The Liaison Workflow Engine Architecture

Karl R.P.H. Leung† Jojo M.L. Chung‡
Department of Computing,
Hong Kong Polytechnic University,
Hong Kong
E-mail: {cskleung,csrchung}@comp.polyu.edu.hk

Abstract

Liaison is a workflow management model that supports complex workflows and human resources management. Liaison System is the implementation of the Liaison Model. This paper discusses the architecture of the Liaison Workflow Engine, the core of the Liaison Workflow Management System, which is responsible for controlling workflows. Besides managing and controlling complex workflows and human resources, this Liaison Workflow Engine architecture is designed to have high flexibility so as to cope with heterogeneous distributed systems and different kinds of distributed workflow applications.

The Liaison Workflow Engine consists of five major components, namely: Scheduler, Task Manager, Actor Manager, Information Manager and Actor Interface Manager. These components are responsible for workflow management, task management, job assignments, information management and interface management respectively. There are also Exception Handlers associated with Scheduler, Task Manager, Actor Manager and Information Manager to manage and handle exceptions in each of these components.

Keywords: Workflow Engine Architecture

1 Introduction

Workflows are collections of related activities, called tasks, interacting with each other to achieve some business goals. In contemporary complex business applications, besides sequential workflows, there are also branches and merges or even loops in workflows. Hence the controls of synchronization between tasks in workflows are very complex. The business processes as well as organization structures change from time to time. The traditional manual control procedures, quite often, are found to be too slow in responding to dynamics changes. Agents, who can either be people or machines, are assigned to carry out the tasks. We call people who process tasks actors. Moreover, associating with these complex dynamic tasks, problems such as exceptional handling and agent assignments are becoming more complicated also.

The fast developing data communication technology facilitates different tasks of a workflow to be carried out in different locations, and hence create a demand for distributed workflows. For example, an organisation may have departments using different computer platforms or distributed in different geographical locations. Hence these department require to communicate with each other in order to complete the workflows which involve them.

The Liaison model [8], is designed to model all these workflow features in terms of its specification language Valmont [9]. One of the goals of the Liaison project is to develop a Liaison system which is capable to manage the workflow systems captured. The architectural design of the Liaison Workflow Engine is discussed in this paper. This engine is responsible for the control of workflows. The discussion will be evolved from centralized to distributed design.

There are a number of workflow management systems available in the market [12, 13, 11]. Few details about these commercial workflow system architecture have been released to public.

Some reports of workflow management systems have
discussed their system architectures. The architecture of WIDE [5] focused on database technology. The workflow processes are managed through extended transaction processing techniques and active rules management. The persistence is implemented by the relational database technology. Its design is based on a client/server architecture on top of Oracle. Another example is METEOR [10], which focused on supporting multi-system workflow applications. The WFMS Reference Architecture [6] is a more complete and more generic workflow management system architecture. In WFMS, actor assignments and task management are optional extension modules. Furthermore, in handling distributed workflows, WFMS only supports co-operative workflow systems.

The workflow features supported by Liaison are richer than in other workflow systems. Besides complex workflows, Liaison also supports explicit specifications of exceptions and constraints, and sophisticated organisation structures. This requires the Liaison workflow engine to handle more sophisticated workflows as well as complicated actor assignments. The design of the Liaison workflow engine architecture is more generic and flexible. No specific computer system nor network architecture is assumed. Furthermore both centralized and distributed workflow systems are supported. In handling distributed workflows, Liaison can support different kinds of distributions, which design is more flexible. Consequently, the rich functional coverage of workflows, generality and flexibility of this Liaison workflow engine architecture design allows it to be acted as a general reference workflow engine architecture.

Liaison conceptual model is discussed in Section 2. The Liaison centralized workflow engine architecture is discussed in Section 3. These include the features of the components constituting the engine and the interactions between these components. This centralized engine is then extended to handle distributed workflows which is discussed in Section 4. The design and implementation principles of the Liaison workflow engine are discussed in Section 5. Afterwards, the results will be concluded in Section 6.

2 An Overview of Liaison

The Liaison workflow model is an extension of the reference workflow model proposed by the Workflow Management Coalition [3]. Like the reference workflow model, it captures the fundamental elements of the workflow paradigm: organisation model, information model, process model, and their relationships. Unlike the reference workflow model, it supports a rich organisation model and sophisticated activity assignment constraints.

2.1 Organisation Modelling

The organisation structure chart (OSC) of an enterprise as shown in Figure 1 captures the position hierarchy and some functional groupings of the enterprise. OSC provide information on the personnel who are related to or affected by the questioned system. These personnel, usually, can help the software engineers to understand the general objectives, requirements, and operation environment. It should be noted that such an OSC captures the delegation and accountability relationships but does not usually reflect all the functional and operational structures of an enterprise. There are some functional groups which members come from different branches of the OSC. These functional groups, very often, are difficult to be included in the hierarchical structure of the OSC. Furthermore senior staff who are also members of these functional groups need not necessary be the leaders of these groups. For example, an IT Committee may consists of members from different departments. Although a senior marketing manager may be a member in the committee, this committee may be lead by a project manager of the IT Department who is junior than the senior marketing manager. Moreover, in modern international enterprise, functional groups may be distributed in different geographical regions. As far as workflow application design is concerned, these functional groups can form a hierarchy.

The functional structures of an enterprise can be captured using teams and the team hierarchies. The grouping of individuals into teams should also take activity assignments into consideration. For example, a large team would be preferred for high volume or repetitive activities. It should be noted that it is possible and likely that the team hierarchy contains free standing teams that are not be related to any
other teams (see the team hierarchy on the right of the abstract organisation structure shown in Figure 2).

![Figure 2. Positions, Staff, Teams, and their Relationships](image)

The relationships between positions, individuals, and teams are shown in Figure 2. Each individual (grouped in the center) is assigned with a position title and is assigned to one or more teams. The position hierarchy is shown on the left where a ...er could stand for a position such as Information Technology Manager or Programmer. Teams and team hierarchies (related by arrows) are shown on the right. The assignment of individuals to positions and to teams is shown by the dotted line arrows.

### 2.2 Information Modelling

The information model defines the data used in a workflow process, governs the operations that can be performed on the data, as well as controls the presentation of data. In other words, the model defines data that are passed between activities within a workflow process as well as data that go across the boundary of the workflow process. Data can be application data or control data. These data are managed by the workflow management system. In additional to these usage, the data can also serves as connections between workflow process and other external systems that may be heterogeneous and distributed in different geographical locations. For instance, the sales workflow has to check the stock level from the inventory database. It also has to invoke another application for the processing of invoicing which may be geographically located somewhere else and which computer system is of different platforms as shown in Figure 3.

A piece of such data is represented in Valmont [9], as an entity whose state may not be known to the workflow systems but usually operations are defined to manipulate the data. All data are defined in the data model which is a sub-model of the information model.

Presentation of data is defined independently in another sub-model called the form model. A form is a unit of presentation that is used whenever interaction with the user is required. It is possible that several forms can be displayed for a single task. Typically a form has a number of fields in which data can be displayed and captured. To improve quality of the display, formatting information can be provided. One form can be derived from another form possibly by removing some fields or restricting operations that can be applied on some fields. This mechanism encourages reusability which, in turn, reduces the maintenance effort.

### 2.3 Process Modelling

The process model plays a central role among the submodels of Liaison. It ties the organisation, data, and form models to activities representing the steps of the application being modelled. To facilitate the modelling of complex systems as well as making the resultant design more comprehensible, the process model supports incremental modelling allowing an activity to be refined into more detailed activities (see Figure 4). In other words a typical design would end up with several layers of activities with more abstract activities called tasks at the upper layers and more concrete activities called base tasks appearing in larger number at the very bottom layer.

Liaison supports modelling of sophisticated workflows. Besides sequential workflows, loops in workflows and, different kinds of branches and merges of workflows such as conditional branch (merge) and un-conditional branch (merge) are supported by Liaison. Furthermore all the children tasks branched by processes can run concurrently with their parents. These control and synchronization between tasks are also modelled by Liaison.

The concern of the first stage of workflow process modelling is to identify the concerned agents and figure out the activities they have to be carried out to fulfill the application. These agents can be programs or actors. The identified actors should have been captured in the organisation model. Activities and their order of execution (i.e. control flows) together with the responsible actors (i.e. activity assignment) are then captured in the process model. Since both control flows and activity assignment can have conditions associated with them, different notations are used to
reflect the situation. Abnormal situations in the process can then be identified as exceptions which may be triggered by activity events that happen in the process as well as temporal events that are defined in the process. For instance, it is possible to raise an exception when no agent can be found to carry out an activity or to send a reminder to the Project Manager when the review of the functional specification is due say in three days.

So far what has been discussed can be applied to both tasks and base tasks; however, base tasks can perform other actions that are not found in tasks. The primary difference is the possibility of displaying forms (described in the form model) to initiate some interaction with the user – note that the use of forms is the sole mechanism in Liaison to pass data between activities. It is not unusual that the data and form models need to be revised during process modelling.

3 Liaison Workflow Engine Architecture

The Liaison Workflow System consists of two major components: the build-time engine and the run-time workflow engine. The build-time engine is responsible for parsing workflow specifications in Valmont to generate the local data. These include workflow definitions, task definitions, organisation structure and actor information, and, form and data definitions. These local data are kept in the repositories of Workflow Definition, Task Definition, Actor Information, Form & Data Store, respectively (see Figure 5).

The workflow engine is responsible for managing and controlling workflows and task executions, actor assignments and information usage. In the Liaison workflow engine, these responsibilities are handled by the co-operation of Clock, Scheduler, Task Manager, Actor Manager, Information Manager, Actor Manager and Actor Interface Manager. This architecture is shown in Figure 5.

The workflow control processes are shown in Figure 6. After selecting tasks to be executed, in step 1, the Scheduler passes the control of these tasks to Task Manager for processing. In step 2, the Task Manager requests Actor Manager to assign actors to process the tasks. If actors can be assigned, the Actor Manager acknowledges the Task Manager with the assignments in step 3. Afterwards, in step 4, Task Manager informs the Information Manager about the actor assignments. The Information Manager prepare the forms and data, and informs the Actor Interface Manager that the tasks are ready for processing in step 5. Actors are then informed by the Actor Interface Manager that they can start processing the tasks. In step 5(a), in case, permitted by the task definitions, actors have to fork some tasks during the processing, the actors can do so by informing the...
Scheduler through the Actor Interface Manager. On completion of tasks, actors inform the Task Manager through the Actor Interface Manager in step 6. In the last step 7, the Task Manager informs the Scheduler, Information Manager and Actor Manager on completion of tasks.

The Scheduler, Task Manager, Information Manager and Actor Manager are equipped with Exception Handlers to handle exceptions. In addition to the stores used in build-time, Worklist and Log are repositories which are used in run time to keep tasks assignments and tasks status, and all the history of the engine, respectively.

Functions, responsibilities and details of each step in the control process of each of the components in the architecture are described in the following sections. These are the top level descriptions of each of the components. In the full description, each of these components have refinements. The functions of each of the components will be handled by different modules in the corresponding refinement modules.

3.1 The Scheduler

The Scheduler controls workflows. Workflows are initiated by users. Each initiation is regarded as an instance of the workflow. Users may initiate a number of instances of the same workflow. Users may also initiate different workflows instances before the completion of the previous workflows instances. Hence the Scheduler has to distinguish and keep check of these Same-Workflow-Multiple-Instances and Different-Workflows instances.

Scheduler is also responsible for scheduling the execution of tasks. Tasks in Valmont are specified incrementally from abstract levels, through refinements, to Base Tasks, i.e. tasks in the concrete base level. Hence the Scheduler should manage these refinements and go through these refinements to identify base tasks for actual executions.

After some base tasks are selected for executions, the Scheduler informs the Task Manager to execute these base tasks. The control of these tasks\(^1\) are passed to the Task Manager. This is done by sending the task name and number to the Task Manager. Task name is the name of the task defined in the Task Definition. Task numbers are the unique identifiers of individual tasks. Task Manager will inform the Scheduler on completion of these tasks. The Scheduler will invoke tasks whenever it is required and the criteria for invoking the tasks are satisfied.

Besides the simple sequential workflows in which a task will execute after the completion of its predecessor task, the Liaison workflow engine also supports complex workflows. The basic structures which constitute these complex structures are as follows:

\(^1\)Since base tasks are also a kind of tasks. From now on, when there is no ambiguities, the term “task” will be referred to “base task”.

**Branching** In a workflow, a task may fork several flows to form some branches. This forking of flows can be static or dynamic. In the static case, all branches are executed. In the dynamic case, there are also two situations:

1. only some selected combination of branches are executed, or
2. a specific number of successive branches are generated and executed.

**Merging** On the inverse, a task merge from some branches. These merging of flows can also be static and dynamic. In the static case, all the branches are merged. In this case, usually, the successive task requires all the data produced by its predecessor tasks. In the dynamic merging, there are also two cases;

1. some combination of branches are merged, or
2. a specific number of branches are merged.

**Concurrent** In Liaison, any task can be invoked whenever it is required and, all the pre-conditions and constraints of the task are satisfied. There is no restriction on concurrent execution of more than one task at the same time.

The combinations of these three structures can lead to very complicated workflows. For example, a task can fork successive tasks while it is still active. Then these successive tasks are executing in parallel together with their predecessor task. Similarly, a task can merge progressively with its predecessor tasks or start execution without waiting for the termination of its predecessor tasks.

The Scheduler controls these complex structures by keeping check of the pre-defined criteria of tasks in selecting successive tasks and, the pre-conditions and constraints associated with the successive task definitions. The Scheduler will also synchronize the tasks that are executing concurrently. The technique of dynamic binding is used in Liaison Scheduler to implement these features. The Scheduler makes the decision during the run time of the workflows according to the compiled workflow definitions. In the branching or merging of a number of tasks, the policy of “first come first serve” is adopted. This means that if the specific number of tasks have been forked, (or merged, respectively,) the Scheduler will discard the late comers.

In case a task has to fork some other tasks while that task is still active, the Actor Interface Manager will inform the Scheduler to do so. If the task definitions allow, users can initiate forking of tasks through the Actor Interface Manager.

Workflows may change. Off-line workflow evolution is supported in this design. New changes are stored in the corresponding definition repositories by the build-time engine.
The Scheduler is also responsible for the configuration management of the workflows. Different versions of workflows will be managed. Unless the outstanding instances are required to be changed, these outstanding instances will follow the version of workflow definitions which they started with. New workflows, without special requests, will use the new definitions.

3.2 The Task Manager

The Task Manager controls the execution of the base tasks. After receiving the controls of tasks from Scheduler, the Task Manager creates records in the Worklist. These records keep the task names, task numbers, states and agent assignments. Then the Task Manager changes the states of the tasks into Initial Validation. In initial validation, the time requirements of the tasks are checked first. It is because some tasks may have time requirements such as they have to be executed at some specific time or have to be completed within some specific durations. Then the Task Manager checks the satisfaction of the pre-conditions and constraints. If the tasks do not satisfy any of the time requirements, pre-conditions or constraints, exceptions will be raised by the Task Manager Exception Handler.

Tasks can be performed manually\(^2\) or automatically by computer. Corresponding agents will be selected to process the tasks. For those automatic tasks, the Task Manager invokes the corresponding programs and, changes the states of the tasks into Active and records the agents as Computer. For manual tasks, the Task Manager changes the states of the tasks into Actor Requested and requests actors from the Actor Manager with the task identity, i.e. task name and task number. When Actor Manager returns with actors, the Task Manager will change the tasks into Actor Assigned state. It then informs the Information Manager that these actors are authorized to use the corresponding forms or data. Moreover, the information includes the position, actor attributes such as skills and qualifications, organisation and team structure, and, accountability of individual actor and functional teams. The Information Manager sends a message to the corresponding departments, in assigning agents to tasks, the possible outcomes are:

1. No suitable actor is found, or
2. Suitable actors are found and the actors accept the tasks, or
3. Suitable actors are found but the actors refuse the task.

Case 2 is the normal case. The Actor Manager returns the actors’ identities to the Task Manager together with the tasks numbers and record the assignments in the worklist. If it is case 1, The message No suitable actor is available will be returned to Task Manager together with the task numbers. If it is case 3, the Actor Manager will record the matter and search for the another suitable actor again.

After the completion of tasks or cancellation of tasks, the Actor Manager will be informed by the Task Manager. The Actor Manager then checks with the selected actors, see whether they are going to accept the task assignments. Hence in assigning agents to tasks, the possible outcomes are:

1. No suitable actor is found, or
2. Suitable actors are found and the actors accept the tasks, or
3. Suitable actors are found but the actors refuse the task.

Liaison Organisation Model of Liaison captures actor information including the position, actor attributes such as skills and qualifications, organisation and team structure, and, accountability of individual actor and functional teams. The Organisation Model also captures the distribution of actors in different locations and the registration of actors in different positions and teams. It should be noticed that Liaison allows an actor to register in more than one team or more than one position. All this information is kept in the Actor Information.

The Actor Manager will keep check of the availability of actor and update the actor attributes. Consequently, it has to link up with the organisational personnel database. In additional to all the information keeping in the Actor Information, in assigning actors to tasks, Actor Manager also considers a number of factors such as:

\(^2\)Manual tasks refers to those tasks which are not performed by computer programs.
1. The requirements as specified in the task definitions, These requirements can be simple requirements such as actors with specific skills or ranks. These requirements can also be complex constraints such as two tasks should be processed by the same actor.

2. The availability of actors,

3. The fairness of actor assignments,

4. The actor assignment policy of the organisation, and

5. The estimated execution time required by the tasks.

During processing of tasks, re-assignment of actors to tasks may be required. This can be caused by the unavailability of actors during processing or re-allocating tasks so as to achieve better assignment results. In performing actor re-assignments, the Actor Manager informs the Information Manager of the change of assignments and updates the corresponding worklists.

### 3.4 Information Manager

The Information Manager is responsible for managing the Forms and Data, i.e. the external data, which definitions are specified in the Information Model of Liaison. These management duties include data integrity and authentication. The Information Manager, according to the actual computer system implementation, should be equipped with the appropriate data integrity mechanism. Since suspension and roll-back of tasks are allowed in the Liaison workflow system, the Information Manager is responsible to handle these complex operations and deadlocks as well.

After receiving actor assignments from the Task Manager, the Information Manager prepares the corresponding forms or data for the actors. These preparations can be an instantiation of a new copy of the forms or data and authorizes the corresponding actors to use these forms and data. The states of the tasks are then changed to Ready and the Actor Interface Manager is informed that the forms and data are ready for processing. After the tasks are finished, the Task Manager will inform the Information Manager about the completion. Then the actors are no longer allowed to access these forms and data. If there are changes of actor assignments, Information Manager will be informed by the Actor Manager. The access right of the forms and data will be changed accordingly.

### 3.5 Actor Interface Manager

The Actor Interface Manager is responsible for managing the actor interfaces, which actors use to communicate with the workflow engine and monitor of the some of states of processing of tasks.

After receiving the ready message from Information Manager, the Actor Interface Manager will inform the actors that they can start processing the tasks. When actors start processing the tasks, the Actor Interface Manager will change the states of the tasks into Active. If there are any messages to be displayed at the beginning of processing, the Actor Interface Manager will display these messages to the corresponding actors and users. If the processing of tasks is paused by the actors, the Actor Interface Manager will change the states of the tasks into Sleeping. When the tasks restart, the states of the tasks are changed to Active again. Actors will inform the Actor Interface Manager on completion of tasks. Then Actor Interface Manager will change the states of tasks to Provisional Complete and informs the Task Manager about the completion.

During the processing of tasks, permitted by task definitions, if some other tasks are required to be forked, the actors can inform the Scheduler through the Actor Interface Manager. The Actor Interface Manager will also prohibit actors accessing those suspended or cancelled tasks.

### 3.6 Exception Handlers

Exceptions are commonly regarded as those abnormal conditions that happened during the execution of the workflows. Liaison classifies exceptions into the following categories.

1. **Computer systems or communication systems errors** — These exceptions are caused by computer malfunctions such as disk crash or communication system problems such as network down and overflow errors.

2. **Workflow operations** — These exceptions are those which violate the assumptions of workflow concepts such as there is no appropriate actor for a task.

3. **Specification violation** — In some cases, the actual outcome may violate the specifications. For example, a task does not complete within the specified duration.

4. **User defined exceptions** — Valmont supports users defining phenomena as exceptions in the workflow specifications.

5. **External exceptions** — External exceptions are those exceptions raised from the environment such as cancellation of workflows requested by users.

A set of procedures relating to handle different abnormal conditions should be defined in advance. Exception detections and recovery handling are the duties of the exception handlers in Scheduler, Task Manager, Actor Manager and Information Manager. Exceptions can occur at six different levels. They are
1. Workflow level
2. Task level
3. Agent assignment level
4. Data level
5. Form level and
6. User level.

All these exception handlers are pro-active, i.e. they will detect and monitor the status of the workflow and system environment continuously. Whenever there are exceptions, the exception handler will raise the exceptions, analyze the exceptions, find out the handling methods and activate the recovery procedures. If the exceptions and recovery procedures are local to a manager, the corresponding exception handler is responsible for resolving the exception locally. Otherwise the exception handling is passed to the Scheduler’s Exception Handler. The Scheduler Exception Handler will also handle those exceptions raised by external users such as cancellation of workflows.

In analyzing exceptions and determining recovery procedures, the exception handlers may need to co-operate with each other. For example, a time-out exception occurred in actor assignment may lead to the cascade of time-out exceptions in Task Manager and Scheduler. The exception handlers have to co-operative with each other to identify the source of exceptions and determine the corresponding recovery procedures. The followings are some of the actions in recovery from exceptional cases.

1. Cancel the task;
2. Suspend the task;
3. Prompt the exception message to the appropriate actors and user and passes the control to them;
4. Carry out some pre-defined recovery procedures.

3.7 Clock

A system clock is required for the workflow engine. It is the time reference for all the units such that the co-operations between different units of the workflow engine and the parallel tasks can be synchronized with each other. In the Liaison workflow engine, all events are time-stamped with reference to the clock and logged. In case of a distributed engine, a distributed clock will be required to serve such purpose.

3.8 Workflow Management Unit

The Workflow Management Unit is part of the Liaison Management Information System which provides information about the progress and status of the workflows. This Workflow Management Unit does not belong to the Workflow Engine. Managers can enquire about the status of the workflows such as the number of active tasks through this unit. Statistical analysis can be included in this unit also. The information is obtained from different repositories such as Worklist and Log. This is also the reason that all events occurring during the execution of the workflow system are kept in the Log repository.

4 Distributed Architecture

Distributed workflows occur when not all the tasks of the workflows can be processed at the same domain. Liaison supports different kinds of distributions. All Scheduler, Task Manger, Actor Manager, Information Manager, Actor Interface Manager and repositories can be either centralized or distributed. As shown in Figure 7, each column represents a domain. A domain can consists of all or some of the components of the workflow engine. Liaison assumes hetero-

![Figure 7. Distributed Architecture](image-url)
ager, i.e., the Actor Manager can only be accessed by the Task Manager.

2. If a domain has Scheduler and Task Manager. Execution of tasks in the domain can only be requested by the Scheduler to the Task Manager, i.e., the Task Manager can only be accessed by the Scheduler.

3. There should be at least one designated domain call Distribution Prefect which keep the availability of all domains in the system. If there is more than one distribution prefects, there should be some mechanisms to ensure the integrity of the domain availability information among these distribution prefects. When a domain is up or shut down, it should register its availability or unavailability, respectively, with the distribution prefects.

It should be noticed that there is no restriction on the distribution of Information Manager and repositories. Furthermore, all components communicate with each other by message passing. This allows very high flexibility to adapt to different domain distributions and computer systems configurations.

This design supports centralized workflow engine, cooperative operations between multiple workflow engines, and any other kind of distributions such as centralized Scheduler but distributed Task Manager and Actor Manager. The design is so flexible that it can cope with different real-life distributed situations. It should be noticed that even though there is a centralized Actor Manager, it does not mean that all the actors are located in the same domain.

5 Discussion

5.1 Design Principles

In the design of the Liaison Workflow Engine architecture, the following principles are followed.

KISS — KISS stands for Keep It Simple and Straight. The Liaison workflow engine is designed to be as simple and straightforward as possible. This will also help in keeping the software size small which is important in distributed systems.

Generic — The design is a generic design such that it can support different workflow applications and computer and network configurations.

Software Engineering Practice — Modular and top-down approaches are adopted and enforced in this design. All modules have well-defined responsibilities, functions and interfaces. These modules are refined into some other sub-modules which these software engineering practices are still enforced.

5.2 Implementation

We are developing a prototype of this Liaison Workflow Engine. A prototype of Vicomte, the Information Manager module, is reported in [7]. This Liaison workflow engine prototype is built on top of a distributed system consisting of Unix Solaris and Window NT platforms. orbix [2], an implementation of CORBA [14], is adopted as the software bus between domains. In this prototype, in order to give more flexibility in managing persistent data on heterogeneous systems, workflows are migrated between domains when necessary and persistent data are managed at the program level. These requirements are achieved by developing the prototyping with PJama [4, 1], a Java extension which is equipped with persistent data features and, extension of PJama with persistent object migration management features.

6 Conclusion

Features of workflow controls are identified with reference to the Liaison model. Based on these features, a workflow control process is proposed and a workflow engine architecture is designed. This architecture can handle complex workflow management including complex workflows, exception and workflow evolution handling, information usage management and agent assignments. It is generic enough to support different kinds of workflow requirements. It also support different kinds of workflow distributions.

This architecture is a simple and flexible design. The design is highly modular. This modular design lets the engine has the flexibility in adapting to different kinds of distributions. The flows between modules are carefully studies so as to eliminate unnecessary flows. Furthermore, this Liaison workflow engine architecture can also be acted as a general reference workflow engine architecture because of its generic design, flexibility and generality.

References

[1] Pjama release 0.4.6.12 (for jdk 1.1.6). http://www.sunlabs.com/research/forest/opj.main.html.


