

# From Actions, Transactions, and Processes to Services

Manfred Broy<sup>(✉)</sup>

Institut für Informatik, Technische Universität München, 80290 Munich, Germany

broy@in.tum.de

<http://www.broy.informatik.tu-muenchen.de>

## 1 Introduction

For the problem domain of business process engineering we introduce, model, and formalize notions of business processes such as action, actor, event, business process, and business transaction. In addition, for the solution domain of service-oriented architectures (SOA) we introduce, model, and formalize notions of service, service composition, service-oriented architecture, and layered SOA in a systematic way. We do that by a rigorous mathematical system model. For that purpose, we first develop a basic mathematical system model for formalizing fundamental concepts of processes and services. The goal is to provide a minimal set of formal modeling concepts, nevertheless expressive enough to formalize key notions and concepts in business process engineering and service-oriented architectures capturing also their mutual relationships. This way, the relationship between central notions in business process models is captured formally, which provides a basis for a methodology for deriving the systematic specification and design of service-oriented architectures from business process modeling. The purpose of the approach is manifold; one goal is a clear definition of terminology, concepts, terms, and models in business process modeling and SOA; another goal is a rigorous formal basis for the specification, design, and development of business processes and, in particular, SOAs. We end up with a strictly formal concept for the development steps from business process models to services as part of a SOA-oriented development process.

This is an extended abstract of [Broy 15].

## 2 Business Process Design

When designing business processes and, in turn, developing business transaction and workflow support software, concepts of domain modeling in terms of business processes and software based business applications are needed. In consequence, various notions and modeling concepts of two disciplines, namely software engineering and business process engineering, have to be related, harmonized, integrated, and conceptually unified. The challenge is to provide a comprehensive modeling framework expressive enough to support the development steps in the construction of service-oriented architectures (SOA) on the basis of business process models.

In the development of a SOA for business applications, it is first to capture and work out the goals and requirements in the application domain. Then based on these

requirements we work out the business processes. Finally, we determine which parts of the software processes are to be supported by computer-based services to derive from them the software requirements and specifications, further on the software service architecture, and finally service implementations. For carrying out such development tasks, models, and modeling concepts are applied both from the field of business process modeling and model based software and system development – more specifically from the field of service-oriented architecture.

Our main concern is to provide formal models for processes and formal models for service-oriented systems in terms of their interfaces to model key notions in business process engineering. We use process models to model notions like *business process* and *business process transaction*. We use the notion of systems and system interface behavior to model business services. One of our goals is to relate these notions to the concept of services as studied in service-oriented architectures.

We aim at a synergy between the problem domains of business process engineering and that of formal system modeling. For both areas, we work out formal concepts as a basis for engineering service-oriented architectures (SOA).

**Remark:** Avoiding clashes in engineering terminology.

One delicate difficulty for our approach is a terminological clash between two disciplines, that of “business process engineering” and that of “formal process modeling”. This difficulty is immediately recognized when studying the general term “process” in the field of event processing. Moreover, in the field of business processes, the term “business process” is used differently addressing a number of different notions like the business process of an organization, the instance of the business process during a day of operation, or the business process of a business transaction as an instance of a specific business case; in formal system modeling, formal models of discrete processes are introduced as generic concepts for modeling discrete activities.

**End\_of\_Remark**

For service-oriented architecture we formalize the notion of the service layer leading to the concept of service-layered architecture.

### 3 Fundamentals in Discrete Process Modeling

Modeling discrete processes is one of the challenges in the development of automation systems – be it in business automation or in production automation.

The term process is used for many quite different notions and concepts. It is used for behavior described by a system or by a system model, execution mechanism (a Petri-net, for instance), or for a particular instance of behavior of a system or system execution mechanism model (an occurrence net for a Petri-net). In addition, given such an instance of behavior, we may consider certain sub-behaviors representing (smaller) instances of behavior.

This shows that we deal with three forms of notions of processes and their description:

- process descriptions (by modeling concepts) with which we associate

- sets of instances (scenarios) of behavior, the members of which are
- instances (scenarios) of behaviors.

In the following, we use the word “process” strictly for an instance of behavior. So we get the following terminology:

- a system description (of a business system), called a *process specification*, describes a
- *set of (business) processes* where
- *(business) processes* are instances of system behavior (representations of system executions).

In this terminology, a process is used to denote the run (“history”) of a system like an occurrence net represents an instance of behavior of a Petri-net. A Petri-net is an example of a process specification. Since we can identify in a given process several sub-processes, a process can also be understood as describing sets of (sub-)processes. This general approach of understanding the term process is used in the following in a more concrete and detailed form by introducing a modeling theory for processes.

## 4 Process Descriptions and Instances Business Process Engineering - Notions and Concepts

In business process engineering, the concept of a process is essential. The term “business process” is used in business process engineering in a very generic way. It addresses the process (in the sense of instance) of a particular business transaction as well as the process of a particular sub-organization as well as the comprehensive set of activities of an entire company or even a network of companies. In this section, we introduce a taxonomy of slightly different form for capturing business processes and their specification.

A business process consists of a set of business actions<sup>1</sup> (often called “steps”) executed by business actors. Their actions are in some logical, causal, and temporal relationship.

For a business process and its specification, we introduce several views and structures. We take the following fundamental viewpoint onto business processes. In a business process, actors execute a set of actions. This way we obtain two essential notions:

- business actions (single steps of activity in a process)
- business actors (humans or machines that carry out actions).

In addition, we might consider business objects and business data as they are used to capture the states related to business transactions. Business actions usually have certain effects on business objects. They may change states of business objects and they may consume or produce events that relate to actions in terms of the exchange of messages (or even exchange of material or energy). We do not consider business objects

---

<sup>1</sup> Note that the choice of a set of actions determines also the granularity of the model of a business process. This granularity can be changed by replacing an action by a set of sub-actions or vice versa.

and business data, in particular, in the following; they can be added, however, in a straightforward manner to the approach.

Actors are humans or computer systems that carry out actions. By carrying out actions, actors provide specific services. While carrying out services actors may observe certain actions executed by other actors. These observations may trigger actors to executions of further actions.

This leads to another key notion in business process engineering, the notion of a service, more precisely to the notion of a

- business service.

A business service is carried out (provided) by some business actor or a set of actors (the service provider); it consists of a business process generated by the actor in reaction to the process and the actions the actor observes. The set of actions the actor observes (which form a sub-process of the overall process) are called its *service input*, the set of actions generated by the actor (which also form a process) are called its *service output*.

To illustrate our notions, we sketch the example of a web shop with the help of our previously introduced terminology. Words referring to key concepts are put in *italic*. The web shop operates by carrying out a number of *actions*. The ordering of a product in the web shop by a customer is done by a set of actions by the customer and the web shop forming a *business transaction (instance)*. The structure of all possible business transactions can be described by a *business transaction specification*. Within a given time interval (one hour, one day, one month, one year) a set of business transactions is carried out; they form what we call a *business transaction flow* (for instance, the process of the web shop consisting of all actions and transactions carried out in one year). Note that there may be and usually are dependencies between the individual business actions and business transactions that occur within a business transaction flow. There are various ways to describe such dependencies (for instance, business objects) and derive sub-transactions (sub-processes) from a business transaction flow.

There are several ways to define the notion of a *business process*. It addresses the set of transactions that serve a certain business purpose (“business purpose viewpoint”):

- (a) We may call the set of all possible business transactions business process.
- (b) We may call the set of all possible business transaction flows business process.
- (c) We may call an instance of a business transaction business process.

We tend to follow position (c) where we include (b) since the business transaction flows include the business transactions as sub-transactions (“sub-processes”). We call what is addressed by (a) and (b) the business process specification. For our running example, the business process specification describes all the business transaction flows and also the set of all possible business transactions of the web shop.

## 5 From Business Processes to Service-Oriented Architectures

Services and service-oriented architectures (SOA) have received much attention in recent years for good reasons (see [Broy 05], [Broy et al. 07]). When dealing with large

systems, it is generally recognized that we need appropriate abstractions. These abstractions have to be useful for the implementation of systems, but they are even more urgently needed for the design and specification of domain specific aspects of information processing systems. Service-oriented architectures aim, in particular, at structuring software systems not so much governed by technical concepts of implementation but rather driven by concepts of the application domain with its domain-specific terms and conditions.

We understand service-oriented architecture as an approach that follows classical general goals and principles of software system construction such as modularity, application domain orientation, and strict concentration onto the user-centric functionality of systems. These principles are in response to the growing demand of putting emphasis on system evolution and maintenance with flexible response to changing requirements. In addition, SOA is expected to lead to a higher quality of software systems in terms of changeability, adaptability, interoperability, and reusability. In the following, we introduce a basic formal model for services as they are used in service-oriented architectures.

A further important issue is a methodological one defining what are systematic steps of the development of service architectures starting from high-level requirements and use cases. In this section, we give a very short and brief preliminary description of how such a top-down development methodology could look like.

We characterize the proceeding by the following steps:

1. We give *use cases*. In each use case, we describe scenarios of service use that correspondent very directly to scenarios that can be formally seen as service instances.
2. From the scenarios, we derive *service processes* they can be described by process diagrams or by interaction diagrams. To do these diagrams we also have to formalize the service messages. The service messages can be directly identified from the service use cases.
3. In the processes, we have steps, which are done manually and steps, which are done by the software systems. Therefore, it is an important decision which of the steps in the systems are done manually and which of the steps are done by the software.
4. By identifying steps that are to be done by software, we can derive from the service processes, the *service hierarchy* as well as *services instances* for each of the services in the hierarchy.
5. For each of the services, we define its *service interface behavior*. Perhaps, we decide to introduce additional auxiliary services that provide small service provision networks for the services to implement.
6. From this, we get on one hand *service architectures* and a black box descriptions of each of the services involved.
7. The black box view of the provided services, have to be correctly represented by the service composition, including a *layered architecture* in terms of a *stack of internal services*.

This shows, how service architectures can be worked out in a top-down fashion. In fact, we could also use the same approach in a bottom up development. What we finally get is an artifact model for service-oriented architecture where all of the ingredients of

a service architecture are given that are more less as described by the notion and concept model by the introduced mathematical techniques.

## 6 Conclusion

In this paper, we base the formalization of SOA concepts on the theory of processes and process oriented service functions. We can use all concepts such as

- service separation
- service refinement
- service specification and implementation.

for the engineering of SOA systems.

An interesting question addresses the difference between object- and service-orientation (OO vs. SO). For a number of practical SOA approaches the answer is not obvious, in particular, when OO concepts are used to represent services. However, there is a significant difference between OO and SOA that has to be understood to see the advantages of SOA over OO:

- Typically, OO-concepts are sequential and method-invocation oriented,
- SOA approaches are, by nature, taking into account time, parallel and concurrent computation and explicitly support of distribution, interaction, and time.

Of course, we may use OO-concepts to implement SOA, but these concepts are too weak to represent SOA ideas explicitly. The strength of SOA can be fully exploited only by a dedicated modeling framework addressing interaction and concurrency explicitly.

By the constructions introduced, we provide the following foundations:

- We describe general models for several important notions and terms in business process modeling.
- We introduce a very compact general formal model and theory for processes and services.
- Due to the form of models, we can define formally the relationship between these notions and how they interact with each other.
- The main idea is to use this as a foundational framework for a methodology and development processes for business systems.

The ultimate goal is to provide a foundation for a formalized approach to service-oriented architectures. Service-oriented architectures claim to be the better approach to develop main business process applications. A key issue here is the step from a descriptive approach to processes where processes are described as structures of activities to an input/output oriented view, which we call services. By our notion, we capture formally the heart of the idea of business process modeling and the formal step from business process modeling into SOA.

## References

- [Aalst, Stahl 01] van der Aalst, W., Stahl, C.: *Modeling Business Processes: A Petri Net-Oriented Approach*. MIT Press, Cambridge (2011)
- [Broy, Stølen 01] Broy, M., Stølen, K.: *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer, New York (2001)
- [Broy 03] Broy, M.: Modeling services and layered architectures. In: König, H., Heiner, M., Wolisz, A. (eds.) *Formal Techniques for Networked and Distributed Systems*. LNCS, vol. 2767, pp. 48–61. Springer, Berlin (2003)
- [Broy 04] Broy, M.: The semantic and methodological essence of message sequence charts. *Sci. Comput. Program. SCP* **54**(2–3), 213–256 (2004)
- [Broy 05] Broy, M.: Service-Oriented systems engineering: specification and design of services and layered architectures – the Janus approach. In: Broy, M., Grünbauer, J., Harel, D., Hoare, T. (eds.) *Engineering Theories of Software Intensive Systems*, pp. 47–81. Springer, Dordrecht (2005)
- [Broy et al. 07] Broy, M., Krüger, I., Meisinger, M.: A formal model of services. *TOSEM - ACM Trans. Softw. Eng. Methodol.* **16**, 1 (2007)
- [Broy 11] Broy, M.: Towards a theory of architectural contracts: schemes and patterns of assumption/promise based system specification. In: Broy, M., Leuxner, C., Hoare, T. (eds.) *Software and Systems Safety - Specification and Verification*. NATO Science for Peace and Security Series D: Information and Communication Security, vol. 30, pp. 33–87. IOS Press, Amsterdam (2011)
- [Broy 10] Broy, M.: Multifunctional software systems: structured modeling and specification of functional requirements. *Sci. Comput. Program.* **75**, 1193–1214 (2010)
- [Broy 15] Broy, M.: From actions, transactions, and processes to services. In: Irlbeck, M., Peled, D., Pretschner, A. (eds.) *Dependable Software Systems Engineering*. NATO Science for Peace and Security Series D: Information and Communication Security, vol. 40, pp. 42–78. IOS Press, Amsterdam (2015)
- [Großkopf et al. 09] Großkopf, A., Decker, G., Weske, M.: *The Process: Business Process Modeling Using BPMN*. Meghan-Kiffer Press, Tampa (2009)
- [Haar 00] Haar, Stefan: Occurrence net logics. *Fundam. Inf.* **43**(1–4), 105–127 (2000)
- [Küster-Filipe 06] Küster-Filipe, J.: Modeling concurrent interactions. *Theoret. Comput. Sci.* **351**, 203–220 (2006)
- [Petri 62] Petri, C.A.: *Kommunikation mit Automaten*. Institut für instrumentelle Mathematik der Universität Bonn (1962)
- [Thurner 04] Thurner, V.: *Formal fundierte Modellierung von Geschäftsprozessen*. Dissertation, TU München, Munich (2004)
- [Torka 13] Torka, P.: *Dienstorientierte Architekturen: Eine konzeptuelle Herleitung auf Basis eines formalen Prozessmodells*
- [Winskel, Nielsen 95] Winskel, G., Nielsen, M.: Models for concurrency. In: Abramsky, S., Gabbay, D., Maibaum, T. (eds.) *Handbook of Logic in Computer Science. Semantic Modeling*, vol. 4, pp. 1–148. Oxford Science Publications, Oxford (1995)

Application and Theory of Petri Nets and Concurrency  
37th International Conference, PETRI NETS 2016,  
Toruń, Poland, June 19-24, 2016. Proceedings  
Kordon, F.; Moldt, D. (Eds.)  
2016, XVI, 345 p. 114 illus., Softcover  
ISBN: 978-3-319-39085-7