Building Workflow Applications on Top of WooRKS*

Gang Lu and Martin Ader

Bull S.A.
7, rue Ampère, 91343 Massy Cedex, France
M.Ader@frmy.bull.fr

Abstract: On top of an object-oriented database management system, we have developed WooRKS, a workflow system used to synchronize a group of users working together based on the circulation of documents. Thanks to the object-oriented development methodology and the generic reusable object class library of WooRKS, we can quickly build a concrete workflow application for a specific customer. In this paper, we will describe how we can obtain this high productivity through a concrete application. This workflow application is used now in Bull's Imaging and Office Solution department. As such, we will present also the initial reactions of WooRKS users.

Key Words: CSCW, workflow, object-oriented application, object-oriented database, object reusability, user interface, object-oriented development methodology and office automation

1 Introduction

A lot of routine office task can be described as structured recurring tasks (called **procedures**) whose basic work items (called **activities**) must be performed by various persons and computer systems (called **actors**) in a certain order (sequential or parallel). Inside a procedure, the coordination between the actors in different places (e.g., the synchronization of activities in a procedure) is characterized by the circulation of a folder, forms or papers. The examples of such routine office tasks include dealing with a customer order requirement in a sales department and preparing a business trip in a big company. The examples of the activities inside the above order processing procedure include "order entry", "inventory check", "shipping", "eval order", etc. A **workflow system** is used to assist people in defining, executing, coordinating and monitoring such routine office procedures based on a shared environment. Unlike other CSCW systems[Grudin 91], such as electronic conferencing [Applegate 86] and real time shared editors [Ellis 90], a workflow system, in general, interacts asynchronously with its actors (e.g., end users) working in different places [Johansen 91], and a workflow procedure can spread over several weeks.

In this paper, we will describe the designs and the first experience of WooRKS, an object-oriented workflow system which is developed as a demonstration of a 4 year

* The research was founded by the Commission of the European Communities through ITHACA, Esprit project No. 2705.

Esprit project ITHACA (Integrated Toolkits for Highly Advanced Computer Applications [Proefrock 89]). The objectives of ITHACA are to build an object-oriented database management system [Elsholtz 90], to develop the tools ([Bellinzona 91], [De Mey 91] and [Vassiliou 90]) to support a complete object-oriented methodology [De Antonellis 91], and to validate the database, the tools and the methodology by applications like office automation, financial management and chemical process control.

The goal of the ITHACA methodology is to reduce the long-term costs of application development and maintenance for standard applications in selected application domains [De Antonellis 91]. The key assumption here is that one must be able to adequately characterize the selected application domains so that individual applications can be constructed largely from a reusable object class library. On the one hand, we have to build different workflow applications for different customers because a workflow procedure in one company is rarely the same as that in another company. On the other hand, different workflow applications for different customers do have many common features. As such, workflow applications meet our assumption to reach a good reusability.

In the ITHACA development life cycle, as shown in Figure 1, we clearly separate two roles: application engineers and application developers. The application engineers build and maintain a generic reusable object class library; based on this object class library, the application developers build and maintain a concrete application application package according to the requirements of a customer.

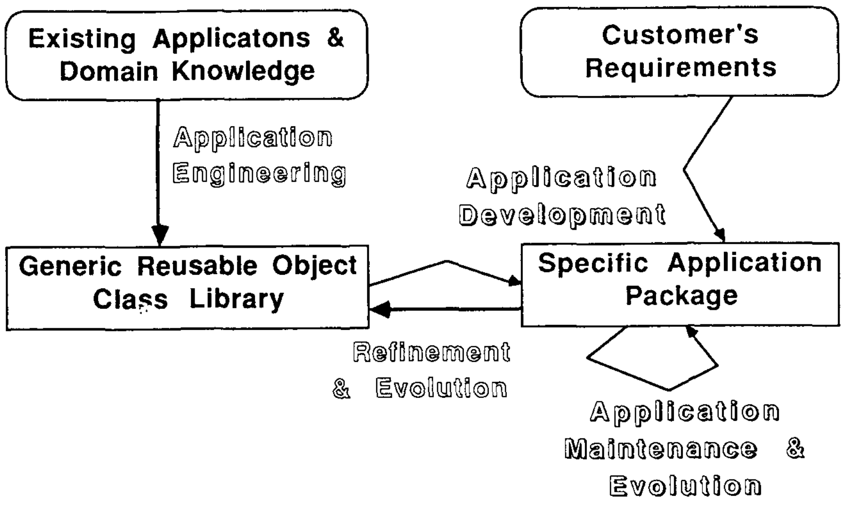

Existing Applicatons & Domain Knowledge
Customer's Requirements
Application Engineering
Application Development
Generic Reusable Object Class Library
Specific Application Package
Refinement & Evolution
Application Maintenance & Evolution

Figure 1. ITHACA development life cycle

WooRKS is implemented on top of an object-oriented database management system (e.g., NooDLE [Elsholtz 90]). WooRKS now consists of nearly four hundred reusable object classes belonging to five modules. The five modules are: **Organization** (e.g., line unit, manager and project), **Information** (e.g., folder, document and letter),

Time (e.g., time point, event and calendar), **Operator** (e.g., mail, print and revise), and **Coordination** (e.g., procedure and workflow basket). The Organization module describes who will work in workflow procedures and which roles they will play. The Information module describes the semantic attributes and contents of the information circulated inside workflow procedures. The Time module provides the basic to define timing constraints for procedures and activities. The Operator module describes the atomic actions of each actor, while the Coordination module describes how a group of actors work together to carry out a procedure. If we compare building a procedure in the Coordination module to writing a sentence, the Organization module, the Information module, the Time module and the Operator module provide respectively four lexical modules: who will work, what will be dealt with, when the work will be done and how the work will be done.

WooRKS is a generic workflow system, instead of one workflow application developed for a specific customer. WooRKS defines an architecture (i.e., client-server communication, sharing information based on an object-oriented database management system, and a method for structuring object classes), imposes a development methodology, and provides a reusable object class library. Based on this generic workflow system, we can develop a workflow application for a specific customer in one month. The short building time of a workflow application is one of the most important features compared with the workflow systems based on a relational database management system, such as FlowPATH of Bull, ProcessIT of NCR and Workflo of Filenet.

In the rest of this paper, we will describe how we can obtain this high productivity in our first pilot application. In Section 2, we will present some guidelines to choose our first pilot workflow application. In Section 3, we will present how we practise the development methodology of WooRKS through the application. In Section 4, we will give an evaluation of the application.

2 Guidelines to Choose the First Pilot Workflow Application

In Chinese, we say that a good beginning is half way to success. As such, we need to carefully choose our first pilot application of WooRKS. Some of our guidelines are as follows:

- Overhead-Benefit Relation

The question of who is paying for the overhead of a CSCW system and who is going to receive the benefits is crucial for its success [Grudin 89]. We should ensure that the persons who benefits from WooRKS can persuade the persons who pay the overhead to use the system.

- Minimum Critical Mass

A CSCW system serves a group of persons. Every person working on the same procedure should use WooRKS in order to maximize the benefit of WooRKS and to minimize the overhead to exchange information between different persons [Francik 91]. This requires that our pilot application should support enough users to cover a critical mass [Markus 87]. In order to simplify our work, we should choose an application having a small critical mass.

- Integration with Existing System
Groupware is based on the computerization of individual's work. This requires the compatibility and interoperability between WooRKS and existing tools. In our pilot application, we have tried to minimize the work to integrate WooRKS with other existing tools without dissatisfying the users. The integration is not limited to technical issues. Other points to consider include existing company procedures and organizational constraints.

- Typical Workflow Problem
We need to design an attractive demonstration to get end users interested in WooRKS. Attractive demonstrations, however, may mislead users as described by [Francik 91] about Wang's Freestyle system. The customers of Freestyle were strongly drawn to the power and ease of use of annotation tools per se, particularly the synchronized playback of their handwriting and voice. As a result, they initially paid less attention to group communication: that is, how the annotated documents would enter and leave their PCs. Therefore, our pilot application should focus on the true workflow problem, i.e., the asynchronous coordination between a group of users in different places.

- Minimum Testing Cost
We could spend at most two man-months to develop the first pilot application and to train the users to use it.

3 Development Methodology

The main steps to build a concrete workflow application using WooRKS include identification of the problem, identification of the objects in the different modules of WooRKS, prototype of the user interface, development of new object classes, and modifications according to the end-users' comments. In this section, we will briefly describe the first three steps.

3.1. Identification of the Problem

Keeping the above guidelines in mind, we start to look for our pilot application. Our secretary is heavily overloaded. As such, we think that she will be interested in WooRKS. She described several routine tasks, and we choose the leave management problem. The way to manage leave in our department of Bull is as follows:

- The employee fills in a specific form and passes it to his* manager;
- The manager approves the request and passes the form to his secretary;
- The secretary checks the leave balance of the employee; and
- If the employee has enough leave remaining, the secretary modifies the leave records; Otherwise, the secretary will notify the employee to modify his request.

At the end of every week, the secretary sends a summary to the payment department.

* Through this paper, the pronoum "he" is used in the neuter sense.

3.2. Identification of the Objects in the Different Modules

For the leave management problem, we identify four kinds of persons: the employee, the secretary, the manager and the persons in the payment department. The manager only signs the request, and the communication with the persons in the payment department is only through the secretary on papers. As such, we decide that WooRKS supports only the employee and the secretary at the first stage.

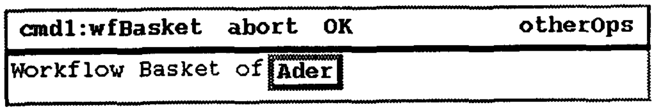
After browsing the reusable object class library, we identify the existing objects which can be reused and the new objects to be created. In the Organization module, we create **bullActor**, a sub-class of actor, where the new attributes include birthday, service duration, leave entitlement, home address and home telephone. In the Information module, **leave request**, a sub-class of information, where the new attributes include leave applicant (bullActor), leave starting date, leave ending date, leave reason, the number of leave days and leave approval status. In the Operator module, we introduce **weeklyReport**, a sub-class of command, to generate weekly leave reports based on agenda objects in the Time module. In the Coordination module, **leaveRequest** where the employee executes the first activity to fill in a request form; the secretary executes the second activity to check the leave balance; WooRKS executes automatically the third activity to print the leave request form; and when the secretary receives the signed leave request form, he executes the forth activity to archive the leave request and to send the employee a notice if his request is not approved.

3.3. Prototype of the User Interface

WooRKS allows the application developers to rapidly build the end-user's working scenario without creating the object classes in various modules (e.g., Organization, Information and Coordination). As such, we can ask the end users to evaluate a WooRKS application before the application is built.

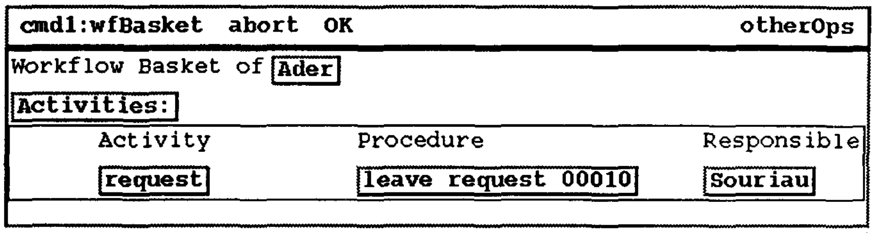
The introduction of workflow will change the way people work. For the employee, the way to require leave using WooRKS will be modified as follows:

- The employee (e.g., Mr. Ader) logs in to WooRKS and sees the top-level menu as shown in Figure 2;
- The employee selects "otherOps", then "leave request" from a pull-down menu. A "leave request" procedure is created and Figure 2 will be modified as Figure 3 to ask the employee to work on the "request" activity of the procedure. The activity is started automatically so that Figure 4 is shown also;
- The employee fills in Figure 4 (e.g., "from", "to" and "Leave reason"). WooRKS has an agenda for each actor (i.e., employee). The agenda of the employee will be modified automatically after the employee confirms his request by selecting "OK" in Figure 4; and
- Figure 3 becomes Figure 2, and the employee can select "OK" to quit WooRKS.



Workflow Basket of Ader

Figure 2. Employee's top-level menu

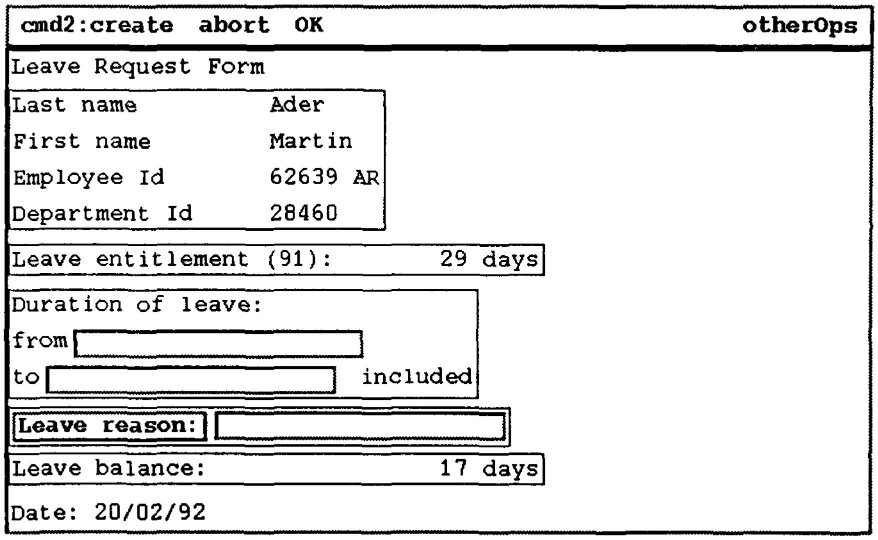


Workflow Basket of Ader

Activities:

Activity	Procedure	Responsible
request	leave request 00010	Souriau

Figure 3. Update of Figure 2



Leave Request Form

Last name	Ader
First name	Martin
Employee Id	62639 AR
Department Id	28460

Leave entitlement (91): 29 days

Duration of leave:

from

to included

Leave reason:

Leave balance: 17 days

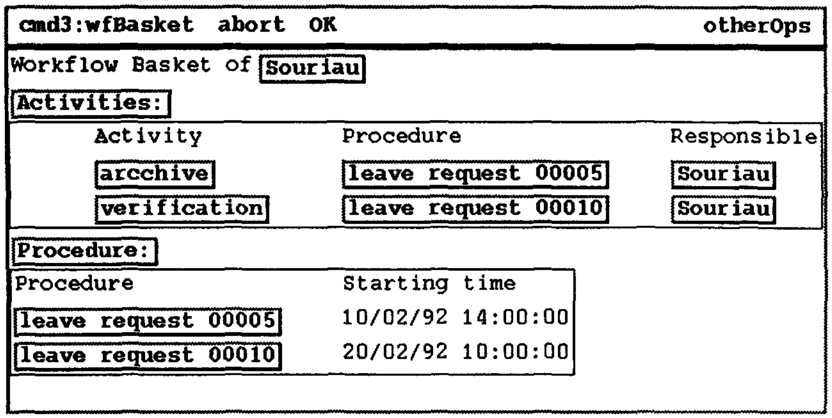
Date: 20/02/92

Figure 4. Leave request form for the "request" activity

For the secretary, the way to deal with a leave request will be as follows:

- The secretary (e.g., Mrs. Souriau) logs in to WooRKS and sees the top-level menu as shown in Figure 5. Figure 5 consists of three parts: an indication line (Workflow Basket of Souriau), a list of activities (e.g., all activities which the secretary is asked to work on) and a list of procedures (e.g., all procedures under the responsible of the secretary);
- The secretary selects "otherOps", then "start activity" from a pull-down menu, and finally selects the activity to be started (e.g., "verification" in Figure 5). Then Figure 6 will be shown;
- The secretary fills in Figure 6 (e.g., "Number of leave days"). After he selects "OK", a complete form will be printed; Then Figure 7 will be shown. The secretary puts the printed form in the employee's mail box; and

- When the secretary receives the form signed by both the employee and his manager, he initiates the "archive" activity in Figure 7. If the leave request is not approved by the manager, a notice will be printed and the employee's agenda will be modified to cancel the leave mark.



Workflow Basket of Souriau

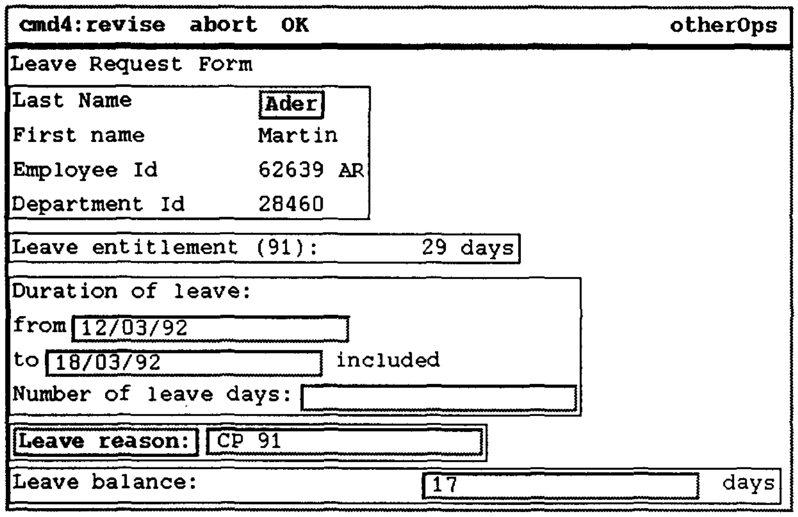
Activities:

Activity	Procedure	Responsible
arcchive	leave request 00005	Souriau
verification	leave request 00010	Souriau

Procedure:

Procedure	Starting time
leave request 00005	10/02/92 14:00:00
leave request 00010	20/02/92 10:00:00

Figure 5. Secretary's top-level menu



Leave Request Form

Last Name	Ader
First name	Martin
Employee Id	62639 AR
Department Id	28460

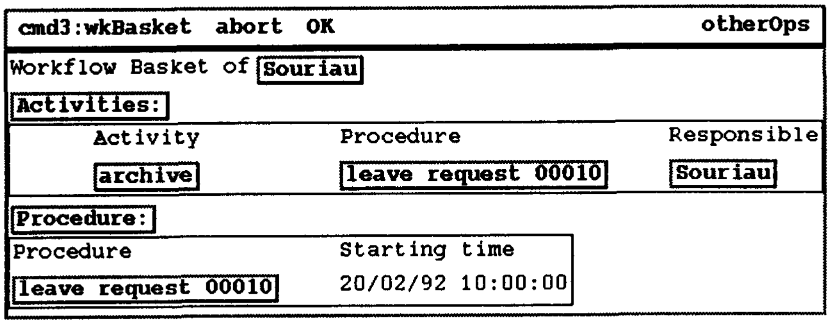
Leave entitlement (91): 29 days

Duration of leave:
from 12/03/92
to 18/03/92 included
Number of leave days:

Leave reason: CP 91

Leave balance: 17 days

Figure 6 Leave request form for the "verification" activity


cmd3:wkBasket abort OK otherOps

Workflow Basket of Souriau

Activities:

Activity	Procedure	Responsible
archive	leave request 00010	Souriau

Procedure:

Procedure	Starting time
leave request 00010	20/02/92 10:00:00

Figure 7. Archive activity

The weekly reports will be printed automatically thanks to the periodic event mechanism of WooRKS.

4. Evaluation

The leave management is a typical workflow problem. The critical mass is small after we limit WooRKS to support only the employees and the secretary. Because we can try our pilot application within our own department, this significantly reduces the testing cost. One of main reasons to choose leave management as our pilot application is that the leave management is not yet computerized. As such, we are quite free to choose systems and information formats. After we resolve the signature control problem as described in the section above, the integration problem is resolved.

4.1. Overhead-Benefit Balance

The employee need to fill in only 3 fields in Figure 4, instead of 16 before WooRKS is used. He need not remember his Bull employee Id and his Bull internal department Id. He need not refer to the calendar to calculate the number of working days during his vacations. He need not worry about mistake of leave balance calculation. The overhead of the employee is that the creation of the leave request form is separated from the signature of the form.

The secretary gets the most benefit from WooRKS. The weekly reports will be generated automatically. The verification of leave request is also simplified because a large part of "leave request form" (Figure 6) is filled in by WooRKS. Because the agenda is modified automatically to take into account the absence of the employees, other persons can retrieve the absence information from WooRKS without interrupting the secretary. The overhead of the secretary is that he has to involve twice for each leave request (i.e., leave balance calculation and archive).

4.2. Security Control

Data security is one of the key issues to decide whether WooRKS can really be used. Each user has to give his password when he logs in to WooRKS. The user can only access the authorized commands according to his roles defined in the Organization module. For instance, only the secretary can access command "revise" to modify the leave entitlement of each employee. The accessible information is context-sensitive. The home address and home telephone number become visible only through command "revise". As such, only the secretary can access this personal information.

The object-oriented database management system guarantees the data recovery from software and hardware errors. When we change the data schema of WooRKS, the ODBMS sometimes cannot automatically transfer the existing object instances from the old data schema to the new data schema. We resolve this schema versioning problem by introducing our own "loader" and "unloader" utilities.

4.3. End-Users' Feedback

We officially introduced WooRKS in our department in February 1992. The users, in general, like WooRKS. WooRKS really simplifies their work. So, they use WooRKS.

People apply for leave only several times per year so that they are always occasional users of WooRKS. Occasional users require the user interface to be simple, flexible and informative. The main criticism about WooRKS comes from the users of Microsoft Windows. The user interface style is not the same as what they use to be.

Using WooRKS, the secretary now spends about 2 minutes to deal with one leave request. This is much shorter than when she deals with leave request without WooRKS. However, the secretary has to frequently wait for WooRKS to deal with his inputs during these two minutes. As such, he requires a better response time of WooRKS.

5 Conclusions

This pilot application proves our idea of generic application framework based on the object-oriented technologies. Less than 10% of the object classes used in this first pilot application were newly developed. Other objects were reused from WooRKS generic object class library. To build this application, we spent:

- a half-day to identify the problem,
- a half-day to identify the reusable object classes and the object classes to be developed,
- four days to prototype user interfaces and to edit the end-user manual,
- ten days to develop new objects and perform integration, and
- two days to make modifications according to user's initial comments.

This well meets our basic objective to **build a concrete workflow application using WooRKS in one month**. This first experience show that the object-oriented technologies increase the productivity of CSCW software development and maintenance. We live in a world changing rapidly. Companies are jointed and reorganized all the times. Their workflow procedures have to follow the changes.

To the best of our knowledge, few papers describe a workflow system built on top of an object-oriented database management system. Our experience shows that the performance and the functionalities of an object-oriented database management system can satisfy the requirements to build a usable workflow product. Our first pilot application of WooRKS is successful. WooRKS simplifies the work of everybody. **It is really used by our department.**

We have ported a part of WooRKS on top of commercial object-oriented database management systems: Versant and Ontos in order to show that WooRKS does not depend on a specific object-oriented database management system. We need to enrich the development and maintenance tools of WooRKS.

6. Acknowledgement

We gratefully acknowledge Mr. Najah Naffah for his support of this research. Frequent discussions with Mr. Clarence Ellis when he worked in our department helped greatly in the design of WooRKS. Mr. Patrick Pons was one of the key members of WooRKS and made many suggestions for the first WooRKS pilot application. Thanks are also due to Mr. Srinivas Raghunandan, Mr. Kabada Srivaths, Mr. Kumar Venkataraman and Mr. Sudarshan Murthy who helped to undertake a large portion of WooRKS implementation.

References

[Ang 91] J.Ang, G.Lu and M.Ader, The Active Office Object Model: its Conceptual Basis and its Implementation, Proc. of the IFIP Conference on the Object Oriented Approach in Information Systems, Quebec City, Canada, Oct. 1991, pp. 419-431.

[Applegate 86] L.Applegate, B.Konsynski and J.Nunamaker, A Group Decision Support System for Idea Generation and Issue Analysis in Organization Planning, Proc. of the First Conference on Computer-Supported Cooperative Work, New York, 1986, pp. 16-34.

[Bellinzona 91] R.Bellinzona and M.Fugini, RECAST Prototype Description, ITHACA.POLIMI.91.E.2.8.#1, Politecnico di Milano, 1991.

[De Antonellis 91] V.De Antonellis and B.Pernici, ITHACA Object-Oriented Methodology Manual: Introduction and Application Developer Manual, ITHACA.POLIMI-UDUNIV.E.8.#1, Politecnico di Milano and Univ. of Udine, 1991.

[De Mey 91] V.De Mey, VISTA Implementation, ITHACA.CUI.90.E4.#1, University of Geneva, 1991.

[Deux 91] O.Deax, The O2 System, Communication of the ACM, vol. 34, no. 12, October 1991, pp. 35-48.

[Elsholtz 90] A.Elsholtz, The NooDLE Database Kernel, Technical Report ITHACA.Nixdorf.90.X.4.#4, Siemens Nixdorf Informationssysteme A.G., Germany, 1990.

[Ellis 90] C.Ellis, S.Gibbs and G.Rein, Design and Use of a Group Editor, Engineering for Human-Computer Interaction, North-Holland, Amsterdam, 1990, pp. 13-25.

[Ellis 91] C.Ellis and S.Gibbs, Groupware Implementation: Issues and Examples, Tutorial Presented at CHI'91, New Orleans, U.S.A., 1991.

[Francik 91] E.Francik, et al, Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems, The Communications of the ACM, vol. 34, no. 12, December 1991, pp. 53-63.

[Gerson 86] E.Gerson and S.Star, Analyzing Due Process in the Workplace, ACM Trans. on Office Information Systems, vol. 4, no. 3, July 1986, pp. 257-270.

[Grudin 89] J.Grudin, why groupware applications fail: problems in design and evaluation, Office: Technology and People, vol. 4, no. 3, 1989, pp. 245-64.

[Grudin 91] J.Grudin, CSCW Introduction, The Communications of the ACM, vol. 34, no. 12, December 1991, pp. 30-34.

[Johansen 91] R.Johansen, Leading Business Teams, Addison-Wesley, 1991.

[Lu 90] G.Lu, A Task-Oriented Architecture of Man-Machine Interaction for Office Automation Systems, Ph.D dissertation, Ecole Nationale Superieure des T\o'e\"l\o'e\"communications, Dec. 1990.

[Markus 87] M.Markus, Toward a "Critical Mass" Theory of interactive Media: Universal Access, Interdependence and diffusion, Commun. Res., vol. 14, no. 5, 1987, pp. 491-511.

[Proefrock 89] A.Proefrock, D.Tsichritzis, G.Muller, and M.Ader, ITHACA: An Integrated Toolkit for Highly Advanced Computer Applications, Object-Oriented Development, Univ. of Geneva, July 1989, pp. 321-344.

[Vassiliou 90] Y.Vassiliou, et al, Technical Description of the SIB, ITHACA.FORTH.90.E2.#2, FORTH, Greece, 1990.