# Supporting Flexible Processes with Adaptive Workflow and Case Handling

Christian W. Günther*, Manfred Reichert†, Wil M.P. van der Aalst*
*Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
Email: {c.w.gunther, w.m.p.v.d.aalst}@tue.nl
†University of Ulm, Institute of Databases and Information Systems, 79069 Ulm, Germany
Email: manfred.reichert@uni-ulm.de

***Abstract*—Workflow management technology has profoundly transformed the way complex tasks are being handled in modern, large-scale organizations. However, it is mostly those systems' inherent lack of flexibility that hinders their broad acceptance, and that is perceived as annoyance by users. In this context, Adaptive Process Management and Case Handling provide two very different paradigms, which both attempt to make process management more flexible and user-friendly. In this paper, we compare strengths and weaknesses of these two paradigms, and point out situations in which each is particularly appropriate. We further outline ways, in which either technology can be enhanced by crucial concepts from the other. This integration of flexibility approaches has the potential to remedy fundamental problems still present in each technology on its own.**

## I. Introduction

Workflow management (WfM) technology has been a tremendous boon for organizations handling complex and long-running processes. Its fundamental idea is quite simple: The process is broken down into small and quickly manageable tasks. Also, the organizational structure (i.e., roles and groups) and actor assignments need to be specified. Based on this information, the Workflow Management System (WfMS) is able to control and monitor the execution of the process, to activate tasks as soon as they become available, and to assign them to appropriate users [10]. The benefits of this approach are obvious. Users of a WfMS can work more autonomously, the system facilitates the exchange of case-related information, and it becomes easier for the management to monitor the progress of all tasks. However, there is also a serious downside to this approach. Once an operational process is implemented based on a traditional WfMS, it is very hard to modify it. With often thousands of running cases (i.e., instances) of a process, it is seemingly impossible for a process (re-)designer to take into account all implications of his changes.

Changes to a process can be classified into two major groups. *Exceptional change* is characterized by rare and unforeseen events during process execution, which require the process to be "bent" in order to fit the case in question [12]. *Evolutionary change*, on the other hand, describes a planned migration of a process to an updated schema version which, for example, implements new legislation, corporate policy, or best practices [5], [7], [14]. Both of these change classes require specific capabilities of the system supporting them. In case of exceptional changes, it is imperative to handle them quickly and effortless, otherwise users will rather bypass the system [16]. For evolutionary change, the most important requirement is that all running cases can be transferred to the new process definition correctly and automatically. It is often impossible to handle such a transition manually, and every error in migrating cases can result in costly downtime.

Many approaches have been proposed to deal with the problem of change in a WfMS [2], [18], [20], [5], [7], [14], yet at this point in time, none of them has seen widespread acceptance nor significant market penetration. One quite recent approach are *Worklets*, which have been implemented in the YAWL system [4]. The basic idea of worklets is to introduce a flexible hierarchy to process definitions. On a high level, the process is designed with generic activities, whose implementation is left open. These "gaps" are dynamically filled with an implementation at runtime, whereas there may be multiple implementations for each high-level activity. A similar approach is provided by *Pockets of Flexibility* [15] and *MOVE* [8]. While these approaches are an adequate solution for many situations, they require the points of flexibility (i.e., where deviation is supposed to occur) to be known in advance. Case-Based Reasoning (CBR) has also been proposed as a solution for managing processes in a more flexible manner [19]. The solution to a given problem is derived from a set of cases, which had been successfully executed previously. The problem with CBR-based approaches is their reliance on a sufficiently large set of "trustworthy" cases. Further, users will need to be given significant freedom in adapting cases, defying the widespread desire to retain a certain amount of control.

In this paper, we focus on two of the more successful approaches for providing flexibility in process execution, which have also gained significant adoption in industrial practice: *Adaptive Process Management* and *Case Handling*. The two subsequent sections introduce these concepts in more detail, followed by an assessment of their strengths and weaknesses in Section IV. Section V discusses potentials for integrating concepts from both approaches in order to overcome these limitations, before Section VI closes the paper with conclusions and an outlook on future work.

## II. Adaptive Process Management

The principle of production workflow is based on atomic, strictly defined activities, whose precedence relations are or-

ganized in a rather rigid process model. This model fares well for relatively stable types of processes (e.g., manufacturing processes), which rarely need to be updated. Also, when cases typically have a rather short runtime, the usual solution for dealing with (evolutionary) change is to finish the already running cases according to the old process definition, while newly started cases follow the updated process.

However, there are many application domains with long-running cases, such as insurances, health care treatments, judicial court cases, and the like. When a law is being passed that affects a certain type of insurance, it is often no option to treat already running insurance cases without taking this new law into account. Comparable problems may arise from changes in corporate policies, mergers and acquisitions, and the general evolution of rules and norms in society. These situations are classified as *evolutionary change* (cf. Section I), meaning that a general adaption of the process, including presently running and future cases, is desired [5], [7], [14].

The second important category is *ad-hoc change*, describing an exceptional modification of a small subset of cases [12]. Some extraordinary situations, such as time pressure, exceptions, or specific customer demands, require an organization to respond quickly and non-bureaucratically. Yet, if employees have to follow a rigidly specified process definition which conflicts with that goal, oftentimes they simply choose to bypass the system, solving the problem "behind its back". One problem resulting from this conflict is, that such cases can no longer be correctly monitored and traced, as they are not faithfully represented in the system. The fundamental problem, though, is that the production WfMS, intended to help and serve the workforce, becomes a pain and a burden to use.

Adaptive Process Management (APM) constitutes an evolutionary extension of the production workflow paradigm which intends to remedy the deficiencies related to dynamic process change. While there exist a number of alternative "flavors" of APM [14], [19], [20], this paper concentrates on the approach implemented in the ADEPT system [12], [13]. The ADEPT approach is based on a simple, block-structured process model notation, i.e. blocks of parallel or alternative behavior have single, dedicated start and end points, and are mutually disjoint. Activities may be equipped with pre- and postconditions, defining whether they are valid within their given context in the process model. These properties allow to define correctness criteria for basic change operations, such as moving an activity to a new position in the process, removing it, or inserting a new activity. From these basic change operations, higher-level modifications can be compiled (e.g., skipping part of the process in order to execute a later activity first). In combination with correctness checks, this model allows end users of the system to intuitively, interactively, and quickly perform ad-hoc changes to single cases, while ensuring that the result will still be able to execute properly [12].

Moreover, the ADEPT framework also allows for the implementation of evolutionary change in a comparatively straightforward fashion. When a process designer applies modifications to a process on the type level (i.e., affecting the general definition of a process, including all cases), he enjoys the same support of pervasive correctness checks, ensuring the final result is a valid process model. On top of that, ADEPT can also check all running cases of the original process definition, whether they can be safely transferred to the modified version (and perform this transfer automatically). The combination of intuitive change operations with pervasive correctness checks makes ADEPT an environment which is *actively supporting* change, rather than punishing it with maintainability nightmares.

Other examples for APM implementations are *WASA2* [20], *WIDE* [6], CAKE2 [11], and *CBRFlow* [19].

## III. Case Handling

While production workflow, emphasizing the routing between atomic activities, is strongly process-oriented, the case handling paradigm focuses mainly on the *case* itself [1], [3]. The case is the primary object to be manufactured, e.g. the outcome of a lawsuit or the response to a customer request. Thus single activities diminish in importance in favor of the larger context. They are no longer considered to be atomic steps that have to be performed in an "all or nothing" manner, but rather make up the logical partitions of work between which a transition from one worker to another is possible.

Like in traditional WfM there exists a set of precedence relations between single activities making up a process. However the primary driver for progress is no longer the event related to explicit activity completion, but the availability of values for data objects. While production workflow clearly separates the process from associated data, Case Handling integrates both far more closely, using produced data not only for routing decisions, but also for determining which parts of the process have already been accomplished. With Case Handling, each task is associated to three sets of data objects, each for distinct purposes. The first association is between a task and all data objects that shall be accessible while performing this task. Further on, all data objects *mandatory* for a task have to be set (i.e., bound to a value) before the task itself is considered to be accomplished by the system. Finally, every data object can have a random number of tasks to which it is *restricted*, meaning that it can only be altered while performing one of these tasks. User-interactive tasks are connected to a *form*, each providing access to a selection of data objects. Note that one form can be associated to multiple tasks. Furthermore it is possible to associate a form to the case itself, i.e. the case can be accessed at any point in time.

In order to provide an abstract introduction into the basic principles of the Case Handling paradigm, Figure 1 shows a simplified example of a case type, the Case Handling analogy to a workflow process definition: Three tasks $A$, $B$ and $C$ are making up the process, sequentially chained by causal relationships denoted by connecting arrows. Their *mandatory* relationships to the three data objects $x$, $y$ and $z$ below are denoted by curved arcs, as are their associations with the forms $M$ and $N$. As can be seen in the illustration, tasks $A$ and $B$ share the same form $M$, providing access to data

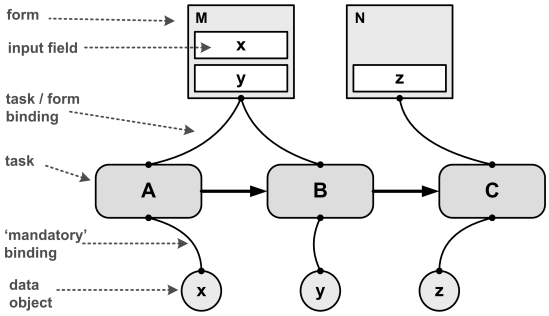


Fig. 1. Simplified example case type

objects $x$ and $y$. If a properly authorized worker now starts handling task $A$, the associated form $M$ will open and he will start providing values for the presented data objects. In a traditional WfMS activity $A$ would not be finished before form $M$ is closed. However the Case Handling system regards $A$ as being finished as soon as a value for $x$ has been provided (and confirmed appropriately), automatically enabling task $B$ in the background. If the worker would now close form $M$, another employee could pick up the case where he left it, starting task $B$, which would provide the same form $M$ with $x$ having a value filled in (that could now be changed again). Another possibility, however, is that the first worker keeps on handling the form, providing also a value for $y$. This would correspondingly trigger the *auto-completion* of task $B$ (as all associated mandatory data elements have been provided) and activate task $C$. Note that if a worker closes a form after filling out only parts of the mandatory data fields of a task, despite the task not being considered as finished, data already entered will remain available to the person continuing to work on that task. Such closely intertwined relationship between data and process obviously abandons their, often unnatural, separation so rigidly pursued in traditional WfM. With the status of case data objects being the primary determinant of case status, this concept overcomes a great deal of the problems described in the introduction:

- Work can now be organized by those performing them with a far higher degree of freedom. Activities can either be performed only partially, without losing intermediary results, or multiple related activities can be handled in one go, surpassing the weakened border between tasks.
- Routing is no longer solely determined by the process model. Case types can be designed in such a way, that multiple activities are enabled concurrently, providing different ways of achieving one goal.

In addition to the *execute* role, specifying the subset of resources allowed to handle a specific task, the Case Handling principle introduces two further roles crucial for operation. The *skip* role allows workers to bypass a selected task, which could be interpreted as an *exception*. Exceptions, like skipping a background check for trusted clients, are likely to occur quite frequently. The ability to grant the *skip* role to a senior worker renders the necessity for implementing such bypass obsolete, thus greatly simplifying the whole case type. It has to be noted that in order to skip a taks all preceding tasks that have not been completed yet have to be skipped (or completed) beforehand. This is necessary to ensure an unambiguous state of the process. Traditional workflow definitions use loops for repeating parts of the process, e.g. because they have not yielded an expected result. Case Handling obsoletes such construct by the introduction of a *redo* role, enabling its bearer to deliberately roll the case's state back and make a task undone. In doing so, the values provided for data objects during this task are not discarded but merely marked as *unconfirmed*, so that they serve as template when re-executing the affected task. Similar to skipping, before a task can be *redone* all subsequent tasks that have already been completed need to be rolled back as well before.

Intertwining authorization with distribution of activities has been one major flaw of traditional workflow technology. In a Case Handling system, the former *in-tray*, i.e. a list of all activities the user is authorized to perform and that he can choose from, has been replaced by a *query mechanism*. This tool can be used to look for a specific case, based on certain features (e.g. case data, or enactment meta-data like the start date of a case instance). Moreover it can be used to create predefined queries tailored to each worker (or, group of workers). A manager is no longer overwhelmed with all possible activities that he can perform, but only those which require a certain role (or, e.g., case instances of an order where the combined value exceeds $1000). Obviously the query mechanism can also be used to perfectly imitate a classic in-tray, as found in a traditional WfMS, if required.

## IV. Strengths and Weaknesses

Both Adaptive Process Management and Case Handling have proven to be viable, flexible alternatives to rigid production workflow, providing effective means for addressing dynamic change. However, their basic approaches differ significantly, which leads to very particular "flavors" of flexibility. This section explores the relative advantages and downsides of each technique, positioning them for specific applications.

### A. Adaptive Process Management

The approach for providing flexibility in Adaptive Process Management reveals its strong heritage from traditional WfM. As such, the problems of rigid solutions are addressed from within the paradigm of WfM, while its well-known metaphors (e.g., defined activities, explicit control flow specification) remain perfectly valid. Although often overlooked, this is probably one of the strongest points of this approach in practice, rendering this technique a highly suitable solution for organizations already familiar with the WfM paradigm. It allows for a smooth transition and prevents lengthy periods of adjustment (as necessary with more revolutionary approaches).

One particular advantage, derived from this property, is that *process owners* (i.e., the management) retain tight *control* over their processes. The ability to modify both single cases and complete process types can be restricted to a subset of

trusted employees [17]. That way, flexibility may be gained in a controlled manner, limited to problematic situations and skilled personnel. The value of this feature is not in keeping a tight grip on your workforce's actions, but in ensuring accountability and the adherence of operations to certain standards or regulations. If each case's process model were to look different, it would be rather difficult to retain an overview on the state of affairs and the progress of individual cases. For example, by enforcing that the process definition always contains a defined set of *milestones* (i.e., points in the process where a well-known state is supposed to be accomplished), it is ensured that *monitoring* the operational state of the organization is possible (albeit, on a coarser-grained level).

In large organizations, it is often infeasible to train every single employee, such that he is able to master modifying the process model of a case. Being able to limit the set of personnel authorized to perform such changes allows to concentrate this training (which may also involve legal matters, organizational policies, etc.) on a dedicated group of employees. There are situations, however, where employees will be very vocal on working autonomously, and not be pressed into working according to a predefined process. One example of this are hospitals, which need to adhere to medical guidelines and legal frameworks, yet are facing the resistance of doctors having strong opinions about their personal, preferred way of working. Adaptive Process Management can alleviate such conflicts, whereas a possible compromise may be to allow such employees the rights to adapt "their" cases.

The medical field is, in general, an excellent example for the suitability of Adaptive Process Management, which we will use to summarize and highlight the most useful traits of this approach [9].

- *Compatibility:* Medical guidelines, which model the sequence of steps in treating a specific kind of illness, are ubiquitous in the medical domain, and strongly resemble traditional workflow definitions. If an organization is already subscribed to the workflow paradigm, APM offers a straightforward migration path.
- *Strict definition:* While changes to a process model are possible, the default behavior of an APM system is to support a rigid, strictly defined process. This is important when there are strict guidelines or laws to abide, as is common in the medical domain.
- *Exception handling:* Especially in emergency situations, being able to deviate from prescribed procedures in a quick and well-supported fashion can be essential. Providing these means, together with integrated correctness checks, APM facilitates the work of doctors while helping them to not overlook potential problems.
- *Accountability:* As a highly sensitive domain, medical institutions such as hospitals always need to be able to lay out their procedure in a specific case. Being able to limit the authority to perform process modifications allows them to prevent too much variation.
- *Retention of control:* While flexibility is a necessity of hospitals, there are also requirements to keep process modifications under control. APM can be used to mirror the traditional structure of an organization, e.g. so that doctors may choose to ignore certain guidelines, while nurses do not enjoy this privilege.
- *Type migration:* Changes in legislation, scientific advances, and evidence-based medicine are constantly changing the way work is supposed to be performed in a hospital. Being able to perform the necessary adaptations once, and having all currently running cases benefit from them immediately, is often a necessity.

While hospitals, where doctors are usually very well aware of the standard procedures, are highly suitable for Adaptive Process Management, this approach may be rather problematic for domains where the processes are more tacit and variable. Employees which are neither aware of the complete process, nor have any knowledge about process specification and correctness criteria, will have a hard time implementing even small modifications. If such tacit understanding of the process is combined with a high variability, where deviations occur on a frequent basis, the application of Adaptive Process Management may be bound to fail. It is also important to point out that one of the greatest strengths of this approach is being able to support rigidly specified, control flow-driven processes. Flexibility is rather an "add-on" to this basic feature. Some processes are, by nature, neither well-structured, nor primarily reliant on pure control flow. For such processes, Adaptive Process Management will introduce similar problems as traditional workflow management technology – the fact that it also brings tools to remedy these problems is not particularly worthwhile here.

In such situations, it is advisable to choose a less rigid solution which better respects the inherently flexible, data-driven nature of the process. This is one of the main strengths of Case Handling, which will be covered in the following subsection.

### B. Case Handling

Adaptive Process Management remedies the problem of dynamic change in the environment by providing means to modify the process definition during execution. In contrast to this approach, Case Handling addresses the same problem by *anticipating* volatile environments and thus *avoiding the need for process modification*. The fundamental property of Case Handling supporting this approach is its strong reliance on the *case metaphor*, where the importance of activities is diminished in favor of the global context, and its use of *data dependencies as primary driver* of the process. These two features are a perfect match for many knowledge-intensive industries, such as insurance, governmental bodies, or call centers. The case corresponds to, e.g., an insurance claim, customer request, etc. Additionally, handling such cases is mainly performed *from within the system itself*, i.e. the outcome of activities is completely represented in the system by the data produced (as opposed to a healthcare process, for example, where the main effects are not visible within the WfMS).

We use the example of an insurance company to illustrate some of the strong points of Case Handling:

- *Case metaphor:* The case, as an insurance claim of a client which needs to be handled, fits naturally into the system. Usually, employees are familiar with this metaphor and will need little training in order to become acquainted with a Case Handling System.
- *Data as the main driver:* All "products" which are used or manufactured while handling a case are contained within the case itself as data (e.g., address, claim value). The concept that a certain state in the process is reached when a set of data becomes available (e.g., a letter can be sent once the address is known), also fits the application area naturally. This makes processes easier to design, and easier to understand for workers familiar with the domain.
- *Weak activity boundaries:* Activities are metaphors used to structure complex processes. The way in which a designer perceives a process to be partitioned into activities does not necessarily coincide with the perceptions of all employees. An employee may prefer to handle two activities in one go, or see that he cannot finish an activity mid-way. In a Case Handling system, this will not affect the correctness of the process, and will be handled transparently to the end user.
- *Skip and redo roles:* These roles are mostly necessary for responding to exceptions, e.g. when a deadline needs to be met. Thus, it is natural not to model these exceptions explicitly (i.e., using loops and bypass paths), but to instead use generic skip and redo operations on an as-needed basis. These roles can be assigned to superiors, who will need to acknowledge exceptions anyway.
- *Case query:* While a simple change from the classical in-tray of WfMSs, the case query makes a profound difference for end users. Work is no longer "pushed" to them by the system, but employees can look for specific types of work to perform which better fits their interests, skills, or simply their current mood. Rather than controlling the workforce in a mechanical way, the system becomes a discreet aid in accomplishing one's tasks.

Obviously, these benefits vanish if Case Handling is used in a context which is neither data-driven, nor has a natural case metaphor. Apart from this more general issue, there are a number of shortcomings in this approach which may pose limitations. One of these limitations is directly related to the strong role of the case metaphor, where the user handling a case has access to all information at once. This feature requires the case to be *exclusively locked* while being handled, which makes it impossible to handle a case concurrently.

Also, while the weak activity boundaries and data-driven nature of Case Handling largely prevent the need for explicit change, there are situations which require modification of the process definition. In this respect, a Case Handling system performs like a traditional WfMS – modified case types may only be used by newly started cases, while running cases need to follow the old definition. There are ways to work around this problem, e.g., by using global case forms to perform additional actions, or using the skip role to bypass legacy parts of the process. However, they remain "hacks" of the paradigm, which essentially does not support such situations.

The next section explores how ideas from Case Handling and Adaptive Process Management may be integrated, in order to overcome some of the shortcomings described, and to provide a more complete solution for realizing flexible processes.

## V. Integration of Approaches

Although Adaptive Process Management as well as Case Handling are successful at overcoming many problems, which occur when inherently flexible processes "clash" with traditional production workflow, we have seen that both paradigms also come with their specific shortcomings. While some of these problems are inherent to the fundamentals of the respective approach, others can be resolved with relatively small extensions. In this section, we explore how ideas from each approach can be introduced into the respective other approach, to remedy some of the described problems.

The rigid nature of process definitions does not necessarily constitute a problem in an APM system. It may even be desirable for applications, for which a larger degree of control and accountability becomes necessary. For many situations, the erosion of activity boundaries as provided by Case Handling is very beneficial, in the sense that it often avoids the need for explicit change in the first place. Integrating data as a primary driver to Adaptive Process Management would introduce this feature, and enable users to work in a more case-based manner in which they are less aware of the single activities handled. Much in the same way, it would also be rather trivial to introduce skip and redo roles to Adaptive Process Management. Many exceptional situations do not require the process model to be explicitly changed, it is just necessary to deviate on an ad-hoc basis. By mapping the effect of skip and redo actions onto equivalent modifications of the process model, their correctness can be ensured on the fly. In an Adaptive Process Management system, introducing data as a driver and dedicated skip and redo roles could serve as the "poor man's flexibility", as they do not require the end user to know anything about process design or correctness.

In addition, it would be straightforward to provide case queries instead of, or in addition to, the standard in-tray for providing work items. Although technically very similar to a push model, the query mechanism is a powerful means for end users to *customize* the system and to use it in a flexible way. On top of this, querying for work instead of having it assigned by the system makes a profound difference on a psychological level, putting the employee back in control.

The worst problem of the Case Handling approach is probably its inability to modify case types on the fly, and to migrate running cases to an updated case type. If one were to implement such a change manually, it could however be accomplished in a rather simple, although tedious, manner: The new case type is designed, based on the previous version,

and a new case is instantiated from it. Subsequently, all data from the (old) case in progress is entered into the new instance. The system will then auto-complete all tasks, for which data dependencies have been satisfied, essentially re-creating the state of the previous instance. From then on, the old case may be removed from the system, while work continues on the new instance, following the updated case type definition.

It is obvious that this procedure, besides the actual change of the case type, can equally be performed in a (semi-)automatic fashion by the Case Handling system itself. When problems occur during the replay of the old case (i.e., transferring its data values and auto-completing satisfied tasks), these problems can be resolved manually. Note that this approach cannot match the ease of performing the same action in an Adaptive Process Management system, as there are no correctness checks which could aid the user in redesigning the case type. However, it is certainly feasible to implement with comparably little effort, and would bring tremendous advantages when explicit change to the case type is required.

## VI. Conclusion and Outlook

In this paper, we have presented two relatively mature and successful approaches for dealing with flexibility in process management, Adaptive Process Management and Case Handling. Adaptive Process Management can be seen as an *evolutionary* technique, solidly based on traditional workflow, while extending it with features to dynamically and safely adapt the process definition at any point in time. Case Handling follows a more *revolutionary* approach, departing from the rigid structure of traditional workflow processes and their strict separation of data and control flow. We have discussed both approaches, comparing them with respect to their strong and weak points. While either of them can be applied in practically any setting, their respective strengths will only be able to play out to the fullest if combined with an *appropriate domain*. When applied to such appropriate domains the respective weaknesses of either approach are mostly negligible, however, there are a number of domains which fit neither approach perfectly. In Section V we have presented a number of ideas to integrate fundamental ideas of either approach into the other. Such extension will benefit the practical usability of both approaches, alleviating many of the identified downsides of each. The aim of our proposed integration is not to merge the approaches, but the proposed extensions have the potential to significantly increase the "radius" of application domains for each approach. Specialization and customization of process management for various domains should be welcomed, and appropriately supported with data interchange standards.

Future work should concentrate on approaches to further alleviate the downsides of each approach, and on making flexibility and change more of a *transparent* action in the process management system. Systems will need to be *designed for coping with a flexible world*, instead of forcing the environment into their limited paradigms. This is ultimately a requirement for the success of process management technology.

*Acknowledgements:* This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

## References

[1] W.M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001)*, pages 42–51. ACM Press, New York, 2001.

[2] W.M.P. van der Aalst and S. Jablonski. Dealing with Workflow Change: Identification of Issues and Solutions. *International Journal of Computer Systems, Science, and Engineering*, 15(5):267–276, 2000.

[3] W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.

[4] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In R. Meersman and Z. Tari et al., editors, *OTM Confederated International Conferences, CoopIS 2006*, volume 4275 of *Lecture Notes in Computer Science*, pages 291–308. Springer-Verlag, Berlin, 2006.

[5] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. *Data and Knowledge Engineering*, 24(3):211–238, 1998.

[6] F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. Snchez. Wide workflow model and architecture. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 1996. http://dis.sema.es/projects/WIDE/Documents/ase30_4.ps.gz.

[7] C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10 – 21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.

[8] T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured models are surprisingly useful for user-centered design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.

[9] R. Lenz and M. Reichert. It support for healthcare processes - premises, challenges, perspectives. *Data Knowl. Eng.*, 61(1):39–58, 2007.

[10] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

[11] M. Minor, D. Schmalen, and A. Koldeho. A workflow supported by a suspension. In *WETICE*, 2007.

[12] M. Reichert and P. Dadam. ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[13] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with adept2. In *Proc. 21st Int'l Conf. on Data Engineering (ICDE'05)*, pages 1113–1114, Tokyo, 2005.

[14] S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey. *Data and Knowledge Engineering, Special Issue on Advances in Business Process Management*, 50(1):9–34, 2004.

[15] S. Sadiq, W. Sadiq, and M. Orlowska. Pockets of flexibility in workflow specification. In *ER*, pages 513–526, 2001.

[16] D.M. Strong and S.M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems*, 13(2):206–233, 1995.

[17] B. Weber, M. Reichert, W. Wild, and S. Rinderle. Balancing flexibility and security in adaptive process management systems. In *OTM Conferences (1)*, pages 59–76, 2005.

[18] B. Weber, S. Rinderle, and M. Reichert. Change patterns and change support features in process-aware information systems. In *CAiSE*, pages 574–588, 2007.

[19] B. Weber, W. Wild, and R. Breu. CBRFlow: Enabling adaptive workflow management through conversational case-based reasoning. In *Proc. Eurpean Conf. on Case–based Reasoning (ECCBR'04)*, pages 434–448, Madrid, 2004.

[20] M. Weske. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In *Proc. Hawaii International Conference on System Sciences (HICSS-34)*, 2001.