

Universität Stuttgart



Institut für Parallele und Verteilte  
Höchstleistungsrechner (IPVR)

**Prüfer:** Prof. Dr. K. Rothermel  
Prof. Dr. F. Leymann

**Betreuer:** Dipl.-Inf. U. Kubach

**Beginn:** 10.12.1999

**Beendet:** 09.06.2000

**CR-Classification:** H1.2, H3.2, H3.3, H4.1, J1.7

## **Diplomarbeit**

No. 1831

### **Workflow Management in Mobile Computing Environments**

Markus Reichart



<b>1 Introduction</b>	1
<b>2 Workflows and Workflow Management Systems</b>	5
<b>2.1 Terminology</b>	5
2.1.1 Basic terms	5
2.1.2 Process definitions and process instances	7
<b>2.2 Basic workflow concepts</b>	7
2.2.1 Typical workflow scenario	8
2.2.2 Three Dimensions of Workflows	8
2.2.3 Workflows and Transactions	9
<b>2.3 Workflow Management Systems (WFMS)</b>	11
2.3.1 Architecture of WFMSs	11
2.3.2 Workflows and application structures	12
<b>2.4 Example for a workflow execution</b>	14
2.4.1 Execution by the WFMS	14
2.4.2 Issues to consider	15
<b>3 Mobile Computing</b>	16
<b>3.1 Characteristics of mobile computing environments</b>	16
3.1.1 Peculiarities of mobile computing	16
3.1.2 Extended Client-Server model	17
<b>3.2 Mobile computing devices</b>	18
3.2.1 Laptop Computers	19
3.2.2 Personal Digital Assistants (PDA)	19
3.2.3 WAP enabled devices	20
<b>3.3 Infostations: a solution to expand mobile computing</b>	20
3.3.1 The infostation infrastructure	21
3.3.2 Benefits of infostations	21
<b>4 Workflow execution with mobile clients</b>	23
<b>4.1 The modified scenario</b>	23
4.1.1 The client as a workflow server	23
4.1.2 The need for a local storage on mobile clients	23
<b>4.2 Modified system architecture</b>	24
4.2.1 The modified workflow server	24
4.2.2 The modified client	25
<b>4.3 Mobile execution of workflows</b>	26
4.3.1 Mobile devices as stationary clients	26
4.3.2 Preparation of the client for disconnected operation	26
4.3.3 Disconnected operation	27
4.3.4 Reconnection at the end of mobile work	28
<b>5 Activity Streams</b>	29
<b>5.1 Blocks and subprocesses</b>	29
5.1.1 Blocks	29
5.1.2 Subprocesses	29

<b>5.2 Streams</b>	30
5.2.1 The micro script stream	30
5.2.2 Transaction streams	30
5.2.3 Workpackage Streams	31
<b>5.3 Determination of activity streams for mobile processing</b>	31
5.3.1 Types of nodes and their use by WFMSs	31
5.3.2 Rules for determining activity streams	33
5.3.3 Example for workpackage determination	35
<b>6 Data Hoarding</b>	36
<b>6.1 Data needed for mobile execution</b>	36
6.1.1 Data needed to enable workflow management	36
6.1.2 Data needed to execute steps	37
6.1.3 Data needed by the hoarding system	37
<b>6.2 Information sources</b>	38
6.2.1 Process definitions	38
6.2.2 External sources	38
6.2.3 Monitoring component	41
<b>6.3 The hoarding mechanism</b>	42
6.3.1 Gathering the information	42
6.3.2 Data retrieval	44
6.3.3 Transfer of data items	44
<b>7 Realization of the modified system</b>	46
<b>7.1 The workload determination component</b>	46
7.1.1 Interactions with the WFMS	46
7.1.2 Elements of the workload determination component	47
<b>7.2 The hoarding system</b>	48
7.2.1 The gathering component	48
7.2.2 The monitoring component	50
7.2.3 The transfer component	51
7.2.4 The controlling component	52
<b>8 Evaluation of the modified system</b>	53
<b>8.1 The simulation environment</b>	53
8.1.1 Simulation Computer	53
8.1.2 Test of the determination component	53
8.1.3 Test of the hoarding system	54
8.1.4 Realization of the model	55
8.1.5 Expectations on the system's behavior	56
<b>8.2 Results of the Simulation</b>	57
8.2.1 Simulation Scenarios	57
8.2.2 Result Graphs and Interpretation	58
8.2.3 Résumé	59
<b>9 Summary</b>	60

**10 Bibliography** ..... 63

# 1 Introduction

The increasing need for enterprises to react to fast changing market situations demands equally fast changes in the information technology infrastructure for business processes in an enterprise. The use of Workflow Management Systems (WFMS) offers the required flexibility for these changes, while still maintaining the control over the different steps of work of one business process. A WFMS allows the modeling, the coordination of the execution and the monitoring of business processes. Therefore, it allows the detection of required changes, the adaptation of the business process to new needs and the controlled execution of the reengineered business process using one tool. This significantly reduces costs in terms of time and resources during the life-cycle of the changing business processes.

WFMSs increase the efficiency of the business processes of an enterprise. Business processes can be performed faster by streamlining single steps of work and by distributing the amount of work for parallel processing to achieve a better balanced workload.

Not only the enterprises benefit from a WFMS, but the users as well. During execution of a business process they are relieved from coordinating tasks. They do not have to think about whom to send the just created document for approval or what application to start for a specific step of work. Instead the user can concentrate on the work to be done, while the WFMS does the necessary coordination.

Using a workflow system, all participants of a business process benefit from cost reduction, improved accuracy, standardization of data and faster processing. Therefore, workflow systems play a major role in the way business processes are reengineered and executed today.

Another development has changed the way business processes are supported by information technology. The availability and continuously increasing possibilities of mobile computing devices offer an broader range of how work can be done using computers. In the year 1994 in [ImielinskiBadrinath94] tens of millions of users carrying a portable computer were predicted for the near future. 1999 a report from Strategy Analytics [Strategy99] foresaw about 230 million mobile computing devices in 2004 in Western Europe and the United States. The numbers show, that more and more work is done using mobile computing devices such as laptops or hand held devices as personal digital assistants (PDAs).

In many cases these mobile devices offer a more flexible and more efficient work than their stationary counterparts. Changing work environments and the pressure for improved productivity require that users become more independent from their office. They can use computing devices wherever and whenever they want. User do not have to wait until they are back in their office to feed back data, resulting from activities away from office. They can work while traveling and save time for other tasks back in the office.

According to Gartner Group the opportunity for wireless data communication in the United States is huge, with 25.3 million of the 112.1 million workforce

having a mobile job requirement [Gartner97]. Mobile computing therefore has a major and still increasing impact on doing work.

This master thesis looks into the question, how to bring mobile computing and workflow management together. As stated in [Exotica95], there is a tremendous potential in combining these two technologies. The execution of business processes is not only still controlled by a WFMS with the benefits presented above, but obtains a great deal of flexibility by supporting mobile clients.

However this gain in flexibility is bought with a price. Information systems with mobile participants have to deal with the peculiarities of these devices. Varying computing power, limited storage capacity, limitations of in- and output devices and working in disconnected mode are the main topics here. Not all work, possible on stationary computing devices, can be done on mobile ones. Different mobile devices have different capabilities, which must be taken care of when the information system deals with its participants. The disconnected mode mobile devices work in most of the time requires special treatment by the information system and the user.

In this thesis the changes to WFMSs are discussed, which are necessary to enable the integration of mobile clients. Contrary to stationary clients, mobile clients only connect to the workflow system from time to time. During the periods of time, in which the client is disconnected from the central system, it must be able to work independently, using its local resources.

The main focus of this thesis therefore is to find mechanisms which allow the WFMS to automatically provide the user the data needed to perform her tasks in disconnected mode. The challenge for these mechanisms is, that nearly all data must be provided to the user prior to disconnection from the WFMS. Further it is not possible to simply copy all data to the client device, because of limited storage capacity. The mechanisms must therefore select which parts of the data to transfer to the client devices.

The data which has to be provided by the WFMS can be divided into two categories.

The first category contains the necessary data for the coordinating tasks of the WFMS. The mechanism used to determine that data, must decide which tasks, i.e. steps of work are suitable for the user and for mobile execution. It is possible that more than one step qualifies for mobile execution by a specific user. If these steps depend on each other, the collection of steps is called an *activity stream* for mobile execution. The rules for determining valid activity streams are described in this thesis.

The second category of data is the data needed for the successful execution of the tasks scheduled for mobile execution. This includes both the input parameters of the applications and other data required during the work on the tasks. These data could be the status of a client's order, the limit for credits granted by a bank, delivery schedules or other information normally stored in a central database system.

If the applications necessary for the execution of the workflow are not installed on the mobile client, the system is even forced to provide the implementations

of the applications themselves. The data in this category is called *operational data* in this thesis. To automatically provide this kind of data, the WFMS must gather as much meta data about the operational data as possible. Different sources of information are described. Another approach described in this thesis is the monitoring of the users during the execution of activities. Analysis of the users' accesses to data items order the data items according to their importance for the activities of the determined activity stream. The process of gathering relevant data is called *data hoarding*. A prototype implementation of a hoarding mechanism for a workflow system is presented in the thesis as is a simulation model used to show the feasibility of the presented approach.

Aside from the mechanisms, used to provide the mobile client the data for the work disconnected from the workflow system, additional precautions must be taken to enable the execution of workflows on mobile clients. During disconnection a part of the client must play the role of a server and at the end of mobile work special measures must be used to check the returning data items to avoid inconsistencies with the data stored on the central system.

These topics are briefly discussed, but not covered in the same detail as the procedures during the preparation of a mobile device.

After the introduction, the thesis is structured as follows:

Chapter 2 covers workflows and workflow management systems. Following the definition of the used terminology, basic concepts of workflows are presented to introduce the reader to the environment of the thesis. The next topic of the second chapter are workflow management systems. Architectural aspects are described with regard to the tasks of such a system. A description of the structure of applications suitable for use with a workflow management system follows. And an example of how a business process is executed by a workflow management system end this chapter.

Chapter 3 contains an overview about important aspects of mobile computing. Especially the peculiarities of mobile computing, such as disconnection from other computing resources, are described. The implications of mobile computing on data management are briefly mentioned. Different mobile computing devices are presented before the view on a possibility for enhanced mobile communication, the infostation concept, finishes the chapter.

Chapter 4 contains the description of the changes to the system architecture, which are necessary if mobile clients are used in a workflow environment. Using a small example, the way, in which workflows are executed with mobile clients, is described.

Chapter 5 contains a short introduction into the terminology of graphical representations of business processes in workflow systems. Using these terms, the rules for determining the possible steps of work for a mobile user are explained. The resulting decision procedure of the workflow system is presented in detail.

Chapter 6's main topic is the 'Data Hoarding System'. The different sources of information, on which the system bases the decision to hoard a data item, are introduced. The chapter describes which sources provide which information for the hoarding system and how that information can be obtained. Finally, the hoarding mechanisms themselves are described.

Chapter 7 presents a prototype implementation of the system using IBM's MQSeries Workflow as the underlying workflow system. The reasons for the implementation decisions are explained and the most important aspects of the JAVA-based implementation are emphasized.

Chapter 8 explains the model used to simulate the behavior of mobile clients. The reader is introduced to the simulation environment and the results of different simulation runs are explained.

A summary proposes further work on the subject and concludes the master thesis.

## 2 Workflows and Workflow Management Systems

This chapter discusses the basic terms of workflow technology to provide a basic understanding of the different aspects of workflows. The terminology used is based on the definitions of the Workflow Management Coalition (WfMC). Before continuing with the architecture and the tasks of Workflow Management Systems (WFMS) a few basic concepts of workflow technology must be explained. The chapter contains a discussion, why so called workflow-based applications provide the features necessary to fulfill the requirements of today's businesses. Finally a small example shows how a workflow system executes a business process.

### 2.1 Terminology

In this section the terms concerning workflows and workflow management are introduced according to the standard issued by the WfMC in [WfMC99].

#### 2.1.1 Basic terms

**Business Process:** A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.

**Process Definition:** The representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data etc.

**Workflow:** The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

**Workflow Management System:** A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines. Workflow Management is software, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.

- Instance:** The representation of a single enactment of a process, or activity within a process, including its associated data. Each instance represents a separate thread of execution of the process or activity, which may be controlled independently and will have its own internal state and externally visible identity, which may be used as a handle, for example, to record or retrieve audit data relating to the individual enactment.
- Activity:** A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources to support process execution; where human resource is required an activity is allocated to a workflow participant.
- Workflow participant:** A resource which performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the worklist.
- Worklist:** A list of work items associated with a given workflow participant or group of workflow participants.
- Work item:** The representation of the work to be processed in the context of an activity within a process instance.

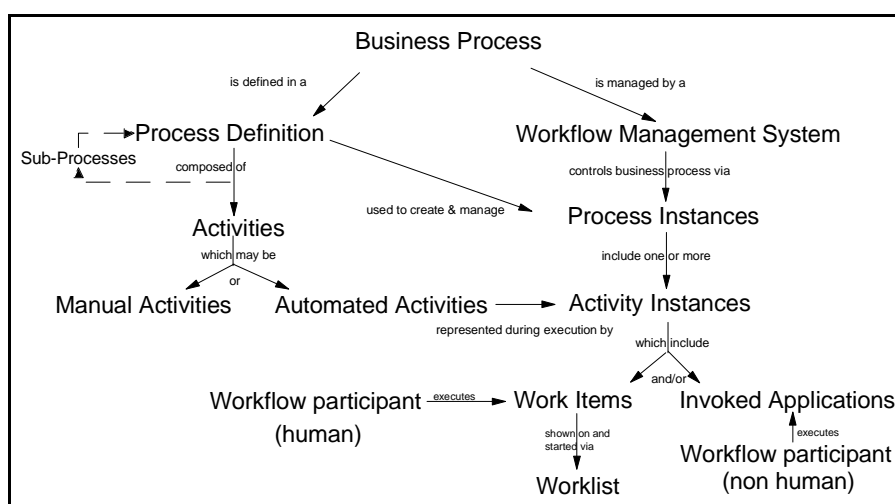


Figure 2.1: Standardized workflow terms

Figure 2.1 shows the relationship between the different terms. The modeling of a business process results in a process definition, which is managed by a WFMS.

The process definition contains the description of the different steps building the whole process and the relations between the steps. Those steps can be tasks, which need human intervention, so-called manual activities, or they can be tasks which can be done automatically, so-called automated activities. The management of the business process by the WFMS is carried out through process instances. Each time a user starts a business process the WFMS uses the process definition of the business process to instantiate a specific process instance. In each instance the activities of the business process are represented by work items in case of manual activities and invoked applications in case of automated activities. The manual activities are presented to the users via their worklists.

The following describes the relation between process definitions and process instances in more detail.

### 2.1.2 Process definitions and process instances

To be able to perform a business process with support from computing devices, the process must be described in such a way that a computer can process it. This form is called process definition. Such a definition is the result of the modeling work of the business process. It consists of the activities identified during that work and connections between those activities. These represent dependencies between the activities. For every activity additional information, needed for execution, is also recorded. This information includes the necessary resources for the activity, the responsible user and the applications to be used. Additionally several conditions have to be stated. This includes pre and post conditions which determine the correct starting and ending of an activity or conditions associated with the connections, which provide a means to dynamically control the order of activities.

The process definition is a model of the real world's business process and is used by the WFMS as a template for the execution of that process. When the WFMS is going to start a workflow, the template of the corresponding business process is used to create a new instance of the workflow. Such an instance represents one execution of the business process. That is why each instance uses its own data, describing the context of the execution. Thus the instantiation of process definitions does allow the system to execute multiple workflows of the same business process at the same time. The different instances have only one thing in common, they originate from the same process definition. During their lifetime, i.e. the running time of the business process, they can individually act differently. Each instance acts according to the concrete execution of the corresponding business process.

## 2.2 Basic workflow concepts

In this section the most important concepts of workflows are presented. The typical scenario of workflows is presented, followed by an introduction of what is defined in a workflow and of the relations between workflows and transactions.

## 2.2.1 Typical workflow scenario

As mentioned in the previous section workflows are used to automate business processes. Those processes normally contain multiple steps of work. In order to work efficiently, the work is shared or distributed between multiple workers. They can belong to the same department using a common distributed system or they participate in other systems of the same company or even of a different company. The WFMS takes care of the distribution of single steps of work to the workers. It assigns the activities to the workers or it might even spawn subprocesses to be executed by other workflow systems.

During work the participants are supported by applications running on different kinds of computing devices. These are connected via the workflow management system. It takes care of the necessary communication in heterogeneous environments and invokes the applications as defined in the workflow. From the users' point of view, the execution of a workflow consists only of workitems appearing on their worklists, the execution of applications associated with the workitems and their disappearance after the work is done.

## 2.2.2 Three Dimensions of Workflows

The short description of a typical workflow scenario shows, that business

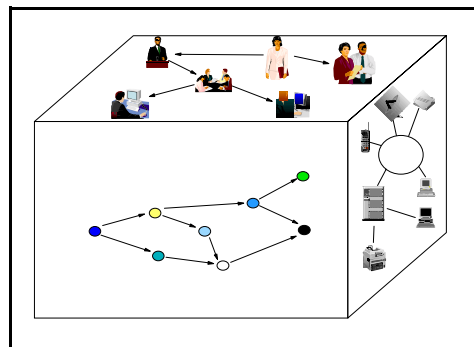


Figure 2.2: Three dimensions of workflows

processes, and therefore workflows, have three independent dimensions (Figure 2.2) [LeymannRoller99].

The first dimension describes the process logic, i.e. *what* activities are to be executed and in *what* sequence. Figure 2.3 shows a graphical representation of the *what-dimension*. The different circles stand for the different activities, which can be carried out by invoking programs or starting another workflow process. The arrows define the flow of control between the activities. By defining parallel flows, the time required to perform a business process can be reduced. While sequential flows

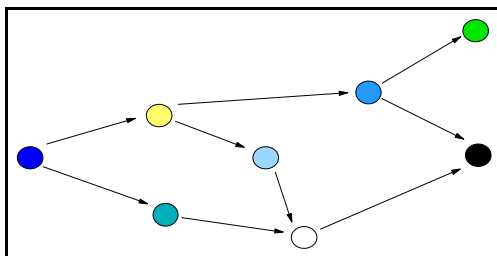


Figure 2.3: what-dimension of workflows

indicate a dependency between the activities, which enforces a special execution sequence.

The second dimension is the organization dimension, called the *who-dimension*. It describes the organizational structure of the company(ies) executing the workflow in terms of departments, roles and people. This information is necessary for the workflow system to be able to determine, *who* should perform a specific task. That does not mean that only one person can be selected for each activity. Instead this dimension allows the system to find all people qualified for the task at hand. The *who-dimension* is queried for qualifying workers in the so-called *staff query*. After the *staff query*, the system can assign the work of the activity.

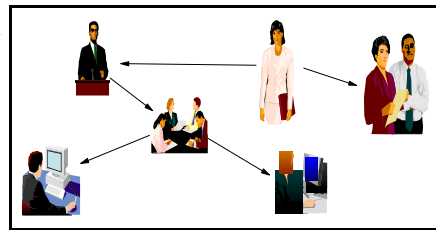


Figure 2.4: who-dimension

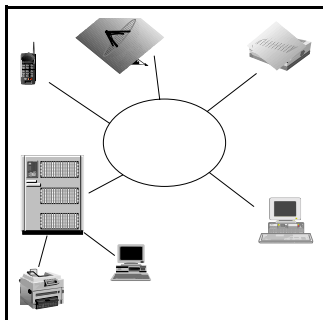


Figure 2.5: which-dimension

The third dimension provides information of the company's IT resources. Through this information the system can assign to the activities, *which* resources are to be used. This includes information about the programs used to perform the activities, their location in the company's computing environment, the mechanisms to invoke them. Additional IT-resources which are needed during execution are also described in the *which-dimension*.

## 2.2.3 Workflows and Transactions

Business processes are often composed of units of work, which have to be executed as one. If one activity in such a unit fails the whole unit cannot be successful. This behavior is called all-or-nothing semantics. Units of work, which require all-or-nothing semantics, can be implemented as transactions. A transaction systems guarantees, that either the whole unit is executed successfully or no effects of the work of the parts become visible. In a distributed environment all participants of a transaction can do their work in parallel, but at the end of the work they must wait for the decision whether the transaction was successful or not. One possibility to safely find and communicate that decision is the two-phase commit protocol, described for example in [GrayReuter93].

Two different problems arise, when transactional behavior is desired in workflows. The first concerns the reusability of implementations of steps of work. The second is the long duration of the execution of typical business processes. Both problems are solved by using so-called *spheres* to group activities to transactional units.

### 2.2.3.1 Atomic Spheres

A small example explains the problem of reusability and transactional behavior. The process of a funds transfer consists primarily of two steps of work. The money must be withdrawn from one account and be deposited on another. It is clear that one activity should only be done, if the other is done too. But the two steps themselves are useful processes for a bank. Thus it would be convenient,

if they could be used independently.

To solve this problem, the activities must be implemented without mechanisms, which assume a transactional behavior while dealing with other participants. The composing of different activities to a transactional unit of work must then be done from the 'outside'. The environment in which the implementations are run, for example a WFMS, must establish transaction boundaries around the implementations. So-called *atomic spheres* [Leymann95] are such sets of activities within transactional boundaries.

If one activity assigned to an atomic sphere fails, the whole sphere fails and the navigation of the workflow restarts the atomic sphere at the entry activities. To avoid infinite looping, a maximum number for retries must be defined. If this number is reached, the atomic sphere is finally declared failed. After a atomic sphere finally fails, different actions can be taken. A responsible user might be notified or the steps can be treated as any failed single step and the WFMS will continue its navigation through the workflow according to the process definition.

### 2.2.3.2 Compensation Spheres

In many practical situations it is not sufficient, to group activities together to transactional units like atomic spheres.

Activities which are themselves transactional must often be grouped together with activities without transactional behavior, still building a unit of work with all-or-nothing semantics. In case of failure those non-transactional steps must be *compensated*. The same applies to steps of an atomic sphere which apparently provided correct results at completion time, but which are recognized as wrong after the atomic sphere is completed. The most important reason why atomic spheres are not always practical, is however, that steps of work in a business process can last very long.

Long lasting activities participating in atomic spheres have to hold locks on resources needed during execution until the outcome of the atomic sphere is decided. That is necessary to guarantee, that no effects of an activity become visible before the sphere is processed successful. However, holding locks for a long time reduces throughput. If throughput is an important issue of the business process, the activities of an transactional unit of work should be able to release their locks after they finished work.

Releasing locks early leads to possible inconsistencies in case of failures of the transactional unit of work. Other activities might have used data written by an activity of an transactional unit. If the transactional unit fails, after the activity wrote its data, the other activities read data, which is not valid anymore. Thus, in case of failure the activities, which release their locks early, must be treated in a different way than just rolling back the transaction. Like non-transactional activities, those activities must be compensated. Compensation is the semantical taking back of the effects of activities already finished. Compensation is done by a set of activities doing backward recovery. These activities 'repair' the error(s) of the transactional unit of work, as soon as they are detected. After the system is brought into a consistent state again, the navigation continues from the point where the compensation started.

Units of work with transactional behavior, which might need compensation, are

called *compensation spheres*. A compensation activity is assigned to each activity in a compensation sphere. The sphere itself can have activities assigned, to compensate the whole unit of work without having to compensate every activity in the sphere.

## 2.3 Workflow Management Systems (WFMS)

Workflow management provides a set of functions that allows customers to define, validate, execute, manage, and reassess their business processes in an automated, yet dynamic way [WhiteFisher94].

Workflow management systems can be used in a great variety of environments, from software development via automated creation of technical documents to the control of production and administration tasks. The system architecture that makes this variety possible is described in the following sections.

### 2.3.1 Architecture of WFMSs

The architecture of most WFMSs is a three tier-architecture, shown in figure 2.6

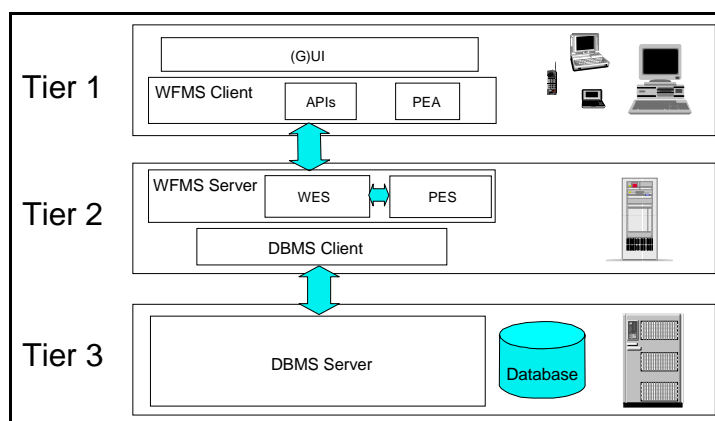


Figure 2.6: Three tier-architecture of WFMS

[MQseries99]. Tier 1 covers the application(s) the end-user runs on her computing device. Typically a graphical user interface is connected to the WFMS client, which consists of two main components: the APIs provided by the WFMS to access the WFMS server components on tier 2 and the program executor component or program execution agent (PEA) used to start the applications associated with the activities assigned to the user.

The main component of tier 2 is the workflow management server component, which itself consists of multiple components. The most important components of the WFMS server are the workflow execution server (WES) and the program execution server (PES). The first one is responsible for navigation through the workflow definition. This process includes querying the organizational information via so-called staff queries to determine the workers for the next activities, dispatching the appropriate workitems for the activities to the users and finding the next activities after the work on an activity ended successfully. The second, the program execution server, controls the execution of those steps in the workflow, that do not need any human interaction. The workflow

execution server sends execution requests to the program execution server, if it encounters an automated activity while navigating through the workflow. Then the PES determines the correct invocation mechanism for the program corresponding to the activity, invokes the program with necessary input data and returns the result of the execution to the WES.

Tier two also contains a client to the database system associated to the WFMS. The server component of a database management system (DBMS) builds the third tier. Relevant data like process definitions and organizational data is stored in that database.

### 2.3.2 Workflows and application structures

The support of a business process by using computing devices can be done in multiple ways. The simplest way would be, to only support the single steps of work with computers. The order of the steps would not be controlled by the computer. But this approach has some major flaws and is therefore not practicable:

Such loosely coupled collections of applications can not guarantee, that the steps of work of the business process are done properly; above all they cannot guarantee that the correct order of steps is maintained. The passing of data from one activity, i.e. from one program to another without centralized organization would require an extra amount of programming, which is not justifiable. This is, because converters, which change output to input data formats, would be necessary for all combinations of applications following each other.

The supporting of single steps misses the main objective, that companies try to achieve by using workflow systems. Secure and efficient work is not guaranteed and the amount of work needed to integrate new applications, makes simple and fast changes impossible.

The loose coupling of applications also misses the objective, because dependencies between the programs exist and have to be taken into account. A possibility to do that, would be the integration of the definition of the business processes into the program logic, of the implementations of the steps of work. The disadvantage of this approach is obvious: total loss of flexibility to respond to changes in the business process or in the environment.

Each change in the business process would mean a corresponding change in the applications. These application changes often are time-consuming, expensive and error-prone. The same applies for the integration of new system resources.

The use of a WFMS to define and execute business processes leads to a two-part structure by separating the applications from the control flow; however, both parts are still under the control of the WFMS. The separation enables independence of the applications for single steps from the whole business process. If an activity changes or must be supported by a different program, the new or modified program is installed and the definition of the business process in the workflow system is updated to use the new program. Because the business process is composed of a set of applications and a definition of the execution order of these programs, changes in the business

process require no changes in the applications. Instead the set of applications and their execution order are changed to mirror the new process. The definition of a process contains among other data the following information:

- the control flow between single steps of work, i.e. different applications
- the data flow during the runtime of the business process
- the applications and manual activities, which have to be invoked
- the persons, who should do the steps of work
- the resources required by the activities, for example which computer, printer or machine to be used.

The separation in two parts, applications and control flow, under control of the WFMS offers needed flexibility without losing the possibility to enforce the defined execution order of the business process. During the definition phase of the business process the user defines the way the work should be done, i.e. the order of the steps and the flow of data between the applications. The WFMS controls the execution order and the flow of data between applications when it processes a workflow.

At execution time the WFMS creates an instance of the definition of the workflow, which is processed by the workflow manager by dispatching the work to the workers, starting programs, recording the execution history and taking recovery measures in case of errors.

Applications, following that two-part structure, are called workflow-based applications [LeymannRoller97]. They are typically the result of a business reengineering process, i.e. the analysis and restructuring of the business and work processes to increase efficiency [HammerChampy93]. At the end of such a restructuring process a new business process is established, which no longer contains any unnecessary tasks. The remaining tasks are engaged with the highest degree of parallelism possible. Multiple persons can work on these tasks and do the business process in an effective way.

## 2.4 Example for a workflow execution

This section describes how activities of a business process are executed using a WFMS. Figure 2.7 shows the activities of a business process describing the course of an order of items:

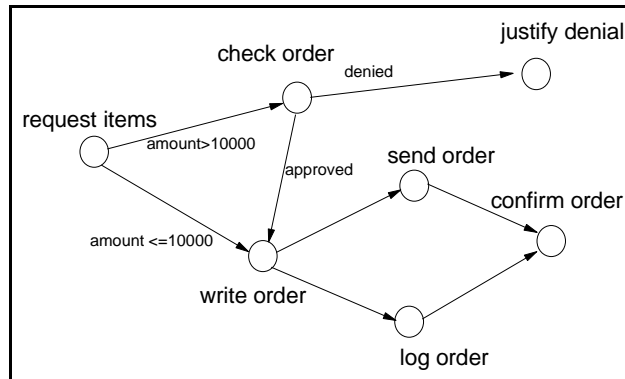


Figure 2.7: Example of a workflow

The process is started, when a user requests some items. She fills out a request form, listing the wanted items. If the items are worth more than 10000, the order must be checked by a manager of the requester. Otherwise the request is always granted. If the request is approved or worth less than 10000, the business process is continued with the writing of the order. The order is sent to the supplier and logged in the companies database. If the order is successfully sent and logged, the request is confirmed. If the manager denied the request, the denial must be justified in writing.

### 2.4.1 Execution by the WFMS

This business process is executed by a workflow management system in the following way:

After the user connected to the workflow system, the activities of running processes she is authorized are presented on her worklist on the client computing device. She also has the right to start new process instances. In order to process her request for some items, she starts a new instance of the 'request items' business process. The workflow system searches for the appropriate process definition and assigns the starting activities. In this case the only starting activity is the 'request items' activity, which is sensibly assigned to the user, who started the process. The assigned activity appears on the user's worklist as a new workitem. By selecting the workitem, the user starts the activity. The WFMS starts the associated program; in the example a request form is displayed on the user's computer. After the form is filled out, the program is finished and the workflow management system stores the gathered information as the result of the activity for further use.

When the activity has ended, the workflow system continues the navigation through the process definition. In the example the result of activity 'request items' determines the next activity. If the requested items are worth less than 10000, the path to the 'write order' activity is followed. Otherwise the next step is the 'check order' activity. Because this activity must be executed by a

manager of the requester, the WFMS queries the organizational database for the persons qualifying as 'manager'. The 'check order' activity is then put on the worklists of the user, who were returned by the staff query. When a manager of the requester starts the activity, it is instantly removed from the other managers' worklists. In that way, the WFMS makes sure, that one activity is not worked on twice.

As with the first activity, the result of the 'check order' activity is stored and used to determine the next steps of work. If the request is approved, the workflow continues with the 'write order' activity. The workflow system assigns the activity to users, who can do this step of work, supplies needed information and stores the result, in this case a written order.

The next steps are assigned in parallel. Because 'log order' and 'send order' can be executed independently, the workflow assigns them at the same time. This makes the process as efficient as possible.

The 'confirm order' activity can only be executed, when both its predecessors were executed successfully. This is guaranteed by the workflow system, because it delays the assignment of the 'confirm order' activity until both results of the predecessors are known to the system. After the 'confirm order' activity is executed, the business process ends.

## 2.4.2 Issues to consider

This section briefly repeats important issues of stationary workflow execution. They are worth noticing, because their absence in a mobile environment makes the special treatment of mobile clients necessary.

The small example shows several aspects of stationary workflow execution. Most important is the way, steps of work are assigned to the user. Whenever an activity becomes ready to be worked on, the workflow system assigns it to all appropriate users connected to the system. When the user starts the activity by selecting it from her worklist, the workflow system gathers needed input data and starts the application, which supports the activity. This can be done at execution time, because it is assumed in a stationary environment, that the connections between client, WFMS and data servers are stable.

Also worth noticing is the fact, that results of activities are immediately available for navigation purposes or as input data for other activities. This ensures, that the workflow system can continue its navigation through the business process. Finally, the synchronization of different paths in the workflow at activities with more than one predecessor is guaranteed by the workflow system.

## 3 Mobile Computing

In this chapter the peculiarities of mobile computing are described. The differences between mobile and stationary computing environments are outlined by presenting the characteristics of mobile computing environments and the possibilities of mobile computing devices. At the end of the chapter the information concept is presented, which represents a more flexible approach to mobile communications than a simple wide area network.

### 3.1 Characteristics of mobile computing environments

#### 3.1.1 Peculiarities of mobile computing

Mobile computing can be distinguished best from non-mobile computing by four characteristic constraints [Satyanarayanan96].

- Mobility is hazardous. Portable computing devices are more vulnerable to loss or damage. Thus, security of data is a major concern, when using mobile devices.
- Mobile elements rely on finite sources of energy. Though battery technology improved enormously through the last years, the need to be sensitive to power consumption results in greater amounts of work while constructing mobile information systems.
- Mobile computing devices are resource-poorer compared to stationary devices. Because things like weight, size, power supply and ergonomics must be considered during construction of a mobile device the same amount of money will be worth less in terms of computing power, memory size or disk capacity buying a mobile computer instead of a stationary computer. This gap nearly has been closed in recent years, but still exists. To enhance the problem, different mobile computing devices differ in the availability of computational resources (see section 3.2).
- Mobile connectivity is highly variable in performance and reliability, if it exists at all. The communication has to be wireless to enable mobility and therefore has to deal with problems like differing-bandwidth connections, loss of connections or gaps in the coverage of the wireless network. Additionally some devices could lack the ability to connect to other computing devices via wireless networks.

The problem of possible loss or damage of mobile computing devices, and thus the loss of data cannot be solved completely. But with security measures like encryption, access control or a company policy rating certain data to be too important to be put on mobile devices, the effects of losing the mobile device can be minimized.

The problem of the devices relying on finite sources of energy is not that important, if the mobile devices' energy sources can be recharged while

traveling. Additionally if the choice of units of mobile work is done with the limited energy source in mind, the problem nearly vanishes. Even if the energy source is failing, most mobile devices issue warnings in time, enabling the user to save her work.

Different capabilities of mobile devices must be taken into account, when the tasks of mobile work are chosen. The major disadvantage of mobile clients compared to stationary ones is not that their resources are less powerful, but the fact, that they must rely on them. In a stationary environment the clients have access to all shared resources of the whole system. A mobile device however has only the built-in resources at its disposal.

The most important constraint for this thesis are the problems associated with mobile connections. If mobile connections were as reliable and efficient as stationary connections, the differences between mobile and stationary work would not require any changes in the systems. Mobile clients could be treated just like stationary devices.

But because these constraints exist, a different approach to building of mobile information systems is required. The demands to cope with the constraints are conflicting. On one hand the relative poverty of resources, lower robustness and security issues argue in favor of reliance on stationary servers. On the other hand the problems of wireless connections and sensitivity to power consumption demand a high degree of self-reliance.

A viable approach to mobile computing must strike a balance between these concerns. Because of the changing circumstances of mobile computing this balance must even be a dynamic one, which can adapt to environmental changes.

### 3.1.2 Extended Client-Server model

Traditionally, i.e. in stationary computing environments the client-server model is used to build distributed information systems. A small number of server sites hold the data and provide efficient and secure access to that data to a much larger number of clients. In a stationary environment the roles of client and server are fixed. Applications on the client request services from the server. The server processes the request and sends the results back to the client.

Mobility requires a slight modification of the highly scaleable and secure client-server system. A server might have to run applications, that are typically executed by a client. This is necessary, if the client lacks the needed computing power. For example the client might be able to display data without being able to compute it. Conversely, during disconnection the client might need to emulate server functions to be able to work properly. In this case the requests of applications running on the client are intercepted and processed on the client. Thus, a computer system with mobile clients must provide means, which allow this change of roles of client and server in certain situations. Figure 3.1 shows the differences between stationary client-server environments and distributed

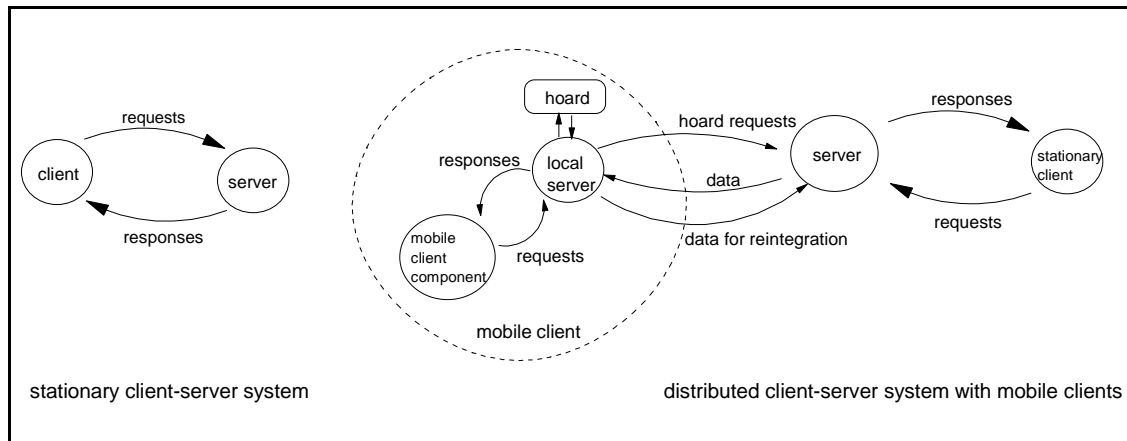


Figure 3.1: Extended client-server system

systems with mobile clients. In stationary client-server systems the client always sends its requests to the server, which processes the requests and sends back response messages. Mobile clients can not access the server at all times, therefore they need a built-in component, which acts as a *local server* during times of disconnection. In order to satisfy the requests the local server must have access to several data items. Because connections to central data repositories might not be available, the local server must access the storage of the mobile client. The local storage space is called *hoard* and is filled before a mobile client disconnects from the server. This creates several potential problems. First data items in the hoard can never be current. That means that a mobile client, which needs current data values, must use a mobile communication mechanism to access the central server or must stop its work until it is reconnected to the central system via a stationary network. The second problem is even more severe. When a data item is transferred to the local storage of a mobile client, it remains available on the stationary server to enable stationary clients to do their work. As long as the items are accessed read-only by all clients, no problems arise. But if one or more clients update data items, two or more different copies of the same data item exist. Therefore special attention is necessary, when mobile clients reconnect to the central system and want to reintegrate data items from their local storage. Different forms of concurrency control are possible [BadrinathPhatak98]. A very common model is the checkout/checkin model [KatzWeiss84], that locks all data items hoarded by a client until they are reintegrated after mobile work. This model disallows updates to data items which are checked out by a mobile client. In an environment which requires the possibility of concurrent updates another method of concurrency control must be used. Timestamp based mechanisms allow concurrent updates and guarantee consistency by checking timestamps on reintegration.

### 3.2 Mobile computing devices

In this section different categories of mobile devices are presented. Their different possible usage and the kinds of services a server must provide to the different clients are described.

### 3.2.1 Laptop Computers

Today's laptop computers can be described as small, lightweight versions of desktop computers. Their architecture is similar to desktop computers and they are equipped with processors of comparable computing power. Their main memory size and hard disk space are normally smaller than in stationary computers, but sufficient for execution of standard software. Typical laptop computers (Figure 3.2) today have a display size of at least 12.1" up to 15", 64 MB RAM, several gigabyte of hard disk space and CPUs running at speeds up to 600MHz. Integrated into the devices are floppy disk drives, CD or DVD drives and communication devices like modems or LAN adapters. Devices with dimensions between 1.4"x11.8"x9.4" and 1.0"x10.2"x8.0" weigh between 5.6 lb. and 2.9 lb. [IBMTSPspec00].

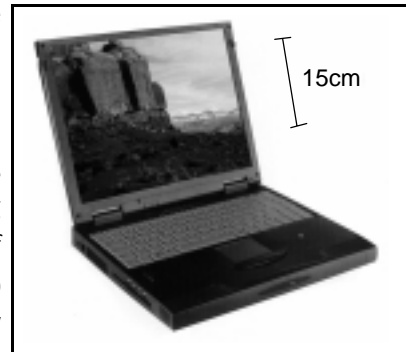


Figure 3.2: Laptop computer

Via docking-stations a laptop can be turned into a desktop computer connected to networks and other resources like printers, greater displays and keyboards or special pointing devices.

If connected, a laptop computer is not distinguishable from a desktop computer. Only the performance of the mobile device might be a bit worse compared to the desktop computer.

Laptop computers are the class of mobile devices this thesis assumes when discussing mobile execution.

### 3.2.2 Personal Digital Assistants (PDA)

PDA's were first used as electronic organizers or notepads. Thus, there was no need for a great amount of main memory or a stable storage like a hard disk.



Figure 3.3: Typical PDA

The computing power of these devices is comparable low, enabling simple tasks, but nothing requiring a lot off computations. In recent months however, PDA's have been discovered by the industry as possible replacement for laptops in certain environments. Software of increasing complexity is written for PDA's, the hardware is upgraded with more main memory and even small hard disks. PDA's are more and more able to do more complex processing, but will not replace laptops in total. This is, because the size of PDA's and therefore the size of their displays make the execution of applications impossible, that need a lot of space for presenting data.

PDA's (Figure 3.3) have between two an 16 MB main memory, approximately 6" display size and normally no hard disk. Connections with other computing devices can be established via an infrared interface, cable connection or antenna [PalmVISpec99]. Some PDA's offer the possibility to attach mini hard disks via PCMCIA compliant interfaces. In that way, the

capacity of storage for data on PDAs can be extended dramatically. With a size of for example 5.06" x 3.17" x 0.67" a PDA weighs only about 6.8 oz. .

### 3.2.3 WAP enabled devices

Even smaller than PDAs are devices like mobile phones (Figure 3.4) or pagers. As these devices have even less computing power than PDAs, special precautions are taken to enable the use of this category of mobile devices. For example a communication protocol especially designed for wireless, i.e. mobile communication, is used by these devices. The Wireless Application Protocol (WAP), standardized by the WAP Forum in [WAPSpec99], is used to transfer data between client and server. Based heavily on existing Internet standards the WAP specification adds several extensions, where special definitions for mobile components are needed. The so called Wireless Markup Language (WML) is used to define the way data is represented in a micro-browser, whose definitions are part of the WAP specification, as are definitions for a lightweight protocol stack.



Figure 3.4: WAP handy

Although wireless communication is the main focus of the WAP Forum's work, the possibilities to use WAP-compliant devices for mobile execution of workflows are limited.

The main goal of WAP devices is, to provide users with information while they are mobile. Therefore the emphasis lies on the presentation of data not on the user working with that data. Typical WAP devices like mobile phones are not suitable for doing work, which requires a lot of user interaction. The small display and limited input devices support only basic interaction between user and device, such as entering requests for data or acknowledging some information.

Thus WAP devices could be used in a workflow environment to show users' worklists and to receive input, signaling the begin, suspension or end of work. But they can not support the user doing her work.

## 3.3 Infostations: a solution to expand mobile computing

The main problem of mobile computing is the nature of mobile communications. Because a wireless wide area network (WAN) has to cover a far greater area than fixed local area networks (LAN) to allow mobility, the communications suffer from low bandwidth, high delays, poor availability and bad reliability. Today's radio technology allows high bandwidth connections only in small areas. Wireless LANs like WaveLAN, RadioLAN or HIPERLAN offer 10 to 20 MBit/s data transfer rate in a range of several hundred meters. While wireless WANs like AMPS, GSM and MODACOM with a range of several kilometers transfer data with 9,6 kbit/s on one radio channel [Tanenbaum96], [Baumann98].

But even the greater areas of wireless WANs can benefit from the high-bandwidth wireless LAN connections. The area covered by the wireless WAN must then be structured as described in the next section [Goldmann97].

### 3.3.1 The infostation infrastructure

The goal of infostations is to decrease the number of connections via wireless wide area networks. Instead most communication is done in areas of limited size, where more reliable, high-bandwidth connections via a wireless LAN are possible. These areas are called *infostations*. Several infostations are placed throughout the coverage area of the WAN (Figure 3.5). The users use the infostations to obtain data, whose transfer via the unreliable WAN would not be efficient.

An infostation works like a cache for the data mobile users need. It is part of a stationary network, which is used to transfer data between infostations and data

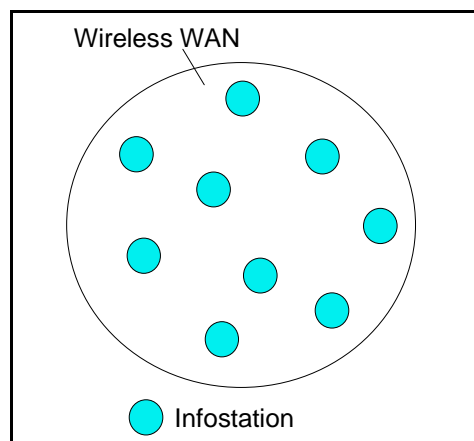


Figure 3.5: Infostation structure

servers. If users connect to an infostation, they either get requested data directly from the infostation, if the data is stored there or the computer system of the infostation fetches the data from an appropriate data server and transfers it to the requesting client. If the storage capacity of an infostation is exceeded, the least recently requested data is discarded to provide space for new data. Infostations can therefore be interpreted as LRU-caches for the mobile clients.

### 3.3.2 Benefits of infostations

Mobile computing benefits from the use of infostations in several ways. First, the possibility to connect to central stationary systems via stable high-bandwidth connections allows the execution of further applications on mobile devices. Those applications are characterized by their need of great amounts of data. With the limited capacity of wireless WANs those amounts could not be transferred efficiently enough. With infostations the amount of data is not such a great obstacle anymore. If it does not exceed the storage space of the mobile devices, great amounts of data can be transferred at an infostation. Data which is already processed and not needed any longer can then be exchanged for new data at the next infostation.

The second benefit of infostations is the greater flexibility of wireless communication for all mobile applications independent from the amount of data to be transferred. If applications urgently need some information, they can use the wireless WAN. Alternatively the applications can use the infostations, if the

requests for data can be delayed until a infostation area is reached.

The infostation structure is a good basis for systems supporting spatial-aware applications. The data provided to an application is selected using information about the location a user is currently at. This data is of interest in a limited area only. Infostations are used in systems like NEXUS [Nexus99] to provide the users with the information important in the limited area of the infostation.

## 4 Workflow execution with mobile clients

In this chapter the environment of a workflow system supporting mobile clients is presented. This environment builds the basis for the discussion in the following chapters.

The discussion starts with an overview over the changes to the workflow system's architecture necessary to support mobile clients. The added and modified components are described in comparison to the system components introduced in chapter 2.3.1 . After the new architecture is explained, two possible modes of mobile execution are presented. One is mobile execution without wireless reconnections to the WFMS, the other is mobile execution with reconnections to the WFMS. With the later, focus is especially on the possible usage of the '*infostations*' concept.

### 4.1 The modified scenario

The overall picture of a workflow system, with a central WFMS-server and its assigned database server serving multiple clients, remains the same, when mobile clients are integrated. The clients still get their tasks assigned by the WFMS, which uses the database system to store the workflow definition and results of activities. But in order to support mobile clients the system's components must be modified according to the extended client-server model presented in chapter 3.1.2 . The general reasons for extending the client-server model given in chapter 3 result from the use of mobile devices. They are put in concrete terms for workflow environments in the following.

#### 4.1.1 The client as a workflow server

As all mobile clients must emulate some server functionality, mobile clients in a workflow system must be able to take over several functions of the WFMS's server. The coordinating tasks of the WFMS must be carried out by a local server on the client, when it is not connected to a stationary server of the WFMS. The single aspects of these coordinating tasks are described in the next section along the modified client architecture. The other important task of the local server is to redirect the application's requests for data to the local storage instead of their location in the stationary system.

#### 4.1.2 The need for a local storage on mobile clients

In a stationary workflow system client device do not necessarily need storage capacity beyond the storage needed for the execution of its applications. The data used to define a workflow is stored in the database of the WFMS as is a major part of the operational data. Operational data not stored in a database is distributed across the system. It is comprehensible, that stationary clients can access the data either via the database management system or by directly requesting it from the other participants of the system.

It is clear, that mobile clients must use a different approach in accessing necessary data, if working disconnected. In disconnected mode clients can only

access local resources, therefore it is necessary that mobile clients of a workflow system store the workflow data and the operational data in a local storage.

Two problems must be solved in association with the local storage: *capacity limitation* of client devices and possibility of *concurrent accesses* on the data. The latter always exists, if copies of data are distributed in a system. The problem with such copies in a workflow system is, that the possibility of concurrent updates of database items is far greater than of data in a filesystem. The CODA system allows updates in disconnected mode, because less than 1% of write accesses to a file are followed by write operations of other users [CODA92]. However, this approach is not feasible in a database environment, as stated in [CODA90]. Therefore other mechanisms must be used to prevent inconsistencies resulting from concurrent updates. Checkin/checkout mechanisms are too restrictive in workflow environments, because they prevent other users from accessing data items hoarded on a mobile client. Timestamping or multiversion concurrency control techniques known from database theory [ElmasriNavathe94] are feasible approaches. If an access operation is identified as invalid at reintegration time, the activity of the workflow becomes invalid and must be compensated, using the compensation means defined for the workflow. The problem of limited storage capacity on mobile clients prevents the workflow system to copy all data in the system to the client. Instead a selection of data necessary for mobile work must be made. The selection process must mainly run automatically, because otherwise it would reintroduce coordinating tasks to the users. That would reduce the benefits of a WFMS in terms of efficiency and distract the users from their actual work.

To cope with these problems, the architecture of client and server must be modified. The necessary changes are described in the following section.

## 4.2 Modified system architecture

In order to enable the use of mobile clients, the system architecture of a WFMS must be expanded to deal with the special needs of these mobile clients. The server and the client must be modified to be able to do so.

### 4.2.1 The modified workflow server

Because mobile clients cannot be treated the same way as stationary clients the workflow server must be aware if it is dealing with a mobile or with a stationary client. The way stationary clients are treated remains the same as described in chapter 2. But to be able to deal with mobile clients the workflow server needs some added components and a few modifications to existing ones. The new system architecture is shown in figure 4.1.

In addition to the components described in chapter 2.3.1 the WFMS needs two new components. One used before a client can begin mobile work, the other after a client has done its mobile work. The first is a component which provides the client with necessary data before it disconnects from the workflow server. This component is necessary, because a mobile client must get the data for its work in advance, not only at execution time as with stationary clients. The component covered in detail in chapter 6 is called '*hoarding system component*'

or shortly *'hoarding system'* in the following. The second new component is responsible for the synchronization of data items after a mobile client has done

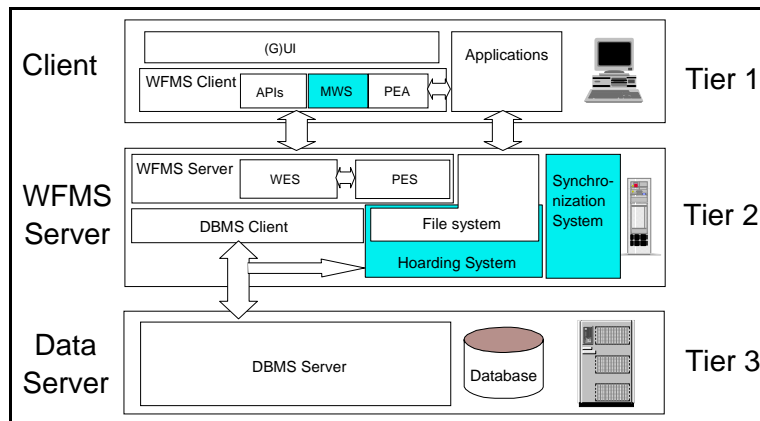


Figure 4.1: Modified WFMS architecture

it's work and has reconnected to the workflow server. With stationary clients such a synchronization mechanism is not necessary, because the clients accesses are synchronized on the data items themselves. The file system or a database management system make sure, that no operations are performed on data items, which could lead to conflicts. But as mobile clients get their own copy of necessary data, synchronization can not be done on the data items. Instead a synchronization component must deal with possible conflicts using the techniques described above.

Finally the functionality of a component already in a WFMS must be expanded. The navigation component of the WFMS must behave differently to mobile clients, than to stationary clients. As with the data, the client has to get information, on which activities to work on while mobile, prior to disconnection from the server. Therefore the navigation component has to build a set of activities called *'workload'* suitable for mobile execution for the client. This process is described in chapter 5.

#### 4.2.2 The modified client

The main difference between a mobile client and a stationary client is, that the first is not connected to the WFMS while doing its work. But the client needs certain services of the WFMS to work properly. Therefore it is necessary, that the client assumes tasks of the workflow server during disconnection time. A

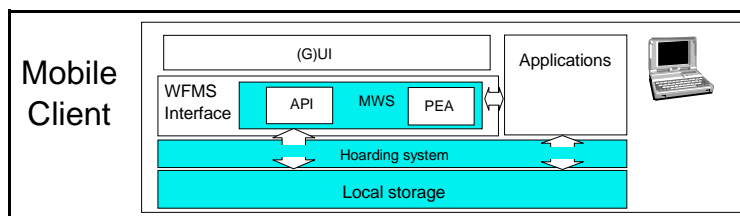


Figure 4.2: Architecture of mobile client

mobile client needs components to navigate through the tasks of its workload, to assign workitems at the correct point of time, start applications and handle input and output data. In other words, the client needs a small version of a

WFMS. This component will be called *'Mobile Workflow Server'* (MWS) in the following (Figure 4.2).

Additionally the client uses a component to improve the decision process of the hoarding system. It collects data on user's accesses on data items during mobile work as well as so-called 'hoard misses', i.e. requests for data items, which have not been transferred to the client. Based on these information, the hoarding system can improve its decisions which data items are necessary and which are not. That way peculiarities in the data accesses during mobile work are taken into account in future hoarding processes.

### **4.3 Mobile execution of workflows**

This section describes the way, in which workflows are executed with mobile clients. Four phases are distinguished by the necessary tasks and the work to be done by the user and the workflow system.

#### **4.3.1 Mobile devices as stationary clients**

In the first phase, the client device is connected to the workflow server via a medium, allowing stable and high throughput connections. This could be a wire connection or a radio LAN with limited extension. This environment allows the WFMS to treat the mobile client like a stationary client. If an activity is ready to be started, the workflow server determines the clients, which could support the activity and puts the activity on the worklists of those clients. As soon as a user starts the execution of the activity, it is removed from the other clients' worklists. The data needed for and during execution of the activity is transferred via the network to the client, if it is not already stored on the client device. The hoarding system component monitors the data access operations of all clients, mobile and stationary. The results of this monitoring are used later, during the preparation of clients for mobile work. At the end of the activity the results are given to the workflow system and the execution of the workflow continues. The phase ends, when the user tells the system, that she is going to disconnect from the WFMS to do mobile work.

#### **4.3.2 Preparation of the client for disconnected operation**

The signal to disconnect is the starting point of the second phase of mobile execution. The WFMS does not assign any new tasks to the client, instead it begins the preparation for mobile work. First it determines which activities the user could work on in disconnected mode. The selection criteria for these activities are presented in 5.3.2 . The selected activities are locked, to avoid the assignment to other clients.

After the activities have been selected, the system searches for data items necessary to execute them. The hoarding system tries to find regularities in the access patterns recorded during earlier executions of the workflow and queries the other sources of information described in 6.2 . The data gathered in this process is meta data on the data needed for the activities and is therefore used

to identify the necessary data. The identified data is transferred to the client device and the user is informed, that the client is ready for mobile work.

### 4.3.3 Disconnected operation

Mobile work starts, when the client device is disconnected from the workflow server. From this point on the client relies on the data in it's storage until it reconnects to the server. Two ways mobile work can be done are described in the following

#### 4.3.3.1 Mobile work without reconnection

In this mode the client does not reconnect to the server until the mobile work is finished or cannot be continued. This mode is necessary for mobile devices lacking the ability of mobile connections or in cases in which the reconnection is not efficient in terms of costs or data transfer.

Because no reconnections are made, the client must get the necessary data for all activities in the workload prior to disconnection. Therefore the requirements on storage capacity are greater than with clients which can reconnect to the workflow system. Another problem are activities requiring the most up to date data. If no mobile connections are possible, the client gets data valid at disconnection time, not the data valid at execution time of the activity.

The advantage of reconnectionless mobile work is the independence of the user from the environment in which the mobile work is done. The client does not depend on the availability and stability of networks or the accessibility of the workflow system. The client can work autonomously.

In summary it can be stated, that mobile work without reconnections is possible, if the activities do not need the most up to date data. Problematic is the determination if an activity does need current data or not. The user can decide that based on knowledge about the meaning of the activity or the developer of the application supporting the activity makes the availability of current data a requirement for the application.

#### 4.3.3.2 Mobile work with reconnections

If activities need current data at execution time, a mobile client must connect to the workflow server when the activity is started to get the current data values. How it does that depends on the environment the user is in at execution time. If available the user might connect a modem to a telephone line or use a radio network. The problem with these media is, that they are very expensive compared with the costs of data transfer via a direct, wired connection to the workflow server. Therefore these connections should only be made, if absolutely necessary, for example to get the latest data, and not to transfer data possibly needed sometime. The hoarding system still puts most data items on the client device prior to disconnection from the server.

The concept of infostations (Chapter 3.3) allows to do mobile work in another way. Using infostations, it is possible to transfer greater amounts of data between client and workflow system. The greater reliability and bandwidth of wireless networks in infostation areas improves the efficiency to a reasonable level.

Even mobile devices with small storage capacity can execute multiple steps of a workflow, if infostations are used. In that case only the data for a limited number of steps must be loaded to the device prior to disconnection. This is done to enable the working on activities until a infostation is contacted. At an infostation the data of activities already done is then exchanged with data for new activities.

With infostations the necessary amount of storage space in a mobile device can be reduced and the user gets newer data than available at disconnection time. But activities needing current data still require mobile connections at execution time.

Mobile work with reconnections to the workflow server allows a more flexible approach and is necessary in cases where the data must be as current as possible at execution time. When reconnections via infostations are possible, the user can use mobile devices with less storage capacity but becomes dependent on the infostation infrastructure.

#### 4.3.4 Reconnection at the end of mobile work

After mobile work is done, the results must be integrated into the workflow. To do this, the client connects to the workflow server and transfers information on activities done successfully, unsuccessfully or activities not started during disconnection. The results of the successful steps are given to the synchronization component, which integrates them into the workflow. Steps, which have not been started, are unlocked in the process instance and can be assigned to every client of the WFMS, qualifying for the activity, again. The steps completed unsuccessfully are treated dependent on the reason which caused the unsuccessful completion. If an error occurred because necessary data was missing, i.e. hoard misses occurred, the system reassigns the affected activity. If all data was present but the activity did not complete correctly, the step is treated as faulty and the WFMS's actions depend on what was defined in the process definition for failure of that step.

Additionally to the data transferred between client and WFMS, the information gathered by the client's hoarding system component is made available to the central hoarding system.

After all data is transferred, the client is treated as a stationary client again.

When data is exchanged at an infostation, the same mechanisms are executed, only the client is treated as mobile again after disconnection from the infostation.

## 5 Activity Streams

A mobile user cannot take all activities which are contained in a workflow with her. This can have different reasons. For example the mobile computing device of the user might lack the computing power to perform an activity or an activity depends on data which comes from another activity and is not yet available. In this chapter a set of rules is proposed, which enables the WFMS to determine the activities which can be taken along by a mobile user. But before these rules are presented, it is necessary to explain the notion of *blocks*, *subprocesses* and *streams* in a WFMS.

### 5.1 Blocks and subprocesses

An activity of a workflow is not necessarily implemented by a single program. Such an activity is called program activity. Instead it might be realized by several smaller activities. Depending on the purpose such a collection of activities building an activity are called blocks or subprocesses. The following describes the different purpose of blocks and subprocesses.

#### 5.1.1 Blocks

Some business processes contain activities which must be executed multiple times. For example in an order system, different items must be assembled to one package and each must be deregistered from the stock. The process of putting an item into the package and deregistering that item is repeated until all items requested by a client are successfully put into the package.

Since the process definition of a workflow is an acyclic directed graph, multiple execution of the same activities can not be realized by looping back to the first activity to be executed multiple times. Instead the concept of a *block* is used to provide this functionality. A block is an activity or a set of activities to which a special exit condition was assigned. When the activities in the block have completed, the exit condition is evaluated. If it evaluates to false, the whole block is repeated by rescheduling its starting activities. That way a do-until-loop can be created allowing the multiple execution of several activities until a given condition is met.

#### 5.1.2 Subprocesses

Activities of one business process can also be implemented by another business process. This allows the reuse of process definitions in more complex workflows. Thus new workflows can be built using already existing implementations of business processes (Figure 5.1).

Such *subprocesses* can be controlled in different ways. They can be carried out by the workflow system, which controls the workflow the subprocess is part of. Another possibility is the execution of a subprocess by a different WFMS. The standard defined by the Workflow Management Coalition contains the description of an interface, which allows the transfer of control to other workflow systems. Thus, it is possible to combine business processes in different

locations using different computer systems to one large business process.

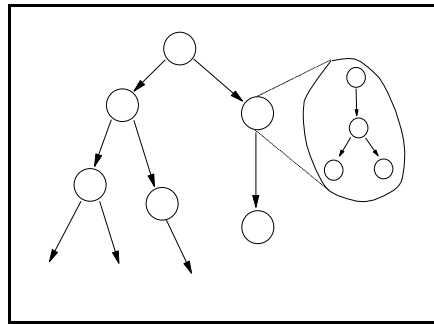


Figure 5.1: Activity implemented as subprocess

The need for efficient business processes in a global market, where multiple companies are part of a business process forming a virtual enterprise [LeymannRoller99], makes the concept of subprocesses an important part of workflow management.

## 5.2 Streams

Streams in WFMSs are sets of activities to be performed by the same agent, i.e. the same user or machine. A stream contains consecutive tasks of one workflow instance. The fact that one agent will perform the stream of activities does not mean that the workitem stream is put to that agent's worklist only. The characterizing fact for an activity stream is, that once a single task out of the stream is started by an agent, all the other tasks of the stream are performed by this agent. In this case the WFMS assures the deletion of the activity stream from other agents' worklists. Streams offer a performance improvement while navigating a workflow, because once a agent started and finished one task, the WFMS can start the following step(s) without delay.

The idea of streams is used by the WFMS in three main ways: The micro script stream, the transaction stream and the workpackage stream.

### 5.2.1 The micro script stream

This kind of stream is used, if a workflow does contain a sequence of activities, which can be worked on without the intervention of a human agent. This stream can be processed very fast, because the WFMS can start the execution of an activity on the chosen agent directly when it detects a workitem ready to be scheduled.

The activities of such a stream could be implemented together in one program and some WFMSs do allow the compilation of a micro script stream into a single program. In that case the benefit of a faster execution is bought with the decrease of flexibility. If the workflow changes the compiled micro script stream has to be recompiled.

### 5.2.2 Transaction streams

The concept of streams can be used to provide transactional behavior of a set of activities. If the stream includes manual activities the system uses the stream to

execute some 'dialog control' reflecting a programming technique, favored by TP monitors, called pseudo-conversation [BernsteinNewcomer97]. This technique solves the problem, which would arise, if a lot of user interactions and screen navigation would be implemented in one single transaction. Instead of such a huge transaction a chain of small, fast transactions is used. Each of these transactions implements a logical end-user step. Each transaction passes its data to the successor, which shows the user the data and demands the next input. The system gives the user the impression, that a conversation between her and the system takes place, which is not the case, it is only 'pseudo-conversation'. If no user interaction is necessary the transaction stream is in fact an atomic sphere with exactly-once semantics as described in chapter 2.2.3 .

### 5.2.3 Workpackage Streams

Workpackage streams are sets of activities suitable for checking out as a whole unit. After the work is done, the stream is checked in again. The WFMS provides the means to check out the whole set of activities for processing in disconnected mode. Of course it must also provide means to check in these activities after reconnection. The meaning of checking out activities is the following. On one hand the user who checks out one or more activities declares her intention to work on these activities. The WFMS on the other hand guarantees, that no checked out steps of work appear on other users' worklists. After return from mobile work the workpackage stream is reintegrated into the workflow as described in 4.3.4 .

Workpackage streams checked out for mobile execution are the activity streams discussed in this thesis.

The following sections describe how the WFMS can determine those activity streams.

## 5.3 Determination of activity streams for mobile processing

This section explains a possible approach to enable WFMSs to (semi-)automatically determine a set of activities, which a user shall take along for mobile processing. First some helpful terms are presented, which make the understanding of the thereafter following rules and the definition of the rules themselves much easier. The rules are explained using a graphical representation of a workflow.

### 5.3.1 Types of nodes and their use by WFMSs

The definition of workflows include the information needed for an ordering of the activities building the workflow. Most WFMSs use a graphical representation to illustrate that ordering. The activities are represented as nodes and the control connectors as directed edges of an acyclic graph. Five kinds of nodes can be identified:

- Start nodes:** These are nodes with no control connectors pointing to. They represent the initial activities to be performed, when a workflow is started. They have no effect on the determination of activity streams, but they may be part of one.
- End nodes:** If a node has only incoming control connectors and no outgoing ones, the node is called end node. Like the start nodes they have no impact on the determination of activity streams, but can be part of a stream.
- Regular nodes:** The term regular nodes is used for nodes with exactly one incoming and one outgoing control connector. They are the most common node in typical workflow definitions. They can be part of activity streams, but their nature does not affect the determination process.
- Fork nodes:** A fork node has, as figure 5.2 depicts, at least two outgoing control connectors. Fork nodes start multiple following steps for potential parallel execution. Fork nodes should be taken into account during the determination process.
- Join nodes:** Join nodes are used to synchronize multiple parallel paths in a workflow. Thus, their characteristic is that they have at least two incoming control connectors (Figure 5.3). Join nodes must be taken into account during the determination process. Different join conditions can be specified. From the condition, that all connectors must be evaluated successfully to the condition, that only one connector must be valid, all different possibilities can be chosen.

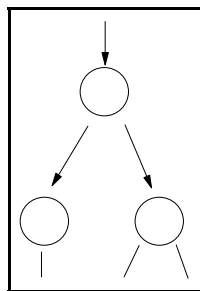


Figure 5.2: fork node

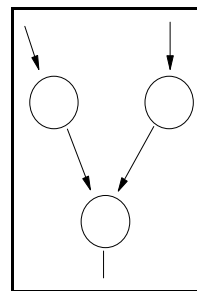


Figure 5.3: join node

Each of the nodes represents an activity and thus can be a program activity, block or subprocess. The different kinds of nodes and different activity implementations are taken into account during the determination of possible activity streams.

### 5.3.2 Rules for determining activity streams

Several aspects must be taken into account, when an activity stream is determined by the WFMS. The user must be able to execute all activities put in the activity stream. To be able to execute an activity, a user must have the right to execute it and the activity must be assigned sometime to the user. The determination mechanism must therefore guarantee, that only activities are selected whose execution demands only rights the user possesses. Further the part of the workflow becoming an activity stream must be navigable from the beginning to the end. Activities, which can never be reached because results from outside the stream are missing, can not be part of the workload. The rules, which must be followed to determine a valid activity stream are presented in the following:

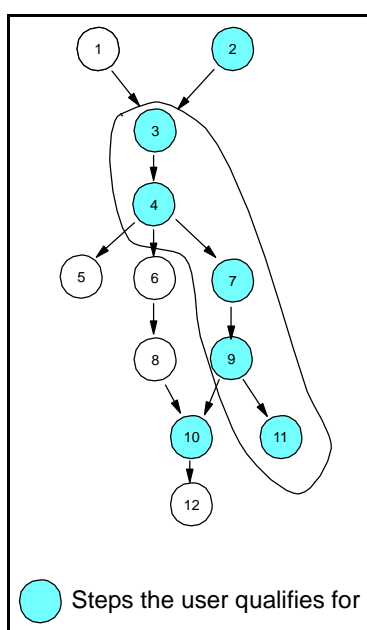


Figure 5.4: Example of an activity stream

Figure 5.4 shows the example used to explain the rules for determining activity streams. The nodes 1 and 2 are start nodes, the set {6,7,8} contains regular nodes, nodes 4 and 9 are fork nodes, while nodes 3 and 10 are join nodes and {5,11,12} are end nodes. The set {3,4,7,9,11} represents a valid workpackage stream. It is determined by the WFMS according the following rules:

1. If a node is part of a predefined activity stream, the node can only be part of the workload, if the whole stream is part of the workpackage. The same applies to an atomic sphere. Either the whole atomic sphere or none of the contained steps is part of the workload.
2. A node can only be part of a workpackage for a specific user, if the user qualifies for the task, i.e. the user is in the set of possible agents returned by the staff query for the activity which the node represents.

3. Nodes can only be part of the workload, if all their input data is available at the time the assembly of the workload is requested. Otherwise the situation can occur, that a user's work is blocked, because the system cannot provide needed data to start the implementation of an activity.
4. If an activity is implemented as a block, the whole block must be part of the activity stream or no activities following the block can be in the workload. Therefore the block must be recursively analyzed by the determination mechanism. If the determination stops inside the block, the path containing the block is not evaluated further and the determination mechanism checks other paths of the workflow if available.
5. Subprocesses are treated principally the same way as blocks, but are subject to another restriction. It is only sensible to include subprocesses, that are executed by the WFMS that controls the current workflow.
6. After the first node, all following nodes are potential candidates for the workpackage, as long as they follow rule one. For each node in the workload the follow-ups are evaluated. This process is repeated until the set does not change anymore or a join node is reached. The different node types must be treated differently:
  - Regular nodes can be put directly into the workload.
  - With fork nodes the system has to take into consideration, that the benefit of the parallelism of the fork nodes following nodes is void, if they are processed by the same user. Nonetheless could those nodes be put into the workload.
  - If a join node is encountered the system must check if the join node can be inserted. This is the case, if all predecessors are part of the workload or the system is able to evaluate the conditions of all control connectors coming from those nodes not in the workload. Otherwise, the join node would never be ready to be worked on and would block following steps of work.
7. An optional restriction of the workload could be the time factor. If this factor isn't taken into account, the workload could grow very large. Normally the user has only a limited amount of time in which to work on the tasks in disconnected mode. So information from the audit trail of the WFMS could be used to determine the average time needed for completion of a task. Further nodes, i.e. tasks are only taken into the workload, if they do not break a set time threshold.

### 5.3.3 Example for workpackage determination

Using the workflow example shown in figure 5.4 the usage of the rules is presented in the following:

At the begin of the example node 3 is assigned to the user, but not started yet. Therefore it is the starting point for the algorithm. The node is not part of a predefined activity stream nor part of an atomic sphere. All input data is available and node 3 is a regular node. Because all requirements are fulfilled, the node can be part of the workload.

Following the algorithm, the successors are checked. Node 4 is the only successor and because all requirements are met, node 4 can also be part of the workload.

It is assumed in the example, that the user, requesting the building of the workload, only qualifies for node 7 out of the successors of node 4. Therefore the nodes 5 and 6 cannot be part of the workload, but node 7 can be.

Node 7's only follow-up node is node 9. Assuming the user qualifies for that node, all other requirements are met and it can be part of the workload too.

Node 10, which follows node 9 might not be part of the workload, because it is a join node, which has a predecessor outside the workload (node 8).

The second successor of node 9, can be part of the workload, because it is a regular node, the user is qualified for and whose input data is coming from inside the workload.

## **6 Data Hoarding**

In this chapter the mechanisms used to provide the mobile user with necessary data prior to disconnection are presented. First the necessary data is described. After that different sources of information about that data are distinguished, by describing what information these sources can provide and how the WFMS can obtain that information. Finally, the mechanisms to extract the data according to the gathered information are explained.

### **6.1 Data needed for mobile execution**

During disconnection phases the mobile user can only access the information stored on her mobile device. Therefore the data needed to do assigned work must be stored on the mobile device while a connection exists. That data can be subdivided into different categories. These categories are presented in the following sections.

#### **6.1.1 Data needed to enable workflow management**

The execution of a part of or a whole workflow on a mobile device can not be done without data about the workflow. The definition of the workflow, i.e. the steps and connectors have to be available to the navigation component of the client. During disconnection this component plays the role of the WFMS's workflow execution server. It puts steps, which are ready to be worked on, on the user's worklist, detects the successful end of an activity and navigates through the workflow definition to find following steps.

Depending on the mobile device's abilities and the way the client component is implemented, the workflow data can not be stored in the same way as it is for the workflow execution server. But it is important, that the content of the data remains the same. Therefore, it is necessary to provide appropriate conversion mechanisms, if the data format must be changed.

Because laptop computers are the mobile devices assumed in this thesis, the clients store workflow data in a database like in stationary environments. The advantage of storing the workflow data the same way as on the stationary system is, that the workflow system's navigation component can be used on the client to control the mobile work. Instead of requesting the information from a remote database, the navigation component accesses the local storage of the client. While mobile, the navigation component assigns the activities only to the mobile user, since she is the only one available. The determination of activity streams made sure, that only activities executable by the user were selected, so a problem with missing rights can not arise.

If the workflow data can not be stored on the mobile device like in the stationary environment a new navigation component has to be developed or at least the stationary one's access functions to the workflow data must be adjusted.

### 6.1.2 Data needed to execute steps

The way a user does her work with a mobile device should not differ from the way the same step would be done with a stationary device. If the user starts an activity by selecting it from her worklist, the associated tool is to be started like on a connected device. To enable this behavior several data items are necessary.

First the application itself must be available. On mobile devices like laptop computers this requirement should not be a great problem, because normally the users already have installed the necessary applications for their daily work.

With devices with limited storage space it might not be possible to store the application code for all steps assigned for mobile work in advance. In this case two reactions are possible.

One is to reduce the workload, i.e. the number of steps, to enable the transfer of all applications.

The other possibility is to require a reconnection of the user to the WFMS after the steps are done, whose applications had fit on the mobile device. During such a reconnection the application code stored is exchanged with code of pending applications. The identification of data belonging to the application code is relatively simple. During installation, the installed data items and their location can be registered in a database, which is queried, whenever an application must be transferred to a client device.

But even if the application code is available on the client, a step can only be executed successfully, if the so-called 'operational data' is also available at execution time. The term 'operational data' describes all data needed during the execution of an application besides the application code. Thus it includes the input parameters of the accompanying application, the data accessed during execution and data objects necessary to store the results of the application.

Because operational data changes dynamically, it is not possible to determine every data item used in advance. Through monitoring of the accesses to data items by the user, it is only possible to arrange these data items according to their probable use by the applications. Data items used regularly are treated as more important than those used less often. If the environment permits, all data items classified as operational data for an application are transferred to the client. But in most cases a decision must be made which data items to take along and which to leave behind. Such a decision based on the probability of user accesses to data items will certainly lead to errors in some cases. Therefore it is a major task of the hoarding system, to improve the process of decision making by not only monitoring the usage of data items during stationary work, but also monitoring the mobile work, recording errors and successful data accesses.

### 6.1.3 Data needed by the hoarding system

The main work of the hoarding system is done prior to disconnection, but a component of it is part of the client. That component monitors the user's behavior during mobile work. The results of this monitoring are used to improve the hoarding decisions in the future. Another purpose of that component can be

the management of necessary reconnections as a mean of treating occurring errors. The main part of data needed by the hoarding system is information about the data in the hoard. The hoarding system on the client must know which data items were transferred to the mobile device. The accesses to these data items are recorded and the value of them is increased to raise the chances of the item to be hoarded again in the future. Other data for this component are parameters describing what kind of user actions must be monitored or which errors have to be managed. This data plays a minor role compared to the operational or the workflow data, because normally only the list of hoarded data items and parameters for programs of the hoarding system on the client must be transferred.

## **6.2 Information sources**

In order to be able to select and collect necessary data for steps, which will be worked on in disconnected mode, the WFMS must analyze meta data about that steps. Several sources can provide these meta data. These are shown in the following sections.

### **6.2.1 Process definitions**

The process definition contains the data needed for the workflow management component of the client. The steps, their order and their dependencies can be extracted from the definition. After that information is transferred to the client, the client can do the administrative work instead of the workflow execution server.

The definition can provide an indication for the necessary applications too. Because the WFMS's program execution server must be able to automatically start the applications supporting the activities, it knows the location and starting mechanism of the applications. This information can be used to determine the necessary components of the application code. But because it is assumed in this thesis that the applications are already installed on the client devices, this point is not looked at further.

Additionally the process definition contributes to the operational data, i.e. the data needed by the applications supporting activities. The definition of the input containers provides information to identify the input parameters of the applications supporting steps of work. The parts of the input containers of steps in the workload are resulting either from steps already finished, i.e. the parameters are available or they result from other steps in the workload. Steps requiring input data from steps not finished at disconnection time can not be part of a workpackage. Thus, all input parameters are available, when an activity is ready to be scheduled.

### **6.2.2 External sources**

External sources are sources of information, which are neither part of the workflow system nor the hoarding system. Their information is gathered via a common interface during the hoarding process. By using a standardized

exchange format, the external sources make their information accessible for the workflow system.

The external sources can be categorized into three groups, described in the following.

#### 6.2.2.1 Data of other applications

Information allowing the reduction of the amount of data to be transferred to mobile clients can be found in data used by external applications. Address lists, time management systems or calendar applications contain information that would allow the categorization of data in groups of data sharing common semantical meanings. Such a categorization would simplify the process of selecting operational data for a specific step. For example the name of a city found in the calendar of an insurance agent for a certain day, allows the reduction of the database entries to be transferred to the agent's mobile device on that day. Only data of those clients living in that city must be hoarded, because it is unlikely that the agent will need the data of people living elsewhere. This approach of using specific values in a database is called *hoard key* approach [BadrinathPhatak98]. To be able to use this approach, the hoarding system must have access to information describing what part of the data of which application can be applied on which database to select data items. This information is best defined during the definition of the business process. If each process definition contains a list of applications, which can be queried for specific steps of work, the WFMS can query them during the hoarding phase.

The problem for the workflow system of having to communicate with a lot of different applications can be solved by defining standard exchange formats. Both the WFMS and the external applications have to support that exchange format. The external application writes the defined information to a prior specified location using the exchange format. During the hoarding process the hoarding system checks the defined location for available information and uses it to reduce the amount of database entries. The use of this approach with files is not easily possible. The selection of database entries using SQL is easier done, than selecting files fitting an criterion describing a part of their content. If files should become part of this approach, it is necessary to enforce specific directory structures. That way the selection of files can be based on their storage location. It is however difficult to change an already existing system to the needs of this approach. In new systems the idea can be taken into account during the design phase and then be used when the systems have been built.

#### 6.2.2.2 Application specific functions

The developers of an application for use in a WFMS can use their knowledge of the details of the application, to provide specific functions with their applications, which collect the necessary data. For each step these functions are called by the WFMS, when it has determined the steps in the workload. An appropriate interface is provided by the WFMS. The developers could then implement that interface by writing the functions as part of the application development process. The functions could run automatically or require special user input, as the developers see fit. Application specific functions can return meta data on

necessary data items like external applications. Additionally important data items can directly be the result of those functions. The first possibility allows the integration of meta data from external applications and from applications, which support activities of the workflow, into the hoarding system via one interface. This approach is followed in this thesis. The second approach, to return necessary data items directly upon a request, has the advantage, that the hoarding system does not need to interpret the meta data returned by the applications to find the data items. But the disadvantage is, that the concrete data items must be known at development time. This may be possible in some, but not in all cases. Therefore this approach is not followed further.

Even already existing applications can be enhanced to be able to provide hoarding information. Because it is not possible in most cases to integrate added functionality into the applications, the new functions must be added 'around' the application. This is comparable to the way legacy applications can be integrated into a workflow system. The problem with legacy applications is, that they were

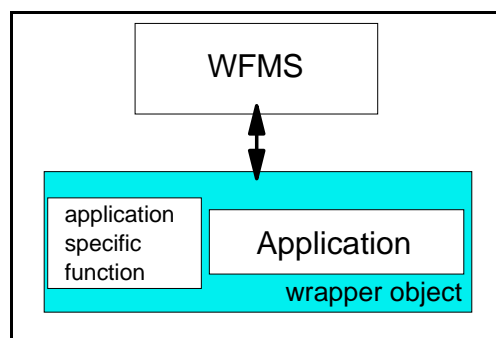


Figure 6.1: Wrapper object

never programmed for use in a workflow system. Therefore special *wrapper objects* are implemented. These can be called by the WFMS and handle the workflow specific work, like fetching input parameters from the input containers, invoke the correct parts of the legacy application and put the results in the output containers. In the same way, an application specific function can be added to an existing application (Figure 6.1). It is however not easy to determine the information to be returned by the application specific function of an existing application. The knowledge of the developers of the application is normally not available to the designers of a workflow system. They have to fall back on their knowledge of the meaning of the application or striking patterns in the accesses to database items recorded for an application.

### 6.2.2.3 User input

Another possibility to gain information on needed data is the concrete input of selection criteria by the user. During the building of the workload, the WFMS queries the user for her input. The entered parameters are then used to identify the needed data for specified activities. The user can also request certain data items directly at the end of the hoarding process. The hoarding system shows the selected data items to the user and she can alter the selection before the data items are transferred to the client device. That way, the user makes the final decision of what to put into the client's local storage. It is assumed, that

the user knows better what she might use for mobile work. If the system's proposals become better, because the decision process improves from execution to execution, the user will have to correct less each time. But she will still be able to request special items.

Theoretically the user's possibility to overrule the decision of the hoarding system can lead to a situation, where the system proposed an item, which the user rejected, but which is actually needed during mobile work. However this is seen as a rarely occurring exception.

### 6.2.3 Monitoring component

Valuable information for determining relevant data comes from a component, which observes the actions of a user while working on an activity. Especially the data accesses of the user are interesting, because if the system can recognize a regularity in the data access patterns of an activity, the probability that the same pattern is used in future executions is high. The data items accessed in such a pattern are first candidates for the hoarding mechanism to be transferred to the mobile client.

The monitoring component must be integrated into a layer between the application and the data. For example it is integrated into the file system to monitor file accesses. This approach is taken in file hoarding systems like CODA [CODA92, CODA98]. However in a workflow environment file hoarding is not enough. Database accesses must also be monitored, even in a very precise manner. To only detect which database is accessed is also not enough, because in a lot of cases it is not possible to transfer whole databases to mobile clients. Technically it is quite possible to monitor every access to a database within the database management system (Figure 6.2). The problem with such trace components is performance. Every SQL-query is redirected to the trace component, which records it and executes it on the database. The performance

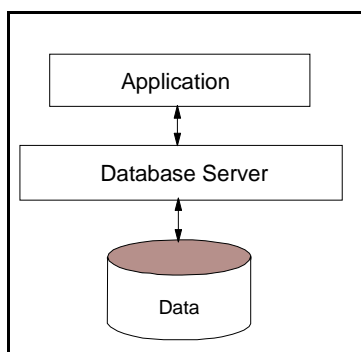


Figure 6.2: Normal access to database data

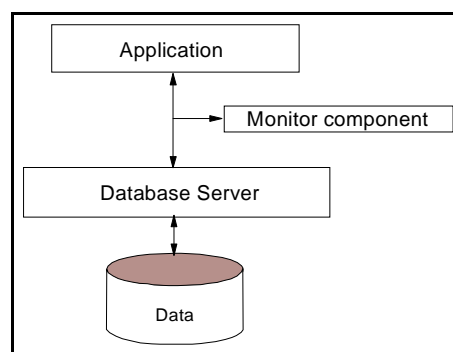


Figure 6.3: Database Access with monitor

decrease when using the trace functionality is so important, that these functions are only used, if the reason of a problem must be found.

Principally the performance problem does also exist with file systems. But because CODA was shown to be feasible, the integration obviously does not have a too negative impact on the user. As no statistical information was available on the impact of trace components on database systems, it is assumed, that the effects would affect the user.

To avoid the performance problem, the monitoring component must not be integrated into the database system. Instead it is installed parallel (Figure 6.3) to the database and the queries and their results are sent not only to the client and the database respectively, but also to the monitoring component, which records them.

To distinguish between important and less important data items a simple counting mechanism is used. When the users access a data item, the counter is increased. By periodically decreasing the counters the system is kept up to date. Data items, which used to be accessed often but which were not accessed during the last executions of the workflow, become less important over time, reflecting the change in the users' behavior. The decrease of the counters must not be done too fast, because otherwise one time changes have too much impact on the hoarding process.

By providing initial values for the counters of special data items the monitoring component's behavior at the beginning of its usage can be influenced. Over time the system adapts to the actual need by monitoring the users' accesses and the results of previous hoarding. The monitoring component on mobile clients increases the counter of data items of a workload, which were actually accessed during disconnected work. More important however is the increase of the counters of data items recognized as hoard misses. The counters of these data items must be increased higher than those of data items which were in the workload but were not accessed. In that way the possibility of a reoccurring hoard miss is reduced. Counters of data items not accessed during disconnected work must be decreased. By defining how much the counters are increased or decreased in certain situations, the way the monitoring component selects data items can be controlled. The most simple form to interpret the counters as access counters. A counter is increased by one, whenever an access is recorded. The frequency of periodic decreases depends on the frequency workflows are executed. If a workflow is executed multiple times a day a daily decrease is sensible. For workflows executed only once in a while, a less frequent decrease is necessary. As stated above the counters of hoard misses must be increased to a level above the counters of unneeded data in the hoard. If all data items in the hoard were accessed, the counter of hoard misses is set to the counter of the least important data item in the hoard.

## **6.3 The hoarding mechanism**

This section describes how the different data sources are queried by the hoarding system and how the obtained information is used to retrieve the data needed for disconnected operation of a client.

### **6.3.1 Gathering the information**

In the first phase of the hoarding process, the system must gather information about the data to be hoarded. To do this the different sources of information must be checked. The presented order in which the sources are queried, is determined by the importance of the sources to the hoarding system and the WFMS.

### 6.3.1.1 The gathering process

For every step in the workload all possible sources of information are queried before the systems continues with the next activity. This approach avoids, that steps get only part of the necessary data because of missing storage capacity. The order in which the data is searched for the single steps is given by the workflow definition. That ensures, that work can be started and done until data is missing on the client.

For each step the sources are queried in the following order:

1. The process definition. Without the data from the process definition the client would not be able to even start an activity. Therefore this data must be transferred to the client.
2. The special demands a user made are assumed to be very important, therefore the user input is considered before the information from the applications and the monitoring component. The decision of a user to request a specific data item is based on the knowledge of the user, that she will need this data item. In this way user-specific data items can become part of the workload, while application-specific data items or data items used by the average user are automatically selected by data from the following sources.
3. The information provided by a function of the application supporting an activity is used before the monitoring component's information, because it was regarded as important by the developers of the application, to enable a correct execution of the application. The developer of an application used in a workflow might have implemented a special function, which returns a list of data items needed by the application. Because the developer has deeper knowledge of her application, she can identify data items needed by every instance of her program. Thus she is able to provide a function which describes these items for the hoarding system.
4. The monitoring component's information does not guarantee, that the data selected on basis of it is really necessary. Therefore the information is used after the one resulting from sources guaranteeing the necessity of the identified data items. If a monitoring component exists, it can provide detailed information about the data items used while executing an activity. Such a component records the data accesses of each application in a given workflow context. These recordings can be used to determine regularities in the data access patterns of applications in a workflow. Regularities in the access pattern give a good hint for data needed by each invocation of an activity.
5. The information from external applications is used last.

As stated before, the user is presented the results of the hoarding process, before the workload is transferred to the client device. That way, the user is able to correct decisions automatically made by the hoarding system, which are wrong in her opinion. If the user rejects items, which were correctly proposed by the hoarding system, they will gain importance in the next hoarding decision, because they will be treated as hoard misses by the client's hoarding system, resulting in an increase of the data item's 'importance' counter.

### 6.3.1.2 Requirements for the gathering process

Several requirements must be fulfilled for the gathering process to be able to be successful:

1. The WFMS must provide appropriate interfaces to access the data of the process definitions. Additionally the client must be able to store this definitions.
2. A data exchange format must be defined for communication between hoarding system and applications. This format is used by the application specific functions, the external applications and the user input component to return the meta data on necessary data items.
3. A generic interface is used to adapt to the different applications while still maintaining the correct exchange format.
4. An internal weighting system in the hoarding system component must build a hierarchy of data items identified by the monitoring component.
5. The results of hoarding process must be recorded, to be able to improve following hoarding processes.

### 6.3.2 Data retrieval

After the workload is determined, the hoarding system must transfer the data to the client device. As applications store data in different ways, for example in files or in database tables, the hoarding system has to provide different means to retrieve the necessary data items for the applications. Because most applications store their data in files or database tables to allow a standardized access to the data, these two storage methods are assumed in this thesis.

### 6.3.3 Transfer of data items

The retrieved data items must be stored on the client device. As on the server, they can be stored in database tables or in files or in both ways. On laptop computers as mobile devices the data items can be stored in database tables and files without problems. The only requirement for the laptop computer is, that a database system is installed. The operating system of the laptop manages the access to files. The approach to store data items in files and databases simplifies the transfer of data items, because data items stored in files are

stored in files on the client and database entries on the server become database entries on the client. Another benefit is, that the access mechanisms of the applications can remain unchanged. Database items are still accessed via SQL and files via the file system.

To store data items in databases and files is sensible on mobile clients with enough storage space and computing power, but mobile clients like PDAs normally do not have the necessary resources to store data items both ways. Although simple database system exist for PDAs, the overhead of such a system compared to the benefits gained by its usage indicate, that it is more sensible to store data items in files only.

On stationary computers database systems are used to avoid storage of the same data for each application in different locations and to control concurrent data accesses. Because the user works alone on her mobile device, the local storage is not accessed concurrently, therefore this function of a database system is unnecessary. Also the issue of multiple versions of the same data exists with or without a database system. Because the client can update data items without the database system used by stationary clients noticing these updates, it is necessary to synchronize the databases. The same is necessary, if the data items are stored in files. Another problem with just using database tables as storage is that database systems for PDAs support only basic types of data making it difficult to store complex data stored in files on stationary computers.

The use of files as the only way to store data items requires a mapping of database structures to an appropriate file format. It must be considered that the access to data items stored in databases on stationary computers must also be mapped to file accesses, if only files are used. Such a mapping component can be part of the monitoring component on the mobile client. This component's task is the recording of accesses to data items, therefore all requests for data items must be sent to it. The request can then be recorded and mapped to the file accesses. In this way, the monitoring component can record the data item accesses in a form comparable to the requests of stationary clients and still use a different storage mechanism.

In this thesis the data on mobile clients is stored in files and databases as on stationary clients. Even if some mobile devices available today do not support databases yet, the increasing capabilities of such devices will make efficient database support available in the near future. Therefore the argument of easier transfer and unchanged access mechanisms won the discussion in favor of the approach using files and databases.

## 7 Realization of the modified system

In this chapter the principle realization of the new components of the workflow system is presented. The concrete implementation is described in the system documentation along with the javadoc documents. The 'workload determination component' (WDC) is presented first, followed by the description of the realization of the hoarding system.

The WDC is responsible for selecting the activities suitable for mobile execution out of the process definition of a workflow. For the activities determined by the WDC, the hoarding system identifies, gathers and transfers necessary data items.

### 7.1 The workload determination component

Before the parts of the component are explained, a brief overview over the interactions with the WFMS is given. The prototype was implemented using the API of IBM's MQSeries Workflow v3.2.1 [MQWProg99].

#### 7.1.1 Interactions with the WFMS

The interactions with the WFMS are best described by presenting the sequence in which they occur:

1. The WDC must first establish a connection to the WFMS's Execution Server.  
During that process, the user authenticates herself to the system. This ensures, that the access to data is limited to data items, the user is entitled to.
2. A list of processes with the state 'running' or 'ready to run' is requested from the server. As the user has only access to processes in which she is involved, the process list contains only processes which contain activities the user might execute.
3. For each process the activity instances and control connectors are fetched from the WFMS. The transferred data contains the state of activities and connectors.

The selected activity instances are locked for the user, to avoid that the WFMS assigns activities to other users while they are selected for mobile work. Then the selected activities are checked, if they can be part of a mobile workload. All activities already finished or already assigned are discarded, because they will not be assigned to the user. The activities in state 'ready' are the starting points for the workload determination algorithm presented in chapter 5.3.2 . After the algorithm terminated, activity instances qualifying for the workload are stored and handed to the hoarding system for further processing. The activities not in the workload are unlocked to enable their assignment to other users.

### 7.1.2 Elements of the workload determination component

The workload determination component consists of a user interfaces, the determination algorithm and a controlling component. The WDC is started on request by the user. If she starts the preparation of her mobile device, the controlling component is invoked, which is responsible for the invocations of the other components. First the system requests information from the user,



Figure 7.1: Activities selection window

including authentication, the WFMS to connect to and the communication mechanism with the WFMS. The given information is used to start the interaction with the WFMS presented above.

The next step in the workload determination process is the presentation of available workflows to the user. Figure 7.1 shows the graphical user interface used for that purpose. Whenever an available workflow is selected, the available activities of the corresponding business process are displayed. This is done by executing the determination algorithm outlined in chapter 5.3.2. The algorithm returns a set of activities the user can select from the activities she wants to take along for mobile work. Activities contained in the set of activities resulting from the algorithm are guaranteed to be independent from activities not in the set. But dependencies between activities out of the set must still be taken care of. Therefore the component ensures, that the user can not select a set of activities, which could never be processed because of missing results of predecessors. If an activity is selected which needs other activities to be executed before, the necessary predecessors are automatically selected too. If an activity necessary for others is deselected, the dependent activities are deselected as well. This is shown to the user by adding or removing the marks of the affected activities.

When the selection is confirmed, the selected activities are returned to the process, which invoked the WDC. However it is important to notice, that the selection made by the user is not necessarily the final workload. For example a shortage of storage space on the client device might enforce a reduction of the workload. In this case the hoarding system encounters an error while preparing the client for mobile execution. The workload will then consist only of those activities, which were processed successfully by the hoarding system before the error occurred. Activities, which were part of the selection but which could not be processed by the hoarding system, are marked as unselected. This enables the system to unlock them at the WFMS, which may then assign them to other users.

## 7.2 The hoarding system

The hoarding system consists of four major components:

- the gathering component,
- the monitoring component
- the transfer component
- the control component of the hoarding process

The whole hoarding process is reflected in the description of the controlling component.

### 7.2.1 The gathering component

The task of this component is the querying of the external sources of information about necessary data items. This includes the data provided by applications, external data and the direct input by the user.

#### 7.2.1.1 Gathering information from external applications

Two ways are possible to query information from applications not directly associated with the currently executed business process.

One way is to examine the data generated and stored by the external application by reading its output data. To do this, the gathering component must know the format of the information and what pieces of data to search for. In addition the information what to do with the information obtained from the external application must be available to the gathering component. For example this data can be used as a selection criterion:

The calendar application of a salesperson can provide data about clients which will be visited on the next sales tour. But in order to be useful the information about the clients must be linked to data items, which are to be gathered for the user's workload, in that case the entries of the client in the company's database. The gathering component therefore needs a table, where the activities of the business processes are associated with sources for external data and descriptions of where to use the external applications' data.

The second possibility requires changes in the external applications. The

information, which data is suitable for examination by the hoarding component, can be stored within the external application. The external applications can then be queried directly, without having to scan their output, for the meta data about the data items. However the need for an association between the meta data and the location of the data still exists.

To enable the further usage of unchanged applications as tools for the activities in the business process, the approach of using the output of the external applications is more sensible. The prototype implementation of the hoarding system uses a database table to store the associations between activity, meta data and data items. That table contains an activity identifier, the external applications output file, the access mechanism to that file and the information where to apply the information obtained from the external application. So far only database entries can be described by external applications' meta data. The database tables and the selection criteria are therefore also part of the information table of the hoarding system.

#### 7.2.1.2 Querying the applications participating in the workflow

In certain cases the application supporting an activity in a workflow can identify necessary data items directly. In this case the developers of the application must provide a function, which can be called during the hoarding process and which returns the data item objects.

Such a function is then called for each application, when the activities, which should be in the workload, are known. Because these functions are part of the activities, it is sensible to include the access functionality into the WFMS. That way all information concerning activities is available via the workflow system and there is no need to gather the information from multiple sources. The hoarding system can access the function via API calls while connected to the WFMS.

But as changes to the WFMS are beyond the scope of this thesis, the prototype implementation simulates the querying. Enhanced copies of activity objects from the workflow system are used to simulate the access to application specific functions.

The applications of a workflow can of course contribute to the external data, whose access was described above. In that case however their data would be queried like all external data, which means their output must be registered in the hoarding system's information table for external data.

#### 7.2.1.3 User Input

The input of selection criteria by the user is a very important source of data for the gathering component. The system assumes, that the user has the best knowledge on necessary data items. The user is asked two times for her input. The first time is at the beginning of the hoarding process:

The user has selected the activities she wants to be part of the workload. For these activities she might want to take special data along. This can be required data which would never be part of the automatically generated workload. Another reason for specifying data items directly is the wish to ensure, that these items are transferred to the client device.

The prototype implementation allows the user to specify files directly by

selecting them in their storage location. Database entries are specified by giving the database and table names and a key value.

The second time, the user is asked for her input is at the end of the hoarding process. Then the data items, selected by the hoarding system, are presented. The user has the possibility to change that selection, before the data is transferred to the mobile device.

## 7.2.2 The monitoring component

The monitoring component is divided into two major parts. This is because the two kinds of data items - files and database entries - must be monitored in different ways. There is a monitoring component for file accesses and a monitoring component for database accesses. These two subcomponents are described in the following.

### 7.2.2.1 Monitoring file accesses

The appropriate way to monitor file accesses is the integration of the monitoring component into the file system. In that way no applications have to be modified, because the monitoring activities are hidden from the user's applications; they still use 'normal' access methods via the operating system's file system. This approach is proven to be feasible by file hoarding systems like CODA [CODA98]. For simplicity the prototype implementation uses a different approach. Applications request access to files at the monitoring component, which records the request and returns the file to the requesting application. That way only file accesses by applications aware of the monitoring component can be monitored. Independently from the mechanism of how the requests are monitored a different problem must be solved. The recording of which application accessed which data item is not enough, if this information is used to make a hoarding decision. Such a decision needs an association between data item, application and activity. The monitoring component therefore needs an identification for the workflow to be recorded with the request. Only the WFMS system can provide that information, because the single applications are unaware of the environment they are started in. The monitoring component must therefore inquire at the WFMS which workflow is associated with the activity, requesting a data item. The workflow system checks, which of the activities assigned to the user are currently running. This information is used to return the corresponding workflow.

### 7.2.2.2 Monitoring database accesses

The integration of a monitoring component into the access methods of the database has an enormous impact on the performance of the database system. Therefore the monitoring component for databases must be independent from the database system. The solution for the problem is the installation of the monitoring component parallel to the database system. Requests of applications to the database system are duplicated and sent to both, the database and the monitoring component. The same applies to the answers from the database, they are sent to the requesting application and the monitoring component.

To avoid the necessity to change applications, the duplication of requests and

answers must take place on a lower level. In the prototype implementation it is assumed, that the request and answers are sent via the TCP/IP network protocol. The monitoring component uses the port(s) the database system normally listens on for requests to intercept messages to the database system. These messages are recorded and forwarded to the new port of the database system. The answer is simply sent back to the requester, because it does not provide new information about the data items.

As with file access monitoring, the monitor component for database accesses must inquire at the WFMS which workflow the requesting application belongs to.

### 7.2.3 The transfer component

As with the monitoring of data items, the transfer of data items distinguishes between database entries and files. Depending on the kind of the data item, the items are transferred differently. Both ways are presented in this section.

#### 7.2.3.1 Transferring files

To avoid changing all applications on mobile client devices, the data items requested by those applications must be found the same way as in connected mode. The file accesses via the network in connected mode must be appropriately mapped to file accesses on the local storage of the client device. This can be achieved by using environment variables to specify the starting point of directory structures in which the requested files can be found. The transfer component copies the directory structure to the client, stores the files in the correct locations and sets the environment variable for the entry point to the directory structure according to the location chosen on the client device. In that way the applications find the correct location of the requested files on the client's local storage, even if the files are spread across the network in the stationary environment.

#### 7.2.3.2 Transferring database entries

To be able to transfer database entries, the transfer component must check the availability of a database on the client corresponding to the database used in the stationary network. Because it is assumed in this thesis, that a database system is available on the client, the checking of the availability is reduced to checking if the appropriate database structures are already present on the client. If that is the case, the database entries are replicated to the client's database. If the structures are not present on the client, the transfer component has to build them first. To do that, the stationary database must be queried for meta data on the necessary database structures. Information about the structure of database entries can be found in the database system's catalog tables [ElmasriNavathe94]. As the database and table names of the database entries are known, the structure of the tables can be extracted from the database catalog and rebuild on the client's database system. The meta data not only contains descriptions of tables, but necessary information like constraints for values, index structures or predefined keys can also be used to build the structures on the client. After the client's database has the same structure as the stationary one, it is filled with the database entries of the workload.

## 7.2.4 The controlling component

This section describes the component of the hoarding system responsible for the coordination of the hoarding process. As described in chapter 6.3.1 the different subcomponents are invoked in the following order:

1. The gathering component
  - User Input
  - Application specific functions
2. The monitoring component
3. The gathering component
  - External data
4. The results of the hoarding are shown to the user
  - User Input
5. Transfer component

The steps 1-3 are iterated for each activity in the workload. This ensures, that the first applications in the workload get their data with a higher probability. To make sure, that not only data for one activity is transferred, the hoarding system determines a maximum number of data items transferred for each application. This number is calculated using a parameter set by the system administrator, describing the average data item size. The available storage space on the client is then equally distributed among the activities of the workload.

## **8 Evaluation of the modified system**

Chapter 8 contains the results of the evaluation of the modified system. Because no real world workflow data was available, a synthetic model is used to simulate the accesses to data items. In the following the used model is described along with its statistical background. The realization of the model is briefly described as are the results of simulations using the model.

### **8.1 The simulation environment**

The modified workflow system uses two new components: the determination component to select the activities for the mobile workload and the hoarding system to provide the operational data for that workload. The prototype implementation of both was tested at the end of the work on this thesis. Because the main focus of the thesis is on identifying and transferring of operational data, the hoarding system was tested in more detail than the determination component.

#### **8.1.1 Simulation Computer**

The simulations were executed on 'aspc7' of the Institute of Parallel and Distributed High-Performance Systems (IPVR) in the department Applications of Parallel and Distributed Systems (AS). Aspc7 is a Pentium III machine running at 500 MHz with 128 MB main memory. The workflow software MQSeries Workflow v 3.2.1 from IBM is used together with IBM's Universal Database DB2v5.2 and MQSeries v5.1 as messaging middleware. The implementation of the simulation software is written in JAVA using Sun Microsystem's Java SDK 2. The underlying operating system is Microsoft Windows NT 4.0 Workstation. The simulation component itself does not need a special environment besides a Java Runtime for Java 1.1.

#### **8.1.2 Test of the determination component**

IBM's MQSeries Workflow v 3.2.1 was used as workflow system during the test. To test the functionality of the determination component, the workflow examples delivered with the product were used. The execution of the determination program in different states of the workflows principally provided the expected activity streams according to the rules described in chapter 5. During testing it became clear, that the approach of selecting possible workflows for a user by just checking her access rights to the workflows is too simple. For example a user defined as replacement for another user has automatically access to the other user's workflows, if she is available or not. The same applies to managers, workflow designers and administrators. They all find workflows in their selection list, which were not intended to be executed by them.

The solution to that problem is to execute staff queries for the activities during their checks of suitability for mobile execution. The staff queries only return users, qualifying for the execution of the activities at the moment of the query. Unfortunately the API of MQSeries Workflow [MQWProg99] provides no access

to the staff query operations of the workflow system. Therefore this advanced approach could not be implemented.

### 8.1.3 Test of the hoarding system

To test the hoarding system, a different approach was necessary. The definition of the workflows in the examples provided with MQSeries Workflow is not sufficient for a simulation of the hoarding system. Because the hoarding system's success is related to the actual accesses to data items by the applications in the workflow, those accesses must be simulated during the test. The best way to simulate the accesses would have been the replay of recorded executions of real world workflows. In that case not only the data accesses, but also the decision for different paths in the workflows would have been realistically distributed.

The lack of actual data about workflow execution made it necessary to simulate the accesses synthetically. A simulator program was developed, which implements the following model for data accesses.

#### 8.1.3.1 Model for simulated accesses to local storage

In a model for access simulations two factors must be determined statistically. One is the frequency of accesses, the second is which data is accessed. The first is normally determined by a random selection, which follows a Poisson distribution. For the determination of which data items to access during simulation a Zipf distribution is used.

Both distributions can briefly be described as follows:

##### **Poisson** distribution:

The Poisson distribution represents the number of occurrences, per unit time, of an event that can occur at any instant of time. For example the number of alpha particles emitted by a radioactive substance in a single second has a Poisson distribution [Knuth97].

The probability function for the Poisson distribution is defined as:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where  $\lambda$  denotes the mean of the distribution.

##### **Zipf** distribution:

This distribution follows *Zipf's Law*, which originally was put up to describe the frequency of words in common texts. Zipf observed, that if the most frequent word occurs  $c$  times in a text, the number of occurrences of the second most frequent word occurs  $c/2$  times and the  $n$ 'th most frequent word occurs  $c/n$  times.

$$p_1 = c, p_2 = \frac{c}{2}, \dots, p_n = \frac{c}{n}$$

### 8.1.3.2 Applicability of the distributions

The execution of a workflow is a process, finished in a certain period of time. In that period various access operations to data take place. The time intervals between accesses are exponentially distributed. But because it is not the time intervals between accesses, but the total number of accesses in a time period, which is important, the Poisson distribution is used. It describes the number of events taking place in a certain time period with exponentially distributed intervals of time between the single events.

To generate the accesses to data items according to a Zipf distribution is commonly used to model requests to a finite amount of information units. For example web requests from a fixed user community to web caches are distributed according to an only slightly modified Zipf's Law [BreslauCao99]. As web caches store those units of information request most frequently during a certain amount of time, the hoarding system, proposed in this thesis, transfers the most frequently used data items to the local storage of a mobile client. In both cases the data is ordered by the access frequency. It is therefore feasible to assume the model for web caches in the hoarding environment too.

### 8.1.4 Realization of the model

The realization of the model presented above consists of the random data items generator, a table representing the global storage and a table representing the local storage.

#### 8.1.4.1 The generator

The first task of the generator is to calculate the number of requests issued during the work on the workflow. The following algorithm [Knuth97] randomly creates an integer  $N$ , which is interpreted as the number of requests and which follows a Poisson distribution with mean  $\mu$ :

- Calculate  $e^{-\mu}$
- Generate one or more uniform deviates  $U_1, U_2, \dots, U_m$ , i.e.  $m$  random floating point numbers between 0 and 1 until  $U_1 * U_2 * \dots * U_m \leq e^{-\mu}$
- $N$  is set to  $m-1$

For each of the  $N$  request the generator must determine which data item is requested. To simulate a behavior following Zipf's Law, the data item is selected as follows:

- Generate a random floating point number  $u$  between 0 and 1
- Subdivide the interval  $[0,1]$  into continuously decreasing subintervals, the first subinterval ranges from 0 to  $c$ , the probability of access to the most frequently accessed data item. The second interval ends at  $c+c/2$ . The  $i$ 'th interval

starts at  $\sum_{n=1}^i \frac{c}{n}$  and ends at  $\sum_{n=1}^{i+1} \frac{c}{n}$ . Note, that  $c$  must be less or equal than  $\frac{1}{2}$ .

- Check if  $u$  is less than the upper bound of the subintervals, starting with the first. The first subinterval whose upper bound is greater than  $u$  defines the data item.

This algorithm determines a random number with the probability stated by Zipf's Law. Because  $u$  is distributed equally over the interval  $[0,1]$ , the probability of it falling into the first subinterval is  $c$ . Because the second interval is only half as broad as the first, the probability is only  $c/2$  as demanded by Zipf's Law. The numbers returned by the generator describe the position in the tables described in the following. If a returned number is larger than the upper bound of the table representing the local storage, it implies a hoard miss. The requested item is then searched in the greater table standing for the global store. If it even exceeds that table, it is interpreted as an error, because no request beyond the global storage are possible.

#### 8.1.4.2 The tables

The tables are generated and maintained by the hoarding system. For simulation purposes a predefined table is used as a starting point. This corresponds to the possible user inputs made at the beginning of the hoarding system's usage. The predefined table is considered to be the global storage. It contains the data items and their corresponding counters. When a hoarding request is received, the hoarding system first discards all entries in the table, which are not part of the workload. The remaining table is still ordered by the counter values of the data items. Depending on the storage space on the client device a varying number of data items is copied to the table representing the local storage. The generated requests for data items are checked against this table, to distinguish hoard hits, from hoard misses.

#### 8.1.5 Expectations on the system's behavior

The simulation of the system should provide results comparable to those found in [BreslauCao99]. There the same distribution, as in this thesis, is used to

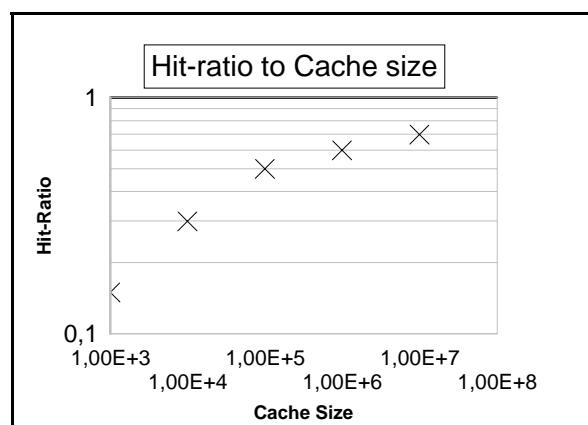


Figure 8.1: Hit-ratio as a function of cache size

determine which documents are requested from a web cache. Because a web cache holds a finite number of the most popular documents requested it works like the local storage of the hoarding system. Therefore the results should be comparable.

Increasing the size of the local storage should lead to a better hit ratio. It should rise fast at the beginning and flatten after a certain point. Figure 8.1 taken from [BreslauCao99] shows this. This is due to the fact, that requests for data items become increasingly improbable with their rank in the global storage. As stated above the probability for the  $n$ 'th item is only  $p_n = \frac{c}{n}$ . If the hoard size is increased, the probability that the first item not in hoard is accessed decreases rapidly.

With constant cache size the hit ratio should increase slightly from execution to execution. Occurring hoard misses lead to a rise of the corresponding data item in the ranking of the global storage. Therefore the probability of another access to the same data item also rises, according to Zipf's Law. But this has only a significant impact, if the data item does not only rise to the end of the local storage's ranking, but further towards the top. The number of activities might also play a role here. If several activities have equally frequent accesses to their favorite data items, all the favorites fill the local storage leaving no space for less frequently accessed data items.

## 8.2 Results of the Simulation

### 8.2.1 Simulation Scenarios

The basis of the simulation builds a table representing the global storage. This table contains 10000 randomly generated data items of 6 activities. After each simulation run, the global storage is updated and resorted. That way the next simulation run can directly use the results of the previous one.

The following scenarios were used during simulation:

1. Local Storage with size of 100 data items:  
The size of the local storage is set to 100 data items. All 6 activities are part of the workload and can request data items. The mean of the Poisson distribution is set to 7 and the probability to access the most popular data item of each activity is set to  $\frac{1}{4}$ .
2. Local Storage with size of 500:  
Only the size of the local storage is changed to 500 in comparison to scenario 1.
3. Local Storage with 500 data items and 4 requesting applications  
Instead of 6 only 4 activities need data in this scenario.

## 8.2.2 Result Graphs and Interpretation

### 8.2.2.1 Scenario 1

The following graph shows the course of the simulation of scenario 1:

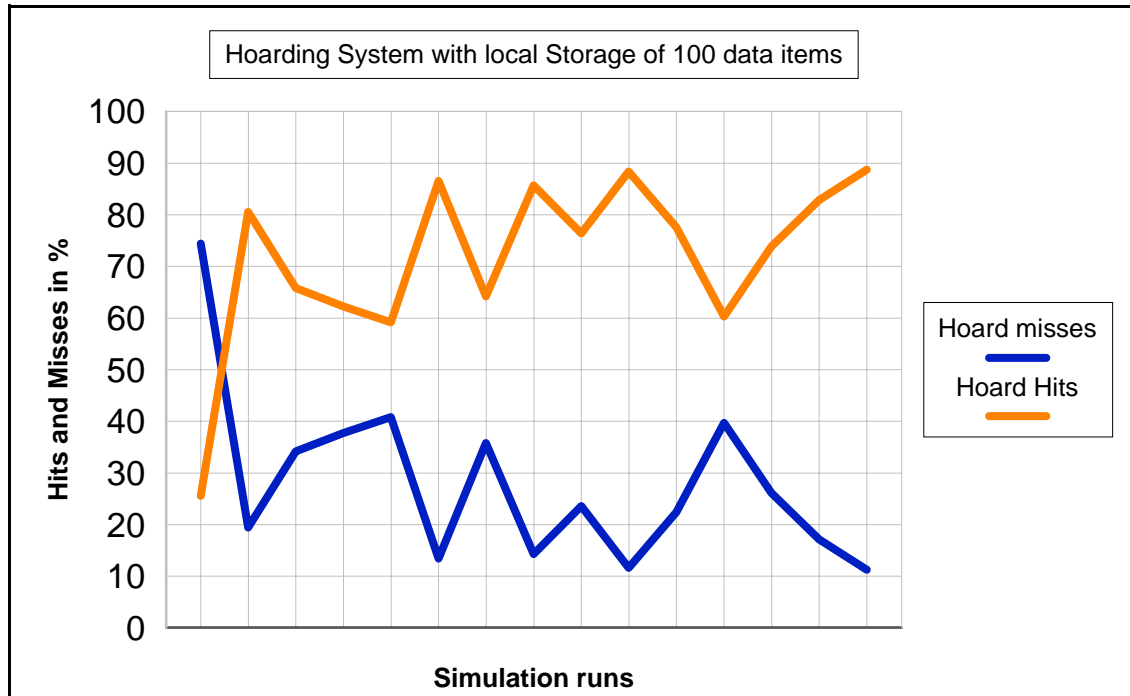


Figure 8.2: Results of scenario 1

In Figure 8.2 the tendency to better hit-ratio can be seen. Even if an execution generates more hoard misses than its predecessor, the system is able to improve its hoarding behavior in the following executions.

### 8.2.2.2 Scenario 2

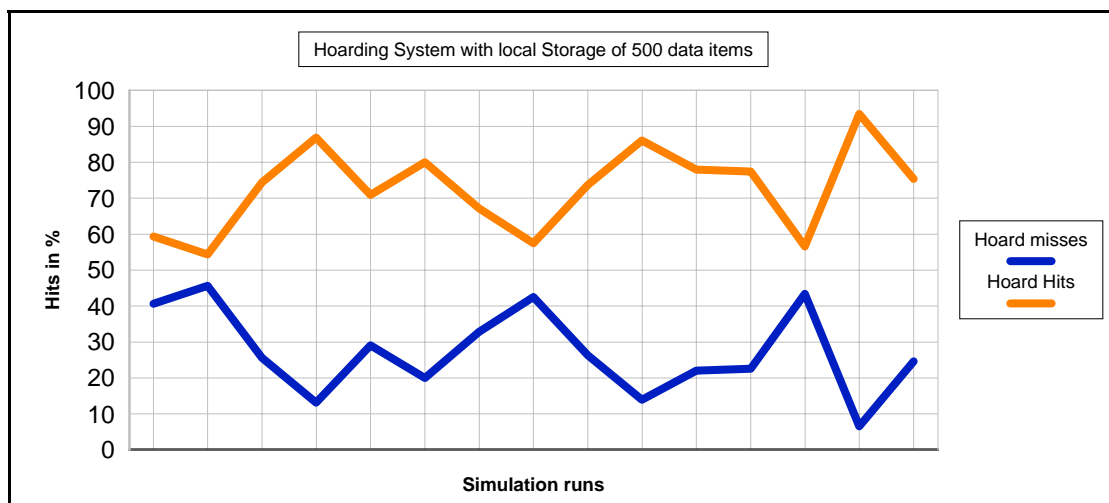


Figure 8.3: Results of scenario 2

Figure 8.3 shows the results of the simulation with 500 data items in the local storage. From the beginning the hoard misses are fewer than with a local storage of 100 data items. As with 100 data items, the occurrence of rising amounts of hoard misses is answered by the hoarding system and the hoard misses become fewer in following executions.

### 8.2.2.3 Scenario 3

As expected, are the hoard misses in an environment with 500 data items and 4 requesting applications even less frequent than in scenario 2. The two peaks in figure 8.4 can be interpreted as seldom occurring special executions of the

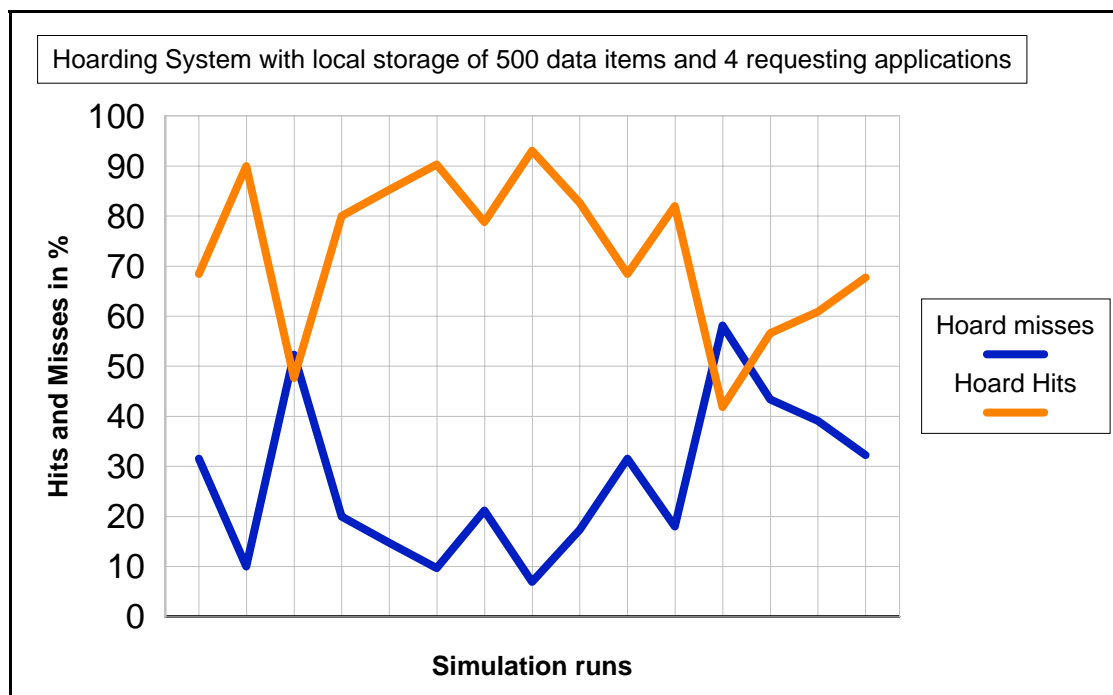


Figure 8.4: Results of Scenario 3

workflow.

### 8.2.3 Résumé

Due to lack of real world data the modified system had to be tested by synthetical means. The time necessary to build the simulation environment enforced, that only simple test runs could be executed. Therefore it would be presumptuous to claim, that the results of the tests prove the suitability of the prototype implementation. Tests over a longer period of time, with far more executions would be necessary to prove that. However, the tendency of the system to try to correct workloads leading to rising numbers of hoard misses could be seen. The results do not contradict the expectations, but are to irrelevant from a statistical point of view, to prove the expectations. A more precise model with real world data seems more suitable to evaluate the hoarding system.

## 9 Summary

The objective of this thesis was the investigation of the possibility to enhance workflow management with mobile clients. Support for mobile clients enables a workflow system to control a broader range of business processes. Mobile work becomes more efficient and secure, due to streamlining and standardizing effects of the workflow system. Therefore both technologies profit from a cooperation. Despite the importance of the topic amazingly few research has been done in this area. A few reports like for example [Exotica95] or [Bussler95] discuss the issue of disconnected client support. Most work on distributed workflow management architectures is pointing in the direction of a migrating workflow system. Workflow control and status of currently running workflows is distributed in those systems. Transfer of control and activities is done via reliable messaging. If processing nodes in such a system become unavailable, disconnection only results in a delay of messages to be send. However these systems lack one important ability of centralized workflow management systems: They can not offer a task to multiple users. This makes the approach not feasible for production workflows. There a great part of the gain in efficiency results from the assignment of pending works to multiple users.

The lack of preceding work made a general approach to the topic necessary. Typical needs of workflow management and mobile computing were determined, to get a general idea on the requirements of a possible integration of mobile devices into workflow systems. It became clear, that mobile computing always demands flexibility and independence from centralized server processes. Mobile execution of workflows is only sensible, if the clients do not rely on their servers. They must be able to work, using their own resources, during disconnection.

Before the necessary changes to the workflow environment were evaluated, the feasibility of **different client devices** was checked. Because today's laptop computers have comparable resources as computers used as desktop clients, they can be used for mobile workflow execution without any doubt. The use of handheld devices like PDAs is already possible in some cases. But special activities can not be executed with such devices. Because of lacking storage space, limited computing power and the small display area. Smaller devices than handheld computers offer even less resources and are therefore not suitable for workflow execution.

The analysis of the needs of mobile clients in workflow environments resulted in two main aspects. One was, how the activities can be determined, which can be executed while disconnected? The other aspect was, what is necessary to execute activities on a mobile device?

The answer to the first question was the development of the **workload determination algorithm**. It ensures, that the selected activities can be executed from the workflow management's view. That means, that the part of the workflow can be navigated on the client and that the input parameters for each application are known and available. The algorithm does not guarantee,

that all data required during execution is available.

This issue was addressed in the proposed solution to question two. It became clear, that the workflow system must select data which should be transferred to the mobile clients. No client device can store the whole amount of information available in the computer system of an enterprise. This conclusion led to the next question. How can important data be distinguished from data not required for the client?

For the task of selecting, gathering and transferring of data the **hoarding system** was designed. It uses different sources of information to get meta data on the data items required for mobile execution of workflows.

It is described in the thesis how the requirements have an impact on the system architecture. Problematic aspects of the integration of components were stated and alternative possibilities discussed.

The new parts of the architecture described in the analysis part of the thesis were realized in a prototype implementation. To ensure the usability of those components in heterogeneous workflow environments, the implementation was programmed in Java.

Finally the proposed mechanisms were simulated using a simple synthetic model of the environment.

The topic of mobile computing and workflow management of course exceeds the extent of a master thesis. The following items could be the focus of further research:

Directly connected to this thesis is the question of the applicability of the model, taken from web caching analysis, to workflow environments. The answer to that question can only be found by intensive observing the accesses to data items in real world workflows. Even if the model is principally sensible for workflow environments, it should be made more precise based on realistic data.

In this thesis only theoretical proposals for dealing with concurrent updates were stated. The adaptation of existing techniques of concurrency control to mobile workflow execution should be also investigated. It might be, that new techniques are more suitable, than those used in distributed database systems.

A specific problem of workflows is their long duration. This enforces the possibility to compensate steps of work. A suitable concept for compensation must be developed for mobile execution. What happens if compensation is necessary in mobile workloads? Must compensation activities and their operational data be prepared with the workload?

Another topic coming from mobile computing could be the integration of spatial-awareness into the hoarding decisions. Route planing tools could provide data for future locations of the mobile client. This information can then be used to select data for future use already prior to disconnection. Such data could also be used to provide infostations with data, which mobile users require in those specific areas.

Advances in communication technology enabling higher bandwidth and more secure wireless connections will someday allow mobile clients to act like stationary ones. But in the meantime the fast growing number of mobile workers must become part of workflow solutions.

Based on this thesis the execution of workflows using mobile devices seems feasible. The modified system could build the foundation of a solution, which combines the benefits of mobile computing and those of workflow management.

## 10 Bibliography

- [BadrinathPhatak98] B.R. Badrinath, S. Phatak: *Database Server Organization for Handling Mobile Clients*, Department of Computer Science, Rutgers University New Brunswick, 1998
- [Baumann98] Joachim Baumann: *Skript Rechnernetze II SS98*. Lehrstuhl 'Verteilte Systeme', Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, 1998
- [BernsteinNewcomer97] P.A. Bernstein, E. Newcomer: *Principles of Transaction Processing*. Morgan Kaufmann Publishers, Inc., 1997
- [BreslauCao99] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker: *Web Caching and Zipf-like Distributions: Evidence and Implications*, Proceedings of the IEEE INFOCOM'99, New York City, NY, 1999
- [Bussler95] C. Bussler, *User Mobility in Workflow-Management Systems*, in Proceedings of the Telecommunications Information Networking Conference (TINA '95), Melbourne, Australia, February 1995
- [CODA90] M. Satyanarayanan, J.J. Kistler, P. Kumar, M. Okasaki, E. Siegel, D. Steere: *Coda: A highly available file system for a distributed workstation environment*, IEEE Transaction on Computers, 39(4):447-459, 1990
- [CODA92] J.J. Kistler, M. Satyanarayanan, *Disconnected Operation in the Coda File System*. ACM Transactions on Computer Systems, Feb. 1992, Vol. 10, No. 1, pp. 3-25
- [CODA98] P.J. Braam, *The Coda Distributed File System*. Linux Journal #50, June 1998
- [ElmasriNavathe94] R. Elmasri, S. Navathe: *Fundamentals of Database Systems second edition*. The Benjamin/Cummings Publishing Company, Inc., 1994
- [Exotica95] G. Alonso, R. Günthör et al. : *Exotical FMDC: Handling Disconnected Clients in a Workflow Management System*. Proc. 3rd Int'l Conference on Cooperative Information Systems, Vienna, May 1995
- [Gartner97] *Wireless Data in the United States: Pieces of the Puzzle are Missing, but a Picture is Taking Shape*, The Gartner Group, The Dataquest Market Analysis Perspective, 1997

- [Goldmann97] D. Goodmann, J. Borràs, N. Mandayam and R. Yates: *Infostations: A new system model for data and messaging services*. In Proceedings of the 47th Annual IEEE Vehicular Technology Conference (VTC '97), pages 969-973, Phoenix, AZ USA 1997
- [GrayReuter93] J. Gray, A. Reuter: *Transaction Processing Systems, Concepts and Techniques*, Morgan Kaufmann Publishers, San Mateo, California 1993
- [HammerChampy93] M. Hammer, J. Champy: *Reengineering the Corporation*. New York: Harper Business, 1993
- [IBMTPSpec00] International Business Machines Corp.: *Technical specification for ThinkPads<sup>1</sup> 240 and 600X*, 2000
- [ImielinskiBadrinath94] T. Imielinski, B.R. Badrinath: *Mobile Wireless Computing: Challenges in Data Management*.
- [KatzWeiss84] R. Katz, S. Weiss, *Design Transaction Management*, Proceedings of the 21st Design Automation Conference, 1984
- [Knuth97] D. Knuth: *The Art of Computer Programming (Vol. 2): Seminumerical Algorithms* 3rd Edition, Addison Wesley Publishing Company, October 1997
- [Leymann95] F. Leymann: *Transaction Concepts for Workflow Management Systems*. In G. Vossen and J. Becker, editors, *Geschäftsprozeßmodellierung und Workflow-Management*, International Thompson Publishing, 1995
- [LeymannRoller97] F. Leymann, D. Roller: *Workflow-based applications in IBM Systems Journal* Vol. 36, No. 1 - Application Development
- [LeymannRoller99] F. Leymann, D. Roller: *Production Workflows*. Prentice Hall Inc., 1999
- [MQseries99] International Business Machines Corp.: *IBM MQseries Workflow Concepts and Architecture*, Second Edition, June 1999
- [MQWProg99] International Business Machines Corp.: *IBM MQseries Workflow Programming Guide Version 3.2*, Fourth edition, June 1999
- [Nexus99] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, M. Schwehm: *Nexus- An Open Global Infrastructure for Spatial-Aware Applications*, Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, 1999

---

<sup>1</sup> ThinkPad is a trademark or registered trademark of International Business Machines Corp.

- [PalmVIISpec99] Palm Computing Inc.: *Palm VII<sup>2</sup> Organizer Specification*, 1999
- [Satyanarayanan96] M. Satyanarayanan: *Fundamental Challenges in Mobile Computing* Fifteenth ACM Symposium on Principles of Distributed Computing May 1996, Philadelphia, PA
- [Strategy99] P. Kendall: *Wireless Data Devices: Product Trends & Market Forecasts*, March 1999, <http://www.strategyanalytics.com>
- [Tanenbaum96] A.S. Tanenbaum: *Computer Networks* Third edition. Prentice Hall Inc., 1996
- [WAPSpec99] WAP Forum: *Wireless Application Protocol Specification* 1999, <http://www.wapforum.org>
- [WfMC99] Workflow Management Coalition, *The Workflow Management Coalition Terminology & Glossary*, Document Number WFMC-TC-1011, revised Issue 3.0 Feb. 1999
- [WhiteFisher94] T.E. White, L. Fischer (ed.): *The Workflow Paradigm: New Tools for New Times*. Future Strategy Inc., Book Division. 1994

---

<sup>2</sup> Palm VII is a trademark of Palm Computing Inc.

Ich versichere hiermit, dass ich diese Arbeit selbständig verfaßt  
und bei der Erstellung nur die angegebenen Hilfsmittel benutzt  
habe.

---

Markus Reichart