BUILDING A WORKFLOW ENACTMENT SERVICE FOR TELEWORK CO-ORDINATION

Diogo Ferreira[a], João Rei[a], José M. Mendonça[b], J. J. Pinto Ferreira[c]
[a]FEUP, [b]FEUP-DEEC / ADI, [c]FEUP - DEEC / INESC - UESP
Rua José Falcão 110, 4000 Porto, Portugal
Phone: +351 2 209 43 00   Fax: +351 2 200 84 87
E-mail: jjpf@fe.up.pt

Key words: Electronic Commerce and Internet Business Applications, Systems Integration: Modelling Concepts and Information Integration Tools, Project Management

Abstract: In its on-going effort to define, specify and build a telework co-ordination system, the Telework Interest Group at FEUP[1] - DEEC[2] has realised the need for a workflow management system that must be able to support business processes that rely on geographically distributed co-operative work. Telework is an innovative form of work organisation for decentralised or information-based organisational structures whose tasks are independent of their location of execution. However, this organisational practice demands an efficient business process co-ordination or, to be more specific, demands a workflow management system. The work we intend to present is a prototype of the workflow enactment service which is a core component of the management system whose construction is the ultimate goal of the Telework Interest Group[1,2]. The workflow enactment service, that is currently being built, is a software service that contains a workflow engine capable of creating, managing and executing workflow instances.

1. INTRODUCTION

The Telework Interest Group (GIT) was formed in September 1997 and since then some major steps have been taken towards the construction of a Telework Co-ordination System. Along this project, the Telework Interest Group has focused its efforts on the so-called "small information-based organisations" where all activities are concerned with information processing and transfer, usually among sub-contracted teleworkers. In this scenario, a company would run by managing several parallel processes, maybe several instances of the same business process, each requiring remote task execution by teleworkers. The GIT promptly recognised the need for a workflow management system supporting geographically distributed co-operative work (telework). That workflow management system should be implemented through a workflow enactment service, i.e. a software service capable of creating, managing and executing telework-related workflow instances.

The workflow enactment service for telework co-ordination should comprise the following components: (1) a modelling tool, able to provide a computer representation of the workflow logic which in turn shall drive the process execution during run time; (2) a workflow engine, providing the run time execution environment for each process instance; (3) a messaging system, allowing asynchronous communication with teleworkers over a communication infrastructure and (4) a

[1] Engineering Faculty of Porto University
[2] Department of Electrical Engineering and Computers

workflow client application, providing an interface between the workflow engine and the teleworker and possibly also some form of managing the teleworker's obligations. Figure 1 ilustrates the scope and relationships between these main components [Lawrence, 1997].

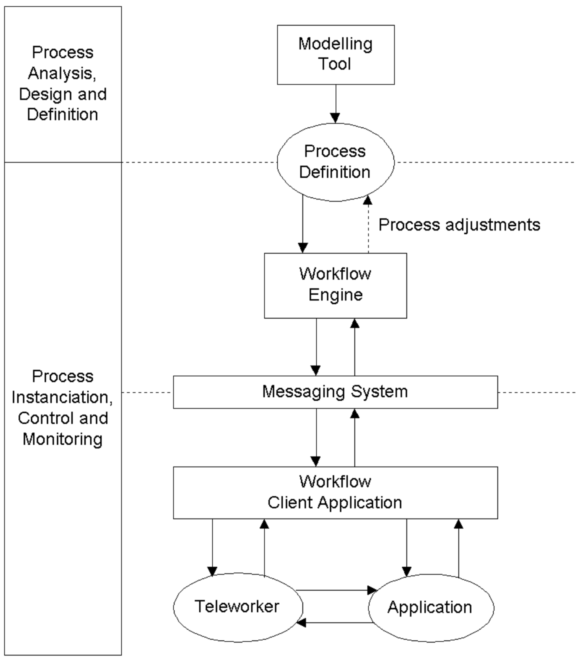
Process Analysis, Design and Definition
Modelling Tool
Process Definition
Process adjustments
Workflow Engine
Process Instanciation, Control and Monitoring
Messaging System
Workflow Client Application
Teleworker
Application

Figure 1. Scope of the enactment service components

Even though the process definition is what drives the execution of the workflow, there will be the need to dynamically adjust the process instance properties either because of missed deadlines, task re-assignment or other unpredictable circumstances.

However, since the modelling phase and execution phases do not overlap, we find it convenient to merge the modelling tool and the workflow engine inside the same software service. In this way, the same computer representation supports planning and controlling of the process within the same environment, therefore making it easier to adjust the process definition during run time.

Thus, we have reduced our workflow enactment service to three components: (1) the enactment service engine, which comprises the modelling tool and the workflow engine; (2) the messaging system which, over a communication infrastructure, allows triggering and managing the tasks' execution and (3) the workflow client application which provides, over the messaging system, the interface between the workflow engine and the teleworkers or applications; in the case of a teleworker the workflow client application should also provide some means of managing the teleworker's obligations, as stated before.

In the remaining of this paper, we shall discuss each of these components.

2. THE ENACTMENT SERVICE ENGINE

2.1. Specifying the Enactment Service Engine

The enactment service engine comprises a modelling tool and a workflow engine which drives the execution of the process in accordance with the process representation that was achieved using that modelling tool.

A business process is often represented as an activity network, each activity demanding the services of one or more functional entities, e.g. teleworkers, in order to accomplish an overall business goal. Even though each activity is to be carried out by an appropriate functional entity, the process definition refers to organisational entities and role functions rather than specific participants. Each activity in the activity network of a business process stands for a particular operation that must be executed by a single functional entity and whose subdivision into smaller activities is of no interest.

Upon instantiation of the process, each activity is assigned to a particular functional entity, i.e. to a particular teleworker, which is to comply with the activity demands, transforming an input state into an output state. In figure 2 is depicted a simple activity network. The workflow engine is the facility responsible for managing the execution of all process instances.

The enactment service engine which is under construction should accomplish a two-fold purpose: representation of the activity network concerning the process which is to be instantiated and managing the execution of that process instance.

In order to fulfil the first purpose, the enactment service engine must contain a process editor allowing the process definition, i.e. the creation of the actual activity network and its refinement with the pertinent data of each activity. On its own, each activity is a self-contained module or construct that cannot be dissociated from its data such as: (1) name or identification number; (2) name or identification of the process to which it belongs; (3) launch and deadline dates; (4) entry and exit criteria; (5) description; (6) input and output files or data; (7) needed expertise and (8) assigned teleworker.

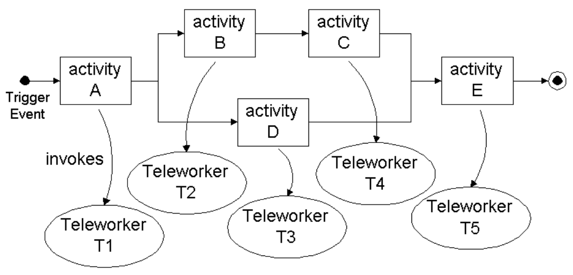

Figure 2. Representation of a business process

Activities are to be connected in their order of precedence between each other.

During its life-cycle, an activity passes through different states and possibly ends up completed. We say possibly because there may be times where, depending on some sort of condition particular to the business process, the execution of other activities may be preferred. For example, if the purpose of one activity is to calculate a budget, then the execution of the following activities may be dependent on its result. These types of conditions are to be part of the entry and exit criteria of each activity. For a large number of activities, however, their entry condition may only depend on the completion of the preceding ones.

Figure 3 depicts the state diagram or dynamic model of an activity, using the syntax of [Rumbaugh, 1991].

The state diagram shows that, when created, an activity starts in an "inactive" state waiting to be assigned to a teleworker. After being assigned, and even during execution, the activity may be re-assigned. Before the activity is put under execution, the possibility of re-assignment allows negotiation to take place, although we are not concerned with the nature of the contract celebrated with the teleworkers; this means that we're trying to maintain a high degree of flexibility and organisational structure independence. Our interest is indeed focused on the workflow enactment. During execution and in the presence of adverse circumstances, re-assignment may be the last resort to employ in an attempt to complete the activity.

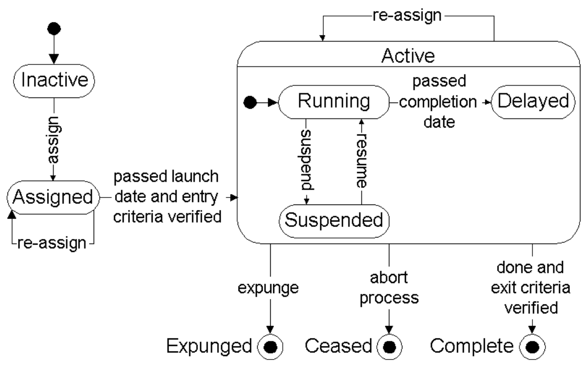

Figure 3. Dynamic model of an activity

At any time during the execution of the activity, the process, or the activity itself, can be interrupted; in that case, the "suspended" state becomes the state of the activity which will remain idle until the process is resumed. For delayed activities, however, we do not allow a state of interruption; because the activity may be suspended temporarily, acknowledging an interruption to the teleworker would serve no purpose other than increasing the completion delay. We are assuming, of course, that the teleworker will be notified, during the predicted period of execution (but not under delay), if the execution of the activity is to be interrupted.

In general, the life-cycle of an activity ends when the teleworker has completed his/her job and the exit criteria become verified.

Nevertheless, there may be occasions when an activity is abruptly terminated: the process to which the activity belongs is aborted – activity becomes "ceased" – or the activity is simply expunged from the activity network. The final state reflects the cause of such termination.

Although we have taken care to be able to cope with unpredictable circumstances or with deviations from the planned course of actions, we have not yet stated clearly who is in charge of acting towards solving these problems. A human co-ordinator will be the one whose intervention shall be requested in order to solve the difficulties that may arise. He or she should be qualified and responsible for decision-making such as: (1) choosing the teleworker, based on his/her expertise and availability, to carry out a given activity and assign him/her to that activity; (2) negotiating with alternative teleworkers the execution of some activity and re-assign that activity; (3) evaluating complaints of the teleworkers (possibly related with the work of each other) and taking the appropriate actions upon that evaluation and (4) terminating or suspending activities or process instances.

Just as an activity, a process also has a life-cycle. In the first place, a process should be defined by analysing, identifying and characterising its different components and by proposing a plan of action for its execution, taking into consideration existing constraints. As soon as the process definition becomes available, the computer representation of the process can be constructed using the modelling tool that is part of the enactment service engine. To facilitate comprehension and provide some structure to that representation, the modelling tool should allow nesting of sub-processes into processes or other sub-processes in a hierarchical perspective. As [Kerzner, 1998] suggests, to further enhance the perspective of the activity network a Gantt chart should also be used to emphasize the temporal relationships between activities (though [Kerzner, 1998] also points out the pitfalls of using a Gantt chart).

Upon instanciation, each activity is assigned to a teleworker and the workflow engine starts the process execution. A process also has a "suspended" state which causes all running activities to be suspended (all but the delayed ones) and at any time during execution, a process instance can be terminated which causes all running activities to be "ceased". A process is said to be "complete" when there are no remaining activities left to be executed.

Clearly, the enactment service engine must rely on some communication infrastructure in order carry out the execution of each process instance. The communication needs of the engine are the following: (1) requesting the execution of an activity from a teleworker; (2) receiving an answer from that teleworker expressing acceptance or rejection; (3) enquiring teleworkers about work throughput; (4) issuing alerts for missed deadlines; (5) receiving complaints from teleworkers; (6) receiving acknowledgements for completed activities; (7) requesting correction or revision of work from a teleworker; (8) exchanging files and/or other types of data and (9) informal communication between co-ordinator and teleworkers.

2.2. Implementing the Enactment Service Engine [3]

To implement the enactment service engine we chose the Microsoft Windows (95, 98 or NT) environment and the C++ programming language with the Microsoft Foundation Classes for the following reasons: (1) the enactment service engine would benefit from a user-friendly, well-established graphic user interface, for both process editing and monitoring; (2) Microsoft Windows already supports workflow applications through its Messaging API (MAPI); (3) the Microsoft Foundation Classes (MFC), which constitute an extensive set of C++ classes, support all aspects of Windows programming and provide a fully object-oriented development framework and (4) previous knowledge and experience favoured

[3] **Microsoft**, **Windows** and **Visual C++** are registered trademarks of Microsoft Corporation

the usage of C++; we also had some experience working with Microsoft Visual C++ and the MFC.

Because the enactment service engine relies heavily on the ability to communicate with the teleworkers, we must also decide what communication infrastructure shall be used and whether or not the development tool supports it. Keeping in mind the intention to reach the broadest range of teleworkers geographically distributed and the need to transfer binary data, we should use a widely accepted asynchronous communication infrastructure: something just like the electronic mail. That decision, however, should not prevent us from using other internet resources and transferring data through FTP, for example. In fact, Microsoft Visual C++ offers extensive support for internet technologies either by the Windows MAPI, MFC internet support classes, WinInet classes or even by low-level windows sockets. Any teleworker should be able to choose if he/she wishes to receive and send his/her data through e-mail attachments or through FTP connection to a server.

3. THE ENGINE MESSAGING SYSTEM

3.1. The Message Format

In the last section we emphasized that the workflow engine should carry out the execution of the process instances on its own, requesting if necessary the intervention of a human co-ordinator. That is, we are looking forward to automate the task of putting the various process instances under execution maintaining, however, a wide range of aplicability regarding the business processes which could benefit from this workflow enactment service. Because the workflow engine generates and receives e-mail messages automatically, there has to be a pre-defined message format to convey the necessary information in both directions. The following message format has been agreed upon.

Every message should include a keyword that identifies its type and possibly its purpose. The keyword appears on the "subject" field of the e-mail message, following a unique string of characters that identifies this message as being telework-related. That identifying string is "TLW" (from TeLeWork); after this string and an arbitrary number of space or tab characters, the keyword is placed. Figure 4 illustrates the message format.

```
From:      coordinator@somewhere.com
To:        teleworker@somewhere.com
Subject:   TLW keyword
______________________________________________
#Company      Telework Company Name
#Process      Process ID
#Activity     Activity ID
#Startdate    dd/mm/yy hh:mm
#Finishdate   dd/mm/yy hh:mm
#Status       Activity Status
#Description  ...........................................
.......................................................…
#Inputdata
_FILES        file1 ; file2 ; file3 ; …
_SITE         ftp.com
_USER         username
_PASS         password

_FILES        file4 ; file5
_ATTACH
…
#Outputdata
_FILES        file4 ; file5
_ATTACH

_FILES        file1 ; file2 ; file3 ; …
_SITE         ftp.com
_USER         username
_PASS         password
…
#Text      ...........................................
.......................................................…
```

Figure 4. Message format

In the body of the message, several tags (similar in appearance to C/C++ preprocessor directives) indicate the presence of pertinent data; its use is self-explanatory. Although figure 4 lists all possible tags, no message needs to contain all tags; any tag should be used if and only if its corresponding data is available and is of interest. This rudimentary set of tags should be enough to cover all our needs.

There are special tags (_FILE, _SITE, _USER, _PASS and _ATTACH) whose purpose is to deal with file input and output. The tag _ATTACH means that the preceding files are included as attachment to this same message; otherwise the username and password are specified for download from a given server. Following the tags #Descritpion and #Text any text excerpt may appear; in particular, the tag #Text may be used for any unspecified communication purpose between co-ordinator and teleworker or vice versa.

The appearance of some tags is somewhat related with the keyword on the "subject" field. Moreover, some keywords are used in messages from the workflow engine (or co-ordinator) towards the teleworkers – request, warning and reply – while others appear in messages that flow in the opposite direction – accept, reject, done, status and problem. Some keywords – complaint and informal – may appear in either way. The possible keywords may be summarized as follows: (1) request: the workflow engine requests the execution of an activity from a teleworker; (2) warning: the workflow engine acknowledges the teleworker of some change to the properties of the activity that he/she has been assigned; (3) complaint: the teleworker expresses dissatisfaction regarding the work performed by a preceding colleague; this keyword is also used by the co-ordinator to inform the preceding teleworkers of those problems; (4) problem: the teleworker is experiencing some kind of problem that is not related with the work of any other colleague; (5) reply: the co-ordinator informs a teleworker that the problems have been solved and that he/she may resume his/her work; (6) informal: used to exchange messages that are not to be parsed or interpreted and whose contents should reach the receiver without modification; (7) accept: the teleworker compromises him/herself to carry out the requested activity; (8) reject: the teleworker refuses to assume responsibility for executing the requested activity; (9) done: the teleworker reports to the workflow engine or co-ordinator the completion of his/her activity and finally (10) status: the teleworker retrieves information regarding the execution of the activity.

3.2. The Messaging Protocol

In the following discussion, messages will be referred to by their keyword. When the process is instantiated, teleworkers will have to be acquainted with the activities that they have been assigned; depending on the nature of the contract celebrated with the teleworkers, it may follow a negotiation phase or not. In any case, teleworkers should acknowledge the request arrival by answering "accept" or "reject". The appropriate time to request the execution of an activity may depend on the expected duration of the execution of the process. To illustrate this, we have envisaged two possible scenarios: (1) if the execution of a process instance is expected to take several months, then maybe it should be appropriate to request execution of an activity two weeks before its launch date, allowing the teleworker to manage his/her obligations or allowing for some sort of negotiation; (2) if the execution of a process instance is to take a couple of days, then maybe two or three days before launching the process into execution all teleworkers should be aware of their duties in order to avoid any execution delay.

Thus, the co-ordinator must choose the right time to request the execution of each activity, after which he/she should expect an "accept" or "reject" answer from the corresponding teleworker. In case of a rejection, further requests may be sent to other teleworkers though, once again, those details do not concern us; our aim is to provide the most flexible means to fulfill any co-ordinator's needs.

The co-ordinator may as well define how regularly a teleworker should report to the workflow engine the status of execution of his/her activity.

When an activity becomes delayed, the teleworker should be reminded of that fact. Once again, there should be an option on the enactment service engine allowing the co-ordinator to specify how regularly the teleworker should be reminded of the delay. Notwithstanding, a certain delay – the "slack" when speaking in terms of PERT/CPM – of some activities may not compromise the completion date of the entire process if those activities do not belong to the critical path of the process. Here the co-ordinator may choose between two different policies: (1) letting the teleworker know the latest time for completion of his/her activity or (2) letting the teleworker know only the earliest time for that same completion; in this case we can afford a delay no longer than the slack of the activity, if the preceding activities are to be completed on time.

The messaging protocol becomes highly useful when a teleworker issues a complaint about a colleague's work. With #Inputdata, the teleworker specifies the offending files and the workflow engine will forward the complaint message to the immediately preceding teleworkers whose activity dealt with those files. The #Text directive and all that follows after it shall also be forwarded to the preceding teleworkers, letting them know of the reasons for dissatisfaction. Those preceding teleworkers should then answer with a "reply" message that contains the corrected data which will be forwarded to the teleworker that issued the complaint. This sequence of events is depicted on figure 5 under a Message Sequence Chart [van der Aalst, 1998].

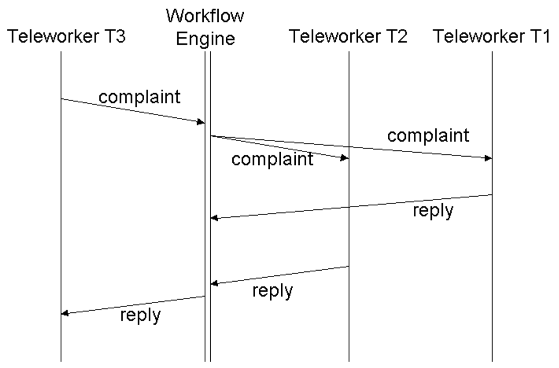

Figure 5. Sequence of events after a complaint

This is a peaceful scenario; teleworkers may as well start disagreeing about each other's work. The co-ordinator, however, is witnessing all this situation and is free to intervene whenever appropriate; if not, all the co-ordinator has to do is to consult the log file that the workflow engine maintains to be aware of the situation.

4. THE WORKFLOW CLIENT APPLICATION

4.1. Specifying the Workflow Client Application

In the preceding sections, we have defined the communication infrastructure that we shall use – e-mail and FTP – and the message format of the workflow engine. We must not expect, however, that every time the teleworker wishes to send a message he/she should choose the right keyword and include the appropriate tags and text. In addition, when receiving an incoming message, the teleworker shouldn't have to interpret its contents. Although the message format is quite evident, there should be some means of interpreting the message and presenting its contents to the teleworker in a user-friendly way. Therefore, the workflow client application is basically an special purpose e-mail client that identifies a telework-related message by looking at its subject field and interprets, or more precisely, parses its content so as to present it in a meaningful way.

The same e-mail client should also provide the reverse functionality: when a teleworker wishes to send a message he/she specifies the type of the message, which is related with the keyword, and introduces its content disregarding tags or other format details. The application will then generate and send the message with the appropriate keyword and tags, so that it can be promptly understood by the workflow engine.

Besides this interface role, the application should implement some means of managing the teleworker's tasks, further allowing some kind of time management facility. That could be done with an enhanced version of the classic Gantt chart, of which an example is depicted in figure 6. The enhancement is the presence of a bottom row that sums up the hours required to fulfill the foreseen tasks for a given day (on the example of figure 6 the teleworker has some real busy days!). It should be noted that the various activities depicted on figure 6 may represent several commitments that the teleworker has established with different telework enterprises.

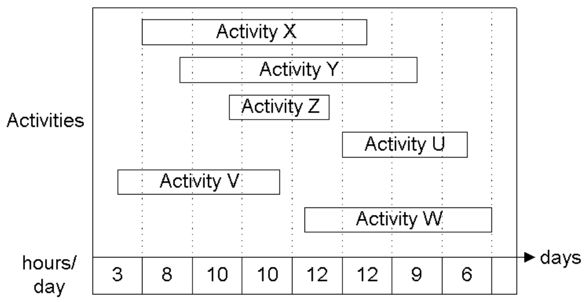

Figure 6. Enhanced Gantt chart

4.2. Implementing the Workflow Client Application [4]

Because each teleworker might have his/her preferred working environment, the key issue about this workflow client application is platform independence. To implement this application we decided to use the Java programming language with the Java Foundation Classes for the following reasons: (1) the Java programming language and its "virtual machine" provide a high degree of platform independence; (2) the Java Foundation Classes (JFC), which constitute an extensive set of Java classes, support several aspects of Java programming and provide a fully object-oriented development framework; (3) the Java Development Kit (JDK), Sun's Java development tool, is freely available from Sun Microsystems and the Java 2 (formerly known as Java 1.2) platform has been released recently by Sun Microsystems with a complete implementation of the JFC and (4) already some knowledge and experience existed working with Java and the Java Foundation Classes.

[4] **Sun** and **Sun Microsystems** are registered trademarks and **Java** is a trademark of Sun Microsystems, Inc.

5. CONCLUSION

This paper illustrated our approach to the construction of a workflow enactment service supporting the co-ordination of decentralised activities over the Internet.

We have concluded that the enactment service comprises three main components – (1) the enactment service engine, (2) the messaging system and (3) the workflow client application – which we have discussed separately though, as we have shown, they are intimately related.

Although our aim was not to build a commercial software product, we have attempted to construct a prototype of an enactment service flexible enough to apply to the broadest range of business processes that involve information processing and transfer. Hopefully, we shall have the opportunity to show a glimpse of the enactment service to the audience of the 1st International Conference on Enterprise Information Systems.

REFERENCES

van der Aalst, W. M. P. 1998, *Interorganizational Workflows*, Proceedings of the Tenth International IFIP WG 5.2/5.3 Conference PROLAMAT 98, Trento, Italy

Kerzner, H. 1998, *Project Management: a systems approach to planning, scheduling, and controlling,* Sixth Edition, Van Nostrand Reinhold, New York

Lawrence, Peter 1997, *Workflow Handbook 1997*, John Wiley & Sons, ISBN 0-471-96947-8

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. 1991, *Object-Oriented Modeling and Design*, Prentice-Hall International Inc., Englewood Cliffs, New Jersey

Silva, J. A. & Ferreira, J. J. Pinto 1998,*From Telework Project Planning to Project Co-ordination, An integrated Approach*, IFIP International Conference PROLAMAT '98, Trento, Italy

Vernadat, F. B. 1996, *Enterprise Modelling and Integration, Principles and Applications*, Chapman & Hall, London