

# Workflow Languages

Mathias Weske and Gottfried Vossen

Lehrstuhl fuer Informatik, University of Muenster  
Steinfurter Strasse 107, D-48149 Muenster, Germany  
{weske,vossen}@helios.uni-muenster.de

**Summary.** We survey the requirements, concepts, and usage patterns of workflow languages which are used in today's commercial or prototypical workflow management systems. After briefly reviewing workflow application development processes, basic notions of workflow modeling and execution and their relevant properties are introduced. A coarse classification of workflow languages is presented, and the main features of common workflow languages are described in the context of a sample application process.

## 1. Introduction

Workflow management aims at modeling and controlling the execution of processes in business, scientific, or even engineering applications. It has gained increasing attention in recent years, since it allows to combine a *data-oriented* view on applications, which is the traditional one for an information system, with a *process-oriented* one in which activities and their occurrence over time are modeled and supported properly [VB96, GHS95]. Workflow management combines influences from a variety of disciplines, including cooperative information systems, computer-supported cooperative work, groupware systems, or active databases. Its major application area has so far been in the business field; as the *modeling* of business processes has become a strategic goal in many enterprises, a further step is to *optimize* or to *reengineer* them, with the goal of automation in mind. Once the modeling and specification of business processes has been completed, they can be verified, optimized, and finally brought onto a workflow management system. It is here where *languages* for describing or specifying workflows, or *workflow languages* for short, enter the picture. These languages will be discussed in what follows.

Generally, workflow languages aim at capturing workflow-relevant information of application processes with the aim of their controlled execution by a workflow management system [RS95, GHS95, She96]. The information involved in workflow management is heterogeneous and covers multiple aspects, ranging from the specification of process structures to organizational modeling and the specification of application programs and their respective execution environments. We here survey the requirements, concepts, and usage patterns of workflow languages which are used in today's commercial or prototypical workflow management systems. To embed workflow languages in the context of their purpose and usage, workflow application development

processes are reviewed, and a simple application process is described which serves as our running example.

Workflow languages are yet another species of languages for human-computer interaction. In contrast to general-purpose programming languages, workflow languages are highly domain specific, i.e., they are tailored towards the specific needs of workflow applications. Moreover, computational completeness is not an issue in a workflow language, since they are not used to describe computations. While control structures play an important role in both programming languages and in workflow languages, low-level constructs are missing in workflow languages. On the other hand, workflow languages support constructs to integrate external applications, and to describe and organize their interaction, cooperation, and communication relationships. They are hence similar in nature to software specification languages, which also have to be able to describe control flow as well as data flow between modules or components. Since workflow models are used as an information basis for the modeling and optimization of application processes, it should be obvious that graphical languages play an important role.

There are numerous approaches to model related and potentially concurrent activities, which stem from different domains. A set of rigorous mathematically founded approaches have been developed in the area of distributed computing, among which process algebras play a key role, namely to formally define concurrently executing processes and their communication behavior. Important approaches are Milner's CCS [Mil80] and Hoare's CSP [Hoa85]. These approaches focus mainly on formal properties of distributed computations; since technical and organizational aspects, which are important for workflow languages, cannot be represented in these calculi, they are not discussed in further detail here.

The organization of the remainder is as follows: In Section 2. basic concepts and notions of workflow modeling are presented, and an example is provided which will serve as our running example. Since process modeling languages have been discussed elsewhere in this book, we focus on the specific aspects workflow languages have to cover. Section 3. focuses on categories of workflow languages. For each category we choose a typical language, and we show how it can be used to model the sample application process as a workflow. A summary and concluding remarks complete our survey.

## 2. Workflow Modeling

Workflow management aims at modeling and controlling the execution of complex application processes in a variety of domains, including the traditional business domain [LA94, GHS95, JB96] and the natural sciences [Ioa93, VW97]. Workflow models are representations of application processes to be used by workflow management systems for controlling the execution of workflows. Workflow languages are used to specify workflow models. Since work-

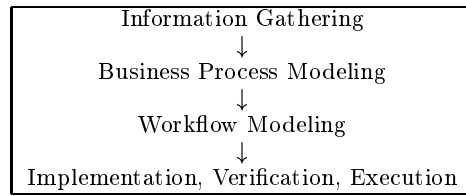


Fig. 2.1. Workflow Application Development Process.

flow modeling aims at mapping relevant information about application processes into workflow models, workflow languages need to have constructs for a variety of aspects, as explained below in Section 2.3.

## 2.1 Workflow Development Process

In general, workflow models capture the information of application processes which is relevant for the purpose of workflow management. Before workflow languages will be discussed, the general development process of workflow applications is described. While the workflow application development process differs from one project to the next, the following phases typically are involved.

The first phase of the workflow application development process, which generally shares a number of aspects and steps with a database design process or an information system development process, deals with gathering information, relevant for the application process under investigation (cf. Figure 2.1). In this phase, empirical studies like interview techniques and available documentation is used. The techniques used in this phase are mostly informal. The activities of this phase are centered around the application, and technical issues are not considered.

The next phase involves *business process modeling*, in which the information previously gathered is used to specify business process models. In this phase semi-formal techniques are used, typically some simple form of Petri net formalism, often without exploiting their formal semantics. The main purpose of business process modeling is to provide a general and easy-to-read notation, which enables information system experts and domain experts to validate and optimize business process models, an activity called business process reengineering. The result of this phase is a business process model, which is used as a basis for the next phase.

The purpose of the subsequent *workflow modeling phase* is to enhance the business process model with information needed for the controlled execution of workflows by a workflow management system. In this phase workflow languages are used. Typically, different languages are used for business modeling and workflow modeling. Hence, business process models have to be translated into the constructs of a workflow language. Notice that there are languages

that cover both phases, as discussed below. Besides the translation, information which is relevant for the controlled execution of workflows by a workflow management system is added to the model. On the other hand, information which is irrelevant for workflow executions is omitted from the business process model. Hence, workflow modeling abstracts from irrelevant information and adds relevant information, mainly of technical nature. For instance, in workflow models application programs used to perform workflow activities are specified, including their execution environment. The result of the workflow modeling phase is a workflow model, which is used by a workflow management system for controlling the execution of the workflow. We point out that the workflow development process can be iterated so that workflow execution data is used to improve business process models; it may also depend on the methods and tools used.

## 2.2 Sample Application Process

In order to keep the presentation of workflow languages concise and to provide a common basis to study and to compare different workflow languages, we now present an example of a business process from the area of credit processing in a banking environment. This example originates from the documentation of FlowMark, IBM's workflow management system [FM96]; when using the example with other workflow languages, it is modified according to the needs of the workflow language used.

Informally, the application process starts when a customer requests a credit from the bank. The customer does so by filing a credit request form and by sending it to the appropriate department in the bank. The information in the credit request form is transferred into the bank's computer system. After the validity of the data is checked, the next step involves an assessment of the risks involved in granting the credit request. Depending on the amount requested, checking activities of different complexity may be involved. We assume that the checking activity is performed by a financial expert, subject to the credit amount requested and the financial situation of the applicant. If the expert grants the credit, administrative activities to allocate the requested amount to the customer's account are launched. If it is not granted in this activity then a second, more advanced expert re-evaluates the case, possibly after getting hold of new information on the financial situation of the customer. Depending on his or her judgment, the credit is rejected or granted. In any case, the customer is informed of the decision.

While this description of a credit processing application simplifies real-world applications considerably, it provides a basis for a presentation of workflow aspects and of typical workflow languages. Notice a typical aspect of such informal descriptions, namely that errors and failures which may be encountered while the process is executed are not included. Indeed, a vastly open problem today is to specify exceptions as well as repair or compensating ac-

tions for possible errors and failures, or to build corresponding features into languages that allow the specification of normal activities in workflows.

### 2.3 Workflow Aspects

As discussed above and as indicated in the example, workflow modeling aims at specifying different aspects of the application process and of the technical and organizational environment in which the workflow will be executed. To provide modularity in workflow modeling and to refer to the different dimensions of workflow modeling explicitly, workflow aspects are described [JB96]. The description of the workflow aspects includes basic notions of workflow modeling and execution.

*Functional Aspect.* The functional aspect covers the functional decomposition of activities as present in application processes, i.e., it specifies which activities have to be executed within a workflow. To deal with the high complexity of application processes, the concept of nesting is used to describe the functional aspect of workflows. In particular, workflows are partitioned into complex and atomic workflows, where complex workflows are composed of a number of (complex or atomic) workflows. Due to their relative position, the components of a complex workflow are known as subworkflows. Hence, workflows typically have a tree structure, such that the root node of the tree represents the top-level (complex) workflow, the inner nodes represent other complex workflows, and the leaf nodes represent atomic activities. While different approaches to workflow modeling denote the entities of the functional aspect differently, we adopt the approach that activities generally are represented by workflows, which can be complex or atomic. Synonyms for complex workflow include process, complex activity, block; atomic workflows are also called (atomic) activities or steps.

In the sample application process, the functional aspect covers the functions performed during the process. When the credit is requested, a credit form is received by the bank. One function is entering the data from the credit request form into the system followed by searching for invalid or missing data. The functional aspect describes what has to be done during a workflow execution. It does not specify how it is done. In the sample workflow, the functional aspect does not define how the data entering and checking is done – this is covered by the operational aspect, discussed below. Constraints on the functions performed in a workflow are also not described in this aspect – these properties are defined in the behavioral aspect, discussed now.

*Behavioral Aspect.* In general, workflows consist of a set of interrelated activities. Hence, the controlled execution of a complex workflow by a workflow management system has to take into account interrelationships of the complex workflow's subworkflows. While the functional aspect does not cover the relative ordering of subworkflows, these issues are covered in the behavioral aspect. This aspect specifies under which conditions the subworkflows of a

given complex workflow are executed during workflow executions. Important components of this aspect are control flow constraints, which represent the control structure of activities of the application process. When in the application process subworkflow  $j$  can only be started after subworkflow  $i$  has terminated then a control flow constraint can be used to model this relationship. When the workflow is started, the workflow management system makes sure that activity  $j$  is started only after  $i$  has terminated. There are other forms of interrelationships between subworkflows, covered by other concepts in the behavioral aspect, for instance start conditions and termination conditions. For each subworkflow, a start condition specifies the precondition of its execution. Hence, an activity is started during a particular workflow instance only if the start condition of that activity is evaluated to ‘true’. The information specified in the behavioral aspect of workflow models is important for a workflow management system to control the execution of workflows. This aspect is covered by all workflow languages, and workflow management systems support mechanisms to guarantee that the interrelationships between workflows as defined in the behavioral aspect of workflow models are satisfied by all workflow instances.

In the sample workflow, the behavioral aspect specifies relationships between workflow activities. For instance, it specifies that entering credit form data is done before the checking for incorrect values, which in turn is performed before the risk is assessed and the decision on granting or rejecting the credit is taken. Another example of this aspect in our example is the branching of control flow depending on the amount requested. If the amount is smaller than a predefined value  $x$  then a quite simple checking procedure is applied. If the requested amount exceeds  $x$  then a more complex procedure is performed to either grant or reject the credit request. In general, the semantics of branches can be parallel, alternative, or it can be controlled by predicates which are evaluated at execution time of the workflow. An example of the latter form is discussed above, which can be specified by  $amount \leq x$  and  $amount > x$ , respectively.

*Informational Aspect.* An important aspect of workflow languages is the modeling of workflow relevant application data. Modeling data is required to permit workflow management systems to control the transfer of workflow relevant data as generated or processed by workflow activities during workflow executions. In graph-based approaches, the informational aspect includes data flow between workflow activities. In particular, each activity is assigned a set of input and a set of output parameters. On its start, an activity reads its input parameters, and on its termination it writes values it generated into its output parameters. These values can be used by follow-up activities in the workflow as input data. This transfer of data between workflow activities is known as data flow. By providing graphic language constructs to represent data flow between activities, the informational aspect can be visualized and used to validate and optimize application processes. While the basic principle

of the informational aspect is straightforward, there are many technical issues to solve, for instance different data formats of a given data flow, which may require the use of filters to allow seamless integration of different tools using different data formats. To this end, it is desirable that data as specified in a data flow is strongly typed. Clearly, this would require a typing scheme for data which occurs as parameters of workflow activities. In doing so, potential typing incompatibilities can be detected in the workflow modeling phase.

The informational aspect in the sample workflow describes the data types involved, for instance data types for customer data, credit forms and risk assessments. Besides the specification of the data types, data flow constraints between activities of a workflow are also described in the informational aspect. Data flow constraints in the sample workflow occur between the activity in which the credit form is entered into the system and follow-up activities, which use this information to decide on granting or rejecting the credit request. In addition, there is a data flow from the decision taking activity to the activity in which the customer is informed of the result of his or her credit request.

*Organizational Aspect.* Workflows are executed in complex organizational and technical environments, and a major goal of workflow management is enhancing the efficiency of application processes by assigning work to persons or software systems as specified by workflow models. To reach this goal, a workflow management system has to be provided with information on the organization and on the technical environment in which the workflows will be executed. In general, atomic workflows can be either automatic or manual. Manual atomic workflows are executed by persons who may use application programs to do so; automatic atomic workflows are executed by software systems without human involvement. Since a strict assignment of workflow activities to persons is not feasible in most cases, the role concept is used. A role is a predicate on the structure of an organization in which the workflow is executed. When an activity is about to start, the system uses predefined role information to select one or more persons which are permitted, competent and available to perform the requested activity. The process of selecting one or more persons to perform a workflow activity is known as role resolution. Depending on the scope of workflow management systems, the role concept has different complexity. While some systems support a simple role concept others provide additional features for substitution of persons or they take into account the overall structure of the organization to select persons to perform activities during workflow executions.

People involved in the execution of the sample workflow are part of the bank's credit department. Activities of the sample workflow are scheduled to persons in the department according to their positions, which are specified by roles. Clerk, financial expert, and credit expert are sample roles. While the data entering is done by clerks, the decision on granting requested credits is done by financial experts, capable of assessing the risks and of deciding on

the credit, provided the requested amount is below a predefined margin. The assignment of workflow activities to persons is done by role resolution, for example the data entering is done by a clerk while the assess credit activity is performed by a financial expert. The role concept can be enhanced to allow context sensitive features, e.g., a person is selected to perform an activity of a credit workflow which the person previously has decided on. In this case, workflow execution data is used to allow more complex role resolution.

*Operational Aspect.* The integration of existing tools and application programs into workflow applications is an important feature of workflow management systems. The information required is specified in the operational aspect. The operational aspect covers mainly technical issues, like the invocation environment of application programs (including host and directory information of the executable program), the definition of the input and output parameters of the application program and their mapping to input and output parameters of workflow activities. As described above, persons are selected by role resolution to perform workflow activities. When a person chooses to perform an activity then the defined application program is started, and the input data as specified in the workflow model is transferred to that application program. When the person completes that activity, the output data generated by that activity is collected in the output parameters of the activity to be transferred by the workflow management system to the next workflow activity, as specified in the respective workflow model. Notice that during business modeling, no information on the operational part is (and needs to be) present. Business process modeling aims at mapping high level and domain specific features of the application process; the technical details – the main components of the operational aspect – are taken into account in the workflow modeling phase.

In the banking example, different information systems are used to perform different tasks. Entering customer and credit request data by clerks is typically done by forms-based software systems as front-ends of an integrated data repository. The activities of assessing the risk of a credit may involve other information systems, some of which may reside remotely. In this case, the execution environment includes detailed information which allows the workflow management system to invoke the desired applications in the respective sites, using different kinds of middleware technology.

*Flexibility Aspect.* Recently, the need to enhance the flexibility of workflow applications arose in different application areas [EKR95, VW97, RD98]. Starting from applications in non-traditional domains like the natural sciences or hospital environments, flexibility also became an issue in business applications. Providing flexibility to workflow applications is based on the understanding that during workflow modeling not all aspects of the application process can be specified completely. There may be unforeseen situations during workflow executions, which require flexible reactions by the user or administrator of the system. Hence, additional features to model workflows

and additional functionality to support the functionality is required by workflow management systems to deal with flexibility issues. We believe that the future success of workflow management systems to a large extent depends on the way workflow model changes or changes to the organizational or technical environment are supported in a user-friendly way. There are different forms of flexibility, ranging from the change of role information and application program information to the change in the functional and behavioral aspects of workflows. Adding an activity to a complex workflow while the workflow executes corresponds to a dynamic change in the functional aspect; changing the control structure of subworkflows of a given workflow (e.g., parallel execution of workflow activities, originally defined to be executed sequentially) corresponds to the change in the behavioral aspect. Providing user intervention operations to allow users to skip, stop or repeat subworkflows is another form of flexibility in the behavioral aspect. A change of role information and of application program information changes the organizational and operational aspects, respectively. We remark that supporting flexibility has to be supported by the workflow language and also by the workflow management system, supporting the respective functionality. For instance, workflow languages should allow to specify which activities can be skipped or repeated, and how data issues due to deleting workflow activities which would generate required data are solved.

Although the general structure of the sample credit request workflow is static, numerous unforeseen events may occur during workflow executions, which require flexible reactions. For instance, assume while a credit request is processed, the applicant comes into an inheritance. This changes the financial situation of the applicant considerably, which may require a re-evaluation of the credit request. On the side of the customer, the inheritance may lead to canceling the credit request. In this case, the workflow has to be canceled, and steps already executed on its behalf have to be undone, for instance the allocation of funds to the customer. Simpler forms of flexibility occur when it comes to changes in role information or in application programs used to process workflow activities.

### 3. Workflow Languages

In general, workflow languages can be classified according to their underlying methodologies and underlying meta models. A meta model describes the constructs and their relationships of workflow models of particular workflow languages. An important class of workflow languages are graph-based languages, which allow the specification of workflows using different forms of directed graphs. While the functional and behavioral aspects can be specified using graph notation, the informational and operational aspects require additional specifications, like data types of transferred data objects or information on the execution environment of application programs. This information can be

provided textually, often supported by workflow management systems using forms interfaces. Hence, the categories discussed below do not indicate that all workflow aspects are specified using the respective notation. The second category of workflow languages use the Petri nets approach to specify workflow models. Petri nets have widely been used to specify the behavior of a dynamic system with a fixed structure.

Besides these classes of workflow languages, script languages are widely used. Often, these languages are closely related to workflow management system development. Workflow languages can also have multiple representations. For instance, there may be a graphical language for the specification of workflow models, which is translated into a script language, to be processed by a workflow management system. An example of this strategy is provided in the remainder of this section. State and activity charts, originally developed to model reactive systems, are used to model workflows. We discuss this approach briefly.

Due to space limitations, we restrict ourselves to workflow languages which are currently used in workflow management systems, commercially available or prototypical. In particular, graph-based languages, net-based languages and script language approaches are considered in some depth; the state and activity chart approach is discussed briefly. Further approaches to workflow languages, like speech act theory are not widely used in workflow management systems and are therefore not discussed here.

### 3.1 Graph-based Languages

Graph-based languages allow to specify workflow activities, their hierarchical relationships and their data flow and control flow constraints using directed graphs. These graphs are enhanced to cover the workflow aspects presented above. Workflow graphs are nested, such that each node can be refined into a subgraph, known as a subworkflow. In addition, there are two forms of directed edges, i.e., control flow edges and data flow edges. Control flow edges belong to the behavioral aspect, while data flow edges belong to the informational aspect. In particular, a control flow edge  $i \rightarrow j$  specifies that activity  $j$  can start only after activity  $i$  has terminated. Data flow edges specify data dependencies between workflow activities; if activity  $i$  generates data which is required as input to activity  $j$  then there is a data flow edge connecting these activities. Explicit modeling of data flow between workflow activities is an important means to describe interrelationships between workflow activities due to the generation and use of data.

*Workflow Process Definition Language.* The Workflow Management Coalition (WfMC) is a consortium of workflow vendors, users and researchers, aiming at promoting the use of workflow technology in business organizations [WfMC96a]. By specifying a set of interfaces of workflow management systems, the WfMC builds a framework to enhance the interoperability of

workflow systems of different vendors. In their effort, the WfMC loosely specifies a workflow language which is based on graphs. Since the language has to be supported by the systems the WfMC members develop and use, the language is described in an abstract way using graph notation. Activities are represented by nodes; control flow is defined by explicit control flow constructs, for instance (AND, OR, XOR) split and the respective join nodes [WfMC96b]. However, the WfMC is focused primarily on technical issues of workflow system interoperability. Instead of defining a complete workflow language which is mandatory for all workflow system vendors which are members in the coalition, it adopts the strategy that different workflow products are free to use different workflow languages.

*FlowMark Workflow Language.* One of the first workflow management systems that reached the market is IBM's FlowMark. This paragraph discusses the graphical workflow language used in FlowMark, while its textual representation (FlowMark Definition Language, FDL) is sketched below. To describe its workflow language, the main components of the FlowMark workflow meta model have to be described [LA94, FM96]. The main entity of the FlowMark workflow meta model is the activity, which can either be complex or atomic. Complex activities are called processes, while atomic activities are called program activities, typically implemented by application programs. Processes are composed of a number of activities with accompanying control flow and data flow constraints. A set of activities can be grouped using the block construct. The activities in a block are executed repeatedly until an end condition of the block signals its termination. With regard to the informational aspect, each activity has an input container and an output container, consisting of a set of input parameters and output parameters, respectively. In FlowMark, data flow is specified by connecting an output parameter of an activity  $i$  to an input parameter of an activity  $j$ . This data flow is permitted only if there is a path of control flow edges from  $i$  to  $j$ . Control flow is defined by control flow edges. Each control flow edge is assigned a transition condition, which is a predicate to be evaluated at runtime. When an activity terminates, the transition conditions of all outgoing control flow edges of that activity are evaluated. Depending on the value, the control flow edge is fired with either 'true' or 'false'. The start of activities is governed by start conditions. In general, start conditions of an activity can be evaluated if and only if all incoming control flow edges have been fired. When a start condition evaluates to 'true', the respective activity can be launched. To guarantee that control flow edges fired with 'false' do not hamper the start of workflow activities, a technique called *dead-path-elimination* is performed [LA94]. Workflow models in FlowMark are generally acyclic; loops can be modeled using the block construct, as discussed above.

The sample workflow can be specified graphically in the FlowMark system as shown in Figure 3.1. Activities are represented by nodes; data flow is represented by dotted lines, and solid lines are control flow edges. Control

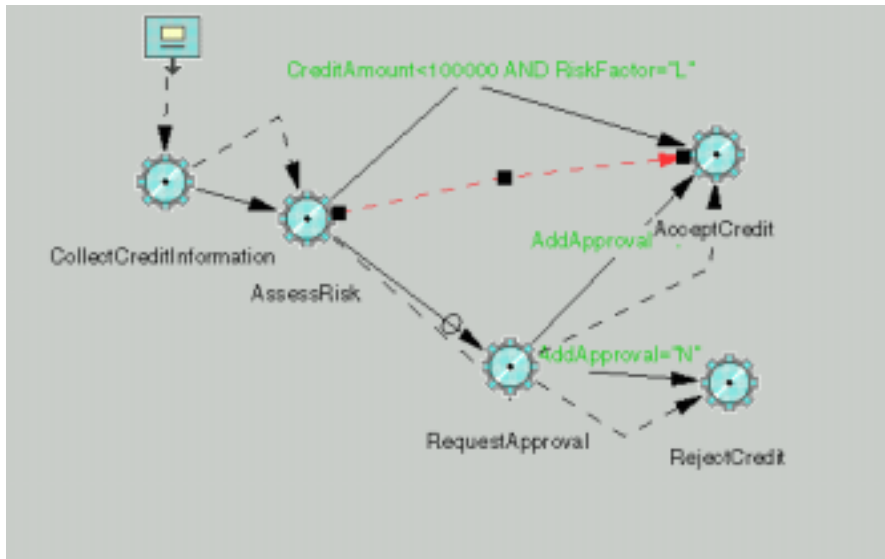


Fig. 3.1. FlowMark: Sample Workflow Model.

flow edges are labeled with transition conditions, which are evaluated on termination of the source node of the respective control flow edge. For instance, to model that a credit request can be granted if less than 100 K\$ is requested and if the risk factor determined by the *AssessRisk* activity is low, the transition condition from the *AssessRisk* to the *AcceptCredit* activity in Figure 3.1 is labeled “*CreditAmount* < 100000 AND *RiskFactor*=”L””. Notice that the *AssessRisk* activity has *CreditAmount* and *RiskFactor* as output parameters. In an alternative form of data flow, a complex workflow can transfer data to its subworkflows. Analogously, data can be transferred from subworkflows to their respective superworkflows. In these cases, the data flow is vertical rather than horizontal (data flow between subworkflows of a common superworkflow is called horizontal data flow). A fragment of a textual representation of the sample workflow using FDL is discussed shortly.

*WASA*. The workflow language in the *WASA* project [VW97] on flexible workflow management is based on the FlowMark workflow language. Since the *WASA* project aims at enhancing the flexibility of workflow management systems, the language supports the flexibility aspect, discussed above [VW97, Wes98]. For instance, for each workflow model the user may specify if it can be skipped, stopped or repeated during particular workflow executions. Skipping workflow activities may lead to missing data. Hence, the workflow language specifies how these data issues are solved by providing the appropriate language constructs. In addition to enhancing the workflow language w.r.t. flexibility, the workflow management system has to support

flexibility operations, e.g., dynamic modeling operations and user intervention operations. With dynamic modeling operations, workflow models of running workflow instances can be changed to reflect changes in the environment of the process. Changes can apply to the changing workflow instance only, or changes may apply to all workflow instances of the changed workflow model. User intervention operations allow the user to perform changes to the control flow structure of the workflow for a particular workflow instance.

*ADEPT.* In the ADEPT framework, workflow models are specified as symmetric graphs with special workflow relevant nodes [RD97, RD98]. Branching nodes are explicitly marked as AND split, OR split, XOR split; these nodes are followed by the respective join nodes. Based on this framework, a complete and minimal set of change operations are specified to define the ADEPT<sub>flex</sub> framework. In this framework, change operations to the structure of running workflows can be performed by users in a controlled manner. For instance, enhancing a workflow model with an activity involves the embedding of the added activity into the workflow model. This is specified by defining a set of activities which have to be completed before the added activity can start and a set of activities which can only be started after the added activity as terminated. The monitoring of data flow constraints between activities is also supported by the ADEPT<sub>flex</sub> framework.

### 3.2 Net-based Languages

We now elaborate on Petri nets, which are specifically tailored towards the requirements of workflow modeling and execution.

*FUNSOFT nets.* FUNSOFT nets are based on higher Petri nets and enhances them to incorporate different workflow aspects [Gru91]; FUNSOFT nets are structured as follows: The node set is partitioned into places and transitions. Each place may include one or more typed data objects. Workflow activities are represented by transitions. Controlling workflow instances is done by passing documents and information between activities. The traditional Petri net formalism is enhanced with special constructs, e.g., there is a special form of transitions to represent alternative execution paths, represented by a ‘switch’ transition.

The basic idea of this approach is the integration of different workflow aspects into a single formalism, namely FUNSOFT nets. This concept has implications on its usability. In particular, it allows to use a single formalism in different phases of the workflow application development process, i.e., FUNSOFT nets can be used in business process modeling, in workflow modeling and in workflow execution. These nets provide a graphical notation to model the control structure of application processes; organizational modeling is also supported by mapping role information to the net formalism. By providing appropriate means to specify external application programs to be used in the workflow executions, the operational aspect is also covered by



Credit Request is transferred to the Request Approval activity. The workflow continues as specified in Figure 3.2.

*Flow Nets.* Ellis et al. present the Flow Net formalism [EKR95]. Based on higher level Petri nets, their focus is on providing flexibility, namely by allowing the change of Flow Nets at runtime. Flow Nets do not provide formalisms for the operational aspect or organizational aspects like tool integration or role management, respectively. The focus of this approach is on the specification of the control flow structure of workflows as Flow Nets and their use to control the execution of workflows in the presence of dynamic modification operations. In particular, dynamic modification in Flow Nets is done by substituting subnets with other subnets, governed by rules for the correct substitution and embedding of subnets into the Flow Net, representing a workflow model.

### 3.3 Workflow Programming Languages

Workflow programming (or script) languages are often used in projects where system development issues play a major role. Workflow programming languages are either used directly to specify workflow models or they are used as an internal representation with the aim of the controlled execution by a workflow management system or to allow the import and export of workflow models. One approach of the first form is the Mobile approach – the second form of workflow script language is present in the FlowMark workflow management system, whose graphical workflow language was sketched above.

*Mobile.* In the Mobile workflow management system [JB96], workflow models are specified using the Mobile script language. The Mobile project aims at supporting different workflow aspects in a modular way. This goal is reflected in the Mobile workflow language by supporting constructs for the definition of different workflow aspects. Besides the focus on workflow aspects, the Mobile workflow language provides extensibility. In particular, based on a set of pre-defined control flow operators, the user can define new control flow constructs to support the specific requirements of particular workflow applications. For instance, constructs to execute a set of activities in any sequential order can be specified. From a system development point of view, each workflow aspect is covered by a server devoted to keeping track of workflows w.r.t. its particular aspect. The aim of this conceptual design and system architecture is to provide the system administrator with facilities and tools to use the aspects which are important for the particular workflow applications and to be able to extend the system with additional aspects as they are required.

An incomplete specification of the sample workflow using the Mobile workflow language is given below. Each workflow aspect is represented by language keywords and accompanying language constructs. Workflow models are specified in sections, delimited by `WORKFLOW_TYPE` and `END_WORKFLOW_TYPE` keywords. Analogously, the behavioral aspect is described in a section delimited

by CONTROL\_FLOW and END\_CONTROL\_FLOW. The set of constructs in this section includes sequential execution and branching, represented by the `sequence` and `ifthen` constructs, respectively. A complete workflow specification of the sample workflow can be found in [JBS97].

```

WORKFLOW_TYPE CreditRequest (IN PersonInfo: CreditRequestor)
/* definition of subworkflows */
WORKFLOW_DATA CreditInfo: c
END_WORKFLOW_DATA

CONTROL_FLOW
sequence (CollectCreditInfo,
sequence (AssessRisk,
ifthen (c.CreditAmount < 100000 AND c.RiskFactor == "L",
AcceptCredit,
sequence (RequestApproval, ifthen (c.AddApproval == "Y",
AcceptCredit, RejectCredit))))))
END_CONTROL_FLOW

DATA_FLOW
CreditRequest.CreditRequestor -> ci.CreditRequestor;
AssessRisk.out_ci -> AcceptCredit.in_ci;
END_DATA_FLOW
/* definition of organizational aspect */
END_WORKFLOW_TYPE

```

*FlowMark Definition Language (FDL)*. While the FlowMark system presents a graphical interface to the user, there is an internal workflow language which is used as an interface to import or export workflow models. Fragments of the FDL specification of the sample workflow are shown below:

```

STRUCTURE 'CreditInfo'
'CreditRequestor': 'PersonInfo';
'Address': STRING;
'RiskFactor': STRING;
'AddApproval': STRING;
'CreditAmount': LONG;
END 'CreditInfo'

PROCESS 'CreditRequest' ('PersonInfo')
PROGRAM_ACTIVITY 'AcceptCredit' ('CreditInfo')
PROGRAM 'NAcceptCredit'
DONE_BY STARTER_OF_ACTIVITY 'CollectCreditInformation'
END 'AcceptCredit'

PROGRAM_ACTIVITY 'AssessRisk' ('CreditInfo', 'CreditInfo')
PROGRAM 'NAssessCreditRisk'
DONE_BY STARTER_OF_ACTIVITY 'CollectCreditInformation'
END 'AssessRisk'

PROGRAM_ACTIVITY 'CollectCreditInformation'

```

```

    ('PersonInfo', 'CreditInfo')
PROGRAM_ACTIVITY 'RejectCredit' ('CreditInfo')
PROGRAM_ACTIVITY 'RequestApproval'
    ('CreditInfo', 'CreditInfo')

CONTROL FROM 'CollectCreditInformation' TO 'AssessRisk'
CONTROL FROM 'AssessRisk' TO 'AcceptCredit'
    WHEN 'CreditAmount<100000'
CONTROL FROM 'RequestApproval' TO 'RejectCredit'
    WHEN 'AddApproval="N"'
CONTROL FROM 'RequestApproval' TO 'AcceptCredit'
    WHEN 'AddApproval="Y"'
CONTROL FROM 'AssessRisk' TO 'RequestApproval'
    OTHERWISE
END 'CreditRequest'

```

*State and Activity Charts.* The statechart formalism is an extension of finite state machines; it was developed by Harel [Har88] to specify the behavior of reactive technical systems. To describe such systems, statecharts specify potentially nested states and state transitions, while accompanying activity charts describe events that may trigger state transitions. Provided with a formal semantics and with a commercially available tool (Statemate [HP96]), statecharts are widely used in designing technical systems, like remote control systems or car radio systems; they are also popular in software engineering environments for system specification. One of the first workflow management systems to exploit the formalism is the Mentor project, where state and activity charts have been used to model workflows [WWWK96]. In terms of workflow aspects, statecharts describe the informational aspects while activity charts describe when state transitions are performed and which activities are launched when a particular state transition occurs. Hence, activity charts define the behavioral aspect. The separation of control flow and data flow in state and activity charts can lead to control structures which are not easily understandable by application domain experts. On the other hand, the statechart formalism provides techniques and tools to formally prove properties of statecharts. These properties are used in the Mentor project to formally prove that the execution of workflows in centralized and distributed environments are equivalent [Wod96].

## 4. Conclusions and Summary

We have discussed general design principles of workflow languages. Starting from general workflow notions, we have described a variety of aspects relevant to workflow management. A sample application process has been provided, and a set of workflow languages was described by presenting their underlying methodology; they have been used to model the sample application process as a workflow.

Due to space limitations, our selection of workflow languages is by no means exhaustive. We have tried to present the major categories of workflow management which are used in today's workflow management systems. For each category, a specific language has been presented in some detail using the sample workflow.

Current research issues in workflow languages focus on usability studies, on flexibility issues, and on correctness properties of workflow models. The latter may require workflow languages to represent additional properties which are not yet found in current languages. Today's workflow languages require a high degree of specialization on the user's side; in other words, workflow modeling has been done by experts who are well familiar with the respective workflow language used. This situation is similar as it was with early database design tools 20 years ago; with database systems becoming a mass product, their design tools have been simplified such that nowadays even non-experts can get a grasp on them. We expect a corresponding development for workflow languages, in particular since the details of an operational workflow are often in the heads of the end-users, so that it is crucial to have them participate more heavily in the description and specification phase of a workflow they are about to become involved in.

Regarding executions of workflows, current workflow languages are still rudimentary with respect to a distinction between transactional and non-transactional tasks, and with respect to recovery issues. One reason for this may be seen in the fact that the proper exploitation of transactional concepts in the context of workflows and their executions is still under heavy discussion [WS97]. On the other hand, there has been positive experience with using a transaction specification framework for describing execution aspects of workflow instances [SK97, Dog97]; it may therefore be expected that future workflow languages will also provide transactional capabilities. In addition, enhancing the flexibility in workflow management systems is a current research topic. It is expected that this topic will lead to new developments in workflow languages and in methods and tools to prove correctness properties of workflow models and workflow executions.

## References

- [Dog97] Dogac, A., et al. *Design and Implementation of a Distributed Workflow Management Systems: METUFlow*. NATO ASI Workshop, Istanbul, August 12–21, 1997. To appear in Springer ASI NATO Series.
- [EKR95] Ellis, C., K. Keddera, G. Rozenberg. *Dynamic Change Within Workflow Systems*. In Proc. Conference on Organizational Computing Systems (COOCS), Milpitas, CA 1995, 10–22.
- [GHS95] Georgakopoulos, D., M. Hornick, A. Sheth. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995.

- [Gru91] Gruhn, V. *Validation and Verification of Software Process Models*. Ph.D. Thesis, University of Dortmund, 1991. Available as Technical Report No. 394/1991, University of Dortmund, Germany.
- [Har88] Harel, D. *On Visual Formalisms*. Communications of the ACM 31, 1988, 514–530.
- [HP96] Harel, D., M. Politi. *Modeling Reactive Systems with Statecharts: The State-mate Approach*. Part No. D-1100-43, i-Logix Inc., Andover, MA 01810, 1996.
- [Hoa85] Hoare, C.A.R. *Communicating Sequential Processes*. Prentice-Hall, 1985.
- [FM96] IBM. *IBM FlowMark: Modeling Workflow, Version 2 Release 2*. Publ. No SH-19-8241-01, 1996.
- [Ioa93] Ioannidis, Y. (ed.). *Special Issue on Scientific Databases*. Data Engineering Bulletin 16 (1) 1993.
- [JBS97] Jablonski, S., M. Böhm, W. Schulze. (Editors) *Workflow Management: Development of Applications and Systems. (in German)* dpunkt-Verlag, 1997.
- [JB96] Jablonski, S., C. Bußler. *Workflow-Management: Modeling Concepts, Architecture and Implementation* International Thomson Computer Press, 1996.
- [LA94] Leymann, F., W. Altenhuber. *Managing Business Processes as an Information Resource*. IBM Systems Journal 33, 1994, 326–347.
- [Mil80] Milner, R. *A Calculus of Communicating Systems*. Springer LNCS 92, 1980.
- [RD97] Reichert, M., P. Dadam. *A Framework for Dynamic Changes in Workflow Management Systems*. Proc. 8th International Workshop on Database and Expert Systems Applications 1997, Toulouse, IEEE Computer Society Press, 42–48.
- [RD98] Reichert, M., P. Dadam. *ADEPT<sub>flex</sub> - Supporting Dynamic Changes of Workflows Without Loosing Control*. To appear: Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management, Vol. 10, No. 2, 1998.
- [RS95] Rusinkiewicz, M., A. Sheth. *Specification and Execution of Transactional Workflows*. In Kim Won (editor): *Modern Database Systems The Object Model, Interoperability, and Beyond*, Chapter 29, 592–620. ACM Press, 1995.
- [She96] Sheth, A., D. Georgakopoulos, S.M.M. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, A. Wolf. *Report from the NSF Workshop on Workflow and Process Automation in Information Systems*. Technical Report UGA-CS-TR-96-003 University of Georgia, Athens, GA, 1996.
- [SK97] Sheth, A., K.J. Kochut. *Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems*. NATO ASI Workshop, Istanbul, August 12–21, 1997. To appear in Springer ASI NATO Series.
- [VB96] Vossen, G., J. Becker. (eds.) *Business Process Modeling and Workflow Management: Models, Methods, Tools (in German)*. International Thomson Publishing, Bonn, Germany, 1996.
- [VW97] Vossen, G., M. Weske. *The WASA Approach to Workflow Management for Scientific Applications*. NATO ASI Workshop, Istanbul, August 12–21, 1997. To appear in Springer ASI NATO Series.
- [Wes98] Weske, M. *Modeling and Execution of Flexible Workflow Activities*. To appear in Proc. 31st Hawaii International Conference on System Sciences, IEEE Computer Society Press, 1998.
- [WWWK96] Wodtke, D., J. Weissenfels, G. Weikum, A. Kotz Dittrich. *The Mentor Project: Steps Towards Enterprise-Wide Workflow Management*. In Proc. 12th IEEE International Conference on Data Engineering (1996), 556–565.
- [Wod96] Wodtke, D. *Modeling and Architecture of Distributed Workflow Management Systems. (in German)* Ph.D. Thesis, University of Saarbruecken, 1996.

- [WS97] Worah, D., A. Sheth. *Transactions in Transactional Workflows*. In S. Jajodia, L. Kerschberg (eds.), *Advanced Transaction Models and Architectures*, Kluwer Academic Publishers, 1997, 3–33.
- [WMC96a] Workflow Management Coalition. *Workflow Handbook 1997*. John Wiley in association with Workflow Management Coalition (WfMC), 1996.
- [WMC96b] Workflow Management Coalition. *Terminology & Glossary*. Document Number WfMC-TC-1011, 1996. (Available from <http://www.aiai.ed.ac.uk:80/project/wfmc>)