2007

Simulating Business Process Scenarios for Event-Based Systems

J Schiefer
js@ifs.tuwien.ac.at

H. Roth
roth@securityresearch.at

M. Suntinger
msuntinger@senactive.com

A. Schatten
aschatt@ifs.tuwien.ac.at

SIMULATING BUSINESS PROCESS SCENARIOS FOR EVENT-BASED SYSTEMS

Schiefer, Josef, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Favoritenstrasse 9-11/188, 1040 Vienna, Austria, js@ifs.tuwien.ac.at

Roth, Heinz, Secure Business Austria Competence Center, Favoritenstraße 16, 1040 Vienna, Austria, roth@securityresearch.at

Suntinger, Martin, Senactive IT-Dienstleistungs GmbH, Phorusgasse 8, 1040 Vienna, Austria, msuntinger@senactive.com

Schatten, Alexander, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Favoritenstrasse 9-11/188, 1040 Vienna, Austria, aschatt@ifs.tuwien.ac.at

Abstract

Today's networked business environment requires systems which are adaptive and easy to integrate. Event-based systems have been developed and used to control business processes with loosely coupled systems. Research and product development focused so far on efficiency issues, but neglected simulation support to build robust and efficient event-driven applications. In this paper, we propose a simulation model that allows imitating real-world operations of business processes in order to improve efficiency and effectiveness of event-based systems. Our approach uses discrete event-simulation and a graphical model for defining event sequences for business process scenarios. For better handling the complexity and variability of business processes, we use a hybrid simulation approach, which is able to combine various ways to compose event sequences and generate representative event data. As an example, we show how annotated WS-BPEL process descriptions can be used to automatically generate event sequences representing typical process execution paths for simulation purposes.

Keywords: Business Process Simulation, Event-Based Systems, Simulation Models

1 INTRODUCTION

Business process solutions form the backbone of many modern enterprises, linking vital systems and business processes with real-time data interchange. Any shortcoming or outage in the execution of a business process solution is likely to affect large portions of the enterprise, potentially causing loss of revenue and data.

The lack of simulation data for business process management (BPM) systems makes it difficult for organizations to evaluate, test, or benchmark the execution environment for business processes. However, the generation of representative simulation scenarios and simulation data for business processes is a challenging and complex task. Difficulties often arise in producing familiar, complete and consistent simulation data on any scale. Producing simulation data manually often causes problems, which can be avoided by automatically generating consistent and statistically plausible data.

In this paper, we show an approach for simulating business processes for event-based systems. Event-based systems are widely used for integrating loosely coupled application components, including sensors, controllers, and databases. Since business processes are becoming more interconnected and

event-driven, event-based systems fit well for supporting and monitoring business processes. Event-based systems are highly composable systems that often provide a close mapping to real-life events. However, all these benefits come at a price. Systems which do not use a synchronous communication style in favor of loosely structured interaction are inherently more difficult to design and manage. Therefore, one should employ management and visualization tools to create a system that is highly dynamic but remains manageable at the same time.

In this paper, we show an approach for simulating business processes for event-based systems, more precisely, their outcome in terms of *business process events*. Since many operations of businesses are interconnected and subject to variability, interconnectedness and complexity (combinatorial and dynamic) it is difficult, if not impossible, to predict the behavior and performance of complex business operations. A simulation model is able to explicitly represent selected aspects that can be used to compare system designs and to determine the effect of alternative policies on the behavior and performance of event-based systems.

The remainder of the paper is organized as follows. In Section 2, we reflect on related work on simulation approaches for business processes and discuss our contribution. In Sections 3 and 4, we define requirements and describe the process for simulating process events. Section 5 and 6 discuss the simulation model in detail. In Section 7, we describe how to automatically extract event sequences from WS-BPEL processes and show a complete example in Section 8. Finally, in Section 9, we conclude our paper and provide an outlook for future work.

2 RELATED WORK AND CONTRIBUTION

Derrick et al. (1989) gave an overview of existing conceptual frameworks for simulation modeling and compared different approaches they pursue. When describing classical frameworks for discrete event-simulation, it can generally be distinguished between event scheduling, activity scanning and process interaction approaches, while others use the system theoretic approach or condition specification. For a detailed discussion and comparison of these approaches, please refer to Pidd (1998).

According to Balci (1988), a (simulated) system can be described in terms of objects (entities), attributes defining these objects, events causing changes in object states, activities that transform an object's state over time and processes that are a sequence of activities or events ordered by time.

Although only recently event-based concepts like event driven architectures have significantly gained popularity, events have played a crucial role in simulation modeling for a long time (Derrick, Balci and Nance, 1989). The event-scheduling approach uses an ordered set of timed and so-called *determined* events on the one hand and, on the other hand, it includes condition checks for other so called *contingent* events.

Activity scanning uses a more modularized approach. The different objects to be simulated and the actions/activities they perform have to be described together with conditions which define under what circumstances these actions are executed.

An enhancement to these two approaches is the three-phase approach which combines their time-based and state-based natures (Balci, 1988). The two types of events used here are called B (bound or booked) events and C (conditional) events (Robinson, 2003). B-events stand for state changes scheduled at a specific point in time, while C-events depend upon the satisfaction of a condition. Each C-event is evaluated after the B-event list has been completed and again after all matching C-events have been executed until no C-event's condition matches anymore. In that case, the simulation advances to the next point in time where B-events are determined to be emitted.

The process-interaction approach focuses on how entities flow through the system. While a process is executed, it can be suspended, which means that either a time-based or state-based delay occurs. Time-

based delays are determined in that they endure for a certain amount of time or until a specific point in time is reached, while state-based delays wait until a given condition is satisfied. So-called dynamic objects are inserted into the process and subjected to these delays as they traverse the process path.

The Entity-Relationship-Attribute approach is a more data-driven approach. The conceptual work necessary to set up the relations in a database helps the modeler to work towards a correct representation of the simulated system. Functions are then used to change the system from state to state, but as Derrick et al. (1989) point out, dynamics are difficult to represent using this approach.

Research on discrete event-simulation mostly dates back to the eighties of the last century and to our knowledge no major progress has been made lately. However, the current emergence of event-driven architectures (EDA) for business applications continues these efforts.

Since our approach solely uses an event-based world view (Lackner, 1962), at first sight, it may seem that our approach relates mainly to event-scheduling but, as we will show, various aspects of the other approaches are reflected as well.

The explicit modeling of event sequences in our simulation model distinguishes our approach from existing discrete event simulation approaches. Our model supports the discovery of event sequences from existing business process models, such as WS-BPEL, which can be automatically populated by the simulator with artificially generated data as well as with data from existing data sources.

A major issue of discrete event-simulation is the definition of causal dependencies between the data items of various events. A simple example would be an order process whose events always have a process instance ID and an assigned account manager for the order processing. In this example, the events for an instance of the order process share the same process instance ID and the same account manager. These values cannot just be generated randomly due to causal dependencies. As we will show later in this paper, we use correlation sets as proposed in McGregor and Schiefer (2003) for modeling such causal dependencies.

With our simulation model, we define event sequences for analyzing and testing so-called *sense and respond loops* which are managed by an event-based system. Sense and respond loops allow event-driven control of business processes. During the processing of sense and respond loops, business information is continuously generated and decisions are made from which a response follows. The response has an effect on the source systems (from which the event-based system originally might have received events) and, consequently, also on the performance and the success of the organization. With each cycle of the sense and respond loop, the system evaluates the received events and reports on the performance of the business process. For more details on sense and respond loops, please refer to Schiefer and Seufert (2005).

3 SIMULATING BUSINESS PROCESS EVENTS

3.1 Rationale for Simulating Business Process Events

Many existing BPM tools provide support for the simulation of business processes and assess performance issues by varying rates of simulation input. Typically, statistics on resource utilization are provided as well as support for animation to visualize how work moves through the model. In addition, BPM tools often offer facilities to identify blockages or bottlenecks in the process definition. A traditional process simulation is primarily used for evaluating the model of a single, self-contained business process (either new or redesigned) before implementing it (which is still very viable).

Typical business process solutions form a network of IT applications which reside inside an organization as well as at sites of business partners, making a simulation more difficult. In particular,

the importance of the latter case is increasing with the appearance of standards for communication protocols (e.g. web services) and trading agreements (e.g. ebXML). By creating business networks, the complexity of business process solutions also rises significantly. In many cases, it is not conceivable anymore for an organization to be in full control of the business process when it is shared among business partners. In order to simulate or test processes in such a networked environment, the requests or responses of business partners need to be mocked appropriately with consistent data sets.

This extends the scope of traditional BPM tools for simulations as described above. With our approach, we are able to model the data exchange of process participants with event sequences which define typical interaction scenarios between different business processes. We extend the application scope of the process simulation by using the simulation results for analyzing and improving the event processing of event-based systems. In a following step, the interaction scenarios can be linked with each other and combined with generated or available data (such as order or production data) from an organization.

3.2 Simulation Modeling Process

The process of creating an event-driven simulation starts with the event-based system itself: A detailed analysis of the application which is to be evaluated forms the foundation for the definition of simulation requirements and application benchmarks, making it possible to not only emphasize the application's key features, but also to cover special cases throughout the simulation process. Once the simulation requirements are defined, an iterative simulation process, running through several stages from the creation and improvement of suitable simulation models to the generation and processing of simulation events, can be started. However, ideally both the simulation model and the event-based system will evolve with each cycle. At the end of each iteration, the simulation run is evaluated: If errors occurred or it is recognized that the simulation events do not reflect the conventional input data of the event-based system, the simulation model can be altered. In addition, the process behavior of the event-based system can be traced and benchmarked to detect problems and malfunctions. The simulation process continues until satisfactory process behavior is achieved. Hence, the number of iterations is not fixed, and it strongly depends on the complexity of the event-based system.

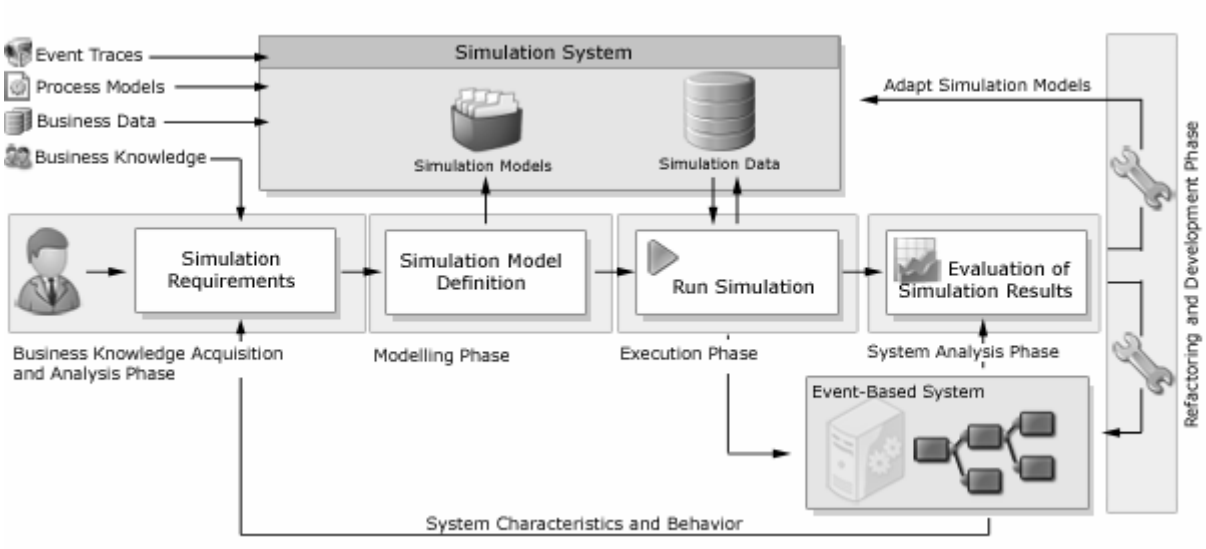

Figure 1: Event-driven simulation process.

Typically, during the first iterations, the user generates small event sequences, and after assuring that all requirements are reasonably fulfilled, the user can continue with simulation runs that combine arbitrary event sequences that have already been tested. By using 'divide and conquer', the complexity of a complete simulation scenario can be decomposed to modular and simple event sequences, while keeping the ability to independently implement and execute each of these subsequences.

A simulation process includes the following five phases: 1) business knowledge acquisition and analysis phase, 2) modeling phase, 3) execution phase, 4) system analysis phase, and 5) refactoring and development phase. Figure 1 provides an overview of the simulation process, the phases and the involved tasks.

During the business knowledge acquisition and analysis phase, all information regarding the event-based system relevant to the simulation process is captured, sample processes and possible exceptional situations are discovered, and existing data is explored and prepared. In addition, this phase may include an empirical study of business characteristics and expected process behavior.

In the modeling phase, the simulation scenarios defined during phase (1) are transposed to a simulation model that can be executed by the simulator. This transposition includes the modeling of accurate event sequences that represent the various simulation scenarios as well as the implementation of event consumers to transfer simulation events to the event-based system. For the simulation events to be coherent and meaningful, it is crucial to identify and model correlations and value sequences. At the end of the modeling phase, simulation scenarios are represented as executable simulation runs that assemble all the necessary operations to generate representative sequences of events and describe how to publish these events to be processed by the event-based system.

The execution phase starts with generating consistent mass event data according to the simulation model created during phase (2). The simulator executes a simulation run by generating all events of the defined sequences and forwarding them to the event consumers to be transferred to the event-based system and processed. The phase (3) is finished as soon as all simulation events have been processed by the event-based system.

During the system analysis phase (4), the performance and behavior of the event-based system is evaluated by tracing processing steps, analyzing automated decisions, calculating performance metrics and checking the processed input data for completeness. The insights gained during this phase serve as the basis for the following refactoring and development phase (5), which consists of adapting the simulation model or improving the event-based system.

4 SIMULATION MODEL

Our simulation model consists of a set of artifacts that can be used to generate simulation data for the behavior of business processes. Its goal is to imitate real-world scenarios for event-based systems, which are also referred to in the simulation model as the simulation targets. Thus, a simulation model specifies not only the sequence of events to be generated and how to fill these events with meaningful data, it also determines where the simulated events should be delivered to.

Figure 2 shows an order process which we assume is executed using a BPM system. For this example, we further assume that the BPM system generates events during process execution which are processed by an event-based system for monitoring purposes. Besides describing how activities are orchestrated using parallel process flows or flows depending on specified conditions, further information such as existing process entry points is available. When constructing event sequences for a business process, this information is very crucial, since it describes the various possible paths through a process. In Figure 2, you can see at the bottom of the picture a typical event sequence for a process path.

In the following, we will show a simulation model for event sequences representing process execution paths. During the simulation, the event sequences are instantiated and filled with meaningful and consistent data. Meaningful in the sense that the event data is realistic and representative, and correct in the sense that event sequences reflect the causal dependencies within the event data (e.g. order requests are related to shipment and billing requests by sharing specific identifiers).

There are two major steps involved in creating a simulation model for a business process: 1) defining relevant event sequences for process paths, and 2) populating the elements of an event sequence with representative and consistent data. Performing these two steps manually can be very time consuming and error prone. The goal of our simulation model is to automate and facilitate these tasks.

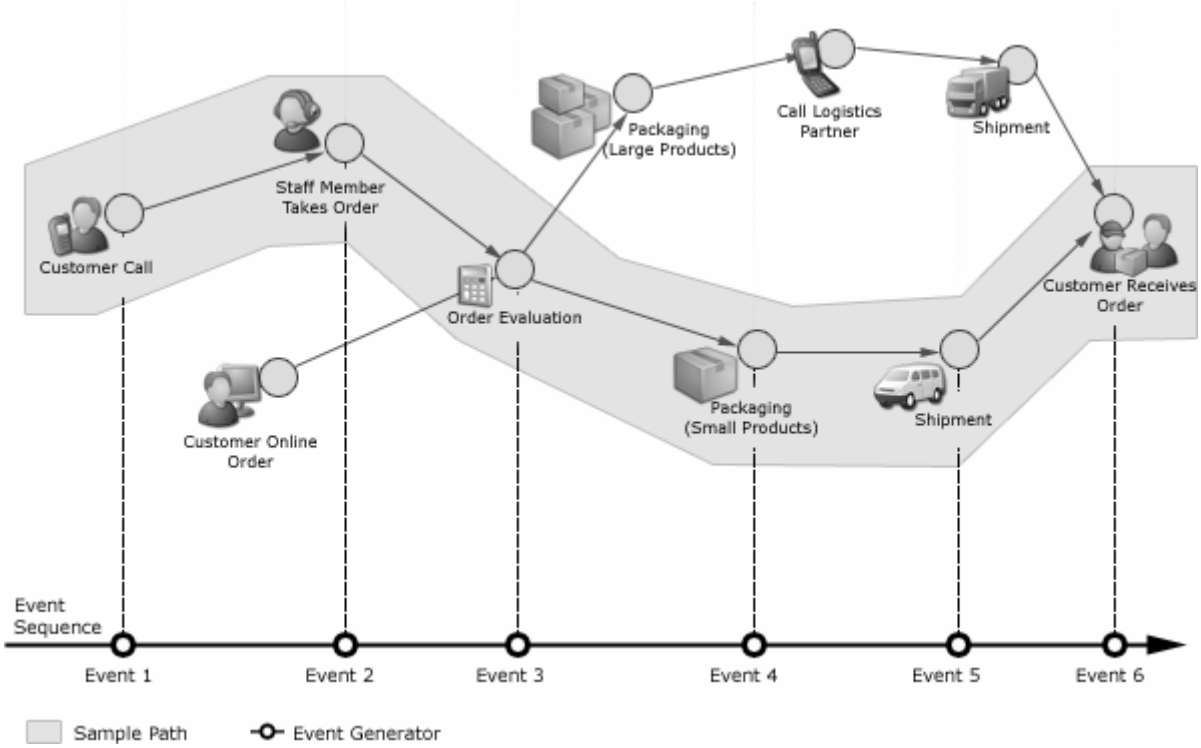

Figure 2: Execution path of an order process.

4.1 Simulation Meta-Model

Figure 3 shows the meta-model for our simulation approach. The key element of the simulation meta-model is the event generator, which is used by the simulator to generate instances of an event during a simulation run. The event generator also produces events of a certain event type und delegates the generation of event attribute values to value generators. Event sequences define a series of events, representing complex business scenarios. Each element of the event sequence has an assigned event generator which is used by the simulator to instantiate the event. An event sequence can include event traces representing segments of the sequence.

The events of an event sequence may have causal dependencies that are specified using correlation sets. Event sequences can further have variables which can be initialized by a simulation run and accessed from event generators. Event sequences can change the value for simulation variables during the simulation run. Event consumers are recipients of the events generated by the simulator and act as a mediator for forwarding the events to the target system, such as an event-based monitoring system.

A simplified usage scenario could be described as this: an event sequence defines a series of event generators and includes an event trace for imitating a specific business scenario. The event generators

make use of value generators and simulation variables to provide meaningful content for the events part of the event sequence. Finally, a simulation run defines the configuration for a simulator allowing event consumers to receive events that are relevant for a target system.

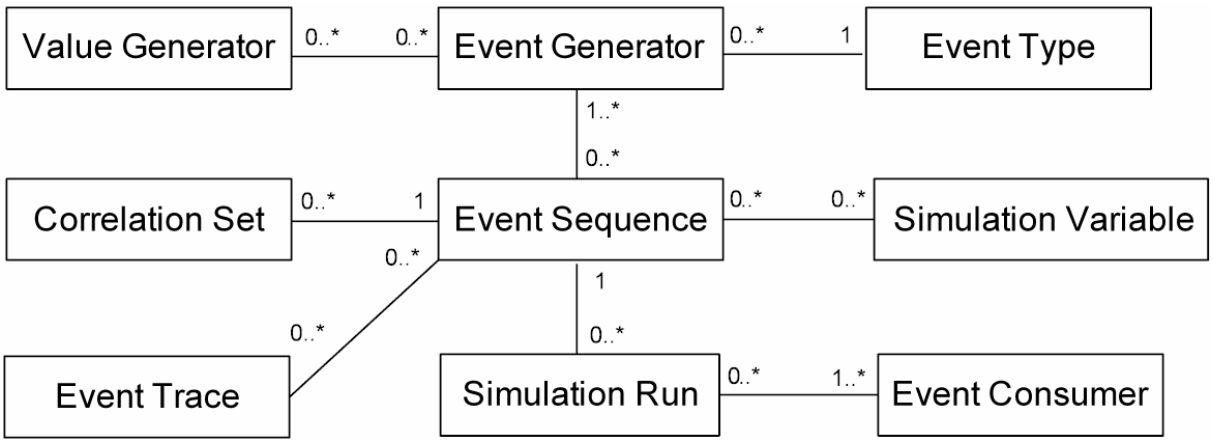

Figure 3: Simulation Meta Model

For event sequences of business process scenarios, complex arrangements of simulation events can occur, such as conditional or parallel flows. Since such event sequence definitions can become very complex and can have causal dependencies among the events, we developed a graphical model for defining complex event sequences. Figure 4 shows an example of an event sequence for the order process. For this event sequence, we assume that a typical order process requires between 1 and 3 transports for the delivery. Every event of the event sequence has an associated event generator. We decided to display the associated event type of the event generator in the event sequence because we believe it is more intuitive for the modeler. For this example, we modeled a sub-event sequence for a transport which is instantiated 1 to 3 times by the event sequence of the order process. Furthermore, the event sequences have defined two causal dependencies between events. The events of the event sequences for the order process share the same customer ID, and the events of the transport event sequence share the same transport ID.

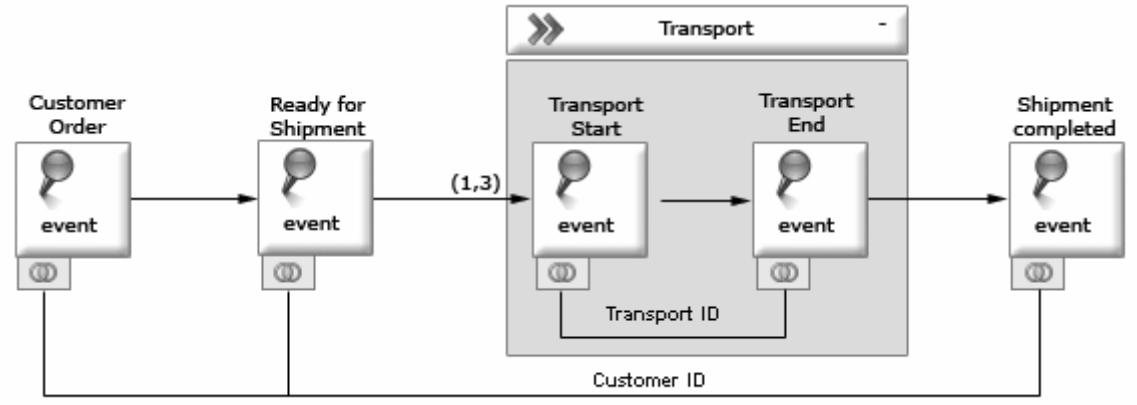

Figure 4: Graphical Model for Event Sequences

5 EVENT SEQUENCES FOR PROCESS MODELS

Although our graphical model can significantly reduce the time for building new event sequences, it suffers from the following two shortcomings:

1. The flows defined in a process model can be very complex, and it is difficult to construct representative event sequences.
2. If a process model changes, the event sequences of the simulation model also have to be updated to reflect the process change.

This section explains how our simulation model can be used to automatically extract event sequences from process models. If a business process is already formally defined (e.g. using WS-BPEL), its definition can be used to automatically discover all potential event sequences that can occur during process execution. For generating event sequences, it is necessary to point out the artifacts (e.g. before or after an activity) within the process model where event generators should be called. A simulator can make use of the process model and find possible process paths within the model. From these process paths, the simulator can automatically generate event sequences if the process paths have been annotated with event generators. In the following, we will explain this approach in detail.

5.1 Introducing Simulation Aspects as Orthogonal Concerns into WS-BPEL Processes

Having already defined a WS-BPEL process, it can be very useful—e.g., for testing or monitoring purposes—to simulate this process, meaning to calculate all possible paths through the WS-BPEL process, and then to use an event generator to create appropriate event streams. This can be a valuable strategy to analyze whether an event-based system can deal with the business process events as planned.

But the WS-BPEL definition alone is not sufficient for simulation purposes. Additional simulation meta-data is required which associates the process artifacts (e.g. activities) with the artifacts of the simulation model (e.g. event generators). This addition to a WS-BPEL process model can be seen as an orthogonal or crosscutting concern, also called aspect, in this context (Kiczales and Mezini, 2005).

In the previous section the meta-model for our simulation approach has been described. In the following, we discuss how simulation concerns (e.g. path probabilities have to be provided for branches in the process) can be added to an existing WS-BPEL process in order to generate event sequences. Generally speaking, several options are available to provide the necessary meta-data:

1. The meta-information can be stored in external configuration (files) and the simulator uses the configuration with the simulation model for generating event sequences. This approach has the advantage of being non-invasive; however, we also face similar problems as deployment descriptors in J2EE or mapping information in O/R tools. The actual process and the meta-information have to be kept in sync, which can be awkward in many practical cases.
2. The second option is to extend the WS-BPEL standard by using extension points. Using the <extensions> element an extension namespace can be defined. As our proposed extensions are not critical to the business process, they can be declared as "mustUnderstand='no'", hence the WS-BPEL processor is allowed to ignore them. Then attributes or elements in this namespace can be used to annotate respective WS-BPEL elements to add the meta-information to the specific aspect. This type of annotation is similar to annotations in modern programming languages like Java 5, where such annotations are used to define aspects or O/R mapping declarations.

From the practical point of view, these annotations should be created by the user interface for modeling the process and interpreted by the simulation extension to create the simulation artifacts. Our proposed approach for simulating WS-BPEL processes is the first step towards enabling business processes with standard-based simulation mechanisms. The ultimate goal of our work is to use the experiences we gained with building simulation tools designed for WS-BPEL processes for proposing simulation extensions of the WS-BPEL process standard.

5.2 Annotating WS-BPEL Processes for Generating Event Sequences

In the following, we want to outline how to automatically generate event sequences from the sample process shown in Figure 2 with the assumption that the process was modeled using WS-BPEL. We decided to use this process, since it has two special characteristics — it has two entry points and it includes a decision point. For illustrating our approach, we will use the decision point as an example for annotating an activity of a WS-BPEL process.

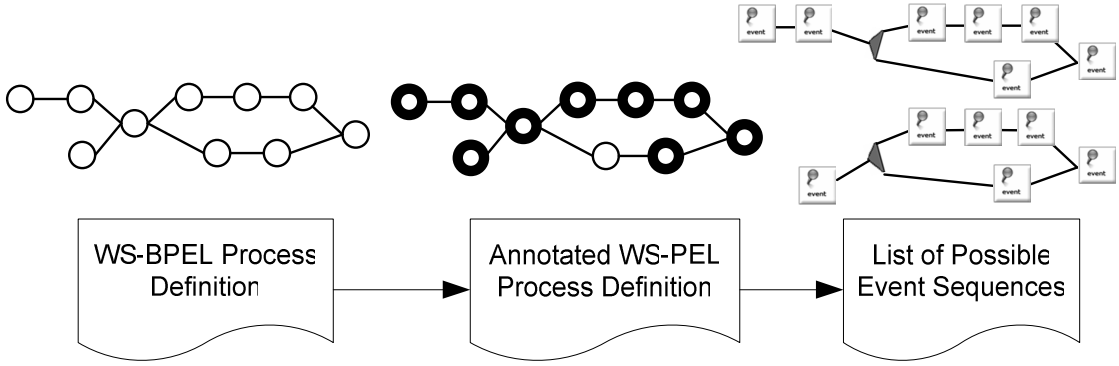

Figure 5: Annotation of a WS-BPEL process and the result event sequences.

Since the event sequences of our simulation model also support conditional elements, the *switch* activity, deciding if a small or a large package is at hand, can be modeled using one event sequence. Nevertheless, the two entry points to the process, namely, online order or call, result in at least two event sequences (see Figure 5). Figure 6 shows one of event sequences in the case where a customer called for placing the order, whereby we expect that 40 % of the orders are large products and 60 % are small products.

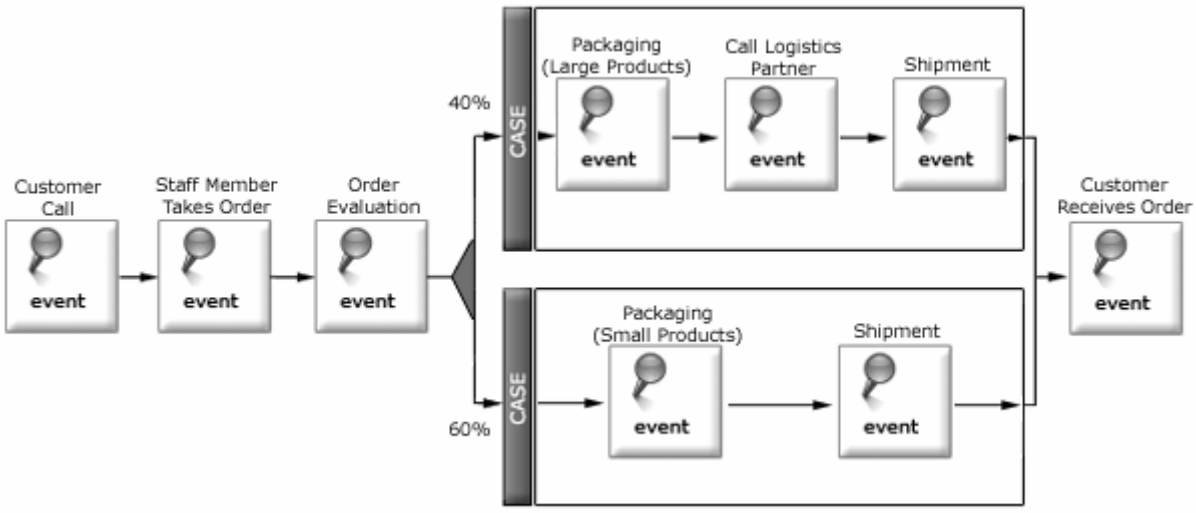

Figure 6: Event Sequence Example – Order Process

The desired result is a set of two potential event sequences, including suitable event generators that are called during a simulation run. We solved this problem by annotating a given WS-BPEL process definition with additional information that is used by the simulator, such as references to event generators. Listing 1 shows an example for an annotated *receive* activity. In this example, we assume that the WS-BPEL process uses a simulation extension for the namespace “sim”.

```
<receive name="customerCall" partnerLink="…>
        <sim:eventGenerator sim:id="OrderCreated" sim:variable= "VarOrderCreated"/>
</receive>
```

Listing 1: Annotating a receive activity.

Please note that the event generator *OrderCreated* is defined in the simulation model and only referenced in the WS-BPEL process. Furthermore, the simulation variable *VarOrderCreated* is declared for later use. When an activity is part of an event sequence, the referenced event generator is a placeholder for the activity within the event sequence to simulate the activity.

In order to capture the event sequence structure of a WS-BPEL *switch* statement, each condition has to be evaluated in the context of the simulation model. As is shown in Listing 2, previously assigned simulation variables can be used within such a condition, e.g. the simulation variable declared in Listing 1. In other words, in this example the simulation variable *VarOrderCreated* is used to hold a previously generated event whose data can be accessed later within case conditions. With this annotated meta-data, the simulator has sufficient information for generating an event sequence for a process path without requiring a process engine or process instantiation.

```
<switch name="customerCall" …>
  <case condition="…">
    <sim:case sim:condition="VarOrderCreated.OrderSize='Large'"/>
      …
  </case>
  …
</receive>
```

Listing 2: Annotation of a switch activity.

Another interesting facet of WS-BPEL processes when deriving event sequences are *compensate* and *throw* activities. Whenever such an activity is encountered, all the annotated activities of correspondent fault and compensation handlers will be appended to the specific event sequence in the same order as a WS-BPEL engine would traverse them.

The WS-BPEL process, as well as each scope of it, can be associated with a set of event handlers that are invoked concurrently if the corresponding event occurs. Event handlers can be triggered at anytime during the execution of a WS-BPEL process and their occurrence cannot be foreseen by analyzing its progress. Therefore, event handlers are represented as parallel event sequences, whereby the sequence for the event handler processing only occurs with a certain probability.

6 COMPLETE SIMULATION EXAMPLE

In this section, we show a complete example for the order process mentioned in the previous sections. For this example, we assume that a process path should be simulated when a customer places an order with a phone call and the placed order includes only small products. We have shown how a WS-BPEL process can be annotated with references of event generators for building event sequences. Next, we will illustrate how various types of event generators can be used to deliver representative event data for the order process.

For the simulation, we assume two external input sources for the simulator: 1) Existing logs and traces from a call center for customer phone calls and placed orders and 2) data warehouse data for the transportation of the products and handoff to the customer. The biggest advantage of including external data in the simulation process is that the captured event data represents real-world scenarios with all known and also unknown facets of a business process. One of the challenges when including external event data is that it is difficult to replicate in order to generate large data sets (e.g. the call

center captured a trace of 1000 phone calls and a simulation requires 10000 phone calls). In our simulation model, we use a technique called bootstrapping that solves this problem. When bootstrapping, data are simply re-sampled at random with replacements from the original trace. Bootstrapping is particularly useful when there is only a small sample of data available. In effect, it is a way of extending the use of traces. For detailed discussions on bootstrapping in simulation, see Cheng (2001) and Demirel and Willemain (2002).

Any remaining event data which is not covered by external data sources has to be generated from scratch with value generators inside the simulator. One of the strengths of our simulation model is that it allows to replicate and complement existing business data with artificially generated data that can be combined within a simulation run. As shown in Figure 7, an event sequence (1) was generated from a WS-BPEL process to reflect a selected process execution path (2).

The event sequence references event generators which use different methods for generating representative process events. The first two event generators of the event sequence extract events from an event trace captured in the call center (3). The following two event generators generate events for the order evaluation and packaging steps with value generators (4). The last two events are generated from fact and dimension tables of a data warehouse (5).

The combination of building event sequences from WS-BPEL process models with a hybrid approach for integrating various data sources for complementing a simulation model with real business data provides an elevated level of accuracy, predictability and adaptability to process changes when simulating events for event-based systems.

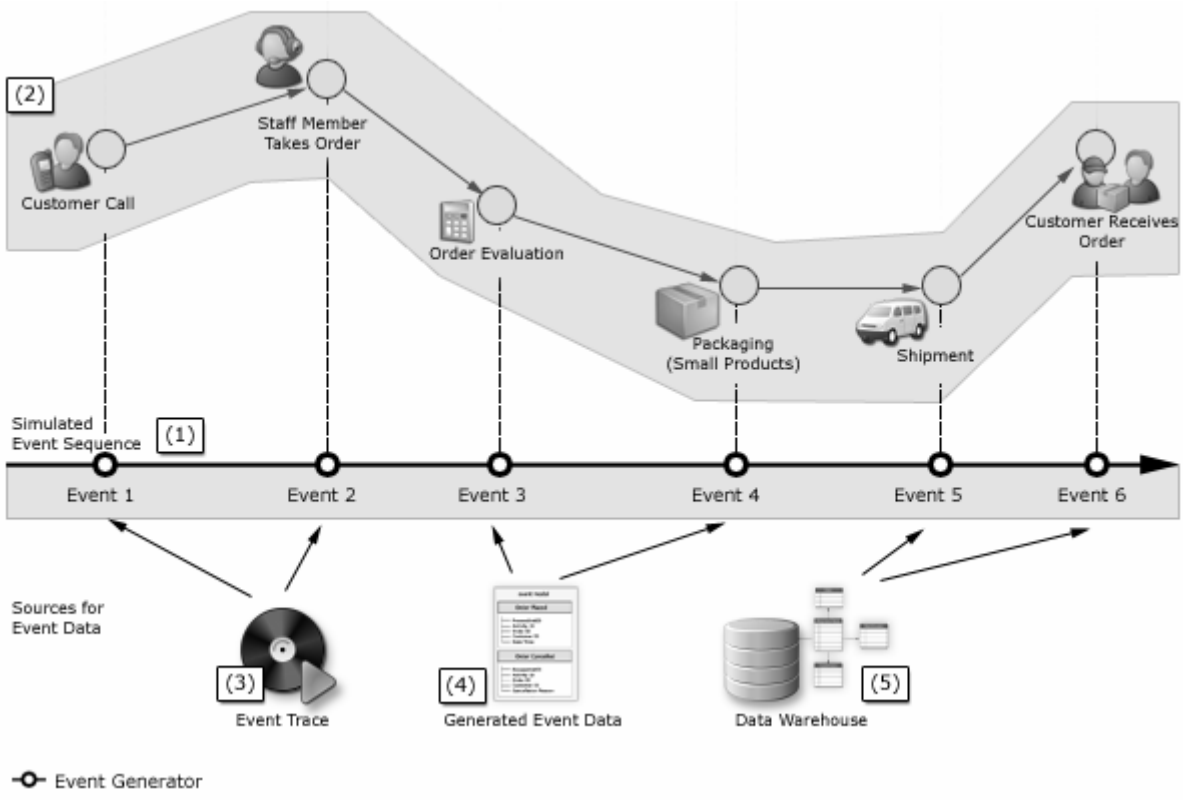

Figure 7: Multiple Sources for Simulating an Event Sequence

7 CONCLUSION AND FUTURE WORK

The event-based systems have been largely studied and used building and monitoring loosely coupled business process solutions. This paper has presented a simulation model for imitating business process events in order to improve event-based systems. The simulation model offers multiple ways for defining event sequences and generating representative and consistent event data. We showed how event sequences can be modeled by a user or generated from an annotated WS-BPEL process model and illustrated our approach with a real-world example.

The work presented in this paper is part of a larger, long-term research effort aiming at developing an event-driven business process management platform. This paper discussed an approach on how to create event sequences from an annotated WS-BPEL process definition. In reality, an event-based system may receive events from more than one (interacting) business process. Therefore, we want to investigate how to simulate interaction scenarios of process engines.

References

Balci, Osman (1988). The implementation of four conceptual frameworks for simulation modeling in high-level languages. Proceedings of the 20th conference on Winter simulation, 287-295, San Diego, CA

Cheng, R.C.H. (2001). Analysis of simulation experiments by bootstrap resampling. Proceedings of the 2001 Winter Simulation Conference, 179-186, Piscataway, NJ.

Demirel, O.F. and Willemain, T.R. (2002). Generation of simulation input scenarios using bootstrap methods. Journal of the Operational Research Society, 53 (1), 69-78.

Derrick, E. J., Balci , O. and Nance, R. E. (1989), A comparison of selected conceptual frameworks for simulation modeling, Proceedings of the 21st conference on Winter simulation, 711-718, Washington D.C

Kiczales, G. and Mezini, M. (2005). Separation of Concerns with Procedures, Annotations, Advice and Pointcuts, European Conference on Object-Oriented Programming, Glasgow, UK.

Lackner,, M.R. (1962), Toward a General Simulation Capability, Proceedings of the AFIPS 1962 Spring Joint Computer Conference, 1-14, Paolo Alto.

McGregor, C. and Schiefer, J. (2004). Correlating Events for Monitoring Business Processes, 6th International Conference on Enterprise Information Systems, Porto.

Nance, R. E. (1981), The Time and State Relationships in Simulation Modeling, Communications of the Association for Computing Machinery 24, 173-179.

Nance, R. E. (1993), A History of Discrete Event Simulation Programming Languages, ACM SIGPLAN Notices v.28 n.3, 149-175.

Pidd, M. (1998) Computer Simulation in Management Science, 4th edn. Chichester, UK, John Wiley and Sons Ltd.

Robinson, S. (2003). Simulation - The practice of model development and use, John Wiley and Sons Ltd.

Schiefer, J. and Seufert, A. (2005). Management and Controlling of Time-Sensitive Business Processes with Sense & Respond, International Conference on Computational Intelligence for Modelling Control and Automation, Vienna.