Anne Baumgraß, Nico Herzberg, Gerti Kappel, Jan Mendling,
Andreas Meyer, Stefanie Rinderle-Ma (Eds.)

Inter-Organizational Process Modeling and Event Processing in Business Process Management

CEUR-WS.org/Vol-1185/CompleteProceedings.pdf

Vienna, 19th March 2014
Workshop Proceedings

Volume Editors

Anne Baumgraß
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany
anne.baumgrass@hpi.de

Nico Herzberg
SAP Germany
Hasso-Plattner-Ring 7, 69190 Walldorf, Germany
nico.herzberg@sap.com

Gerti Kappel
Vienna University of Technology
Favoritenstrasse 9-11/E188-4, A-1040 Vienna, Austria
gerti@big.tuwien.ac.at

Jan Mendling
Vienna University of Economics and Business
Welthandelsplatz 1, A-1020 Vienna, Austria
jan.mendling@wu.ac.at

Andreas Meyer
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam, Germany
andreas.meyer@hpi.de

Stefanie Rinderle-Ma
University of Vienna
Währingerstraße 29, A-1090 Vienna, Austria
stefanie.rinderle-ma@univie.ac.at

Copyright © 2014 for the individual papers by the papers' authors. Copying permitted only for private and academic purposes. This volume is published and copyrighted by its editors.

Contents

Event Modeling and Processing in Business Process Management

1st International Workshop, EMoV 2014
Vienna, Austria, 19 March 2014

Preface

The first workshop on Event Modeling and Processing in Business Process Management (EMoV) was held in conjunction with Modellierung 2014 conference in Vienna, Austria, on 19 March 2014.

The workshop dealt with problem statements, solution proposals, and future development perspectives in the area of modeling business processes and their event-based process execution. The focus was set on executable process models and their dynamic interaction during execution based on events. Furthermore, the utilization of events for process monitoring and analysis were targeted by the workshop.

During the workshop, research results on event correlation, event-based monitoring, and proactive descision support were presented. Complementary, talks from the context of the EU-project GET Service (http://getservice-project.eu/) provided insights from the practioner's side. These talks were introduced by an invited talk given by Prof. Dr. Jan Mendling on challenges of event processing in logistics processes. Altogether, researchers and practioners from multiple domains lively discussed the presented research and industry papers. They identified potential collaborations in the fields of logistics and insurance by bringing newest research results to application. The program was framed by a keynote given by Dr.-Ing. Stefan Schulte from the Vienna University of Technology elaborating on elastic processes which are executed using cloud resources.

We received six research paper submissions. The program committee provided 26 reviews. These resulted in three accepted papers with an acceptance rate of 50

We want to thank the program committee for their detailed reviews and the hosts of Modellierung 2014 conference for their great support.

Potsdam, June 2014

Anne Baumgraß
Nico Herzberg
Andreas Meyer

Organization

EMoV 2014 was organized in conjunction with Modellierung 2014 at the University of Vienna, Austria.

Program Committee Chairs

Anne Baumgraß (Hasso Plattner Institute at the University of Potsdam)
Nico Herzberg (SAP Deutschland AG & Co.KG)
Andreas Meyer (Hasso Plattner Institute at the University of Potsdam)

Program Committee

Rafael Accorsi (University of Freiburg)
Thomas Allweyer (University of Applied Sciences Kaiserslautern)
Dimitris Apostolou (University of Piraeus)
Opher Etzion (IBM Haifa Research Lab)
Dirk Fahrland (Technical University of Eindhoven)
Bogdan Franczyk (University of Leipzig)
Avigdor Gal (Technion – Israel Institute of Technology)
Holger Giese (Hasso Plattner Institute at the University of Potsdam)
Georg Grossmann (University of South Australia, Adelaide)
Christian Janiesch (Karlsruhe Institute of Technology)
Stefan Krumnow (Signavio GmbH)
Niels Lohmann (University of Rostock)
Andre Ludwig (University of Leipzig)
Jan Mendling (Vienna University of Economics and Business)
Adrian Paschke (Freie Universität Berlin)
Frank Puhlmann (Bosch Software Innovations GmbH)
Manfred Reichert (University of Ulm)
Stefanie Rinderle-Ma (University of Vienna)
Stefan Sackmann (University of Halle)
Sigrid Schefer-Wenzel (FH Campus Wien)
Stefan Schulte (Vienna University of Technology)
Stefan Sobernig (Vienna University of Economics and Business)
Nenad Stojanovic (FZI Forschungszentrum Informatik, Karlsruhe)
Mark Strembeck (Vienna University of Economics and Business)
Rainer von Ammon (Centrum für Informations-Technology Transfer, Regensburg)
Barbara Weber (University of Innsbruck)
Matthias Weidlich (Imperial College London)
Mathias Weske (Hasso Plattner Institute at the University of Potsdam)
Uwe Zdun (University of Vienna)

Event Correlation for Business Processes on the Basis of Ontologies

Tobias Metzke
Hasso-Plattner-Institute at the University of Potsdam, Germany
tobias.metzke@student.hpi.uni-potsdam.de

Abstract: Business process execution generates great amounts of information on process results and intermediate activities. Most of this information is represented by events linked to their related process execution. Process monitoring uses such events and links to correlate incoming events to their corresponding process instances. These links may however get lost in the process of event capturing or not even be present in distributed process environments like logistics. Nonetheless, monitoring depends on the correct correlation of events in such scenarios. Furthermore, the correlation of external data like weather and traffic information, which has no connection to process executions, can be useful in the planning, execution and monitoring of processes as well. The approach presented in this paper uses semantic technologies to automatically identify those process executions that are related to the data of an occurring event. It uses linked data principles and graph-based algorithms to detect relatedness of events and process instances. The approach allows for the inclusion of external data and the correlation of external events without relying on process specific queries.
Keywords: Event Correlation, Semantic Event Processing, Business Processes

1 Introduction

Process monitoring is an essential method for improving a company's processes and procedures. In fully automated process environments the monitoring can be managed by process engines that keep track of triggered events and their origin. As stated in [HMW13], such logging mechanisms are however not always available and the correlation of events needs to be done based on the processes' context data like associated transport plans, vehicle drivers, and transported goods.

Furthermore, external information (e.g. weather information that is provided by an external weather service) can be valuable for the monitoring and execution of a process. However, there is no native connection to their corresponding process instances. The necessary background data for the correlation of such information can be organized in a hierarchical manner as well as it can be changing over time, which makes the correlation of external information to processes even more complex.

In order to correlate incoming events to existing process instances, current approaches use specific queries or rules that are linked to process instances as shown

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

in [HMW13]. Incoming events are then queried against the designed rules and queries in order identify relevant events, extract information from them, and provide the extracted information to the instances. Query designers can thus be forced to gain profound background knowledge of the domain before writing complex queries and procedures to ensure that all relevant event data is connected to the relevant process executions. This can complicate the inclusion of arbitrary event sources and hinder the dynamic consideration of new sources without updating existing routines and queries. The approach presented in this paper thus includes the following contributions:

Graph-based correlation. The presented approach operates on semantic graphs in order to identify process executions that need to be informed about incoming event data. The search is thereby directed from event data to relevant process instances. It uses path detection in graphs to identify relatedness of events and instances. Semantic filters furthermore improve the precision of the approach compared to native path finding.

Independence from process-specific queries. The approach does not rely on process-specific correlation queries or routines and eliminates the need for query updates and extensions when new event sources are added or background knowledge changes. It rather employs path finding in knowledge graphs to identify processes related to event data. Furthermore, events from new sources will directly be considered in the correlation process. The approach can also be used as a complement to traditional query-per-process based event correlation.

This work is structured as follows: Section 2 provides introductory information on basic terms from the fields of semantic technologies and business processes. Afterwards, Section 3 introduces use cases that further outline the need for a new approach before Section 4 details the taken approach that helps identifying the relevant process instances for incoming data events. Section 5 then presents one prototypical implementation of the approach, after which Section 6 positions the approach in the field of event correlation. Section 7 then concludes the paper.

2 Background

The approach presented in this paper bases on the concepts of process models and process instances. Based on the definition provided in [Wes12], a process model can be described as a directed graph, containing a set of nodes and a set of edges, whereas the edges represent the control flow in the model. Based on this definition of a process model, an instance of a process in [Wes12] is defined as a partially ordered set of events which contains events for all node instances of the corresponding process model. The events are ordered according to the execution constraints defined in the model.

In this work, the first outline of a semantic-based correlation approach details how to use these concepts in combination with semantic technologies in the search for process instances that correlate to the information of occurring events. Especially, the basic principles of a *semantic knowledge base* and a *knowledge graph* play a major role in the approach.

A *semantic knowledge base* SKB is a set of statements *(subject, predicate, object)*, with subject and predicate being *Uniform Resource Identifiers* (URI)[1], and the object being a string expression or a URI. The subject and the object of a statement are also called **semantic entities**. Every semantic entity is of a certain *type* that is assigned to it by the property **rdf:type**[2]. This connection is a standard property that is declared as best practice when working with semantic knowledge.

The language that is used to describe an SKB is often based on Description Logic (DL). The knowledge described by DL can be divided into a *TBox* (terminological box) and an *ABox* (assertional box), where the TBox describes the concept hierarchies while the ABox states where individuals belong in this hierarchy. Furthermore, DL allows the creation of *restrictions* and *rules* on concepts and individuals that enable the deduction of new knowledge from the concepts described in an SKB with existing reasoning tools like Pellet[3]. Beyond that, the concepts and individuals of an SKB can be depicted in a *semantic knowledge graph*.

A *semantic knowledge graph* is *directed graph* representation of a semantic knowledge base. Subjects and object in the knowledge base build the set of vertices of the graph, connected by directed edges from subject to object.

In the context of event correlation, a semantic knowledge base and its corresponding graph provide a way to model concepts like *processes*, *process instances*, and *events* in a formal, explicit, and unambiguous way. The concepts then hold a defined semantic meaning and can be shared, used, and reused between people and software agents.

As described by Lopez et al. [LdCCVG+10], semantic correlation is not limited to syntactically exactly equal values of event attributes and process context data attributes but rather allows a consideration of the semantic meaning of attributes and their relationships between each other. As stated by Lopez et al. [LdCCVG+10], semantic correlation can thus be understood as an evolution of traditional correlation techniques.

[1] The definition of a URI can be found in http://www.ietf.org/rfc/rfc2396.txt, last accessed at 01/17/2014.

[2] The definition can be found in http://www.w3.org/TR/rdf-schema/#ch_type, last accessed at 01/17/2014.

[3] Visit http://clarkparsia.com/pellet/ for more information, last accessed at 03/05/2014

3 Scenarios

This section details three scenarios from the logistics domain. These use cases are examples of current demands and real-world scenarios. For the correlation of incoming events to existing business processes they outline the need for (1) the inclusion of external knowledge that can be hierarchically structured and changing over time (Section 3.1), (2) the consideration of location data (Section 3.2), and (3) the ability to add new event sources that should be automatically considered (Section 3.3).

3.1 Parcel Tracking – Changing Hierarchical Data

Track and trace as described in [VD02, SH11] is one of the most common use cases in logistics. Consider a logistics company, shipping goods worldwide and providing status websites for parcels that allow customers to track their goods. Behind every status website there is a process execution instance connected to the specific parcel ID. The parcels are transported in containers on ships or trucks. The company's information system receives events with information on ships and trucks like their current locations.

Incoming information needs to be checked whether it is relevant for the parcel or not. Either created manually or automatically, a query must include the specific container number the parcel is transported in as well as the ship or truck the container is transported on.

Such a query can work as long as this data does not change. However, in transportation processes, containers are often loaded from ships onto trucks and vice versa. In case of unloading, all queries that are connected to the truck or ship need to updated accordingly. Otherwise, the status website may display the parcel position as if it was still on a the ship, although it is transported by a truck now.

State of the art approaches either require the query designers to have knowledge of the background information (e.g. which parcel is transported in which container on which vehicle) or that queries are build and can operate based on it. Furthermore, they can imply update procedures of queries in case of changes in that background knowledge.

3.2 Weather Information – Location Based Correlation

Consider a logistics company that transports goods with a fleet of ships and trucks worldwide. All their transportation processes are instantiated with a specific transport plan comprising of the used vehicles, their routes, drivers, and estimated arrival times at specific points.

The company's information system receives events that hold information on weather conditions all over the world. In order to identify whether the incoming weather information is valuable for the company, it needs to identify the region affected by this weather condition and evaluate if any of the company's transport plans go through this region.

Current approaches require a check of the event's region against every region of a transport plan. The complexity of such a query depends on the information provided in the event as well as in the transport plan. The better the two data structures match, the simpler the query will be. If the event and the transport plan comprise of location data of the same granularity (e.g. GPS coordinates), the query will be simpler compared to the event holding region information like 'Northern Germany' and the transport plan comprising GPS coordinates only.

3.3 Critical Events – Adding Event Sources

Consider the scenario described in Section 3.2. In order to improve a company's ability to react to critical incidents that may occur near to their transport plans, the logistics company subscribes to a new event source. It publishes events with location information in case of incidents like pirate attacks, road blockages due to riots, forest fires and the like.

For current approaches, either a new query for every process instance needs to be created or existing queries need to be adjusted in the information system dealing with region checks against the new event's region. The rules of complexity are comparable to those mentioned in Section 3.2.

4 Approach

The approach presented in this section enables automatic identification of relevant process executions for occurring events. It is based on *data connectivity* which will be explained in detail in Section 4.1. It furthermore allows the automatic consideration of hierarchical data, changes in that data, and dynamically added event sources.

Throughout the presented work, the focus lies on ABox knowledge that represents individual processes, events, and instances of other public knowledge and their relationships between each other. More specifically, the presented approach focuses on the evaluation of the *semantic knowledge graph* that can be derived from ABox data.

This limitation to a specific part of the knowledge aids in keeping the search space for the identification of processes related to incoming events to a reasonable minimum and will be further supported by the means presented in Section 4.2 and

Section 4.3. These sections present different methods for data exclusion that will be explained in detail for use cases from the logistics domain. Other domains may require different exclusions. Finally, a mechanism for identifying all instances that have a location-based interest in an occurring event will be detailed in Section 4.4.

The approach will not be based on deduction rules or subsumption matchmaking at this point of research. These techniques will play an important role in future work on this topic and need to be compared to the results achieved with the work provided here.

4.1 Data Connectivity

The scenarios shown in Section 3 state a need for the use of external data for sophisticated event correlation to business process executions. This data can also be of a hierarchical nature and change over time.

The approach uses semantic ontology data as a central information database that captures public data, event data, and process data. Public data can comprise different ships and their characteristics, event data stores event attributes like the location of events and their connections to public data like specific ships, and process data holds existing processes, their activities, current running instances, and process context data like a specific ship, truck, and driver. Figure 1 illustrates a knowledge base that stores semantic information on two logistics processes, whose execution instances are connected to context data like ships or parcels. Furthermore, the connections between parcels, ships, and containers are visible.

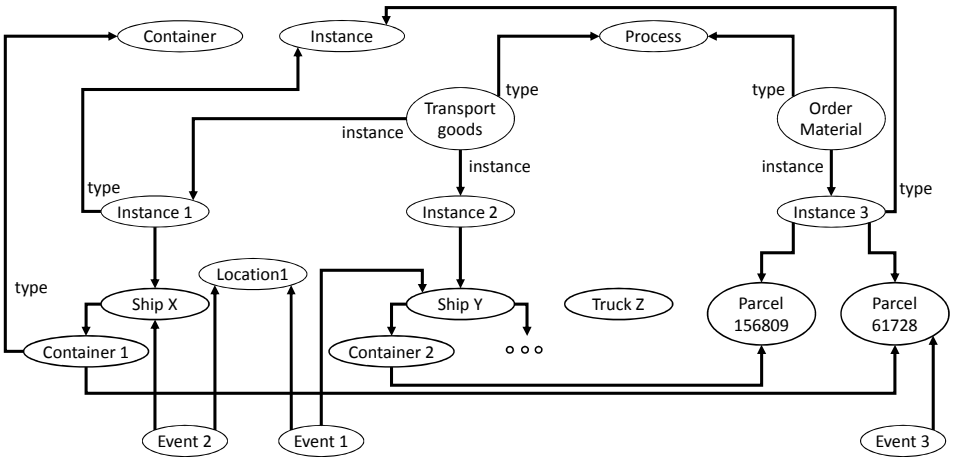

Figure 1: Exemplary semantic knowledge graph containing public data, process data, and event data related to the domain of logistics

Since semantic data is an integral part of this approach, semantic data integration and knowledge engineering as described in [Gar05, ES+07, BBR+11] are the preconditions for the approach to work. The precision of the approach presented in this work depends on up-to-date knowledge especially concerning process data and public data.

The identification of relevant process instances for an event is based on its *data connectivity* to those instances. Only those instances that have a data path to the event's information are considered as relevant. A *path* in graph theory is defined as a walk between two vertices where neither an edge nor a vertex is repeated [Fou92]. Based on this definition, a subject and an object are connected if there is a path between them in the undirected underlying graph of a knowledge graph.

The knowledge base shown in Figure 1 is enriched by an incoming event *Event1*. All process instances that are interested in the *Event1*'s data need to be identified. The correlation is based on the data connectivity between the event and the instances.

In order to identify instances that are related to the event's data, the approach:

1. Retrieves a list of current process instances from the knowledge base.
2. Checks for data connectivity between the event and every instance from the list.
3. Returns all instances that have a data connection to the event.

Regarding the knowledge base in Figure 1, the approach would return *Instance1*, *Instance2*, and *Instance3* as relevant instances that have a data connection to the event. Note that, if changes in the knowledge base occur (e.g. in Figure 1, the *Container 2* is unloaded and transported by *Truck Z*, not *Ship Y*), these updates are directly considered in this approach and therefore ensure an up-to-date result.

However, this approach always identifies all instances as interested in the event's data. Every semantic entity, e.g. the occurring event's entity itself, is of a certain type like *Instance*. These types are often hierarchically ordered in semantic ontologies and inherit from one global entity. Thus, all semantic entities in a knowledge base are connected. Therefore, a connection to all executions can be found if the searchable graph space is not limited.

4.2 Allowed Direction Changes

The problem of *over identifying* (i.e. identifying too many instances as related to the event's data) with the basic data connectivity approach can be tackled by a limitation of the search space. This can be achieved by restricting the allowed number of direction changes on the path. This is a simple method preventing the escalation of the search for instances to all parts of the graph.

In a semantic knowledge graph SKG $= (V, E)$, a direction change between two edges (u, v) and (w, x) can be found if $u \neq x$ *and* $v \neq w$, with $(u, v), (w, x) \in E$ and $u, v, w, x \in V$. The number of direction changes on a path between a subject and an object can thus be counted, with the minimum number of direction changes possible being decisive.

Given the example knowledge base shown in Figure 1, a limitation of the approach to only search for connections that have no direction changes on the path would yield no instance to be interested. A limitation to a maximum of one direction change would lead to the identification of *Instance2* and *Instance3* as relevant. *Instance1* would not be identified as related to the event's data. All allowed numbers greater than one would yield the result achieved by the basic data connectivity approach. Thus, this rather small knowledge base already highlights that it is of great importance how the number of changes is limited. It has not yet been evaluated if it is even possible to always find a suitable restriction for the whole knowledge base.

This task becomes even more complex with growing knowledge base sizes and complexities. Furthermore, this approach is very dependent on the modelling style of the knowledge base and the edge directions. Besides, although this restriction works for some parts of the graph, it might not be suitable for others that for example show a higher rate of direction changes between the data entities.

Beyond all, it does not take the semantics of the edges into account. Some edges are of more interest than others when it comes to searching for relevant instances. How this can be accomplished is shown in Section 4.3.

4.3 Graph Cutting

An alternative approach to limiting the search space of the data connectivity approach is the selective cutting of the knowledge graph. With this concept, the search is intentionally restricted to specific areas of the knowledge base. In the following, four exclusions will be detailed. Regarding the knowledge graph displayed in Figure 1, these restrictions lead to a search space as shown in Figure 2 and results equal to those of a limitation to one allowed direction change on the path.

4.3.1 Meta Level Exclusion

All semantic entities are of a specific *type*. By restricting the search space to exclude meta level entities, reaching other entities of the same *type* (e.g. all other instances or ships) can be avoided. If information is given on one individual (e.g. a specific ship), the other individuals of that type are not of interest just because they are of the same type. They may be of interest as well, but not through this connection. For the approach this implies: when looking at a semantic entity, ignore the outgoing edges named *rdf:type* when looking for a data connection to the event. In Figure 2,

all edges named *type* are therefore marked as *irrelevant* for the search of a data connection.

4.3.2 Process Exclusion

Business processes are connected to their instances. When searching for relevant instances, finding one instance directly leads to the discovery of all sibling instances due to their edge to the parent process. This sibling edge does not qualify an instance for being interested in an event's data, therefore this edge is excluded. For the approach this implies: when looking at an instance, ignore the edge that leads to the parent process[4] when looking for a data connection to the event. In Figure 2, all edges named *instance* are therefore marked as *irrelevant* for the search of a data connection.

4.3.3 Instance Context Exclusion

Instances are connected to context data like a vehicle, a driver, cities, loaded goods and more. When the edge to one entity of this data leads to the identification of an instance as interested in the event, all other context data will be used as a path to search for other instances. This sibling context data edge however does not qualify for the identification of relatedness to the event's data. For the approach this implies: when reaching an instance, ignore the edge that points to the instance's context data[4] when looking for a data connection to the event. In Figure 2, the edge of *Instance3* to *Parcel61728* is therefore marked as *irrelevant* for the search of a data connection.

4.3.4 Former Events Exclusion

The knowledge base is enriched by events and their information over time. These events point to other entities in the knowledge base and therefore insert links between entities. These links however do not qualify for the identification of relevant instances and are thus excluded. For the approach this implies: when looking at a semantic entity, ignore edges that link to former events[4] when looking for a data connection to the current event. In Figure 2, all edges to data from events other than the current *Event1* are therefore marked as *irrelevant* for the search of a data connection.

4.4 Location

As shown in Section 3, the location of an event can be an important factor in the search for relevant instances. In previous work [MRSB+14], the importance

[4]The name of this edge depends on the used ontology.

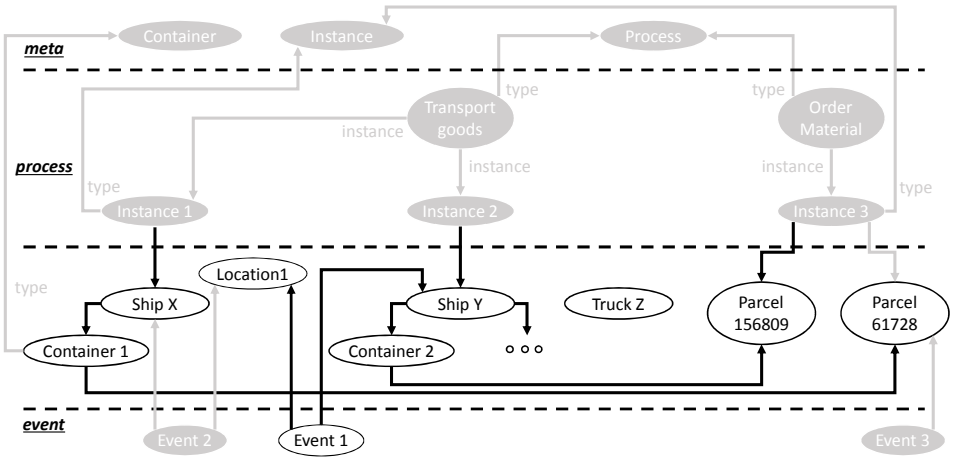

Figure 2: Data model used for the identification of interested process instances. It is restricted to public data only, excluding the meta, process and event level from the search in order to prevent the *over identifying* of the general data connectivity approach. Grey connections are not actively used for the search.

of locations for event processing in the domain of logistics has been shown. The technique detailed in that work will be reused in the automatic search for relevant instances. Thus, the approach is extended by a domain-specific filter mechanism that highlights the flexibility and extensibility of the overall approach.

In particular, process instances can be associated with transport plans as shown in use cases in the work done by Herzberg et. al [HMW13]. Among other things, these transport plans contain GPS coordinates that describe the route the associated vehicle takes. With the help of this data as well as public geographical data, a *nearby function* can determine, whether the location of an event is near to a transport plan.

If any instances are connected to that transport plan, they automatically become relevant for the occurring event and its data although they may not be connected to the event's data by a path through the knowledge graph.

5 Architecture/Implementation

The presented approach can be implemented in various ways. In a proof-of-concept prototype, it has been realized in a single SPARQL query that is executed on various datasets via the Apache JENA framework[5] in Java. The query language *SPARQL*[6] can be used for querying knowledge bases that are written in RDF. It is

[5]To be found at http://jena.apache.org/, last accessed at 01/13/2014.
[6]Recursive acronym for *SPARQL Protocol And RDF Query Language*

a graph-based query language that allows to retrieve data from and manipulate the data of an SKB.

The query in Listing 1 returns (line 1) all current process instances (line 2) that can be reached by at least one path starting from the event. The used method of *property paths*[7] (lines 3 to 10) translates the *graph cutting* into the SPARQL query. In the search for a path from event to instance, the defined edges are excluded by wrapping the disjunction of all irrelevant predicates in a negation.

The resulting set is merged (line 11) with those instances that have a location-based interest in the event due to their transport routes (lines 12 to 16). For the latter, it uses the *nearby function* defined in [MRSB+14].

Listing 1: The basic search algorithm for unlimited direction changes and the incoming event *Event1* in a SPARQL query. Prefix definitions are omitted for brevity.

```
1  SELECT DISTINCT ?instance WHERE {
2    ?instance rdf:type/rdfs:subClassOf* bp:Instance .
3    {
4      event:Event1 !(rdf:type|
5      bp:hasInstance|
6      bp:hasAttribute|
7      ^bp:hasInstance|
8      ^event:hasEventData|
9      ^event:hasEventInfo) ?instance .
10   }
11   UNION
12   {
13     ?instance bp:hasAttribute ?transportPlan .
14     ?transportPlan a dbo:transportation_route .
15     FILTER ( %nearby(event:Event1, ?transportPlan, 30) )
16   }
17 }
```

6 Related Work

Event correlation has been a prominent research topic for several years.
Lopez et al. [LdCCVG+10] examine a variety of methods and implementations for event correlation and compare them regarding their strengths, weaknesses, and possible fields of use. They also detail how semantic event correlation can overcome some of the limitations of basic approaches. Based on the principles explained in their work, the approach presented here uses semantic technologies to create links between events, external knowledge, and process data. However, the approach rather focuses on the correlation of events to specific process instances than to other events.

In [ZSP12], Zhou et al. examine the use of semantic technologies in complex event processing. They detail an architecture with a state of the art CEP engine extended

[7] To be found at http://www.w3.org/TR/sparql11-property-paths/, last accessed at 01/17/2014.

by semantic event queries that enable the querying of past, present and future event data. Teymourian et al. [TP10] outline an architecture that uses a rule-based engine to allow comparable queries to those of Zhou et al. The approach presented in this paper focuses on the use of already correlated and aggregated events and the detection of process instances that are related to the insights provided by such complex events. It uses similar technologies to those presented by Zhou et al. and Teymourian et al.

Rozsnyai et al. [RSL11] detail an algorithm that allows for the detection of correlation rules from an arbitrary list of data sources that provide information stored in inhomogeneous data structures. They process the incoming events, establish relations between them and are thus able to build aggregate groups of events that can be used in further analyses of the executed processes. The approach presented in this paper also strives to detect correlations in an automatic manner, but rather focusing on connecting events to specific and well-defined process instances and not to each other. Rozsnyai et al. try to (semi-)automatically build correlations from the events' data without pre-defined rules, an approach that aligns with the mechanism presented in this paper.

Herzberg et al. [HMW13] present, in a straight-forward approach, how data of manual process executions can be matched to relevant process instances using basic value-based methods. The presented approach in this paper goes beyond the value-based matching approach and operates on linked data that allows for a broader search for relevant instances that is not limited to the equality of defined variables but can take semantics of values into account.

7 Conclusion

The monitoring of business processes relies on sophisticated event handling mechanisms. The inclusion of external knowledge in the event correlation process can improve these capabilities. Adding new event sources dynamically in the process of correlation is a common task that needs to integrate seamlessly in the whole event handling procedure and should not require complex updates and reconfigurations of the event handling system. The approach introduced in this paper provides a semantic based solution that moves the correlation task away from the instances towards the event itself, thus enabling an always up-to-date correlation that is more flexible than current approaches. It can be used as a standalone method of identifying related instances to incoming event data or as a complement to an existing correlation infrastructure.

References

[BBR+11] Zohra Bellahsene, Angela Bonifati, Erhard Rahm, et al. *Schema matching and mapping*, volume 20. Springer, 2011.

[ES+07] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[Fou92] Leslie R Foulds. *Graph theory applications*. Springer, 1992.

[Gar05] Stephen P Gardner. Ontologies and semantic data integration. *Drug discovery today*, 10(14):1001–1007, 2005.

[HMW13] Nico Herzberg, Andreas Meyer, and Mathias Weske. An Event Processing Platform for Business Process Management. In *Proceedings of the 17th IEEE International Enterprise Distributed Object Computing Conference*, pages 107–116, 2013.

[LdCCVG+10] Sergio Lopez, Maria del Carmen Calle Villanueva, Emitza Guzman, Tobias Röhm, Benoit Gaudin, and Newres Al Haider. State-of-the-art of event correlation and event processing, 2010.

[MRSB+14] Tobias Metzke, Andreas Rogge-Solti, Anne Baumgrass, Jan Mendling, and Mathias Weske. Enabling Semantic Complex Event Processing in the Domain of Logistics. In *ICSOC 2013 Workshops*, 2014.

[RSL11] Szabolcs Rozsnyai, Aleksander Slominski, and Geetika T Lakshmanan. Discovering event correlation rules for semi-structured business processes. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, pages 75–86. ACM, 2011.

[SH11] A Shamsuzzoha and Petri T Helo. Real-time tracking and tracing system: Potentials for the logistics network. In *Proceedings of the 2011 international conference on industrial engineering and operations management*, pages 22–24, 2011.

[TP10] Kia Teymourian and Adrian Paschke. Enabling knowledge-based complex event processing. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 37:1–37:7. ACM, 2010.

[VD02] Kees-Jan Van Dorp. Tracking and tracing: a structure for development and contemporary practices. *Logistics Information Management*, 15(1):24–33, 2002.

[Wes12] Mathias Weske. *Business process management: concepts, languages, architectures*. Springer, 2012.

[ZSP12] Qunzhi Zhou, Yogesh Simmhan, and Viktor Prasanna. SCEPter: Semantic complex event processing over end-to-end data flows. Technical report, Technical Report 12-926, Computer Science Department, University of Southern California, 2012.

Leveraging Model-Driven Monitoring for Event-Driven Business Process Control

Falko Koetter

University of Stuttgart IAT
Allmandring 35
Stuttgart, Germany
falko.koetter@iao.fraunhofer.de

Monika Kochanowski, Maximilien Kintz

Fraunhofer IAO
Nobelstraße 12
Stuttgart, Germany
firstname.lastname@iao.fraunhofer.de

Abstract: Event-driven business process management can help businesses to monitor and guarantee correct and efficient execution of their business processes. Two examples of this are using events to monitor processes and control execution. However, many companies lack mature, executable process models and rely on legacy software systems to execute processes. Creating a customized event infrastructure in such an environment is a cumbersome task. In previous work we detailed aPro, an architecture for model-driven design and implementation of business process monitoring. This paper describes how to use real-time monitoring data to control business process execution in real-time, extending the model-driven approach.
Keywords: Business Process Monitoring, Complex Event Processing, Compliance, Event-driven Business Process

1 Introduction

Today many companies have a need to improve the quality of their business processes. This need arises from volatile market conditions, increasing cost pressure, higher rate of innovations as well as regulatory compliance requirements. Efforts cover all parts of the business process lifecycle [Wes12], including process modeling and management (e.g. CMMI [GG03]), workflow execution, monitoring, and adaptive case management.

However, many companies have a low level of business process management maturity [RdBH04, PCBV11], lacking the capability to execute workflows and optimize their processes. Existing approaches for business process monitoring and execution control often build on top of executable process models running in a process engine[WSL09, BTPT06, BG05]. However, in our work with the industry, we found this to be the exception rather than the norm. Economic and organizational constraints prevent a switch of technology. Existing process models are often not linked to process execution in any way. They only serve documentation purposes. This can lead to discrepancies between process model and actual process, either because the model is out-of date or because the model represents the ideal rather than the actual state of affairs. Additionally, IT resources and budgets are often constrained. IT departments are busy managing daily operations and implementing

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

new products and requirements. Structural improvements without immediate gains are often deferred.

In previous work we developed the aPro architecture to develop and introduce business process monitoring in these heterogeneous, distributed environments [KK12, KK13]. In a model-driven approach, a monitoring infrastructure is created in three steps. In the first step a business process model is augmented by a platform-independent goal model, describing where to measure data as well as the Key Performance Indicators (KPIs) and goals of the process. This goal model is then used to automatically configure and deploy the aPro architecture in a web container[KWR12]. This includes monitoring data measurement by providing web services and interchange formats[KK12], processing via Complex Event Processing (CEP) rules[KK13] and display by configuring a dashboard[Kin12]. In the third step, existing systems are integrated using platform-specific monitoring stubs, making only minimal changes to the executing system necessary.

We applied this monitoring infrastructure in multiple use cases, including monitoring of process metrics, business metrics as well as compliance monitoring. In comparison to business monitoring, monitoring of compliance to rules and regulations presents a unique set of challenges. While the violation of business goals may be handled at the discretion of a company, violations in compliance should not occur at all and need to be handled immediately to avoid legal problems. As some real-world processes are executed in short timeframes and in high volumes, manual counteractions to goal violations may not be practical. Even if personnel are notified immediately, they may not act fast enough. Thus, an automated execution control is needed to initiate countermeasures to goal violations immediately after they occur.

In this work we will show how real-time monitoring data generated by aPro can be used to control business process execution in heterogeneous environments, providing the following contributions: We show how existing monitoring events and rules can be used for Business process control. WE extend our model-driven approach to encompass the modeling and creation of business control elements. We show how model-driven business process execution control in heterogeneous, distributed systems can be achieved. We implemented these concepts in a prototype.

2 Related Work

In this section we will present related work regarding event-driven process management and execution control. We have outlined related work in the area of process monitoring in previous work[KWR12, KK] and will focus this section on process control. Related work in the field of process adaptation and control focuses on executable process models running within a process engine. While in such homogenous environments these approaches provide a high degree of automation, they provide little assistance for implementing and configuring different process monitoring and execution environments.

[JMM12] propose an event-based approach, gathering events from a process engine, processing them in a CEP engine and providing results to a dashboard and the CEP

engine, thus creating a feedback loop used for process control. It was implemented using standardized formats and off-the-shelf-components. However, considerable effort was spent integrating components, converting data and defining rules and configuration. In comparison, a model-driven approach as presented in this work automates many of these tasks and is not limited to a process engine.

In [TZ11] running BPEL process instances are adapted using adaptation patterns. This approach is not limited to a specific process engine, as only monitoring and adaptation operations need to be implemented. It is used to change process models on the fly, for example by inserting a process fragment, to migrate processes during runtime, with monitoring information serving to adapt each specific process instance. Adaptations are not based on process goals or KPIs as in this work. While this approach provides more flexibility compared to previous approaches allowing only substitution of activities (e.g. [MRD08]), it is still limited to executable BPEL process models. In comparison, the model-driven approach in this work supports any execution environment that can be integrated via stubs.

[UGSC11] presents multiple approaches to control process execution in distributed systems. An *integration rule* allows measuring at the start and end of activities. *Assurance points* define a checkpoint within the process and allow monitoring and condition checking. *Exception rules* are used to implement countermeasures for exceptions. Finally, *invariant rules* evaluate an invariant throughout process execution and allow countermeasures as soon as it is violated. In comparison to this work, a way to define the different kinds of rules in a unified way and connect them to a graphical process model is missing. Process execution by decentral agents is currently supported, but process engines are not. In comparison, this work supports both, individually or in combination.

[MZD09] describes an approach to detect compliance violations during execution of business processes in a service-oriented architecture. Web service calls are handled as events, which are correlated to event trails and business activities, similarly to correlating events from measuring points in this approach. From these trails CEP rules are created to detect compliance violations. Compliance controls are assumed to be in place and can be invoked if a violation is detected. In comparison, this approach generates event rules directly from business (compliance) goals and provides assistance in implementing compliance controls. However, in a homogenous environment [MZD09] provides a higher degree of automation in the gathering of monitoring events. Similarly, the MASTER project[LPF+08] defines a compliance architecture reading events from services in a SOA, monitoring and assessing them and using the result for compliance enforcement, gathering monitoring data and propagating control actions similarly to this work. For this purpose, the MASTER project defines a high-level language for monitoring data and interfaces.

3 Concept

3.1 Conceptual Overview

This section gives a short overview of the aPro architecture for process optimization and related concepts [KK12, KWR12, KK13, KK].

Figure 1 shows the aPro architecture with the necessary components for monitoring (elements in grey show extensions for control). A BPMN [Obj09] process model is created in the modeling step and stored in a process repository. Depending on the implementation of the business process, the model is either directly executed by a process engine or the process model represents the process execution by one or multiple application systems.

Regardless how the process is executed, monitoring data needs to be gathered by so-called monitoring stubs. These monitoring stubs are used to integrate different kind of application systems, for example as a part of program code or a script. Whenever monitoring data is measured by a monitoring stub, it is sent to a monitoring web service, which in turn creates an event for monitoring data processing. These events are processed by a CEP engine, which aggregates measurements, calculates process goals and KPIs. Monitoring data is presented on a dashboard [Kin12].

To use this architecture for a specific process, configuration files for each component are needed. Using a model-driven approach, these do not need to be created manually, but can be created from a platform-independent model. A so-called goal model describing the monitoring is modeled alongside the process model in a graphical notation. In Figure 2 the modeling elements are shown. A measuring point may be attached to a BPMN element and indicates a measurement to be taken whenever this element is reached during execution. Parameters within the measuring point describe the data to be measured. Based on measured parameters, KPIs and goals can be calculated. A KPI is an important value of a process. A goal is a Boolean condition imposed on a parameter or KPI. If it evaluates to false, the goal is violated. A timing goal imposes a maximum time interval between two measurements of the same process instance. If the second measurement (and its associated activity) is not performed within a specified time after the first, the goal is violated.

After modeling, the process and goal model are stored in a so-called ProGoalML file, an interchange format used as an input to generate component specific configuration files. From this, monitoring stubs, data formats, event processing rules and dashboard layout are created automatically. For the end user, the monitoring infrastructure can be deployed directly from the modeling tool. However, depending on the systems running the process, monitoring stubs need to be deployed manually, though (semi-)automatic assistance is possible. In earlier prototypes, monitoring stubs can be created as a web forms, e-mail forms, JAVA classes, and batch scripts.

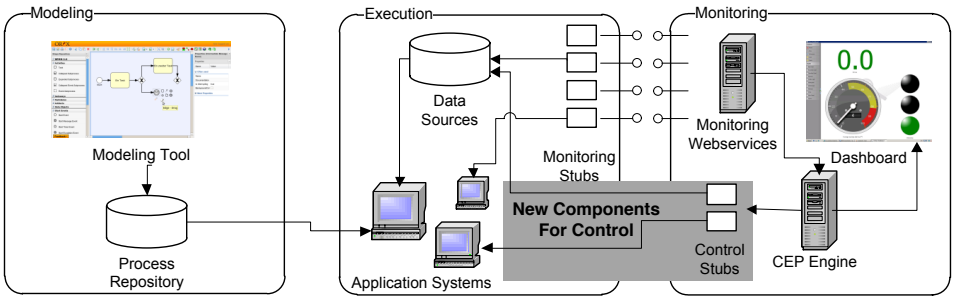

Figure 1: aPro architecture extended for process execution control, new elements marked grey

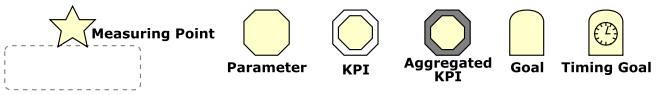

Figure 2: ProGoalML modeling elements

3.2 Running Example

Figure 3 top shows a simplified real-life insurance claim management process consisting of four steps. In the step *Receive claim* an insurance claim is received from a customer after an event of damage. This claim is usually sent in paper form and contains a description of the claim as well as a damage assessment from a local claim assessor which may or may not be associated with the insurance company. During the receive step, the data is parsed in a structured form and any containing images are extracted. This data is used in the next step *Process claim* to examine the claim based on multiple factors, for example if the damage is covered by the contract, if the damage description is plausible, if the claimed amount is realistic, etc. In the third step *Decide claim* a decision about the claim is made. A claim can be accepted, accepted with deductions or rejected. In the fourth step *Send claim* a letter with the result is sent to the customer. This process is executed in a heterogeneous environment involving a claim management system used by insurance personnel, a separate image extraction system, a service provider for text extraction and an expert system for claim assessment. As these systems handle a large volume of about 300 claims a day, of which approximately a third fulfills one or more special cases, a high degree of automation is desired.

To monitor the system with aPro, each activity has an associated measuring point. The step *Receive claim* is measured after the claim is received and the image extraction system is called. It measures the ID of the claim, the timestamp the claim is received and the amount of images that have been extracted from the document. An aggregated KPI *avg_img_count* is defined measuring the average

count of extracted images over the last. The goal *avg_img_above_zero* is fulfilled if this average is above 0. If the goal is not fulfilled, this indicates a problem with the image extraction system. An alert is generated and an administrator needs to investigate.

The second step *Process claim* is measured at the claim management system. Here, the *assessor_address* is measured, indicating who has assessed the claim. Additionally, a lookup of this assessor is performed within an internal database. If the assessor is found, his or her ID in the database is given as *assessor_db_id*. In case the assessor is unknown, this value is null and the goal *known_claim_assessor* is violated. The insurance company plans to keep its database of local assessors up-to-date, as they need to commission assessors themselves. This goal serves to identify new opportunities to add to the database.

The third step *Decide claim* is measured at the expert system after a decision about claim acceptance has been made. Here the *savings* generated in claim processing are measured. The value *details_missing* indicates if all details were available. While a claim can be valid with missing details, additional details may enable additional checks. For example the zip code of the claimant may not be extracted properly from the print document, limiting local comparisons of claimed damages. As the amount of claims cannot be handled manually, only select claims may be examined in detail. The goal *decision_viable* is used to select these claims. It fails if both details are missing and no savings have been generated.

The last step *Send claim* is measured at the claim management system after the claim answer letter is sent and represents the end of the process. The *timestamp* of the process end is measured.

The aggregated KPI *open_claims* calculates the current number of open claims by counting the measurements at *Receive claim* and subtracting the measurements at *Send claim*. The amount of open claims is an indicator of the general load of the system, as it rises with incoming claims and process duration.

The timing goal *notification_within_14_days* is related to a compliance requirement. The German Insurance Association (GDV) has issued a code of conduct regarding data privacy, which is to be followed by the claim management process [Ger12]. Requirement 5-8 of the code stipulates that a customer who has provided personal data has to be asked to agree with the use of this data for marketing purposes. This needs to occur with the next message to the customer or within a short time span (in this example 14 days). If the claim is answered within 14 days, the agreement can be asked within the letter. However, if claim processing takes longer, the goal is violated. This may happen under high load or in case of complex claims which are checked manually. After the goal is violated a separate letter asking for agreement needs to be sent to the customer manually.

While process monitoring helps to identify problems within the process by sending alerts, counteractions need to be performed manually. To lessen manual work and increase reliability, the monitoring events shall be used for process execution control. The following controls shall be implemented:

- Whenever the goal *avg_img_above_zero* is violated, the image extraction service may have encountered a problem. In this case the service shall be restarted automatically by running a batch script.
- If an unknown claim assessor is detected (goal *known_claim_assessor* is violated), the address of the claim assessor shall be sent to a different service, which automatically mails a questionnaire to the claim assessor, asking him to provide information about his or her services.
- If the goal *decision_viable* is violated for a claim, the claim shall be assigned to an expert within the claim management system.
- Depending on the amount of *open_claims* the length of time until automatic finalization of the claim shall be regulated so experts are not overwhelmed.
- If the goal *notification_within_14_days* is violated, a letter asking for agreement shall be created and sent automatically by a separate service.

In the following sections we will define the requirements to implement these control measures as well as the necessary modifications in aPro.

3.3 Requirements

In this section we will describe the requirements for process execution control both from a business and compliance perspective.

R1 *Goal violations create an event, which may be processed within the process.* From a compliance perspective, process control aims to counteract detected compliance violations. In aPro, compliance requirements are modeled as goals. Thus, goal violations need to trigger counteractions.

R2 *Whenever a KPI is calculated, an event is created, which may be processed within the process.* While the same applies to the business perspective, in a business use case in addition to the goals KPIs may be relevant for execution control. One example are the *open_claims* in the running example. While target values may be checked with a goal, a balance depending on the concrete value needs to be kept.

R3 *Events are created for both aggregated and nonaggregated goals and KPIs.* Goals and KPIs may be calculated for a single process instance or aggregated by time and length, which both may lead to control actions.

R4 *Events and control actions are modeled alongside the process and goal model.* As aPro uses a model-driven approach, process execution control needs to be part of the model.

R5 *All parts of the aPro architecture relevant for process execution control are configured automatically.* Events and control actions need to be taken into account when configuring the aPro architecture. This includes monitoring event creation and functionality for triggering control actions.

R6 *All modeling extensions need to work with executable process models.* As described in Section 1, aPro supports heterogeneous execution environments. This

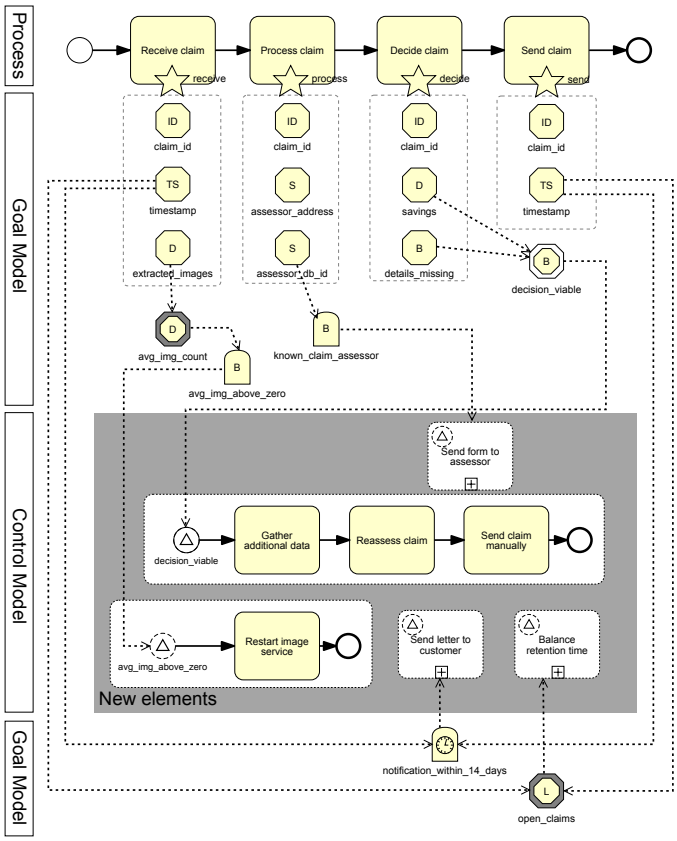

Figure 3: Claim management process (top) with goal model for monitoring the process (middle and very bottom, consisting of measuring points, KPIs, aggregated KPIs, and goals) and control model (marked grey, new elements introduced in this work consisting of events and event subprocesses for controlling the process)

means an event may trigger different kinds of control actions, depending on the system(s) executing the process. In case of a process engine, this may be achieved by simply catching a BPMN event [Sil09].

R7 *aPro assists in the creation and integration of control actions.* Without a process engine, control actions include performing an additional activity, calling a web service, executing a script or changing a configuration parameter. As these control actions need to be defined in a platform-specific way, they cannot be created automatically. Similar to the monitoring stubs for gathering measurements, control actions need to be integrated with the executing system. Thus, instead of full automation, semi-automatic assistance for control point creation needs to be provided.

3.4 Modeling extensions

To fulfill requirement **R4**, we need to extend the ProGoalML notation to include the definition of events and control actions. BPMN already contains modeling elements for event handling. Events can be thrown and caught to control process flow, escalate process instances and handle exceptions [Sil09]. To keep compatibility with executable process models (requirement **R6**), the existing event modeling capabilities should be used. To implement a control action, the action itself as well as the triggering event needs to be modeled. The event is thrown if a KPI is measured or a goal is violated. It is then caught to trigger the control action.

As monitoring data is gathered and processed in an event-based fashion using CEP, no separate modeling of events is necessary. From a conceptual point of view, control actions are not triggered by events, but by goals and KPIs. Thus, instead of explicitly modeling the throwing event, implicitly an event is assumed to be thrown at each KPI and goal, whenever they are calculated. BPMN 2.0 has multiple event types which are used for different purposes. To decide which event is implicitly thrown by KPIs and goals, we investigated the usage of each event type. Event types differ in the scenarios where they may be thrown and caught. Message events are used to communicate between pools. Error events handle errors in subprocesses and cannot be caught as a start event or at a gateway. Escalation events can only be caught at the border of the subprocess they're thrown in. If maximum flexibility in event handling is to be provided, these event types are unsuitable. BPMN 1.1 introduced the Signal event type to provide more flexible event communication [Sil09]. Signal events may be used to listen to external events within a process model, which is suitable for goal and KPI events generated by the CEP engine. Thus, events are to be modeled as signal events named by the KPI or goal they refer to. Using catching signal events, control actions can be modeled with standard BPMN elements, e.g. as an event subprocess or as part of an event gateway. An event subprocess is a subprocess which is started if its start event is thrown. An interrupting event subprocess stops its parent process, while a non-interrupting event subprocess runs in parallel without stopping its parent process [Sil09].

Figure 3 shows the running example extended by implementing the specified controls. Control actions are modeled as event subprocesses. For example, if the goal *avg_image_ above_zero* is violated, a signal event named *avg_img_above_zero* is thrown. It is caught in an event subprocess, as indicated by an association. In this subprocess, the image service is restarted without interrupting the main process. Similarly, three other control actions are modeled, shown as collapsed event subprocesses. If *decision_viable* is false, the event triggers an interrupting event subprocess in which an expert finishes the process manually.

Associations are used to indicate event flow within the process. During creation of the ProGoalML file these as well as the goal model are removed from the process model to create a plain BPMN model. Additionally to the process and goal model, a control model is added to the ProGoalML file indicating which goals and KPIs

may trigger control actions.

3.5 Model transformation

Using the control model, the aPro architecture has to be configured (requirement **R5**). Figure 1 shows the extended aPro architecture, new control elements marked in grey. Requirements **R1**, **R2** and **R3** are fulfilled by the CEP engine which already generates relevant events for monitoring purposes [KK]. Thus, no additional CEP rules are necessary. Below an automatically generated example CEP rule for the timing goal *notification_within_14_days*. It generates an alert event whenever a measurement of *receive* is not followed by a matching measurement of *send* within 14 days.

```
INSERT INTO ‘ALERTnotification_within_14_days‘
SELECT receive.claim_id AS claim_id FROM PATTERN [EVERY
       receive = INreceive -> (timer:interval(14 days)
       AND NOT send = INsend(claim_id = receive.claim_id))]
```

As control actions are specific to the application systems, an integrating component is necessary. Similarly to the monitoring stubs, which send unified monitoring data from a specific system, we introduce so-called *control stubs*. *Control stubs* are specific to a control action and listen to the relevant event from the CEP engine. If an event is received, the control stub performs an implementation-specific task (e.g. calling a script or sending the event to a process engine). To configure this, a listener is registered for each relevant event at the CEP engine, which is called whenever such an event occurs.

3.6 Control stubs

During deployment of the aPro architecture, the specific control stubs for each control action need to be available as Java classes. These are implemented by extending a control stub abstract class, which in turn is a listener to the CEP engine. Only a method handling the event needs to be implemented. For example regarding the timing goal *notification_within_14_days*, a control stub listening to *ALERTnotification_within_14_days* events is implemented which calls a service generating the letter whenever an alert occurs. The process instance is identified by the *claim_id* within the event.

To map control actions to control stubs, an XML format called a *control mapping* is used. For each control action it contains the implementing control stubs as well as configuration parameters. This allows custom as well as predefined control stubs. Class files need to be provided for each custom control stub. A default control mapping is created automatically, mapping each control action to an empty control stub.

4 Prototype and Evaluation

We have extended the existing prototype to model a control model as well as store it in a ProGoalML file as described in 3.4. Figure 3 has been created using the

Oryx-based [1] modeling component of the prototype. During deployment, the control mapping as well as all control stubs are added to a Java web archive (war) file containing the monitoring infrastructure.

The open source CEP engine Esper [2] is used to generate events. We adapted existing aPro functionality for sending e-mail alerts to implement control stubs. The abstract control stub implements the Esper *UpdateListener* interface. During startup of Esper, the *control mapping* XML file is read. Each control stub is instantiated as a Java object with the specified parameters and registered as a listener to the specified event at the CEP engine.

We implemented predefined control stubs allowing for invoking an executable (e.g. a batch script), calling a URL and sending an event to the process engine Activiti [3]. These can be selected and configured in the *control mapping* without writing custom source code. Therefore, requirement **R7** is fulfilled. However, we encountered an issue regarding events from aggregated goals and KPIs in Activiti. As no process instance ID is known, the only option is to broadcast these events to all process instances. This may lead to undesired behavior, as control actions may be executed multiple times. Thus, requirement **R6** is only fulfilled for nonaggregated goals and KPIs.

We evaluated the prototype using the example process implemented with a test driver as well as deployed within Activiti. Using the test driver, control actions were invoked as desired. However, using Activiti we learned that event subprocesses are currently only partially supported. We were able to test the general mechanism of sending signals to Activiti but will need to further test when development of Activiti has progressed.

In future work we plan to evaluate the prototype further using the real-life claim management process. While this process is currently monitored non-intrusively by aPro, implementing controls is a critical alteration and requires further testing.

5 Conclusion and Outlook

In this paper we described a model-driven approach for business process monitoring and execution control in heterogeneous, distributed environments. The existing monitoring infrastructure is used to provide events which trigger control actions. We described how control stubs can be used to define these control actions with minimal effort. We implemented these concepts in a prototype and evaluated it with an example process containing business and compliance controls.

Monitoring and controlling compliance at runtime is only one aspect of guaranteeing compliance of a business process. In future work in the C.om.B. project[4] we plan to unify compliance management in a generic descriptor encompassing process modeling, deployment, runtime and control [KKR+13].

[1] http://bpt.hpi.uni-potsdam.de/Oryx/WebHome
[2] http://esper.codehaus.org/
[3] http://www.activiti.org/userguide/#bpmnSignalEventDefinition
[4] http://www.iaas.uni-stuttgart.de/forschung/projects/COMB/

In future work we plan to extend the prototype and evaluate it in a real-life environment. Planned extensions include full process engine support and easier configuration management. While deploying aPro is simple, advanced options like e-mail alerting, passwords, visualization preferences and control mappings need to be specified with each deployment. We plan to unify these options in a deployment descriptor, further increasing ease-of-use. We also plan to extend the generation of monitoring dashboards to the generation of control panels, so as to provide a GUI for manual controlling tasks, and for overseeing automatic ones.

Acknowledgement

The work published in this article was partially funded by the Co.M.B. project of the Deutsche Forschungsgemeinschaft (DFG) under the promotional reference SP 448/27-1.

References

[BG05] Luciano Baresi and Sam Guinea. Towards dynamic monitoring of ws-bpel processes. In *Service-Oriented Computing - ICSOC 2005*, volume 3826 of *Lecture Notes in Computer Science*, pages 269–282. Springer Berlin / Heidelberg, 2005.

[BTPT06] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. In *International Conference on Web Services ICWS '06.*, pages 63 –71, 2006.

[Ger12] German Insurance Association (GDV). Verhaltensregeln fuer den Umgang mit personenbezogenen Daten durch die deutsche Versicherungswirtschaft, 2012.

[GG03] D. Goldenson and D.L. Gibson. Demonstrating the impact and benefits of CMMI: an update and preliminary results. Technical report, 2003.

[JMM12] Christian Janiesch, Martin Matzner, and Oliver Mueller. Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Process Management Journal*, 18(4):625–643, 2012.

[Kin12] Maximilien Kintz. A semantic dashboard description language for a process-oriented dashboard design methodology. In *Proceedings of MODIQUITOUS 2012*, Copenhagen, Denmark, 2012.

[KK] Falko Koetter and Monika Kochanowski. A Model-Driven Approach for Event-Based Business Process Monitoring. *Information Systems and e-Business Management (to appear).*

[KK12] Falko Koetter and Monika Kochanowski. Goal-oriented model-driven business process monitoring using progoalml. In *BIS*, volume 117 of *Lecture Notes in Business Information Processing*, pages 72–83. Springer, 2012.

[KK13] Falko Koetter and Monika Kochanowski. A model-driven approach for event-based business process monitoring. In *Business Process Management Workshops SE - 41*, volume 132 of *Lecture Notes in Business Information Processing*, pages 378–389. Springer, 2013.

[KKR+13] Falko Koetter, Monika Kochanowski, Thomas Renner, Christoph Fehling, and Frank Leymann. Unifying Compliance Management in Adaptive Environments through Variability Descriptors (Short Paper). In *Proceed. of SOCA 2013*, 2013.

[KWR12] Falko Koetter, Anette Weisbecker, and Thomas Renner. Business process optimization in cross-company service networks - architecture and maturity model. In *Proceedings of the 2012 Annual SRII Global Conference*, 2012.

[LPF+08] Volkmar Lotz, Emmanuel Pigout, Peter M Fischer, Donald Kossmann, Fabio Massacci, and Alexander Pretschner. Towards systematic achievement of compliance in service-oriented architectures: The MASTER approach. *Wirtschaftsinformatik*, 50(5):383–391, 2008.

[MRD08] Oliver Moser, Florian Rosenberg, and Schahram Dustdar. VieDAME-flexible and robust BPEL processes through monitoring and adaptation. In *Companion of the 30th international conference on Software engineering*, pages 917–918. ACM, 2008.

[MZD09] E. Mulo, U. Zdun, and S. Dustdar. Monitoring web service event trails for business compliance. In *Service-Oriented Computing and Applications*, pages 1–8, Jan 2009.

[Obj09] Object Management Group (OMG). Business Process Model and Notation (BPMN) Version 2.0, 2009.

[PCBV11] Susanne Patig, Vanessa Casanova-Brito, and Barbara Vogeli. IT Requirements of Business Process Management in Practice âĂŞ An Empirical Study. In *Business Process Management*, volume 6336 of *LNICS*, pages 13–28. Springer, 2011.

[RdBH04] Michael Rosemann, Tonia de Bruin, and Tapio Hueffner. A model for business process management maturity. *ACIS 2004 Proceedings*, page 6, 2004.

[Sil09] Bruce Silver. *BPMN Method and Style: A levels-based methodology for BPM process modeling and improvement using BPMN 2.0*. Cody-Cassidy Press, 2009.

[TZ11] Simon Tragatschnig and Uwe Zdun. Runtime Process Adaptation for BPEL Process Execution Engines. *2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops*, 0:155–163, 2011.

[UGSC11] Susan D Urban, Le Gao, Rajiv Shrestha, and Andrew Courter. The dynamics of process modeling: new directions for the use of events and rules in service-oriented computing. In *The evolution of conceptual modeling*, pages 205–224. Springer, 2011.

[Wes12] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[WSL09] B. Wetzstein, S. Strauch, and F. Leymann. Measuring Performance Metrics of WS-BPEL Service Compositions. In *Fifth International Conference on Networking and Services (ICNS 09)*, pages 49 –56, 2009.

Proactive Decision Support During Business Process Execution

Kimon Batoulis
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
Kimon.Batoulis@student.hpi.uni-potsdam.de

Abstract: The execution of business processes produces lots of events that can be used by complex event processing systems to analyze and improve the processes. Typically, events are stored in an event log repository that can be used to identify meaningful patterns of events, such that it is possible to react to them during process execution. However, in many cases it is beneficial to deal with those events before they actually occur to avoid an undesirable outcome, e.g., machine failure or poor performance indicator. Therefore, we present a system architecture connecting the operation of a process engine with a proactive framework. The framework forecasts events, provides the best corresponding action and generates appropriate business rules that can be used by the process engine to make optimal decisions during process runtime. An elaborated example demonstrates the utility of our concept.

Keywords: complex event processing, proactive computing, operational support, business process management

1 Introduction

By introducting the term complex event processing (CEP) [Luc01], Luckham emphasized the need for methods and tools supporting the management of today's IT systems being characterized by their event-driven nature. This is apparent by the large amount of scientific work that has been published on the subject since. In principle, CEP is about deriving complex events on the basis of patterns of events and reacting to them appropriately, e.g., by executing simple actions like buying a certain amount of stocks [EB09].

CEP can be related to business process management (BPM). BPM is concerned with concepts and techniques that can be used to model, enact and analyze business processes [Wes12]. The connection to CEP is established by realizing that the execution of business processes produces events that can be consumed by CEP systems. Those events typically have a meaning in the context of the business process or influence what will happen at a later point in time. CEP techniques can be used to respond to the occurring events in a reactive manner. However, in [EN10] it has been noted that CEP could not only be used in a reactive but also

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

in a proactive manner. The difference is that in proactive event-driven systems prediction techniques are applied in order to forecast future events and react to them *before* they occur, instead of after [EE11, EEF12, WGET12]. A little over a decade ago this has been considered "beyond the state of the art" [Luc01, p. 46], but in recent years the research field of proactive computing in CEP has evolved [EE11, EEF12, WGET12]. not only in CEP [EE11, EEF12, WGET12] but also in different areas of computer science, e.g., in dependable systems [SLM10]. The advantages are that it is possible to prevent the event from occurring, e.g., in case of failure of a machine, or to influence it in some way such that it leads to an improved performance indicator value. A detailed example dealing with the latter case is given in Section 2.

The remainder of this paper is structured as follows. Section 2 describes the example used to illustrate our concept and formulates the requirements for proactive decision support in a generic way. Section 3 explains how this problem can be approached by a proactive framework. Furthermore, it describes the complete system architecture of our concept and shows how the running example can be applied to it. This is followed by related work in Section 4. Section 5 concludes the paper.

2 Motivating Example

As an example of how the application of proactive techniques can improve a performance indicator value like time to repair, we present an example elicited and adapted from an event log of a simulated process used in the tutorial of the process mining tool ProM [1] and illustrated as a BPMN diagram in Figure 1.

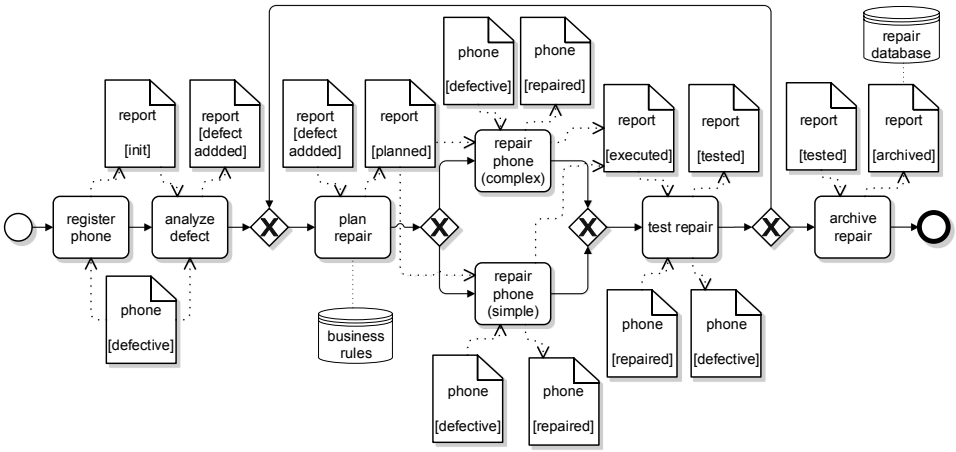

Figure 1: Telephone repair process

[1] http://www.promtools.org/prom6/

In this scenario a telephone company takes repair requests of its customers. There are two data objects in use: the defective phone and a repair report that holds various kinds of information which is updated during the execution of the process. The report's data model is shown in Figure 2. After the phone has been registered, it is analyzed to determine the type of the defect and the report is updated appropriately. Subsequently, the repair procedure is planned in that values for the fields *repairType*, *solver* and *tester* are set. The value of *repairType* is dependent on the defect type (some defects are simple, others are complex) and is the basis for the following gateway decision, leading to a simple or complex repair. The *solver* field determines the technician assigned to the current repair case and *tester* indicates who will test the phone afterwards. The activity *plan repair* is connected to a data storage containing business rules. Those rules are used to set the values of the report and will be mentioned again below. If the repair was successful, it can be archived, otherwise it needs to be replanned and restarted.

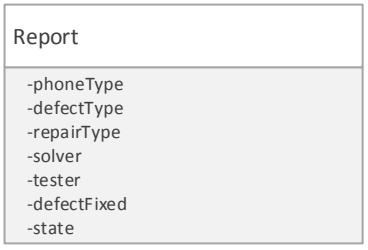

Figure 2: Repair report data model

In our example, the telephone company has three kinds of customers, namely gold, silver and standard. Different kinds of customers have different service-level agreements (SLA). For gold customers it is stated as follows: *For gold customers the repair time is faster than the mean repair time in 2/3 of the cases.* For silver customers the same holds in 1/2 of the cases and for standard customers no guarantees exist.

Thus, especially when dealing with gold customer phones, the objective is to minimize the time to repair in order to conform to the SLA. We could pursue this intent by first recognizing that, on average, the activity *repair (complex)* takes longer than *repair (simple)*. Hence, preferring the latter might be beneficial. However, the simple repair may not really solve the problem with the phone so that the test activity afterwards reveals that the repair procedure needs to be executed once again, thereby increasing the time to repair. So we are better off sticking to the result of *analyze defect*. Yet, analyzing further, we realize that not only the type of defect is crucial for deciding whether a simple or a complex repair should be executed but also other context information like the experience of available technicians. For instance, consider a scenario in which the detected defect could be solved with *repair (simple)* and that it has a certain probability of failing, meaning that the repair needs to be executed again with that probability. However, suppose that at the current time there are very experienced technicians available that can quickly execute a complex repair with a much lower probability of failing the test. Thus, we would like to recommend this action in case of gold customers.

The example given above demonstrates that there are two aspects that need to be covered. On the one hand, a prediction framework is necessary that takes as inputs the events that occurred in the system like a particular result of the *analyze defect* activity and that outputs probabilistic events. In the described scenario, the

output could be the probabilistic event that the process will be faster than the mean process time.

On the other hand, the probabilistic events need to be processed by an act framework that implements the proactive behavior by choosing actions that provide the highest utility. In the example above the utility of executing the action *repair (simple)* is weighed against that of *repair (complex)*. The decision can be based on several factors such as available resources (experienced technicians), process data (result of *analyze defect*), overall execution times and costs (more experienced technicians are more expensive).

3 Proactive Framework and System Architecture

In [EE11] a suggestion is made how to implement these two aspects of the overall framework. For example, to predict the occurrence of an event with some probability, a Bayesian network (BN) [Pea88] could be constructed (automatically). Such a network has a qualitative and a quantitative aspect. The qualitative component is a directed graph whose nodes represent random variables and whose edges represent their dependencies. The quantitative component then assigns probabilities to those dependencies. With a BN, one could state that the probability of failing the test depends on the repair type and technician chosen earlier and how this probability affects the objective of being faster than the mean repair time in 2/3 of the cases.

For the act framework some optimization procedure is necessary. Here, Markov decision processes (MDPs) have been proposed [EE11] to model a sequence of decision points over time. However, MDPs concentrate on unstructured problems involving very long or even infinite sequences of decisions [KF09]. Consequently, they are not suited to be applied to business processes characterized by their structured execution with well defined start and end events. Rather, we propose the use of influence diagrams, which essentially are BNs enhanced with decision and utility nodes. Thus, one can compute how decisions like executing a simple repair influence the probabilities of other random variables. Furthermore, one can define the utility of each combination of values of the random variables and thus compute the action/decision that yields the highest utility value.

In this work, we connect the proactive framework with the operation of a process engine such that it is possible to incorporate the proactive behavior into running business processes. This is achieved by realizing that the actions specified by the influence diagram can be stated as business rules [OMG13] that can be used to guide the process engine during decision making. In the running example, business rules are stored in the business rules storage associated with the *plan repair* activity in Figure 1. Depending on the kind of customer (gold, silver, standard) and thus the concrete SLA, the business rules will prescribe how the values of the repair report are to be set. For example, in case of gold customers the rule could say that a complex repair should be executed by a technician known to have a high success rate, such that the SLA is not violated.

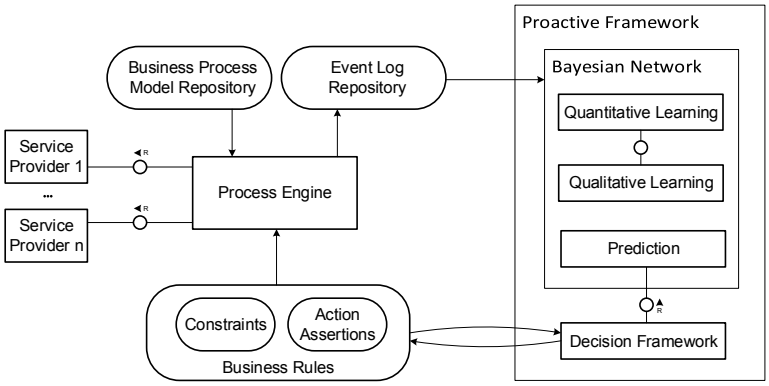

Figure 3: System architecture

The complete system architecture is illustrated in Figure 3. Business processes such as the phone repair process are executed by a process engine according to their models. This generates events such as the determined defect type that are stored in an appropriate repository. The events are then used by the BN component of the proactive framework to model the qualitative and quantitative dependencies between the events (interpreted as random variables). For instance, a dependency between the phone type and the defect type could be established and also how often a certain phone has a specific defect.

The novelty of our concept rests on the fact that the decision or optimization component interacts with a business rules storage. On the one hand, the component reads constraints from the storage, which in the example above would be the SLAs. On the basis of those constraints it makes requests to the BN in order to find action assertions that fulfill the constraints with the predicted probability. Concerning the repair process, this component could try different combinations of repair types and solvers and ask for the probability of being faster than the mean process time, given those combinations.

On the other hand, the assertions in turn are written to the business rules storage and can be read by the process engine during execution. In the example in Figure 1, this corresponds to the *plan repair* activity reading from the business rules storage. An example rule would be to instruct the experienced solver *Solver3* to execute a complex repair, given a gold customer phone of type $T3$ and defect type 5.

Note that the event log repository in Figure 3 is updated whenever a process instance is executed. Consequently, the proactive framework has more information available and can construct more accurate models, which in turn lead to more accurate actions assertions to be used in future process executions.

4 Related Work

In the area of CEP, important publications dealing with proactive behavior include [EE11] and [EEF12], where the current conceptual architecture of CEP presented in [EN10] is extended to accommodate new artifacts, like predictive and proactive agents in addition to standard event processing agents. Furthermore, concepts for implementing these new artifacts are investigated, like BNs for predictions and MDPs for actions.

There also exists a lot of work dealing only with the uncertainty present in event-driven and rule-based systems. A good overview of the nature of uncertainty in those systems is given in [WGE06]. This treatment of uncertainty is extended in [WGET08, WGET12], where the problem of the impact of uncertainty at the event source on the materialization uncertainty at the event sink is tackled with the help of probabilistic reasoning using dynamic BNs. However, this work assumes that rules for event generation and the associated probabilities are already known. This is often not the case, because even for domain experts it is hard to accurately specify those rules.

5 Conclusion

This paper presented a system architecture that couples the operation of a process engine with a proactive framework. The latter consists of a prediction and an action component. Thus, the probability of occurrence of an event can be computed on the basis of an event log and correponding actions are proposed that deal with those events. The actions depend on the specific context of the current process instance and are supplied to the process engine as business rules that are continuously updated as more information is gathered.

In future work, more use cases will demonstrate the utility of our concept. Also, BN learning algorithms tailored to the domain of BPM and CEP will be investigated and the concept of generating business rules from an optimization component like an influence diagram that can be used by a process engine needs to be detailed.

References

[EB09] Michael Eckert and François Bry. Complex Event Processing (CEP). *Informatik-Spektrum*, 32(2), 2009.

[EE11] Yagil Engel and Opher Etzion. Towards Proactive Event-driven Computing. DEBS '11, New York, 2011. ACM.

[EEF12] Yagil Engel, Opher Etzion, and Zohar Feldman. A Basic Model for Proactive Event-driven Computing. DEBS '12, New York, 2012. ACM.

[EN10] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, 1st edition, 2010.

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Luc01] David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[OMG13] OMG. Semantics Of Business Vocabulary And Business Rules (SBVR), V1.2, 2013.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, 1988.

[SLM10] Felix Salfner, Maren Lenk, and Miroslaw Malek. A Survey of Online Failure Prediction Methods. *ACM Comput. Surv.*, 42(3):10:1–10:42, March 2010.

[Wes12] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[WGE06] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. A Taxonomy and Representation of Sources of Uncertainty in Active Systems. NGITS'06, Berlin, Heidelberg, 2006. Springer-Verlag.

[WGET08] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Complex Event Processing over Uncertain Data. DEBS '08, New York, 2008. ACM.

[WGET12] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Efficient Processing of Uncertain Events in Rule-Based Systems. *IEEE Trans. Knowl. Data Eng.*, 24(1), 2012.

Challenges for Processing Events in Logistics Processes

Jan Mendling

Wirtschaftsuniversität Wien, Austria
jan.mendling@wu.ac.at

Extended Abstract

Business Process Management has proven to be a useful approach to increase the performance of enterprises, with most of its success stories being associated with office work and service processes. Specific challenges are faced in logistics processes, which partially involve physical transportation activities and also casual office work. On the other hand, there are various opportunities for taking advantage of the increasing availability of GPS data stemming for instance from transporters of airplanes and ships or from onboard devices of trucks. This can be of value for monitoring transportation activities in multi-modal logistic processes.

The mentioned challenges are addressed by GET Service, a research project in the EU's 7th Framework Programme. The main aim of this project is to develop the European Wide Service Platform for Green European Transportation. One area of the specific challenges relates to the combination of BPM concepts and complex event processing with needs of logistics monitoring, including the discretization for monitoring status based on streaming events, aggregation for monitoring activities based on fine-granular events and correlation for monitoring cargo based on events of different focus [CBM+13, GCM09].

Acknowledgements The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service).

References

[CBM+13] Cristina Cabanillas, Anne Baumgrass, Jan Mendling, Patricia Rogetzer, and Bruno Bellovoda. Towards the Enhancement of Business Process Monitoring for Complex Logistics Chains. In *11th International Conference on Business Process Management Workshop on "Process-Aware Logistics Systems"*. Springer, 2013.

[GCM09] Kerstin Gerke, Alexander Claus, and Jan Mendling. Process Mining of RFID-Based Supply Chains. In *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing*, CEC'09, pages 285–292, 2009.

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

Combining Event Processing and Support Vector Machines for Automated Flight Diversion Predictions

Cristina Cabanillas, Andreas Curik, Claudio Di Ciccio,*
Manuel Gutjahr, Jan Mendling, Johannes Prescher and Jan Simecka

Institute for Information Business
Vienna University of Economics and Business, Austria
claudio.di.ciccio@wu.ac.at

Extended abstract

Multi-modal logistics chains are those transportation processes in which different modes of transportation are involved for the delivery of goods. These modes are adopted in consecutive *legs*, which have to be synchronized. An example scenario is the delivery of goods from a production center in New York, USA, to a plant in Utrecht, the Netherlands. The transportation chain consists of two legs: *(i)* an aicraft carries the goods from John F. Kennedy International Airport (New York) to Schiphol (Amsterdam) and *(ii)* a truck sent by a *Logistics Service Provider* (LSP) transports the cargo from Dutch airport to Utrecht. The growth in international and inter-continental trade has led to a significant increase of multi-modal transports worldwide. Therefore, guaranteeing its efficiency is of crucial relevance for the successful completion of such transportation processes. In the example scenario, the goal of the LSP is to deliver the goods in time, in conformance with the *Service Level Objectives* (SLOs).

The example scenario presented in this work stems from the experience that the authors gained during the development of the GET (Green European Transport) Service[1] project. GET Service is an ongoing European research project aiming at the enhancement of logistics processes, from the viewpoint of their ecological impact and efficiency.

In this work, the research is focused on transportation processes involving aircrafts. In particular, the objective is to design and realise a service-oriented software architecture allowing for the run-time automated detection of aircrafts' diversions. A diversion consists in the landing of the aircraft in an airport that differs from the planned one. Though rare, diversions can seriously prejudice the successful

*Corresponding author
[1]http://www.getservice-project.eu/

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

completion of the transportation process. In the example, adverse weather conditions in the area of Schiphol impose the pilot to make the aircraft land in Brussels. Therefore, the LSP must reroute the truck from Schiphol to the Belgian airport to let goods be delivered to the final destination. In order for these corrective actions to be effective, it is crucial that the LSP is aware of the aircraft diversion as soon as possible. Unfortunately, experience reveals that the communication between LSPs and cargo airlines are not as prompt as required. Specifically, LSPs do not have access to real-time information and are only notified of the diversion once the aircraft has landed at another airport. This delayed notification threatens the ability of LSPs to meet their objectives. For this reason, the approach presented here sets out to reduce the impact of diversions by detecting them in a timely manner, i.e., as soon as an anomalous behaviour is recognised, while the aicraft is still flying. This approach utilises data that are publicly available, i.e., event streams reporting subsequent flight positions, altitude and speed. Thus, it is independent of the communication with airlines.

The architecture of the proposed solution is mainly based on two core modules: *(i)* a module wrapping a complex event processing engine [EN10] and *(ii)* a module based on an automated discriminative classifier [Mit97], namely a Support Vector Machine [CV95]. The latter is meant to read events reporting the flight data, in order to select, filter, aggregate over a time-span and finally transform them in a way that is readable for the former. The automated discriminative classifier is in charge of discriminating which processed flight data represent a normal behaviour from those that show the characteristics of a diverted flight. Support Vector Machines (SVMs) are *supervised learning models*, in the sense that they classify new data on the basis of a previous training phase, performed on already classified historic information. Namely, they define their decision function on the basis of input data that were previously labelled with the category they belong to. The training sessions are known to require a higher computational effort, whereas the classification on new data is a by far lighter operation.

The work of the authors in GET Service has led to promising results in the usage of SVM-based systems for the automated detection of diverted flights, in terms of accuracy. However, the tests have been conducted on log files containing a dump of collected events, thus disregarding the computational issues that may arise in a on-line elaboration of data. Therefore, given the number of flights which can be potentially monitored (order of hundreds of thousands) and the update rate of their position (order of per-second), the core modules are decoupled and deployed as services on two separate architectural components, in a distributed environment. In the presented approach, the classifier module acts as a service for the event processing engine. The event processing engine module is a service as well, invoked by client applications in order to monitor given flights.

A further advantage brought by the logical and physical decoupling of the service-oriented architecture resides in the opportunity of retraining the classifier, without empairing the time performance of the classification task. Indeed, the historic information can be enriched by newly categorised flight data. Thus, the training set

can be extended and used for a new training phase, possibly increasing the accuracy of the classifier in a continuous improving fashion.

The core modules in the proposed architecture have been respectively realised adopting Esper,[2] a well-known Java-based framework for complex event processing, and Scikit-learn,[3], a Python-based framework providing implemented Machine Learning algorithms. The prototype has been deployed in two separate web-based containers on Amazon EC2 platform. The approach has been validated connecting the event processing module to the Flightstats API[4], providing updated information on current flights.

Keywords: Complex Event Processing, Support-Vector Machines, Machine Learning, Prediction Model, Aircraft

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service).

References

[CV95] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[EN10] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Company, 2010.

[Mit97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[2] http://esper.codehaus.org/
[3] http://scikit-learn.org/stable/
[4] https://developer.flightstats.com/

Towards a Prediction Engine for Flight Delays based on Weather Delay Analysis

Cristina Cabanillas, Enver Campara, Claudio Di Ciccio, Bartholomäus Koziel, Jan Mendling, Johannes Paulitschke, and Johannes Prescher*

Institute for Information Business
Vienna University of Economics and Business, Austria
johannes.prescher@wu.ac.at

Extended abstract

Arrival performance for the United States shows that over 83 percent of flights are actually on time. However, 17 percent delayed flights are still an indisputable high number, having almost eight million commercial travel flights per year, only in the US. Knowledge of the conditions leading to flight delays may be used in a monitoring and prediction tool to diminish its impact on commercial flight operations. From a broader perspective, we also consider multi-modal logistics chains, which involve different modes of transportation. These modes are adopted in consecutive *legs*, which have to be thus synchronized. Determining whether a delay is going to be verified for the aircraft can thus be of advantage, in order to rearrange the overall transportation process involving such leg. This work reports on investigations carried out in the context of the GET (Green European Transport) Service[1] project. GET Service is an ongoing European research project, whose objective is to improve the ecological impact and efficiency of logistics processes.

Among the possible causes of flight delays, we focus on weather. Weather is observed throughout the world and the need to make future predictions is noteworthy.By now, research has analysed the influence of weather on airports, on flight delays in general, and on how a flight may be influenced by certain weather conditions. Furthermore, models for flight delays with respect to the weather and traffic index have been devised. However, there is little insight on the quantification of the prediction of flight delays.

In our work, we investigate in how far weather conditions have an actual impact on the punctuality of a flight. Following up on the insights gained in this step, we determine categories of impacts to allow for more generalisation. Subsequently, we

*Corresponding author
[1]http://www.getservice-project.eu/

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

use the categories and apply them in a prediction model. We fill the model with historical data. Accordingly, the model and corresponding data are the foundation for live predictions on actual flights. Our work builds upon the combination of two data sources being weather information and flight data. The relevant weather information is retrieved accessing the Meteorological Aerodome Report (METAR), which is an internationally established reporting instrument for weather information. METAR data is gathered at every airport and airfield and is usually generated every 30 minutes. A dataset of METAR contains station meta data (which we use to map the information to flights) as well as the information related to the weather itself. The flight data we use is of two different types, historical data and current flight data. We use historical data sets to analyse the cause for a delay and to validate our prediction model. The timeframe for our dataset ranges from 2005 to 2008. We analyse flights choosing a single route, which contains both *(i)* enough observable weather stations and *(ii)* a high amount of flights to be analysed. In order to observe and analyse the weather at a specific point in time and the position of each flight, we merge the collected information from METAR with the flight information by time, date and location for flights and weather stations. We evaluate whether there is a significant dependency between the delays of flights and certain weather conditions occurring in the meantime. We also examine at which stage of a flight specific weather conditions show the strongest impact. In order to conduct the analysis of our dataset containing 869 recorded flights we use SPSS.[2] In addition to the integration of information we suggest a conceptual description of a monitoring tool for current flights. The tool is able to predict flight delays considering the categorised impacts in conjunction with previous flight delays and may present the predicted delay. Basing on our data set we analyse specific weather conditions which potentially impact flights through multiple linear regression [DS98]. The conditions are light rain, rain, heavy rain, haze/fog, thunderstorms, light snow and snow. Our findings indicate that there is no significant influence on delays for light rain, rain and heavy rain. However, haze/fog, thunderstorm, light snow and snow seem to have a noticeable impact on flight delays. We investigate these conditions to figure out in how far they explain the delay of a flight. As a result of this analysis we derive a factor which allows for a calculation of delay time. However, while light snow will lead to delays when it appears close to the airport, it does not influence the flight at all while the airplane is flying at 30,000 feet. We therefore consider weather conditions within different stages of a flight (close to an airport or *en route*).

Our investigation indicates that the conditions' impact increases significantly once they appear closer to the airports. It identifies four weather conditions which have a significant impact on flights. These conditions lead to different lenghts of delay, which are considered within the linear equitation to predict the delays for prospective flights.

A major limitation of our work is that the influence of wind has not been considered within the analysis. Wind can be an important factor during the landing procedure in which airplanes may not be able to reduce the speed to an optimal level due to

[2] http:/www.ibm.com/software/analytics/spss/

tailwind. Additionally, wind at the cruise altitude may also be an important factor for delays. The results of our analysis are strongly dependent on the quality of the data. The historical data set of the flights includes the minutes of delays which are based on bad weather conditions. Few cases in the data set are flights which are stated to be delayed due to weather but do not show bad weather conditions according to the weather stations on their way. Furthermore, in order to predict delays, the described model obtains weather data from another prediction model, namely the weather forecast. This forecast is afflicted with uncertainty and may lead to deviations in our prediction. Although the predictions seem to be precise for the flight in our scenario, the model needs to be tested with different departure and destination airports to enhance the universal usage of the model.

Keywords: Flight Delay Prediction, Aircraft, Prediction Model, Weather, Data Acquisition

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service).

References

[DS98] Norman R. Draper and Harry Smith. *Applied Regression Analysis (Wiley Series in Probability and Statistics)*. Wiley-Interscience, third edition, 1998.

Modeling Inter-Organizational Processes

1st International Workshop, MinoPro 2014
Vienna, Austria, 19 March 2014

Preface

The First International Workshop on Modeling Inter-Organizational Processes (MinoPro) was held on 19 March 2014 as a satellite event of the Modellierung 2014 conference.

Supporting inter-organizational processes in an adequate way constitutes one of the most important challenges for process-oriented information systems nowadays. The MinoPro workshop targets at the design and evolution of such inter-organizational process scenarios.

In this spirit, the MinoPro 2014 workshop featured two keynotes by Mathias Weske on Process Abstraction, Choreographies, and Design and Manfred Reichert on The Challenges of Modeling and Evolving Cross-Organizational Processes.

Moreover, two research papers were presented which are included within these proceedings:

- Towards a Foundational Framework for Developing and Testing Inter-organizational Business Processes by Philip Langer, Stefan Sobernig and Gustaf Neumann
- Challenges of Testing Business Process Models in Intra- and Inter-Organizational Context by Stefan Mijatov and Tanja Mayerhofer

We want to thank the reviewers for providing helpful and detailed feedback. We also thank the organisation team of Modellierung 2014 for the great support.

Vienna, May 2014

Gerti Kappel
Jan Mendling
Stefanie Rinderle-Ma

Organization

MinoPro 2014 was organized in conjunction with Modellierung 2014 at the University of Vienna, Austria.

Program Committee Chairs

Gerti Kappel (Vienna University of Technology)
Jan Mendling (Vienna University of Economics and Business)
Stefanie Rinderle-Ma (University of Vienna)

Program Committee

Wil van der Aalst (Eindhoven University of Technology)
Rafael Accorsi (University of Freiburg)
Ruth Breu (University of Innsbruck)
Marlon Dumas (University of Tartu)
Schahram Dustdar (Vienna University of Technology)
Johann Eder (Alpen-Adria-Universität Klagenfurt)
Christian Huemer (Vienna University of Technology)
Gerti Kappel (Vienna University of Technology)
Julius Köpke (Alpen-Adria-Universität Klagenfurt)
Philip Langer (Vienna University of Technology)
Jürgen Mangler (University of Vienna)
Jan Mendling (Vienna University of Economics and Business)
Gustaf Neumann (Vienna University of Economics and Business)
Erik Proper (Henri Tudor Luxembourg)
Hajo Reijers (Eindhoven University of Technology)
Manfred Reichert (University of Ulm)
Stefanie Rinderle-Ma (University of Vienna)
Stefan Schulte (Vienna University of Technology)
Stefan Sobernig (Vienna University of Economics and Business)
Ingo Weber (NICTA Sydney)
Barbara Weber (University of Innsbruck)
Matthias Weidlich (Imperial College London)
Mathias Weske (Hasso Plattner Institute at the University of Potsdam)

Towards a Foundational Framework for Developing and Testing Inter-organizational Business Processes

Philip Langer
Vienna University of Technology
Vienna, Austria
langer@big.tuwien.ac.at

Stefan Sobernig, Gustaf Neumann
Vienna University of Economics and Business
Vienna, Austria
stefan.sobernig@wu.ac.at
gustaf.neumann@wu.ac.at

Abstract: Modeling and analyzing inter-organizational business processes (IOPs) is complicated substantially by heterogeneous process-modeling languages (e.g., surface vs. analysis languages) as well as by their inherent properties of loose coupling and local control (views). On top, multiple concerns must be addressed when modeling and analyzing IOPs, such as process data, behavior, distribution, and resource management. Whereas significant advances have been accomplished by the research community to model and analyze IOPs within the boundaries of a subset of these concerns, a holistic approach that enables a unified view across the strongly intermingled IOP concerns is missing. We believe that more research effort is needed towards establishing such a holistic, unified view based on a common language-oriented foundation for modeling and analyzing IOPs. This paper lays out one direction of language engineering towards this goal, collects generic and specific requirements on a language framework for inter-organizational business processes, and discusses them tentatively.

1 Introduction

As a result of the specialization and globalization of businesses in a world in which political boarders disappear gradually, cooperation among multiple organizations is increasingly important. Certainly this trend has a major impact on the development, implementation, and maintenance activities of business processes. It is often insufficient to consider only the internal processes of an organization in isolation. Instead, techniques for ensuring an efficient interplay of multiple processes across organizational boundaries and for maintaining correctness and conformance of the involved processes, while ensuring flexibility in adapting and replacing certain services on the fly, are increasingly required.

The design, enactment, and analysis of inter-organizational business processes (IOPs) has attracted substantial research, under the umbrella of inter-organizational

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

workflows [vdAW01], inter-enterprise business processes [CDT06], and process-driven SOA [HZ12]. The field, however, still poses major challenges [BDE+13]. As different organizations may use different languages and even different paradigms to specify processes [van13], we face the challenge of *heterogeneity* in the involved process models hampering a uniform analysis of the processes' interplay. In an inter-organizational setting, with *loose coupling and no control over partners*, external processes may change unnoticed and partners need to be replaced frequently. This implies continuous checking of the correctness and the conformance of the IOP. To enable the efficient development and maintenance of IOPs, as well as to ensure interoperability among the involved processes, all concerns, ranging from data, behavior, distribution through resources [BDE+13] should be treated as a conceptual whole. It remains an open challenge to come up with a *unified, holistic engineering approach* which takes all concerns of IOPs, as well as the interplay of concerns, into account.

In this paper, we present our initial ideas about a unified, foundational modeling framework for IOPs and about critical design requirements on such a framework up for discussion. The proposed framework aims to address both the heterogeneity of concepts, languages, and methods for designing, developing, and testing IOPs *and* the integration of IOP concerns, while at the same time enabling the efficient separated development of distinct concerns with tailored languages. It is evident that such a framework cannot be established without the help and the feedback of the broader research community, including the MinoPro committee and the MinoPro participants. To this end, this paper should act as a stimulus for joint research and critical discussion.

In Section 2, we first elaborate on the language-oriented engineering process that we plan to apply in our work towards the foundational modeling framework. To gather a first set of IOP-specific requirements on both the engineering process and the resulting language artifacts, we provide a selective survey of existing work in designing, developing, and testing IOPs in Section 3. This way, we outline how we plan to aggregate the current state of the art in IOP languages, concepts, semantics, and formal techniques by identifying commonalities and variations. Based on these aggregated findings, we then briefly discuss cornerstones of a foundational IOP modeling framework in Section 4.

2 Language Engineering for Integrated IOP Modeling

A *model* of an IOP is a precise definition of the intended characteristics of multiple interacting business processes. An IOP model is expressed in a formally defined software language; that is, a language having a mathematical definition of both its syntax and semantics. Therefore, the model becomes amenable to verification and validation. In addition, an IOP model is executable [Fuc92] to allow for immediate behavioral inspection (e.g., in a simulated environment) or to actually enact a process (e.g., by instrumenting a process-driven, service-based software system). An IOP model covers either one or even multiple process concerns at the same time (data,

behavior, distribution, resources; see also Section 3). To allow for concern-specific, partial IOP models or for concern-specific viewpoints on an integrated IOP model, a language-based IOP framework is realized as a family of modeling languages. A *software-language family* [ZKD+10] is an infrastructure to create a number of integrated, tailored software languages based on a number of shared language-implementation assets. These can be used to define tailored abstract syntaxes, tailored concrete syntaxes, as well as derived semantics specifications, based on a common core of language abstractions and semantics. An IOP modeling language can be tailored towards an IOP concern (concern-specific view, in short; [BDE+13]) and/or towards a domain-specific process view [CDT06].

We plan to follow documented procedural guidelines for domain-specific software-language engineering ([Fra13, SZ09]; see also Figure 1). Early in the engineering process, the *scope* for and the *purpose* of the IOP language framework must be established. This involves systematic reviews and a systematic mapping of existing IOP research, in particular language projects and language-driven development tasks (process enactment, verification, validation). In addition, the availability of reusable IOP language artifacts and profiles of IOP stakeholders should be investigated. This step is particularly critical because the framework should incorporate and integrate with existing IOP languages. Collecting and eliciting *generic requirements* involves consulting collections of documented software-language design rationale (e.g., architectural patterns, language-implementation patterns, workflow patterns) as well as decisions on the subsequent engineering steps (e.g., extraction-based vs. mockup-based language development; [SZ09]). The evaluation of different language-implementation strategies (e.g., metamodeling, grammars, language embedding) and of different concrete-syntax types (e.g., diagrammatic vs. textual) also fall into this category. Realizing the *semantics* of modeling languages may either be specified by mapping the language onto an existing semantic domain (such as petri nets) or by weaving the semantics into the metamodel (operational semantics [MFJ05, MLW13]) using a language for which the semantics are already available [BGM+11]. Then, there are *IOP-specific requirements* to be documented using a precise representation. For example, usage scenarios, concrete-syntax mock-ups etc. can be extracted from the systematic map of language projects drawn up in a previous step. In addition, domain-specific sources are consulted, such as material on workflow patterns [vdATHKB03]. Principles of view-based process modeling, referring to concern-specific and stakeholder-specific process views, will be screened [TZD11, Val10, THZD09].

Specifying *language-implementation assets* comprises the construction of a core abstract syntax and semantics, as well as syntax and semantics extensions covering the variable language parts. This is then complemented by realizing one or several *concrete syntaxes*. Finally, repeated empirical *evaluations*, for instance, based on the initially collected scenarios, are conducted and the results are fed back into re-designing iterations. The present paper contributes to a first structuring of relevant IOP sub-domains in preparation of the first three steps (scoping and requirements gathering; see Figure 1).

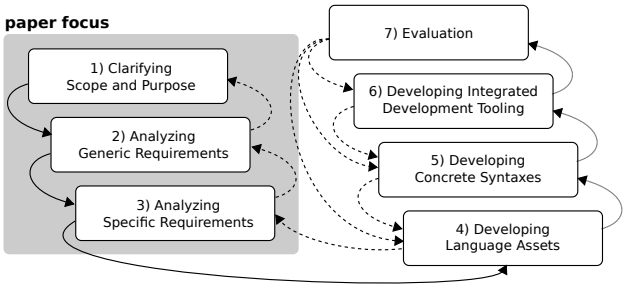

Figure 1: Overview of language-family engineering steps based on [Fra13, SZ09]

3 State of the Art in Developing and Testing IOPs

There is a large body of material on software languages for IOP development and on corresponding techniques of verification and validation of business-process models. In the following, we highlight selected contributions (e.g., by identifying a single flagship publication per technique) to identify the underlying concepts and formal techniques, rather than providing a comprehensive survey, within the page limits set for this paper.

3.1 Data

The concern of data addresses the structural and semantical aspects of messages that are exchanged among participants of the IOP. As IOPs might not have one party serving as central control point, also the data can be distributed and heterogeneous posing the challenge of transactions and semantic integration, respectively.

Languages. The approaches to modeling data structures have converged over past decades. We can distinguish among relational approaches, such as the relational model by Codd [Cod70] and the Entity-Relationship Model by Chen [Che76], tree-oriented approaches, such as XML Schema, object-oriented approaches, such as UML class diagrams, graph-oriented approaches, such as RDF and the introduction of ontologies with OWL, and document-oriented approaches [KD10], such as UN/CEFACT Core Components. Although they are based on different paradigms, there has been a consensus and a clear understanding about these paradigms and how they can be translated among each other without loosing information (e.g., object-relational mapping and JAXB[1]).

Formal techniques. To address the challenges of data distribution and distributed data integrity, transaction models of databases have been adapted to the workflow context [AAE+96] and web services [SD05]. With respect to heterogeneity and interoperability, for instance, Deutsch et al. [DHPV09] proposed a verification technique of exchanged data for data-centric processes. In case the exchanged artifacts are incompatible—requiring data transformations—semantic annotations [VIK+10] were applied successfully. To cope with evolution of data structures, semantic anno-

[1] `https://jaxb.java.net`

tation paths were proposed for enriching WSDL interfaces [KE12]. Moreover, Weber et al. [WHM10] combine the control-flow verification with checking pre-conditions on data items expressed in semantic process descriptions (OWL-S and WSMO).

3.2 Behavior

IOP behavior refers to control- and data-flow aspects (at design time), as well as possible runtime states during process execution. Key properties are correctness (e.g., soundness, liveness), as well as process conformance with business rules.

Languages. When it comes to modeling the behavior of business processes, the landscape of paradigms and languages is more diverse and little consensus exists on fundamental concepts [van13]; especially when looking at language adoption in industry. The applied languages in modeling of business processes mainly depend on the underlying purpose of the IOP model. Modeling processes with the aim of conceptually defining them is mainly done using semi-formal languages, such as BPMN, EPCs, and UML activities. Recently, declarative approaches such as DECLARE [vdAPS09], gained momentum for specifying loosely structured processes. Rather than defining the control flow explicitly, declarative process models specify a list of activities and constraints regarding the execution order of those activities, usually captured using linear temporal logic (LTL) expressions. If the purpose of an IOP model concerns analysis and verification, rather than modeling them conceptually only, *formal languages* are employed, such as petri nets, finite state automata, and process calculi. Whereas both conceptual and formal process models abstract from implementation detail, *execution languages*, such as BPEL, aim at capturing enough detail to enable the deployment and the enactment of the modeled processes. Note that formalizations of conceptual modeling languages and execution languages exist [ODv+09], however, these formalizations are often incomplete and ignore ambiguities in the source languages.

Formal techniques. The verification of business processes correctness has been intensively studied for well-defined correctness properties [WVvdA+09]. Most of this work focuses on control-flow analysis based on process models conforming to a clear token-flow semantics, such as petri nets and workflow nets. These formal concepts and verification methods have also been applied successfully to execution languages, such as BPEL, and more conceptual languages, such as UML activities. This, however, implies limiting the support to a subset of the languages' concepts or abstracting from ambiguities stemming from the informally defined semantics of conceptual languages. Design-time verification methods have also been developed for declarative process models to check whether, for instance, a so-defined process contains dead activities (i.e., there is no valid execution order containing the respective activity) or conflicting constraints (i.e., there is no valid execution order at all; [vdAPS09]). These methods have later been adopted also for run-time verification [MWMvdA12].

Besides existing work on process verification, several researchers also addressed the validation of business processes according to certain functional requirements. Functional requirements are mostly specified using expected execution orders [ZPW12]

or conformance relationships to reference models, which are checked based on graph comparisons [DDVD+11], on execution traces [GCC09], on causal dependencies of activities [WPMW11], and on equivalence notions using model checking [LKRM+11]. Besides, there is a unit-testing framework for UML activities [MLMK13], which considers alongside execution orders, also the object states. Similarly, a unit-testing framework for BPEL processes [ML06] enables to specify and to check assertions on messages. Furthermore, methods from the domain of program analysis have been adopted to process models, such as symbolic execution [BPZ09] and test-case generation [YLY+06].

3.3 Distribution

One of the most obvious and, at the same time, urging concerns of IOPs is distribution, which includes the coordination and choreography of organizations participating in IOPs, as well as testing of the interoperability and conformance of interacting processes.

Languages. For modeling the distribution of control, two approaches are applied predominantly [DKB08]. Using *interaction models*, the choreography of processes is defined in terms of a workflow containing activities that represent the message exchange among participating partners (e.g., BPMN 2). As the interaction is specified in terms of a process, choreographies may be specified using the same formalisms as used for specifying the behavior of the processes themselves. In contrast, with *interconnected interfaces modeling*, the control flow is defined per participant, i.e., the individual interface behavior models are stitched together using message links. Approaches to bridge between the two schemes have been proposed [MH08].

Formal techniques. From the perspective of distribution of IOPs, most of the work is concerned with validating the functional conformance and interoperability with respect to the behavior of the interacting processes. The conformance requirements can be specified using reference models, contracts [vdALM+10] or, more generally, conformance rules turning into assertion checks on the execution order and, potentially, on the system state. Therefore, validation methods for checking the conformance of *intra*-organizational processes have been adopted and extended also for validating conformance and interoperability of IOPs, such as checking structural and semantic conformance [LRMGD12], using the notion of causal dependencies of activities [WPMW11], or applying model checking to verify certain conformance rules or correctness properties [KRFRM13].

3.4 Resources

Key to IOPs are means of adaptive resources management. Resources include computational, network, storage, and human resources required to carry out technical and non-technical tasks set by a business process. From a resources perspective, a business process translates into a set of precedence-related activities, which are to

be executed on a set of resources. In an inter-organizational setting, however, processes are required to run under a variable, unbounded set of resources. Therefore, achieving on-demand (*elastic*) re- and de-allocation of resources to tasks has been identified as a key challenge [BDE+13, DGST11].

Languages. The notions of different types of resources (e.g., network, computation, human) and different types of elementary resource management activities, such as monitoring, scheduling, allocating, must be made explicit in IOP models. Attempts to integrate business-process modeling with resources modeling at the language level are still rare and in an early stage. Tai et al. [TLD12] put forth the conceptual notion of infrastructure units which extends to human process resources (a.k.a. *social compute units* as in [DB11]). More recently, Janiesch et al. [JWKM14] have proposed a set-theoretic metamodel to model key deployment abstractions of business processes (e.g., entities such as services, virtual machines, processes, and tasks as well as their relations in terms of service or task dependencies) on distributed resources infrastructures. Previous approaches on adaptive resource management build on similar conceptual metamodels, e.g., to express their reactive algorithms and their predictive statistical heuristics [HSD13]. Alternatives are formal metamodeling techniques, such as an EMOF metamodel and auxiliary semantics expressed in OCL [CEM+10]. For predictive scheduling, colored petri nets have also been proposed to model dependency structures between tasks [AMT13].

Resources modeling borrows many abstractions from languages for modeling software-system policies [DDLS01], including service-level agreements [HSD13], and for modeling (business) rules and rules interchange [MVG11]. Languages for defining the various deployment descriptors in business-process execution as well as for software services, distributed software components, and distributed computation containers (e.g., cloud nodes, virtual machines) are the second point of reference. The latter include extensions to description languages such as OVF [CEM+10, CMTD13]. To this end, the resources concern strongly relates to the IOP distribution concern (see above). A second linkage between the two concerns are automated resource management approaches building, for instance, on distributed service-based QoS monitors [HSD13] and on agent-based resources (re-) negotiation [WB13].

Formal techniques. In the business-process context, work on adaptive resources management and formal resources modeling addresses primarily optimality properties of resource-aware business processes (e.g., cost and/or time optimality [BYO+12]) from the perspective of different process stakeholders (e.g., customer, process participant, IaaS provider) and for different scopes (e.g., for a single and across multiple processes or process instances). Second, identifying service level agreement (SLA) violations ahead of time is an open research issue. A first set of works relies mainly on constructing, evaluating, and calibrating *statistical prediction systems* based on monitoring data. Leitner et al. [LFHD13], for example, devise an approach to create and to train statistical prediction models (e.g., decision trees, neural networks, and auto-regression models) based on process monitoring data to deliver forecasts on service level objectives (SLOs), such as delivery time, service availability, etc. in service-based systems. Another family of approaches employs

online-testing techniques to establish whether SLA/SLOs are expected to be violated and to trigger adequate resources adaptations. Online testing involves auxiliary, model-based test-case generation and selection techniques [DMK10]. Ivanović et al. [ICH11], on the contrary, apply an analytical approach based on deriving and solving *constraint sets* over atomic QoS probes (e.g., available for single services) and orchestration structure to predict upper and lower bounds for the expected, orchestrated QoS. A third challenge is correctness checking of adaptation (elasticity) operations [DGST11]. Amziani et al. [AMT13] employ an equivalence method on condensed state spaces of *colored petri nets* to proof that the adaptation operations proposed (resource duplication and removal) do not have unwanted side effects during process enactment (e.g., an increase in invocations). In addition, the authors propose model checking on *reachability graphs*, derived from colored petri nets representing resource-aware processes, to establish whether important properties hold under certain adaptation operations, including QoS violation by exceeding maximum capacities, deadlocks during call transfers, and adaptation loops.

4 Sketching Language Foundations for IOPs

Ensuing from the body of existing concepts, languages, and methods across all four crucial IOP concerns, we aim to clarify the scope of the foundational framework, elicit important generic and specific requirements (see Section 2), and discuss initial ideas towards the foundational core for IOPs.

Generic Requirements. The language-oriented, foundational IOP framework must balance between two opposing but closely related forces. To provide effective modeling support, on the one hand, the framework must enable the development for distinct IOP concerns using tailored languages at different levels of abstraction (e.g., design vs. analysis languages). On the other hand, to allow for global IOP testing, partial concern-specific view models of an IOP must be integrated at some point and must be kept consistent during model co-evolution. Due to the magnitude of existing concepts, languages, and methods, another highly critical generic requirements is to accomplish a minimal foundational core language that enables a convenient integration of existing work and that provides an inherent extension mechanism to meet future requirements in IOP research. Therefore, this mechanism should allow for building syntactic and semantic extensions by instantiating and reassembling concepts and semantics of the foundational core language.

IOP-specific requirements. In Section 3, we highlighted flagship contributions on fundamental concepts and formal techniques from the otherwise extensive body of existing work on IOPs. Key findings are that relevant *structural concepts* include document-oriented or object-oriented data structures, constraints on data structures, through to concepts from knowledge representation, such as description logics. Based on these structural formalisms, conformance checking between messages and process/service interfaces or contracts is performed. Additionally, inference mechanisms for semantic alignment of heterogeneous data structures in messages becomes available.

Behavioral concepts of IOPs are described at different abstraction levels. This includes the notions of opaque or precisely specified activities and their execution dependencies, either using an explicit control-flow specification or temporal logics. Regarding their semantic expressiveness, for many analytical applications, both can be seen as equivalent. For instance, in both approaches a labeled transition system (cf. reachability graph of petri nets) can be computed to represent the state space. However, in temporal logics there is no explicit notion of concurrency, although nondeterminism can be modeled. Nevertheless, explicit concurrency modeling—such as enabled by petri nets, actor models, and process calculi—can be paramount. Besides, modeling synchronous and asynchronous message passing through channels is crucial, especially for choreographies using interconnected interfaces modeling. Other behavioral paradigms being used are event-driven process design, building on event-condition-action (ECA) rules, and state charts.

As for *formal techniques*, petri nets and declarative workflow models are subjected to *control- and data-flow analyses* extensively. By computing a concrete or symbolic state space from IOP models, e.g., in terms of a labeled transition system, also *model checking* techniques have been proven useful to verify business rules and IOP conformance rules. Since process enactment can be simulated, based on paths and path conditions, another line of research applies techniques from program analysis to business processes, such as symbolic execution and test-case generation. Besides these behavioral formalisms, we also identified *stochastic and non-functional concepts* being introduced to process models, such as time and computing resources. These formalisms allow to run predictive analyses and simulations regarding SLAs, quality of service, and resources management.

Towards a Foundational Core for IOPs. In next steps, the specific and generic requirements identified above will guide us towards condensing existing languages, concepts, and semantics into a common foundational core for IOPs. The objectives are that existing formal verification techniques and validation techniques should be supported as is, while also allowing for combining these techniques in novel ways and across IOP concerns. Please also note that the foundational core must not necessarily correspond to popular surface languages (i.e., the languages currently adopted IOP stakeholders). More importantly, the core should focus on key concepts and semantics, capable of reflecting and realizing the semantics of several surface languages *across* IOP concerns (e.g., ECA semantics in the distribution and resources concern). To accommodate changing requirements, a minimal but extensible subset of language abstractions is clearly preferable over a union of all available concepts and languages, which is why we envision to adopt principles and techniques from (model-driven) language engineering for providing built-in extension points regarding abstract syntax, semantics, and concrete syntax to address different IOP concerns.

In a model-driven approach, the foundational core of IOPs could build on a small object-oriented structure modeling language, such as EMOF or a subset of the UML class metamodel. For behavioral concerns, given the large body of research on verification and correctness of workflow nets, we plan to evaluate the adoption of compatible control-flow and object-flow semantics. This way, deriving labeled

transition systems from process models as a basis for model checking becomes possible. In addition, modeling facilities for synchronous and asynchronous message flows will be considered. Interactions modeling between different process partners (and partner-specific process views) across organizational boundaries will require particular structuring concepts in behavioral models. At the same time, the manipulation of objects expressed in the structural language core should be supported (e.g., to allow for representing and for reasoning about state changes in precisely specified processes). For simulation-based analyses, IOP models should have the ability to represent runtime information; therefore, the foundational core should be capable of capturing execution states, events, and traces, which in turn would enable symbolic execution and test-case generation.

A potential candidate for adoption as a languages in the IOP core is the recently standardized foundational UML[2] (fUML), which consists of a subset of activities and classes with formally specified execution semantics. fUML adopts the token-flow semantics of petri nets and supports—besides object manipulation—also concurrency and signals for asynchronous message passing.

For providing *syntactic extensions* of languages (e.g., to add stochastic distributions to edges), the foundational core should provide lightweight extension mechanisms [LWWC12], as known from UML profiles [FFVM04]. The *extensibility* of core semantics in modeling languages is still an open research topic [BGM+11]. One idea is to introduce *semantic profiles*; that is, profiles for which an operational or translational semantics, again based on the foundational core for IOPs, can be specified. With such semantic profiles (e.g., one for event-driven behaviors), they can be applied on top of the core semantics by injection.

5 Concluding Remarks

With this paper, we present initial ideas on a language-engineering approach to construct a foundational modeling framework for designing, analysing, and testing inter-organizational processes. A resulting framework is foundational in the sense of enabling designing, developing, and testing IOPs in a unified manner, across the boundaries of interdependent IOP concerns. In addition, following this language-engineering procedure, we set the scope and identified first requirements for such a framework based on a mapping of existing concepts and methods available for BPM.

We see this paper as a first step towards building and establishing such a framework. A key challenge is designing the framework at the sweet spot of expressiveness, of facilitating IOP analyzability, and of equipping the framework with extensibility to meet future requirements in IOP research. We kindly invite the readers and the wider research communities on modeling inter-organizational processes to provide feedback and to join forces on a community-driven effort to tackle this challenge.

[2] fUML; `http://www.omg.org/spec/FUML/1.0`

References

[AAE+96] Gustavo Alonso, Divyakant Agrawal, Amr El Abbadi, Mohan Kamath, Roger Günthör, and C. Mohan. Advanced Transaction Models in Workflow Contexts. In *Proc. ICDE*, pages 574–581. IEEE, 1996.

[AMT13] Mourad Amziani, Tarek Melliti, and Samir Tata. Formal Modeling and Evaluation of Stateful Service-Based Business Process Elasticity in the Cloud. In *Proc. OTM*, volume 8185 of *LNCS*, pages 21–38. Springer, 2013.

[BDE+13] Ruth Breu, Schahram Dustdar, Johann Eder, Christian Huemer, Gerti Kappel, Julius Köpke, Philip Langer, Jürgen Mangler, Jan Mendling, Gustaf Neumann, Stefanie Rinderle-Ma, Stefan Schulte, Stefan Sobernig, and Barbara Weber. Towards Living Inter-Organizational Processes. In *Proc. CBI*. IEEE, 2013.

[BGM+11] Barrett R Bryant, Jeff Gray, Marjan Mernik, Peter J Clarke, Robert B France, and Gabor Karsai. Challenges and directions in formalizing the semantics of modeling languages. *Computer Science and Information Systems*, 8(2):225–253, 2011.

[BPZ09] Lina Bentakouk, Pascal Poizat, and Fatiha Zaïdi. A formal framework for service orchestration testing based on symbolic transition systems. In *Proc. TESTCOM*, volume 5826 of *LNCS*, pages 16–32. Springer, 2009.

[BYO+12] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan. Resources allocation and scheduling approaches for business process applications in Cloud contexts. In *Proc. CloudCom*, pages 496–503. IEEE, 2012.

[CDT06] Issam Chebbi, Schahram Dustdar, and Samir Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56(2):139–173, 2006.

[CEM+10] C. Chapman, W. Emmerich, F.G. Marquez, S. Clayman, and A. Galis. Elastic service definition in computational clouds. In *Workshop Proc. NOMS*, pages 327–334. IEEE, 2010.

[Che76] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[CMTD13] Georgiana Copil, Daniel Moldovan, Hong Linh Truong, and Schahram Dustdar. SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications. In *Proc. CCGrid*, pages 112–119. IEEE, 2013.

[Cod70] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[DB11] Schahram Dustdar and Kamal Bhattacharya. The Social Compute Unit. *IEEE Internet Computing*, 15(3):64–69, 2011.

[DDLS01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder Policy Specification Language. In *Workshop Proc. POLICY*, volume 1995 of *LNCS*, pages 18–38. Springer, 2001.

[DDVD+11] Remco Dijkman, Marlon Dumas, Boudewijn Van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516, 2011.

[DGST11] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong-Linh Truong. Principles of Elastic Processes. *IEEE Internet Computing*, 15(5):66–71, 2011.

[DHPV09] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proc. ICDT*, pages 252–267. ACM, 2009.

[DKB08] Gero Decker, Oliver Kopp, and Alistair Barros. An Introduction to Service Choreographies. *Information Technology*, 50(2):122–127, 2008.

[DMK10] Dimitris Dranidis, Andreas Metzger, and Dimitrios Kourtesis. Enabling Proactive Adaptation through Just-in-Time Testing of Conversational Services. In *Proc. ServiceWave*, volume 6481 of *LNCS*, pages 63–75. Springer, 2010.

[FFVM04] Lidia Fuentes-Fernández and Antonio Vallecillo-Moreno. An introduction to UML profiles. *UPGRADE*, V(2):6–13, 2004.

[Fra13] Ulrich Frank. Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In *Domain Engineering*, pages 133–157. Springer, 2013.

[Fuc92] N.E. Fuchs. Specifications are (preferably) executable. *Software Engineering Journal*, 7(5):323–334, 1992.

[GCC09] Kerstin Gerke, Jorge Cardoso, and Alexander Claus. Measuring the compliance of processes with reference models. In *Proc. OTM*, volume 5870 of *LNCS*, pages 76–93. Springer, 2009.

[HSD13] P. Hoenisch, S. Schulte, and S. Dustdar. Workflow Scheduling and Resource Allocation for Cloud-Based Execution of Elastic Processes. In *Proc. SOCA*, pages 1–8. IEEE, 2013.

[HZ12] Carsten Hentrich and Uwe Zdun. *Process-Driven SOA: Patterns for Aligning Business and IT*. Infosys Press, 2012.

[ICH11] Dragan Ivanović, Manuel Carro, and Manuel Hermenegildo. Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations. In *ICSOC*, volume 7084 of *LNCS*, pages 62–76. Springer, 2011.

[JWKM14] Christian Janiesch, Ingo Weber, JÃűrn Kuhlenkamp, and Michael Menzel. Optimizing the Performance of Automated Business Processes Executed on Virtualized Infrastructure. In *HICSS*, pages 3818–3826, 2014.

[KD10] Yildiray Kabak and Asuman Dogac. A Survey and Analysis of Electronic Business Document Standards. *ACM Computing Surveys*, 42(3):11:1–11:31, 2010.

[KE12] Julius Köpke and Johann Eder. Logical invalidations of semantic annotations. In *Proc. CAiSE*, volume 7328 of *LNCS*, pages 144–159. Springer, 2012.

[KRFRM13] David Knuplesch, Manfred Reichert, Walid Fdhila, and Stefanie Rinderle-Ma. On Enabling Compliance of Cross-organizational Business Processes. In *Proc. BPM*, volume 8094 of *LNCS*. Springer, 2013.

[LFHD13] Philipp Leitner, Johannes Ferner, Waldemar Hummer, and Schahram Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, 2013.

[LKRM+11] Linh Thao Ly, David Knuplesch, Stefanie Rinderle-Ma, Kevin Göser, Holger Pfeifer, Manfred Reichert, and Peter Dadam. SeaFlows Toolset–compliance verification made easy for process-aware information systems. In *Proc. CAiSE Forum IST*, volume 72 of *LNBIP*, pages 76–91. Springer, 2011.

[LRMGD12] Linh Thao Ly, Stefanie Rinderle-Ma, Kevin Göser, and Peter Dadam. On enabling integrated process compliance with semantic constraints in process management systems. *Information Systems Frontiers*, 14(2):195–219, 2012.

[LWWC12] Philip Langer, Konrad Wieland, Manuel Wimmer, and Jordi Cabot. EMF Profiles: A Lightweight Extension Approach for EMF Models. *Journal of Object Technology*, 11(1), 2012.

[MFJ05] Pierre-Alain Muller, Franck Fleurey, and Jean-Marc Jézéquel. Weaving Executability into Object-Oriented Meta-languages. In *Proc. MoDELS*, volume 3713 of *LNCS*, pages 264–278. Springer, 2005.

[MH08] Jan Mendling and Michael Hafner. From WS-CDL choreography to BPEL process orchestration. *J. Enterprise Inf. Management*, 21(5):525–542, 2008.

[ML06] Philip Mayer and Daniel Lübke. Towards a BPEL unit testing framework. In *Workshop Proc. TAVWEB*, pages 33–42. ACM, 2006.

[MLMK13] Stefan Mijatov, Philip Langer, Tanja Mayerhofer, and Gerti Kappel. A Framework for Testing UML Activities Based on fUML. In *Workshop Proc. MoDeVVa*, volume 1069, pages 1–10. CEUR-WS.org, 2013.

[MLW13] Tanja Mayerhofer, Philip Langer, and Manuel Wimmer. xMOF: A Semantics Specification Language for Metamodeling. In Yan Liu, Steffen Zschaler, Benoit Baudry, Sudipto Ghosh, Davide Di Ruscio, Ethan K. Jackson, and Manuel Wimmer, editors, *Demos/Posters/StudentResearch@MoDELS*, volume 1115 of *CEUR Workshop Proceedings*, pages 46–50. CEUR-WS.org, 2013.

[MVG11] D. Moran, L.M. Vaquero, and F. Galan. Elastically Ruling the Cloud: Specifying Application's Behavior in Federated Clouds. In *Proc. CLOUD*, pages 89–96. IEEE, 2011.

[MWMvdA12] Fabrizio Maria Maggi, Michael Westergaard, Marco Montali, and Wil van der Aalst. Runtime verification of LTL-based declarative process models. In *Proc. RV*, volume 7186 of *LNCS*, pages 131–146. Springer, 2012.

[ODv+09] Chun Ouyang, Marlon Dumas, Wil van der Aalst, Arthur HM Ter Hofstede, and Jan Mendling. From business process models to process-oriented software systems. *ACM transactions on software engineering and methodology*, 19(1):2, 2009.

[SD05] Benjamin A. Schmit and Schahram Dustdar. Systematic Design of Web Service Transactions. In *Workshop Proc. TES*, LNCS, pages 23–33. Springer, 2005.

[SZ09] M. Strembeck and U. Zdun. An Approach for the Systematic Development of Domain-Specific Languages. *Software: Practice and Experience*, 39(15):1253–1292, 2009.

[THZD09] Huy Tran, Ta'id Holmes, Uwe Zdun, and Schahram Dustdar. *Handbook of Research on Business Process Modeling*, chapter Modeling Process-Driven SOAs, pages 27–48. IGI Global Hershey, 2009.

[TLD12] Stefan Tai, Philipp Leitner, and Schahram Dustdar. Design by Units: Abstractions for Human and Compute Resources for Elastic Systems. *IEEE Internet Computing*, 16(4):84–88, 2012.

[TZD11] Huy Tran, Uwe Zdun, and Schahram Dustdar. VbTrace: using view-based and model-driven development to support traceability in process-driven SOAs. *Software and System Modeling*, 10(1):5–29, 2011.

[Val10] Antonio Vallecillo. On the Combination of Domain Specific Modeling Languages. In *Proc. ECMFA*, volume 6138 of *LNCS*, pages 305–320. Springer, 2010.

[van13] Wil van der Aalst. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, 2013, 2013.

[vdALM+10] Wil van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. Multiparty contracts: Agreeing and implementing interorganizational processes. *The Computer Journal*, 53(1):90–106, 2010.

[vdAPS09] Wil van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2):99–113, 2009.

[vdATHKB03] Wil van der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[vdAW01] Wil van der Aalst and Mathias Weske. The P2P Approach to Interorganizational Workflows. In *Proc. CAiSE*, volume 2068 of *LNCS*, pages 140–156. Springer, 2001.

[VIK+10] Marko Vujasinovic, Nenad Ivezic, Boonserm Kulvatunyou, Edward Barkmeyer, Michele Missikoff, Francesco Taglino, Zoran Marjanovic, and Igor Miletic. Semantic mediation for standard-based B2B interoperability. *IEEE Internet Computing*, 14(1):52–63, 2010.

[WB13] Yi Wei and M.B. Blake. Decentralized Resource Coordination across Service Workflows in a Cloud Environment. In *Workshop Proc. WETICE*, pages 15–20. IEEE, 2013.

[WHM10] Ingo Weber, Jörg Hoffmann, and Jan Mendling. Beyond soundness: on the verification of semantic business process models. *Distributed and Parallel Databases*, 27(3):271–343, 2010.

[WPMW11] Matthias Weidlich, Artem Polyvyanyy, Jan Mendling, and Mathias Weske. Causal behavioural profiles–efficient computation, applications, and evaluation. *Fundamenta Informaticae*, 113(3):399–435, 2011.

[WVvdA+09] Moe Thandar Wynn, HMW Verbeek, Wil van der Aalst, Arthur HM ter Hofstede, and David Edmond. Business process verification–finally a reality! *Business Process Management Journal*, 15(1):74–92, 2009.

[YLY+06] Jun Yan, Zhongjie Li, Yuan Yuan, Wei Sun, and Jian Zhang. BPEL4WS unit testing: Test case generation using a concurrent path analysis approach. In *Proc. ISSRE*, pages 75–84. IEEE, 2006.

[ZKD+10] Steffen Zschaler, DimitriosS. Kolovos, Nikolaos Drivalos, RichardF. Paige, and Awais Rashid. Domain-Specific Metamodelling Languages for Software Language Engineering. In *Proc. SLE*, volume 5969 of *LNCS*, pages 334–353. Springer, 2010.

[ZPW12] Stefan Zugal, Jakob Pinggera, and Barbara Weber. Creating declarative process models using test driven modeling suite. In *CAiSE Forum IS Olympics*, volume 107 of *LNBIP*, pages 16–32. Springer, 2012.

Challenges of Testing Business Process Models in Intra- and Inter-Organizational Context

Stefan Mijatov, Tanja Mayerhofer

Business Informatics Group
Vienna University of Technology, Austria
{mijatov, mayerhofer}@big.tuwien.ac.at

Abstract: In order to address the issue of complexity and changeability of business processes, as well as of the information technology used to implement these processes, business process models are used. Validating the functional correctness of such models is essential. Many approaches exist, that deal with validating the functional correctness of business process models in an *intra-organizational* context (i.e., the process is controlled and maintained by one business entity). However, today, business processes are usually carried out by several business partners, each providing its own services to accomplish more complex *inter-organizational* business processes. This induces new challenges for validating the functional correctness of business processes based on models. In this paper we provide an overview of existing approaches to validate business process models in both intra- and inter-organizational context and discuss arising challenges.

1 Introduction

In order to cope with complexity and changeability of business processes on one hand, and the information technology used to implement these processes on the other, *business process models* are created and maintained. These models can be defined at various levels of abstraction, from capturing the high level concepts of a business process (high level activities performed by different parties to realize a business process) to a low level specification of a business process (information and object manipulation activities). Furthermore, these models can be defined with different modeling languages, from standardized modeling languages such as BPMN [OMG11a], UML [OMG11b], and BPEL [OAS07], to proprietary modeling languages.

As business process models become central artifacts in business process management as well as in the development of information systems, ensuring their quality is of uttermost importance. The quality of a software artifact (e.g., a business process model) represents the correspondence of its realization to the specified requirements. These requirements can be functional (i.e., what the artifact should do) and non-functional (i.e., the level of performance that the artifact has to achieve). In this

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

paper we focus on techniques for validating the functional correctness of business processes based on models.

There are many challenges that have to be addressed in this context. On one hand, organizations use different modeling languages for specifying their business processes. For validating the functional correctness of models, precisely defined semantics of the used modeling languages is a prerequisite. However, many organizations use modeling languages without or with partially defined semantics. On the other hand, in today's highly globalized business world, business processes are often realized by several cooperating organizations in order to achieve a common business objective. In this context, new challenges arise, such as autonomy, heterogeneity, and the support for coordination, which all may induce additional challenges to validating business processes.

This paper is structured as follows. In Section 2 and Section 3 we present existing approaches for validating functional properties of business processes based on business process models in an intra- and inter-organizational context. Thereby, we consider analysis, simulation, verification, as well as testing techniques. We focus on the three most popular modeling languages for business process modeling, namely BPMN, UML activity diagrams, and BPEL. In Section 4, challenges arising for validating both intra- and inter-organizational business processes are discussed. In Section 5 we briefly describe our own testing framework for UML activity diagrams based on the fUML standard [OMG11c] and discuss how the identified challenges can be addressed by this framework. Finally, we conclude our paper in Section 6.

2 Validating Intra-Organizational Business Process Models

In this section we present relevant existing work on the validation of business processes modeled with the Business Process Model and Notation language [OMG11a], UML Activity Diagrams [OMG11b], and the Business Process Execution Language (BPEL) [OAS07], which represent the most popular modeling notations used in both academy and industry.

2.1 Business Process Model and Notation (BPMN)

One of the most popular modeling languages for specifying business processes at the level of domain analysis and high-level system design is the Business Process Model and Notation (BPMN) language [OMG11a]. BPMN models are composed of activity nodes denoting business events or tasks performed by people or software applications within the business process, and control nodes capturing the flow of control between activity nodes.

There exist many approaches on the functional validation of BPMN models. What is mutual to the findings of all these approaches is the lack of precisely defined

semantics of BPMN which is a prerequisite for the behavioral interpretation of BPMN models necessary for realizing their validation.

Dijkman *et al.* [DDO08] present an approach for statically analyzing business process models defined with BPMN. In this approach mapping rules are defined for transforming BPMN models into Petri Net models. The authors identified several issues with the BPMN specification, such as imprecisely defined semantics of executing a business process with several start events and proper process instance completion. The obtained Petri Net model is used as input for the ProM tool to check (1) the absence of dead tasks (i.e., tasks that cannot be executed) and (2) the proper completion of the business process model (i.e., the state of a process model in which an end event is reached and no other task is enabled).

Another similar approach is described by Raedts *et al.* [RPU+07]. Their approach is based on translating a BPMN model into an Extended Petri Net model and then translating the Extended Petri Net model into mCRL2, a process algebraic language, after which the mCRL2 toolset can be used. Based on this approach, the modeled business process can be statically analyzed, simulated, or verified, e.g., for analyzing its performance (e.g., process execution time) and verifying soundness. A process model is considered sound if each state in the Petri Net can be reached at least once in one of the possible executions, the final state of the Petri Net is reachable from any state of the possible executions (deadlock free), and for each execution of the Petri Net the final state is reachable (proper process completion) [vdA98].

A different approach is presented by Ligeza *et al.* [LKNP12]. Their approach is based on leveraging a rule-based system which is implemented in Prolog and composed of business rules and BPMN models specifying a business process to analyze previously defined correctness criteria, as well as the correctness of data flows. These criteria include the correctness of process components (tasks), data flows, splits, merge nodes, and finally of the overall process.

2.2 UML Activity Diagrams

The Unified Modeling Language (UML) [OMG11b] is a popular modeling language for specifying structural and behavioral aspects of a software system. The structural aspects describe the objects that exist in the system, as well as relationships among these objects. Behavioral aspects define both the history of interactions between objects over time, as well as the communication of objects to accomplish certain goals. UML contains many types of diagrams, such as class, object, sequence, activity, and state machine diagram, that enable a user to comprehensively specify all necessary details of a system, both on a high and a low level of abstraction. An investigation of the expressiveness and adequacy of the UML activity diagram notation for specifying business processes is presented by Dumas and ter Hofstede [DtH01]. They conclude, that despite some issues with the semantics specification and few missing modeling concepts, UML activity diagrams are expressive enough to be used for specifying business processes.

One approach that enables to validate the functional correctness of UML activity diagrams is presented by Staines [Sta08]. In this approach UML activity diagrams are translated into Petri Net models, which can subsequently be translated into Colored Petri Net models for which already existing analysis tools can be used to check properties, such as soundness.

Engels *et al.* [ESW07] propose another approach for verifying the correctness of UML activity diagrams. Their approach is based on applying Dynamic Meta Modeling (DMM) for defining the behavioral semantics of activity diagrams based on graph transformations. They apply the existing model checker tool GROOVE to check the soundness of the analyzed activity diagram.

Eshuis and Wieringa [EW04] present a tool for verifying activity diagrams which is an extension of the graph editing tool TCM. Upon specifying functional requirements and an activity diagram, both are translated into a transition system according to their formal semantics specification. This transition system serves as input for an existing model checker. In case that the requirements are not satisfied, a counter example is provided in the form of an execution trace that lead to the requirement violation.

All of these approaches specify the semantics of UML activity diagrams on their own, e.g., by translating them into Petri Nets. However, the semantics of a subset of UML has recently been precisely defined and standardized by the fUML standard [OMG11c] which specifies a virtual machine for executing UML activity diagrams. Based on the fUML virtual machine we elaborated a testing framework in previous work [MLMK13], which will be described in more detail in Section 5.

2.3 Web Services and Business Process Execution Language (BPEL)

In order to achieve interoperability between applications based on Web standards (e.g., SOAP, WSDL, UDDI), Web services are commonly used as implementation technology as they enable a loosely coupled integration of heterogeneous systems.

In order to define and execute a complex business process based on Web services, the Business Process Execution Language (BPEL) can be used. It provides a model and a grammar based on XML for describing collaborating business processes through which a complex business process is realized. It entails interaction through Web service interfaces and a logic for coordinating these interactions.

Mayer and Lübke [ML06] propose an architecture and implementation of a unit testing framework for validating processes defined with BPEL. The architecture of the framework is composed of four layers: test specification (i.e., how the test data and its behavior are specified), test organization (tests are organized into test cases, test suites, and test runs), test execution (test and process under test must be executed), and test result layer (results must be gathered, logged, and presented to the user). Possibly incorrect data received from the process under test is categorized into three types: incorrect content, no message returned, or an incorrect number of received messages.

Li and Sun [LS06] propose another implementation of a unit testing framework for BPEL, called BPELUnit. The basic idea of the framework is to transform process interactions through Web service invocations to class collaborations using method calls, and to then apply object-oriented testing techniques. Firstly, interface descriptions of the Web services under test are transformed into corresponding Java interfaces upon which mock objects are created that represent partner processes that are directly invoked by the process under test. A mock object for a process defines expected invocations and return values of the process and validates that invocations are performed with correct parameters during runtime. The authors identified several advantages of their framework: the testing process does not rely on the availability of partner processes, test case writing is simplified, the test case execution time is improved, and automatic regression testing is enabled.

Weber *et al.* [WHM10] suggest that soundness of business process models is a necessary but insufficient condition for correctness, and that it might me necessary to take into account also the effects of executing individual activities within the process. Their approach includes the verification of effect conflicts (consistency of effects of parallel activities), precondition conflicts (consistency of preconditions of activities), reachability (are there activities that are not reachable within possible executions), and executability (are there activities whose preconditions are false at the time they need to be executed).

3 Validating Inter-Organizational Business Process Models

To maintain a certain level of competitiveness, collaboration between partner companies is essential. In this context, companies focus on their specialized functions for which they have expertise, complementing their business processes with partners and suppliers. In such an environment, companies rely on inter-organizational business processes to realize their business. Bouchbout and Alimazighi [BA11] define inter-organizational business process as an organized group of joined activities carried out by two or more organizations to achieve a common business objective. The design and implementation of inter-organizational business processes open up new challenges that need to be addressed: the flexibility and ability to cope with change, decentralized management, peer-to-peer interactions, preservation of enterprise autonomy and privacy, and support for interoperability.

Breu *et al.* [BDE+13] point out several important aspects of inter-organizational processes: *distribution* (either one party is serving as a controller of the complete process, or no party is in control of the whole process), *behavior* (different languages for describing choreographies exist), *data* (data can be distributed between collaborating parties), and *resources* (quality of service and service-level agreements are necessary for establishing trust between parties concerning provided resources). The authors identify four main challenges of inter-organizational processes: flexibility, correctness, traceability, and scalability. *Flexibility* is defined as the variability of participating intra-organizational processes, looseness of their interactions, adapta-

tion as capability to adapt the process to emerging events, and evolution as ability to change the process according to changed business requirements. *Correctness* is a prerequisite of both intra- and inter-organizational processes, as it ensures non-disrupted functioning of a business process. *Traceability* is defined as the need to monitor which progress is made at which point in time for which steps of the process. And finally, *scalability* is necessary for ensuring the reliability of the process execution as the computing resource requirements change over time.

In Section 2 we gave an overview of existing approaches for validating intra-organizational business process models. However, for validating inter-organizational business processes, their peculiarities compared to intra-organizational business processes have to be taken into consideration. In the following, we provide an overview of existing approaches for validating the functional correctness of inter-organizational business processes.

Van der Aalst [vdA98] presents an approach for analyzing and verifying inter-organizational workflows. Each workflow of each participating partner is modeled as a Petri Net model and additional modeling concepts are used for defining the connections between communicating workflows. For each individual Petri Net model describing an intra-organizational workflow, the concept of *soundness* can be verified by using standard Petri Net techniques. The communication between partner processes in an inter-organizational process can be either synchronous or asynchronous. Synchronous communication is modeled as direct connection between transitions of Petri Nets defining the participating partner processes. Asynchronous communication is modeled as a connecting place between transitions of the Petri Nets defining the participating partner processes. A Petri Net model describing an inter-organizational business process is considered sound if it is both *locally* sound (each Petri Net model of each participating partner is sound) and *globally* sound (the complete Petri Net model of the inter-organizational process is sound).

Martens *et al.* [MMGF06] present a similar approach for analyzing and verifying the *compatibility* of BPEL processes. They define two dimensions of compatibility: *syntactical* compatibility (two BPEL processes can be composed only if the provided interfaces of those two processes are mutually compatible) and *behavioral* compatibility (the behavior provided by two processes must be compatible). The approach is again based on Petri Net models. BPEL processes are translated into Petri Net model representations which are then combined into a single Petri Net model. Thereof the soundness is checked for the combined Petri Net model.

Similarly, Dumas *et al.* [DBN08] discuss the notion of protocol compatibility between Web services and review a number of techniques for detecting incompatibilities and for synthesizing adapters for otherwise incompatible services.

Benatallah *et al.* [BT04] define different types of protocol compatibility, corresponding to different levels of service interoperability. They show how given two services and their specifications it is possible to formally determine their level of compatibility. In addition, they present how to identify similarities and differences between protocols, in order to asses their equality and replaceability.

Intra-Organizational Business Processes					
Authors	**Technique**	**Static analysis**	**Simulation**	**Verification**	**Testing**
BPMN					
Dijkman *et al.* [DDO08]	Translation into Petri Nets	×			
Readts *et al.* [RPU+07]	Translation into Extended Petri Nets and mCRL2	×	×	×	
Ligeza *et al.* [LKNP12]	Rule-based system		×	×	
UML AD					
Staines [Sta08]	Translation into Colored Petri Nets	×	×	×	
Engels *et al.* [ESW07]	Graph transformations		×	×	
Eshuis and Wieringa [EW04]	Translation into transition systems			×	
BPEL					
Mayer and Lübke [ML06]	Unit test framework based on XML and Xpath				×
Li and Sun [LS06]	Unit testing framework based on JUnit				×
Weber *et al.* [WHM10]	Verification method exploiting semantic annotations			×	
Inter-Organizational Business Processes					
PetriNet					
Van der Aalst [vdA98]	Composition of Petri Nets	×		×	
BPEL					
Martens *et al.* [MMGF06]	Translation into annotated Petri Nets	×		×	

Table 1: Overview of approaches for validating business process models.

Another approach for ensuring the correctness of a business process in an inter-organizational context is to analyze its compliance with so-called *reference models*. Reference models provide a set of generally accepted, sound, and efficient processes and can help to speed up and optimize the design of business process models as well as ease the compliance with regulations and requirements. An approach for measuring the compliance of processes with reference models is described by Gerke *et al.* [GCC09]. In their approach they define compliance as the degree to which a process model behaves in accordance with a reference model. Compliance is measured by validating the process instances of a model (i.e., possible executions) against the reference model by determining whether they are valid instances of the reference model or their degree of deviation.

A similar approach is provided by Weidlich *et al.* [WMPW11]. In this approach the behavioral consistency between a business process model specifying the system and a workflow model as implementation of that system is analyzed. Behavioral consistency of the two models is defined in terms of a causal behavioral profile, which represents a behavioral abstraction that includes dependencies in terms of order, exclusiveness, and causality between pairs of activities in the models. Besides the order of potential occurrences, also optionality and causality are described. Optionality of a transition is defined as the existence of a firing sequence leading from the initial to the final marking of the system which does not contain the transition. Causality is defined as the restriction that the transition can occur in the firing sequence only after another given transition. One application for causal behavioral profiles is to check the conformance of process logs captured for the execution of a system to the modeled process. This conformance is measured as to what degree the behavior of the log is captured in the respective model.

4 Challenges of Validating Business Process Models

In Section 2 and Section 3 we presented existing approaches for validating the functional correctness of intra- and inter-organizational business processes. An overview of the discussed approaches is provided in Table 1. Most of these approaches focus on analyzing the soundness of the modeled processes and are restricted to a specific modeling language—most of the modeling languages have no standardized formal behavioral semantics defined. Furthermore, there are many challenges in validating inter-organizational business processes, such as compositions of process models where some depending processes of business partners might only be available through contracts while their internals are not visible. Another challenge is the need for flexibility in terms of enabling the evolution of business processes without central control. All this leads to some still open challenges in validating inter-organizational business processes.

Another important challenge in validating business processes is the *variety of available modeling languages* for defining business processes and the lack of standardized behavioral semantics of these languages. Having a common language to specify the behavioral semantics of modeling languages might lead to standardized methods and tools for validating the correctness of business process models regardless of the used modeling language.

In inter-organizational business processes, several intra-organizational business processes of collaborating business partners are combined in order to accomplish a common business goal. If all of these intra-organizational business processes are defined by business process models and the connection between those business process models are defined, it is possible to combine them into one process model defining the complete inter-organizational process. Hence, based on this *combined business process model*, it is possible to validate the correctness of the inter-organizational process using the approaches for validating business process models presented in Section 2. This approach was followed by van der Aalst [vdA98] and Martens *et al.* [MMGF06] as described in Section 3.

However, one important challenge that has to be addressed when dealing with inter-organizational business processes is flexibility in terms of the *variability of participating intra-organizational processes*. Thus, cooperating partners might be unknown in advance and might change during the execution of an inter-organizational business process. In such a context, it is not realistic to assume the existence of a business process model defining the complete inter-organizational business process. In order to enable the validation of the correctness of each participating intra-organizational business process in this situation, the technique of *mocking* can be applied. In the realm of Web service testing, there exists some work on the automatic or semi-automatic generation of so-called mock services that imitate the services provided by partner organizations. The work done by Li and Sun [LS06] presented earlier is an example for such an approach. Furthermore, *reference models* and *behavioral profiles* can help to validate the correctness of inter-organizational business processes in the situation where no business process model for the overall

process is available. In this context, validating the conformance of the business processes to mock objects or reference models respectively behavioral profiles is required. Work done in this area by Gerke *et al.* [GCC09] and Weidlich *et al.* [WMPW11] was described earlier. However, there is still an open issue of implementing these approaches in different modeling context.

The challenge of flexibility of inter-organizational business processes also includes the fact that *business processes can change* over time. *Regression testing* can be an essential technique for validating the correctness of inter-organizational processes in the event of changes. In software testing, regression testing is the process of selectively re-testing the software system to ensure that new bugs have been fixed and no other previously working functions have failed (regressed) as a result of changes to the system. With the application of test automation as provided by unit testing frameworks, regression testing is eased as it enables to specify tests and re-run them automatically. In Section 2 we have discussed two unit testing frameworks for business process models developed by Mayer and Lübke [ML06], and Li and Sun [LS06]. However, advanced techniques available for regression testing of software systems, such as automatically selecting the test cases that have to be re-run after a change, measuring the test coverage, and automatically generating additionally required test cases, have not been addressed so far by testing approaches for business processes.

5 Testing UML Activities: A Framework Based on fUML

In order to provide a precise and complete specification of the behavioral semantics of a subset of UML, the fUML standard [OMG11c] was developed by OMG. The subset of UML addressed by the fUML standard consists of modeling concepts for defining the structure of a system with UML classes and the behavior of a system with UML activities. For defining activities, the fUML subset contains a set of predefined actions which can be used for defining object manipulations (objects and links between these objects can be created, destroyed, and modified) and communications between activities (activities can call other activities synchronously or asynchronously). Furthermore, modeling concepts for defining the flow of control and the flow of data between actions and activities are included in the fUML subset. The fUML standard defines a virtual machine that is capable of executing models compliant to this subset. This precise and complete semantics specification of UML activities enables the development of validation and verification methods and tools for business processes which are defined by UML activity diagrams.

In recent work [MLMK13], we elaborated a testing framework for validating the functional correctness of UML activity diagrams based on the virtual machine provided by the fUML standard[1]. An overview of our testing framework is depicted

[1]We provide an implementation of our framework integrated with the Eclipse Modeling Framework. For more information about the implementation, we refer the interested reader also to our project website:

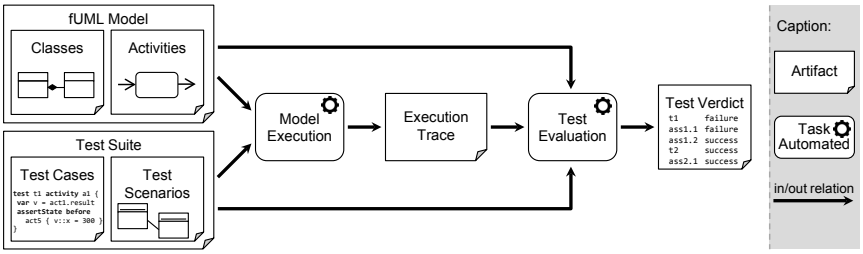

Figure 1: Testing framework for UML activity diagrams based on fUML.

in Figure 1. It consists of a *test specification language* and a *test interpreter*. The test specification language enables to define *test suites* composed of *test cases* defining assertions on the behavior of UML activities as well as *test scenarios* defining input values for the execution of the UML activities under test. Each test case in the test suite is evaluated by executing the activity under test with the input values defined by a selected test scenario using the fUML virtual machine. From the execution of the activity under test, an execution trace is obtained from the fUML virtual machine, which captures the runtime behavior of the executed activity. Based on this captured information, the assertions on the behavior of the activity defined by the test case are evaluated and the test verdict is calculated.

We support two kinds of assertions that can be defined on the behavior of a UML activity: *execution order assertions* for validating the execution order of nodes contained by the activity and *state assertions* for validating the state of objects modified by the activity.

Execution order assertions are specified by defining the order in which nodes contained by the activity under test have to be executed. Furthermore, the testing framework supports the assertion of the expected execution order of selected activity nodes only. Validating the correct execution order of nodes contained by an activity is an important capability when they are used to define the steps of business processes. However, this is not considered by the approaches discussed in Section 2. These approaches focus on the analysis of soundness properties which could also be done using our testing approach. In this respect, we are currently working on extending our testing framework to support parallelism in activity diagrams. For this, execution paths of an activity relevant to the specified execution order may be computed and can be checked against the specified order assertions. Furthermore, by analyzing the activity and the calculated execution paths, we could for instance identify dead tasks and validate the proper completion of an activity for a given input.

State assertions can be used to validate the state of an object at a specific point in time of the execution of the activity under test. To express the point in time at which the object shall be validated, a temporal quantifier, a temporal operator,

`http://www.modelexecution.org`.

and a reference activity node can be defined. Combining the temporal quantifier and the temporal operator it is possible to specify which states of the object should be checked after or before the execution of the reference activity node. A state assertion can contain several state expressions, each defining the expected value of one of the checked object's features. State assertions enable not only to validate the output of an activity execution, but also its intermediate results. This capability is usually not provided by unit testing frameworks, as they only enable to validate input-output relations.

Our framework enables to specify repeatable tests that can be automatically executed. Hence, it provides the necessary foundation for regression testing, which was identified as an enabling technique for supporting the evolution of inter-organizational business processes. However, it still remains to investigate techniques for test selection, measurement of test coverage, and the automated generation of test cases.

In order to support testing of inter-organizational business processes in the situation where each business partner has only access to his own intra-organizational business process, mocking the required partner processes would be valuable. In such case it is necessary to specify and analyze the correspondence of the partner process implementations to mock activities. For this, the already described approach by Weidlich *et al.* [WMPW11] could be used. We intend to investigate the possible application of this approach to UML activity diagrams.

6 Conclusion

As business process models become the main artifacts in business process management and information system development, validating their functional correctness becomes essential. In this paper we have presented several approaches for validating business process models in both intra- and inter-organizational context.

Most of these approaches are based on translating business process models into another formalism, such as Petri Nets, upon which standardized tools for analysis, simulation, and verification already exist. The reason for this is that the used modeling languages lack in providing formal execution semantics. Furthermore, most of the approaches concentrate on finding parts of a model that cannot be executed, as well as validating the proper execution completion of a model.

In the inter-organizational context, several business processes of collaborating partners are combined. Due to the needed flexibility of inter-organizational business processes, each partner might only have access to those parts of the process that they provide. In this situation it might be necessary to apply the techniques of mocking, where the processes provided by other partners are imitated, to enable the validation of the inter-organizational business process. The automatic or semi-automatic generation of such mocks might provide huge benefits. However, specifying behavioral contracts between business partners and checking the conformance of

implementations to those contracts are required.

The need for enabling the evolution of business processes constitutes another challenge in validating inter-organizational business processes which can be addressed by regression testing. However, advanced techniques, such as automated test case selection and generation do not exist so far.

In previous work we elaborated a framework for testing UML activity diagrams based on the fUML standard. Based on the identified challenges in validating business processes we are considering further extensions of our framework.

References

[BA11] K. Bouchbout and Z. Alimazighi. Inter-Organizational Business Processes Modelling Framework. In *Proc. of 15th East-European Conference on Advances in Databases and Information Systems (ADBIS'11)*, pages 45–54. CEUR-WS.org, 2011.

[BDE+13] R. Breu, S. Dustdar, J. Eder, C. Huemer, G. Kappel, J. Köpke, P. Langer, J. Mangler, J. Mendling, G. Neumann, S. Rinderle-Ma, S. Schulte, S. Sobering, and B. Weber. Towards Living Inter-organizational Processes. In *Proc. of 15th IEEE Conference on Business Informatics (CBI'13)*, pages 363–366. IEEE, 2013.

[BT04] B. Benatallah and F. Toumani. Analysis and Management of Web Service Protocols. In *Proc. of 23rd International Conference on Conceptual Modeling (ER'04)*, volume 3288 of *LNCS*, pages 524–541. Springer, 2004.

[DBN08] M. Dumas, B. Benatallah, and H. R. M. Nezhad. Web Service Protocols: Compatibility and Adaptation. *IEEE Data Engineering Bulletin*, 31(3):40–44, 2008.

[DDO08] R. M. Dijkman, M. Dumas, and C. Ouyang. Semantics and analysis of business process models in BPMN. *Information & Software Technology*, 50(12):1281–1294, 2008.

[DtH01] M. Dumas and A. H. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In *Proc. of 4th International Conference on the Unified Modeling Language (UML'01)*, volume 2185 of *LNCS*, pages 76–90. Springer, 2001.

[ESW07] G. Engels, C. Soltenborn, and H. Wehrheim. Analysis of UML Activities Using Dynamic Meta Modeling. In *Proc. of 9th International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'07)*, volume 4468 of *LNCS*, pages 76–90. Springer, 2007.

[EW04] R. Eshuis and R. Wieringa. Tool Support for Verifying UML Activity Diagrams. *IEEE Transactions on Software Engineering*, 30(7):437–447, 2004.

[GCC09] K. Gerke, J. Cardoso, and A. Claus. Measuring the Compliance of Processes with Reference Models. In *Proc. of Confederated International Conference On the Move to Meaningful Internet Systems (OTM'09)*, volume 5870 of *LNCS*, pages 76–93. Springer, 2009.

[LKNP12] A. Ligeza, K. Kluza, G. J. Nalepa, and T. Potempa. AI Approach to Formal Analysis of BPMN Models. Towards a Logical Model for BPMN Diagrams. In *Proc. of Federated Conference on Computer Science and Information Systems (FedCSIS'12)*, pages 931–934, 2012.

[LS06] Z. J. Li and W. Sun. BPEL-Unit: JUnit for BPEL Processes. In *Proc. of 4th International Conference on Service-Oriented Computing (ICSOC'06)*, volume 4294 of *LNCS*, pages 415–426. Springer, 2006.

[ML06] P. Mayer and D. Lübke. Towards a BPEL unit testing framework. In *Proc. of 2006 Workshop on Testing, Analysis, and Verification of Web Services and Applications (TAV-WEB'06)*, pages 33–42. ACM, 2006.

[MLMK13] S. Mijatov, P. Langer, T. Mayerhofer, and G. Kappel. A Framework for Testing UML Activities Based on fUML. In *Proc. of 10th International Workshop on Model Driven Enginering, Verification and Validation (MoDeVVa'13)*, pages 1–10. CEUR-WS.org, 2013.

[MMGF06] A. Martens, S. Moser, A. Gerhardt, and K. Funk. Analyzing Compatibility of BPEL Processes. In *Proc. of Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW'06)*, pages 147–154. IEEE, 2006.

[OAS07] OASIS. Web Services Business Process Execution Language (WS-BPEL), Version 2.0, April 2007. Available at: `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[OMG11a] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011. Available at: `http://www.omg.org/spec/BPMN/2.0/`.

[OMG11b] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011. Available at: `http://www.omg.org/spec/UML/2.4.1`.

[OMG11c] OMG. Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.0, February 2011. Available at: `http://www.omg.org/spec/FUML/1.0`.

[RPU+07] I. Raedts, M. Petkovic, Y. S. Usenko, J. M. E. M. van der Werf, J. F. Groote, and L. J. Somers. Transformation of BPMN Models for Behaviour Analysis. In *Proc. of 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS'07)*, pages 126–137. INSTICC Press, 2007.

[Sta08] T. S. Staines. Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets. In *Proc. of 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'08)*, pages 191–200. IEEE, 2008.

[vdA98] W. M. P. van der Aalst. Modeling and Analyzing Interorganizational Workflows. In *Proc. of 1st International Conference on Application of Concurrency to System Design (ACSD'98)*, pages 262–272. IEEE, 1998.

[WHM10] I. Weber, J. Hoffmann, and J. Mendling. Beyond soundness: on the verification of semantic business process models. *Distributed and Parallel Databases*, 27(3):271–343, 2010.

[WMPW11] M. Weidlich, J. Mendling, A. Polyvyanyy, and M. Weske. Causal Behavioral Profiles - Efficient Computation, Applications, and Evaluation. *Fundamenta Informaticae*, 113(3-4):1009–1025, 2011.

Author Index