Modeling and Verifying Workflow-based Regulations

Daniel Fötsch[1], Elke Pulvermüller[2], and Wilhelm Rossak[2]

[1] University of Leipzig,
Department of Computer Science, Augustusplatz 10-11, 04109 Leipzig, Germany,
foetsch@informatik.uni-leipzig.de
[2] Friedrich-Schiller-University Jena,
Department of Computer Science, Ernst-Abbe Platz 1-4, D-07743 Jena, Germany,
[pulvermu|rossak]@informatik.uni-jena.de

Abstract. In this paper we present our approach to model and verify workflow-intensive systems. Besides the functional properties (given by the temporal workflow description) we augment the model and model checking with additional property treatment to deal with multifarious non-functional properties and property hierarchies. This enables a more powerful verification of requirements such as given business-driven regulations in these system workflows.

1 Introduction and Problem Statement

Workflow-intensive systems are systems consisting of hardware and software which focus on workflows, i.e. on the temporal order of processing documents and data.

Workflow-intensive systems require both a model incorporating the functional and non-functional properties as well as a temporal verification of (some of) these properties. The latter allows to ensure that certain (temporal) regulations hold in the system model. Symbolic time and order are an important issue in workflow-intensive systems. This characteristic makes them a suitable task for their verification by means of model checking.

Workflow models describe a set of activities and their relationships in a temporal order, start and termination criterions, and information about the individual activities, such as associated IT applications and data, etc. [1]. In the web service development, which is a specific workflow domain, workflows can be defined using XML-based (standardized) languages such as BPEL4WS, WSCI, and BPML. BPEL4WS [2] is quickly emerging as the language of choice for the description of process interactions.

However, in all these standardized languages (and also in BPEL4WS) no explicit support for the non-functional quality characteristics, e.g. efficiency or reliability, is available. There are no explicit quality guidelines and no explicit modeling elements are foreseen yet to capture or describe those in a structured way. But, modeling process rules and business- or government-driven regulations require both to consider functional (directly workflow related) and multifarious non-functional properties.

Applying **model checking**, the workflow system is modeled as a special kind of finite state automata, so-called Kripke automatas or Kripke models [3]. The relevant properties are connected to the states (labeling). Selected temporal requirements the system is expected to fulfill are formulated as temporal logic formulas. The temporal

logic languages used are CTL or LTL, for instance. Based on graph algorithms like reachability algorithms, the model checking algorithm determines if the given temporal requirement is fulfilled in the system model.

In the current practice, however, the world of Kripke automata with labelled states and transitions offers a limited potential for (non-functional) property treatment. Labels attached to states are atomic propositions. Neither their structure (e.g. to reflect a quality hierarchy) nor their operations can be explicitely defined. However, for multifarious properties different processing rules have to be taken into consideration within the model checking algorithms (e.g. the efficiency properties are processed differently to certain reliability properties).

In the following we describe our approach to XML-based modeling (extension to BPEL4WS) and our verification approach (extension to CTL model checking algorithms) to tackle the current problem of insufficient (non-functional) property treatment.

2 Model and Verification Extensions

To approach the above mentioned deficiencies both, existing modeling and verification techniques have to be adapted. The verification algorithms have to exploit the available model information to detail the verification results.

2.1 Non-functional Extensions in BPEL4WS Models

Figure 2 shows an example for a hierarchy of non-functional properties. To incorporate such non-functional properties in the workflow model, we augment the BPEL4WS model definition.

```
<?verifier type="propertyDef" name="quality" ?>
<?verifier type="propertyDef" name="reliability" superProperty="quality"?>
<?verifier type="propertyDef" name="maturity" superProperty="reliability" algebra="maturityAlg"?>
<?verifier type="propertyDef" name="faultTolerance" superProperty="reliability" ... ?>
<?verifier type="propertyDef" name="recoverability" superProperty="reliability" ... ?>
...
<?verifier type="algebraDef" name="maturityAlg" domain="developerExperienceDom"
   operations="developerExperienceDomOp"?>
<?verifier type="domainDef" name="developerExperienceDom" value="3,5,15"?>
<?verifier type="operationDef" algbraOpType="developerExperienceDomOp" name="AND"
   operator="opANDsemantic" parameter1="developerExperienceDom" parameter2="developerExperienceDom"?>
<?verifier type="operationDef" algbraOpType="developerExperienceDomOp" name="OR"
   operator="opORsemantic" parameter1="developerExperienceDom" parameter2="developerExperienceDom"?>
<?verifier type="operationDef" algbraOpType="developerExperienceDomOp" name="NOT"
   operator="opNOTsemantic" parameter="developerExperienceDom"?>
```

Fig. 1. Non-functional extensions for a BPEL4WS model definition.

To obey the BPEL4WS standard, the properties are added by means of processing instructions which can be processed by our transformation framework. These processing instructions are structured as depicted in figure 1. The structured instructions define an excerpt of the ISO 9126-1 quality model (cf. figure 2).

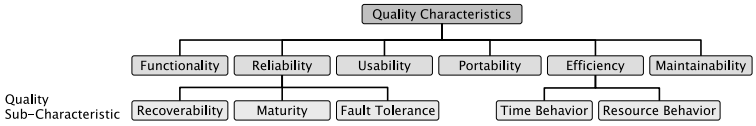

Fig. 2. Excerpt of the ISO 9126-1 Quality Model [4]

The main types of the instruction structure are the `propertyDef`, the `algebraDef`, the `domainDef` and the `operationDef` types. The `propertyDef` type defines a specific non-functional property type in a hierarchical manner referring to the super-property (`superProperty`). The `algebraDef` connects a user-defined algebra to a quality property. This algebra is defined with a value domain via the `domainDef` type and with corresponding operations via the `operationDef` type. The example in figure 1 details the maturity quality property. Its domain is defined as the number of years of development experience. The semantics behind this is the assumption that the higher the development experience the higher the maturity is.

The `opORsemantic`, `opANDsemantic` and `opNOTsemantic` character strings in figure 1 represent (each) a processing rule for the specific maturity property. The three Boolean operations `AND`, `OR` and `NOT` have to be defined as these are used by the model checking algorithms. In the case of the "developer experience" property, the `AND`-operation (character string `opANDsemantic`) may be defined as a maximum function, for instance. In syntax:

```
opANDsemantic :=
   AND(developerExperienceDom,developerExperienceDom)
```

and semantics: `AND(e1,e2) = e1 iff e1 > e2 else e2`.
The concrete operation semantic definition is user-defined and depends on the expected verification result semantic.

The BPEL4WS code depicted in figure 3 gives an example for the application of the extended non-functional properties in BPEL4WS activities.

```
<invoke partnerLink="shipping" portType="lns:shippingPT"
   operation="requestShipping" inputVariable="shippingRequest" outputVariable="shippingInfo">
   <?verifier type="property" name="maturity" value="5"?>
   <source linkName="ship-to-invoice"/>
</invoke>
```

Fig. 3. Example for applying the properties in BPEL4WS activities.

The activity defines a synchronous invocation with variable "shippingRequest" and return value "shippingInfo". This `invoke`-activity is augmented with an additional non-functional property: the information about the web service that it has been developed by somebody with equal or less than 5 years of development experience.

2.2 Transformation Framework

In order to verify the workflow definitions and also their non-functional properties, we implemented the framework illustrated in figure 4. Our framework realizes a multi-level transformation process from workflow definitions such as BPEL4WS in a formal intermediate representation and further in a specific input model to verification tools such as CoV [5]. As intermediate representation we use a guarded labelled automata stored in XML.

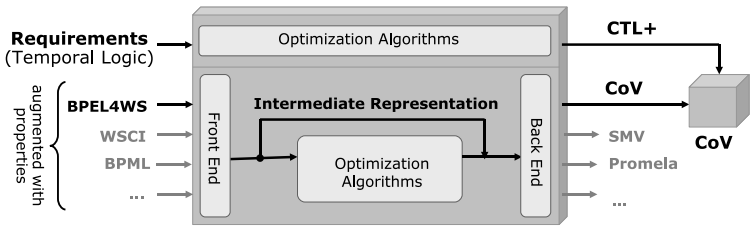

Fig. 4. Transformation Framework

To implement the transformations, we define new complex operators on top of elementary ones. This is provided by an XML transformation technology (similar to XSLT) applying the operator hierarchy concept which is discussed extensively in [6].

Besides the transformation of higher-level into verification specific models, the transformation framework enables to incorporate optimizing operations on the intermediate representation (e.g. minimizing product automatas for parallel workflow parts).

2.3 Model checking Extensions

Model checking CTL temporal formulas requires a special finite state automata as model input (besides the temporal formula). The presented transformation framework allows the transformation of the extended BPEL4WS workflow models into an extended Kripke model preserving the following information in addition to traditional Kripke structures (details may be found in [5]):

- Property algebras: Instead of simple atomic propositions of current Kripke models these extended Kripke models define property algebras. By this means, it is possible to model check user-defined types (i.e. user-defined domains and their operations). The operation `AND`, `OR` and `NOT` build the set of operations which have to be defined for each quality. The user-defined algebras form a parameter to the model checker.
- Property hierarchies: Instead of simple atomic propositions of current Kripke models a property tree is attached to the model elements (the states) reflecting the reality of quality hierarchies. The property assignment of the properties `p1` and `p2` to the states `s1` and `s2` may be defined hierarchically as follows, for instance: `s1 := p1, s2`; `s2 := p2` (hierarchically assigning `s2` to `s1` besides the regular leaf property `p1`).

3 Related Work

The transformation framework is based on an earlier XML-based transformation approach [7]. Other aspect-oriented invasive composition approaches based on XML (e.g. [8]) ignore operator hierachies and non-functional properties.

In [9] a requirement-driven approach is proposed for the design and verification of web services. The formal requirements are used to derive process skeletons in BPEL4WS and to validate the refined BPEL4WS process against the constraints described in the requirements applying the symbolic model checker NuSMV. A number of alternative approaches to validate workflow-intensive systems based on explicit state model checking and SPIN may be found in [10–12], for instance. However, all of these approaches do not support the validation of non-functional regulations and property hierarchies.

Details to the CoV model checker and its extended property treatment may be found in [5]. Other existing extensions to model checking algorithms consider special properties like *time* [13] or *dynamic variable assignment* [14]. Again, user-definable property algebras and the hierarchical non-functional property treatment are not considered. A related approach in user-definable property algebras may be found in [15] (based on Quasi-Boolean lattices). Their intention, however, is the realization of a multi-valued model checking to deal with the synthesis of different models of one and the same system.

4 Summary and Conclusion

In the web environment, standards (and BPEL4WS in particular) gain growing importance for the modeling of workflows. Besides the workflow models to capture the system, specific requirements to the system form the base for the verification of the model. These requirements are business- and government-driven regulations, for instance. As workflows are closely connected to time and order temporal requirements are natural. Therefore, model checking, which is suited for temporal logic, is the means of choice to verify that a requirement is satisfied in a given workflow model.

In our approach we extend the BPEL4WS definition language exploiting BPEL4WS processing instructions. Structured instructions allow the augmentation of non-functional and hierarchical properties with user-defined property algebras. A transformation framework converts the input into an intermediate XML format and further into the verifier input format. By means of an operator hierarchy different input formats may be processed and also different output formats may be produced. In our approach, we refer to the CoV output format. CoV is a model checker which is able to deal with property hierarchies and user-defined property algebras.

The intermediate model of our transformation framework does not yet handle some advanced BPEL4WS features (e.g. the correlation sets and dynamic process instantiation). Further work is also needed to support the optimization of the intermediate models (minimizing their size, for instance). We are currently working on optimizing operators. Moreover, the verification algorithm efficiency in open (web) systems can be improved directly. AI and compositional techniques are currently under investigation. The management of model evolution and the consequences for the reuse of existing verification results is a further domain for future research.

References

1. Workflow Management Coalition 2 Crown Walk, Winchester, Hampshire SO23 8BB, UK.: Terminology & Glossary – WFMC-TC-1011, Version 3.0. (1999)
2. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1, 5 May 2003. (2003)
3. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. 3 edn. The MIT Press, Cambridge, Massachusetts; London, England (2001)
4. ISO/IEC: FCD 9126-1.2: Information Technology – Software Product Quality. Part 1: Quality Model. (1998)
5. Pulvermüller, E.: Verifikation von Komponenten-basierten Systemen auf Basis eines erweiterten temporalen Verifikationsverfahrens. PhD thesis, Friedrich-Schiller-University of Jena, Jena, Germany (2006)
6. Fötsch, D., Speck, A., Hänsgen, P.: The Operator Hierarchy Concept for XML Document Transformation Technologies. Berliner XML Tage 2005 (BXML'05), 12.-14. September 2005 in Berlin. In Eckstein, R., Tolksdorf, R., eds.: Berliner XML Tage 2005, XML-Clearinghouse (2005) 59–70
7. Schonger, S., Pulvermüller, E., Sarstedt, S.: Aspect-Oriented Programming and Component Weaving: Using XML Representations of Abstract Syntax Trees. In: Proceedings of the 2nd German GI Workshop on Aspect-Oriented Software Development; - Technical Report No. IAI-TR-2002-1, Rheinische Friedrich-Wilhelms-Universität Bonn, Institut für Informatik III (2002) 59 – 64
8. Kessler Piveta, E., Zancanella, L.: Architecture of an XML-based Aspect Weaver. In: Proccedings of Workshop on Correctness of Model-based Software Composition (CMC), ECOOP 2003. No. 2003-13, Darmstadt, Germany, Universität Karlsruhe (2003) 9 – 14
9. Pistore, M., Roveri, M., Busetta, P.: Requirements-Driven Verification of Web Services. Electr. Notes Theor. Comput. Sci. (**105**) 95–108
10. Nakajiama, S.: Model-Checking Verification for Reliable Web Service. In: OOPSLA 2002 Workshop on Object-Oriented Web Services. (2002)
11. Kazhamiakin, R., Pistore, M., Roveri, M.: Formal Verification of Requirements using SPIN: A Case Study on Web Services. In: 2nd International Conference on Software Engineering and Formal Methods (SEFM 2004), 28-30 September 2004, Beijing,China, IEEE Computer Society (2004) 406–415
12. Fu, X., Bultan, T., Su, J.: Analysis of interacting BPEL web services. In: WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, ACM Press (2004) 621–630
13. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, Ph., McKenzie, P.: Systems and Software Verification: Model-Checking Techniques and Tools. Springer-Verlag (2001)
14. Cho, S.M., Kim, H.H., Cha, S.D., Bae, D.H.: Specification and Validation of Dynamic Systems Using Temporal Logic. IEE Proceedings Software **148** (2001) 135–140
15. Chechik, M., Devereux, B., Easterbrook, S., Gurfinkel, A.: Multi-Valued Symbolic Model-Checking. ACM Transactions on Software Engineering Methodology **12** (2003) 371–408