A Workflow Management System for Coordinating Distributed Information-Based Business Processes

Diogo Ferreira[a], J. J. Pinto Ferreira[b]

[a]FEUP/INESC, [b]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

Abstract

In its on-going effort to define, specify and build a telework co-ordination system, the Telework Interest Group (GIT) at FEUP[1]-DEEC[2] has realized 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 organization for decentralized or information-based organizational structures whose tasks are independent of their location of execution. However, this organizational practice demands an efficient business process co-ordination or, to be more specific, demands a workflow management system.

The initial goal of our group was to develop a prototype of a workflow enactment service for telework coordination. That prototype, however, has turned out to be a sufficiently generic tool capable of coping with a broad range of distributed information-based processes.

In this paper, we present the main components of the software service we have developed, we discuss its principles and its simple architecture and we offer some insight into how it may be used as a workflow management system for coordinating distributed information-based business processes.

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 organizations" 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 concurrent processes, maybe several instances of the same business process, each requiring remote task execution by teleworkers.

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

As reflected in our earlier approaches such as [Silva and Ferreira, 1998] the GIT promptly recognized the need for a workflow management system and has established the development of such system as its ultimate goal. The starting point was to specify and implement a workflow enactment service, i.e., a software service capable of creating, managing and executing telework-related workflow instances. The workflow enactment service [Ferreira et al., 1999] would be a core component of the workflow management system.

As we will show, we have added to that enactment service some modeling features, a messaging system and the support of a client application. The end result is that we have surpassed the scope of the enactment service and we've come closer to the actual management system we intended to build. Also, we have attempted to maintain flexibility through simplicity and, as a consequence, the resulting software service is not bound to telework purposes. In fact, we believe that it may serve as well any distributed information-based business process.

2. The Process Definition

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 organizational 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 such as a particular teleworker, which is to comply with the activity demands, transforming an input state into an output state. In figure 1 is depicted a simple activity network.

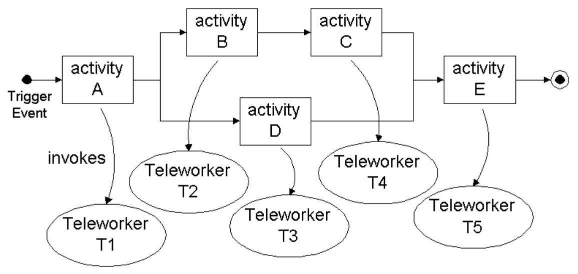

Figure 1. Representation of a business process

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.

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

During its life cycle, an activity goes through different states and possibly ends up completed. We say possibly because there may be times when, 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 this result. These types of conditions are to be part of the entry and exit criteria of each activity. In order to accomplish this feature we introduced the concept of state variables. A "state variable" stands for a numeric attribute or state that is known to all process activities. Some activities may determine the value of any state variable while other activities will rely on that value as their entry criteria. In the context of the preceding example, we could define a state variable named "budget" that carries a value set by a particular activity and which is used as entry condition of others. For a large number of activities, however, their entry condition may only depend on the completion of the preceding ones.

Figure 2 depicts the state diagram or dynamic model of an activity, using the syntax of [Rumbaugh, 1991].

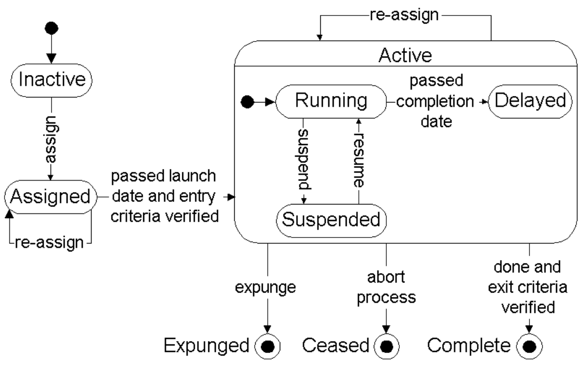

Figure 2. Dynamic model of an activity

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 organizational 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.

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.

3. The Workflow Management System Components

The workflow management system for telework coordination should comprise the following components:

(1) a modeling tool in the form of a business process editor, able to provide a process definition, i.e., a computer representation of the workflow logic which 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 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 3 illustrates the scope and relationships between these main components [Lawrence, 1997].

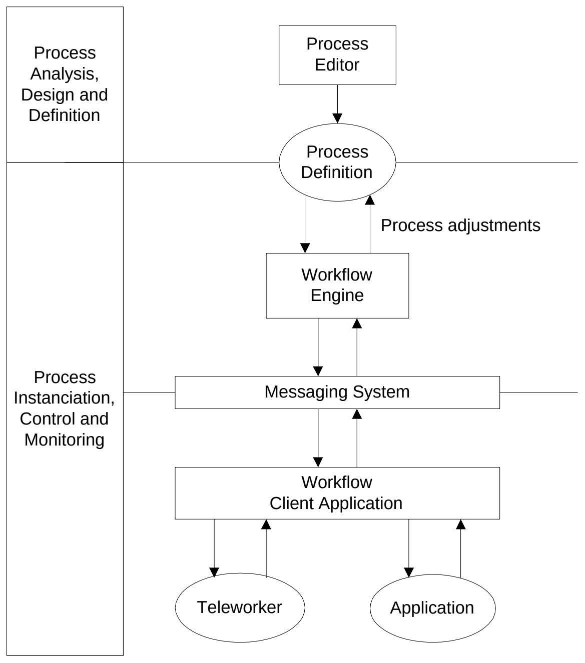
Process Analysis, Design and Definition
Process Editor
Process Definition
Process adjustments
Workflow Engine
Process Instanciation, Control and Monitoring
Messaging System
Workflow Client Application
Teleworker
Application

Figure 3. Scope of the management system components

The management system contains a business process model editor allowing its definition, i.e., the creation of the actual activity network and its refinement with the pertinent data of each activity.

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 modeling phase and execution phases do not overlap, we found it convenient to merge the process editor 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.

The workflow engine iterates through each activity in accordance with the process definition, triggering and managing each task execution by sending and receiving messages and data over a communication infrastructure.

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 coordinator 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.

3.1. The Process Editor

Just as an activity, a process also has a life cycle. Besides the inactive and running states, 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 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.

But in the first place, a process should be defined by analyzing, identifying and characterizing 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 process editor that is part of the management system. To facilitate comprehension and structure to that representation, we provided the process editor with the capability of nesting sub-processes into processes or other sub-processes, producing a hierarchical perspective of the activity network. Also, to further enhance the perception of the time relationships between activities, we have provided a Gantt chart view for each sub-process. Figure 4 illustrates the hierarchical and Gantt chart views with an example.

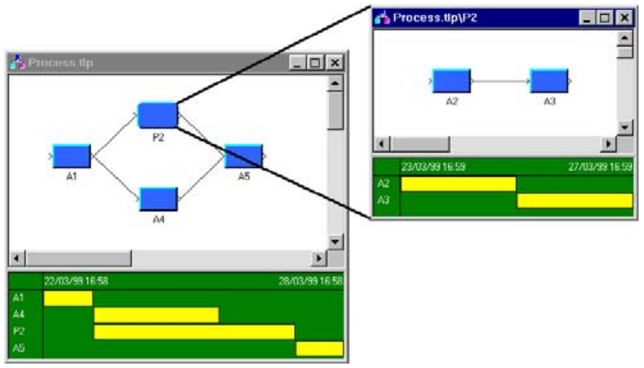

Figure 4. Enhancements to the process representation

To provide reusability, we introduced two types of process representations: the "process template" and the "process instance". Although similar when observed from within the process editor, they are conceptually different: a process template cannot be put under execution whereas a process instance, the one that can be released for runtime execution, must be obtained by instantiating a process template. The idea is that the process definition should be created as a process template, remaining as a reusable representation of a particular activity network. From a process template, we may obtain as many instances as we wish and put them under execution concurrently. Notwithstanding, instances of the same process template may present differences since it is still possible to edit the activity network even after it has been instantiated. Also after instantiating, direct changes in the process template may be reflected onto its instances or not, depending on the user's choice. When a particular instance is put under execution, however, the link to its parent template is broken, and that process instance begins a life of its own. From that moment, it is no longer possible to modify that process representation beyond some restricted adjustments such as re-assignment and rescheduling of activities.

The coordinator environment that comprises the process editor and the workflow engine was implemented as a Microsoft[3] Windows[3] (95, 98 or NT) application using the C++ programming language and the Microsoft Foundation Classes for the following reasons: (1) the process editor 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 supporting all aspects of Windows programming and providing a fully object-oriented development framework and (4) previous knowledge and experience favoured the usage of C++; we also had some experience working with Microsoft Visual C++[3] and the MFC.

3.2. The Workflow Engine and the Messaging System

Clearly, the workflow 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) inquiring 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
(9) informal communication between coordinator and teleworkers.

Because the enactment service engine relies heavily on the ability to communicate with the teleworkers, we had also to decide what communication infrastructure should be used and whether or not the development tool supported it. Keeping in mind the intention to reach the broadest range of teleworkers geographically distributed and the need to transfer binary data, we chose a widely accepted asynchronous communication infrastructure: the e-mail. Also, we have allowed transactions through FTP whenever it becomes more convenient, such as when transferring large amounts of data that would be inappropriate to attach to an e-mail message.

We have already 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 coordinator. That is, we are looking forward to automate the task of putting the various process instances under execution maintaining, however, a wide range of applicability regarding the business processes which could benefit from this workflow management system. 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 5 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;…
_ATTACH
…
#Outputdata
_FILES        file3; file4; file5; …
_SITE         ftp.somewhere.com
_USER         username
_PASS         password
…
#Statevar       variable1; variable2; …

#Text     ..........................................………
```

Figure 5. 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 5 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

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

rudimentary set of tags should be enough to cover all our needs.

There are special tags (_FILES, _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 #Description and #Text any text excerpt may appear; in particular, the tag #Text may be used for any unspecified communication purpose between coordinator and teleworker or vice versa. The purpose of the #Statevar tag is to enumerate those process state variables which will be assigned a new value after the activity is complete. 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 coordinator) 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 coordinator 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 coordinator 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 coordinator the completion of his/her activity and finally

(10) status: the teleworker retrieves information regarding the execution of the activity.

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 coordinator 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 coordinator's needs.

When an activity becomes delayed, the teleworker should be reminded of that fact. 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 coordinator 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 6 under a Message Sequence Chart [van der Aalst, 1998].

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

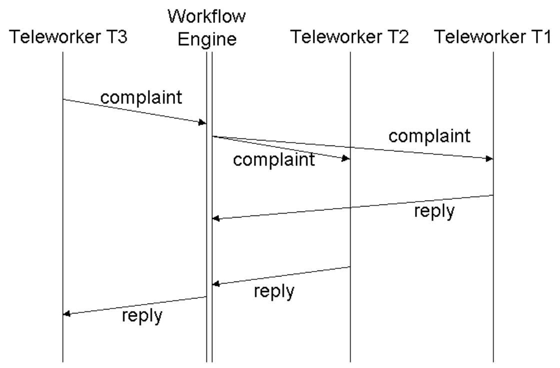

Figure 6. Sequence of events after a complaint

3.3. The Workflow Client Application

In the preceding section, 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 a 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 also provides 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 the workflow engine can promptly understand it.

Besides this interface role, the application also implements some primitive means of managing the teleworker's tasks, further allowing some kind of time management facility.

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 chose the Java[4] programming language for the following reasons: (1) the Java programming language and its "virtual machine" provide a high degree of platform independence; (2) the Java Development Kit (JDK), Sun's Java development tool, is freely available from Sun Microsystems[4] and (3) already some knowledge and experience existed working with Java.

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

3.4. Architecture Overview

Figure 7 illustrates the architecture of our workflow management system, which has been implemented as an integrated set of two lightweight software applications: the enactment service (that comprises the process editor and the workflow engine) and the workflow client.

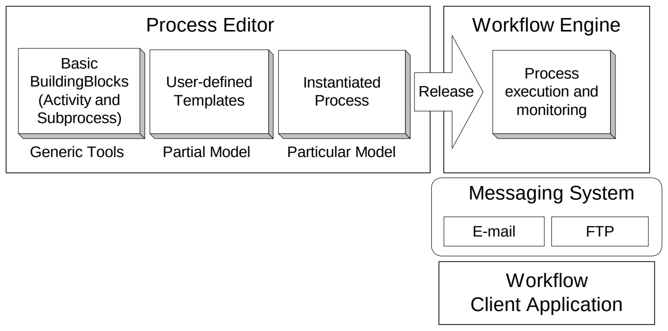

Figure 7. Architecture of the management system

4. Towards a Multi-Purpose Coordination Tool

While presenting the management system components, we have focused mainly on telework coordination because we were assuming that our functional entities were restricted to teleworkers. But as far as our management system is concerned a teleworker remains as being someone whose work is independent of location and time, the only restriction being a pre-established deadline. Our management system, we believe, is still able to support the coordination of any distributed process that concerns information processing and transfer among a group of people geographically.

In the future, we intend to make the enactment service capable of interfacing not only teleworkers but also other applications or application servers. To that end, we are evaluating the possibility of developing a new version of the management system using the Common Object Request Broker Architecture (CORBA). We are also considering the enhancement of the present version with special-purpose workflow clients that would interface other applications (instead of teleworkers) allowing us to maintain the same messaging system.

5. Conclusion

This paper presented our approach to the construction of a workflow management system supporting the coordination of decentralized activities over the electronic mail messaging infrastructure. Although our initial goal was to develop a workflow enactment service as the first step towards a workflow management system for telework coordination, we believe that our software service remains sufficiently generic to support the coordination of a broader range of distributed business processes that concern information processing and transfer.

Finally, it is our strong belief that the simplicity of the presented solution is key factor for its effectiveness. In fact, not only did we manage to have a most simple modeling language and modeling interface, but also a widely used communication infrastructure and an open messaging protocol. Overall, we hope to address the wide community of e-mail users by providing them with the means for structuring some of their most typical business processes.

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

Ferreira, D., Rei, J., Mendonça, J. M., Ferreira, J. J. Pinto 1999, *Building a Workflow Enactment Service for Telework Co-ordination*, Proceedings of the First International Conference on Enterprise Information Systems, Setúbal, Portugal

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