Proc. Workshop Informatik '99, P. Dadam, M. Reichert (eds.): Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, Paderborn, Germany, October 1999
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-24/

# Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows

Thomas Bauer, Peter Dadam
University of Ulm, Dept. of Databases and Information Systems
{bauer, dadam}@informatik.uni-ulm.de, http://www.informatik.uni-ulm.de/dbis

## Abstract

In large workflow management systems (WfMS), it is particularly important to control workflows (WF) in an efficient manner. A very critical factor within this context is the resulting communication overhead. For this reason we have developed an approach for distributed WF control, which tries to keep the communication overhead low. In this paper, this approach is described and examined by means of a simulation.

## 1 Introduction

Enterprise-wide and cross-enterprise WF scenarios are characterized by a large number of users and many concurrently active WF instances. Therefore, the WF servers have to cope with a high load in total. Furthermore, in such an environment, the different organizational units (OU) are often far away from each other and connected by slow wide area networks (WAN). For this reason, the load of the communication system is an extremely critical aspect. Because of the resulting communication overhead a centralized WF control is often not applicable (at least not at reasonable costs). Another reason is that the WF systems used are often very heterogeneous which makes a centralized WF control rather complicated if not even impossible. In the ADEPT project[1] we, therefore, have developed an approach for distributed WF control which addresses these issues.

In the next section, some approaches for distributed WF management are presented and the distribution model of ADEPT is described. In Section 3 the different distribution models are compared by means of a simulation. Section 4 discusses related work and Section 5 concludes with a summary and an outlook on future work.

## 2 Distribution Models

In this section, different approaches for distributed WF management are presented and an appropriate model for enterprise-wide and cross-enterprise usage is developed.

[1] ADEPT stands for Application Development Based on Encapsulated Pre-Modeled Process Templates.

## 2.1 Distribution of Entire Workflows

The simplest approach for distributed WF management is to control a WF instance always completely by one WF server. This can be done, e.g., by distributing the WF control to WF servers by WF type. That is, whenever a WF instance of type $x$ is started, it will be completely controlled by the WF server responsible for workflows of this type. By doing so, the total load is divided up among the WF servers. This approach works well if almost all actors performing the activities of one WF belong to the same OU.

## 2.2 Partitioning of Workflows

If the actors of the different activities of a WF are geographically located far away from each other, the distribution model described above generates high communication costs. The reason is that all activities of a WF instance are controlled by the same server and, therefore, have to communicate with this server (even if this server is far away). In such cases it would be favorable, if for each activity a WF server which is closely located (at best in the same subnet) to the actor of the activity could be used.

To achieve this goal, the WF is partitioned and each *partition* is assigned to that WF server which is located next to the potential actors of the activities belonging to it (c.f. Fig. 1). At run-time, when the control moves from one partition to the next, a *migration* becomes necessary: The current state of the WF instance (WF control data and the values of the data elements[2]) is transferred to the WF server of the subsequent partition. Subsequently, this WF server takes over the control of the WF instance. Migrations are not for free, however. They also cause communication costs and contribute to the total load of the WF servers. In ADEPT, therefore, migrations are used only if they are profitable in the sense that they improve the total communication behavior. For example, in most cases it does not make much sense to migrate a WF instance to another server and back again just for the execution a single activity which is performed by an actor of another OU. These two migrations cause higher costs than to control the activity by an unfavorable server. For a WF designer, it is difficult to decide which are the most appropriate server assignments for the activities. For this reason, ADEPT supports the WF designer by sophisticated build-time components, which calculate those server assignments which will lead to a minimization of the total communication costs at run-time (see [BD97]).

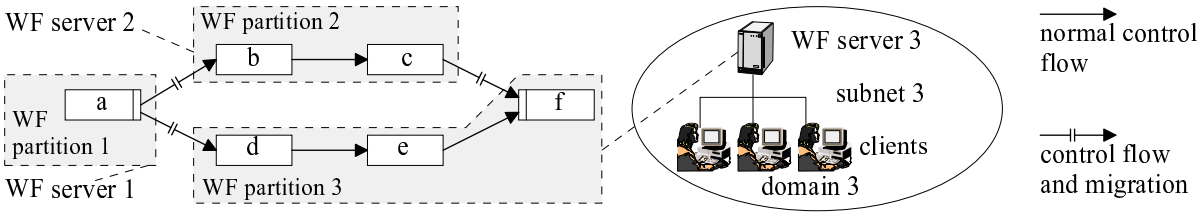


**Figure 1** Partitioning of a WF schema and distributed WF execution.

[2] The WF schema information itself is completely replicated at all WF servers. Therefore, only the relevant instance information has to be transferred to the target server. Opposed to the WF schema, WF instance information is not replicated or stored centrally in order to avoid synchronization overhead or (long distance) communication during the "normal" execution of WF activities (i.e., execution without migration).

## 2.3 Variable Server Assignments

The approach described in Section 2.2 is reaching its limiting factors if a WF contains *dependent actor assignments* (c.f. activities 2, 3, and 4 in Fig. 2). Consider the case where a hospital has several wards of the same type and where a patient is (more or less arbitrarily) transferred to one of these wards after his admission in the emergency room. The subsequent activities 2, 3, and 4 (c.f. Fig. 2) will then only be performed by the medical staff of this ward. At build-time, however, it is not known which actor and thus which ward will be selected in activity 1 (at best, probability considerations are possible). This means, in turn, that no suitable (static) server assignments for this WF can be determined.

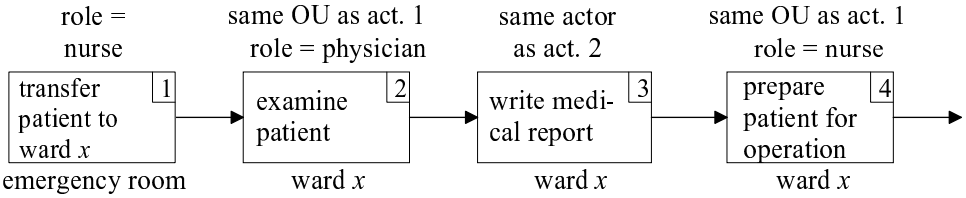


**Figure 2** Example of a WF with dependent actor assignments.

In principle, problems of this kind could be solved by determining the server assignments fully dynamically at run-time. I.e., after the completion of each activity a computation to determine the most appropriate WF server for the subsequent activity or activities takes place. In doing so, one would (basically) achieve the optimal distribution, since at this point in time most information is available for this decision. Unfortunately, this approach is not feasible because of performance reasons, in general. At run-time the WF servers have already to manage a high workload and should therefore not be burdened with additional (and partially rather complex) computations to determine the optimal distribution.

*Variable server assignments* [BD98b, BD99a] are a compromise between static server assignment at build-time and dynamic server selection at run-time: At build-time, a logical server assignment expression is determined which only has to be evaluated at run-time. In the example shown in Fig. 2, the server assignments for the activities 2, 3, and 4 result as: "server in the domain[3] of the actor performing activity 1". After the completion of activity 1, its actor and because of that also the OU (i.e. ward) for the actors who shall perform the activities 2, 3, and 4 are known. Therefore, at this point in time, the WF server of the right OU can always be chosen to control these activities. This means that information is used which was not existing at build-time and also not at the point in time this WF instance was started. Therefore, this approach is more powerful than mere static server assignments.

In ADEPT the following server assignment expressions are used at present:

1. $ServAss_k =$ "$S_i$"
   Server $S_i$ is statically assigned to activity $k$ (c.f. Section 2.2).
2. $ServAss_k =$ "$Server(x)$"
   The activity $k$ shall be controlled by the same server as activity $x$.
3. $ServAss_k =$ "$Domain(Actor(x))$"
   Activity $k$ is assigned to the server which is located in the domain of the user who has executed activity $x$.

[3] A domain is a subnet together with the corresponding WF server and clients.

4. $ServAss_k =$ "$f(Server(x))$" or $ServAss_k =$ "$f(Domain(Actor(x)))$"
   A function $f$ can be applied to server assignments of type 2 and 3.
5. $ServAss_k =$ any given expression, which does not correspond to type 1-4
   The WF designer may specify own server assignment expressions.

The ADEPT WfMS is supporting the WF designer in determining the most appropriate server assignment expression. At build-time, on request, the optimal (variable) server expressions of type 1 - 4 for this type of WF are computed. At run-time of a WF instance only these expressions have to be evaluated which can be done very efficiently. Therefore the load of the WF servers is only negligible higher as with static server assignments, but the communication costs are significantly reduced. Because of lack of space, it is not possible to describe the computation of the optimal server assignment expressions in this paper. The interested reader is referred to [BD98b, BD99a].

# 3 Evaluation

Dependent actor assignments occur in many application domains. Take the processing of a loan request at a bank, for example. Many steps of the examination of this request take place at the branch office of the customer. To compare the distribution models described in Section 2, we use a (simplified) clinical WF. The comparison is based on a simulation (for details and other simulations see [BD99b]). The actors belong to 7 OU. To each OU belongs one subnet (and one WF server in the distributed cases). The WF consists of a sequence of the following activities:
3 activities in the emergency room
1 activity to be performed by a ward physician (he transfers the patient to his ward 1-5)
5 activities to be performed by a physician of this ward $x$
1 activity to be performed in the laboratory
5 activities to be performed by a physician of ward $x$

The simulation presented subsequently is only used to compare the different distribution models. It was not the intention to detect overload situations. For this reason, even for the central case, it is assumed that the WF server is not overloaded. The data produced by the simulation are used to calculate the total load of the WF servers, the load per WF server, the load per subnet, and the load of the gateways. Fig. 3 shows a graphical representation of the result. The values are normalized in a way that the load of the central case results as 100.

## 3.1 Central WF Server

The load of the single WF server is defined as 100. The same applies to the average load per subnet and to the load of the gateways. The central WF server is located in exactly one subnet. This means that this subnet is burdened with the whole communication of this central WF server. Therefore, it represents a potential bottleneck. This irregular distribution of the load leads to the high load 387.1 of this subnet, whereas the other subnets are only burdened with 52.1 on the average. (That is, we have (as defined) an average load of 100 over all subnets, but have a peak of 387.1 in the subnet of the server.)

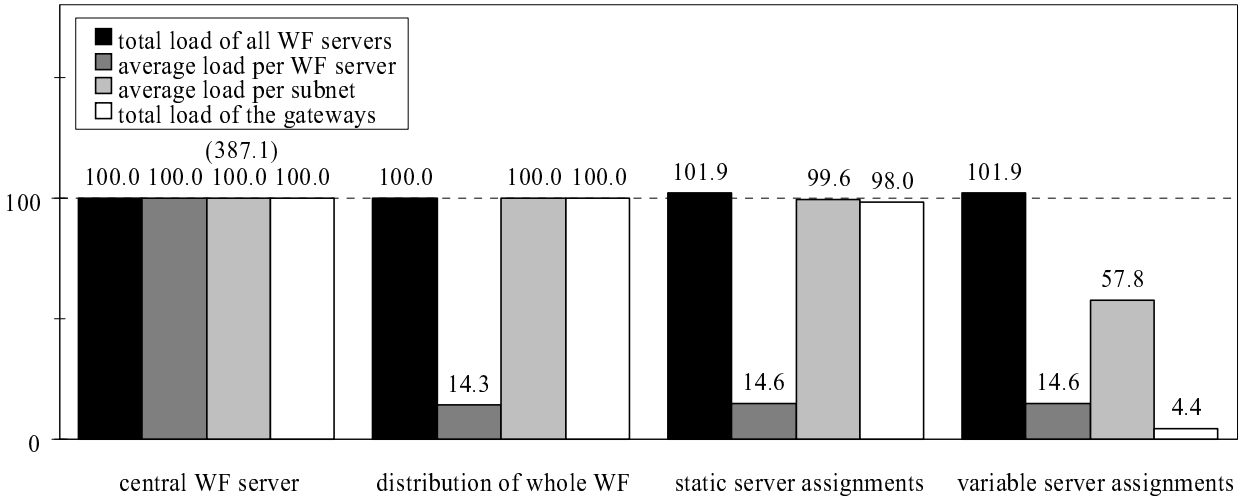


**Figure 3** Results of the simulation of a clinical WF for different distribution models.

### 3.2 Distribution of Entire Workflows

If always entire WF are assigned to the WF servers, the total load of the WF servers is the same as in the central case because there are no migrations as well. By doing so, this load is divided up among the 7 WF servers of the WfMS, in principle[4]. Therefore, the load per WF server is 14.3, which is the best value of all distribution models. The load of the subnets and the load of the gateways are identical to the central case because in both cases the server of ward 1 is used. The bottleneck in the subnet of the WF server, however, does not exist any more, since different WF types may be controlled by different WF servers.

### 3.3 Static Server Assignments

If the WF is partitioned, migrations become necessary (e.g., there is a migration from the WF server of the emergency room to the WF server of the selected ward). This results in a higher total load of the WF servers. This load, however, is shared among several WF servers. The load of the subnets and the load of the gateways are reduced because appropriate WF servers are chosen for the partitions. Since static server assignments are hardly suitable for the WF considered, the improvements are very small.

### 3.4 Variable Server Assignments

Variable server assignments allow to control activities performed by a physician of ward $x$ by the WF server of this ward. Because of this, the load of the subnets and the load of the gateways can be reduced significantly. The load of the subnets is almost halved (57.8). Nearly all the communication between WF server and client can now be handled within one subnet instead of two subnets (the subnets of the

[4] In this simulation, however, one WF server was burdened with the whole load, because we have simulated the execution of instances of only one WF type.

WF server and the client). Gateway communications occur very seldom (4.4) because variable server assignments always allow to select the WF server in the appropriate ward.

A goal of ADEPT is to reduce the communication load. The simulation has shown that this can be achieved by the use of variable server assignments. A disadvantage is that the total load of the WF servers increases due to the migrations. The use of additional WF servers, however, can compensate this effect.

# 4 Related Work

This section gives an overview of different approaches for distributed WF management. Due to lack of space, the concepts beyond these approaches are only briefly mentioned. A more detailed discussion can be found in [BD98a, BD98b, BD99b]. Some research prototypes (e.g., Panta Rhei [EG96], WASA [WHKS98]), which do not primarily consider scalability issues, and most of the commercial WfMS use a central WF server. Another extreme is a completely distributed system (Exotica/FMQM [AMG+95], INCAS [BMR96]). The machine of the user currently performing an activity also controls the WF instance. Therefore, there is no need for any WF server.

There are several multi-server approaches: In METUFlow [Dog97] the WF instances are controlled in a distributed manner. It is not discussed, however, how the location for a WF server is chosen. The Exotica/Cluster approach [AKA+94] and MOBILE [HS96] assign WF servers to whole WF instances. In addition, MOBILE enables remote WF servers to control subprocesses [SNS99]. MENTOR [MWW+98], WIDE [CGS97], CodAlf, BPAFrame (both [SM96]) and METEOR$_2$ [DKM+97] use WF partitioning. The partitions are statically assigned to WF servers. In addition, CodAlf and BPAFrame use a trader to select one of the assigned WF servers at run-time. The TEAM Model [Pic98] discusses the treatment of cooperations between autonomous enterprises. Activities are (statically) assigned to the WF server of the corresponding enterprise. To our best knowledge, ADEPT is the only approach that uses variable server assignment expressions.

# 5 Summary and Outlook

In order to avoid overloading of the WF servers and of the communication network, distributed WF control in enterprise-wide application scenarios is indispensable. This requires partitioning of WF schemas and migrations. The communication behavior can be further improved if variable server assignment expressions are used. These expressions can be determined at build-time, allow the selection of a suitable WF server to keep most of the communication local within the same subnet, and require almost no additional effort at run-time.

In cross-enterprise WF applications [Pic98], the activities shall be controlled by the WF server of the enterprise the activities “belong” to, in most cases. For this reason, if different activities of the same WF are performed by actors of different enterprises, then partitioning of WF and migrations are required. If an activity may be performed by actors of more than one enterprise, then static server assignments are not adequate. With variable server assignments, it is possible to assign such an activity to a WF server depending on WF instance data; i.e., the WF server may belong to different enterprises. The decision – which enterprise performs and controls a specific activity – may depend on previous activities (e.g., the actor of the activity “select job”) or it may depend on WF data (e.g., the data element “contractor”).

In addition to scalability and performance aspects, the treatment of heterogeneity is important for the support of enterprise-wide and cross-enterprise WfMS. This problem must be solved by defining appropriate standards for the interoperability of (heterogeneous) WfMS. What kind of functionality should be offered in order to adequately support such scenarios under communication aspects has been discussed in this paper. Certainly, a lot of other aspects, like transaction support [Ley97] to guarantee the robustness of the WfMS or the possibility to adapt running WF instances [RD98, RBD99] to deal with exceptional cases are also very important.

**Acknowledgements:** We would like to thank our colleagues Manfred Reichert and Clemens Hensinger for their valuable suggestions.

# References

[AKA+94] G. Alonso, M. Kamath, D. Agrawal, A. El Abbadi, R. Günthör, and C. Mohan. Failure Handling in Large Scale Workflow Management Systems. Technical Report RJ9913, IBM Almaden Research Center, 1994.

[AMG+95] G. Alonso, C. Mohan, R. Günthör, D. Agrawal, A. El Abbadi, and M. Kamath. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *Proc of the IFIP Working Conf. on Information Systems for Decentralized Organisations*, Trondheim, 1995.

[BD97] T. Bauer and P. Dadam. A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. In *2nd IFCIS Conf. on Cooperative Information Systems*, pages 99–108, Kiawah Island, SC, 1997.

[BD98a] T. Bauer and P. Dadam. Architekturen für skalierbare Workflow-Management-Systeme – Klassifikation und Analyse. Ulmer Informatik-Berichte 98-02, Universität Ulm, 1998.

[BD98b] T. Bauer and P. Dadam. Variable Migration von Workflows in ADEPT. Ulmer Informatik-Berichte 98-09, Universität Ulm, 1998.

[BD99a] T. Bauer and P. Dadam. Variable Migration von Workflows und komplexe Bearbeiterzuordnungen in ADEPT. Ulmer Informatik-Berichte, Universität Ulm, 1999. (to appear).

[BD99b] T. Bauer and P. Dadam. Verteilungsmodelle für Workflow-Management-Systeme – Klassifikation und Simulation. Ulmer Informatik-Berichte 99-02, Universität Ulm, 1999.

[BMR96] D. Barbará, S. Mehrotra, and M. Rusinkiewicz. INCAs: Managing Dynamic Workflows in Distributed Environments. *Journal of Database Management*, 7(1):5–15, 1996.

[CGS97] S. Ceri, P. Grefen, and G. Sánchez. WIDE – A Distributed Architecture for Workflow Management. In *7th Int. Workshop on Research Issues in Data Engineering*, Birmingham, 1997.

[DKM+97] S. Das, K. Kochut, J. Miller, A. Sheth, and D. Worah. ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for $METEOR_2$. Technical Report #UGA-CS-TR-97-001, Department of Computer Science, University of Georgia, 1997.

[Dog97] A. Dogac et al. Design and Implementation of a Distributed Workflow Management System: METUFlow. In *Proc. of the NATO Advanced Study Institute on Workflow Management Systems and Interoperability*, pages 61–91, Istanbul, 1997.

[EG96] J. Eder and H. Groiss. Ein Workflow-Managementsystem auf der Basis aktiver Datenbanken. In J. Becker, G. Vossen, editor, *Geschäftsprozeßmodellierung und Workflow-Management*. International Thomson Publishing, 1996.

[HS96] P. Heinl and H. Schuster. Towards a Highly Scaleable Architecture for Workflow Management Systems. In *Proc. of the 7th Int. Workshop on Database and Expert Systems Applications*, pages 439–444, Zurich, 1996.

[Ley97] F. Leymann. Transaktionsunterstützung für Workflows. *Informatik Forschung und Entwicklung, Themenheft Workflow-Management*, 12(2):82–90, 1997.

[MWW+98] P. Muth, D. Wodtke, J. Weißenfels, A. Kotz-Dittrich, and G. Weikum. From Centralized Workflow Specification to Distributed Workflow Execution. *Journal of Intelligent Information Systems*, 10(2):159–184, 1998.

[Pic98] G. Piccinelli. Distributed Workflow Management: The TEAM Model. In *Proc. of the 3rd IFCIS Int. Conf. on Cooperative Information Systems*, pages 292–299, New York, 1998.

[RBD99] M. Reichert, T. Bauer, and P. Dadam. Enterprise-Wide and Cross-Enterprise Workflow-Management: Challenges and Research Issues for Adaptive Workflows. In *Proc. Workshop Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications, 29. Jahrestagung der GI (Informatik '99)*, Paderborn, October 1999.

[RD98] M. Reichert and P. Dadam. ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[SM96] A. Schill and C. Mittasch. Workflow Management Systems on Top of OSF DCE and OMG CORBA. *Distributed Systems Engineering*, 3(4):250–262, 1996.

[SNS99] H. Schuster, J. Neeb, and R. Schamburger. A Configuration Management Approach for Large Workflow Management Systems. In *Proc. of the Int. Joint Conf. on Work Activities Coordination and Collaboration*, San Francisco, 1999.

[WHKS98] M. Weske, J. Hündling, D. Kuropka, and H. Schuschel. Objektorientierter Entwurf eines flexiblen Workflow-Management-Systems. *Informatik Forschung und Entwicklung*, 13(4):179–195, 1998.

Most of our publications are available at the following URL: http://www.informatik.uni-ulm.de/dbis/papers