bal
convted_quality_data of Process 115 is sent back to MARCA
```

Figure 3. Graphical User Interface for Process Control inside the Firewall

MARCAs communicate with each other through an Agent Communication Language, which is specific to MARIFlow agents, hence they do not communicate with agents in the outside world. This is because communicating with outside world agents is not necessary within the scope of the work. During a session, each message is preceded by an activity identifier along with workflow and instance ids. This information is necessary to identify how a received document or message should be managed according to the activity identifier.

A MARCA sends a document and process related information inside the firewall through an e-mail. Since the content of the documents may be binary, they are encoded prior to the attachment by traditional mime base64 encoding used in e-mail messages.

A program inside the firewall processes the incoming mail and extracts the documents, process related information, workflow and instance ids, the activity that will use this document inside the firewall and displays this information through a GUI as shown in Figure 3. When a document arrives through an e-mail it is shown in the *Incoming Documents* list of the interface. When a document is highlighted (i.e. selected), the activity that will use this document appears in the *Activity* box. The *Activate in-house Cycle* button may be used to start the given activity provided that API of the activity is available. *Send back to MARCA* button is used to transfer the document selected in the *Outgoing Documents* list to the MARCA outside if the related information form of the document is filled via the interface displayed when *Generate Process Info* button is used.

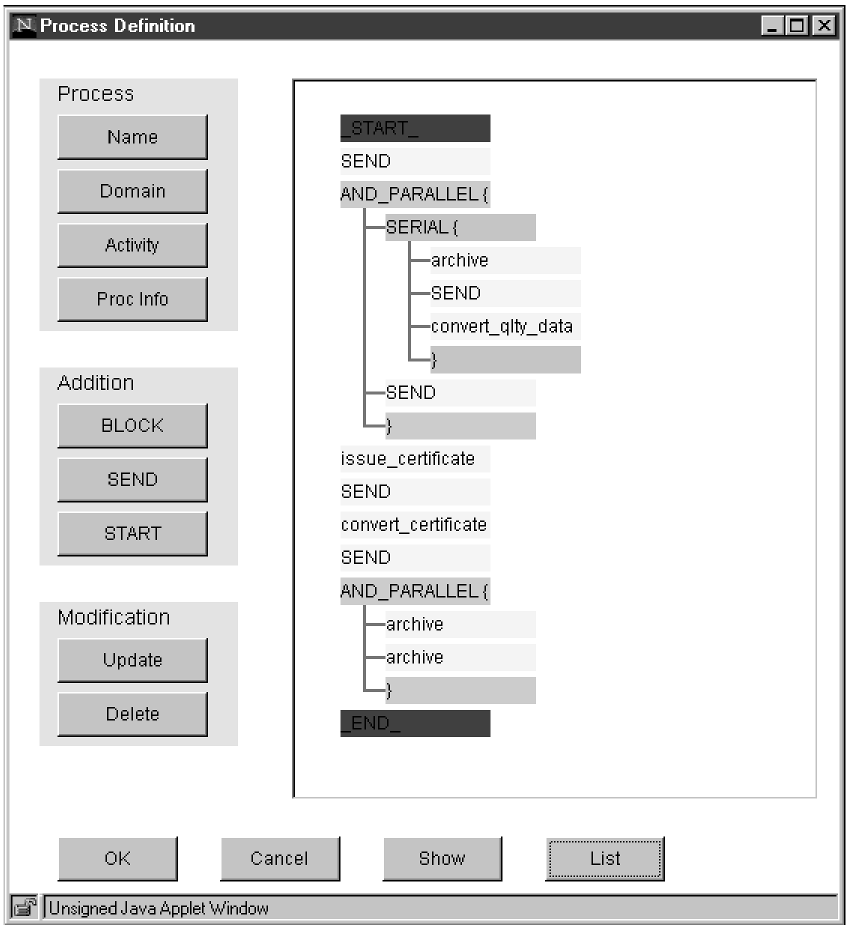
We have chosen to send the response back to MARCA through e-mail mechanism to be system independent as much as possible, since some firewall architectures disallow packet transmission in both directions. The reader program for the mailbox is based upon POP3 and receives the messages through the POP3 server port. This choice is also an effort to be system independent in MARIFlow since POP3 is independent of the structure of the mailbox in different operating system implementations.

2.3 Workflow Definition Language: FlowDL

In MARIFlow system, a workflow process definition is given in FlowDL as a collection of blocks, tasks and other sub-processes as well as some explicit declarations and commands to specify Internet domain addresses, sources of documents, process specific information to be used for monitoring the document flow, and activities for further processing on the documents. The term *activity* is used to refer to a block, a task, or a (sub)process.

FlowDL contains several kinds of blocks, which are used to define different kinds of flow types in the process definition: the activities that run in parallel, in serial or under conditions etc. These blocks along with the declarations done in the process definition code define the whole workflow system. The blocks types encapsulate the workflow primitives defined in Hollingsworth (1996), which are sequential, AND-split, AND-join, OR-split, OR-join and repeatable task.

In MARIFlow, an inter enterprise workflow is defined graphically by using the tool as shown in Figure 4. This tool allows the workflow designer to make the declarations, specify domains, tasks and process information. Afterwards, different types of blocks can be generated and added to the workflow by using the declarations in order to preserve consistency. The workflow definition is mapped to the textual FlowDL language. When this definition is parsed, the guards of each MARCA participating the system are generated and the agents are initialized with this information. The guards provide for the behavioral definitions of the MARCAs.



```
_START_
SEND
AND_PARALLEL {
  SERIAL {
    archive
    SEND
    convert_qlty_data
    }
  SEND
  }
issue_certificate
SEND
convert_certificate
SEND
AND_PARALLEL {
  archive
  archive
  }
_END_
```

Figure 4. Graphical User Interface for Process Definition in MARIFlow

The advantages brought by block structured FlowDL language can be summarized as follows:

- The current workflow specification languages are unstructured and/or rule based, as noted in Sheth (1996). Unstructured languages make debugging/testing of a complex workflow difficult and rule based ones become inefficient when they are used for specification of large and complex workflow processes due to the large number of rules and overhead associated with rule management. FlowDL avoids these disadvantages through its block-structured nature.

- A block structured language confines the inter-task dependencies to a well-formed structure that in turn proves extremely helpful in generating the guards of activities for distributed scheduling of a workflow.
- Blocks not only clearly define the data and control dependencies among tasks but also present a well-defined recovery semantics through compensation for a failed or aborted block.

In addition to activities, there are also assignment statements in FlowDL which access and update workflow relevant data. In this way the workflow designer has the ability to assign values to variables. Note that these variables are used in conditional and loop blocks.

2.4 Monitoring of Workflow Processes

Each MARCA stores all the messages it receives in a persistent log so that after a crash or a site failure the MARCA can be brought back to a consistent state by using the information in the log. The MARCAs also store additional process related information specified by the workflow designer through FlowDL inside the database system. Since Java is used in coding the MARIFlow system, a Java native JDBC interface is used for database connectivity. Consequently any database with a JDBC interface can be used by the MARCA.

Each MARCA sends a copy of the information it stores to the coordinating MARCA. Thus coordinating MARCA constitutes a data warehouse site for monitoring information. This site is available to any authorized user on the Internet through a Web interface and may further be replicated for availability purposes.

The authorized user can track the flow of a process instance through a graphical user interface as shown in Figure 5 by giving the process instance identifier and the workflow definition it belongs to. The Web interface reads the process definition of the selected workflow from the database, and produces a graphical representation from that information, with different colors for each block type and with lines connecting the blocks showing the flow of data and messages. Also the interface reads the instance information from the coordinating MARCA’s database and reflects it on the graph so that the user can see the instant state of the process instance on the screen, along with the finished, on going and yet to be started activities.

The information kept in the database system of the coordinating MARCA can be queried through Java Applets directly from the Web.

The advantages of this monitoring architecture are as follows:

- It provides for high availability since the data is stored both in the MARCAs locally in a distributed manner and also in a data warehouse in a centralized manner.
- Authorized user can query the data warehouse from anywhere on the Internet by using a simple Web browser.
- Response to monitoring queries will be fast since data is obtained from a single store rather than performing distributed query processing.

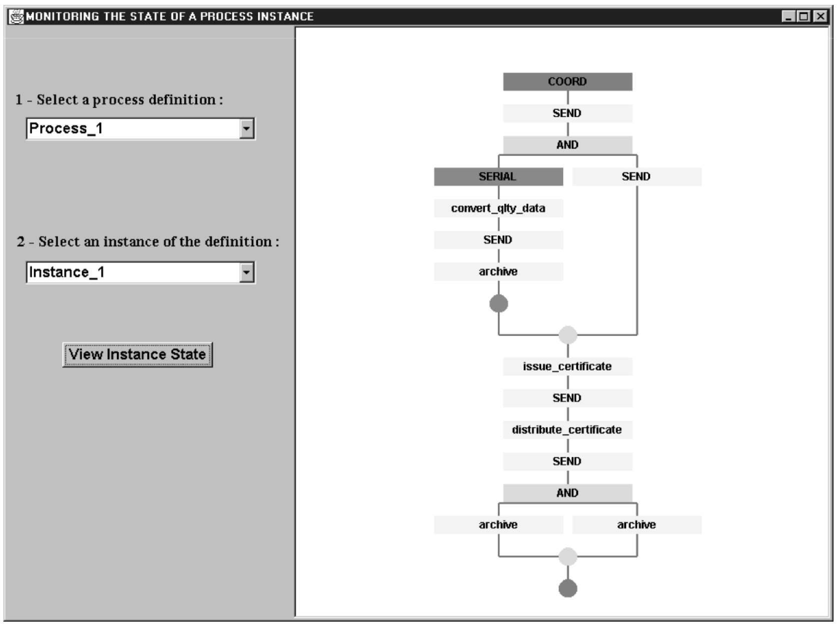


Figure 5. Graphical Monitoring Interface of the MARIFlow System

The databases of each MARCA can also be queried locally by authorized users.

2.5 The **Certification** *Process Definition*

The following is an example workflow defined in FlowDL reflecting a business scenario for a certification process in maritime industry:

```
PROCESS Certification ();

ACTIVITY archive (IN document arch_document);
ACTIVITY convert_qlty_data (IN document quality_data
          OUT document convted_q_d);
```

```
ACTIVITY convert_certificate(IN document certificate
        OUT document convted_cert);
ACTIVITY issue_certificate (IN document conv_quality_data
        IN document product_spec OUT document certificate);
ACTIVITY delete_from_archive();
ACTIVITY cancel_converted_data();
ACTIVITY cancel_certificate();

DOMAIN_DEFINITION {
        salzgitter_ag.de        szag;
        germanlloyd.org         gl;
        balance_bremen.de       bal;
        isisanisi.com           isisan;
}

struct process_info {
        string  order_no;
        string  material_no;
        string  customer_name;
        string  supplier_name;
        string  class_society_name;
        string  certificate_number;
}

DEFINE_PROCESS MARIFlow ()
{
        szag SENDS (quality_data) TO bal GENERATES (order_no
                material_no customer_name supplier_name
                class_society_name);
        AND_PARALLEL {
                SERIAL {
                        START convert_qlty_data (IN quality_data OUT
                                convted_q_d) AT bal COMPENSATED BY
                                cancel_converted_data();
                        bal SENDS (convted_q_d) TO gl AND isisan;
                        START archive (IN convted_q_d) AT isisan NON_VITAL
                                COMPENSATED BY delete_from_archive();
                }
                isisan SENDS  (prod_spec) TO gl;
        }
        START issue_certificate (IN convted_q_d IN prod_spec OUT
                certificate) AT gl GENERATES (certificate_number)
                COMPENSATED BY cancel_certificate();
        gl SENDS (certificate) TO bal;
        START convert_certificate (IN certificate OUT convted_cert)
                AT bal;
        bal SENDS (convted_cert) TO szag AND isisan;
        AND_PARALLEL {
                START archive (IN convted_cert) AT szag NON_VITAL
                        COMPENSATED BY delete_from_archive();
                START archive (IN convted_cert) AT isisan NON_VITAL
                        COMPENSATED BY delete_from_archive();
        }
}
```

Example 1. An Example Workflow Definition for the Maritime Industry

The process starts when the steel company (*szag*) sends the "quality data" to a service company (*bal*) to be transformed into EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) standard. Then within the scope of a block the following activities run in parallel:

- At *bal*, "quality_data" is converted into EDIFACT standard by invoking "convert_qlty_data" activity. The document produced, "convted_q_d", is sent to *gl* and *isisan*. At *isisan*, the arrived document is archived by invoking *archive* activity. Note that the block that comprises these activities is a SERIAL block where the items in the block are executed sequentially.
- The steel user (*isisan*) sends the product specification document to the classification society (*gl*).

Once *gl* receives the converted quality data, its system is notified to start the "issue_certificate" process. When the issued certificate, "certificate", is generated and delivered to the MARCA outside the firewall, it is transferred to *bal* again for conversion to EDIFACT standard and this document is sent to *szag* and *isisan*. Afterwards "archive" activities at *isisan* and *szag* are started and executed in parallel for converted certificate, "convted_cert". This completes the cycle of one instance in *Certification* process definition.

The guard structures for each MARCA is generated when the process definition is parsed and process tree is created as in Figure 6. The guards evaluate to true when the necessary messages and/or documents arrive at MARCAs either from the network or from the inside firewall application. Hence the necessary action is taken such as an activity is started or the execution through the blocks continues. In the running example, when "prod_spec" arrives at *gl* the guard still evaluates to false since for the guard to evaluate to true the arrival of document "convted_q_d" is also necessary. Therefore, only when both of the documents arrive at *gl* the guard evaluates to true and the necessary course of action is taken, that is, the "issue_certificate" task is invoked, to create the "certificate".

For monitoring purposes, each MARCA stores the following process related information persistently in a database system: order_no, material_no, customer_name, supplier_name, class_society_name and certificate_number. Note that the activity responsible for providing a specific piece of information, such as order_no, or certificate_number is obtained through the GENERATES statement of the activities. For example the activity "START issue_certificate (IN convted_q_d IN prod_spec OUT certificate) AT gl GENERATES (certificate_number);" declares that certificate_number will be produced by this activity.

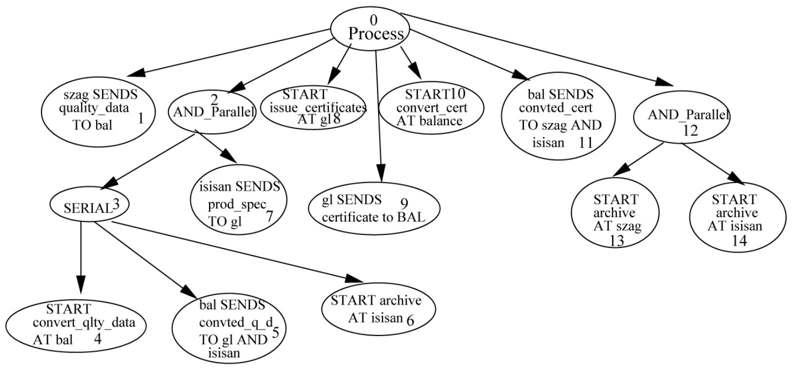

0 Process
szag SENDS quality_data TO bal 1
2 AND_Parallel
START issue_certificate AT gl 8
START 10 convert_cert AT balance
bal SENDS convted_cert TO szag AND isisan 11
AND_Parallel 12
SERIAL 3
isisan SENDS prod_spec TO gl 7
gl SENDS certificate to BAL 9
START archive AT szag 13
START archive AT isisan 14
START convert_qlty_data AT bal 4
bal SENDS convted_q_d TO gl AND isisan 5
START archive AT isisan 6

Figure 6. Process Tree of the Example Process Definition

Some example queries that can be readily answered by the database system include: Status of a certificate given its number, status of all certificates for a given steel order, number of certificates issued by the classification society etc. These queries can be executed through the User Interface specialized for this process definition, using any World Wide Web Browser capable of running Java Applets. Figure 7 depicts a customized GUI available for the MARIFlow project.

3 Authentication and Authorisation

Conducting business over a public network, like the Internet, requires mechanisms for authentication of messages, for authorization to access data and execute operations, and for privacy and security in general. We discuss briefly some of the requirements in our application domain, and how they relate to our software architecture.

Clearly, activities that are performed inside a company’s firewall are irrelevant to our scenario; a company may use whatever mechanisms it chooses. Specific requirements arise with respect to company-to-company operation, to distributed intra-company operation, and to the overall operation of the MARCA network. We consider each in turn.

An example of a company-to-company operation is sending a quality document from a manufacturer to a classification company. In our scenario, this involves three companies, since a service company translates the document before being sent to the classification society. When a set of companies are involved in a well-defined business procedure, they naturally want the details to be protected from potential competitors. The obvious solution is to encrypt documents inside the firewall, using a scheme that is common to all involved parties. Such encryption also provides authentication of the sender of a message that can easily be verified by the receiver. Finally, an unauthorized receiver will not be able to read a message.

A common situation in our scenario is that representatives of the classification society are present in manufacturing sites, as part of the certification process. These representatives need access to documents of materials and parts, from their society's repository. Although this is an intra-company operation, it is similar in nature to the previous case. An intra-company scheme can be used to encrypt documents before they are passed outside the firewall. An encryption of the request for a document from a field agent can provide authentication, as well as privacy, since the message contents, namely the actual request, is then also not easily available to external entities.

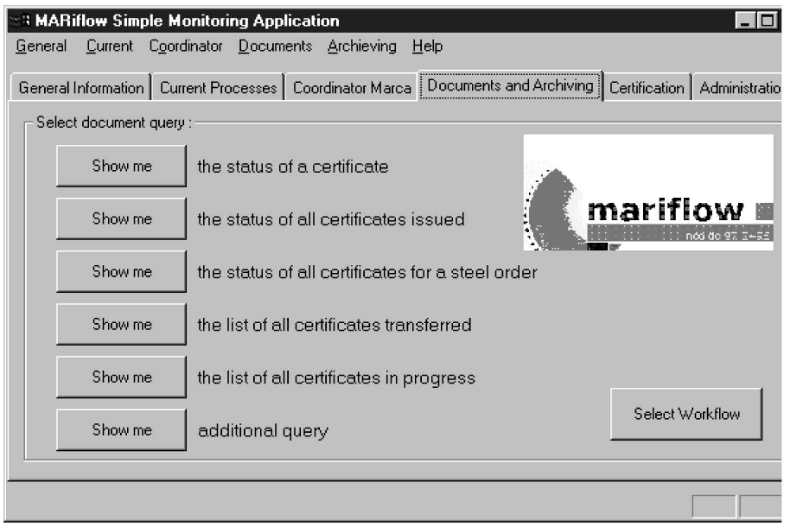

MARiflow Simple Monitoring Application
General Current Coordinator Documents Archieving Help
General Information | Current Processes | Coordinator Marca | Documents and Archiving | Certification | Administratio
Select document query :
Show me
the status of a certificate
Show me
the status of all certificates issued
Show me
the status of all certificates for a steel order
Show me
the list of all certificates transferred
Show me
the list of all certificates in progress
Show me
additional query
Select Workflow
mariflow

Figure 7. A monitoring Interface specific to the given Example Definition

In both situations above, as the MARCAs need to know the sources and destinations of messages, and the process to which they belong to, such details will have to be provided additionally to the encrypted message. As the MARCA system is outside the firewall, it cannot participate in the internal encryption scheme. To further increase the level of protection and of authentication, i.e., the certainly regarding the sources of received messages; the MARCAs use an independent encryption scheme. While this provides a reasonable degree of protection against casual listeners, clearly it does not provide the same level as the internal schemes, simply because the MARCAs are outside the firewalls, hence their code is open for analysis. Nevertheless, we believe that the solution is satisfactory, given the additional level of protection and authentication provided by the intra company systems. The alternative, namely the development of a trusted authority to manage protection and authentication schemes for all involved companies, requires both further research, such as development of virtual firewall for clusters of companies, as well as a rather drastic change in business procedures.

4 Availability and Scalability

When a site goes down, restarting the MARCA is under the responsibility of the Operating System's start up control. The site's start up control is to analyze the persistent logs of the MARCA and start a new instance using these stable logs created before the site crash. However, there is need for a further mechanism to prevent any Operating System related problem.

In MARIFlow, for each MARCA installed there is a background process at that site, called the "rescue process". The rescue process is responsible for monitoring the lifetime of the agent and checks the MARCA at specific time intervals through a predetermined socket. A thread of the MARCA listens to this socket and responds to the signals. If the MARCA does not respond to this process for a given period of time, the process starts sending signals more frequently. If the MARCA still does not respond, after sending a bunch of signals the process assumes that the MARCA is not functional. The two possibilities in this case are the MARCA could be blocked or it could be dead. When the rescue process is unable to find the OS process that belongs to this MARCA (i.e., it is dead), it instantiates a new MARCA by the help of the persistent logs related with the state of the agent.

Otherwise if the MARCA is blocked, it is necessary to kill the old instance prior to installation of a new instance. Since the logs are persistent it is possible to recover the state of the MARCA killed and hence the site does not suffer from any inconsistencies.

For the described mechanism to work correctly it is necessary to make sure that rescue process stays alive. Therefore, just as the rescue process checks to see that the MARCA stays alive, the MARCA also checks to ensure that the rescue process stays alive by signaling the rescue process at predefined time intervals. It is MARCA who re-instantiates the rescue process when it dies.

5 Future Work

The current prototype of the system does not yet include the mechanisms for the following issues:

5.1 Compensation

Compensation activities are used to logically undo the effects of the activities with which they are associated - Dayal (1991). They are used when a failure occurs in the system and the system should be taken to a stable state before the failure. On the other hand, workflow processes are long-running activities consisting of many nested sub-activities. In an activity hierarchy, the failure of a sub-activity may cause its parent or higher level ancestors to abort. However, it is not acceptable to roll all the finished activities back in case of failures. A hierarchical approach to failure handling which allows for partially rolling back to the nearest point in process history tree where it is possible to restart execution is required. Moreover, failures should be handled in a timely and efficient fashion. Chen and Dayal (1996) provide such a mechanism that can be adapted into the MARIFlow system. It provides a failure handling mechanism consisting of two phases. When a sub-activity *T* fails, it is necessary to determine the impact of that failure on the ancestors of *T* by finding out the highest level ancestor that should be aborted. The root of the activity sub-tree to be logically undone upon *T*'s failure is called the *Logical-Undo Root(LUR)* of *T*. Every activity in an activity hierarchy has a corresponding *LUR*, which may be one of the following:

- the closest non-vital ancestor since its failure can be ignored by its parent,
- the closest ancestor with a contingency activity (children of a contingency block),
- the closest ancestor without a parent, which may be the top-level process or a compensation activity.

Bottom-up searching for *LUR* in the process tree constitutes the first phase of the approach. After the *LUR* is found, the effects of the activities in the sub-tree with *LUR* as the root are removed in a top-down fashion. Applying the undo operation top-down provides a timely reaction to a failure by halting the activities in scope of *LUR* promptly. In this second phase, starting from the *LUR*, the finished activities are compensated (if they have compensation activities) taking the semantics of the blocks that enclose them into consideration. The approach also allows compensations to be made as high level as possible since compensating a high-level activity is more general than compensating a lower-level activity. After this two-phased algorithm is applied to the activity hierarchy, the execution restarts and rolls forward from the next activity after *LUR*.

5.2 Exception Handling

Hagen and Alonso (1998) propose a flexible approach to exception handling. In this approach, the business logic is separated from exception handling logic, which makes it easier to keep track of each. Exception handling code is separated from the normal code, which also provides reusability of components in addition to simplicity.

In Hagen and Alonso (1998), exceptions are treated as separate objects. Each has a name and parameters. Each exception type should be registered with the system giving its name and interface. Also, each one has a category that indicates the behavior of the handler with which it is associated. An exception handler is a special process that is started when an exception has been signaled. At the beginning, workflow components (activities), exceptions and handlers each handling an associated exception are defined in the system. Then, a workflow process is composed using these components. This component architecture approach provides both reusability and flexibility where different components can be used in different combinations in different workflow definitions. However, the approach does not have a solution for handling unpredictable events that may occur at execution time. All the exception cases and their handlers are given at the beginning.

6 Conclusions

In this paper we described an architecture for a workflow system for document flow over the Internet realized through cooperating agents. The architecture provides for the declarative specification and automatic generation of the application rather than producing large amounts of application specific code by the help of block structured nature of the workflow definition language.

The system attempted to bring solutions to user interaction through the Java based Web interfaces, reliable and automated document and data transfers through a distributed agent based architecture and it is more resistant to localized failures than that of centralized systems. The block structured nature of the language avoids unreachable states in the workflow execution and also the deadlocks - Alonso (1995). The system also proposes an adaptable design and implementation so that the engine will run in different platforms ranging from a small set of personal computers to high end workstations and mainframes by the universal programming language Java.

The architecture is general enough to be applied to any domain however the example application is provided for maritime industry and therefore some of the user interfaces are customized accordingly. Nevertheless the core of the system, workflow specification, communication details and database activities are suitable for any kind of domain definition thanks to the layered architecture.

The first prototype of the system including the details that are worked around in this paper is available. The issues like compensation and exception handling are yet to be tackled.

References

G. ALONSO, R. GUNTHOR, M. KAMATH, D. AGRAWAL, A. El ABBADI, C. MOHAN, (1995), "Exotica/FDMC: A Workflow Management System for Mobile and Disconnected Clients", *Parallel and Distributed Databases*, The Netherlands.

G. ALONSO, C. HAGEN, H. SCHEK, M. TRESCH, (1998), "Towards a Platform for Distributed Application Development", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 195-221.

P. C. ATTIE, M. P. SINGH, A. SHETH, M. RUSINKIEWICZ, (1993), "Specifying and Enforcing Intertask Dependencies", *Proceedings of the 19th VLBD*, Dublin, Ireland

Q. CHEN, U. DAYAL, (1996), "A Transactional Nested Process Management System", *in Proceedings of 12th International Conference on Data Engineering*, New Orleans, LA.

A. CICHOCKI, M. RUSINKIEWICZ, (1998), "Migrating Workflows", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 339-355.

U. DAYAL, M. HSU, R. LADIN, (1991), "A Transactional Model for Long-Running Activities", *Proceedings of the 17th International Conference on Very Large Data Bases*, Barcelona

U. DAYAL, Q. CHEN, TAK W. YAN, (1998), "Workflow Technologies Meet the Internet", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 423-438.

A. DOGAC, E. GOKKOCA, S. ARPINAR, P. KOKSAL, I. CINGIL, B. ARPINAR, N. TATBUL, P. KARAGOZ, U. HALICI, M. ALTINEL, (1998), "Design and Implementation of a Distributed Workflow Management System: METUFlow", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A Sheth (Eds), Springer-Verlag pp. 61-91.

D. GEORGAKOPOULOS, M. HORNICK, A. SHETH, (1995), "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure", *Distributed and Parallel Databases*, Ahmed K. Elmagarmid (Ed-in-chief), Volume 3, Number 2, pp. 119-153.

C. HAGEN, G. ALONSO, (1998), "Flexible Exception Handling in the OPERA Process Support System", *18th International Conference on Distributed Computing Systems (ICDCS 98)*, Amsterdam, The Netherlands.

D. HOLLINGSWORTH, (1996), "The Workflow Reference Model", *Technical Report TC00-1003, Workflow Management Coalition*, Accessible via: http://www.aiai.ed.ac.uk/WfMC/

J. MILLER, D. PALANISWAMI, A. SHETH, K. KOCHUT, H. SINGH, (1997), "WebWork: METEOR$_2$'s Web-based Workflow Management System", *Journal of Intelligent Information Systems*, The Netherlands

P. MUTH, D. WODTKE, J. WEISSENFELS, G. WEIKUM, A. DITTRICH, (1998), "Enterprise-Wide Workflow Management Based on State and Activity Charts", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 281-303.

MING-CHIEN SHAN, J. DAVIS, W. DU, Y. HUANG, (1998), "HP Workflow Research: Past, Present, and Future", *in Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 92-106.

A. SHETH, D. GEORGAKOPOULOS, S. JOOSTEN, M. RUSINKIEWICZ, W. SCACCHI, J. WILEDEN, A. WOLF, (1996), "Report from the NSF Workshop on Workflow and Process Automation in Information Systems", *SIGMOD Record*, 25(4):55-67

A. SHETH, K. KOCHUT, (1998), "Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems", in *Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 35-60.

M. SINGH, (1996), "Synthesizing Distributed Constrained Events from Transactional Workflow Specifications", *in Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, New Orleans

G. VOSSEN, M. WESKE, (1998), "The WASA Approach to Workflow Management for Scientific Applications", *in Workflow Management Systems and Interoperability*, NATO ASI Series, A. Dogac, L. Kalinichenko, T. Ozsu, A. Sheth (Eds), Springer-Verlag pp. 145-164.