Building Conference Proceedings Requires Adaptable Workflow and Content Management

Jutta A. Mülle Klemens Böhm
{muelle|boehm}@ipd.uni-karlsruhe.de

Nicolas Röper Tobias Sünder
{roeper|suender}@ipd.uni-karlsruhe.de

Institute for Program Structures and Data Organisation
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany

ABSTRACT

ProceedingsBuilder is a system that helps the proceedings chair of a scientific conference to carry out his chores. It has features of both workflow management systems (WFMS) and content management systems (CMS), in order to collect the material for the printed proceedings and other products. ProceedingsBuilder has been operational at several conferences, including VLDB 2005. When using ProceedingsBuilder, we had a very intense lesson which kinds of workflow adaptations may become necessary. Existing WFMS do not offer support for most of them. The concern of this article is to describe and classify these various requirements regarding adaptation. ProceedingsBuilder is an example of a broad class of systems, namely editorial systems that collect content in order to publish it. Our findings are of interest to a broader audience, not only to conference organizers.

1. INTRODUCTION

The chores of the proceedings chair of a scientific conference are extensive: collecting the material from authors, and making sure that it arrives on time; ensuring that the material conforms to the guidelines (layout guidelines in particular), a task we refer to as *verification*; generating additional material, such as cover pages and tables of content. One of the authors of this article has served as the Proceedings Chair of VLDB 2005. Shortly after having been invited to this position, and after discussions with previous VLDB proceedings chairs, it occurred to us how labor-intensive the proceedings-production process actually is. This has led to the idea of designing and implementing a system that helps the proceedings chair of a scientific conference to carry out his chores.

The resulting system, *ProceedingsBuilder*, has both features of workflow management systems (WFMS) and content management systems (CMS). Collecting the material as well as verification are workflows that involve several participants. Participants include the authors of research papers, with a differentiation between contact authors and other authors, the authors of other contributions such as invited papers, panel moderators etc., the proceedings chair, and his helpers. The system has CMS features as well. A CMS models and supports the content life cycle, including creation and publication of content. ProceedingsBuilder covers the phase of the life cycle where content is collected from authors.

ProceedingsBuilder has worked well and without any disturbance at VLDB 2005. The proceedings-production process took place in May and June 2005. There have been 466 authors with 155 contributions. By now (March 2006), we have deployed ProceedingsBuilder at other conferences, namely MMS2006, a German conference on mobile information systems, and (some of) EDBT 2006. Originally, we intended to use ProceedingsBuilder as a showcase for WFMS and CMS technology, and how to combine their advantages. We had hoped to be able to demonstrate, by a rigid assessment of user interactions and by comparisons to other conferences where the proceedings chair does not use a system yet, that such technology incurs significant productivity gains. However, in spite of a thorough analysis of the application domain, our modeling of the processes has not always been adequate and has been too undifferentiated at certain points. The reason is that it is difficult to impossible to anticipate all eventualities. This is not due to any slackness from our side, but is a general problem. The following example illustrates unpredictability. *Example: One author had passed away before the deadline for camera-ready copies. ProceedingsBuilder kept indicating to the proceedings chair that this author had not yet confirmed the correct spelling of his name and affiliation. To ensure progress of the system, we had to solve this situation by hand.* Further, adaptations may be complex and extensive. *Example: Local conference organizers had asked us to use ProceedingsBuilder to collect the presentation slides as well. The necessary modifications have been significant. They included the user interface, the various workflows including verification, and the upload functionality.*

Contingencies of this kind had forced us to adapt our system while it was operational. On the one hand, these adaptations went along with productivity leaks. They have prevented us from demonstrating that the technology used is indeed superior. On the other hand, we have had a very intense lesson which requirements regarding adaptivity of

VLDB '06, September 12-15, 2006, Seoul, Korea.

workflows may arise. In a nutshell, the concern of this article is to describe and classify these requirements. ProceedingsBuilder is an example of a broad class of systems, namely editorial systems that collect content in order to publish it. Our experiences are of interest to a broader audience, not only to conference organizers.

The contributions of this paper are as follows: First, we list the different requirements regarding adaptation of processes that have resulted from operating ProceedingsBuilder for VLDB 2005 and from putting the system into operation for other conferences. Second, we propose a classification of these requirements. We see four dimensions of the space of adaptations, namely (1) initiation vs. realization, (2) global vs. local, (3) logical vs. user support, and (4) adaptations resulting from data-workflow relationships vs. adaptations resulting from datatype-workflow relationships vs. independent adaptations. Third, we describe our system and pass on our experiences. An important conclusion resulting from our work with ProceedingsBuilder is that the relationship between data related to the workflow and the workflow itself needs to be investigated in more detail. Fourth, we have examined different WFMS and CMS research prototypes and systems to which extent they fulfill our requirements. We report on the most interesting conclusions resulting from this study.

2. PROCEEDINGSBUILDER – OVERVIEW

ProceedingsBuilder is a system that manages the conference-proceedings production process. ProceedingsBuilder comes in after author notifications – the point of time where conference management tools typically stop, and thus complements these tools well. It goes much beyond a simple upload feature that some conference-management tools now offer.

2.1 ProceedingsBuilder – Features

Guides verifications at fine detail. For each conference, there is a list of verifications which need to be carried out for each contribution. Verification means ensuring that the material is complete, correct, and that it has the right format. Examples of completeness are as follows: The authors have faxed the copyright form, they have provided all author information (e.g., affiliation, country). An example of correctness is that the spelling of an author name and affiliation is correct and consistent. Layout verification for VLDB includes the following points: the abstract for the conference brochure must not be too long, the paper is in two-column format and does not exceed the maximum number of pages allowed.[1] For each property that needs to be verified, there is a checkbox as part of a browser screen. The person carrying out the verification must tick the checkbox if the particular property is not met. The system then notifies the authors per email. The system carries out any bookkeeping automatically. The list of properties that need to be checked as part of verification can be easily extended at runtime. This is because we did not know all faults beforehand, in spite of much valuable feedback from previous VLDB proceedings editors.

[1]While most of the verifications can only be done by a human, some might be automated. But this has not been our concern – we have carried out all verifications by hand. We do not expect any difficulties when one wants to integrate implementations of verifications into ProceedingsBuilder.

Handles author communication. ProceedingsBuilder automatically handles the part of the communication that is predictable. This includes reminders to the contact author, reminders to all authors if the contact author does not respond after a certain number of reminders, and confirmations.

Lets authors do the corrections. Spelling errors in names are irritating, and they keep occurring in conference proceedings, in tables of content or elsewhere. ProceedingsBuilder asks authors to enter/correct such data themselves. This not only shifts the responsibility to authors. Email messages asking authors to enter their data are logged (as is any interaction). The proceedings chair can now document that he has carried out his duties. Another advantage of this feature is that it means less work for the proceedings chair.

Eases spontaneous author communication. To specify the recipients of unforeseen email messages without difficulty, ProceedingsBuilder allows to formulate queries against the underlying database schema, to flexibly address groups of authors. Of course, one must know the database schema. However, there are only 23 relations, and our experience has been that formulating such queries is easy.

Supports delegation of work. ProceedingsBuilder provides around a dozen of user roles, described in Subsection 2.2. From the perspective of the proceedings chair, it is important that the system supports delegation of layout verifications. The system sends an email message to a helper, with the URL of the page where to enter verification results.

Lets organizers view current status of publication process from many perspectives.

Finally, there are two further arguments in favor of ProceedingsBuilder. First, it is particularly helpful when there is more than one product to build and more than one item to collect per contribution. In our case, the products have been the printed proceedings, CD, and conference brochure. The items have included the camera-ready article in pdf, the abstract in ASCII (for the brochure), the copyright form, photo and short biography of panelists, and the correctly spelled name and affiliation of each author. We refer to the last kind of item as the *personal data of an author*. Second, products have turned out to be of high quality, and verifications typically have taken place right after the upload. Compare this to the nuisances of a late 'bulk verification' only when almost all contributions have been uploaded.

On the more technical side, ProceedingsBuilder expects XML files as input, in particular one containing the list of authors and their email addresses. A conference-management tool such as that from Microsoft Research can generate this without difficulty.

2.2 Roles and System States

ProceedingsBuilder provides different user roles. These include authors of the different categories (Research, Industrial&Application, etc.), conference organizers, the proceedings chairs, helpers, secretaries, system administrators, and

observers. Conference organizers are individuals who must provide information needed for the printed proceedings (e.g., forewords of the various chairs) or the conference brochure (e.g., description of conference venue). The proceedings chair and the administrators have all system privileges, e.g., adjusting system parameters such as number of reminder messages sent out, or entering new helpers. Helpers can only carry out the verification chores. They do not have any other privileges. Observers are individuals who participate in the organization, e.g., PC chair. They can view the current status of the production process.

An item goes through different states:

Incomplete. The item is still missing.

Pending. The authors have uploaded the item, and it needs to be verified.

Faulty. The item has not passed verification, and a new one has not arrived yet.

Correct. We have received the item and have verified it successfully.

Figure 1 is a screenshot of ProceedingsBuilder, showing the status of one particular contribution. The four different symbols on the left-hand side correspond to the different states, the checkmark to 'correct', the magnifying lens to 'pending', the pencil to 'missing', and the cross to 'faulty'. Figure 2 is a screenshot of ProceedingsBuilder, showing some of the contributions with the overall state of each contribution.

2.3 Workflows in ProccedingsBuilder

ProceedingsBuilder exhibits WFMS functionality. The two most important processes are the *verification workflow* and the *collection workflow*. The verification workflow models the verification process, the collection workflow models the process of reminding authors. Figure 3 graphs a simplified version of the verification workflow. As a result of a verification, the system sends email to the authors, be it to confirm that everything is OK, be it to inform them that an item has not passed verification. The system also sends an email message to a helper once an author has uploaded an item that needs to be verified. More specifically, ProceedingsBuilder sends out such messages at most once per day per recipient, listing all items that need to be verified (not shown in the figure). Note that verification consists of several activities that are specific to the material to be verified. To avoid clutter in the figure, there is only one verification activity depicted.

The collection workflow in turn (not graphed here for lack of space) works as follows: ProceedingsBuilder sends reminder messages to authors if an expected interaction has not occurred for a certain period of time. The first n reminders go to the contact author, the next ones to all authors. The verification workflow features a similar 'escalation strategy' (not shown in the figure): If a helper does not react after a number of messages, the next message goes to the proceedings chair. Both workflows are heavily parameterized, e.g., period of time between reminders, their number n, etc.

2.4 Implementation of ProccedingsBuilder

The following requirements are fundamental and have determined the system architecture: the user interface needs to be web-based, workflow support is mandatory, and documents play a prominent role in the system. Initially, we had intended to use a CMS to realize our system. However, an extensive analysis of technology available has shown the following points: CMS are not as flexible as WFMS when it comes to process modeling. The predefined workflows that are part of CMS are not sufficient for our particular problem domain. From a slightly different perspective, CMS are 'too document-centric': processes are always related to documents and cannot be freely defined to support the processing of arbitrary data objects. They also do not support explicit relationships from activities or (sub-)workflows to time well enough. Finally, installing and administering the CMS has caused an enormous effort. We ended up with an implementation of ProceedingsBuilder based on MySQL, and we have implemented the dynamic aspects, the workflows, and the web user interfaces with PHP scripts. There are about 12000 lines of code. The database schema consists of 23 relation types with 2 to 19 attributes, 8 on average.

2.5 Operating ProceedingsBuilder

The proceedings production process for VLDB 2005 started on May 12th 2005 and ended on June 30th. More specifically, it has been only the production process for the 123 contributions from categories Research, Industrial&Application, and Demonstrations that started on that day. We received information about the remaining contributions (workshops, panels, tutorials, keynote speeches; 32 contributions altogether) later on June 9th. The total number of authors was 466. The deadline announced to the authors for categories Research, I&A, and Demonstrations was June 10th. Authors have received 2286 emails. This includes 466 welcome emails, 1008 notifications regarding the outcome of verifications, and 812 reminders.

Using reminders, we tried to influence author behavior. We expected most author activities to take place just before the deadline. For various reasons, we had a preference for obtaining the material earlier. Figure 4 gives an overview of the author activities and reminders of the system. We had configured the system so that it sent out the first reminders on June 2nd. The number of messages generated on that occasion was 180. On the next day, 185 transactions took place. Compared to the day before, the number rose by 60%. On the next day, without reminders, there were only 51 transactions. Both days were workdays. On the following days, subsequent reminders stimulated author activity significantly. June 4th is an exception, probably because it was a Saturday. Probably due to the reminders, we could collect 60% of all items during the nine days following the first reminder and almost 90% of all material on June 10th.

3. WORKFLOW ADAPTABILITY – REQUIREMENTS FROM PROCEEDINGSBUILDER

When operating ProceedingsBuilder at VLDB 2005, many kinds of adaptations of the system and the workflows have become necessary. WFMS already provide adaptation mechanisms, and there exist various approaches and classification schemes [13]. These kinds of adaptations have also occurred

Figure 1: Screenshot of an individual contribution of VLDB 2005.

with ProceedingsBuilder – see Subsection 3.2. However, ProceedingsBuilder has also indicated many new requirements beyond existing WFMS support, to be discussed in Subsection 3.3. In the following, we will also combine requirements to groups. To ease presentation, there will be a capital letter for each group.

3.1 Classification of New Requirements

A workflow type specifies the arrangements of activities allowed. By creating one or several instances of a workflow type, operation starts. A workflow instance consists of activity instances that contain information about the current state of the workflow instance.

We see four important dimensions of the space of adaptations, namely (1) initiation vs. realization, (2) global vs. local, (3) logical vs. user support, and (4) adaptations resulting from data-workflow relationships vs. adaptations resulting from datatype-workflow relationships vs. independent adaptations. Dimension 1 is the extent to which the adaptation is supported, i.e., 'the change is initiated' vs. 'the change is realized'. Dimension 2 stands for the following differentiation: Some participants are tied to one or a few particular activity instances, so-called *local participants*, other participants have a perspective on all workflow instances of a certain type, so-called *global participants*. Think of the verification workflow. An example of the first class are authors, examples of the second one are the proceedings chair and his helpers. We differentiate between two kinds of changes, those initiated or carried out by a global participant (*global case*), and those initiated or carried out by a local participant (*local case*). Regarding Dimension 3, the logical perspective relates to the space of modifications of the structure of workflow types and instances that are feasible. This dimension stands for the degree of user support in carrying out any changes. Dimension 4 differentiates between adaptations which result from data and datatype-workflow relationships where data or data-type changes trigger or guide changes of the workflow vs. adaptations which are independent of the data. One could also perceive the common differentiation between the functional and the behavioral perspective of a workflow as a fifth dimension. The functional perspective covers the means of structural composition of workflows from activities and subworkflows, the behavioral aspect describes the causal and temporal relationships between subworkflows. However, this differentiation is more basic than the dimensions described here and does not specifically aim at adaptations.

3.2 Adaptation with Existing WFMS

This section describes adaptations of the workflow that have become necessary with ProceedingsBuilder, and respective adaptation functionality is part of existing WFMS or research prototypes. In general, a WFMS eases adaptations by separating workflow definition and program code. This is because the process flow is explicitly specified in a workflow definition language and is separated from application-programming code. *Example: Initially, there has been one workflow type for both refereed papers and invited papers. But invited papers have other requirements, e.g., uploading*

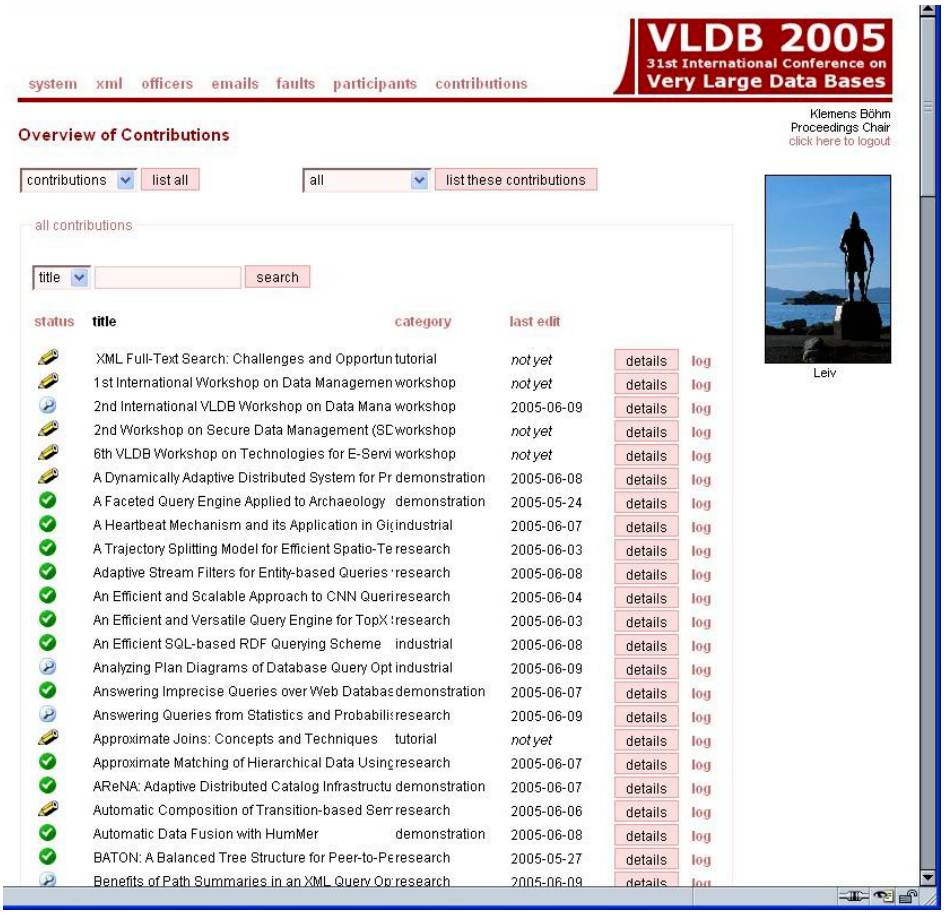
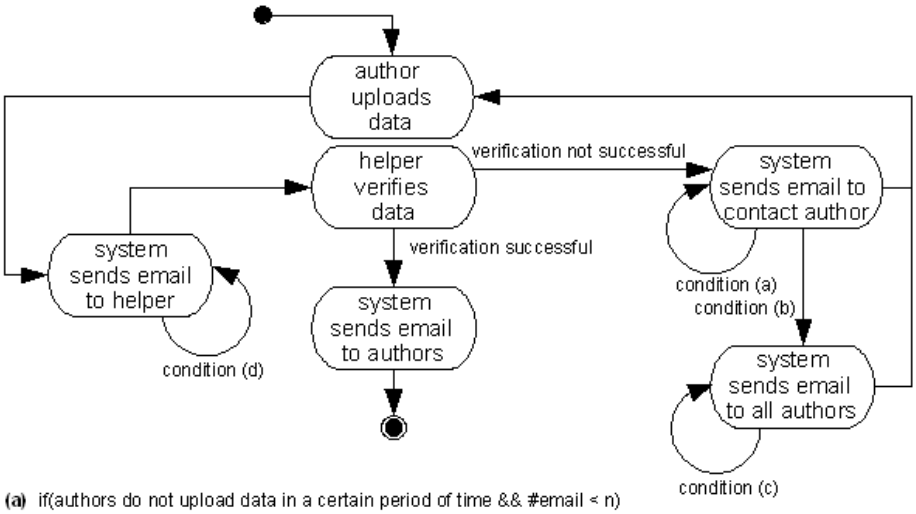


Figure 2: Screenshot of the list of contributions of VLDB 2005.



(a) if(authors do not upload data in a certain period of time && #email < n)
(b) if(authors do not upload data in a certain period of time && #email > n)
(c) if(authors do not upload data in a certain period of time)
Note: after a number of emails to all authors of a contribution without any reaction, the system notifies the proceedings chair by email (not shown in the diagram).
(d) if(helper does not verify in a certain period of time)

Figure 3: Verification workflow (simplified).

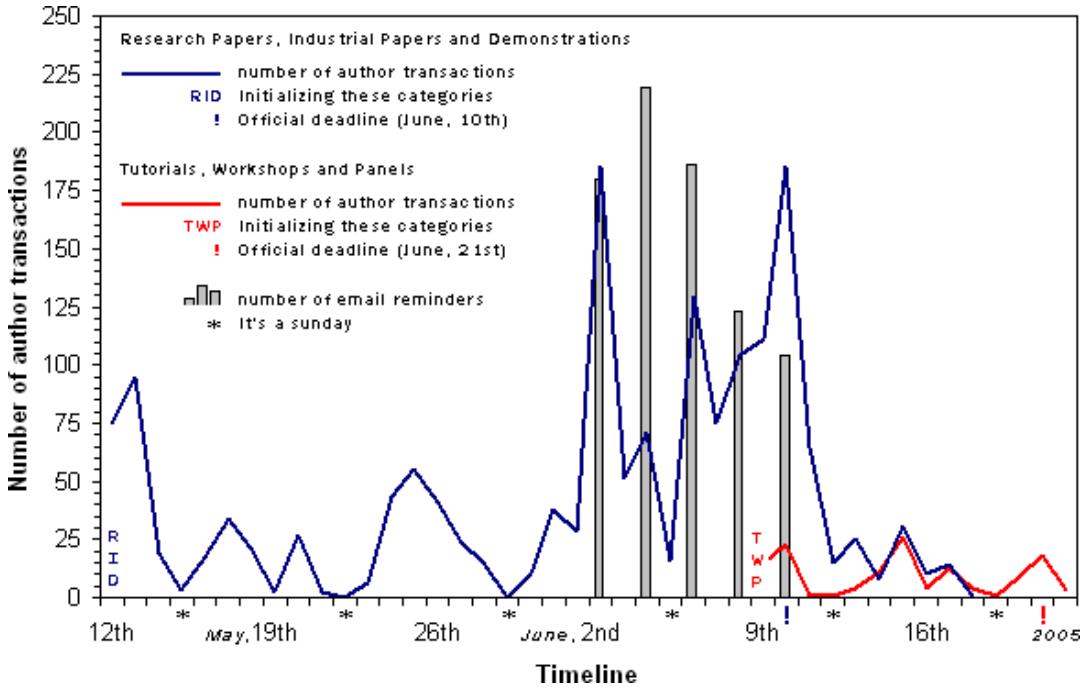


Figure 4: Reminders influence author behavior.

an article for the proceedings is optional. Unfortunately, we had not been aware of this a priori. The necessary change is an additional branch in the workflow type definition. No program code needs to be changed. Adaptations concerning the workflow definition typically require adaptations of the user interface as well.

We now describe adaptations covered by existing systems. Letter S identifies this group of requirements.

S1 – Explicit References to Time. Explicit references to time have been important with ProceedingsBuilder. *Example: When operating ProceedingsBuilder, we have become somewhat anxious at the beginning of June, and we decided to have more reminders, i.e., in shorter intervals, than originally intended.* Defining time dependencies and initiating time events periodically must be possible. One also wants to define time constraints on a set of activities. This is typically done by defining a subworkflow and assigning it a time constraint. *Example: In ProceedingsBuilder, helpers should verify material within a certain timeframe. Thus, the subworkflow for article verification is restricted to that period of time.* Clearly, these requirements are conventional.

We now turn to adaptations specific to design time and runtime, respectively.

Adaptations of ProceedingsBuilder at design time take place when preparing for other conferences. To anticipate most of the necessary changes, as we had hoped, there are many configuration parameters, e.g., number of reminders. But this has not been sufficient.

S2 – Material to Be Collected May Change. Changes regarding the categories of contributions and the items they consist of have turned out to be necessary. *Example: Contributions to MMS 2006 were either full papers or short papers, there have not been any other categories. The layout guidelines have been different as well. For EDBT, we had been asked to let ProceedingsBuilder collect only some of the material.* The fact that the material to be collected may change is a data-dependent change to the workflow. It has been relatively easy to handle, because activity instances do not need to be migrated. The necessity of data-dependent changes at runtime has also arisen, as described later (Requirement D2).

Changes at runtime are well-known, at least the global case regarding the arrangement of activities. These changes have also become necessary with ProceedingsBuilder.

S3 – Insertion of Activities. *Example: With ProceedingsBuilder, authors initially could not change the title of their contribution. We thought that the proceedings chair should do this to keep things under control. We changed the first few titles by hand, after authors had sent email. However, this change request has become too frequent. Therefore, we inserted a respective activity into the workflow and adapted the user interface.* Note that insertion is not limited to a single activity, but also extends to subworkflows.

S4 – Back Jumping. It should be possible to undo activity instances that are already finished. *Example: We verified most of the information uploaded, but not the personal data of authors. The process was designed such that we could not reject a modification of personal data. However, there have been situations where this has been necessary. For instance, some authors provided a 'very sloppy' abbreviation of their affiliation, or the affiliation on the paper was not identical to the one in the system.* To allow rejecting modifications

of personal data required a change in the workflow. We realized a reject by inserting a new verification activity and conditionally jumping back to the step where authors have to upload their personal data, together with an email message. The condition uses a workflow variable which contains the result of the verification.

3.3 Adaptation – New Requirements

We group the new requirements in four categories, which we describe subsequently.

A runtime changes of workflow types and instances without reference to any data

B changes initiated by local participants

C user support for workflow adaptation

D adaptations resulting from the relationship between data and workflow structure

This classification is not trivial. Categories should be somewhat general, and they subsume several cases that might be unrelated at first sight.

A: Runtime Changes of Workflow Types and Instances Without Reference to Basic Data

Adaptations in this category cover the global case along Dimension 2, the logical perspective regarding Dimension 3, and the data-independent case regarding Dimension 4. In contrast to Subsection 3.2, existing systems do not always reflect these requirements.

A1 – Insertion of Activities in a Workflow Instance. It may be necessary to insert an activity, but only into selected workflow instances. This is because the change only applies to a few instances and should not go to the type level because of its exceptional nature. *Example: Helpers have to verify contributions. In some borderline situations, the helpers have been unable to carry out the verification, and they wanted to pass it on to a more knowledgeable person such as the proceedings chair.* To avoid doing this outside of the system, a new activity needs to be inserted. However, delegation should be an exception – the helpers should do the verifications. The example requires a migration of active workflow instances.

A2 – Abort of an Instance. A workflow instance may need to be aborted. There may be relationships between this instance and other instances. *Example: The reader might find it hard to believe, but in one case authors have withdrawn their paper from VLDB 2005 after acceptance. We had not foreseen this with ProceedingsBuilder. At first sight, one should just abort the respective instances of the collection and the verification workflow and delete the authors. However, things are not as easy as they may seem. This is because some of the authors have been authors of other papers as well, and must remain in the system.* The design pattern "abort of a case/process" [18] allows to delete workflow instances, but ensuring that only the right authors are deleted would require programming work. Further, one cannot simply assume that all elements depending only on the instance to be deleted should be deleted as well – this is application-specific. In other words, there is no generic solution which could be specified in advance. A CMS does not offer this functionality either: Workflows in CMS model the life cycle of documents. If a document is deleted, the corresponding workflow instance is deleted, too. However, the problem with the authors of several contributions would be there as well.

A3 – Changing Groups of Workflow Instances. We have motivated both changes to all instances of a workflow as well as changes to individual instances. Both might be too narrow – one might want to change workflow instances with a certain characteristic. *Example: ProceedingsBuilder collects contributions of different categories. Initially, there has been only one workflow type for all categories. But we had been informed after some time that the material for the brochure is only needed later than that for the proceedings.* A simple solution at the type level has not been possible, because only some of the workflow instances are concerned. A solution is to group the workflow instances and to adapt the instances per group. I.e., it should be possible to define a new workflow type and to migrate the instances in a group.

B: Changes Initiated by Local Participants

Letting an administrator carry out changes of workflows using an administration interface is not sufficient. Namely, local participants frequently observe the necessity of adaptations and should at least be allowed to initiate changes. The distinctive features of this group are a focus on local participants (Dimension 2) and data independence (Dimension 4). Regarding the first dimension, the adaptations described here cover both alternatives, initiation and execution by local participants. The advantages of local execution are higher flexibility and less work for the administrator. But this goes along with a loss of control.

B1 – Insertion of an Activity by a Local Participant. A local participant may wish to add a new activity to a workflow instance. *Example: Originally, all authors could modify personal data of any co-author of their contributions, e.g., correct spelling errors. In one case, a co-author corrected the name of another author (inserted a middle initial), this author then set it back, but the co-author 'corrected' it again! We think that an author should have the right to decide on the spelling of his name. One solution could be that an author inserts an activity at the end of the workflow, to check that his name is spelled correctly.* Such an adaptation only affects a specific workflow instance. The difference to A1 is that there must be a local participant who initiates or executes the insertion. It is directly related to the author carrying out the change.

B2 – Change of Data Structures by Local Participants. ProceedingsBuilder has shown to us that changes of data structures by a local participant would yield a higher level of flexibility. *Example: In some parts of the world, e.g., parts of Southern India, persons have only one name, as opposed to a first name and a family name. We had not been aware of this. A meaningful adaptation carried out by the respective persons themselves would be the insertion of a new attribute saying how a person name should appear in the conference proceedings. If this field is empty, the*

'usual' combination of first name and family name would appear. More specifically, local participants, i.e., authors in this case, should be in the position to carry out such modifications themselves. The example illustrates that modifications of the data structures by local participants may be meaningful. While the adaptation suggested here is powerful, it is unclear how its realization could look like. It would incur significant modifications of various system layers. In addition, changes by local users must take place in a controlled manner, see Category C.

B3 – Local Participants May Need to Modify Access Rights. Modifying access rights to workflow activities is a meaningful kind of change by local participants. *Example: Remember the example from B1 ('co-author has undone correction of personal data'). A co-author should not be allowed to change the personal data of the author once the author himself has confirmed it.* We see two further ways how to solve this. One is to introduce a workflow variable defining a conditional branch in the workflow. An alternative would be withdrawing the access right for the respective change activity. More specifically, the local participant should be allowed to change the access rights. It is unclear which solution is superior. This needs further investigation. A potential extension that goes even further is to differentiate within the workflow instance. To continue the example, Author 1 may want that another author must not correct his name any more, Author 2 in turn does not care for this restriction. Of course, all this requires enhanced workflow definitions. With that second alternative, activities would have to be entitled to change access rights for activities.

B4 – Local Participants May Need to Change Roles. Roles control the activities an individual is allowed to carry out. If local participants could change roles, this would add much flexibility. *Example: The role of contact author has been assigned at the beginning, and ProceedingsBuilder did not offer the option of reassigning it. This has turned out to be too restrictive. Further, the authors should be able to change this themselves.* This anecdote illustrates the necessity of adapting roles. The user interface should offer a feature that lets authors do this. We addressed this specific situation in ProceedingsBuilder with a hard–coded mechanism. But we would definitely like to see more generic solutions.

Summing up, Group B aims for better support for adaptations by local participants. It is not only important that the system provides mechanisms for adaptations initiated and carried out by workflow users, but also supports them in deciding which changes are useful and result in a consistent workflow. On a more abstract level, the adaptations indicate that workflow changes could again be modeled as a workflow. This workflow specifies change options and restrictions. A change option could be how many participants have to confirm a proposed change, and if they have to do so subsequently or in parallel, to give examples. We for our part think that such an explicit modeling of a change workflow would result in a more structured procedure and beneficial.

C: User Support for Workflow Adaptation

The previous subsections have discussed possible adaptations on a logical level. This group in turn focuses on the realization, i.e., how to support the user and to provide better control over adaptations. The distinctive feature of this group is that it covers the user support perspective of Dimension 3.

C1 – Defining Invariants of Changes – Fixed Regions. Specifying that parts of the workflow may not be changed is a necessary feature. This is obvious in the case of local participants. But it is also helpful for global participants, as an integrity constraint. *Example: As usual, authors must sign a copyright form for their article. Verification includes ensuring that its text has not been modified. Otherwise, ProceedingsBuilder notifies the authors, and the process is repeated. Clearly, authors should not be allowed to change or delete this part of the workflow.* It may be necessary to define parts of the workflow as a fixed region. To realize fixed regions, it should be possible to couple activities with the access-right model.

C2 – Hiding Workflow Elements with Dependencies between Elements. In some situations, an activity or a subworkflow has to be suspended for a certain time, but everything else should go on. *Example: The personal data of authors contain an attribute 'affiliation', to be provided by the authors. We ended up with many different versions of the same institution, e.g., 'IBM', 'IBM Almaden', IBM Alamden', 'IBM Research', 'IBM Almaden Research Center', and many more. In one case, it took the proceedings chair a couple of days to find out the official name of the institution. During that period of time, the helpers should not verify any of the affiliation names in question; this should be deferred.* At first sight, a solution is to hide/temporarily delete the activity of those instances concerning authors with affiliation names in question. However, there are some gory details which complicate the situation: The system should not send any emails asking the helpers to carry out tasks that are currently hidden. But once the activity is not hidden any more, the system should send out such a message. Speaking more generally, hiding activities would be easier if the system was able to identify dependent activities. It would hide these activities as well.

C3 – Support for Informal Collaboration on Top of Workflows. Informal collaboration is somewhat uncommon in WFMS and is not exactly in line with the philosophy of explicitly modeling all processes. However, ProceedingsBuilder has indicated that support for informal collaboration on top of workflows may be worthwhile. *Example: Remember the problem that we ended up with different names of the same institution. We therefore started some data cleaning by hand. The real problem now was that one author explicitly requested a variant of the affiliation name that was different from that of authors of another group from the same institution, to express that the groups are independent. The proceedings chair had to remember this exception, and he had to inform his helpers about it by email, i.e., in a way outside of ProceedingsBuilder.* Communication channels outside of the system are undesirable. We therefore propose the following solution: It should be feasible to add an optional annotation to each basic element, in this case the affiliation

names in question. These annotations would be displayed every time the system displayed or processed the element. In the example, the annotation would read 'Author explicitly requested this version of affiliation.'. Helpers etc. would know that they are not supposed to clean this data item, and they would learn about this exactly when being about to touch the item. This would require changes at different system layers, including the underlying data structure, as well as changes of the workflow, e.g., inserting activities to process the annotation, and at the user interface. However, such mechanisms might be quite powerful; they might also help to solve the situation illustrated in B2.

D: Adaptations Resulting from the Relationship between Data and Workflow Structure

A significant number of modifications in ProceedingsBuilder occurred because of data-workflow interdependencies. There also exist relationships between the type of the data processed by the workflow and the workflow structure. These requirements are orthogonal to the ones from Categories B and C. Regarding Dimension 4, we look at adaptations resulting from data-workflow and datatype-workflow relationships.

D1 – Fine-granular Access to Data Elements. With ProceedingsBuilder, the necessity of fine-granular access to data elements out of workflows has become evident in many situations. *Example: ProceedingsBuilder has sent notification emails to authors, once a helper has verified a modification of their personal data. But this is too verbose: Think of an author or co-author who corrects a phone number. Verifying this information and, in particular, sending email that we have verified it simply is a nuisance. On the other hand, if an author has changed an email address, there should be a notification.* It should be possible to access and connect data elements to workflows in a fine-granular manner.

D2 – Insertion of Data Items and Attributes. Often the insertion or modification of a data item or attribute makes a change of the workflow necessary. *Example: While our work had already been in progress, the publisher of the printed proceedings informed us that the authors had to provide their paper not only as pdf. They also wanted the sources, together with the pdf, as a zip-file.* Changing the format of data items or allowing for new formats results in many changes to the system: new error messages are necessary, the user interface has to be adapted, there need to be activities to upload and read data in the new format. Ideally, the system should be able to carry out such workflow changes automatically, or should 'at least' propose them to the user. Automatic refinement of the user interface is desirable as well. We want to generalize this requirement as follows: Data-type evolution should guide workflow adaptation. For instance, if data types form a generalization hierarchy, the specialization of a data type will entail a refinement of the related workflow or of its activities.

D3 – Execution of an Activity Depends on Data Values. *Example: ProceedingsBuilder sent email to an author whenever his personal data had been filled in or had been corrected, be it by the author himself, be it by one of his co-authors. This was sometimes confusing for the author – an author who has not yet logged into the system does not need to be notified about any change. Further, there are other, more complex reasons why ProceedingsBuilder should not send a notification email to an author after his personal data has been modified.* In general, the execution of an activity (a notification email in our example) may depend on conditions defined over data elements. In the example, the condition would be complex, it would refer to the data element "logged-in" for the author as well as to other attributes. With existing WFMS in turn, data that controls a workflow is limited to workflow variables or input and output parameters of activities. The use of content elements is restricted with workflows in CMS, e.g., they only allow to use data of the document routed and not of child nodes or parent nodes in the document hierarchy. ProceedingsBuilder demonstrates the necessity of formulating conditions based on any data. This would be much more direct and more powerful than defining workflow variables.

D4 – Changing Data Types to Bulk Data Types. An activity may operate on data of a specific type. In some situations, it is necessary to replace a data type by a corresponding bulk data type, and the workflow needs to be adapted as well. *Example: Authors may upload one version of their article at a time. It becomes part of the proceedings. When ProceedingsBuilder was operational, we wanted to change this: It should be able to administer not only one, but up to three versions of an article, and the most recent version would go into the proceedings. In other words, the type is changed from 'article' to 'list of articles'.* The system should support the users by proposing changes of the part of the workflow that operates on instances of the refined type. In the example, the transition from 'article' to 'list of articles' may entail insertion of a loop into the various workflows and extensions of the user interface, i.e., the user gets to choose between the versions whenever necessary.

We think that WFMS should support these relationships much more as is currently the case. We discuss this issue in the following section.

4. SUPPORT OF WORKFLOW ADAPTATIONS WITH EXISTING SYSTEMS

This section compares process adaptation in existing systems to the requirements from Section 3. Most existing approaches still react to changes in the application environment with adaptation of the workflow structure and new workflow types, with no effect to running instances. This is referred to as *adaptive workflows*. So-called *dynamic workflows*[2] also handle changes of workflow instances. They will support a broader range of applications which require changes of workflows. These changes have varying degrees of dynamics up to ad-hoc workflows. It typically has to be refined during execution, because the workflow is only to some part specified in advance. Application areas in the literature are manifold, like workflows for cooperation support [16], scientific workflows [17, 3], and inter-organizational workflows in service-oriented architectures [21]. Service-oriented architectures also promise high flexibility. But workflows in

[2]Sometimes also called: the dynamic case of workflow adaptation or workflow schema evolution, which results in migrating the instance schema to the new workflow schema, too. ([6])

these architectures aim at highly distributed systems and at flexibility by dynamically finding and binding services. Those mechanisms address different problems than ours, in a distributed way. Adaptability is fundamental for business process management and specifically for workflow management systems [7, 9].

The first group of requirements, i.e., Category S, with global changes of workflow types and instances, independent of the data, are subject of many approaches, e.g., ADEPT [11], Breeze [14], Flow Nets [8], MILANO [2], TRAMs [10], WASA2 [20], WF-Nets [19], and WIDE [5]. Changes like insertion, reordering, and deletion of activities or insertion of parallel branches are well understood. Changes in loops, forward and backward jumping at design time and runtime are possible while guaranteeing soundness of the resulting workflow [12, 13].

Because of the prominent role of documents in ProceedingsBuilder, we originally thought that CMS are a good platform for our system. We even realized a variant of ProceedingsBuilder based on the IBM DB2 CMS. However, there is not enough flexibility to cope with the requirements. The most critical point has been that the workflow support in the CMS was not sufficient, because we had to implement changes by programming the workflow logic. An integrated system with both CMS and WFMS functionality might be a solution, but how to combine these types of systems is future work.

Existing approaches hardly support the other requirements.

Group A: Several approaches can handle migration of workflow instances when adapting the workflow type, e.g., [10, 11, 20]. Some of the mechanisms are applicable to instance migration without workflow-type change to some extent as well, as long as the changes are within the same very specific category. This is not the case for A2 and A3. A1 requires ad hoc changes, so that further support to initiate and propose changes should be available. Flow Nets [8] allows to postpone migrations until they become feasible. Breeze [14] proposes to describe complex migration tasks, e.g., with compensation actions and rollbacks, with a graph-based formalism. But how to construct this graph is an open issue.

Group B: The requirements in this group call for mechanisms that allow for dynamic changes of workflow resources like roles, access rights and data elements by local participants. Further, users must be in a position to flexibly observe, initiate or execute changes. WFMS usually do not support this. Groupware or other systems supporting cooperation might be more appropriate in this respect. But such technology would have to be integrated into the WFMS in the first place. [21] proposes the usage of access rights in the context of inter-organizational workflows in service-oriented architectures. But problems occurring in the inter-organizational area are different from ours, and the proposal is not readily applicable here.

Group C: Systems support users in changing workflows at different levels. Design patterns are an important step [18]. They do not only support users in workflow design but also when modifying workflows. In [19] hiding regions of a workflow is a workflow modification that is allowed. But [19] does not consider properties of activities like relationships to other activities, see C2. Providing mechanisms for reuse of process elements may allow to support users to change workflows. [4] and [1] present first concepts in this direction. Approaches that propose the use of automatic planners may provide advanced user support. [15] proposes a planning system to derive adaptations of the workflow structure triggered by external events, like the change of a precondition of an activity. The approach requires that rules anticipating the different kinds of adaptations are given. Hence, the objective of this work is different from ours.

Group D: Process flow and data flow are often handled separately. Most proposals concentrate on modifications of the process flow. ADEPT [11] handles data exchange between activities with the help of global workflow variables, so-called data elements. Its values can be derived from input and output parameters of the activities. These data elements are also used in the change operations proposed in ADEPT, e.g., looping, backward jumping. WASA2 [20] ensures type safety in the presence of adaptations. This a very specific but promising way of supporting the relationship between data types and workflow adaptations.

5. CONCLUSIONS AND FUTURE WORK

The confluence of workflow and content management is a field that bears much practical relevance, e.g., all kinds of editorial systems that collect content in order to publish it. We have designed and implemented a system called ProceedingsBuilder that supports the proceedings chairs of scientific conferences. We have deployed ProceedingsBuilder for the proceedings-production process of several conferences, including VLDB 2005. In a nutshell, this paper is a summary of our experiences with the system. The main issue is that a broad variety of adaptations of workflows has become necessary while operating ProceedingsBuilder. This article provides a systematic description of the various kinds of adaptations that were needed. We also have put much effort into evaluating different WFMS and CMS as to whether they are applicable in our particular setting, and we have reported on our findings as well. As a result, several of our requirements currently are not well addressed, and the field needs further investigation.

There are some important conclusions that we have drawn from our work with ProceedingsBuilder: The relationship between data related to the workflow and the workflow itself needs to be investigated in more detail. A promising approach may be a tight integration of WFMS and CMS technology, in order to enhance adaptability. We think that applications that need adaptations at runtime are the rule – our work with ProceedingsBuilder has taught us that it is impossible to anticipate all eventualities. This calls for enhanced user support for workflow adaptation, but in a controlled manner.

Acknowledgements

We are grateful to Christian S. Jensen, the Program Chair of VLDB 2005, for much support. We thank the VLDB endowment for their goodwill and encouragement.

6. REFERENCES

[1] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Implementing dynamic flexibility in workflows using worklets. In *BPMCenter Report*, volume BPM-06-06. BPMCenter.org, 2006.

[2] Alessandra Agostini and Giorgio De Michelis. A light workflow management system using simple process models. *Computer Supported Cooperative Work*, 9(3/4):335–363, 2000.

[3] Ilkay Altintas, Adam Birnbaum, Kim K. Baldridge, Wibke Sudholt, Mark Miller, Celine Amoreira, Yohann Potier, and Bertram Ludaescher. A Framework for the Design and Reuse of Grid Workflows. In *SAG'2004*, volume LNCS 3458, pages 120–133, Bejing, China, 2005. Springer.

[4] Shawn Bowers, Bertram Ludaescher, Anne H.H. Ngu, and Terence Critchlow. Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow. In *Query Languages and Query Processing (QLQP)*, to appear, 2006.

[5] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual modeling of workflows. In M. P. Papazoglou, editor, *OOER'95*, volume 1021, pages 341–354. Springer, 1995.

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

[7] James Champy. People and process – minimizing the pain of business process change. *acm queue*, 4(2):34 – 38, March 2006.

[8] Clarence Ellis, Karim Keddara, and Grzegorz Rozenberg. Dynamic change within workflow systems. In *COCS '95: Proceedings of the Conference on Organizational Computing Systems*, pages 10–21, New York, USA, 1995. ACM Press.

[9] Y. Han, A. Sheth, and C. Bussler. A taxonomy of adaptive workflow management. In *Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work*, November 1998.

[10] Markus Kradolfer, Andreas Geppert, and Klaus R. Dittrich. Workflow specification in Trams. In *18th International Conference on Conceptual Modeling, ER'99*, volume 1728, pages 263–277, Paris, France, 1999. Springer.

[11] Manfred Reichert and Peter Dadam. ADEPTflex – Supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[12] Manfred Reichert, Peter Dadam, and Thomas Bauer. Dealing with forward and backward jumps in workflow management systems. *Software and System Modeling*, 2(1):37–58, 2003.

[13] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Correctness criteria for dynamic changes in workflow systems: a survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.

[14] Wasim Sadiq, Olivera Marjanovic, and Maria E. Orlowska. Managing change and time in dynamic workflow processes. *Intl. Journal of Cooperative Information Systems*, 9(1-2):93–116, 2000.

[15] Hilmar Schuschel and Mathias Weske. Triggering replanning in an integrated workflow planning and enactment system. In *Proceedings ADBIS*, pages 322–335, 2004.

[16] Basit Shafiq, Arjmand Samuel, and Halima Ghafoor. A GTRBAC Based System for Dynamic Workflow Composition and Management. In *ISORC'05*. IEEE Computer Society, 2005.

[17] Srinath Shankar, Ameet Kini, David J. DeWitt, and Jeffrey Naughton. Integrating databases and workflow systems. *SIGMOD Record*, 34(3):5–11, 2005.

[18] Wil M. P. van der Aalst, Alistair P. Barros, Arthur H. M. ter Hofstede, and Bartek Kiepuszewski. Advanced workflow patterns. In *CooplS'00: Proceedings of the 7th Intl. Conference on Cooperative Information Systems*, pages 18–29, London, UK, 2000. Springer.

[19] Wil M. P. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computing Sciences*, 27(1-2):125–203, 2002.

[20] M. Weske. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In *HICSS'01*, volume 7, pages 7051–7061. IEEE Computer Society, 2001.

[21] Andreas Wombacher. Issues of access control in cross-organizational workflows. In *Proceedings of the Intl. Workshop on Dynamic Web Processes, DWP 2005*. IBM Research Report, RC23822, 2005.