SAP WebFlow Made Configurable: Unifying Workflow Templates into a Configurable Model

F. Gottschalk, W.M.P. van der Aalst and M.H. Jansen-Vullers

Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
{f.gottschalk,w.m.p.v.d.aalst,m.h.jansen-vullers}@tue.nl

Abstract. To facilitate the implementation of workflows, enterprise and workflow system vendors typically provide sets of workflow templates for their software. Each of these templates depicts a variant of how a particular business process can be supported by the software. The user of such a system can save the effort of creating the models and the links to system components from scratch by selecting and activating the best fitting template. A combination of the strengths from different templates is however only achievable by manually adapting one of the templates which is cumbersome. We therefore suggest in this paper to combine the different workflow templates into a single configurable workflow template. Using the workflow modeling language of SAP's WebFlow engine, we depict how such a configurable workflow modeling language can be created by identifying the configurable elements in the original language. Requirements imposed on configurations inhibit invalid configurations. Based on a default configuration such configurable templates can be used as easy as the traditional templates. The suggested general approach is also applicable to other workflow modeling languages.

Keywords: Process Configuration, Business Process Reference Model, Process-aware Information System, Workflow Template, SAP WebFlow

1 Introduction

Workflow engines facilitate the execution of business processes by guiding and monitoring cases of the business process while "running through the company". That means, a workflow engine executes processes as automatically as possible. Whenever needed it provides all required information for the execution of tasks to the responsible individuals, notifies them when new work items arrive, and takes action in case tasks are not performed in time [24].

To execute a workflow in a workflow engine, it must first be specified in the engine's workflow modeling language. The workflow modeling language can be seen as an extended business process modeling language (like Event-driven Process Chains (EPCs) [21], BPMN [34], UML-activity diagrams [14] etc.). Besides depicting the process, it allows for the integration of activities specified in the process model with other systems like enterprise systems, groupware, office software, or intranet portals.

The effort to establish this integration is typically high. When modeling a workflow, it is not only required to ensure the correct control flow, but also the data flow between the different steps and components must be "programmed" and assignment rules for resources must be set up. Thus, the re-use of workflow models promises huge costs savings when implementing workflows in similar system environments. This holds especially for enterprise systems, which due to the system structure already imply the set-up of processes in quite specific ways.

The biggest enterprise system vendor worldwide is with more than 100,000 installations SAP [29]. The workflow engine that is delivered by SAP together with every installation of its enterprise system since the R/3 Release 3.0 is called WebFlow[1]. Together with the engine, SAP also delivers hundreds of simple, predefined workflow templates for all areas of the system – from logistics and material management to personal time management, sales and distribution, or compensation management [24]. Thus, the template repository can be regarded as a reference model of common workflows in SAP's enterprise system. The templates, which typically fit comfortably on one A4 page, can easily be activated in the SAP system. Without a workflow designer having ever spent a significant amount of time on the workflow definition, they are then triggered automatically whenever their execution is required. Often SAP users are therefore working on the predefined workflows without even knowing it.

For many business processes the repository includes several workflow templates, each suggesting a different implementation of the particular process. For example, there is a dedicated workflow template not only for the approval of a travel request, but also for the automatic approval of a travel request. In addition, there are workflow templates for the approval of a travel plan, the approval of a trip, and the automatic approval of a trip.

All these templates are in their structure of course similar. To decide on the appropriate template, each template is documented in SAP's online help system, typically also combined with an EPC of the process. As an example figures 1 and 2 show two EPCs from the online documentation of SAP [28]. These two EPCs refer to the templates for supporting the approval of a travel request and the automatic approval of a travel request. However, there is no information available that highlights the differences between the two templates. Instead, the workflow designer has to familiarize herself with the details of each workflow template, compare them manually on her own, and find the small differences. If, as in the example from figures 1 and 2, there is a certain degree of inconsistency in the documentation of the templates because it is unclear if "Create travel request" and "Enter travel request" actually depict the same task, this comparison requires even further efforts. In the worst case, the workflow designer might even come up with the conclusion that a combination of two

[1] In early releases the WebFlow engine was known as SAP Business Workflow. The ideas presented in this paper are basically also applicable to these versions of the engine.

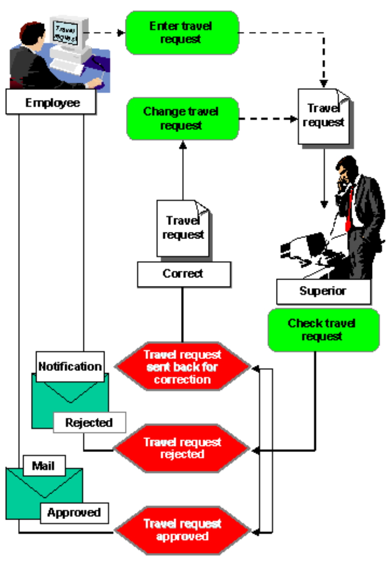

Fig. 1. The documentation of the SAP WebFlow template for approvals of travel requests [28]

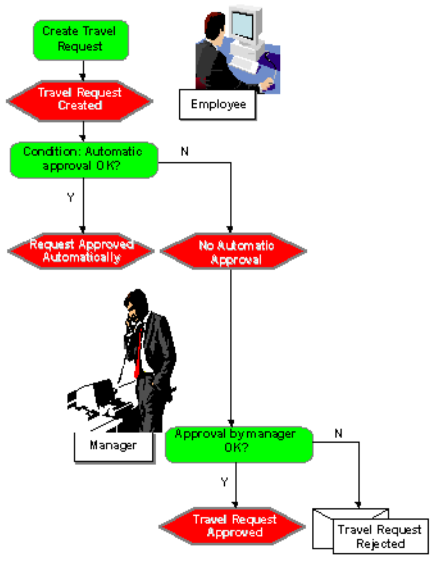

Fig. 2. ... and the documentation of the template for the automatic approval of travel requests [28].

templates would be the optimal solution as each template has its strength at a different point in the model. As such a template is not available she can then only manually adapt one of the templates at its weak point to match the not selected one here as close as possible.

To help the workflow designers in getting the optimal workflow template we propose in this paper to combine the different workflow template variants into a single template using an extension to the workflow specification language making it a configurable workflow specification language. This configuration extension allows the workflow designer to just select or eliminate the relevant or irrelevant template parts. Thus, the designer can focus on the requirements on the workflow instead of searching for the possibilities in the different templates.

For this we will in the following first depict, how configuration can be performed on a workflow model and how a workflow modeling language can be extended with such configuration options. Afterwards we will apply these concepts to SAP's workflow modeling language and depict the integrated workflow template for the example shown above. The paper concludes with a short summary and an outlook on open issues.

2 Making workflow models configurable

As depicted in the introduction, configurable workflow modeling languages are useful whenever an individual workflow variant should be derived from a more general model. For this, configurable workflow modeling languages enable the restriction of the behavior of workflow models in a controlled manner. This section outlines a general approach for the development of such configurable workflow modeling languages.

By using the term "workflow models", we explicitly focus on executable business process models, although the approach might be applicable to non-executable, conceptual modeling languages as well. We assume that every workflow modeling language that explicitly depicts the flow of cases, i.e. executed workflow instances, through a system with steps, tasks, activities, functions or similar concepts of performed actions can be made configurable. Before defining such a configurable workflow modeling language, it is however first required to identify what a configuration of a model in a particular language is.

In our previous research on configurable process models [2,17], we identified two general applicable methodologies to configure, i.e. restrict, a workflow model, namely *blocking* and *hiding*. The insight that these are the two basic configuration operations was obtained by a systematic study of inheritance notions in the context of business processes [1,5]. If an action in a workflow is *blocked*, it cannot be executed. The process will never continue after the action and thus never reach a subsequent state or any subsequent action. If an action is *hidden*, its performance is not observable, i.e. it is skipped and consumes neither time nor resources. But the process flow continues afterwards and subsequent actions will be performed. For that reason we also talk about a *silent action*[2] or simply about *skipping the action*. If an action in a workflow model is neither blocked nor hidden, then we say it is *enabled*, which refers to its normal execution. The enabled action is performed as it would be in a classic, un-configurable workflow.

As a rule of thumb this means that if the performance of an action is not desired and the action is not mandatory for subsequent steps, than hiding is the preferred configuration method. If a whole sequence or line of actions is not desired or if the non-desired action is mandatory for subsequent actions, blocking is typically the preferred configuration method. But note that there might be a lot of exceptions to this rule.

To develop configurations for a particular language, it is required to identify those element types of the language which represent actions that can be performed. These are usually elements like activities, functions, steps, etc. For an action to be executed, it must be triggered. Triggers are typically represented by arcs pointing into an action. However, the meaning of these arcs leading into the action vary not only among different workflow modeling languages but also within a single workflow modeling language because of different joining patterns for preceding paths leading into the action. For example, some actions require that all preceding paths are completed for the action to be triggered (AND-join), whereas other actions can be triggered via each arc pointing into the action (XOR-join). We call each combination of incoming paths through which an action can be triggered an *inflow port* of the action (see the left side of Figure 3). Thus, an action with an AND-joining behavior for the incoming paths has

[2] The term silent action comes from concurrency theory where silent actions are denoted as τ and form the basis for equivalence notions such as branching bisimulation.

just a single inflow port whereas a task with an XOR-joining behavior has an inflow port for each incoming path.

After an action has completed, it releases the particular case via the arcs leaving the action. Also here the number of triggered paths depends on the semantics specified for the particular action. An action with an AND-splitting behavior triggers all outgoing paths, whereas an action with an XOR-splitting behavior only triggers one subsequent path. Of course there can also be semantics allowing the triggering of a specific number of paths (OR-split). Aligned with the specification of inflow ports, we say that each case can leave the action only through one distinct *outflow port*, but then triggers all paths connected to this outflow port (see the right side of Figure 3).

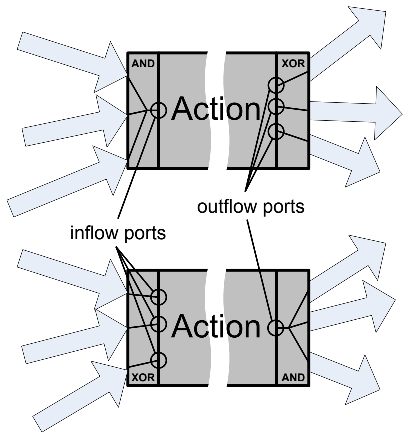

Fig. 3. The number of ports of an action depends on its joining and splitting behavior

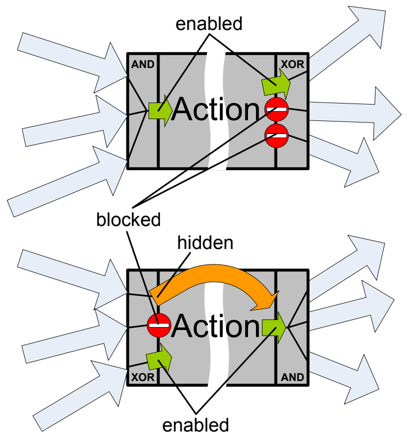

Fig. 4. Ports can be enabled or blocked, and in case of an inflow also be hidden

The different ports of an action thus represent runtime alternatives in a process modelling language. By making ports the configurable elements of the net, i.e. by making them the elements which can be enabled, blocked, or hidden, we provide the opportunity to select desired variants among the alternatives and eliminate undesired variants on a distinct configuration level. Every port can be enabled or blocked while inflow ports can also be hidden.

An enabled inflow port allows the triggering of the action through this port. If an inflow port is however blocked, no cases can flow into the action through this port. The triggering of the action via the inflow port is inhibited. If an action is triggered via a hidden inflow port, the action itself is skipped and the case is directly forwarded to one of the outflow ports (usually but not necessarily a default output port; see Figure 4).

If an outflow port is enabled, the action can select this port as the port through which the case is released. If an outflow port is however blocked, the port cannot be selected as the used outflow port. Instead another enabled outflow port must be selected. Thus, the blocking of an outflow port inhibits the performance of actions subsequent to the port. However, as cases should always be able to leave a triggered action, at least one outflow port must always be enabled. The hiding of an outflow port is impossible because outflow ports trigger paths instead of actions. A path just forwards the case to the next action without containing any action itself. Thus, a path contains nothing that can be skipped (and any subsequent action should be hidden via its own input ports).

By deriving ports from the definition of a workflow modeling language instead of defining them as elements which have to be added to the workflow models of the language, each model can serve as the basis for a configurable model without any change. Such a model represents the "Least Common Multiple" of all possible model variants. It contains the maximal possible behavior which can be achieved by enabling all variants, i.e. configuring all ports as enabled. We call this initial model therefore the *basic model* whose behavior can be restricted by hiding or blocking of selected ports.

To transform such configuration decisions into a model executable in the traditional workflow engine, blocked elements and all their dead successors must be removed from the model and hidden elements must be replaced by shortcuts.

Obviously, not all models created by such a transformation conform to the definition of the used modeling language or to behavior executable by the connected systems. For example, blocking of too many ports or of a "wrong" port might result in a deadlock during the execution of the process. In a similar way also hiding essential actions might result in semantically incorrect and thus non-executable models. To avoid the occurrence of such situations, a configurable model must not only consist of the basic model, but also of a set of requirements restricting the set of permitted configurations and therefore ensuring both syntactical and semantic validity of models. An example for a syntactical motivated requirement would be "Each action must have at least one enabled outflow port to allow the outflow of cases"; an example for a semantically motivated requirement would be "If a manager can decide on the approval of a travel request, she must at least be able to accept or refuse it." (but might also be able to, e.g., delegate the decision), or better "If a port is enabled that allows cases to flow into the action *Check travel request*, then the ports allowing for cases flowing into the actions *Travel request approved* and *Travel request rejected* must be enabled". That means, although the requirement is semantically motivated, it still should be formulated in terms of the model's port configuration.

For a better understanding, we presented the example requirements in a rather informal way. However, to detect and prevent invalid configurations automatically, a formal specification of requirements is indispensable. We therefore suggest either the use of a subset of a programming language, or to formulate logical expressions (similar to, e.g., the validity constraints suggested in [23], or the formulas suggested in [10]).

As mentioned above, the basic model uses the traditional, i.e. non-configurable, version of the particular modeling language. Assuming that all ports are enabled, the model should therefore satisfy all of the language's syntactical requirements.

It might however contain semantically conflicting elements whenever two variants of the workflow exclude each other. To be able to use such a basic model in a workflow engine, it is thus required to explicitly set up a configuration that satisfies all requirements. By requiring that every valid configurable workflow model contains at least one valid configuration as a default configuration, we ensure the existence of such a configuration.

The default configuration also serves as a “starting point” for any individual configuration. In this way, configuring a configurable workflow model to individual requirements just means to modify those port configurations that need to deviate from the default configuration – usually a limited effort even if there are many configurable ports.

3 Configurable workflow models in SAP

Workflows that should be executed in SAP’s WebFlow engine must first be specified in the engine’s workflow modeling language which for simplification we just call SAP workflow in the following. SAP workflow is mainly based on so-called steps and events which are organized in a block structure[3]. Steps are depicted by boxes with different symbols, representing either routing constructs or functionalities offered by the system. Figure 5 depicts the SAP workflow template for the travel approval process from Figure 1. We will use this example to explain the different modeling elements.

– The most basic step type is the *activity* (▶) which is in the example used for the *Set trip status to approved*, *Change trip*, *Enter and send message*, and *Send mail: Request approved* activities. An activity step is always connected to a task maintained in the SAP system which is executed when the step is triggered. After completion of the task, the step is completed and the next task is triggered.
– *User decisions* (?▶) such as the *Approve travel request* step shown in Figure 5 are providing the user with a list of answers from which she can chose one. Based on this answer the corresponding subsequent path is selected.
– *Conditions* (◆) such as the *Travel request approved?* step evaluate a Boolean condition. Based on the outcome of this evaluation they follow one of the two paths. Similar, *multiple conditions* (◆, not depicted in the example) contain a set of subsequent paths, of which one is selected based, e.g., on the value of a (non-Boolean) data element of the workflow. If no path is connected to the value of the data element, an “other values” branch is selected.
– *Forks* (⇶, ⇶) allow parallel processing of paths. All paths leaving the splitting fork are triggered by this step. The joining fork allows the specification of a condition when it is completed. This condition can be the number of preceding paths that have to have completed. The condition can also be based on data elements of the workflow.

[3] SAP also provides an EPC translation for workflows specified in SAP workflow. The usage of EPCs is however restricted to constructs realizable in SAP workflow. For this reason we stick to the original SAP workflow notation.

– As soon as a *Wait for event* step ( ) is triggered, it waits for a linked event to occur and the workflow is only continued after the event has occurred.

Events ( ) can be raised by business applications to communicate with workflows. The trigger to raise an event can be manifold. For example, it can be the creation or the change of a document, a general status message, an exception occurring in an information system, or a business transaction event which occurred in the financial system. Even a workflow can raise events on its own by

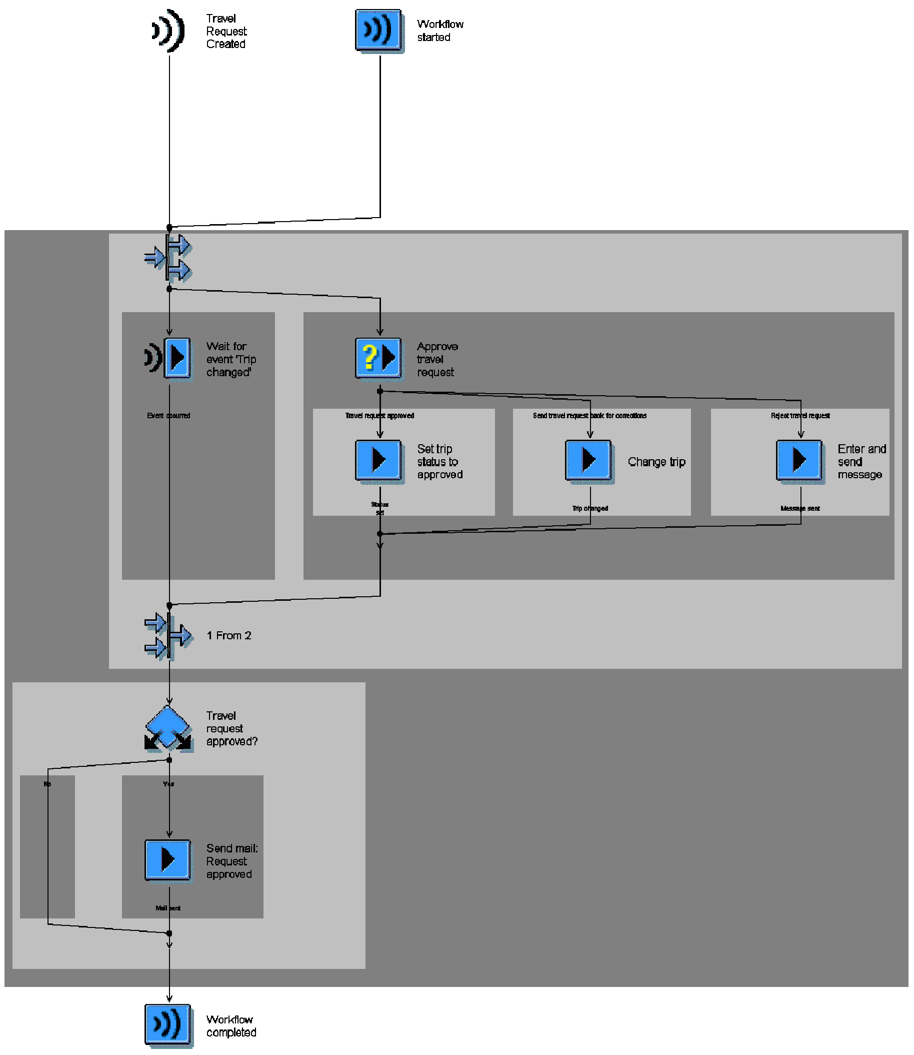

Fig. 5. The workflow template of the travel approval process depicted in Figure 1 in the SAP workflow notation (accessible in SAP as workflow WS20000050)

Event creator steps (▶). Events can be linked to workflows (and tasks which then are handled as if they are workflows on their own) as triggering events to start the workflow (e.g., as the *Travel Request Created* event in the example), or as terminating events to stop a workflow or a *wait for event* step inside a workflow. Note, that the same event can be linked for different purposes to different workflows at the same time, e.g. to terminate one workflow and instead trigger another workflow.

The linkage between steps or events and workflows includes also the linkage of the data in the data containers of the step or event and the workflow. This linkage enables the start of a workflow or a step with the right parameters, e.g. to select responsible resources or correct documents. We will skip further implementation details about this here, but not without repeating that this modeling and customizing effort is far more time-consuming than the pure creation of a process model. These efforts therefore motivate the development of a configurable SAP workflow.

What are the actions and their ports in SAP workflow?

SAP workflow models are block structured. In the simplest case a single step as, e.g., an activity represents a block. However, whenever a step causes the branching of the control-flow (as a fork, a condition, a user-decision, or any other step that contains different outcomes) the branching of the control flow is matched by exactly one corresponding join and all elements until (and including) the matching join belong to the block of the branching step. The elements in each of the branches represent then sub-blocks of the branching block. This is also highlighted in Figure 5. The block of the fork is highlighted in light grey. This block contains two sub-blocks, one for each of the two branches. The block of the user-decision step *Approve travel request* branches again in three branches, each containing a block for the particular activity.

Thus, each block can be seen as an action. Bigger blocks are actions on a higher level of abstraction; smaller blocks are typically actions on a lower level of abstraction. Each block contains basically just one unique input path and one unique output path which are the ports of the action.

The biggest block is the block of the complete workflow itself. It is the only block which can be triggered in multiple ways as it cannot only be triggered by a manual start of the workflow but also by (various) events which are linked to the workflow block. In addition, events can also be linked to a workflow block to terminate it or to re-start it. Thus, each of these links connecting events to the workflow block can also be seen as a port. As they have some different characteristics from the input and output ports of a block, let us call them *event ports*. In the same way, we also call the linkage between events and *wait for event* steps and between *event creator* steps and events *event ports*.

How to configure the ports?

As explained in the previous section, inflow ports of actions can be enabled, hidden, or blocked. This concept can be applied to the input ports of blocks in SAP workflow in a straightforward manner. If the input port of a block is enabled, cases can normally enter and be executed in the block. If the input port is hidden, a case entering the block is directly forwarded to the unique exit port of the block, quasi bypassing all the content of the block. If the input port is blocked, the case cannot enter the block at all and needs to continue via other alternative branches.

Common soundness criteria for workflow models require that cases must always have a chance to complete a workflow. That means they should never get stuck within a workflow. Thus, a block can only be blocked if an alternatively executable branch exists. This is no problem in case of the *Change trip* step's block because alternatively the *Set trip status to approved* or the *Enter and send message* steps' blocks can be executed. It is however impossible to block the input port of the *Travel request approved?* step's block as the workflow does not contain any alternative to it. In the case of this particular fork step, it is possible to block one of the two sub-blocks, but only because the join requires just one of the two branches to complete. If the condition at the join would have been "2 From 2" a blocking of one of the sub-blocks would have made it impossible to later satisfy this condition and thus caused a deadlock. Therefore, when configuring the sub-blocks of a fork, the condition at the joining fork determines the maximal amount of sub-blocks that can be blocked.

As the output port of a block is unique, each case entering a block must be able to leave the block via this output port. Thus, a blocking of this port is not practicable as long as cases are able to enter the block. Only if the input port is blocked, the output port can be blocked as well. Such a blocking is however just of a theoretic nature because if no tokens can arrive at an output port, its configuration has no influence on the process execution anyway. In addition, we already showed in Section 2 that hiding of an output port is not really feasible either because the path to the next block does not contain any action. If any subsequent block should be hidden, this should rather be done on the subsequent block's input port[4]. We can therefore assume that all output ports in an SAP workflow model are always enabled and consider the configuration of output ports in SAP workflow as practically irrelevant.

In SAP workflow an event only triggers a workflow if the link between the event and the workflow is activated. The SAP workflow system already supports the deactivation of such a link, quasi corresponding to the blocking of the particular event port. Although a triggering event port is an inflow port, hiding of such a port is quite useless because it would basically mean skipping the whole workflow block without performing any step. Terminating event ports for wait-for-event steps and the event ports of event creator steps are outflow

[4] If it is desired to hide a series of blocks, SAP workflow even provides a block step into which a sequence of blocks can be encapsulated such that only one input port needs to be hidden.

ports. Even though terminating events are externally triggered, they basically enforce the removal of the case from the particular block. Thus, the functionality of SAP to activate or deactivate such linkages already provides exactly the required functionality to enable or block event ports.

In Figure 6 we combined the SAP workflow template from Figure 5 with the template for the automatic approval described by the EPC of Figure 2. By blocking the *change trip* step the corresponding block is quasi removed from the workflow. By hiding of the *Travel request approved?* step, also the sub-block of mailing the request's approval is skipped. All other blocks are enabled. The result of this configuration then corresponds exactly to the SAP workflow template WS12500021 for the automatic approval of a travel request whose EPC documentation was depicted in Figure 2. By blocking the sub-block of the *Criteria for Automatic Approval* step's *Automatically Approve Travel Request* outcome and instead enabling the *Change trip* and the *Travel request approved?* blocks, we would end up with the traditional approval workflow from Figure 5. Of course

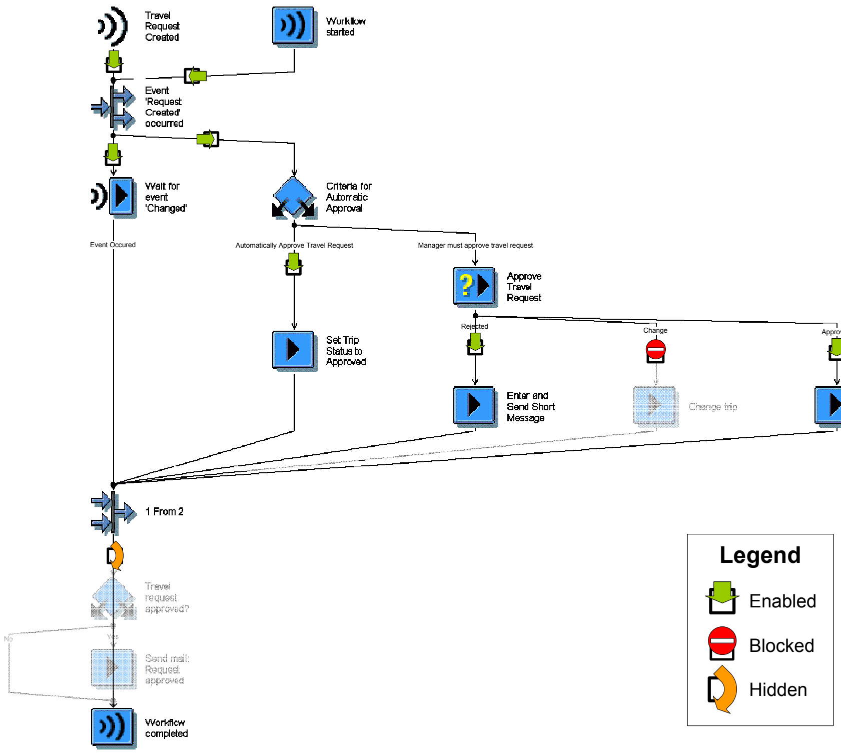
Travel Request Created
Workflow started
Event 'Request Created' occurred
Wait for event 'Changed'
Criteria for Automatic Approval
Event Occured
Automatically Approve Travel Request
Manager must approve travel request
Approve Travel Request
Rejected
Change
Approved
Set Trip Status to Approved
Enter and Send Short Message
Change trip
Set Trip Status to Approved
1 From 2
Travel request approved?
No
Yes
Send mail: Request approved
Workflow completed
Legend
Enabled
Blocked
Hidden

Fig. 6. The combined workflow template of SAP's travel approval workflow template (WS20000050) and the automatic approval template (WS12500021), configured as the automatic approval workflow.

this workflow would still contain the *Criteria for Automatic Approval* step, but its only possible outcome is the manual approval sub-block.

Restricting the configuration opportunities

As indicated before, not all such combinations are feasible in practice. We already mentioned the requirement that a workflow always has to have the opportunity to complete. In addition, there are typically a lot of semantic requirements. For example, it is well possible to block or enable the *Wait for event 'Changed'* step's block. However, hiding it would prevent the workflow from working correctly as it would cause a direct forwarding of cases to the joining fork whose condition would immediately be satisfied. Thus, the other branch would get superfluous and cancelled before any decision on the approval can be made. As another example, blocks such as the accept and reject sub-blocks of the *Approve travel request* step might be mandatory for the particular workflow. Then these two blocks must always be enabled.

Applying the idea of using logical expression to denote these requirements, we could for example write `configuration("Enter and send short message") =ENABLED` to depict that the particular block must be enabled or `configuration("Wait for event 'changed'")!=HIDDEN` for the requirement that the block cannot be hidden (of course we would rather use unique block IDs than the step names as steps can occur more than once in a single workflow). Such atomic logical expressions can then be combined, e.g., to formulate a requirement that if the *Change trip* block is blocked then the *Travel request approved?* must be hidden (`configuration("Change trip")=BLOCKED => configuration("Travel request approved?")=HIDDEN`).

To test if a configuration fulfills all requirements, the requirements can be combined using `AND` operators. By determining the blocks which can change their configuration values without breaking the requirements, it is even possible to identify those blocks of the workflow which are not bound to their current value and thus really configurable. But note that this calculation is complex and with changing the configuration value of one element, the configurability of other elements can change. Still, a tool that is able to regularly re-evaluate the configuration opportunities for reasonable sized models might use this information to provide configuration opportunities only to currently configurable blocks.

Plug & Play

The current SAP workflow templates allow for an easy integration of the predefined workflow templates into a running SAP system by just assigning the relevant resources to the steps and activating the triggering events. To enable such an easy activation also for configurable workflow templates, each workflow template has to have a default configuration. A default configuration can be any configuration satisfying the requirements specified for the workflow. For example, the configuration of Figure 6 representing the automatic approval template could be the default configuration for the combined travel approval workflow template. When activating the triggering event, the workflow corresponding to this configuration would automatically be enabled. However, if it is for example desired, that the manager is also able to ask for a change of the travel request, it is sufficient to assign the responsible resource to the *Change* step and enable the currently blocked port. Without any modeling effort the new configuration of the workflow template can be used.

As configured templates can easily be transformed into executable workflow models according to today's SAP WebFlow notation, it is not necessary to change SAP's current workflow engine to run workflows derived from the suggested configurable language. Instead, the implementation of the suggested approach in the context of SAP's enterprise system would solely require an extension of the user interface that depicts the configuration options, a tool that checks the fulfillment of the requirements by the configuration to prevent invalid configurations, and an implementation of the transformation algorithm to derive the configured workflow model from the configuration.

4 Related work

The workflow templates of SAP's WebFlow engine depict suggestions how to execute the particular processes in SAP. Thus, they can be regarded as a reference model for processes executable in SAP. Motivated by the "Design by Reuse" paradigm, reference models simplify the process model design by providing repositories of generally valid and potentially relevant models which can be used to accelerate the modeling process [9,15,16,22,33].

One of the most cited reference models in the context of business process modeling is the *SAP reference model* [12,30], which was developed in the nineties in cooperation between SAP and the IDS Scheer AG and covers more than 1000 business processes as EPCs. However, different from the executable workflow templates of SAP workflow which we address in this paper, the SAP reference model was designed just on a conceptual level.

To be applicable in a particular context (e.g., a specific enterprise), a generally valid reference model must typically be adjusted to individual requirements (of the enterprise). This can be done by adding additional content to the model or by configuring the existing content of the model [6,7]. To enable the adaptation of reference models by means of configuration, several variants of the process must be integrated within the model. Extensions to conceptual process modeling languages that allow for defining such an integration are suggested by Becker et al. [6], Rosemann and van der Aalst [27], and Soffer et al. [31]. Although the potential efficiency benefits of using configurable process models during the implementation of enterprise systems are highlighted by all the authors [8,13,31], the suggested usage of the three approaches remains on the conceptual level. Only in [13] a potential applicability of configurations to executable workflow models such that the configuration can be enacted using a workflow engine is indicated.

We followed up on this in our previous research [2,17] where we derived the general applicable methodologies of *blocking* and *hiding* for configuring a workflow model from a systematic study of inheritance notions in the context of business processes [1,5], and compared these ideas with the ideas of Rosemann and van der Aalst. In this paper, we now added an approach for using these previously derived methodologies to form concrete modelling languages.

The idea of providing configurable workflow models as suggested here implies to have different variants of the process in different contexts. Of course, a context and thus also the required workflow configuration can change over time which then requires the transfer of running workflow instances from one configuration to another. Systems tackling these problems are also called configurable, reconfigurable or adaptive workflow systems [11,18,19,20,25,32], but they typically neglect the preceding aspect of how the change of the workflow model can be supported. For our approach, we just rely on the mechanics provided by the version management of SAP's WebFlow engine and therefore neglect this aspect.

5 Summary & Outlook

In this paper we demonstrated the advantages of the integration of several workflow templates into a single workflow template from which then workflow variants can be derived by means of configuration. To make a workflow modeling language configurable the elements representing *actions* and the *ports* through which cases are routed through these actions must be identified. These ports which are representing runtime alternatives can then be configured as either *enabled* to allow the normal execution of the action, be *hidden* to skip the particular action, or be *blocked* to prevent any flow of cases in the direction of the action. *Requirements* on the configuration ensure the configuration's applicability on the workflow. A *default configuration* enables the usage of a configurable workflow template even without any configuration effort and serves as the starting point for any configuration.

To show how easily these ideas can be used in existing workflow modeling languages, we applied the concepts to the block-structured workflow definition language of SAP's workflow engine WebFlow which is part of every SAP enterprise system installation since the mid nineties. SAP's WebFlow engine comes with a huge set of pre-defined workflow templates. We used this set to demonstrate that the merging of similar workflow templates into configurable workflow models allows to individually combine the strengths of different templates during the workflow implementation.

In future research, we have to show that our ideas are also applicable to non-block-structured workflow modeling languages. For this purpose, we are currently applying these ideas onto YAWL, an open-source workflow system supporting much more patterns than SAP workflow [3,4,35]. Given the openness of YAWL also the implementation of configuration functionalities as a software tool will be easier in the YAWL environment than in SAP.

To provide further assistance for the configuration of workflow models, we aim at integrating the idea of configurable workflow modeling languages into a configuration framework, allowing on the one hand to use advanced decision-making tools for performing the configuration [26], and on the other hand a synchronized configuration between the workflows and other system components as e.g. software applications.

Acknowledgements

The authors would like to thank Marcello La Rosa for his feedback on the general methodology presented in this paper.

References

1. W.M.P. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(1-2):125–203, January 2002.
2. W.M.P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M.H. Jansen-Vullers. Configurable Process Models as a Basis for Reference Modeling. In C. Bussler and A. Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 512–518. Springer Verlag, February 2006.
3. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
4. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
5. T. Basten and W.M.P. van der Aalst. Inheritance of behavior. *Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.
6. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuropka. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In *Proceedings of the 15th IRMA International Conference*, New Orleans, 2004. Gabler.
7. J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In *Proceedings of the Reference Modeling Conference 2006*, Passau, 2006. to appear.
8. J. Becker, P. Delfmann, T. Rieke, and C. Seel. Supporting Enterprise Systems Introduction through Controlling-enabled Configurative Reference Modeling. In *Proceedings of the Reference Modeling Conference 2006*, Passau, 2006. to appear.
9. P. Bernus. Geram: Generalised enterprise reference architecture and methodology version 1.6.3. Technical report, IFIPIFAC Task Force on Architectures for Enterprise Integration, March 1999.
10. T. Calders, S. Dekeyser, J. Hidders, and J. Paredaens. Analyzing Workflows Implied by Instance-dependent Access Rules. In *Proceedings of the 25th ACM SIGMOD (PODS)*, pages 100–109, Chicago IL, USA, 2006. ACM.
11. I. Classen, H. Weber, and Y. Han. Towards Evolutionary and Adaptive Workflow Systems-infrastructure Support Based on Higher-Order Object Nets and CORBA. In *Proceedings of the 1st International Enterprise Distributed Object Computing Conference (EDOC '97)*, pages 300–308, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

12. T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model.* Prentice Hall, Upper Saddle River, NJ, USA, 1998.
13. A. Dreiling, M. Rosemann, and W.M.P. van der Aalst. From Conceptual Process Models to Running Workflows: A Holistic Approach for the Configuration of Enterprise Systems. In *Proceedings of the 9th Pacific Asia Conference on Information Systems*, pages 363–376, Bangkok, Thailand, 2005.
14. M. Dumas and A.H.M. ter Hofstede. UML activity diagrams as a workflow specification language. In M. Gogolla and C. Kobryn, editors, *Proc. of the 4th Int. Conference on the Unified Modeling Language (UML01)*, volume 2185 of *LNCS*, pages 76–90, Toronto, Canada, October 2001. Springer Verlag.
15. P. Fettke and P. Loos. Classification of Reference Models – a Methodology and its Application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.
16. P. Fettke, P. Loos, and J. Zwicker. Business Process Reference Models: Survey and Classification. In C. Bussler and A. Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 469–483, Berlin Heidelberg, February 2006. Springer Verlag.
17. F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. Configurable Process Models – A Foundational Approach. In *Proceedings of the Reference Modelling Conference 2006*, pages 51–66, Passau, Germany, 2006.
18. Y. Han, T. Schaaf, and H. Pang. A Framework for Configurable Workflow Systems. In *Proceedings of the 31st International Conference on Technology of Object-Oriented Language and Systems*, pages 218–224, Los Alamitos, CA, USA, 1999. IEEE Computer Society.
19. Y. Han, A. Sheth, and C. Bussler. A Taxonomy of Adaptive Workflow Management. In *Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work*, Seattle, Washington, USA, November 1998.
20. P. J. Kammer, G. A. Bolcer, R. N. Taylor, A. S. Hitomi, and M. Bergman. Techniques for Supporting Dynamic and Adaptive Workflow. *Computer Supported Cooperative Work (CSCW)*, V9(3):269–292, November 2000.
21. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
22. V. B. Misic and J. L. Zhao. Evaluating the Quality of Reference Models. In A.H.F. Laender, S.W. Liddle, and V.C. Storey, editors, *19th International Conference on Conceptual Modeling*, volume 1920 of *Lecture Notes in Computer Science*, pages 484 – 498, Salt Lake City, Utah, October 2000.
23. E. S. Raymond. The CML2 Language, 2000. `http://catb.org/esr/cml2/cml2-paper.html`.
24. A. Rickayzen, J. Dart, C. Brennecke, and M. Schneider. *Practical Workflow for SAP – Effective Business Processes using SAP's WebFlow Engine.* Galileo Press, 2002.
25. S. Rinderle, M. Reichert, and P. Dadam. Disjoint and Overlapping Process Changes: Challenges, Solutions, Applications. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*, volume 3290, pages 101–120, January 2004.
26. M. La Rosa, J. Lux, S. Seidel, M. Dumas, and A.H.M. ter Hofstede. Questionnaire-driven Configuration of Reference Process Models. In *Proceedings of the 19th*

Conference on Advanced Information Systems Engineering (CAiSE 2007), pages 424–438, Trondheim, Norway, 2007.

27. M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, March 2007.
28. SAP AG. *SAP Library – Workflow Scenarios in Travel Management (FI-TV)*, 2006. `http://help.sap.com/saphelp_erp2005vp/helpdata/en/d5/202038541ec006e10000009b38f8cf/frameset.htm`.
29. SAP AG. SAP History: From Start-Up Software Vendor to Global Market Leader, April 2007. `http://www.sap.com/company/history.epx`.
30. S. Seidler. From Business Process to SAP Configuration - Deployment of ARIS for SAP Netweaver and SAP Solution Manager in Projects. ARIS Platform Expert Paper, IDS Scheer AG, September 2006.
31. P. Soffer, B. Golany, and D. Dori. ERP modeling: a comprehensive approach. *Information Systems*, 28(6):673–690, September 2003.
32. S. Tam, W.B. Lee, W.W.C. Chung, and E.L.Y. Nam. Design of a re-configurable workflow system for rapid product development. *Business Process Management Journal*, 9(1):33–45, February 2003.
33. O. Thomas. Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation. In C. Bussler and A. Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 484–496. Springer Verlag, 2006.
34. S.A. White et al. Business Process Modeling Notation (BPMN), Version 1.0, 2004.
35. YAWL Home Page. `http://www.yawlfoundation.org/`.