

## **Integrated Document/Workflow Management System Architecture**

W. Staniszki, B. Kiepuszewski, M. Nahum Ferber, T. Piórkowska, D. Puternicki, and K. Subieta

RODAN SYSTEM  
Jagielska 50c  
02-886 Warszawa  
POLAND

Witold.Staniszki@rodan.waw.pl

### **Abstract**

An integrated DWMS architecture comprising IBM FlowMark, Lotus Notes and Office Objects is presented. Office Objects comprises a library of C++ object classes supporting document imaging and hierarchical storage management. The object-oriented analysis and design platform based on OMT, extended with the Use Case and Dialogue Model notations, provides the system design and implementation paradigm. Mappings from design models into the Lotus Notes document and FlowMark process models are discussed. A pilot application of the architecture is outlined.

### **1. Introduction**

The rapid proliferation of workflow systems, i.e. information systems utilising the workflow management technology, has been inhibited by several important factors resulting from relative immaturity of this area. A thorough discussion of the current gaps and the required development, both on the research as well as engineering level, has been presented in [GEOR95]. Our practical experience with workflow systems, both from the proprietary software as well as application developers perspective, has shown that the principal challenges lie in the system architecture as well as the application development methodology areas. Therefore, we are focusing on these two aspects of our current development work in the area of workflow systems.

The Integrated Document/Workflow Management System (DWMS) is proposed as a development platform providing a common control layer for document management as well as transaction-oriented application functions. A study of application requirements presented in [JOOS94b, JOOS96] has shown that the advanced workflow systems require document management, such as document imaging, storage, and retrieval functions, as well as general purpose processing functions, that may require access to various application systems concurrently active within a computer network. We discuss the architecture and requirements of such systems in the following section.

Our application development work at the Ministry of Labour and Social Policy (MOLP) of Poland, having an objective to automate principal office procedures of the ministry, has shown in a very early stage that a classic information system development methodology is not sufficiently universal to allow for the rigorous systems analysis and conceptual design specification, required both by the future system users as well as by the developers involved in the ensuing system design and implementation phases of the MOLP project. The methodology initially used was, following our company standard, the Object Management Technique (OMT) proposed in [RUMB92]. Although an object-oriented methodology is suitable for distributed information system analysis and design, rather important enhancements of the OMT methodology were necessary, before a workable workflow system could be designed and implemented. We outline the methodology in section 3 of this paper and further clarify its notation with the use of a brief example presented in the appendix.

Finally, we conclude the paper discussing the pros and cons of the approach adopted for the MOLP system development concentrating both on the architecture as well as the methodology aspects of our work. Future directions of our research and development work in the area of workflow systems are succinctly outlined.

## 2. The DWMS architecture

The advent of distributed client/server architectures freely deployed on open computer platforms has considerably increased the complexity of information software design and implementation process. The advanced computing environments comprise many heterogeneous, autonomous and distributed systems (HAD), starting from personal computing environment, e.g. Microsoft Office, deployed on the client workstations, and leading to mainframe transaction systems managing complex business data. The technical issues associated with HAD environments as exhaustively discussed in [GEOR95]. The general principles underlying such environments in conjunction with the workflow management systems are illustrated in figure 1.

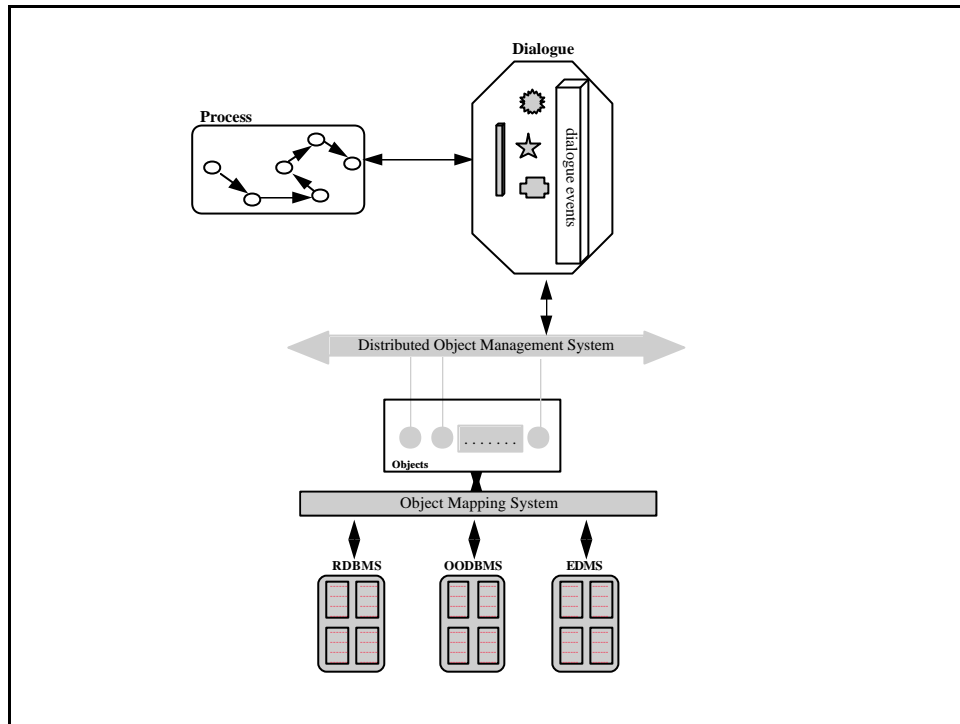


Figure 1. A generalised workflow HAD architecture.

A workflow specification, represented by a directed graph, provides a process class description catering for all possible conditions of process instance executions, which are corresponding to a particular path in the process graph. The important characteristic of the process graph is that the process activities, represented by graph vertices, may be executed automatically by application functions (computer programs) or manually by users participating in the business process controlled by the workflow system.

In the first case, the workflow specification, i.e. the workflow directed graph, corresponds to a script, ranging over a number of processing platforms integrated within a computer network, representing the control flow in a fully automated distributed computing process invoking processing functions either through a classic RPC protocol, or with the use of a distributed object management system, such as OMG CORBA [OMG95] or Microsoft's DCOM. In this case, the fully automated process resembles a distributed transaction. Hence the ACID model requirements may be met, provided that all underlying object mapping subtransactions are supported by the ACID model compliant transaction control functions. Alternatively, a customised transaction management can be used. The object mapping management function is required in order to provide a facility to store and update persistent objects. The most common repositories used to store persistent objects are relational database management systems (RDBMS), object-oriented database management systems (OODBMS), and electronic document management systems (EDMS). The workflow transaction management solutions would in such cases employ techniques like compensating subtransactions and process event logs. The obvious requirement is that the workflow management systems support the required functionality, e.g. the process activity log [IBM95].

In the second case, the manual activity creates an entirely different processing environment, namely an interactive man/machine environment, usually supported by a GUI, hosted in the client workstation invoking

local and/or remote processing functions. In an object-oriented environment, one would expect to have an event-driven GUI management, usually developed in Visual Basic, Smalltalk or similar languages, featuring object classes as program variables. The object instances, representing business objects or legacy system wrappings [OMG95], may be accessible either by an ORB [OMG95] or similar middleware (e.g. Micosoft's DCOM). Most of the application logic would normally reside in the object class specification. A monolithic system architecture may also be used provided that the required invocation granularity is supported by an API.

In both of the above cases, distribution independence is supported, and, in the case of the object-oriented approach implementing object persistence with the use of an object mapping system, also data independence may be attained. Business objects, like compound documents, may be mapped into various data repository types, and the mapping algorithms are usually comprised in the object class method logic. Standard mappings, like for example the object - relational data mapping, are automatically supported by object-oriented CASE tools, e.g. Cayenne's ObjectTeam, that generate the required mapping method logic to store, update, and retrieve the object data structures. The referential integrity constraints are also supported by generation of the required database procedures and triggers.

The generalised workflow system architecture represents technological requirements that should be met by any advanced DWMS architecture. Our design approach has been based on the above requirements and the resulting architecture is shown in figure 2.

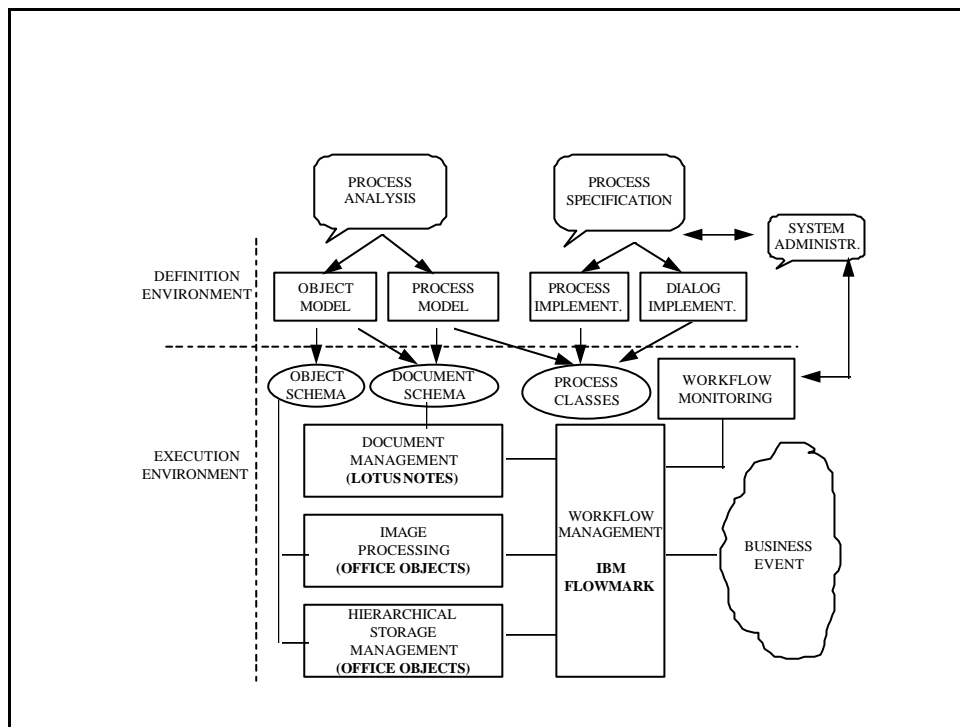


Figure 2. The DWMS architecture.

The integrated DWMS architecture features the IBM proprietary software systems, such as Lotus Notes and IBM FlowMark, the Office Objects library of classes developed and marketed by RODAN SYSTEM, and the object-oriented CASE tool ObjectTeam for OMT developed by Cayenne Software. Our added value lies in seamless integration of the above proprietary software environments, as well as in the application development methodology and skills. The dialogue scripts are currently developed in Visual Basic and CA OpenROAD.

The definition environment comprises model specification facilities used according to our design methodology, namely ObjectTeam to support the analysis and conceptual models, the IBM FlowMark Development Client to implement the process graph, and the Lotus Notes client to define document classes. The TCL [OUST94] have been developed within the ObjectTeam lower CASE functionality to generate the FDL skeletons of FlowMark processes as well as the partial LN document descriptions. Thus, consistency of the conceptual and logical levels of the workflow system specification is maintained.

The DWMS execution environment comprises client and servers of three principal building blocks, namely IBM FlowMark, Lotus Notes, and Office Objects. However, the end user sees the system through the FlowMark Runtime Client perspective and the process task execution is supported by the dialogue scripts developed in Visual Basic and CA OpenROAD. Lotus Notes and Office Objects functions are invoked via the respective API's or the OLE protocol. The DWMS execution environment program module structure is shown in figure 3.

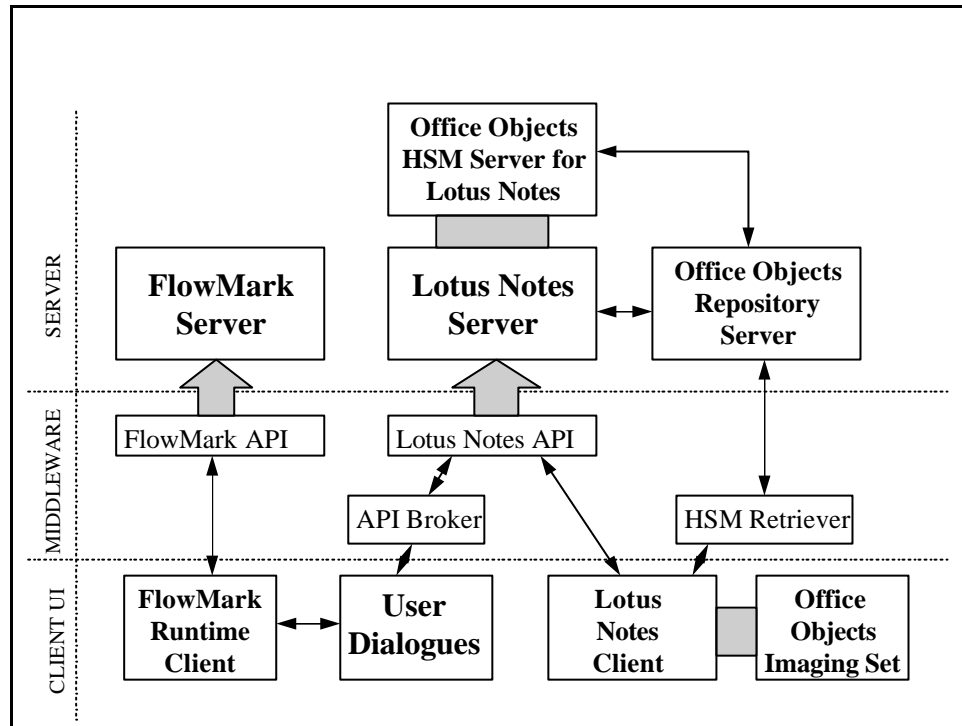


Figure 3. The DWMS execution environment.

The client user interface running in the MS Windows 3.11 or Windows 95 environment comprises the FlowMark Runtime Client and a collection of User Dialogue modules jointly providing the principal platform the user/workflow system interaction. The dialogue modules communicate with the Lotus Notes server via the API Broker, that may easily be extended to handle other API's or CORBA object messages. The user dialogues supporting workflow process tasks selected from a FlowMark work list communicate with the document database, or possibly with any other information system, presenting documents, parts of documents, document images, or electronic forms required to fulfil the process task data requirements. The LN document database is also accessible via the Lotus Notes Client providing the standard LN functionality extended with the document imaging functions supported by the Office Objects Imaging Set functions accessible via the OLE protocol. The middleware level comprises also the HSM Retriever module that supports selection of the Lotus Notes document fields stored in devices controlled by the Office Objects Hierarchical Storage Management functions.

The Office Objects Hierarchical Storage Manager (HSM) for Lotus Notes has been implemented as the Lotus Notes Server Add-in task and its principal function is to support the bi-directional migration of LN document field among storage devices appropriately defined within the migration paths hierarchy levels, namely the magnetic disks, the jukeboxes and magnetic tapes. The architecture of the Office Objects HSM is shown in figure 4.

Office Objects HSM is responsible for the migration of documents to lower cost storage, ensuring a cost effective and still efficient document storage. Documents are taken from the specified sources and are stored in a set of mass storage devices of different kinds: magnetic, optical, WORM, CD-ROM, DAT, following custom made predicates, called the migration rules. Documents or part of the them, can be reconstituted on demand, by simply retrieving them from the closest available data repository to the source, making HSM completely transparent to the end-user. Migration rules are triggered by time events, such as frequency or specific dates when the rule is to be executed, or external events, such as conditions on the available space in a repository.

The architecture of HSM consists of the migration server available as the Add-In task to Lotus Notes used as the engine to migrate Notes items, the retrieve client and server used to restore the migrated items to their original location, the HSM editor used by the system administrator to define the migration paths and rules, and the Repository server used as the interface to the different storage devices comprised in the storage hierarchy as repositories to migrated items throughout the network.

The HSM manipulates data objects which are generally non-structured binary data. These objects are stored in data repositories and an index is created for further references. A pointer to the object is stored in the original document to be used when retrieving the migrated parts, if necessary. Data repositories are organized in migration paths. A migration path is a chain of repositories of different kinds through which data objects are migrated. Each repository in a migration path represents a level, level 0 being the beginning of the chain, thus, the source of documents.

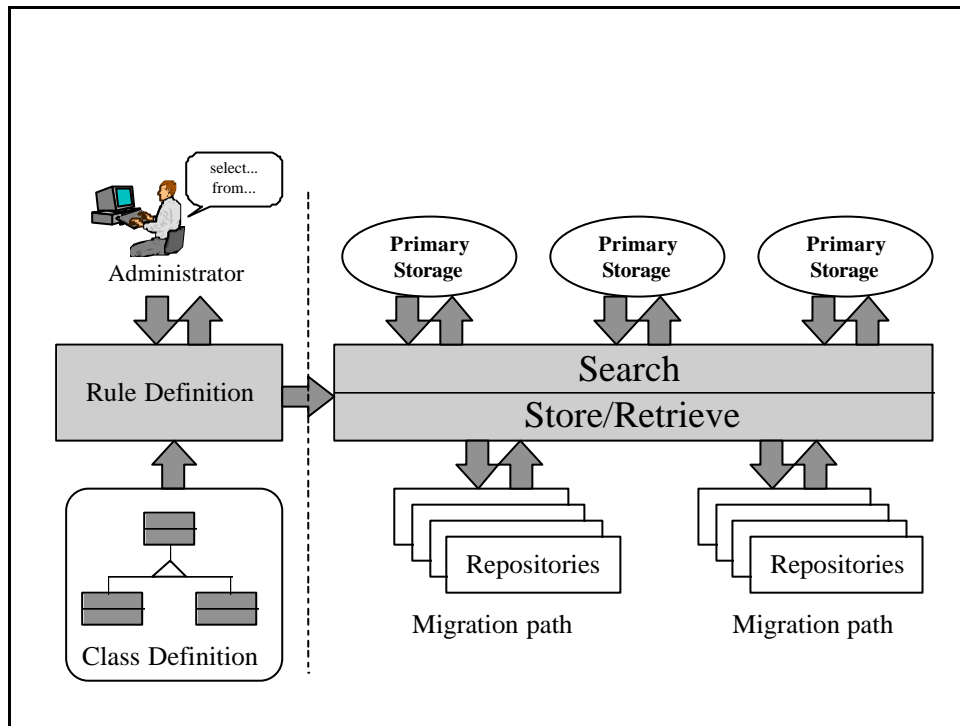


Figure 4. The Office Objects Hierarchical Storage Management architecture

The migration rules have two aspects: the scope of the rule and the schedule execution schedule of the rule. The scope of the rule identifies documents or parts of documents that are to be migrated. The rules are defined as standard (ANSI compliant) SQL sentences, comprising arguments, such as names and attributes dependent on how the source documents are organized. The execution scheduling of the rule indicates when the rule should be evaluated. A rule can be scheduled on specified intervals (seconds, minutes, hours, days) with a resolution of 10 seconds, on specified days (day of the week, day of the month) at a specific hour, or on the basis of an external condition or event, such as for example filling up of a document repository.

The task of HSM for Lotus Notes is to migrate Note's fields that are not involved in full-text indexing such as file attachments and presentation records of composite fields. These are binary data fields, which are usually large. Entire documents are not removed from the Notes databases, making possible to search and find „migrated documents”; whenever the document is requested, a demand for the missing fields is made to migrate upwards the necessary data items.

The migration rules text have the following syntax and appropriate semantic:

```

SELECT <item name>, <item name>, ?
FROM <document class>, <document class>, ?
WHERE <where clause>

```

where:

<item name> are the field names to store; the symbol '\*' is used for all the storable fields.

<document class> are the different classes of documents as explained below

<where clause> are conditions on attributes of the documents, such as the value of a field, or the last access date, etc.

The HSM for Lotus Notes is a set of Notes Add-In tasks, this makes it possible to monitor the HSM/LN from the Notes server remote console. Each rule is executed by a different add-in task process providing for parsing of the migration rules and translation to the Lotus Notes API calls, selection and extraction of LN objects eligible for migration according to the corresponding rules, modification of the Lotus Notes document fields in order to reflect the fact that certain objects were archived.

The Office Objects Imaging Set comprises the standard imaging functionality, i.e. the scanning, viewing, and printing functions. The OCR functions are accessible directly from the OO Imaging Set Scanning Station Client. The standard image processing functions, such as scanner control, image compression/decompression, as well as the image viewing operations are based on the Pixel Translation Tool Kit [CORNxx].

### 3. The workflow system development methodology

Importance of the appropriate workflow system design methodology has already been pointed out in the introduction. The lack of a formal, generally accepted workflow system methodology has been identified as a major risk factor in a study of 12 selected workflow system projects in Holland [JOOS94b]. We have experienced a similar situation, where analysis and conceptual design of a complex workflow system has been hindered by the lack of an appropriate methodology and analysis tools. The workflow system being developed requires a collection of hierarchically decomposed workflow processes spanning many departments. It is also required that the Lotus Notes database supports the active database capabilities in the form of database triggers and alerters.

To meet the challenge of the project, we have decided to develop an ad-hoc workflow system development methodology, stemming from the OMT methodology being currently used by our project teams developing object-oriented systems. For obvious reasons, we have attempted to stay as close to the well accepted OMT methodology as possible, in order to minimise the conversion effort. In particular, the question still open is the applicability of the proposed methodology extensions to development of the general purpose information systems. We have made a tentative assumption, that the proposed methodology will be useful in all cases, where the information system development is directly following, or indeed is a part of, a business process re-engineering (BPR) project. This may be reinforced by the fact, that most of the BPR efforts should evolve into a workflow system development project, due to existence of a common denominator, i.e. the process-oriented view of the business.

Additionally, our long involvement with the structured information system development methodologies indicates, that the most important aspect of any methodology is the modelling compatibility among its distinct abstraction levels. This property is paramount to an orderly mapping of design information between development phases leaving no representation gaps that may lead to the loss of information or, at least, to undue design effort resulting from translation between incompatible representation models. Hence, our prime constraint has been to specify a methodology that allows for maintaining the same modelling paradigms on all workflow system design and implementation levels.

The starting point of our work has been the observation that the Use Case approach to the requirements analysis [JACO92] is strictly corresponding to the client view of the business process, which is the principal BPR analysis and design paradigm. This point has been amply illustrated in [JACO95]. The Use Case notation has already proliferated into the later versions of OMT, and indeed, is a principal feature of the Unified Method.

Hence, the first step of our methodology is to identify the principle business processes and represent them as Use Cases. The business process realms, represented by Use Cases, provide a convenient classification platform by focusing on the principal functional characteristics to be supported by the workflow system. Each use case is defined in terms of its principal aspects represented by the process model, the object model, the dialogue model, as well as by the applicable metrics. The collection of the Use Case models and the corresponding notation are shown in figure 5. It should have become clear by now, that the concept of a Use Case is equivalent to the concept of a process class in a workflow system.

The Process Model of a Use Case is represented as a directed graph, not necessarily acyclic, with the use of a standard State Transition Diagram notation. However, the semantics of the STD notation have been redefined in the following way. The State shape represents an action of the process diagrams and the Transition lines show the flow of control, i.e. process graph links, in the process graph. Special shapes representing the AND and OR process graph nodes were introduced in order to support the graph topology definition required by the Workflow Management Coalition [WORK94]. An important advantage of the above approach is the compatibility with the FDL process class implementation model used in FlowMark [IBM95].

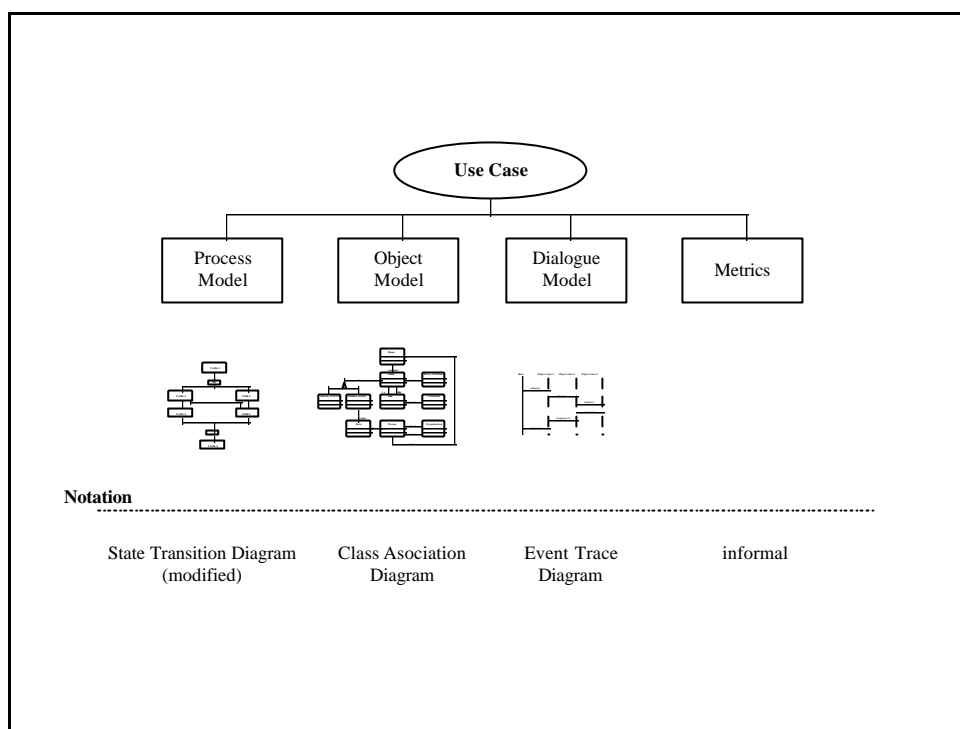


Figure 5. The workflow system design models.

The Object Model is to be defined in the standard OMT way , i.e. as a Class Association Diagram (CAD), with the exception of the special feature definitions, namely the trigger and the HSM migration rule specifications. The partial CAD's are integrated to create the global Object Model. The integration process uses the methodology presented in [BATI92]. The Object Model is further refined during the subsequent system design and implementation phases. In the case of the object-oriented information system development, the standard OMT methodology is used to develop the object classes to be invoked by the workflow system. The object-oriented approach is also used to design the Lotus Notes databases.

The object-oriented approach is compatible with the object-oriented features of Lotus Notes version 4, hence the modelling paradigm remains the same throughout the entire workflow system development life cycle. Generalised document classes considered typical for all Use Cases, and indeed for many different DWMS implementations, have been developed with their respective behaviour implemented with the use of the LN scripting language. A subset of the LN document class inheritance hierarchy is shown in figure 6. All user defined document classes are appended at the bottom of the generalised document class inheritance hierarchy. The multiple inheritance is supported on the specification level.

The trigger rules are supported by the *Activator* document class and may be inherited by any user document class. The trigger rules are defined as standard Event-Condition-Action (ECA) rules using the following syntax:

[ON {LNoperation, DAY, HOUR}] IF {predicate} THEN ['message'], [recipient-1, ..., recipient-N]

where:

{a,b} - mandatory either a or b

[a,b,c] - optionally any one element of the list

a,...,a - a list of elements of the same type

If all optional elements following the keyword *THEN* are omitted a default action, defined for the *Activator* document class or the subject user document class is employed. The default event is *ON DAY*, meaning that the trigger rules will be evaluated at the beginning of every processing day.

The Dialogue Model, specified with the use of the standard Event Trace Diagram (ETD) notation of the OMT, is to be defined for at least each process graph manual activity, i.e. an activity that will result in a task inserted into one or more work lists maintained by the FlowMark Runtime Client. Automatic activities, i.e. activities that either invoke a computer program or a FlowMark subprocess, may optionally have a dialogue model

in order to document the object message passing logic only. The dialogue model for a manual activity specifies the information system support received by the person handling this particular task. The object class symbols, i.e. the vertical dotted lines, correspond to object classes, if an object-oriented information system is invoked, or to LN documents or parts of documents. Calls to monolithic information system API may also be specified. The internal dialogue logic is specified as a message passing interaction among object classes. The dialogue specification is refined gradually in the subsequent design life cycle phases to be ultimately mapped into a dialogue script specified in Visual Basic or a similar GUI development environment.

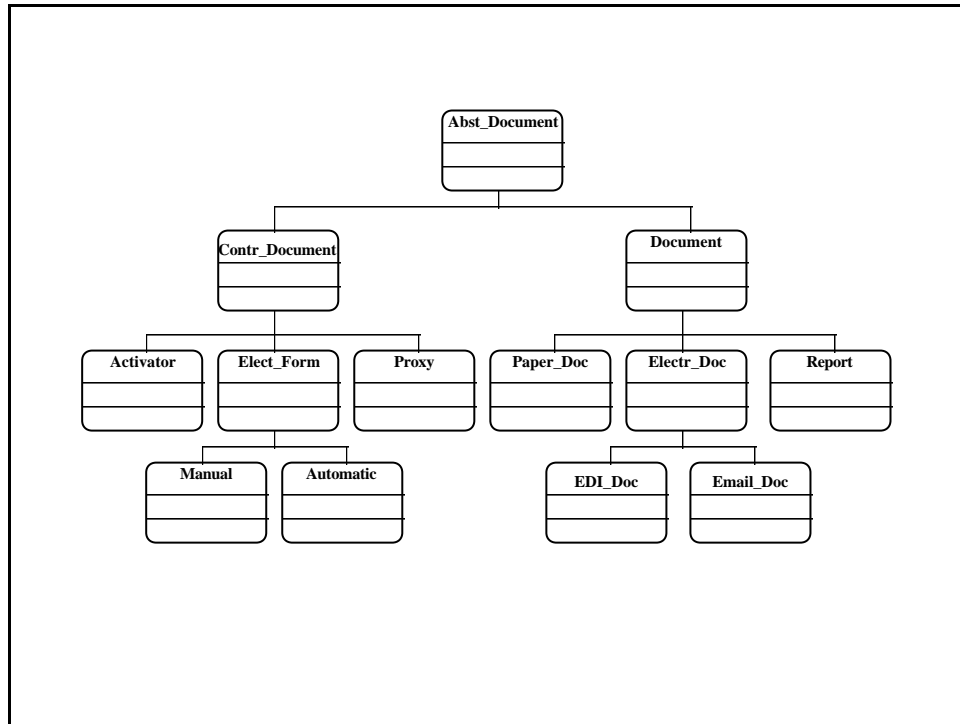


Figure 6. The Lotus Notes document class hierarchy.

An important set of design information, established during the analysis and conceptual design phases, are the workflow system metrics. The workflow system metrics fall into two principle categories, namely the workload category and the performance constraints. The above two sets of metrics are related in a sense, that the first category determines the performance-oriented design decisions, and the second delimits the eligible design solution space. The workload metrics specify, among others, the process class frequency, the relative activity frequencies, and the information system user function execution frequencies within each dialogue. The first two metrics should already be established at the BPR level, since they influence the business process design, in particular the human resource assignment to particular business roles. The latter category reflects the processing requirements of each task identified within the business process. The performance constraints deal with the system performance constraints, such as response time, storage capacity, and system availability, or with the business process performance constraints such as the task completion limits, the process duration limits, and the mean process duration. A good source of methodological information useful for analytical modelling of workflow processes may be found in [LAZO84], and a tool for performance-oriented analysis and design has been presented in [ORLA88].

The above methodology presentation pertains mostly to the systems analysis and conceptual design phase of the workflow project life cycle. The subsequent phases employ the workflow system specific techniques of business process specification. The structure of the FlowMark process specification is shown in the form of a Class Association Diagram in figure 7. In this case, the object classes represent the principal FDL description categories and their relationships. In order to maintain the presentation clarity the diagram is extracted from the complete FDL CAD version maintained in the ObjectTeam repository. The classification of FDL categories and their relationships provided a semi-formal model of the workflow system definition, thus enhancing understanding and supporting quality control of the FlowMark process specifications. A similar approach has been presented in [JOOS94a].

An important advantage derived from the above analysis of the anatomy of FlowMark has been the capability to design workflow system specification forms [RODA96], used during analysis and conceptual design, providing guidance for the analysts and quality assurance officers. We believe that such forms are workflow management system independent and that they may be used in any workflow system environment, provided that the directed graph paradigm is used for workflow specification.

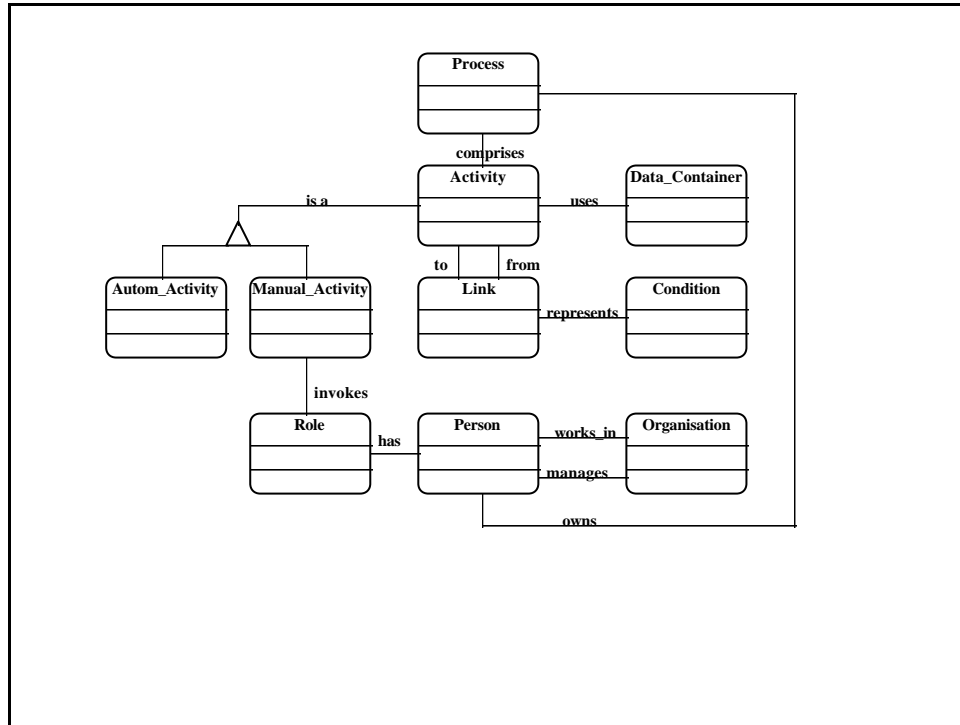


Figure 7. The FlowMark process schema relationships.

#### 4. Conclusions

The principal contribution of our work is integration of a generalised DWMS architecture comprising seamlessly interoperating heterogeneous systems, each meeting a clearly defined functional role. The object-oriented specification approach has been useful in defining the semantics of the system functions and in implementing the integrating software components. The architecture has been successfully applied to a complex workflow system and has proven to be a reliable and performant platform for a wide range of workflow systems.

The successful development of complex workflow applications has been to a great extent determined by an early adaptation of a workflow system design methodology, developed within the project team as an extension of the OMT methodology used as a company standard. Although the semi-formal, graphical process specification is presentable to the end-users, the final design verification is only feasible with the use of the workflow process fast prototyping approach.

The disadvantage is the relative immaturity of the applied methodology and the current lack of the formalised performance-oriented design framework. It seems, that the methodology requires disproportional effort during the analysis and conceptual design stages, due to a high level of formal and completeness requirements at this stage. The design documentation is voluminous and hard to read for the end-user. However, the ensuing workflow system life cycle phases proceed relatively fast, possibly due to reuse of standard solutions, both on the design level in the manner similar to the design patterns approach described in [GAMM94], as well as on the implementation level thanks to definition of the Lotus Notes generalised document model and the resulting reuse on the behaviour inheritance level.

The future work will concentrate on further specification and refinement of the workflow system design methodology, in particular in the area of the performance-oriented analysis and design of the workflow systems. We shall also refine the TCL mapping procedures to improve the mappings between the conceptual design level models and the DWMS specification models.

## References

- BATI92 Batini, C., Ceri, S., Navathe, S., Conceptual Database Design: An Entity-Relational Approach, Benjamin Cummings, 1992.
- CORN94 PixTools, Programmer's reference Manual, Pixel Translations, Inc., Cornerstone Imaging, San Jose, CA, December 1994.
- GAMM94 Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series, 1994
- GEOR95 Georgakopoulos, D., Hornick, M., and Sheth, A., An overview of workflow management: from process modeling to workflow automation infrastructure, in Distributed and Parallel Databases, 3, 1995, Kluwer Academic Publishers, Boston, USA.
- IBM95 IBM FlowMark User manuals, IBM, 1995.
- JACO92 Jacobson, I., Object-Oriented Software Engineering - A Use Case Driven Approach, ACM Press, Addison-Wesley, 1992.
- JACO95 Jacobson, I., The Object Advantage: Business Process Re-engineering with Object Technology, Addison-Wesley, 1995.
- JOOS94a Joosten, S., Trigger modelling for workflow analysis, in Proceedings CON '94: Workflow Management, Challenges, Paradigms and Products (October 1994), A.B.G. Chroust (Ed.), Oldenbourg, Wien/Munchen.
- JOOS94b Joosten, S., Aussems, G., Duitshof, M., Huffmeijer, R., and Moulder, E., Wa-12: an empirical study about practice of workflow management, Tech. Rep., University of Twente, Dept. of Computer Science, 1994.
- JOOS96 Joosten, S., Brinkkemper, S., Fundamental concepts for workflow automation in practice, Proceedings of the 5th Int. Conference on Information System Development, Gdańsk, September 1996.
- LAZO86 Lazowska, E.D., Zahorian, J., Graham, S.G., and Sevcik, K.C., Quantitative System Performance, Computer System Analysis Using Queueing Network Models, Prentice-Hall, Englewood Cliffs, USA, 1986.
- MARS94 Marshak, R., Software to Support BPR - The value of Capturing Process Definitions, Workgroup Computing Report, Patricia Seybold Group, Vol. 17, No. 7., July, 1994.
- OMG95 Common Object Request Broker Architecture and Specification, Object Management Group, June, 1995.
- ORLA88 Orlando, S., Perri, V., Scrivano, S., Staniszki, W., Database Analyzer and Predictor - An Overview, Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, IEEE 1989.
- OUST94 Ousterhout, J.K., Tcl and the Tk Toolkit, Addison Wesley Professional Computing Series, Reading, Mass., 1994.
- RODA96 Object-oriented System Design Methodology RODAN NT (in Polish), Rodan System Sp. z o.o., Warszawa, 1996.
- RUMB92 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., Object-Oriented Modelling and Design, Prentice-Hall International Inc, 1992.
- SILV95 Silver, B., BIS Guide to Workflow Software: A Visual Comparison of Today's Leading Products, BIS Strategic Decisions, September 1995.
- WORK94 Workflow Management Coalition, Information Pack, Grenoble, France, July 1994

## Appendix A workflow system specification example

We assume an idealised purchase procedure in some company. An employee (applicant) applies for acquisition of a new product. The application is sent to the respective manager who starts an instance of the purchase process and assigns to some other employee all further matters related to the purchase. The employee prepares an application form and the product information related to the planned purchase. Then, the manager accepts or rejects the application. If the application is accepted, the ensuing procedure depends on the estimated cost. If the cost is greater than 1000 ECU, then the application must also be accepted by the company president. Finally, the application must be accepted by the financial department. It may have some doubts; in this case the decision must be consulted with the president. If the application is accepted by all necessary authorities, the applicant is informed about the positive decision and the purchase is effected. If the application is rejected, the applicant is informed about the negative decision. Fig. 8 presents this scenario as a formal process model.

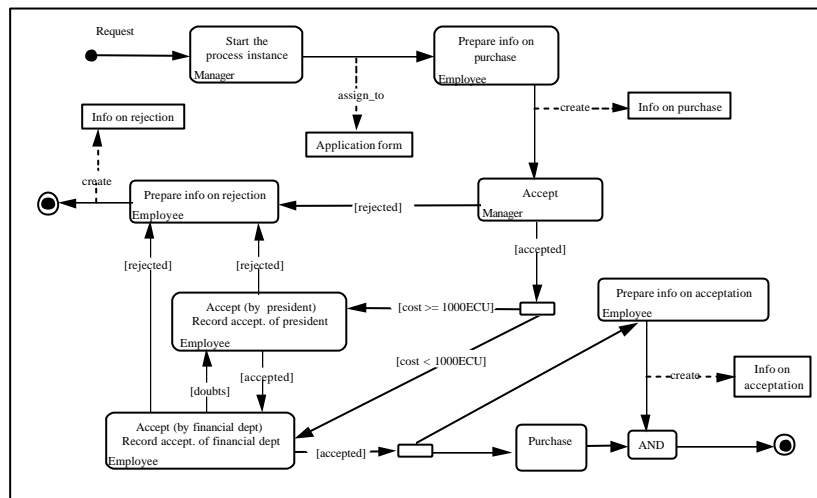


Figure 8. Purchase: the process model

Figure 9 presents the object model, which reflects all object classes participating in the process, including some roles. Note cases of the generalisation hierarchy (denoted by deltas) as well as cases of the aggregation hierarchy (denoted by diamonds). We use the standard notation of the OMT methodology.

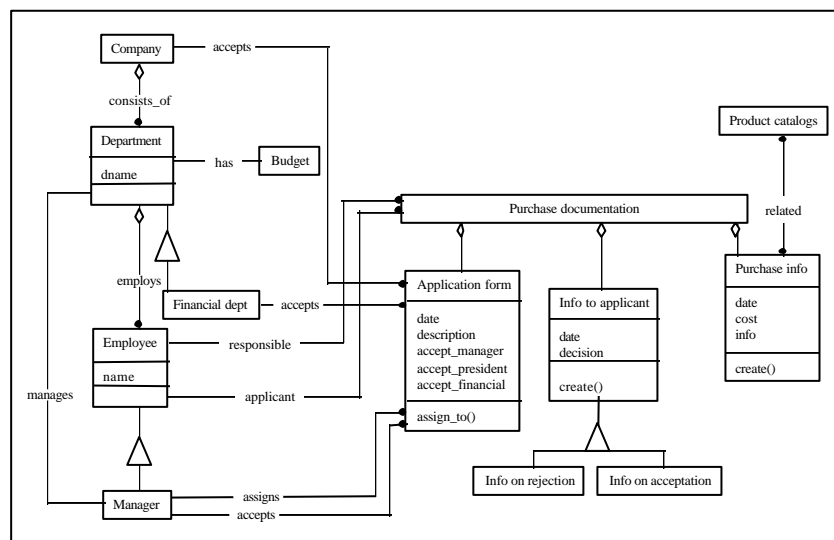


Figure 9. Purchase: the object model

Figure 10 presents a dialogue model which supports acceptance of the application by the manager. He or she needs the synthetic information about the budget, about the application, and about the product to be purchased. The latter information is collected via the chain of messages to classes Purchase documentation, Purchase info, and Product catalogues.

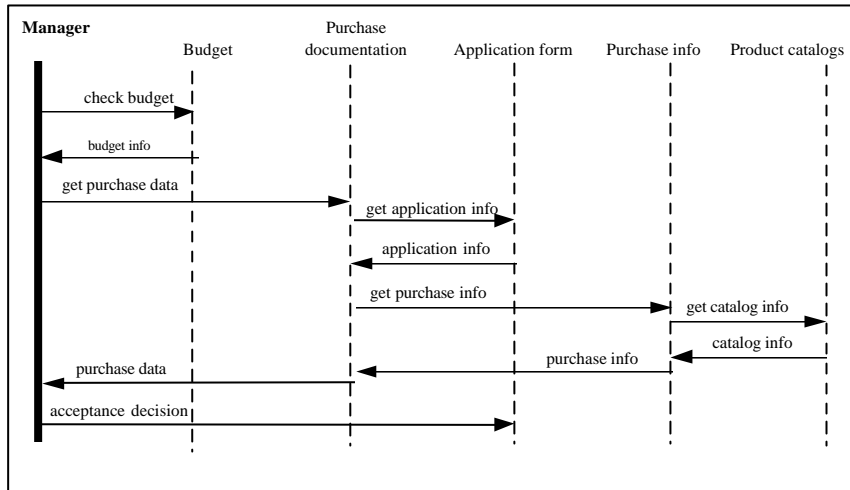


Figure 10. Purchase: a dialogue model

Figure 11 presents the storage optimisation facility HSM which makes it possible to change dynamically the media for data storage. The rules of the data migration consist of two parts: „what?” and „when?”. The first part, which determines the data that are candidates for migration, consists of one or more SQL queries. The second part determines the temporal condition for an automatic action which moves the data selected in the first part to another medium.

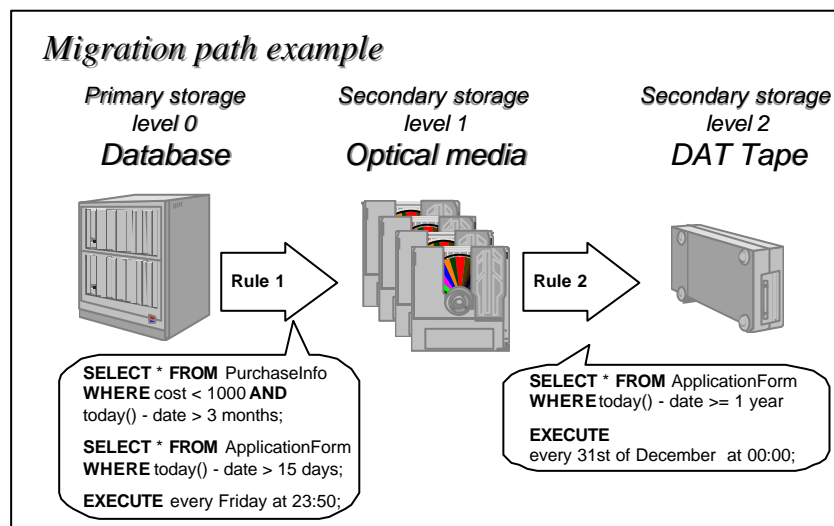


Figure 11. Purchase: a Hierarchical Storage Management (HSM) definition

Figure 12 presents a FlowMark diagram resulting from the process model shown in Figure 8. The nodes and arrows of this diagram almost 1:1 map the activities presented in the conceptual process model. Some peculiarity of FlowMark concerns the absence of loops, which are to be modelled as (repeated) action blocks, as in the case of Accept\_financial\_dept.

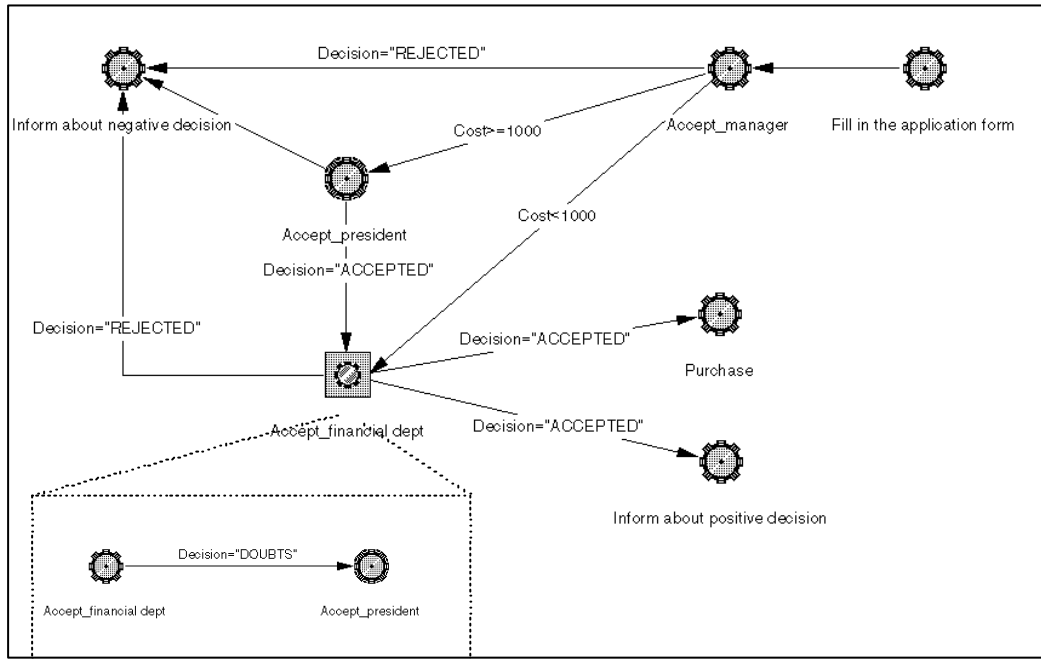


Figure 12. Purchase: the workflow process diagram