Towards a New Dimension in Clinical Information Processing

P. Dadam, M. Reichert
University of Ulm, Dept. Databases and Information Systems, 89069 Ulm, Germany

1 Introduction

Clinical information processing is a big challenge for computer-based information systems, especially when looking at the support of clinical workflows (i.e., clinical processes) [1, 2, 3, 4]. From the viewpoint of the medical personnel the current situation can be characterized as follows (cf. [1, 5]): Organizational and medical tasks are usually accompanied by a large amount of information which has to be intellectually processed and which has to be put into relation to the problems of the individual patient. In addition, the personnel has to (intellectually) obey and manage non-trivial workflows (as laid down, e.g., in medical guidelines). Typically, these workflows have to be performed in parallel under care and attention of time and resource constraints [1]. In addition, execution sequences and interdependencies among tasks have to be obeyed. Today's (clinical) information systems usually support the execution of single tasks, but they leave the monitoring and execution of the process itself to the users (cf. [5]). (And if there is some computer-based support then it usually covers only small sections of the overall process.)

The reason for this situation is that the direct support of workflows by computers is not as simple as it may look at first glance. In principle, a (clinical) workflow can be implemented as a large computer program which has to perform different tasks during its execution (as many other computer programs do as well). Opposed to an ordinary computer program, it has, however, to switch between different users, it must be able run very long (days, weeks, months) even in the presence of system crashes (i.e., the typical "all or nothing" semantics of database programs is not applicable), it must be able to invoke foreign programs, and it must be executable in heterogeneous, distributed environments, to name only some of the differences and challenges (cf. [1, 5, 6]).

In normal programs, the execution sequence (i.e., the so-called control flow) is usually an integral ("hard-wired") part of the program code. This causes a severe problem for supporting real-life processes which are changing over time quite frequently [10]. If they have been coded deep inside the programs, computer specialists would be needed whenever a change will become necessary. As we know, source code modifications always bear the risk of program errors and thus instability. And this "hard-wiring" approach has also another significant weakness: The test whether the computer-based process is really reflecting correctly what the user wants to have, can only be performed after large portions of the overall implementation have been completed. That is, this evaluation is performed at a very late (and thus very expensive) point in time. – Taking all together, this approach is not reasonably feasible, especially not in clinical environments.

2 Methods

Workflow management systems (WfMS) which appeared recently at the market-place point out how a computer-based process support can be implemented in a better way [7, 8]. By separating the control flow (i.e., the "process logic") from the pure application code, these systems offer a lot of interesting features: The process is defined separately from the application functions and is made known to the WfMS in a very early stage of the implementation process. This approach gives the chance to validate the process flow by the user before the implementation of the application functions starts. In addition, (high-end) WfMS (cf. [7]) have been typically designed to invoke application functions. These may be existing external programs (with external invocation capabilities) or especially developed application components. Thus, these WfMS address the issues of integrating existing (heterogeneous) application systems as well as component-based software development. In addition, WfMS allow to track and to monitor the status of processes and even single tasks, and they make it possible to retract a task from an actor and to assign it to another user at run-time.

Although looking very promising, today's WfMS reveal severe weaknesses which limit their applicability to clinical processes [1]. Most severe is the lack of flexibility [9]. That is the possibility to deviate, if required (e.g., in an emergency case), ad hoc in the running process (and only for the patient under consideration) from the pre-planned task sequence by inserting additional steps, by skipping steps, by moving steps, or by changing step attributes [9, 10]. If features of this kind are presently offered at all, then either the workflow designer has to anticipate already at modeling time that such an exception may occur when executing this workflow (i.e., "ad hoc" is not really "ad hoc" in this case), or it requires profound knowledge of the workflow description formalism (e.g., the usage of a Petri Net editor) to perform the desired modification. Commercial WfMS which support such dynamic changes are limited to rather simple workflow (WF) applications where only text documents are exchanged ("e-mail attachment workflow") or the changes may lead to inconsistencies (e.g., lost updates or deadlocks) or even program crashes (e.g., invocation of a task program with missing input parameters) in later steps of the WF [9]. In addition, the support of temporal constraints, if available at all, is usually restricted to a simple supervision of deadlines. More advanced features like the supervision of minimal and maximal time distances, which would be very helpful in the clinical domain, are usually not available [1].

Since a few years, the aspect "flexibility" has received more and more attention in the WF research community (e.g., [9, 11, 12, 13]. The approaches pursued are quite different and only an in-depth analysis of the underlying WF meta model, the kind of deviations that are supported, and the kind of consistency guarantees in the context of such dynamic changes of WF instances at run-time (if supported) can reveal the real differences and limitations. Due to lack of space we have to omit this detailed treatment here and will only sketch the different approaches. A more detailed discussion of related work can be found in [1, 9].

Numerous approaches do not support ad-hoc changes of the kind described above, but they give the user at least some flexibility at run-time. MOBILE [14], for example, allows the WF designer to omit some aspects of the WF model at build-time (e.g., the order of tasks), which then can be specified by end-users at run-time. HOON [15] does not offer this flexibility but it allows to use a kind of "placeholder"-tasks which are substituted by the concrete (already existing) task (or sub-workflow) at run-time. MOVE [16] goes even further and allows this task or sub-workflow to be dynamically defined at run-time. That is, it supports "late modeling" of tasks. Many of the approaches, which support dynamic changes in the broader sense, are based on the object migration model which regards a WF instance as an object ("circulation folder") which is sent from actor to actor according to the mod-

eled control flow [17]. This approach works best if the simple metaphor of a circulation folder with no parallelism or more advanced concepts like temporal constraints is sufficient. Other approaches combine graph-based WF models with rule-based extensions for exception handling [18, 19, 20] or deal with the evolution of the WF schema [21, 22].

To sum up, it can be said that workflow management is receiving more and more attention in the research community. Many valuable contributions to different aspects of this type of technology and applications have already been made and further developments can be expected.

3 Results

From the very beginning, the ADEPT workflow project was heavily influenced by the clinical application domain (cf. [5, 6, 23]). Within such a domain, *robustness* and *security* on the one hand and *flexibility*, i.e., ad-hoc deviations from the preplanned execution order, on the other hand, must be considered in conjunction. As one cannot expect to have computer experts as end-users (just the opposite), the specification of ad-hoc deviations must be made simple and user-friendly. In addition, instant consistency checks concerning the desired modification must be performed by the WfMS and be completed within seconds. The WfMS-based support of temporal constraints, like deadlines for tasks or minimal and maximal time distances between tasks, is also a very important issue. This feature becomes especially interesting in the context of dynamic workflow changes, where the insertion of a new step may lead to problems with time constraints in the sequel.

In the ADEPT project, we, therefore, have decided from the very beginning, to consider all these issues not separately but in conjunction. In the following we outline the features and the architecture of the ADEPT WfMS. This gives some insights what functionality future WfMS may offer. ADEPT uses a *graph-based WF model* which allows to model sequences, exclusive and parallel branchings, and loops, in a very direct and natural way. It also allows to express priorities and expected deviations from the "normal" execution sequence, and it allows to model the data flow between the different WF activities.

In addition, the ADEPT WfMS supports *dynamic WF changes*. That is, in-progress WF instances can be modified by authorized end-users during run-time. The end-user only has to express what he or she wants to do, e.g., to insert one or more steps, to delete a step, to move a step, or to change a step attribute (e.g., an actor assignment or a temporal constraint). All consistency checks necessary in this context, like, e.g., ensuring a correct data flow, is automatically done by the ADEPT WfMS. For this purpose, a sound theoretical basis for the ADEPT WF model and its operations for structural changes has been developed. It allows to express all kinds of changes via (perhaps combinations of) basic WF graph manipulation operations. In this context, the system checks whether the desired operation is valid and it determines how the internal state markings of task nodes and control edges have to be set after the modification has been performed. Moreover, the WF model has been carefully designed such that these checks and state transitions can be performed very efficiently (cf. [1, 9, 10] for details). These features are complemented by the ability to support temporal constraints like those described above. Whenever a temporal constraint is specified or whenever a dynamic change is performed, the ADEPT WfMS checks (as far as possible) whether these constraints can be met in the sequel. If this is not the case, a respective warning will be given. Fig. 1 illustrates some of these concepts.

Clinical information systems, especially *hospital-wide* information systems, must be capable to deal with many patients and with many end-users. The same does also hold for WfMS-based clinical information systems. This means that the WfMS must be able to deal with hundreds up to thousands of active WF instances, especially if short-term and long-

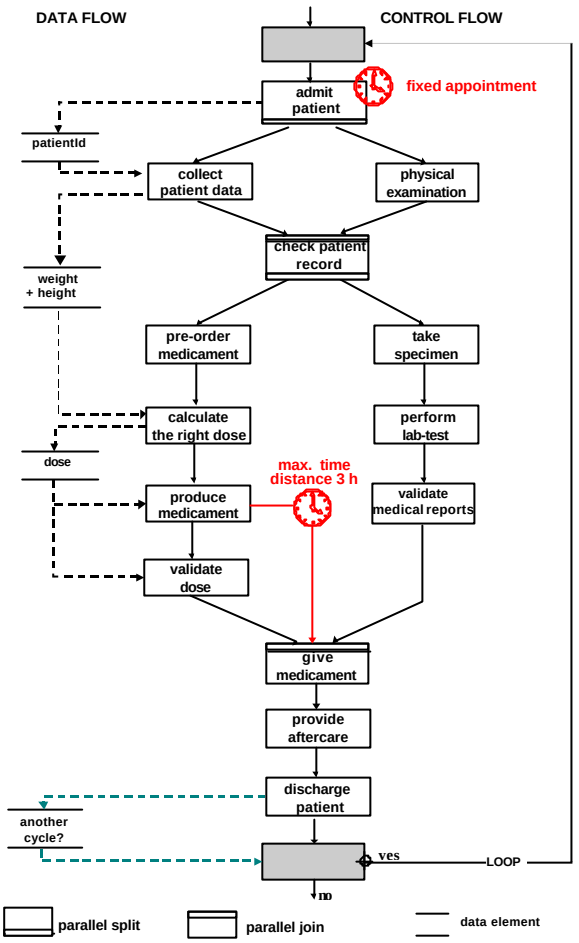

Fig. 1: *Modeling of Processes in ADEPT*

term patient treatment (e.g., cancer therapy) has to be supported. Thus server load, network traffic, and scalability become an issue. The ADEPT WfMS solves the problem in two ways: When running in a hospital-wide environment, ADEPT attempts to minimize communication costs by transferring the control for the running WF instance to another WF server (in another domain) if this helps to reduce communication costs. Most of the necessary computations are already done at build-time by analyzing the probabilities for the execution of the different WF tasks in the different domains. This helps to minimize the resulting overhead at run-time significantly. – *Migration* of WF control can mean, that two parallel branches of a given WF instance are controlled by different WF servers as indicated in Fig. 2. To keep the communication overhead low, the different servers controlling one WF instance do only communicate with one another if really necessary. This is, e.g., the case, if the user wants to perform a dynamic change which affects parts of the WF instance (perhaps due to temporal constraints or because of data flow dependencies) which are controlled by another WF server. More details on these features can be found in [24, 25, 26].

Another important aspect is the handling of dependencies among different workflows (“*inter-WF dependencies*”). With this, we mean the specification of (global) dependencies "across" different (independently modeled) workflows and their enforcement at run-time. The goal is that some kind of "high-level" WF scheduler supervises the execution of steps from different WF instances and proposes desirable execution sequences for them (or refuses forbidden ones). The kind of application we have in mind here is the coordination of different examinations for the same patient; each of them is typically represented by an

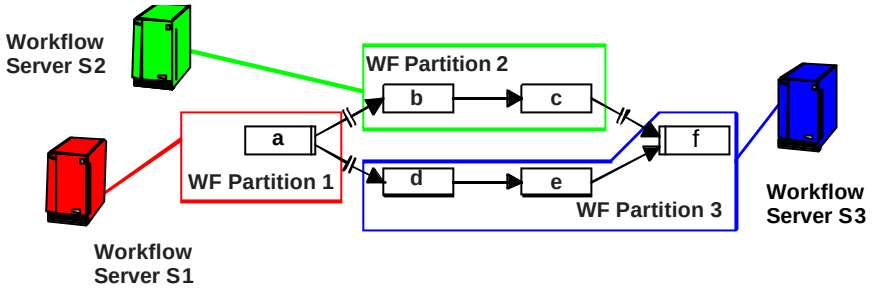

Fig. 2: *Partitioning of Workflows and Migration of WF Control*

individual WF instance, whereas some of these examinations or workflows respectively should be preferably performed in a certain order, however.

For long-running processes, ordinary transaction rollback is not applicable. Instead the WfMS must support some kind of *logical rollback* for already completed steps, if semantic failures (e.g., abortion of a medical examination) occur at run-time [7]. For this, the WfMS must allow the WF designer to assign appropriate compensation programs to each WF step. At this point, we have spent most time to understand and to solve the rollback problem in conjunction with dynamic WF changes. This includes, for example, issues related to the adaptation of a workflow's internal state (marking of nodes and edges, content of data elements etc.) as well as of its flow structure (e.g., when previously performed temporary WF modifications shall no longer be present after the rollback). Furthermore, one has to consider that the kind of compensation for a WF step may depend on the current state of the WF instance, on the point in time the failure occurs, or on other aspects. These dependencies complicate the definition of recovery spheres and they complicate application development. It, therefore, will be a key factor for the success or the break-down of WF technology, whether it will be able to adequately support WF designers in describing such spheres and in implementating WF steps as well as their compensation programs.

Many other aspects would also be worthwhile to mention within this context: The life time of dynamic changes which are applied within a loop construct [10], component-based software development, authorization issues, WF schema evolution, end-user interfaces, etc., but have to be omitted here because of lack of space.

4 Conclusions

Process-oriented information systems can be very valuable for the clinical personnel since they may actively support the processes in a hospital. By offering tasks right in time and when all information is available to perform them, and by obeying deadlines and other time constraints, it reduces the administrative overhead. Today's WF technology is still too limited in order to be broadly applicable in this scenario. However, research in WF technology is making quick progress. In the foreseeable future one can expect very powerful WfMS to appear at the market place, offering a powerful platform for implementing process-oriented information systems, also in the clinical domain. – The ADEPT WfMS prototype which has been sketched in this paper, is among the functionally most powerful WfMS and proves that one can really build systems of this kind which offer all this functionality within one system.

It also shows, however, that such high-end WfMS are large software systems, easily reaching the code complexity of high-end database management systems. The current implementation of the ADEPT prototype already comprises approx. 120,000 lines of Java code. This would probably increase by a factor of 4 (or even more) for a product version.

References

1. Dadam, P.; Reichert, M.; Kuhn, K.: Clinical Workflows - The Killer Application for Process-oriented Information Systems? *Proc. 4th Int'l Conf. on Business Information Systems*, BIS 2000, Poznan, Poland, April 2000, pp. 36-59
2. Müller, R., Heller, B.: A Petri Net-based Model for Knowledge-based Workflows in Distributed Cancer Therapy. *Proc. Int. EDBT-Workshop on Workflow Management Sys.*, Valencia, March 1998, pp. 91-99.
3. Stefanelli, M., et al.: Building Patient Workflow Management Systems by Integrating Medical and Organizational Knowledge. *Proc. MEDINFO'98*, Seoul, August 1998, S. 28-32.
4. Penger, O.-S.: Zur Entwicklung und Anwendung einer werkzeugunterstützten Methode für die Gestaltung von Prozessen und vorgangsorientierten Informationssystemen im Krankenhaus. Ph.D. Thesis, University of Hildesheim, Shaker, 1997.
5. Reichert, M.; Dadam, P.; Mangold, R.; Kreienberg, R.: Computer-based Support of Hospital Processes - Concepts, Technologies, and their Application. *Zentralbl Gynakol*, 122 (2000), 56-70 (in German)
6. Reichert, M.; Schultheiß, B.; Dadam, P.: Experiences with the Development of Process-oriented, Clinical Application Systems on Top of Workflow Technology (in German). *Proc. GMDS'97*, Ulm, Sept. 1997, pp. 181–187.
7. Leymann, F.; Roller, D.: Production Workflow – Concepts and Techniques. Prentice Hall, 2000
8. Dadam, P.; Reichert, M. (eds.): Proc. Workshop on Enterprise-Wide and Cross-Enterprise Workflow-Management: Concepts, Systems, Applications, (Workshop held in conjunction with the 29th Annual Meeting of the German Computer Society (GI - Informatik'99)), Paderborn, October 1999.
9. Reichert, M.; Dadam, P.: ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Inf. Sys, Special Issue on Workflow Management Systems*, 10 (1998), 93-129
10. Reichert, M.; Hensinger, C.; Dadam, P.: Supporting Adaptive Workflows in Advanced Application Environments. *Proc. EDBT-Workshop on Workflow Management Systems*, Valencia, March 1998, pp. 100-109
11. Klein, M.; Dellaroca, C.; Bernstein, A.: Towards Adaptive Workflow Systems. Workshop held in conjunction with the CSCW-98 Conf. on Computer-Supported Cooperative Work, Seattle, WA, November 1998
12. Borgida, A.; Murata, T.: Tolerating Exceptions in Workflows: A Unified Framework for Data and Processes. *Proc. Int'l Conf. on Work Activities Coordination and Collaboration*, San Francisco, USA, February 1999, pp. 59-68
13. Joeris, G.; Herzog, O.: Towards Flexible and High-Level Modeling and Enacting of Processes. *Proc. Int'l Conf. on Advanced Information Systems Engineering (CAiSE)*, Heidelberg, Germany, June 1999, LNCS 1626, Springer Verlag, pp. 88-102
14. Jablonski, S.; Stein, K.; Teschke, M.: Experiences in Workflow Management for Scientific Computing. *Proc. 8th Int'l Workshop on Database and Expert Systems Applications DEXA 97*, Toulouse, France, September 1997, pp. 56- 61
15. Han, Y.: Software Infrastructure for Configurable Workflow Systems. PhD Thesis, TU Berlin, 1997
16. Hagemeyer, J.; Herrmann, T.; Just-Hahn, K.; Striemer, R.: Flexibility in Workflow Management Systems. In: Likowski, R. et al. (eds.): Usability Engineering. Teubner-Verlag, 1997, pp. 197-190
17. Karbe, B.; Ramsperger, N.; Weiss, P.: Support of Cooperative Work by Electronic Circulation Folders. *ACM SIGOIS Bulletin*, 11 (1990), 109-117
18. Joeris, G: Defining Flexible Workflow Execution Behaviors. In [8], pp. 49-55
19. Müller, R.; Rahm, E.: Rule-Based Dynamic Modification of Workflows in a Medical Domain. *Proc. BTW 99*, Freiburg, Germany, February 1999, pp. 429-448
20. Casati, F.; Ceri, S.; Paraboschi, S.; Pozzi, G.: Specification and Implementation of Exceptions in Workflow Management Systems. *ACM Transactions on Database Systems*, 23 (1999)
21. Casati, F.; Ceri, S.; Pernici, B.; Pozzi, G.: Workflow Evolution. *Data and Knowledge Engineering*, 24 (1998), 211-238
22. Kradolfer, M.; Geppert, A.: Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration. *Proc. IFCIS Int'l Conf. on Cooperative Information Systems*, Edinburgh, Scotland, September 1999, pp. 104-114
23. Dadam, P.; Kuhn, K.; Reichert, M.; Beuter, Th.; Nathe, M.: ADEPT: An Integral Approach to the Development of Flexible, Reliable Cooperative Assistant-Systems in Clinical Application Domains. *Proc. GISI'95*, Zurich, Switzerland, Sept. 1995, pp. 677-686 (in German)

24. Bauer, T.; Dadam, P.: A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. *Proc. CoopIS '97, Int'l Conf. on Cooperative Information Systems*, Kiawah Island, South Carolina, June 1997, pp. 99-108
25. Bauer, T.; Dadam, P.: Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows. In [8], pp. 25-32
26. Bauer, T.; Dadam, P.: Efficient Distributed Workflow Management Based on Variable Server Assignments. University of Ulm, Faculty of Informatics, Report No. 99-09, November 1999