Smart Files: Combining the Advantages of DBMS and WfMS with the Simplicity and Flexibility of Spreadsheets

Alexander Hilliger von Thile, Ingo Melzer

DaimlerChrysler Research and Technology
P.O. Box 2360, 89013 Ulm, Germany
alexander.hilliger_von_thile@daimlerchrysler.com
ingo.melzer@daimlerchrysler.com

Abstract: Even though database management as well as workflow management systems have significant advantages, spreadsheets exchanged by e-mail are still in widespread use for many processes within an enterprise, causing problems such as poor data-quality and lack of process monitoring. This paper analyzes reasons for office-document based workflows (ODBWf) and presents an alternative solution that combines the advantages of DBMS and WfMS with the flexibility and simplicity of office documents.

This paper introduces an autonomous mobile document-management-system based on a 'smart' file. This file is executable and contains a managed resource part where files can be stored. Any read/write access to its resource part is managed by the file itself; it controls who can do what and automatically synchronizes changes to other systems such as process-monitoring or business intelligence tools. Proven concepts from DBMS, such as triggers, integrity constraints and multi-user support are utilized to improve ODBWfs without restraining their flexibility.

1 Introduction

Most processes within the automotive sector span between a changing network of suppliers, are started spontaneously, are subject to frequent changes, and therefore need to be highly flexible which is out of scope of many products on the market today. Furthermore, these processes are executed by non IT experts without deeper knowledge of DBMS or WfMS. For this group of users, the by far easiest way to define a schema and process is by just creating a spreadsheet, assigning names to the columns, and exchanging them via e-mail. In this case, simplicity and cost effectiveness dominates over functionality causing serious problems such as poor data-quality, security problems, lack of process monitoring, and missing concurrent multi-user-support.

Our paper solves this dilemma by combining the advantages of DBMS and WfMS with the simplicity and flexibility of spreadsheets. It introduces an autonomous mobile document-management-system based on a 'smart' file that is responsible for its contents. It guarantees consistency, security, and role-based multi-user support. In addition, it presents a solution to make the process-model and the process-activity-data transparent to modern BI-systems without restraining the flexibility of these office document based workflows.

The content of this paper is organized as follows: in chapter 2 we discuss reasons for the widespread use of office-document-based-workflows, its advantages and disadvantages and alternative solutions, in chapter 3 a new solution based on 'smart' files is introduced and its features are described. Chapter 4 contains a technical view on how smart files can be realized. The paper concludes with an overview of related work and a short summary.

2. Office document based workflows

The vast majority of processes managed with ODBWfs are ad-hoc workflows and processes that are not supported by special applications. Such processes are enacted spontaneously and are only partly specified prior to execution because responsibilities, roles and activities are determined during execution of the process. As a consequence, processes are changing frequently. Moreover, the process is usually infrequently executed, complex, long-running and cross-organizational.

In the following, we want to briefly characterize advantages w.r.t. the user group of ODBWfs and the process characteristics identified above, and describe missing features compared to alternative solutions.

2.1 Advantages

From a technical perspective, listing ODBWf's advantages might seem absurd because disadvantages are predominating. However, for non IT experts in many cases other criteria such as simplicity and flexibility are key criteria. The advantages of ODBWfs can be briefly summarized as follows.

- ODBWfs allow process execution with a minimum-time of preparation.
- The process model is not required prior to process execution.
- Easy adoption to process changes is supported by adding/removing columns of spreadsheets at any time.
- Office documents are easily exchangeable, even between companies.
- Data is available offline (i.e. if networks are inaccessible due to firewalls).
- Spontaneous integration of external partners is easily accomplished because e-mail and office suites are available almost anywhere.

2.2 Missing features

The above advantages are contrasted by a number of disadvantages. In particular, constraint checks, up-to-date data, process monitoring, triggers/alerters, concurrent multi-user support and automatic integration of different versions are not supported.

2.3 Alternative solutions

There are plenty alternatives to ODBWf, therefore we will focus on the most common:

Web based three-tier-applications accessing a RDBMS: Development of such applications requires too much time for spontaneous processes in general, is expensive because the requirements are constantly changing (due to process changes) and require an online connection. *Workflow Management Systems* require the process model, roles and user-privileges prior to process execution. *Groupware and Document Management Systems* are the most promising approach, because office documents remain as a central element, keeping end users benefits. Documents are stored centrally and are therefore available to other systems. Team members are limitedly able to track process-execution by setting notifications on full document updates. Limited support for triggers and alerters is available, and concurrent editing of office documents is not supported. Additionally, these systems either require a client installed (such as Lotus Notes) or are available online only.

3. Smart Files

None of the alternatives above satisfy all requirements; therefore we combined the advantages into a new concept by integrating features of ODBWfs (ease of use, flexibility, offline data availability, no need of a full process model prior to process execution, maintainability by end users), WfMSs (multi-user support such as access-control and synchronized concurrent access, no need to code, process monitoring capabilities), DBMSs (constraint checking, specification of triggers and alerters), and DMSs (file management features).

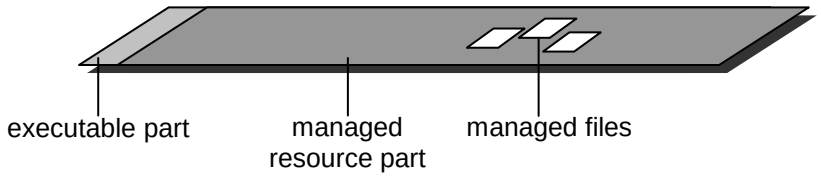

Figure 1: – Smart file concept

The basic principle is based on a mobile document management system consisting of a single file (figure 1) that can easily be exchanged by e-mail. This file is executable and contains a resource part where managed files can be stored. Since this concept enhances traditional file-usage paradigms it is called ‘smart’ file (SF). A SF is a file-container, comparable to self-extracting zip-archives. However, SFs have the advantage that read- and write access within the resource part is possible. Any read/write access is managed by the SF itself, it controls who can do what, when, and where. This concept enables an SF to be autonomous and therefore responsible for its contents which opens great possibilities for consistency and security. SF can be thought of as a mobile database, but instead of relational tables with tuples, SFs use files as granularity. Instead of a query-language like SQL, a file system is emulated allowing SFs to be mounted as a virtual drive. Queries are therefore performed using file-access functions to retrieve (generated) office-documents and changes are stored within the SF and are synchronized with a smart file coordination server as soon as a connection is available. Therefore, all process-data is available to other systems allowing report generation on up-to-date data. Consistency checks can be performed by the SF, too, denying check-in of incorrect managed files. An advantage is that working with SFs looks very much the same as working with zips for end users. Because the SF (a single file) can be e-mailed as easily as any other file, all advantages of office documents are preserved.

4. Inside Smart Files

This chapter illustrates required modules to realize a SF.

4.1 File System

The managed part of the SF must be able to store files and directories, therefore it needs an internal file system (IF). In most common file systems files are stored within a set of clusters. Since meta-data (e.g., for constraints and access permissions) must be supported, fixed directory entries as used in a FAT-filesystem are insufficient. Because file IO inside a SF is low and the meta-data needs to be extensible, XML can be used for directory entries. To avoid loss of data or inconsistencies within the IF, logging and recovery for IF-operations comparable to the respective features found in DBMS or journaling file systems, such as Reiser FS [RFS] are recommended. To be able to store multiple versions of files, the IF has to support a built-in versioning mechanism.

To integrate the IF into an existing file system, several protocols qualify, such as NFS, SMB/CIFS [CIFS], FTP [FTP] or WebDAV [WDav] (used in our prototype). To avoid installation of client-software all features of the file system are available using 'file system functions': Assume the name of a file is */mySheet.xls.* Opening this file returns its latest version. To access other features supported by the IF, the operator "!/" can be appended to the filename. For example, to list all versions of this file, a user can simply "cd" into the virtual directory */mySheet.xls!/versions*. To access the first version of this file, */mySheet.xls!/versions/0* can be used.

4.2 Transaction

One of the key benefits of a SF is that its content is always consistent. This is ensured by integrity constraints allowing valid changes only, others are rolled back. Since a SF could be used by multiple users concurrently, isolation is required, too, and so are the A and the D of ACID [HR99]. However, the standard ACID paradigm known from DBMS is insufficient for SFs because user-TAs (editing of files) are always long-running (human in the loop) and the granularity of files is coarse. Another problem of ACID is the handling of constraint violations: if a constraint is violated within a TA inside a DBMS, its changes are rolled back immediately. However, if a spreadsheet is stored inside a SF, the user won't accept that all of his changes are lost, just because of a single cell containing a wrong value. Therefore, SFs have to allow error corrections detected by constraint violations. This is realized supporting so-called pending transactions. A rollback is executed only if the TA was aborted by the user explicitly.

The following figure gives an overview of the transaction states of a SF.

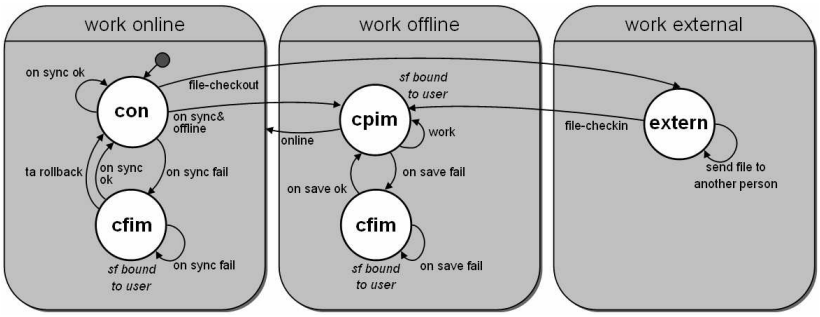

Figure 2: Transaction states

Working with files (using SFs) can be differentiated into three distinct categories: saving a file into the virtual drive of the SF while the SF has a connection to the internet is referred to as *work online*. If no such connection exists, we refer to this as *work offline.* If the edited file is stored outside the SF, e.g. on the home directory of the user, it is out of control of the SF and referred to as *work external*.

Depending on these work categories, some features of the SF, such as constraint-validation, are unavailable. Constraints are differentiated into the following classes: *on-edit* constraints can be checked by applications immediately, e.g. that a cell inside a spreadsheet must be a date. More complex constraints such as multi-cell comparisons would require macros for validation. However, macros are disabled for security reasons in many cases and are not supported by all application used for editing. These constraints can be validated by the SF at the time the file is saved within the SF (*on-save*). Constraints that require a connection to a server, e.g. to validate an address, can only be validated *working online* and are referred to as *on-sync* constraints.

SFs use transactions to ensure data-quality. A SF is always consistent (*con*, see figure 2) w.r.t. specified constraints, such as data inside a DBMS. If a file is saved into the SF all constraint classes are validated. If this validation was passed, the transaction commits and the edited file is written to the SF (*on sync ok*) otherwise (*on sync fail*) the SF changes its state to *checks failed/ isolation mode* (*cfim*). In this state the transaction is still active. Violated constraints are displayed as error-messages to the user and the SF is bound to this user, which means that this file can not be edited by others. This prevents users from exchanging files with pending transactions. The user now has to correct all errors to commit the transaction (*on sync ok*) or can abort the transaction causing all changes to be rolled-back (*ta rollback*). Isolation guarantees that other users can not see uncommitted (inconsistent) changes.

If *on sync* constraints can not be validated because the user is working offline, a SF enters the *checks pending/isolation mode (cpim)*, which indicates that the *on edit* and *on save* constraints were passed and the *on sync* constraints are scheduled to be checked. In this state the SF is bound to the user as well. If *on save* constraints were violated, the *cfim*-state is entered (see above). If the user is *working online* again, pending checks are performed. If those were passed, the transaction commits and the SF is consistent (*con*) again, otherwise the *cfim*-state of *working online* is entered.

Of course, saving a file into the regular file system (*file-checkout*) outside the control of a SF can not be eliminated. Those files are checked for inconsistencies (see above) after *file check-in*.

4.3 User interaction

To allow begin and commit/abort of TAs and to display error messages, a SF needs to interact with users via a GUI. Since all protocols used to integrate the SF into an existing file system do not require a SF to be installed locally, user interaction is also designed to be usable remotely without installation of additional tools. HTML is appropriate but causes two problems: page-flickering because HTML pages are loaded entirely as well as lack of update notifications, because the browser initiates the request to the server. There is no support for a server to contact a browser to send a notification of updates. Therefore, to pop up an error dialog when a TA fails, the browser would have to constantly poll the server.

Both problems are addressed using javascript. Instead of reloading an entire page, only a hidden frame is reloaded containing javascript statements that modify the document object model (DOM) of the visible page for all parts that changed since the last page access. To allow notifications, the same method is used within another hidden frame, but the HTTP-GET request is not answered by the server until the page has changed. Client side connection timeout is handled using javascript as well.

4.4 Meta-data

Meta-data is used to specify integrity constraints within the SF. These constraints are functions that have to evaluate as true, false indicates violation of the constraint. We realized a small web based integrated development environment (WIDE) [HMS04] that can be used by non-programmers, too, because it supports visual programming.

However, the vast majority of users prefer wizards to easily specify constraints. We realized a sample wizard that allows specification of constraints using drag&drop inside an Excel spreadsheet without the need to write any macro: the SF connect to Excel's COM interface. A special XML-stylesheet (XSLT) renders the GUI described above as Excel commands that display the wizard. Therefore, no specific macros have to be programmed and other office products (even by other vendors) can be integrated easily by only changing the XSLT.

4.5 Meta-meta data

DBMS have good support for schema changes, but DBMS are not used directly by end users in general; access to the database is wrapped by applications. Therefore, changes within the process requiring a schema change also require a change of the application as well which are out of scope by end-users.

However, if office-documents such as spreadsheets are used, end users do have the ability to easily change the schema, e.g. by adding a column. This flexibility must be preserved by SF to allow fast adaptation to process-changes. Unlike DBMS, schema changes can not be limited to database administrators. Nevertheless, to still keep control over changes, constraints for the data-dictionary must be definable. This meta-meta-data is stored within a 'data dictionary dictionary' (DDD) as mentioned above. How to specify DDD constraints is currently under research by us.

4.6 Notification

In most ODBWfs, the process model evolves during process execution. Therefore, the process model is not defined prior to its execution but reconstructed from the execution log using methods such as process mining (PM) [Her00]. PM requires a log which is maintained by the SF. Its log-entries store which file has been updated/created/deleted by whom and when. To be able to log and monitor events in more detail, e.g. who changed the value of a certain cell inside a spreadsheet, these events can be subscribed to as well. As soon as this event occurs, the SF notifies the subscriber or logs this event if the SF is used offline. Web Services Notification [GN04] qualified as an appropriate standard for this because it allows an easy, hierarchical definition of events using topic path expressions.

4.7 Smart File coordination server

The SF coordination server (SFCS) is used by the SFs to synchronize their changes and can be used by external tools to monitor the process or to create reports, e.g. to create Business Intelligence reports, SFs can be seen as a distributed logical OLAP-cube. Each SF contains a unique identifier created on a SF's creation (see section 3.1) and the host-name of its SFCS to synchronize its changes. By making a copy of a SF and sending it to a group of users via email the host-name is used to connect to the SFCS and the ID is used by the SFCS to identify the SF's instance and to merge the changes of the split workflow. Since most office files are binary, delta-shipping and integration can not be used for merging changes. We are working on an application-aware mechanism comparable to "operation-based update propagation" [LLS99]. [KS93] gives an overview of flexible and safe resolution of file conflicts using "application-specific resolvers" as used inside the Coda Distributed File System [Br98].

5. Related work

Even though many of the methods used to realize smart files are already in widespread use, their combination is new. Smart files combine aspects of file system integrated document-management-systems, WfMS and mobile DBMS and target the problems of ad-hoc workflows. Therefore many aspects described in this paper are derived from these systems. Creating virtual file systems based on a single file is very common under Linux/Unix environments using loopback-devices. Self-modifying executable files were used in the DOS-era but became rare recently.

The Coda File System [Br98] is a good example of a mobile file system that can be used disconnected and with minimum bandwidth usage; [SK93] gives a comprehensive overview of experiences made with Coda. Handling concurrency is an interesting research topic, especially if data is edited in a disconnected manner; [LLS99] describes operation-based update propagation in a mobile file system which is required to merge changes of binary office documents. If changes cannot be serialized due to cycles within the serialization-graph, conflicts must be handled; [KS93] describes flexible and safe resolution of such conflicts. However, Coda does not support transactions (see section 4.2), and not a single file and therefore cannot be exchanged by e-mail.

Concepts from mobile DBMS, such as consistent data stores using integrity constraints and access control can be utilized directly. To handle storage of files instead of tuples, journaling file systems such as Reiser FS [RFS] and file versioning systems such as the Concurrent Versioning System (CVS) and Subversion provide a good starting point. Reiser FS allows its meta-data to be accessed without additional tools, too, by providing virtual directories and files (see section 4.1).

6. Summary

It is obvious that database management systems (DBMS) and workflow management systems (WfMS) have significant technical advantages over office documents being exchanged by e-mail. DBMS and WfMS both support high data quality by constraint checks and allow reaction to events by triggers and alerters. They contain up-to-date data that can be accessed by other tools, e.g. to monitor the process or to visualize activity data and they support concurrent multi-user access.

In contrast, office documents are easy to create without deeper IT-skills and allow process execution with a minimum time of preparation and no specification of the process model. The documents can be exchanged easily even between companies and allow very flexible processes, because process changes can be adopted fast, e.g. by adding columns of spreadsheets.

We combine these heterogeneous worlds by transferring the technical advantages of DBMS and WfMS to workflows based on office documents without restraining their advantages such as simplicity and flexibility.

A smart file (SF) is a single executable file with an embedded container that can hold other files. All access to its container is managed by the SF itself, allowing it to be responsible for its contents. It has support for long-running transactions and meta-data, such as integrity constraints that can be specified to only allow consistent changes of the data stored inside a SF. SFs contain an internal file system with versioning and event notification support. Events can be subscribed to for external process monitoring purposes. Concurrent multi user access is coordinated by the SF and changes within documents are merged automatically.

Smart files can be used to store office documents. They can be copied and exchanged by e-mail, just like a zip-file and can be used without installation of special software by integrating into the user's file system. Interaction with a SF is as easy as navigating a directory tree. Even complex functions like versioning are accessible without extra tools via virtual directories.

End-users can work with their applications the same way they did before, but with smart files the advantages from DBMS and WfMS are available - even in spreadsheets.

Currently, we are continuing to implement our prototype and are collecting first experiences of SFs being used in productive processes. Due to the fact that already existing documents can be used unchanged, the migration from ODBWf to smart files has been simple and effective.

7. References

[Br98] Braam, P. J.: The Coda Distributed File System. Linux Journal, #50 p. 46-50. June 1998.

[CIFS] Leach, P.; Perry, D.: Standardizing Internet File Systems with CIFS. MIND (Microsoft Internet Developer Magazine). November 1996

[FTP] File Transfer Protocol, RFC 959: http://www.ietf.org/rfc/rfc0959.txt

[GN04] Graham, S.; Niblett, P. and others: Web Services Notification (WS-Notification). January 2004. www.ibm.com

[Her00] Herbst, J.: A Machine Learning Approach to Workflow Management. Proceedings of the 11th European Conference on Machine Learning, p.183-194, June, 2000

[HMS04] Hilliger von Thile, A.; Melzer, I.; Steiert, H.-P.: Managers don't code: Making Web Services Middleware Applicable for End-Users. European Conference on Web Services. Erfurt, Germany. September, 2004.

[HR99] Härder, T.; Rahm, E.: Datenbanksysteme. Springer Verlag 1999

[IFS] Oracle: Oracle Internet Filesystem: http://www.oracle.com/technology/documentation/ifs_arch.html

[Kis93] Kistler, J.J.: Disconnected Operation in a Distributed File System. School of Computer Science, Carnegie Mellon University, May 1993, CMU-CS-93-156

[KS93] Kumar, P., Satyanarayanan, M.: Flexible and Safe Resolution of File Conflicts. Proceedings of the USENIX Winter 1995 Technical Conference. Jan. 1995, New Orleans, LA.

[LLS99] Lee, Y.W.; Leung, K.S.; Satyanarayanan, M.: Operation-based Update Propagation in a Mobile File System. Proceedings of the USENIX Annual Technical Conference. June 1999, Monterey, CA.

[RFS] ReiserFS File System, http://www.namesys.com/

[SC02] Sayal M., Casati F., Dayal U., Shan M.C.: Business Process Cockpit. Proceeding of the 28th International Conference on Very Large Databases, Hong Kong, China 2002.

[SK93] Satyanarayanan, M.; Kistler, J.J.; Mummert, L.B.; Ebling, M.R.; Kumar, P.; Lu, Q.: Experience with Disconnected Operation in a Mobile Computing Environment. Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing. June 1993, Boston, MA.

[SNIA] Storage Network Industry Association: Common Internet File System Technical Reference. 2002, www.snia.org

[WDAV] WebDAV: Web-based Distributed Authoring and Versioning Protocol. http://www.webdav.org/