

Workflow and Service Composition Languages

Mathias Weske¹, Gottfried Vossen², and Frank Puhlmann¹

¹ Hasso-Plattner Institute for Software Engineering, University of Potsdam,
Germany, {weske, puhlmann}@hpi.uni-potsdam.de

² European Research Center for Information Systems, University of Münster,
Germany, vossen@uni-muenster.de

Summary. We survey the requirements, concepts, and usage patterns of modern workflow languages which are employed in today's workflow management systems as well as in Web service environments. 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. Workflow nets are introduced as well as their extension for improved coverage of workflow patterns: YAWL (Yet Another Workflow Language). In Web services composition, the *Business Process Execution Language for Web Services* and the graphical *Business Process Modeling Notation* are described. These workflow languages are illustrated by a common sample business process from the financial sector.

1.1 Introduction

Workflow management aims at modeling and controlling the execution of processes in business, scientific, or even engineering applications. It has gained increasing attention during the 1990s, 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. The benefits of process modeling and workflow management have meanwhile been recognized in almost every application domain, and most recently the emerging field of *Web services* has increasingly been based on workflow concepts.

A major application area of workflow management has originally 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 (or process) languages aim at capturing workflow-relevant information of application processes with the aim of their controlled execution by a workflow management system [GHS95, AWW03]. The information involved in workflow management is heterogeneous and covers multiple aspects, ranging from the specification of process structures to organizational and data 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 workflow or process 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 essentially yet another species of languages for human-computer interaction. In contrast to general-purpose programming languages, workflow languages can be highly domain specific, i.e., they can be 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. However, a precisely defined semantics *is* an issue even in a workflow language, since otherwise there would be no guidance to how to employ or use the elements provided by the language. While control structures play an important role in both programming languages and in workflow languages, low-level constructs are typically 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 of this chapter is as follows: In Section 1.2 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 are also discussed elsewhere in this book, we focus on the specific perspectives workflow languages have to cover. Section 1.3 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; this survey will also take us into the emerging field of Web services [CD02, H+03, A+04], where several specification and description languages are currently undergoing standardization [New02]. A summary and concluding remarks complete our survey.

1.2 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 [GHS95, LR00], the emerging field of electronic learning [VJO02, G+04], or the natural sciences [Ioa93, VW98, VW99]. Workflow models are representations of application processes to be used by workflow management systems for controlling the automated execution of workflows. Workflow languages are used to specify workflow models. Since workflow modeling aims at mapping relevant information about application processes into workflow models, workflow languages need to have constructs for a variety of perspectives, as explained below in Section 1.2.3. We mention as an aside that from a technical point of view there is a distinction between *process languages* and *workflow languages*, where the former emphasize the application logic, while the latter specify the portions of this logic that can be put under WFMS control; we mostly discuss the latter here.

1.2.1 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 a workflow application development process, which generally shares a number of perspectives 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 1.1). 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.

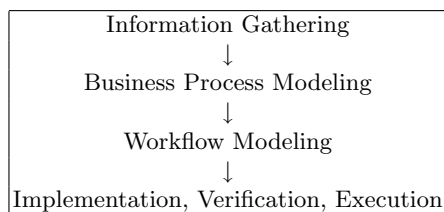


Fig. 1.1. Workflow Application Development Process.

The next phase involves *(business) process modeling*, in which the information previously gathered is used to specify process models, which often are *business* process models, i.e., deal with the processes carried out in a given business. 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. Often the business process model established in this phase is complemented by a *data model* describing the underlying objects on which the process components are based and which are consumed or produced by the various activities; finally, it may also comprise an *organizational model* specifying the organizational structure of the underlying business as well as resources that are available at the various departmental levels for carrying out activities. A typical example of this three-part approach is the *INCOME methodology* as originally described in [LNO89].

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.

1.2.2 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 or brokering environment. Informally, a new workflow in this application starts when a customer requests a credit offer from the bank or from his or her credit broker; we here assume that a broker is used. The customer has to specify his or her requirements or needs in some form to the broker; this information is then used to assign a credit rating, which is transmitted to two distinct financial institutions in order to obtain credit offers. One of these institutions is our house bank which always responds in a timely fashion, while the other is freely chosen from a number of available options; for this one, a deadline is defined until which a response is expected. After having been informed by the broker about the available options, the customer chooses the lowest offer, and is then connected with the respective institution through his or her broker. Although we could in principle iterate over the subprocess of hunting for offers, we will here assume for simplicity that one of the two offers is already acceptable to the customer.

Note that several subworkflows are hidden in this general description; for example, the broker is addressing two institutions for their credit offerings, and in the end, the one approached by the new customer needs to check the validity of the customer's data (e.g., employer and salary information). Once granted, administrative activities to allocate the requested amount to the customer's account need to be launched.

While this description of a credit application simplifies real-world applications considerably, it provides a basis for a presentation of workflow perspectives and of typical workflow languages. Notice a typical perspective of such an informal description, namely that errors and failures which may be encountered while the process is executed are not included, at least not to a full extend. Indeed, a vastly open problem today is to specify exceptions as well as repair or compensating actions for possible errors and failures, or to build corresponding features into languages that allow the specification of normal activities in workflows.

1.2.3 Workflow Perspectives

As discussed above and as indicated in the still informal example, workflow modeling aims at specifying different perspectives 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 perspectives* are described in this section. The description of the workflow perspectives also includes basic notions of workflow modeling and execution.

Functional Perspective

The functional perspective 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 perspective of workflows, and is typically applied across multiple levels. 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 perspective 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 our sample application process, the functional perspective covers the functions performed during the process. When the credit estimate is requested from the broker, a form needs to be filled out and submitted. One function is gathering information from the two credit institutions, which may be different for each addressee. The functional perspective describes what has to be done during a workflow execution. It does not specify how it is done. In the sample workflow, the functional perspective does not define how the data entering and checking is done – this is covered by the operational perspective, discussed below. Constraints on the functions performed in a workflow are also not described in this perspective – these properties are defined in the behavioral perspective, discussed next.

Behavioral Perspective

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 perspective does not cover the relative ordering of subworkflows, these issues are covered in the *behavioral* perspective. This perspective specifies under which conditions the subworkflows of a given complex workflow are executed during workflow executions. Important components of this perspective 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, 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 perspective, for instance start as well as 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 perspective of workflow models is important for a workflow management system to control the execution of workflows. This perspective is covered by all workflow languages, and workflow management systems commonly support mechanisms to guarantee that the interrelationships between workflows as defined in the behavioral perspective of workflow models are satisfied by all workflow instances.

In the sample workflow, the behavioral perspective specifies relationships between workflow activities. For instance, it could specify the branching of control flow depending on the amount requested or on the bank finally chosen. 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.

Informational Perspective

An important perspective of workflow languages, as mentioned, 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 perspective includes data flow between workflow activities. In particular, each activity is assigned a set of input and a set of output parameters. Upon its start, an activity reads its input parameters, and upon 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 (such as the state information in a Petri net), the informational perspective can be visualized and used to validate and optimize application processes. While the basic principle of the informational perspective 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 perspective in the sample workflow describes the data types involved, for instance data types for customer data or credit offerings.

Besides the specification of the data types, data flow constraints between activities of a workflow are also described in the informational perspective. Data flow constraints in the sample workflow occur between the credit offering of a particular bank and follow-up activities, which use this information to decide on an offer transported by the broker.

Organizational Perspective

Workflows are often 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 “resources,” i.e., 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 organizational as well as 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 often 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 our sample workflow are the broker, the customer issuing the request, and eventually even people in the credit department of the bank finally chosen. Activities of the sample workflow are assigned to these resources as appropriate, eventually as specified by roles. The assignment of workflow activities to persons at runtime is done by *role resolution*. 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 Perspective

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 perspective. The operational perspective 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 perspective — are taken into account in the workflow modeling phase.

In the credit brokering 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 activity of assessing a credit offer 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 Perspective

The need to enhance the flexibility of workflow applications originates from different application areas [VW98, VW99, RD98]. More recently, Rinderle et al introduced a survey on flexible workflow management [RRD04]; van der Aalst et al compared a number of workflow language approaches with respect to advanced features, including flexibility aspects.

Starting from applications in non-traditional domains like the natural sciences or hospital environments, flexibility also became an issue in business applications and is nowadays a well-understood requirement. Providing flexibility to workflow applications is based on the observation that during workflow modeling often not all perspectives of the application process can be specified completely. Indeed, 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. The 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 perspectives of workflows. Adding an activity to a complex workflow while the workflow executes corresponds to a dynamic change in the functional perspective; 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 perspective. Providing user intervention operations to allow users to skip, stop or repeat subworkflows is another form of flexibility in the behavioral perspective. A change of role information and of application program information changes the organizational and operational perspectives, 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 brokering workflow is static, numerous unforeseen events may occur during workflow executions, which require flexible reactions. For instance, assume while a credit request is processed at a bank, the applicant wins the lottery. This changes his or her financial situation of the applicant considerably, which may require a re-evaluation of the credit request, or the customer may withdraw the requests completely. In the latter 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.

1.3 Workflow Languages

Based on the workflow concepts introduced above, this section describes some of the most influential current workflow languages. Since workflow technology plays an increasingly important role in composing existing service in service-oriented environments, a service composition language is also discussed. In particular, Workflow nets and the Business Process Execution Language for Web Services are introduced as well as a graphical notation for business processes. Workflow nets are a formal concept to model workflows that is based on higher Petri nets and that extends them with additional properties and syntactic constructs for efficient representation of workflows.

Service composition is a promising approach to facilitate a major shift in workflow technology, i.e., from applications within organizations to applications between organizations. In the workflow context, this development was coined interorganizational workflow [AW01]. In a conceptual level, results in this area include the Public-to-Private approach [AW01], as well as work on contracts between cooperating organizations, as conducted in the Crossflow

project [HLGA01]. In the software side, the quest for supporting business processes spanning organizations was supported by the new Service Oriented Architecture (SOA) paradigm. Since Web services are the current implementation of SOA, Web services composition is often regarded as a means to facilitate interorganizational workflows. In this context, workflow activities are implemented by Web services that are provided by the organizations participating in the interorganizational workflow.

While a number of approaches have been discussed, industry now seems to have agreed on the *Business Process Execution Language for Web Services* (BPEL) as a common standard. While there is some controversy on the language design of BPEL [Aal03] it can be seen as a major milestone in service composition. BPEL is discussed in Section 1.3.3. Since BPEL lacks a graphic notation, the recently introduced Business Process Modeling Notation (BPMN) proposes language constructs that facilitate the modeling of workflows at a high level. A number of BPMN constructs can be directly mapped to BPEL which means that they can be executed by a BPEL workflow engine.

1.3.1 Workflow nets and YAWL

As mentioned above, Workflow nets have been derived from high-level Petri nets [AHH94]. The advantages of using Petri nets for workflow modeling is their ability for comprehensible visualization of business processes combined with expressive power as well as the availability of formal analysis and verification techniques. The formal foundations of Petri nets allow the automated proving of criteria for workflow nets, for instance absence of deadlock. Furthermore, Workflow nets or variants of Workflow nets can be used as input for a workflow management system which also provides the other workflow perspectives introduced above [AH02].

Workflows are mapped to Petri nets as follows: A transition in a Workflow net represents an activity of a workflow. The places of the net may contain tokens; each token represents a specific case in the current workflow, for instance a credit request from a particular customer. Workflow nets use the Petri net color extension, i.e., tokens may contain structured values. The transitions may fire depending on the values of the token, which means that the next task can be selected depending on case data. Furthermore the time extension to Petri nets can be used to model performance. To simplify modeling, the hierarchical extension is also included, allowing the design of sub-workflows. Elaborate split and join constructs are available in Workflow nets to allow modeling complex business processes.

The transitions have been enhanced with a graphical notation. The extensions are shown in Figure 1.2 (a). Transitions can be enhanced further by triggers, as shown in Figure 1.2 (c). The triggers represent external events that affect workflow executions. Hence, a trigger specifies an event that has to happen before a task can be executed. This could be a resource trigger like

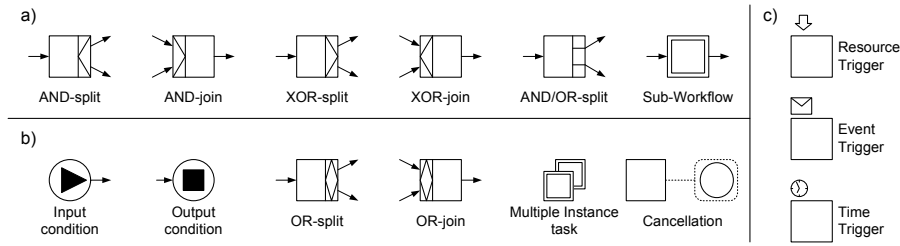


Fig. 1.2. Elements of workflow nets: (a) Basic workflow nets, (b) YAWL, (c) Triggers.

an employee selecting a task, an external event like the arrival of a message or a time-out trigger.

Workflow nets have to adhere to certain syntactic rules: A workflow net has always one input place and one output place. The input place is the only place in the Workflow net without incoming edges and the output place is the only place without any outgoing edge. There are additional conditions on the structure of workflow nets that have proven useful: (1) Every token that is placed in the input place should appear in the output place after finishing the workflow. (2) When this happens, no token should be left in the net. (3) The input place of every transition in the net should be reachable by firing sequences starting from the input place. A net that complies with these rules is called *sound*; it has no potential deadlocks, livelocks, dead tasks, and is guaranteed to terminate. Those criteria can be proved formally for a given Workflow net.

Inspired by the research on workflow patterns, a new language based on Workflow nets has been designed. Workflow patterns [AHKB00] thereby describe small pieces of ever recurring behavior in the workflow modeling domain. The general idea is based on design patterns in software engineering introduced by Gamma et al [GHJV94]. Besides providing a repository of modeling constructs, workflow patterns are useful to classify existing workflow management systems into their suitability and expressive power regarding the different patterns. Based on this research, a language extending Workflow nets that supports patterns more directly was developed, named YAWL: Yet Another Workflow Language. The most important language constructs are shown in Figure 1.2 (b). The new language includes *OR-splits* and *OR-joins*, which allow the easy modeling of the multiple choice and the m-out-of-n join patterns. The *Multiple Instance task* allows multiple instances of a task to run concurrently in different variations based on the multiple instances patterns. The *Cancellation* task deactivates all elements within the dotted region by clearing the contained tokens and is based on the cancel case pattern. The input and output places of the workflow are now marked with specific symbols.

The sample workflow can be specified graphically using a workflow net as shown in Figure 1.3. Each task is represented by an individual transition. The

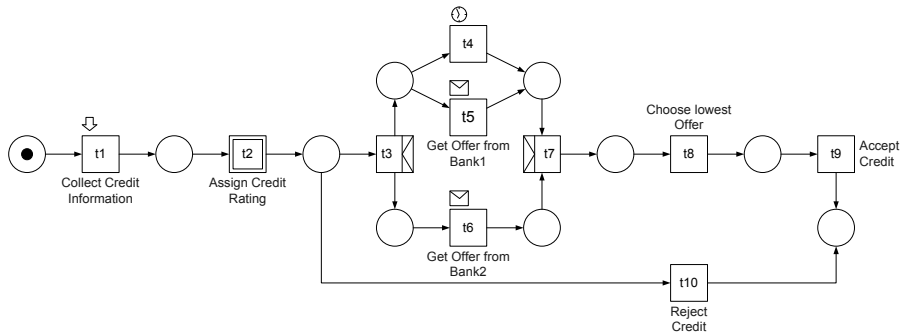


Fig. 1.3. Workflow nets: Sample workflow model.

workflow starts with a new case (token) in the leftmost place. To continue the processing, an external trigger is required for the task *Collect Credit Information*. This can be the customer or a clerk entering the data. After completing this activity, the next task assigns a credit rating to the case, based on the credit information given. In this composite task, the case is enhanced with a credit rating or a flag that the customer doesn't even get a rating. However, the specific tasks are not modeled in this workflow. Based on that value of the case, a data-based choice is made between the continuation of the workflow or the *Reject Credit* task. If the workflow continues, the credit rating is transmitted to both financial institutes (t3). Afterward there is an AND-split. Each of the tasks *Get Offer from Bank 1* and *Get Offer From Bank 2* waits for their respective response. This procedure is modeled by an event trigger. As the chosen financial institution does not always answer in time, we modeled a deferred choice between a timeout (t4) and the receipt of the message (t5). The choice which task is executed is thereby completely depending on the environment. The other possibility is then discarded. After both tasks have been finished, the best credit offer is selected, and *Accept Credit* is performed. Finally, the case arrives in the output place, and the workflow is finished. The given example only included Workflow net elements, however more complex workflows could be modeled much easier by using YAWL elements.

1.3.2 Business Process Modeling Notation

The *Business Process Modeling Notation* (BPMN) is designed as a standard notation for business process modeling that facilitates communication on business processes involving heterogeneous stakeholders, including domain experts, business analysts, technical developers, and business managers [BPM04]. In BPMN, business processes can be modeled at different levels of abstraction, supporting internal (so called private), external (public) as well as abstract (global) views. Furthermore, BPMN is well suited for modeling Web services based business processes and is closely related to service

composition languages like BPEL. The graphical notation supports sequence flow for coordinating activities inside a company and message flow for communication with other participants.

The Business Process Modeling Notation expresses business processes in so called business process diagrams. A business process diagram is based on flowchart techniques and consists of three basic categories that include elements for grouping objects, connecting objects as well as representing the flow of objects. The grouping objects include pools and swimlanes. A pool is represented by a rectangle drawn around a process and resembles a process participant, for instance a company. Swimlanes can be nested, i.e., the departments of the company can be represented by swimlanes inside the pool. In case of a single participant, the pool is usually omitted.

The flow objects consist of a set of three core elements, namely event, activity, and gateway. All flow objects must be placed inside a pool. The event is represented by a circle and is something that "happens" during the execution of a business process. Events can start, temporary suspend or end the process flow. Activities are represented by rounded rectangles. An activity is work a unit of a company performs, which can be either atomic or compound, resembling the functional workflow perspective. A gateway is represented by a diamond shape and controls the order of execution. Every core object can be further refined graphically and enhanced semantically by invisible attributes.

Concepts for connecting objects include sequence flow, message flow and associations. The sequence flow is represented by a solid line with a solid arrowhead that connects activities, events and gateways to define the behavior of the process. Sequence flow cannot cross the border of a pool. To communicate with another participant, message flow is used. Message flow is represented by a dashed line with an open arrowhead and can be seen as a vertical communication line. It connects activities and events between different participants, modeled by different pools. Associations are represented by dotted lines and used to enhance the diagram with data, text or other artifacts. A common artifact is a data object, represented by a note. This can be used to visualize the input and output attributes of an activity.

BPMN is designed for direct support of the workflow patterns. Advanced features include the support for exceptions, transactions and compensation handling as well as communication. The BPMN standard specification defines as direct mapping between BPMN and BPEL as an executable language.

The sample workflow can be specified using the Business Process Modeling Notation as shown in Figure 1.4. As BPMN supports data flow only by attributes, the case is not explicitly represented graphically. Instead, associations can be used to model the data flow informally, shown by the attached notes. The first task is *Collect Credit Information*, which somehow gathers the required credit information. Afterwards, the sub-process *Assign Credit Rating* is invoked, which assigns a credit rating based on the credit information. As BPMN directly supports exception handling, the sub-process raises an exception if it cannot assign a credit rating in a specific case. This leads to the task

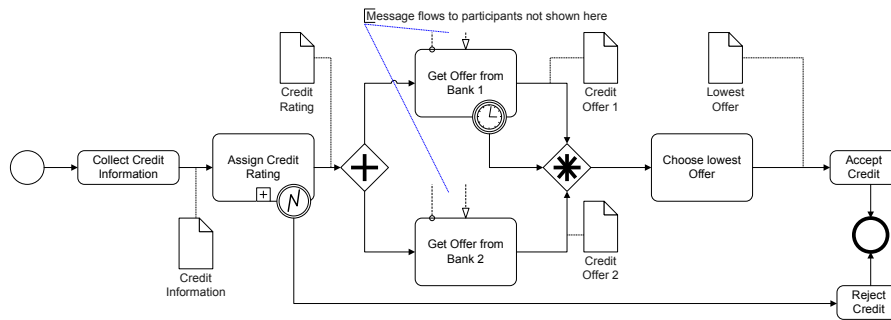


Fig. 1.4. BPMN: Sample workflow model.

Reject Credit and finally finishes the process. If no exception is risen, the sequence flow is split in two parallel flows by an AND-split, activating the tasks *Get Offer from Bank 1* and *Get Offer from Bank 2*. Each of the tasks sends a message to the corresponding financial institute (in BPMN textually modeled using attributes) and waits synchronously for their responses. The activity regarding the free chosen institute (Bank 1) has an intermediate timer event attached that is used to model the timeout. If the timeout occurs inside the activity, the sequence flow is continued from the timer event. The sequence flows are then joined using a complex gateway. This combines an OR-gateway for the two possible sequence flows from *Get Offer from Bank 1* followed by an AND-gateway combining the result of the or-gateway as well as the sequence flow from the activity *Get Offer from Bank 2*. Subsequently the *Choose lowest Offer* task is executed, and the sequence flow continues at the *Accept Credit* task, finishing the process.

1.3.3 Business Process Execution Language for Web Services

The Business Process Execution Language for Web Services deals with the orchestration of Web services for specifying business processes [BEA03], where orchestration means that services are sensefully composed to be executed in the context of a business process. Each Web service represents one or more tasks of the business process. BPEL is an XML-based language for describing Web service based business processes. It relies on W3C Web services standards, including the Web Services Description Language (WSDL) [CCMS01] and XPath [CD99] for selecting elements. BPEL is a convergence of prior specifications, namely XLang [Mic01] and the Web Services Flow Language (WSFL) [IBM01].

BPEL can be regarded a hybrid approach, as it combines block-structured and graph-based modeling in a single language. The different approaches are inherited from the block-based XLang and the graph-based WSFL. The notation can be used to model abstract and executable business processes. Abstract business processes handle only protocol-relevant data; they give a public view

of the process. Executable processes include the complete process specification; this can be seen as the private view.

In BPEL, Web services are invoked using so called partner links. The types of the partner links and their operations are defined in an external WSDL document. The root of a BPEL process definition is the `<process>` element which defines the required namespaces for the XML-structure and contains all other elements. A `<partnerLinks>` element is used to define the communication interfaces as specified in the WSDL document, followed by XML-variable definitions by the `<variables>` element. The main process consists of elements for sequential control flow (`<sequence>`), parallel control flow (`<flow>`), and routing constructs like `<switch>`, `<pick>` or `<while>`. Activities (i.e., Web services) are called asynchronous or synchronous using the `<invoke>` element, answers are sent using `<reply>` and answers are asynchronously received using the `<receive>` element. Variable assignment is achieved through the `<assign>` element.

The elements of the BPEL main process can be grouped by the `<scope>` element which is similar to a block definition in programming languages, like Java or C. The elements discussed so far are all block-structured. The graph-based approach is made possible inside (parallel) flow blocks by defining links. Any activity can be the source or the target of a link and a target can only be activated if the source has completed. Advanced BPEL features include fault and compensation handling using `<faultHandlers>` and `<compensationHandlers>` with associated `<catch>` and `<compensate>` elements. Those handlers have to be attached to an already defined scope.

The complete definition of the sample workflow in BPEL consists of a WSDL file containing the partner link definitions as well as a BPEL file containing the BPEL code. For ease of presentation, we omit the WSDL file, the partner link definitions, the variable definitions, as well as attributes for the elements of the example. As there is a direct mapping between BPMN and BPEL, we sometimes refer to the BPMN example given in Figure 1.4.

The BPEL example in Figure 1.5 covers the first tasks of the sample workflow. These tasks are placed inside a `<sequence>` element (1). The first task is *Collect Credit Information*, which is modeled as a synchronous invocation of a Web service (2). The result is an output variable containing the requested *Credit Information*. The following *Assign Credit Rating* sub-workflow is modeled as a scope (3). Every scope can have a fault handler, which handles the exceptions (4). We define one exception called *assignError* inside the fault handler (5). After the definition of the fault handler, several activities of the sub-process can follow (6). Those can include assignments for variables, invocations of Web services, branching structures, and so on. Also a `<throw>` element for generating the exception caught in the fault handler can be used. The process is continued by the parallel flow of the two *Get Offer from Bank* activities. The parallel execution is modeled in BPEL using the `<flow>` element (7). Inside the flow element, parallel activities could be defined (8, 10). Each of them is executed independently and the control flow is joined after

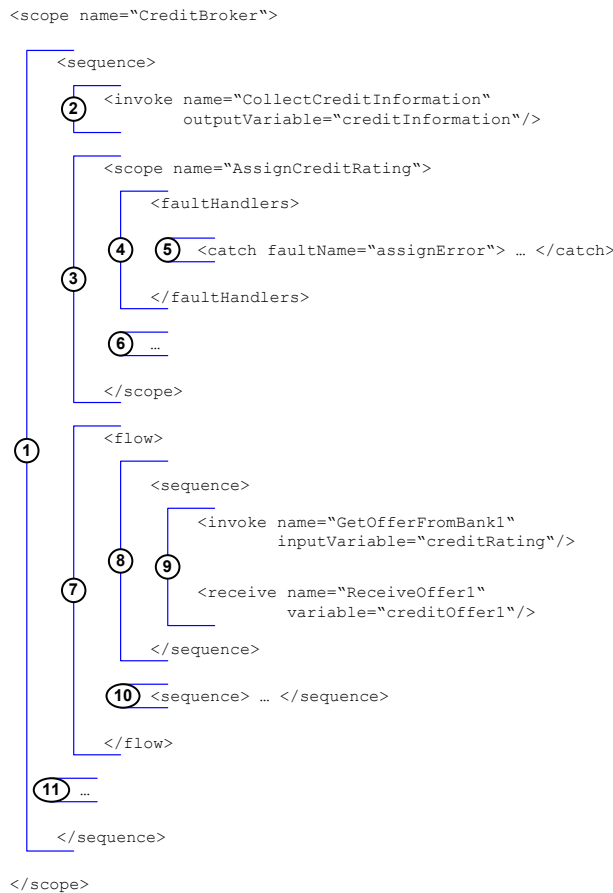


Fig. 1.5. BPEL: Sample workflow model.

finishing all elements contained inside the flow element. In (9) we have modeled an asynchronous call to the financial institute that is placed using the `<invoke>` element with only input variables defined. The result is received asynchronously through the `<receive>` element. In (10), the other financial institute is called in parallel, the code inside is the same as in (9), except for the names and variables. After joining the control flow, the sample process continues in (11). The example given considered only a small part of the BPEL definition, as we omitted the timeout for example, but should give a short overview of the general layout of a BPEL process.

1.4 Conclusions

Process technology in general and workflow management in particular have been active research areas in recent years. Based on this general observation, there are two areas in workflow management with specifically strong developments, i.e., formal investigation of workflow properties and workflow technology in service-oriented environments.

The structure and content of this chapter resembles these new developments by introducing Workflow nets including extensions to support additional workflow patterns as well as service composition issues, both with respect to modeling aspects (BPMN) and XML-based service composition languages (BPEL). On the other hand, other approaches are either outdated or have made their way into current proposals. For instance, the Flow Definition Language is no longer used to specify workflows. Instead, its meta model based on directed graphs can be found in service composition languages. In particular it has made its way via the Web Services Flow Language into BPEL. Other formerly important proposals based on programming languages are no longer relevant either; these can also be regarded as superseded by service composition languages.

In the future we expect some strong momentum to the workflow community by semantic characterizations of processes that might lead to better process composability and that ultimately make interorganizational business processes a reality. A second area that might fuel developments in workflow management is Grid computing [FK04], where computation grids not only facilitate the scheduling and provision of computing power but also the management of processes that are being executed in the grid.

References

- [A+04] Alonso, G., Casati, F., Kuno, H., Machiraju, V.: *Web Services — Concepts, Architectures and Applications*. Springer 2004.
- [Aal03] van der Aalst, W.M.P. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.
- [AH02] van der Aalst, W., van Hee, K. *Workflow Management*. MIT Press, 2002.
- [AHH94] van der Aalst, W.M.P., van Hee, K.M., Houben, G.J. Modeling and analysing workflow using a Petri-net based approach. In De Michelis, G., Ellis, C., Memmi, G. eds.: *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31–50, 1994.
- [AHKB00] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P. Workflow patterns. Technical Report BETA Working Paper Series, WP 47, Eindhoven University of Technology, 2000.
- [AW01] van der Aalst, W.M.P., Weske, M. The P2P approach to Interorganizational Workflow. In Dittrich, K.R., Geppert, A., Norrie, M.C., eds.:

- Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), volume 2068 of LNCS*, pages 140–156, Berlin, 2001. Springer-Verlag.
- [AWW03] van der Aalst, W.M.P., Weske, M., Wirtz, G.: *Advanced Topics in Workflow Management: Issues, Requirements, and Solutions*. Journal of Integrated Design and Process Science. Volume 7, Number 3. Austin: Society for Design and Process Science 2003
- [BEA03] BEA Systems, IBM, Microsoft, SAP, Siebel Systems. *Business Process Execution Language for Web Services, Version 1.1*, May 2003.
- [BPM04] BPML.org. *Business Process Modeling Notation*, 1.0 edition, May 2004.
- [CCMS01] Christensen, E., Curbera, F., Meredith, G., Sanjiva, W. *Web Service Description Language (WSDL) 1.1*. IBM, Microsoft, March 2001. W3C Note.
- [CD99] Clark, J., DeRose, S. *XML Path Language (XPath) Version 1.0*. W3C, November 1999. W3C Recommendation.
- [CD02] Casati, F., Dayal, U., eds.: *Special Issue on Web Services*. IEEE Bulletin of the Technical Committee on Data Engineering, 25 (4), December 2002.
- [FK04] Foster, I., Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure*, 2nd edition. Morgan-Kaufmann Publishers, San Francisco, CA, 2004.
- [GHS95] Georgakopoulos, D., Hornick, M., Sheth, A. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995.
- [GHJV94] Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns*. Addison-Wesley, Reading, 1994.
- [G+04] Grüne, M., Keferstein, K., Lenz, K., Oberweis, A., von Mevius, M., Vossen, G. *Individualization of E-Learning Processes by Workflow-Based Learning-Object Management*. Proc. 7th International Conference on Business Information Systems (BIS) 2004, Poznan, Poland, 214–226.
- [HLGA01] Hoffner, Y., Ludwig, H., Grefen, P., Aberer, K. Crossflow: integrating workflow management and electronic commerce. *SIGecom Exch.*, 2(1):1–10, 2001.
- [Hoa85] Hoare, C.A.R. *Communicating Sequential Processes*. Prentice-Hall, 1985.
- [H+03] Hull, R., Benedikt, M., Christophides, V., Su, J.: E-Services: A Look Behind the Curtain. *Proc. 22nd ACM Symposium on Principles of Database Systems (PODS) 2003*, San Diego, CA.
- [IBM01] IBM. *Web Services Flow Language (WSFL 1.0)*, May 2001.
- [Ioa93] Ioannidis, Y.. eds.: *Special Issue on Scientific Databases*. Data Engineering Bulletin 16 (1) 1993.
- [LNO89] Lausen, G., Nemeth, T., Oberweis, A., Schönthaler, F., Stucky, W.: The INCOME approach for conceptual modelling and prototyping of information systems. Proc. 1st Nordic Conference on Advanced Systems Engineering, Stockholm, Sweden, 1989.
- [LR00] Leymann, F., Roller, D. *Production Workflow — Concepts and Techniques*. Prentice Hall, 2000.
- [Mic01] Microsoft. *XLang Web Services for Business Process Design*, 2001.

- [Mil80] Milner, R. *A Calculus of Communicating Systems*. Springer LNCS 92, 1980.
- [New02] Newcomer, E.: *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley 2002.
- [RD98] Reichert, M., Dadam, P. *ADEPT_{flex} – Supporting Dynamic Changes of Workflows Without Loosing Control*. Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management, Vol. 10, No. 2, 1998.
- [RRD04] Rinderle, S., Reichert, M., Dadam, P.: *Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey*. In: Weske, M., van der Aalst, W.M.P., Verbeek, H.M.W., eds.: Data and Knowledge Engineering, Special Issue on Business Process Management. Elsevier (to appear)
- [VB96] Vossen, G., J. Becker. (eds.) *Business Process Modeling and Workflow Management: Models, Methods, Tools* (in German). International Thomson Publishing, Bonn, Germany, 1996.
- [VJO02] Vossen, G., Jaeschke, P., Oberweis, A. *Flexible Workflow Management as a Central E-Learning Support Paradigm*. Proc. 1st European Conference on E-Learning (ECEL) 2002, Uxbridge, UK, 253–267.
- [VW98] Vossen, G., Weske, M. *The WASA Approach to Workflow Management for Scientific Applications*. in: Dogac, A., Kalinichenko, L., Özsu, T., Sheth, A., eds.: *Workflow Management Systems and Interoperability*, NATO ASI Series F: Computer and System Sciences, Vol. 164, Springer-Verlag, Berlin, 1998, 145–164.
- [VW99] Vossen, G., Weske, M.: *The WASA2 Object-Oriented Workflow Management System*. Proc. ACM SIGMOD International Conference on Management of Data 1999, Philadelphia, PA, 587–589.