A workflow management system based on intelligent collaborative objects

G. Fakas\textsuperscript{a}, B. Karakostas\textsuperscript{b,}\textsuperscript{*}

\textsuperscript{a}Department of Computer Science, University of Cyprus, Nicosia, P.O. Box 537, Cyprus
\textsuperscript{b}Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, UK

Received 8 September 1998; received in revised form 10 May 1999; accepted 11 May 1999

Abstract

This paper describes an architecture for workflow management systems based on Workflow Intelligent Business Objects (WIBOs). The design of WIBOs is based on principles of intelligence, autonomy, collaboration and co-operation. Using WIBOs that carry out tasks on users’ behalf, it is possible to build workflow systems that bring further improvements in process automation and dynamic management, and achieve dynamic (re)allocation of resources to Actors. A WIBO prototype architecture has been implemented using Java. A Java Remote Method Invocation (RMI) has been used to enable WIBOs to communicate over an Intranet or Internet. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Workflow; Collaborative workflow management; Intelligent agents; Workflow intelligent business objects

1. Introduction

Productivity and customer service are key business issues for companies today. They are both inextricably linked to the technology of a workflow management software. A workflow management system is software program(s) that will either completely or partially support the processing of work item(s), in order to accomplish the objective of a workflow process activity instance or instances [1]. Workflow management software is forecast to show rapid increase in sales over the next few years according to a report by the international market research consultancy Ovum [2].

Workflow is the name given to procedures that involve the routing of tasks from person to person in sequence, allowing each to make a contribution before moving it on to the next stage. Examples are the processing of a business order requisition, or an insurance claim. Increased productivity in workflow systems comes from an automated support of storing, creating, accessing, processing and routing of forms, letters and so on. Increased management of business processes will also play a part, as management see where the effort is going, and tune the processes accordingly.

However, the current generation of workflow management systems suffers from a number of limitations such as:

- Management of the workflow process is often limited to warnings. It is difficult to dynamically correct a poorly functioning process by, for example, changing priorities or reassigning work. [2,3]. Some workflow management systems allow dynamic change to procedures [4], while others ensure organisational flexibility by adaptation and negotiation [5]. Few systems, however, support the dynamic process management [2].
- In current workflow tools, pure roles (i.e. the association of tasks with roles rather than individual workers) have their limitations: if they are used, it is not always possible to block the indirection which this introduces. The use of a role-based reference, while powerful, can obscure important relationships between tasks and the individuals who perform them. [2].

The Workflow Intelligent Business Object (WIBO) approach described in this paper has been devised to overcome limitations of current systems such as those discussed above. WIBOs are intelligent business objects that are used to model and implement workflow management. Using WIBOs, acting on user’s behalf, it is possible to build workflow systems that exhibit intelligence, autonomy, collaboration and co-operation. The use of WIBOs can result in a better dynamic management of the process and utilisation of resources.

The remaining of this paper further explains the WIBO...
2. Technologies and platforms for workflow management

Technologies for workflow management systems are evolving across three different dimensions namely, object-orientation, intelligent agents and Internet/Intranets. Workflow management systems developers should take into account this evolution.

2.1. Object-oriented workflow management systems

Object orientation represents an ideal technology for designing and building workflow management systems because of the advantages it offers to both workflow users and developers. In an object oriented approach, real life entities (people, resources etc) are modelled as objects in a natural manner. Flexibility is facilitated by the use of polymorphism, inheritance and encapsulation. In a truly object-oriented workflow system, there must be a uniform object-oriented feel to the product and support for the fundamental object-oriented concepts of reusability, abstract data typing, inheritance, object identity.

The fact that a workflow product is developed in an object-oriented environment does not mean that it can be described as being object-oriented. If, however, a workflow management system has an open architecture which can be specialised and extended through inheriting from an object-oriented language class library, labelling the workflow product as object-oriented is appropriate.

2.2. Intelligent agents in workflow management systems

Intelligent agents, software processes living on the network, can be trained to manage information on the user’s behalf, and can therefore improve the flow of information. The development of intelligent agents is likely to proliferate and expand the possibilities for electronic communication and workflow, by enabling workflow systems to respond to the countless little variations in timing and work that run most processes [3].

2.3. Workflow over the Internet/Intranets

Currently, vendors of workflow software are working hard to adapt their products for Internet use. We are already starting to see the use of Web servers as workflow servers. The real benefit is that new workflow processes can be spawned on the Internet, and can begin to co-ordinate the interaction of people across the Internet in a workflow-like fashion [3]. Internet-enabled workflow addresses two major needs. The first is to improve communication with customers (order processing, technical support etc.), the second need is to enhance any processes that require the participation of multiple parties who must collaborate to deliver a product or service.

3. Intelligent objects for workflow management

3.1. Overview of workflow intelligent business objects

WIBOs as proposed in this paper are objects that are used to implement workflow management systems. The motivation behind developing WIBOs is to overcome limitations of current workflow systems such as lack of dynamic process management and dynamic allocation of work to users and resources as well as role indirection.

An intelligent process management is a key requirement for workflow tools. This is catered for in our approach as WIBOs are able to manage themselves. Process management in current workflow systems is often limited to warnings. It is difficult for the designer to create a system which automatically corrects a poorly functioning process environment. For example, many products contain event triggers for time-based alerting, but few open up this facility fully to the designer and allow the system to take corrective action automatically (besides notification or escalation of priorities). Once an environment gets out of control, a storm of notifications is not sufficient to correct it; more radical steps are necessary [2]. These include:

- altering the resourcing of tasks, to enable more workers to help clear bottlenecks
- altering the allocation of tasks, to transfer a backlog of pending tasks from one user to another, or from one group of users to another.

3.2. Object-oriented characteristics of workflow intelligent business objects

WIBOs exhibit typical characteristics of object oriented systems:

- Reusability: New workflow objects can be built by using pre-defined objects. Any requirement changes can be easily accommodated for, since only the affected objects will need to change rather than the whole application.
- Inheritance-specialisation: Workflow objects can be generalised by extending their structure or behaviour. Alternatively, classes can also be specialised through restricting attributes or services.
- Composite workflow: Objects can contain or refer to
other objects. Programmers can dynamically construct arbitrary composite or complex objects, objects that are constructed from sub-objects.

- **Encapsulation**: The information and processing rules are combined into single objects. A typical workflow object contains not only the information being routed, but also the knowledge with its path through a process and the roles that act on it.
- **Instantiation**: The WIBOs are class objects from which instances of objects can be created at runtime.

### 3.3. Object collaboration

WIBOs are examples of collaborative agents. Collaborative agents emphasise autonomy and co-operation (with other agents) in order to perform tasks for their owners. They may learn, but learning is not typically a major emphasis of their operation. To have a co-ordinated set-up of collaborative agents, they may have to negotiate in order to reach mutually acceptable agreements on some matters. [6,7]

- **Co-operation** with other WIBOs is paramount: it is the raison d’etre for having multiple WIBOs to implement the workflow management system. Co-operation is achieved through the exchange of messages and asynchronous execution of operations.

### 3.4. The workflow intelligent business objects meta-model

WIBOs belong to one of a number of pre-defined inter-related classes or types which together form a meta-model i.e. Process, Role, Actor and Resources. Fig. 1 (which is based on a UML notation [8]) shows the WIBO meta-model.

#### 3.4.1. Process

Process is a collection of co-ordinated Activities that have explicit and/or implicit relationships among themselves in support of a specific process objective [9]. Process is responsible for co-ordinating other WIBOs (such as Actors, Roles and Resources). Its main functionality, therefore, is to manage, assist and route the workflow.

The process WIBO manages the workflow in the following ways:

- **Via alerting using deadlines**: A deadline is assigned to each Activity. If the Activity is not completed before the deadline, then it sends an alert message to the Actor responsible for executing the concerned Activity.
- **By prioritising**: Every Activity is characterised by a priority level relative to other Activities. This knowledge is used by the role object for more efficient task allocation and scheduling.
- **By real-time monitoring**: The process WIBO is capable of reporting its current state; i.e. total running time, current Activity and its status (waiting, deadline, role and Actor responsible) etc. This information is useful for tracing any bottlenecks of the process.
- **By estimating the time and resources required for execution**: The process is capable of estimating the total duration time and resources required for the process execution.
- **By dynamic process optimisation**: WIBOs support dynamic process optimisation by:
  - dynamically altering the resourcing of tasks, to enable more workers to help clear bottlenecks
  - dynamically altering the allocation of tasks, to transfer a backlog of pending tasks from one user to another or from one group of users to another.

The following tables summarise the functionality of process and Activity WIBOs.
3.4.2. Role

It is important to define roles independently of the Actors who carry out the Activities. It ensures the flexibility of process scheduling. Roles assign Activities to Actors. If an Actor is unavailable, then some other Actor is selected to carry out the work [2].

In current workflow tools, pure roles have their limitations: if roles are used, it is not always possible to block the indirection which this introduces [2]. The use of role-based reference, while powerful, can obscure important relationships between tasks and the individuals who perform them. This is most easily seen in the case of process loops, as they might occur in an approval cycle: ‘do it until it is done right’. On the second iteration of a task, it may be desirable either for the same person to adopt the role that it had in the previous iteration, or for a different person to adopt the role. Process maintains knowledge about each Activity, so it can block or not (according to the current needs) role indirection.

Role WIBOs have the following features and responsibilities.

3.4.2.1. Allocation of Activities to Actors  It is the Role’s responsibility to allocate Activities to Actors. Its aim is to dynamically make an optimised allocation of work by taking into account parameters such as:

- **Actors’ levels of experience** Every candidate Actor has a different level of experience (e.g. novice, expert, guru) in performing an Activity.
- **Actor’s workload** Actors with a heavy workload are less preferable
- **Use of role-base reference** In the case of process loops, Roles have the option to allocate iterated Activities either to the same Actor or to a different one.

3.4.2.2. Report Actors overload  The Role WIBO examines the Actors’ workload. If none of the Actors are able to execute the Activity before its deadline because they are overloaded, then Role can do one of the following:

- extending the Activity’s deadline
- allocating more Actors to the process
- changing the Activity’s priorities

If the Role discovers an Actor that will not be able to execute any of the Activities allocated to it before the deadlines, then the following might take place:

3.4.2.3. Reallocation of work  It is the Role’s responsibility to alter the allocation of Activity, and to transfer a backlog of pending Activities from one Actor to another. Reallocation of work employs the same criteria as allocation of work (i.e. taking into account the Actor’s level of experience, workload, use of role-base reference etc.).

The table below summarises the responsibilities of a role WIBO.

<table>
<thead>
<tr>
<th>Role Attributes</th>
<th>WIBO Specialisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>WIBO Specialisation</td>
</tr>
<tr>
<td><strong>FindThisRoleActors()</strong></td>
<td>Finds all the candidate actors for executing this activity.</td>
</tr>
<tr>
<td><strong>AssignActivityToActor()</strong></td>
<td>The Role assigns the activity to an Actor according to the following criteria: Actor’s level of Experience, Actor’s Workload and role-base reference.</td>
</tr>
<tr>
<td><strong>AlertActor()</strong></td>
<td>When the deadline assigned to activity is expired the actor assigned to perform the activity is sent an alert message</td>
</tr>
<tr>
<td><strong>ShallIReAssign()</strong></td>
<td>The activity is checking up whether is preferable to reassign itself to a different person less busy to perform it and if so inform role object for reassignment.</td>
</tr>
</tbody>
</table>

3.4.3. Actor

An Actor can either be a person or a software entity. An Actor can perform an Activity or be responsible for it [10]. Actor WIBOs have the following features:

3.4.3.1. The capability to schedule its Activities  Actors are empowered with knowledge about their working environment. They use this knowledge to schedule
Activities, based on policies such as for example performing the earliest due job or the shortest job first.

The table below describes the functionality of the Actor WIBO.

<table>
<thead>
<tr>
<th>Actor Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The actor’s name</td>
</tr>
<tr>
<td>Location</td>
<td>The actor’s location</td>
</tr>
<tr>
<td>Roles</td>
<td>The set of Roles that the actor could perform.</td>
</tr>
<tr>
<td>RolesExperience</td>
<td>The experience of the actor on each Role.</td>
</tr>
<tr>
<td>Worklist</td>
<td>The Set of work allocated to the Actor.</td>
</tr>
<tr>
<td>ScheduleWork()</td>
<td>The way the actor schedule his/her work i.e.</td>
</tr>
<tr>
<td>FIFO, ShortestFirst etc.</td>
<td></td>
</tr>
</tbody>
</table>

### 3.4.4. Resource

Resources are assets of the organisation that enable processes or are generated by processes. Examples of resources include products, documents, equipment, policies etc.[11]. Resources enable processes, and may result from processes. Resources may be created, destroyed, consumed, reused, owned, or controlled in other way.

<table>
<thead>
<tr>
<th>Resource Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creator</td>
<td>The name of the process creating the resource</td>
</tr>
<tr>
<td>Status</td>
<td>{busy, idle, waiting}</td>
</tr>
<tr>
<td>Creation Time</td>
<td>The time it was created</td>
</tr>
<tr>
<td>Availability</td>
<td>{occupied, available}</td>
</tr>
<tr>
<td>ReportAvailability()</td>
<td>reports whether it is available or not</td>
</tr>
<tr>
<td>CommitSelf()</td>
<td>commits (allocates itself)</td>
</tr>
<tr>
<td>ReleaseSelf()</td>
<td>de-allocates itself</td>
</tr>
</tbody>
</table>

### 3.5. A case study

In this section, a description is given of how the WIBOs discussed above are specialized in the design of a workflow management system for scheduling meetings. In a meeting scheduling process, an initiator is trying to arrange a meeting with a number of participants. The process starts when the initiator sends a message to the potential attendees suggesting possible dates for the meeting, to which they respond with their preferred dates for the meeting. The initiator is then trying to find a set of mutually suitable dates; if that does not prove possible, a new round of negotiations starts, by extending the meeting dates.

```java
class ExtendMeetingDate extends Activity
{
    void AssignActivityToRole(role)
    {
        roleBaseReference = 0
        role.AssignActivityToActor()
        IF deadline expires THEN
            IF ShallIReassign() = YES
```

```java
IF role.AreActorsOverloaded() = YES
    role.DealWithActorsOverload()
ELSE
    role.ReassignActivityToActor()
ELSE AlertActor()
}

class MeetingScheduler extends Process
{
    void processLogic()
    {
        startMeetingProcess.AssignActivityToRole(Initiator)
        FOR all Attendees
            sendPreferencesForm.AssignActivityToRole(Attendee)
        WHILE NOT MeetingDateFixed
            DO
                extendMeetingDatesN.AssignActivityToRole(Initiator)
                FOR all Attendees
                    sendPreference.AssignActivityToRole(Attendee)
            IF MeetingDateFixed
                THEN
                    FixMeeting.AssignActivityToRole(Initiator)
    }

The above fragment of code illustrates the Meeting Scheduler process and ExtendMeetingDate Activity which are specialised from the Process and Activity WIBOs, respectively. The above code shows how WIBO operations are invoked by other WIBOs. WIBOs support dynamic process management through methods such as AssignActivityToRole, and ReassignActivityToRole. This is better shown in the extendMeetingDatesN Activity. Firstly, the roleBaseReference count is set to 0. By setting it to 0, the system will try to assign the next repetition of this Activity to the same Initiator Actor. The Initiator Role assigns the extendMeetingDatesN Activity to an Actor. If it is a second or third repetition of the Activity, the system will preferably use the original Initiator (or assign it to a different Initiator, if the original one is overloaded or unavailable). The Activity is in a waiting state until either the Actor assigned the work carries it out, or the deadline for its completion expires. If the Activity deadline expires before the Actor performs it, then the method is called shallIReassign(). If shallIReassign() returns ‘true’, the Actor is overloaded, and therefore, not able to perform this Activity. The role method AreActorsOverloaded() is called to check the Actors’ workload. If the method returns true, it means that all Actors are overloaded, and therefore, unable to take on extra work. In that case, the
DealWithActorsOverload() method is invoked, which solves the overload problem either by extending the deadline for the Activity or by allocating more workers to the process. If AreActorsOverloaded() is false, another Actor could take over the work, and the role method reAssignActivityToActor() is invoked.

If shallIReassign() is false, then the system sends an alert message to the user. Eventually, when the Activity is executed, the Process will move on to the next Activity in the sequence.

3.6. Workflow intelligent business objects interaction diagram

Fig. 2 shows the interaction between the five fundamental types of WIBOs namely, Processes, Activities, Actors, Roles and Resources in order to execute a workflow.

4. Implementation Of workflow intelligent business objects

The Java language was chosen for implementing WIBOs. For the networking needs of the system, Java RMI was used which allows WIBOs distributed over an Intranet or Internet to exchange messages with each other. The architecture of the WIBO system follows a peer-to-peer philosophy i.e. all different types of WIBOs act both as clients and servers.

4.1. Implementing WIBOs in Java

In our approach, every WIBO is implemented as a Java class. WIBOs attributes and services are coded as Java class attributes and methods, respectively. Fig. 3 shows how the Actor WIBO and its specialisation AttendeeActor (used for implementing a meeting scheduling workflow) WIBO are implemented in Java as Actor and AttendeeActor classes, respectively. The code given in this figure also illustrates the extends keyword used for inheritance. Inheritance allows the AttendeeActor class to inherit the attributes and services from the Actor class’s.

4.2. A peer-to-peer system architecture

The WIBO system architecture is based on a peer-to-peer organisation. The connection between the workflow engine and the workflow client is peer-to-peer in the sense that all WIBOs can act both as clients and servers.

5. Discussion

In the WIBO approach, intelligence, autonomy, collaboration and co-operation reduces the need for human intervention, and therefore, the workers’ workload, to carry out tasks WIBOs utilise encapsulated knowledge. For instance, in the meeting scheduler workflow, knowledge about the various participants is encoded in WIBOs. Such knowledge concerns the hierarchical status of participants, the importance for a participant to attend a particular meeting relatively to other participants or to other meetings, the ease with which a participant can change their availability and their agenda. This knowledge is used in the conflict resolution process when a meeting is being scheduled.

By utilising such knowledge, WIBOs can fix meetings with relatively little human intervention. The user only becomes involved when the WIBO cannot make a decision given its available knowledge and rules. Effectively, the user is relieved from trivial or menial tasks and can afford to spend time on other things of higher complexity. The advantage that WIBOs bring are therefore many-fold. They try to further improve workflow process performance and overcome limitations of current workflow systems by taking advantage of state of the art in object-orientation, Internet/Intranets and intelligent agents. Section 5.1 discusses other related work.
5.1. Workflow process modelling approaches

Most commercial workflow tools use proprietary process meta-models. Two of the most well-known meta-models for workflow by the Workflow Management Coalition and TriGSflow [12] developed by the University of Linz in Austria, are discussed below.

5.1.1. WfMC process definition meta-model

WfMC members do not consider the initial creation of process definitions to be an area of standardisation. Rather, this is considered to be a major distinguishing area between products in the market place. A variety of different modelling and definition tools may be used to analyse, model, describe and document a business process. The workflow meta-model of WfMC is not concerned with the particular nature of such tools nor with how they interact during the build-time process. However, the result of the build-time operation, i.e. the process definition, is identified as one of the potential areas of standardisation to enable the interchange of process definition data between different build-time tools and run-time products [1].

The WfMC process definition meta-model and the proposed meta-model based on WIBOs are complementary. The final output from the workflow process modelling proposed in this paper is a process definition, which can be interpreted at runtime by the workflow engine(s) within the enactment service of WfMC.

5.1.2. The TriGSflow workflow management system

The TriGSflow [12] workflow management system is based on an object-oriented database management system, enhanced with active concepts in terms of roles. In this way, flexible modelling and enactment of business processes is supported, allowing changes even during workflow execution. The TriGSflow differs from our approach primarily in the concepts employed to model the workflow process. The TriGSflow uses rules, objects and roles in contrast to our approach that uses Process, Roles, Actors and Resources. It also differs in the technologies chosen to implement the workflow (i.e. object oriented database management system vs. Java platform).

5.2. Workflow architectures

The most recent trends in workflow management advocate the use of distributed object technology as the implementation infrastructure. Two such approaches are discussed below.

- **Object management group workflow facility** OMG is currently developing the workflow facility. The workflow facility defines interfaces and semantics required to manipulate and execute interoperable workflow objects and their metadata. Ultimately, the workflow facility will serve as a high-level integrating platform for building flexible workflow management applications incorporating objects and existing applications [13].

- **The CORBA-based run-time architecture for workflow management systems** Miller et al. [14] propose five run-time architectures for implementing a workflow management system (WFMS). The architectures range from fully centralised to fully distributed. All the WFMS architectures are designed on top of a CORBA implementation. The CORBA specification defines the architecture of an Object Request Broker (ORB), whose job is to enable and regulate interoperability between objects and applications.

In our approach, Java was chosen instead of CORBA because it is simple, yet powerful, without incurring the costly overhead of an ORB. For the networking needs of the system, Java RMI was used which is a lightweight Java communication protocol.

5.3. Intelligent agents in workflow

At present, there are two primary uses for intelligent agents in a workflow system; to provide assistance for individuals and workgroups (e.g. sorting newly arrived work) [15,16] and to continually watch for information that meets a specified criteria and then do something (e.g. send a fax or retrieve information from a database). Artificial intelligence and expert systems are employed to achieve this [3]. Workflow vendors are currently trying to develop more intelligent workflow systems by employing collaborative agents.

5.3.1. Agent-based workflow (BT Labs)

The BT laboratories are currently developing agent-based workflow (ABW). ABW automates the collaboration...
between different resource units and assigns resources automatically. Agents negotiate with other agents to buy and sell services, automating what is usually done by middle managers. ABW makes sure that the end-to-end business process is suitable [17].

In this paper, intelligent agent technology is employed to automate the collaboration between WIBOs. WIBOs negotiate and communicate with other WIBOs autonomously, automating what is usually done by users. Collaborative agents emphasise autonomy and co-operation (with other agents), in order to perform tasks for their owners. Because WIBOs are Intelligent Agents that can assist people and act on their behalf, they have the following advantages:

- **Improve productivity and reduce cost** WIBOs can do things on user’s behalf; that leads to further automation of the workflow process and less user contribution to the process.
- **Hide complexity** WIBOs do the “hard work” for the user. Users can let their WIBOs do complicated jobs like trying to fix meetings by considering different people agendas.
- **Help users collaborate** WIBOs act as an intermediary for processes. Users can let their WIBOs negotiate for things like meeting times.
- **Monitor events and processes** WIBOs can be programmed to inform the users about events, and raise signals if an exception occurs.

WIBOs’ features such as autonomy, however, can be a mixed blessing. As with other work automation approaches, the introduction of WIBOs can have numerous cultural effects on the organisation such as information overload, the introduction of new levels of control and ultimately diminished interest of the workers in the process. We believe, however, that the advantages of WIBOs outweigh the disadvantages as their introduction in a suitable organisation culture can lead to genuine employee empowerment.

5.4. Dynamic and flexible workflow

WFMS such as those discussed below, although they support dynamic or flexible process definition changes, do not take in to consideration other important workflow components such as roles, agents, resources etc. In contrast, our approach supports dynamic process management, i.e. dynamic allocation and reallocation of work and resources.

5.4.1. Dynamic change within workflow systems

This approach is concerned with dynamic change to procedures in the context of workflow systems [4]. The system is able to change “on the fly” in the midst of continuous execution of the changing procedure (change such as deletion of a procedure Activity etc.).

5.4.2. Workflow management systems adaptation and negotiation

Herrmann [18] supports that characteristics of WFMS must be adaptable by the users themselves, according to the dynamism of organisational structures and of the conditions of co-operative tasks performance. Adaptations which are initiated by individuals also affect other users in most cases. Therefore, a process of negotiation is required that can help users affected by adaptations to comment on them or to reject, accept or modify them. In this WFMS, adaptation based on negotiability can be used to make those structures more flexible.

5.4.3. Hypermedia approaches

CHIPS [5] (Co-operative Hypermedia for Process Support) is a co-operative hypermedia based process support system aimed on flexible business processes. It uses hypermedia based activity spaces to model the structural, relational and computational semantics of both individual tasks and processes. It addresses the analysis of the communication, co-ordination and co-operation requirements of business processes by extending a co-operative hypermedia system with a process support.

5.5. Workflow over the Internet/Intranets

Intranets challenge the dominance of workflow. Like Intranets, workflow and groupware tools let end-users share information in a central repository and support electronic discussion groups. It is predicted that when the large number of Intranet applications becomes available off-the-shelf (or off-the-Internet), the traditional groupware or workflow will be toppled from their throne. However, it is unlikely to disappear completely, especially when considering the investment many corporations have already made in the technology [19].

Many workflow systems provide interfaces, but use other mechanisms for underlying communication distribution (e.g. sockets, RPC, CORBA). Systems that provide Web interfaces are called Web-enabled, while systems with Web technology, as the only infrastructure used to build the workflow system, providing both interfaces and communication/distribution, are called Web-based [20]. A survey of existing commercial workflow systems that use Web technology revealed that a majority of them are merely Web-enabled. But the trend appears to be to add more and more Web capability to workflow systems [10].

5.5.1. The METEOR workflow Management system

The METEOR project at the University of Georgia is based on the WebWork, a purely Web-based workflow management system, the OrbWork, a primarily CORBA based system with Web interfaces, and the fact that WebWork and OrbWork will conveniently interoperate. CORBA provides many advantages compared to Web technology, as well as some disadvantages. However, as Web
technology evolves to include CORBA-like capabilities some of these advantages will likely diminish in the future. In addition, with capabilities like IIOP and Java to CORBA interfaces, the two technologies can be combined where ever needed. [20]

This paper shares the above view i.e. that Internet/Intranet will be the future technology for implementing workflow systems. For this reason, WIBOs have been developed using a Web technology namely the Java platform and are totally Internet/Intranet capable.

6. Conclusion

In conclusion, this paper proposes WIBOs as the paradigm for designing and implementing WFMS. The WIBO approach promises improvements in productivity and efficiency in comparison to the current generation of WFMS. Further research along this line could address issues of interoperability between WIBOs and other groupware and workflow systems (in particular Internet-based ones). Such interoperability should be based on existing (e.g. WfMC) and emerging standards (e.g. OMG).

References