Electronic Communications of the EASST
Volume 6 (2007)

Proceedings of the Sixth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2007)

Evaluating Workflow Definition Language Revisions with Graph-Based Tools

René Wörzberger, Markus Heller, Frank Häßler

12 pages

Guest Editors: Karsten Ehrig, Holger Giese
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Evaluating Workflow Definition Language Revisions with Graph-Based Tools

René Wörzberger, Markus Heller, Frank Häßler

(rwoerz|heller|haessler)@i3.informatik.rwth-aachen.de
Lehrstuhl für Informatik 3 (Softwaretechnik)
RWTH Aachen, Germany

Abstract: In industry, there are many workflow management systems (wfms) which have been incrementally developed over a long time. The workflow definition languages that come along with these wfms are mostly graph oriented. These incrementally developed definition languages sometimes lack important modeling constructs as well as a clear conceptual foundation. Any workflow definition language revision has to be evaluated against end user's acceptance before implementation in the respective wfms.

In this paper, we report about an industrial application of graph-based techniques in the workflow domain. We present an evaluation environment which has been developed in a graph-based rapid prototyping approach. The evaluation environment comprises editing support for workflow definitions conforming to the language's revision. Furthermore, it provides a translator that maps definitions from the revised to the original language. Thus, the wfms can be used to enact definitions of the revised language without modifying its implementation.

Keywords: workflow, graph transformation

1 Introduction

Workflow Management Systems (wfms) are software systems that target at the support for collaborative processes. They usually provide a graphical language for workflow definitions wherein nodes represent the activities that constitute a process type and edges determine the possible activity execution sequences. A workflow definition can be instantiated in a wfms in order to support one single process of a certain type. In a workflow instance, activities are executed in one of the possible sequences by humans or software services which thereby produce instance specific data.

In this paper, we report about an industrial *application* of graph-based methods in the workflow modeling domain. Our industrial partner develops a commercial workflow management system (WfMS) which is embedded in a large enterprise resource planning system (ERP). The ERP and WfMS are continuously developed. It is particularly difficult to completely anticipate all requirements for the workflow definition language of the WfMS. Our industrial partner wanted to improve their workflow modeling language to reflect new requirements gathered by their modeling consultants in recent years while adapting the WfMS to customers' needs.

Despite its current shortcomings, longtime customers are used to the workflow definition language to some degree. Hence, replacing the language by some standard language (e.g. UML, BMPN) is not an option for our partner. Furthermore, changes in the definition language have to be introduced with extreme care. Together with our industrial partner, we have revised the workflow definition language of the WfMS and developed an Evaluation Environment (EE) with the benefit, that modeling experts can use the EE to extend their modeling language in a try-and-test manner. They can model workflow definitions in the revised workflow modeling language with the WDE and can discuss their suggestions with consultants or customers by using realistic examples. Using this approach, cost and time consuming modifications of the WfMS' implementation for evaluation purposes can be reduced. The Evaluation Environment described in this paper is based on the graph-based rapid prototyping approach (PROGRES, UPGRADE, GRAS) developed at our department.

The paper is structured as follows: Section 2 describes the original workflow definition language of the WfMS and points out some of its current deficiencies and possible improvements in a revised language. Section 3 describes the design of the evaluation environment which provides editing support for workflow definition in the revised language and a translator that converts workflow definitions in the revised language to definitions in the original language. Section 4 sketches similar approaches of other research groups. The paper concludes with Section 5.

2 Problem Description

This section briefly describes the workflow definition language of the WfMS (original language) thereby pointing out some language features whose modification in the revised language is also depicted.

2.1 Original Workflow Definition Language

The WfMS provides a language that allows for modeling of workflow definitions as typed and directed graphs. The language is separated in two modeling-layers with two different types of workflow definitions.

People to application (P2A) workflow definitions. A P2A workflow definition normally models a rather short-running interaction between a single human workflow participant and technical resources, i.e. other software modules of the ERP. Usually, these definitions are used to model possible sequences of GUI-dialogs which constitute the interface between a workflow participant and technical resources.

Figure 1 depicts an example of a telephone marketing process modeled as a P2A workflow definition. It contains all existing model elements that can be used in P2A workflow definitions.

- A *P2A start activity* (big octagon) marks the unique start of a P2A workflow and has no incoming transitions. Phone marketing in Figure 1 is an example for a P2A start activity. At runtime, this activity is invoked at first when the P2A workflow definition is instantiated. It has no other effect but to invoke its successor activities.

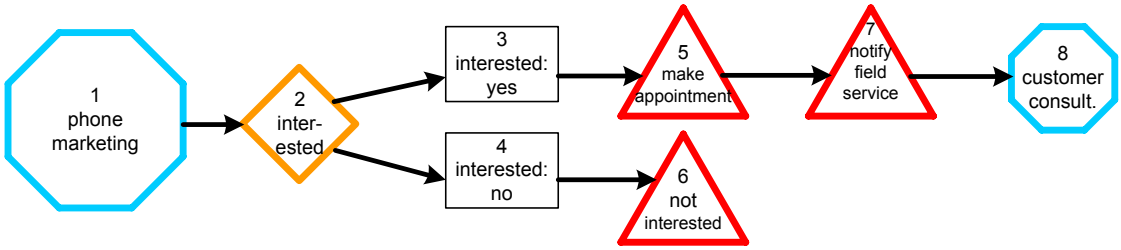


Figure 1: Example of a P2A workflow definition

- A *P2A call activity* (triangle) represents the runtime invocation of a hard coded subprogram in a software module of the ERP. A subprogram can run either with or without workflow participant interaction. In the former case the workflow participant can enter data via activity specific GUI forms.
- A *P2A XOR-decision* (rhomb) models a branching between two or more alternative paths of P2A activities. The example in Figure 1 shows the *P2A XOR-decision* interested which reflects the decision whether the phoned customer is interested in further information or not. Every P2A workflow instance can only follow exactly one of these alternative paths. The decision is made at runtime by a workflow participant with a standard GUI that lists the alternatives. In the workflow definition, each selectable alternative is represented by a *P2A alternative* (box) which is also the start of each alternative path. Therefore, P2A XOR-decisions can only be succeeded by P2A alternatives. In the example, there are just two P2A alternatives: yes and no.
- A *P2A reference* (small octagon) represents the instantiation of another P2A workflow definition. The new P2A workflow instance is automatically assigned to the same workflow participant that interacts with the referencing P2A workflow instance. Paths of P2A activities can either end in a P2A call activity like not_interested in the example or in a P2A reference. In the former case the P2A workflow instance terminates after termination of the last P2P call activity. In the latter case the P2A workflow instance with the reference is suspended but does not terminate.

The listed model elements can be connected with directed *transitions* (arrows) which specify the order of invocation and associate a P2A XOR-decision with its P2A alternatives, respectively.

People to people (P2P) workflow definitions. P2P workflow definitions model interactions among human workflow participants. The only model elements on this layer are the transition and the *P2P activity* (octagon).

Every P2P activity refers to one P2A start activity. Invoking a P2P activity in a P2P workflow instance creates a new P2A workflow instance. The P2A workflow definition of this instance is uniquely identified by the referenced P2A start activity. Termination of the P2A workflow instance conversely terminates the referencing P2P activity.

P2P activities are assigned to one or more workflow participants. At runtime, workflow participants receive a notification when one of their assigned P2P activity is invoked. P2P activities are

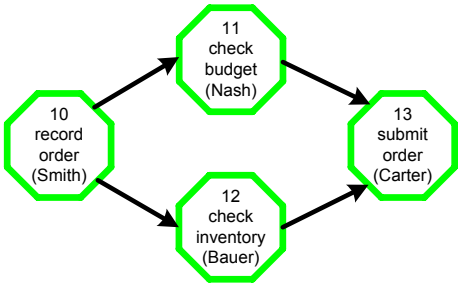


Figure 2: Example of a P2P workflow definition

connected with directed transitions. A P2P activity is invoked when all preceding P2P activities have terminated. A P2P activity signals its termination to *all* succeeding P2P activities.

Figure 2 shows an example of a P2P workflow definition. All P2P activities refer to P2A workflow definitions which are not shown for brevity. Assigned workflow participants are written in brackets under the name of each P2P activity. The termination of record order invokes both check budget and check inventory. After termination of the latter two submit order is invoked.

2.2 Shortcomings of the Original Language

The WfMS has been repeatedly adapted to new requirements in recent years according to newly risen customer needs. Therefore, its workflow definition language now has a less clear conceptual design compared to other workflow definition languages which have been developed from scratch. In the following, we describe some of these shortcomings.

2.2.1 No Real Decomposition Hierarchy

The P2P layer and the P2A layer beneath it have a decomposition relationship inasmuch that a P2A workflow definition can be considered as a refinement of the corresponding P2P activity. Furthermore, P2A workflow definitions can be refined by using P2A references to some degree. However, there is no way to model decompositions within the P2P layer, i.e. to refine a complex P2P activity by another P2P workflow definition.

2.2.2 XOR in the P2P Workflow Definitions

Alternative paths in P2A workflow definitions can be modeled via P2A XOR-decisions. Currently, there is no equivalent model element for P2P workflow definitions. Instead, decisions between alternative P2P activity paths have to be modeled by a pattern of several P2P and P2A modeling elements.

2.2.3 No Explicit Start and End Elements

In contrast to other workflow definition languages, workflow definitions of the WfMS have no dedicated model elements that mark the origin and end of the control flow. This is particularly disadvantageous in P2P workflow definitions where those P2P activities are determined to be

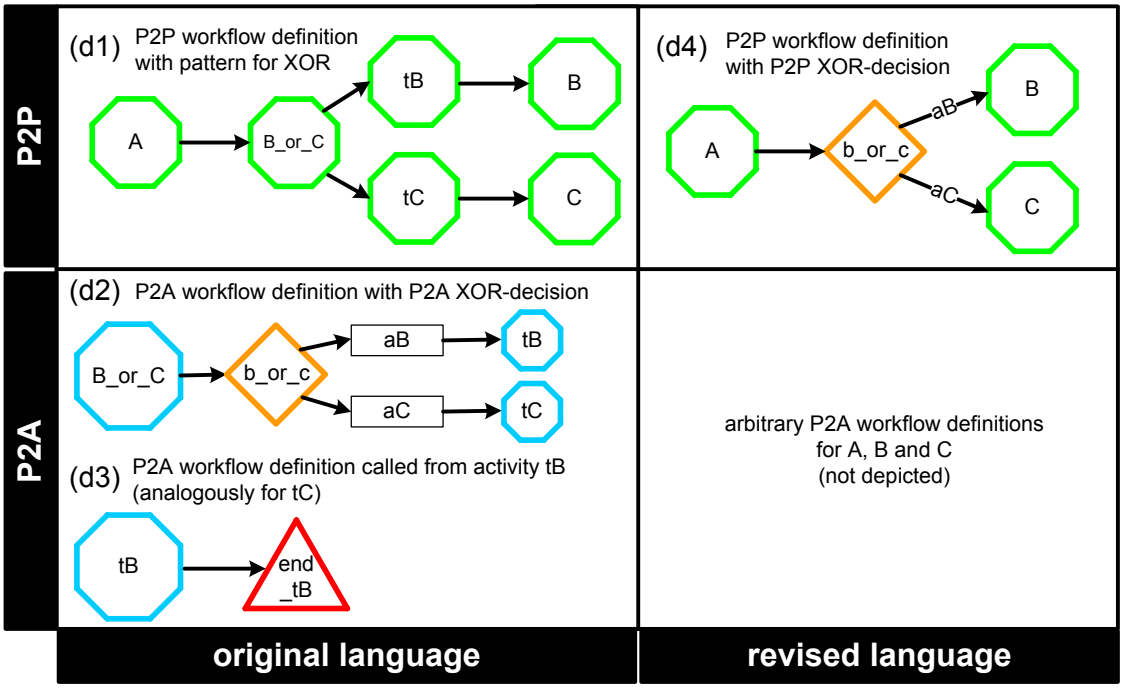


Figure 3: Modeling XOR-decisions in P2P workflow definitions

start (end) activities that have only outgoing (incoming) edges but do not further differ from other P2P activities.

Figure 3 shows how XOR-decisions in P2P workflow definitions are modeled in the original (left hand side) and the revised workflow definition language (right hand side). The P2P workflow definition (d1) contains a P2P activity B_or_C with two outgoing transitions to alternative paths. The P2P activity B_or_C invokes a P2A workflow definition (d2) which contains a P2A XOR-decision b_or_c. Each alternative path ends in a P2A reference (e.g. tB) which instantiates a P2A workflow definition exemplified by (d3). This P2A workflow instance is assigned to the same workflow participant as the instance of (d2). It just prompts the workflow participant for termination via P2A activity end_tB.

Termination of the instance of (d3) leaves the instance of (d2) in a suspended state. Consequently, the corresponding P2P activity B_or_C in the instance of (d1) is left suspended and does not invoke any successor P2P activities. Furthermore, the P2P activity tB in (d1) terminates due to the termination of the instance of (d3) and invokes its successors, B in this case, which can be assigned to a different workflow participant.

2.3 Revised Workflow Definition Language

In Subsection 2.2 we described some of the shortcomings of the original workflow definition language. These shortcomings were subject of a revision of the workflow definition language [Häß05] with two major goals: (1) to eliminate the deficiencies described above and (2) to keep

most model elements and language properties, particularly the graph oriented paradigm and the distinction between P2P and P2A workflow definitions. The second goal is due to the requirement that users which are familiar to the original language should be able to quickly adapt to the revised one. Furthermore, we have to assure that workflow definitions in the revised language can be automatically translated into the original language (s. Subsection 3.4). The first goal is addressed as follows:

- Decompositions can be modeled in the revised language by means of a new model element named *P2P subflow call*.
- In addition, the revised language provides dedicated model elements for both language layers that are used to explicitly indicate the start and end of the control flow within a workflow instance.
- Furthermore, the revised language allows for expressing decisions in P2P workflow definitions, too.

The right hand side of Figure 3 depicts a P2P workflow definition (d4) in the revised language which is equivalent to the pattern in the original language on the left hand side. The workflow definition (d4) has a *P2P XOR-decision* b_or_c which is incident to two *P2P XOR-transitions*. Each of the P2P XOR-transition carries an attribute (aB, aC) that represents the respective alternative.

A comparison of both sides in Figure 3 clearly shows that the new model element types help reducing the complexity of workflow definitions if a decision on the P2P layer has to be modeled.

The revised workflow definition language represents just one among many possible language derivatives. To evaluate each alternative by implementing a dedicated variant of the WfMS cannot be afforded. In the sequel, we present an alternative approach that leaves the WfMS as it is but provides support for the revised workflow definition language.

3 Evaluation Environment (EE)

In order to evaluate the revised workflow definition language without changing the implementation of the WfMS, we have developed an Evaluation Environment (EE) which provides (a) a workflow definition editor (WDE) to model workflow definitions according to the revised language and (b) a workflow definition translator (WDT) to translate workflow definitions from the revised to the original language.

3.1 Graph-Based Rapid Prototyping Support

Both the WDT and the WDE have been developed using a *rapid prototyping approach* which is developed at our department for several years now. The approach facilitates the specification of graph-grammar specifications in PROGRES [SWZ99] as well as the rapid construction of prototypical applications based on the specification. Briefly, a PROGRES specification comprises the following parts: (1) A *graph schema* that defines a set of valid graphs. This is done by the declaration of node and edge types and the definition of valid combinations of their respective

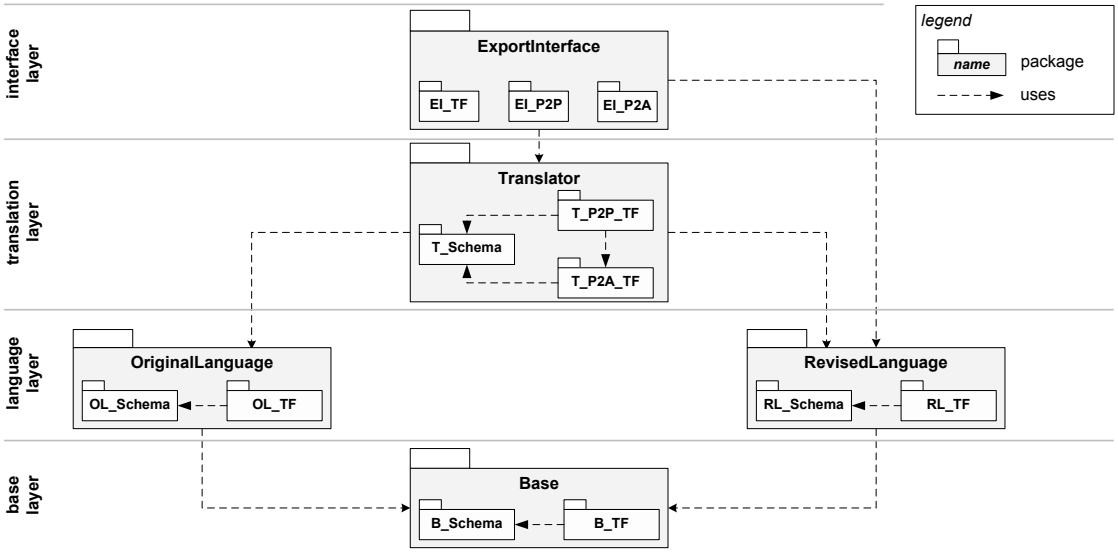


Figure 4: Package structure of the PROGRES specification

instances, (2) a set of *graph transformations* that transform a valid graph into another, and (3) a set of *graph queries* that test a graph for certain properties.

PROGRES can generate executable C-code from a PROGRES specification. The C-code is embedded into the rapid prototyping framework UPGRADE [BJSW02] resulting in an executable UPGRADE prototype. The prototype allows for displaying multiple views of a graph instance (host graph) stored in a graph database and manipulating this host graph by invoking transformations of the application logic resp. PROGRES specification.

The described graph-based prototyping framework has been successfully applied in a number of research projects at our department, for example, to develop an advanced workflow management system for dynamically evolving development processes for different application domains (e.g.software engineering, chemical engineering) ([HJS+04]).

The application logic of both the WDT and the WDE is fully specified in a single PROGRES specification. Based on this specification, the Evaluation Environment is realized as an UPGRADE prototype.

3.2 Architecture

Figure 4 presents the coarse architecture of the PROGRES specification for the Evaluation Environment (EE). The parts of the PROGRES specification are organized in packages modeled as sections and subpackages which are associated with architectural layers. The use-relationship indicates which parts of a package are used in which other package.

The package Base contains two subpackages. B_Schema just comprises the declaration of the node class `ELEMENT` (a node class is a type that is not instantiatable). The class `ELEMENT` serves as the root of a class hierarchy for all model elements of workflow definitions in the original as well as the revised language and declares common attributes like `name` storing the name of a

model element. B_TF contains very basic graph transformations, like `SetElementName` which sets the name of a given element.

Package RevisedLanguage includes the schema part for model elements in the revised languages (RL_Schema) and transformations that a can be applied to workflow definitions in the host graph (RL_TF). The package OriginalLanguage is build up analogously.

Translator is the package that encompasses all transformations which constitute the translation algorithm. Only one edge type `transformed_to` is declared in T_Schema which is used for embedding transformed edges in the workflow definitions of the original language. Transformations for translating P2P and P2A workflow definitions are located in dedicated subpackages T_P2P_TF and T_P2A_TF, respectively.

The package ExportInterface covers all transformations that can be invoked from the UPGRADE prototype. Edit operations concerning P2P workflow definitions in the revised language are located in the subpackage EI_P2P whereas transformations for P2A workflow definitions are in subpackage EI_P2A. EI_TF only contains a single transformation that starts a translation run.

It is the main goal of the architecture to accommodate future changes of the revised languages in the PROGRES specification with local modifications only. Such changes ideally imply modifications in RevisedLanguage and Translator only (whereas OriginalLanguage and Base can remain unchanged). In the same manner, the package ExportInterface needs to be modified when extending the revised language with new workflow edit operations.

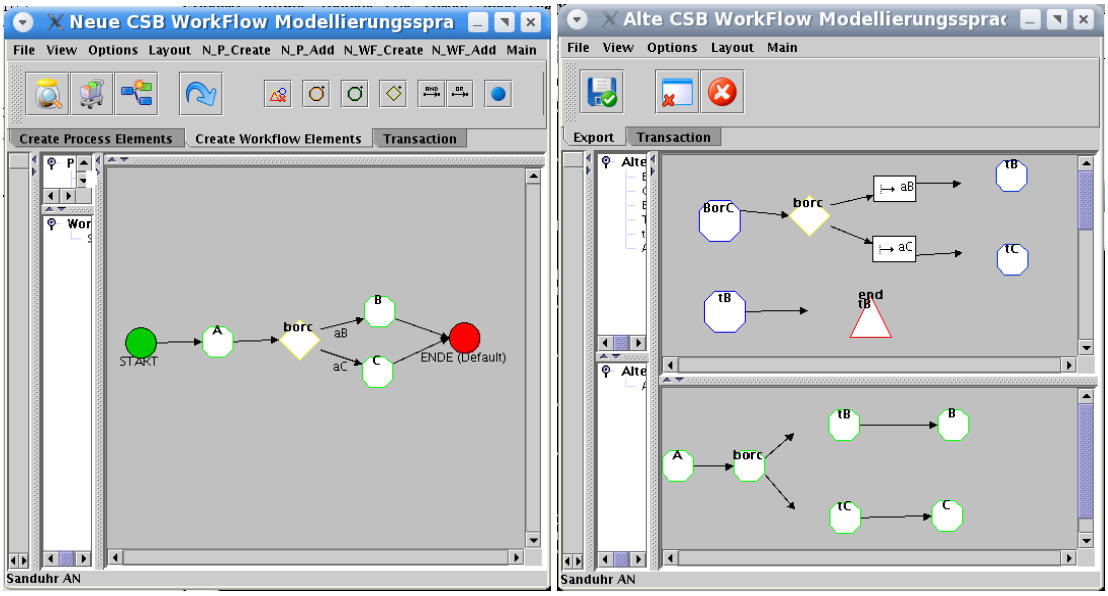
3.3 Workflow Definition Editor (WDE)

The PROGRES specification contains several graph transformations which constitute the interface of the workflow definition editor (WDE). Each graph transformation represents an edit operation on the host graph that contains all workflow definitions in the revised language. Figure 5(a) is a screenshot of the WDE captured during modeling of the P2P workflow definition (d1) of Figure 3. The WDE provides toolbars with buttons each of which executes a transformation, e.g. insertion of a P2P activity in a P2P workflow definition.

At every point in time, a workflow definition modeled with the WDE complies with certain conformance rules which can be subdivided in two sets:

- *Strong conformance rules (SCR)* are rules which every workflow definition of the revised language has to comply with at any time. For instance, this rule set includes a rather common rule “Every transition has a source and a target”, but also a rule “The outgoing transitions of a P2P XOR-decision can be only P2P XOR-transitions” which is specific to the revised language. The violation of SCR is immediately signaled to the WDE’s user by means of a message box.

- *Weak conformance rules (WCR)* are rules which every workflow definition has to comply with in order to be a valid expression of the revised language, but which can be violated *temporarily*. The rule “A P2P XOR-decision has at least two outgoing P2P XOR-transitions” is a weak conformance rule. Even though supporting the user to model valid workflow definitions in the revised language is clearly a goal of the WDE, the permission of temporary violations of WCR eases the modeling task. In the mentioned example the user would be forced to create the context of a P2P XOR-decision before creation of



(a) Workflow Definition Editor (b) Result of a Workflow Definition Translator run

Figure 5: UPGRADE Prototype

the P2P XOR-decision could take place. Such restrictions might unnecessarily contradict user's modeling preferences. Hence, violations of WCR are not signaled to the user before the workflow definitions are translated (s. Subsection 3.4).

The distinction between SRC and WCR is not specific to our EE but likely to be applicable for similar cases, i.e. WDEs for revisions of other workflow definitions languages.

3.4 Workflow Definition Translator (WDT)

The host graph of the Evaluation Environment maintains graph structures for workflow definitions according to the revised modeling language and graph structures for workflow definitions in the old modeling language. The workflow definition translator (WDT) translates workflow definitions form the revised language into workflow definitions in the old language. These newly generated workflow definitions can then be imported and executed in the CBS-WFMS. The translation algorithm is specified as a set of graph transformations (in package *Translator* in Figure 4). A translation run can be initiated from within the WDE prototype as an automated batch run without any user interaction. The translation algorithm of the WDT comprises several steps:

1. Check conformance of all workflow definitions against WCR. Stop if there is a non-conforming workflow definition.

2. For each P2A workflow do

   (a) Transform all P2A call activities

(b) Transform all P2A references

(c) Transform all P2A XOR-decisions

(d) Transform all P2A transitions

3. For each P2P workflow do

   (a) Transform all P2P activities

   (b) Transform all P2P XOR-decisions

   (c) Transform all P2P transitions

Some characteristics of the WDT concerning the sequence of translation steps are described in the following:

1. *Language layers and coarse grained relationships.* The WDT works bottom up inasmuch it first transforms the P2A and afterwards the P2P workflow definitions (step 2 and 3). This sequence is crucial since P2A workflow definitions are referenced by P2P workflow definitions. Generally speaking, the sequence is aligned to coarse grained reference relationships between language layers.

2. *Nodes and edges.* Both the revised and the original workflow definition language are graph oriented. Therefore, model elements can be specified as node elements and edge elements. During translation a directed edge has to be embedded between a source and a target node at any time whereas a node might exists at least temporarily without any incident edges. Thus, it is necessary to transform model elements classified as nodes (e.g. P2A call activities in step 2a) before those model elements classified as edges (e.g. P2A transition in step 2d).

3. *Model element types.* Each coarse grained translation step is decomposed into fine grained steps (2a to 2d and 3a to 3c). Every fine grained translation step transforms model elements of only one type. This is a consequence of the fact that mainly the type of a model element (e.g. P2P XOR-decision) determines the static specification of its translation while other model element properties (e.g. number of incident activities) just affect the execution dynamics (e.g. actual loop iterations).

The described criteria can also be applied in the specification of the EE for revisions of other workflow definition languages. Therefore, the coarse design of the algorithm can be reused in other cases.

The result of a WDT run is a set of workflow definitions in the original language which are also constructed as graph structures in the host graph during the transformator run. The UPGRADE prototype provides an export function that serializes these generated workflow definitions into a file according to a proprietary file format readable by the WfMS. This export function has been implemented in Java and is independent of the specification of the revised language. Thus, future revisions of the revised language will surely lead to changes in the PROGRES specification but not in the export function.

4 Related Work

The WDE presented in this paper allows for editing workflow definitions whose validity is partly checked during editing and fully checked before translation against certain conformance rules specified within the PROGRES specification. We shortly discuss important related work with the focus on the workflow domain.

In the workflow area, there are various groups working on translators between *different* workflow definitions languages. The following approaches do not use graph-based techniques. Van d. Aalst et al. [ODHA06] describe an algorithm that translates workflow definitions from the graph oriented Business Process Modeling Notation (BPMN) used for notation of workflow definitions to the block-structured Business Process Execution Language (BPEL) which specifies workflow implementations on top of web services. Guelfi and Mammar [GM06] provide a set of formal transformation rules for mapping UML activity diagrams to workflow definitions in the workflow definition exchange format XPDL. [Sch02] describes an UPGRADE prototype specified in PROGRES for editing dynamic task nets together with the transformation of workflow definitions modeled as UML class diagrams and collaboration diagrams into PROGRES code.

The proposed approach with the WDT unidirectionally translates between different versions of *one* graph oriented workflow definition language without user interaction based on a graph-based approach.

The focus of this paper was on reporting about an *application* of graph-based techniques in industrial application domains and not on the development of new *general* transformation approaches. Therefore, the workflow definition translator (WDT) of the presented approach works in a rather straightforward manner in a automated batch mode without user interaction. For more advanced incremental bidirectional and interactive integration frameworks see [BHW05, KS06].

5 Conclusion and Impact

In this paper we have reported about an industrial *application* of graph-based methods in the domain of workflow modeling. Our industrial partner wanted to revise the workflow definition language of their commercial workflow management system (WfMS) according to additional requirements. Using our graph-based tools for rapid prototyping (PROGRES, UPGRADE and GRAS), we have developed an Evaluation Environment which offers an editor (WDE) for workflow definitions in the revised language. Furthermore, it provides a translator (WDT) that translates these definitions back to definitions in the original language for enactment within the unmodified WfMS. This approachs helps reducing costly experimental reimplementation cycles for the WfMS.

The modeling experts and development teams for the WFMS have used the Evaluation Environment to model sample workflows in the described manner. Within the EE, the improvements contained in the revised language were evaluated and, by the time of writing, some of them have already been adapted in the implementation of the commercial WfMS.

Using PROGRES and UPGRADE as technical foundation for the Evaluation Environment has proven to be a good decision due to the facts that the WDE as well as the WDT could be rapidly

developed in a relatively short time-span. In the future, the investigation of workflow definition language translation as described in this paper serves as preliminary work for the development of a dynamic workflow management systems based on existing static wfms [HNW07].

Bibliography

[BHW05] S. M. Becker, T. Haase, B. Westfechtel. Model-based a-posteriori integration of engineering tools for incremental development processes. *Software and Systems Modeling* 4:2, 2005.

[BJSW02] B. Böhlen, D. Jäger, A. Schleicher, B. Westfechtel. UPGRADE: A Framework for Building Graph-Based Interactive Tools. *Electronic Notes in Theoretical Computer Science* 72:2, 2002.

[GM06] N. Guelfi, A. Mammar. A formal framework to generate XPDL specifications from UML activity diagrams. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*. 2006.

[Häß05] F. W. Häßler. Analyse und Erweiterung einer bestehenden Workflow-Modellierungssprache. Master's thesis, RWTH Aachen University, 2005.

[HJS+04] M. Heller, D. Jäger, M. Schlüter, R. Schneider, B. Westfechtel. A Management System for Dynamic and Interorganizational Design Processes in Chemical Engineering. *Computers & Chemical Engineering* 29:1, 2004.

[HNW07] M. Heller, M. Nagl, R. Wörzberger. Dynamic Process Management Based Upon Existing Systems. In *Collaborative and Distributed Chemical Engineering Design Processes: From Understanding to Substantial Support*. LNCS. Springer, 2007. (to appear).

[KS06] A. Königs, A. Schürr. MDI - a Rule-Based Multi-Document and Tool Integration Approach. *Special Section on Model-based Tool Integration in Journal of Software and System Modeling*, 2006.

[ODHA06] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, W. M. P. van der Aalst. From BPMN Process Models to BPEL Web Services. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*. 2006.

[Sch02] A. Schleicher. *Management of Software Development Processes: An Evolutionary Approach*. PhD thesis, RWTH Aachen University, 2002.

[SWZ99] A. Schürr, A. Winter, A. Zündorf. The PROGRES Approach: Language and Environment. In *Handbook on Graph Grammars and Computing by Graph Transformation – Volume 2: Applications, Languages, and Tools*. World Scientific, 1999.