

Structured Workflow Management with Lotus Notes Release 4

Berthold Reinwald, C. Mohan
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
{reinwald,mohan}@almaden.ibm.com

Abstract

In this paper we describe the design and implementation of workflow management applications on top of Notes Release 4. We elaborate on various design issues for Notes workflow applications and introduce Notes Release 4's native workflow concepts like agents, events, macros, Lotus-Script, OLE2 capabilities, and doclinks, which make Notes a powerful workflow tool. The idea of the paper is the use of the Workflow Reference Model of the Workflow Management Coalition [7] to define structured workflows, and execute these workflows through the exploitation of Notes Release 4's native workflow concepts.

1. Introduction

Workflow management is currently a hot topic in industry and research. More and more products come out claiming workflow management functionality, and customers ask their vendors for this emerging technology. According to recent market forecasts the revenues for pure workflow engines will exceed \$1 billion in the next couple of years. Groupware as the overall technology includes workflow management apart from many other features, although groupware systems mostly only bundle some basic workflow capabilities which can be used to process workflows. This makes it very hard to compare different groupware/workflow products and for this reason the definition of a common Workflow Reference Model by the Workflow Management Coalition (WfMC) enjoys growing interest in industry [7]. The reasons for different definitions of workflow are primarily due to the different origins of the various products. Document and image processing, smart e-mail systems, forms packages, and data sharing systems are typical origins of various workflow management systems [9]. From an application type rather than legacy product type point of view, workflow applications can be classified into production, collaborative, administrative, and ad hoc workflow [23]. Production workflow ap-

plications are characterized by transactional features and are targeted by high end workflow management systems like FileNet/Workflo [6], IBM/FlowMark [10], Staffware, XSoft/InConcert [22, 1] and many others. Collaborative, administrative and ad hoc workflow applications instead require flexible workflow management environments and depend on built-in database and communication infrastructure. This type of integrated workflow management functionality is best provided by groupware systems like Lotus Notes [21]. Whereas the classification into the four workflow application types is useful from a structural point of view, the classification of real applications is fuzzy, and rather represents a continuum as opposed to a strict separation. Different workflow management systems specialize on different parts of the continuum.

Notes has an integrated database and communication architecture. The Notes client serves as a single desktop user interface for various kinds of end-user applications. Notes also provides the basic infrastructure to coordinate, track and manage business processes, and share data in a distributed environment. Notes is able to support the end user in receiving work, e.g., in the form of a document, and acting on it. The system might automate certain steps and perform the routing of the work to the next steps. From this perspective, Notes can be used for a broad range of workflow applications. Typical Notes workflow applications are inquiry tracking at help desks, bug tracking, and travel authorization request applications [20].

Workflow management systems like Action Technologies/WorkManager [16, 2], WorkFlow Inc./FlowMaker, Pavone/GroupFlow [17, 15], and many others [12], have been built on top of Notes. These systems have their own workflow model and they are currently based on Lotus Notes Release 3 technology.

The main idea of this paper is the use of the WfMC workflow model to define structured workflows and the performance of these workflows through the exploitation of Notes Release 4 features. IBM's FDL (FlowMark Definition Language [8]) which is WfMC compliant was chosen as a concrete WfMC workflow model implementation for prototyp-

ing purposes. The WfMC workflow model provides an additional development concept for building and generating workflow applications on top of Notes. The higher level of functionality provided in this fashion should help reduce the work required on the part of the developer for producing structured Notes workflow applications.¹

The rest of the paper is organized as follows. In the next section, we describe a travel request workflow sample application and identify the required functionality to process the workflow. In section 3, we introduce Notes' native workflow concepts which can be exploited for workflow processing. In section 4, we model the workflow sample application in FDL and describe a FDL/Lotus Notes Builder, which makes use of Notes' native workflow concepts. Section 5 summarizes the paper and outlines possible improvements of the pursued approach.

2. Sample Workflow

The definition of workflows roughly involves the definition of (1) data representing the "work", (2) people in an organizational model performing the work, (3) applications employed by the users to execute the work, and (4) the routing of work from work step to work step. In this section, we will first introduce a sample workflow application and then associate the application with the above introduced four workflow elements in order to envision a high level workflow definition.

The sample workflow is a typical Notes travel request application as described in [20, 21]. An employee fills out a travel request form with the travel information and submits the request to the manager for approval. The manager reviews the request and either rejects the request and sends it back to the employee or approves the request and the approved request is sent to the travel bureau. If the trip amount exceeds a certain limit, the manager's approval additionally needs to be confirmed by the manager's manager or by somebody else.

The contents of the travel request form constitute the workflow data which is stored as a document in a database. A workflow is created by the travel requestor who fills out a travel request form, creating a Notes document. With the submission of the travel request, the Notes workflow application resolves the travel requestor's manager and routes the work to that manager. Routing can either mean moving or copying the travel request data to the manager or sending a notification about the travel request to the manager along with a "link" to the travel request data. The pros and cons of moving, copying, or linking the travel request data will be discussed later under the terms "send" and "share

¹ Even though we refer to specific IBM and Lotus Notes products in this paper, no conclusions should be drawn about future IBM product plans based on this paper's contents. The opinions expressed here are our own.

model". The manager acts upon the request using a travel approval form and the workflow management system routes the workflow to the next step depending on the manager's decision.

For the implementation of the above described sample workflow, we need to have a data store for the travel request data, a description of the organizational model to look up the manager information, an interface to perform the work steps, i.e. fill out and submit the travel request form, review and decide on the request or launch an application to perform a budget analysis program, etc., and of course a workflow navigator which routes the work from work step to work step. Additional workflow components are required to support an audit trail, perform periodic notifications for delayed requests, etc.

3. Workflow Concepts in Notes

Notes has a document oriented data model for compound multimedia documents, and a Notes application roughly consists of forms, views, and documents. Documents are stored in Notes databases, views are used to list the documents, and forms are the interfaces to edit the documents. Forms can contain formulae, macros, and agents. The Notes design elements are stored along with the documents in Notes databases.

Notes provides many features to easily manage workflow applications. In this section we introduce these features, in particular we focus on the Notes Release 4 features [11, 13, 14]. We introduce the features in the order of how they can potentially be exploited for the implementation of workflow applications. Agents and LotusScript will be used for routing and possibly work step implementations, the Notes OLE2 support can be exploited for desktop application integration, Notes documents and databases serve as an application data store in a distributed environment, and the Notes address book can be used for the organizational model.

3.1 Agents and LotusScript

Notes supports *agents*, which automatically fire on the occurrence of certain events and perform specified tasks. Agents may be event-driven, are stored in the database, and run in the context of a database, i.e., either on the client or server side. In this capacity, they are quite similar to triggers in active database systems. The Notes agent builder supports users in creating agents using a GUI. Each agent has a name and consists of three components: (1) when to run, (2) which documents to act on, and (3) what to do (see Figure 1). An agent starts when a certain event occurs, queries documents from the database, and performs an action on the qualified documents. From this perspective, agents can be

exploited for various purposes. They can be used to implement the routing of workflows, send periodic notifications to users about outstanding or delayed actions, send mail, lookup information in databases, perform document maintenance tasks, disseminate information quickly and perform all kinds of periodic or non-periodic routine tasks.

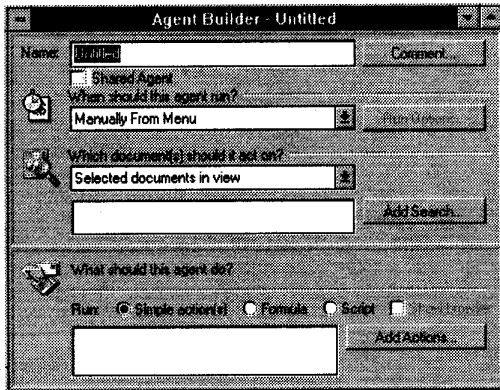


Figure 1. Notes Agent Builder

Notes supports the following options with regard to agent executions: (1) manually-activated by the user from the Notes client action menu or the agents window, (2) change-activated agents, which are invoked when new mail arrives, a document has been modified in the database or pasted, and (3) time-scheduled agents (hour, day, week, month). In addition, agents may be directly executed from actions, macros, and LotusScript programs.

Depending on the event which fires an agent, various document selection criteria apply. These criteria are determined by the search space and the search condition. In the case of a manually-activated agent, the documents in the search space can be the full database, all new or modified documents since the last run of the agent, or unread documents, all documents or selected documents in an open view, or the current document. For change-activated agents, the changed or newly arrived documents are in the search space, and for time-scheduled agents, all documents that are new or modified since the last run constitute the search space. In addition to the above search space, the Notes search builder can be used to further qualify the documents in the search space by defining predicates or conditions which are applied on the documents. Search conditions can be simple expressions on the author, date, or any other fields, multiple fields on a form or words and phrases.

The action part of an agent can be (1) simple actions, (2) a formula or (3) a LotusScript program. Simple actions are actions to manage and modify documents, send documents, execute @function formulae, or run other LotusScript agents (i.e., agents without a “which document” part).

The “run agent” action allows to nest agents within other agents with the semantic that the main agent completes its search and action first, and then passes that information on to the secondary agent. @function formulae require knowledge of the Notes @function language. Simple actions can combine @functions and LotusScript programs.

LotusScript is an object-oriented Visual Basic-like language. It runs on multiple platforms, comes with its own development environment, and has customized functions for Notes products. It provides Notes object class libraries, script-level access to Notes object classes, OLE2 automation support and an integrated development environment to develop and debug scripts. The object class libraries include UI classes, Back-End classes like sessions and databases, and ODBC classes. The script-level access to the Notes object classes allows the development of scripts which can be used by user actions or by agents. The LotusScript objects can run as OLE2 automation servers that can be called from Visual Basic. LotusScript programs can also call OLE custom controls (OCXs [18, 19]).

In addition to the events used to build agents, Notes also supports *user-defined events*, which allow one server add-in task to notify another add-in task that something has happened. User-defined events are used in a producer/consumer manner of processing: an event producer raises an event by placing it on an event queue, whereas an event consumer removes the event from the event queue, acting on it as appropriate. Additionally, Notes supports an *extension manager*, which allows an application DLL to register a callback routine that will be called before or/and after Notes performs selected internal operations. These internal operations act as events in this way.

With the support for events being available in various forms and LotusScript being a powerful development environment, Notes provides a rich infrastructure to implement the routing as well as application invocation in workflows.

3.2 Forms, OLE2, Notes/FX, and NotesFlow

Notes forms are an integral part of a Notes application. Forms are used to display and edit Notes documents. For workflow applications, where documents are routed to many different users (with different access privileges), Notes supports *collapsible sections in forms* that automatically expand or collapse depending upon the user. As a document travels through its workflow, the document can expand and collapse depending upon the current workflow participant and/or the workflow state.

Notes embraces OLE2 and takes a major step towards application integration [24, 4]. The integration of workflow with existing applications is an important property of workflow management systems. It is required for interoperability on control level as well as for data exchange on data level.

Notes documents can also serve as *OLE2 object containers*. This is of particular interest as the workflow data created by any kind of interoperable application (e.g., OLE2-enabled application) can be stored as OLE2 objects in Notes documents and databases. *Notes/FX (Field eXchange)* can be used for bi-directional exchange of data between the contents of embedded OLE2 objects in a document and fields in the document. The fields can in turn be used to build indexes and views on these values and for supporting queries against the contents of embedded OLE2 objects in Notes databases. They can also be used in expressions. For example, depending on the value of the travel request, the request can be automatically rejected or approved.

OLE2 applications can be seamlessly used and launched from Notes documents through *in-place activation*. OLE automation allows the invocation of OLE automation servers from Notes applications, specifically from LotusScript programs. A WordPro document, which was received by an end user as part of a workflow, can easily be edited through in-place activation of WordPro. Notes also supports so-called "stopping points" between launching the OLE server and handing control over to the server. This can be exploited in various scenarios in particular to transfer workflow data to the OLE server.

NotesFlow presents the user an action bar with "push buttons" in a non-scrollable area of a Notes Form. The action bars can also be handed over to OLE2 applications, which are NotesFlow-enabled, to appear in the menus of an application. The actions implemented in the buttons or menus can typically range from simple actions (e.g., close document) to field changes in the document (e.g., approve travel request) and the launching of LotusScript or external applications. These action bars can be used to launch external applications or present workflow routing alternatives to the end user, e.g., reject or approve a travel request.

3.3 Notes Database and DocLinks

Notes has its own *database* with its own document-oriented data model. Notes supports authorization, authentication, RSA-based encryption as well as digital signatures. Signatures are of particular importance for workflows in banks and insurance companies. The Notes database is accessible through its client as well as a C/C++ Notes API, its macro language (@DBfunctions), LotusScript object classes, and NotesSQL, an ODBC-based driver for Lotus Notes to allow SQL access to Notes databases. External data sources are accessible through ODBC-based LotusScript Data Object and Data Access Tools. The Notes database is accessible in a client/server environment and can be replicated in a distributed environment. *Replication* can in particular be exploited for workflow management in mobile environments. The performance of replication was en-

hanced in Release 4 through field-level replication and other Notes server modifications (e.g., multithreaded replicators).

Notes supports *view* and *database links* as well as *doclinks*. Doclinks are links referring to other documents which do not need to be in the same database as the document containing the doclink. Release 4 also provides an OLE link server, which allows OLE container applications to make doclinks.

Considering doclinks and the requirement of passing around workflow data, Notes supports two fundamentally different concepts for the passing of work from one work step to the next: *send model* and *share model*. The two concepts roughly correspond to what is known in programming languages as "call by value" and "call by reference". In the send model, the entire self-contained workflow document is sent to the next step in a workflow. The sending of the document can happen through "@MailSend" in action buttons, field formulae, macros, or SmartIcons. In the share model, a central Notes database stores the workflow documents and only messages with doclinks to the documents are sent to the participants of a workflow. The documents themselves are not physically routed. Both approaches, send and share model, have their advantages and disadvantages. The send model is good for disconnected operations where no direct access to a central database is required. On the other hand, no sub-workflows can be spawned, the workflow can involve only one document, and the document must contain all the workflow data including the workflow logic. In the share model, as commonly implemented, all participants have to have direct access to the required database (or replica copies of it) in order to perform work steps. With direct access, documents which are stored in other databases can easily be incorporated, the workflow can contain sub-workflows and involve multiple documents.

3.4 Address Book

Notes supports a *Name and Address database*, which can be exploited for addressing messages. A company-wide address book contains a document for each employee, including the employee's name, address, and other information. The address book can be looked-up in the Notes mail environment as well as in Notes applications, e.g., through macros. The usefulness of the address book for workflow management and in particular role resolution is limited in the default Notes configuration as it contains only the name of the participant of a workflow but not the organizational model. Information like the name of an employee's manager is not part of the database. On the other hand, the address book is user-extendible and contains some organizational options, which can be exploited to provide some additional functionality.

4. Implementing Workflow with Notes

In this section, we show the use of Notes to implement workflow applications. As Notes does not provide a workflow modeling tool, we use the FlowMark buildtime client to design workflows. We extend the FlowMark FDL compiler with a FDL/Lotus Notes build component in order to generate Notes workflow applications.

4.1 Modeling Workflows with FDL

Figure 2 shows a graphical representation of the sample workflow application of section 2, as modeled with the graphical FlowMark buildtime client. The figure shows nodes and two types of arcs. The nodes are called activities and represent the work steps of a workflow. Depending on the level of abstraction, FDL distinguishes between program activities (single work step), process activities (refined by a whole workflow), and block activities (a group of activities to build repeating blocks). Various properties are set for each activity: start/exit conditions, role resolution, data mapping, as well as the associated application program. The solid arcs in the graph with annotated transition conditions specify the control flow, and the dashed arcs the data flow between activities. The whole graph is called a process model with a data source and a data sink. As FDL does not directly support a loop construct, the original travel request workflow specification was slightly modified. If a travel request is rejected by the approver, the request does not directly go back to the requestor, but is routed to a notification work step. Block activities to simulate the loop back were not used for simplicity of presentation.

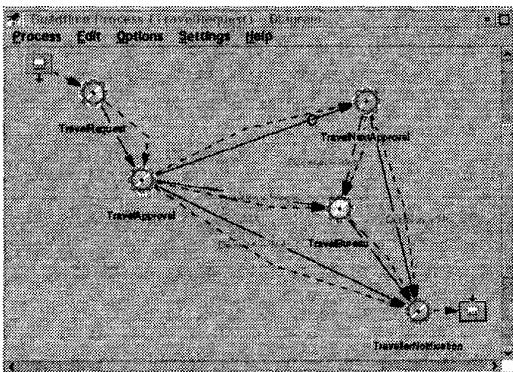


Figure 2. FlowMark Buildtime Client

The graphical workflow definition can as well be imported from FDL or exported to FDL. The following FDL excerpt in Figure 3 shows the FDL definition of the travel request sample application. The FDL program contains the

data structure definitions for the data flow, the organizational model, the program descriptions, and the process definitions. The process definition includes the activity definitions as well as the definition of the control and data flow (see [8] for a detailed description of FDL).

```

STRUCTURE 'TravelRequest'
  'Requestor' : 'TravelerInformation';
  'Trip' : 'TripInformation';
  'Decision' : STRING;
END 'TravelRequest'

STRUCTURE 'TravelerInformation'
...
END 'TravelerInformation'

ROLE 'Manager'
END 'Manager'

PROGRAM 'TravelRequestFormEditor' ( 'TravelRequest', 'TravelRequest' )
  TCPIP ADDRESS 'SPIRIT'
...
END 'TravelRequestFormEditor'

PROCESS 'TravelRequest' ( 'Default Data Structure', 'Default Data Structure' )

PROGRAM_ACTIVITY 'TravelApproval' ( 'TravelRequest', 'TravelRequest' )
  PROGRAM 'TravelRequestFormEditor'

  END 'TravelApproval'

  CONTROL NAME 'Submit' FROM 'TravelRequest' TO 'TravelApproval'
  CONTROL NAME 'Approve' FROM 'TravelApproval' TO 'TravelBureau'
  WHEN 'Decision' = "Yes"
  CONTROL NAME 'Reject' FROM 'TravelApproval' TO 'TravelerNotification'
  WHEN 'Decision' = "No"
  CONTROL NAME 'NextApprover' FROM 'TravelApproval' TO 'TravelNextApproval'
  OTHERWISE

  DATA FROM SOURCE TO 'TravelRequest'
  DATA FROM 'TravelRequest' TO 'TravelApproval'

...
END 'TravelRequest'

```

Figure 3. Travel Request in FDL

4.2 FDL/Lotus Notes Builder

The FDL modeled workflow needs to be translated into a Notes environment. In a FlowMark environment, the FlowMark buildtime client has a built-in FDL compiler, which translates the modeled workflow into FlowMark runtime objects. In order to run the workflow in a Notes environment, we extend the FlowMark FDL compiler to include a "FDL/Lotus Notes Builder", which generates the Notes runtime required to process the defined workflows. In this section we sketch the mapping of the FDL into the Notes environment. A similar mapping approach can be used for the implementation of workflow management with triggers on top of active relational databases [5]. The partitioning of a workflow model into various parts for distributed workflow processing based on message queuing is supported by Exotica/FMQM (FlowMark on Message Queue Manager [3]).

The FDL data structures can easily be mapped to field values in Notes documents. Although Notes does not support nested types, numeric and char types are straightforward to deal with, and the Notes rich text field type could serve for the storage of any complex multimedia types like graphics, voice, scanned images and OLE2 objects (if FDL were to support these types). Notes-specific document field types like AuthorNames, ReaderNames, and Names can be exploited to dynamically set read/write permissions on

the documents. The question of whether all pertinent data should reside in a single Notes document or in several is an open issue and the answer probably depends on the specific workflow type. In our sample workflow, it makes sense to have the travel request in one document but there might be workflows in which a “folder” of documents needs to be routed, and different documents of a folder are processed in parallel by different workflow participants at the same time. Developers must be aware of and account for the possibility of concurrent update conflict scenarios. Notes handles concurrent update detection, and flags concurrent updates as “conflict resolution documents”, or in the case of replication as “replication conflicts”.

The FDL organizational model (roles, levels of expertise, etc.) as well as the static and dynamic role resolution features necessitate enhancements to the Notes address book’s basic functionality. The Notes address book is thought of as a “mail address book”. In its basic form, it does not contain, e.g., hierarchical organizational information or role information. For this reason, an organizational database needs to be available at runtime, if FDL-based applications are to be directly supported. This database can either, as in GroupFlow [17], be a Notes database or an external relational database which can be looked-up through Notes’ interfaces to external data sources. The look-up logic and the role resolution algorithms can be implemented via Notes Macros, LotusScript, or Notes add-in tasks. A Notes add-in task is an executable program that runs alongside the other tasks that make up the Notes server, and performs a range of API operations [11].

The information provided by the FDL program description part of the workflow definition (see PROGRAM ’...’ in Figure 3) must be used for application program invocation. The program description contains information about application program names, working directories, and network information about where to execute the application program. For each program description, we create an agent which encapsulates this information and invokes the application properly. The agents are implemented as Notes simple actions, in Notes formula language, or in LotusScript. The formula language employs @functions and @commands which also allow to launch external applications (@Command([Execute])). LotusScript programs can invoke DLLs.

The implementation of the control and data flow in the FDL process definition depends on the type of activity: (1) manual activities like simple forms editing activity or interactive desktop applications, or (2) automatic activities which run on any machine in a network. For the first kind of activities, the routing alternatives can either be presented in several Notes “push buttons” in forms or be hidden in one “routing push button” which evaluates the transition conditions based on some field values and does the proper routing. Figure 4 for instance shows two push buttons to either

approve or reject a travel request [21]. The same purpose could be achieved by providing only one button which looks up a field value in the document and does the proper routing. Part of these push button implementations could be a digital document signature.

In the current prototype, the control and data flow is coded into the macro implementation. This provides the most flexibility in implementing the control flow. For the future, we plan to implement a generic “navigator” which looks-up the routing information in a workflow definition database and does the proper navigation. The navigator approach has the advantage of being able to write the audit trail in a separate log as part of the routing. Today, we keep only the current workflow status in the document itself.

The formula implementation in Figure 4 shows that currently we use the send model for the workflow implementation. In the future, we want to offer a combined send and share model. Interactive desktop applications which go beyond simple forms editing, e.g., word processing, spreadsheet editing, etc., are invoked by the end user as agents or through some in-place activation mechanism. Automatic activities do not depend on an end user event, but start through an agent as soon as they are ready to execute. The updating of the document by a previous work step provides the required event for the agent.

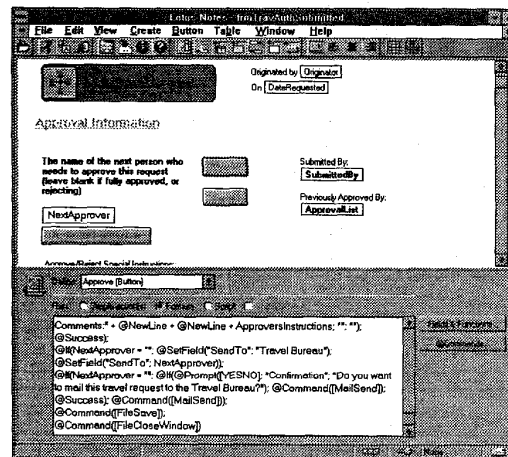


Figure 4. Notes “Push Buttons” and Formula

The FDL/Lotus Notes builder can provide some default forms and views. These forms and views can easily be customized with Notes by the end user.

5 Summary and Outlook

Notes Release 4 is a high-end groupware system which provides, apart from many other features not even mentioned in this paper, a solid piece of middleware to build

workflow applications. Notes provides built-in facilities for desktop application integration, flexible work routing, and storage of multimedia documents. Notes also offers many "hooks" to customize and extend the system. This paper describes, how formalized workflow models and tools that support workflow analysis and application generation can be used to build upon the native workflow power in Lotus Notes. We used the WfMC workflow model and the FlowMark workflow definition language FDL to define structured workflows. We outlined the processing of these workflows in a Notes environment. The synergy among dedicated workflow modeling tools like the FlowMark build-time client and groupware systems like Lotus Notes with native workflow support has tremendous productivity implications.

The paper also reveals some shortcomings, both on the workflow model side as well as on the Notes side, which are future research issues. Today, FDL does not support (external) events, ad hoc routing (exceptions), and explicit loops. The type system of the data flow structures in FDL could be enhanced similar to the rich text field data type in Notes documents in order to support images, OLE2 containers, or even URLs. Notes lacks direct transaction support (available only via LotusScript and ODBC), which is important for mission-critical applications as well as for the efficient handling of race conditions in the case of parallel workflows accessing the same documents, or different users trying to process the same workflow at the same time. The introduced FDL/Lotus Notes Builder needs to be improved to deal with process, block, and bundle activities (multiple instantiations of a single activity determined at run time). The Notes workflow processing environment will include a generic navigator as well as add-in for role resolution and logging of the audit trail.

Acknowledgments Our thanks go to Peter O'Kelly from Lotus Notes for reviewing this paper.

References

- [1] K. Abbott and S. Sarin. Experiences with workflow management: Issues for the next generation. In *Proc. of the Conf. on Computer-Supported Cooperative Work, CSCW*, pages 113–119. ACM Press, 1994.
- [2] Action Technologies Inc, Alameda, CA. *What is Action-Workflow*, 1994.
- [3] G. Alonso, C. Mohan, R. Günthör, D. Agrawal, A. El Abbadi, and M. Kamath. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *IFIP WG8.1 Working Conference on Information System Development for Decentralised Organizations*, pages 1–18, Trondheim, Norway, Aug. 1995. accessible via <http://www.almaden.ibm.com/cs/exotica>.
- [4] K. Brockschmidt. *Inside OLE2*. Microsoft Press, Redmond, WA, 1994.
- [5] J. Eder, H. Groiss, and H. Nekvasil. A workflow system based on active databases. In *Connectivity 94*, pages 249–265. Oldenburg, 1994.
- [6] W. Fisher and J. Gilbert. FileNet: A Distributed System Supporting WorkFlo; a Flexible Office Procedures Control Language. In *IEEE Computer Society Office Automation Symposium*, pages 247–249, Gaithersburg, MD, April 1987.
- [7] D. Hollingsworth. Workflow management coalition: The workflow reference model. Document TC00-1003, Workflow Management Coalition, Dec. 1994. accessible via <http://www.aiai.ed.ac.uk/WfMC/>.
- [8] IBM. *FlowMark - Modeling Workflow, Version 2.1*. IBM, mar 1995. Document No. SH19-8241-00.
- [9] S. Khoshafian and M. Buckiewicz. *Introduction to Groupware, Workflow, and Workgroup Computing*. John Wiley & Sons, Inc., New York, 1995.
- [10] F. Leymann and D. Roller. Business Process Management with FlowMark. In *Proc. 39th IEEE Computer Society Int'l Conference (CompCon), Digest of Papers*, pages 230–233, San Francisco, California, February 28 – March 4 1994.
- [11] Lotus Notes, Cambridge, MA. *Application Program Interface (API) User Guide Release 4.0*, 1995.
- [12] Lotus Notes, Cambridge, MA. *Automating Workflow with Lotus Notes*, 1995. (Online documentation workflow.nsf).
- [13] Lotus Notes, Cambridge, MA. *Lotus Notes Developer's Guide, Version 4.0*, 1995.
- [14] D. Marshak. Lotus Notes Release 4.0. In-Depth Research Report, Patricia Seybold Group Inc., Jan. 1995.
- [15] R. Marshak. Pavone GroupFlow - Providing a Workflow Suite for Lotus Notes Environments. Workgroup Computing Report Vol. 18, No. 11, Patricia Seybold Group Inc., Nov. 1995.
- [16] R. Medina-Mora, T. Winograd, and F. Flores. The Action workflow approach to workflow management technology. In *Proc. of the Conf. on Computer-Supported Cooperative Work, CSCW (Toronto)*, New York, 1992. ACM Press.
- [17] L. Nastansky and W. Hilpert. The GroupFlow system: A scalable approach to workflow management between cooperation and automation. In *13th IFIP World Computer Congress*. IEEE Computer Society, 1994.
- [18] K. North. *Windows Multi-DBMS Programming*. John Wiley & Sons, Inc., New York, 1995.
- [19] R. Orfali, D. Harkey, and J. Edwards. *The Essential Distributed Objects Survival Guide*. John Wiley & Sons, Inc., New York, 1996.
- [20] L. Pyle. *Creating Lotus Notes Applications*. Que Corporation, Indianapolis, IN, 1994.
- [21] E. Rayl. *Lotus Notes Developer's Guide*. Sams Publishing, Indianapolis, IN, 1st edition, 1994.
- [22] S. Sarin, K. Abbot, and D. McCarthy. A process model and system for supporting collaborative work. In *Proc. of the Conference on Organizational Computing Systems (Atlanta), SIGOIS Bulletin 12 (1991) 2/3*, pages 213–224, New York, 1991. ACM Press.
- [23] B. Silver. The BIS guide to workflow software. Draft, BIS Strategic Decisions, Norwell, MA, 1995.
- [24] J. Toohey. *Using OLE 2.x in Application Development*. Que Corporation, Indianapolis, IN, 1994.