

## 2 Background

In this chapter we review related work and briefly present approaches and technologies used in the development of the presented approach for multilevel business process modeling and data analysis. Multilevel business process modeling builds on existing research on multilevel domain modeling, especially multilevel objects [76]. The proposed modeling approach qualifies as data- or artifact-centric business process modeling [82] and relies on the notion of behavior-consistent specialization of life cycle models [108]. Multilevel business process models are executable in an automated way, the resulting traces being subject to business process intelligence. The multilevel business process modeling approach relies on UML (with OCL) as modeling language and for the definition of the formal metamodel; XML serves as the format for logical representation.

### 2.1 Multilevel Modeling

A plethora of multilevel modeling approaches exists in the literature [10]. A common feature of these approaches is the support for arbitrary-depth instantiation/classification hierarchies. For a multitude of use cases, multilevel modeling approaches lead to a more accurate representation of reality than “traditional” modeling approaches with two-level instantiation/classification hierarchies [59].

The *multilevel object* [77] (m-object) is a versatile approach to multilevel modeling. The m-object combines elements of other multilevel modeling approaches, most notably powertypes, materialization, and deep instantiation (see [81] for a detailed comparison). A mapping of m-objects to OWL [79] renders multilevel modeling accessible to ontology engineers. The design of data warehouses using m-objects leads to an improved representation of heterogeneities in OLAP cubes [80]. In this spirit we extend the m-object modeling approach to the realm of business process management.

The notion of powertype derives from powersets in set theory [20] and has since been included in the UML standard [88, p. 54]. The instances of a powertype are subtypes of another object type, thereby providing metamodeling

capabilities [85, p. 28]. Gonzalez-Perez and Henderson-Sellers [33] propose a powertype-based approach without strict separation of traditional two-level instantiation with classes and objects, using instead the notion of “clabject”. In this approach, the powertype pattern consists of “a pair of classes in which one of them (the powertype) partitions the other (the partitioned type) by having the instances of the former be subtypes of the latter” [33, p. 83]. The relation of the m-object modeling approach to powertype-based approaches [29] is as follows. A level of an m-object may act as partitioned type and powertype at the same time. In an MBA’s level hierarchy, a parent level is the powertype of the child level.

Other concepts related to m-objects are materialization and deep instantiation. Materialization [95, 25] blurs the boundaries between aggregation and instantiation. Deep instantiation [11] introduces potencies to multilevel instantiation hierarchies. Attributes of a class may have potencies assigned. An attribute’s potency specifies the number of instantiation steps to be taken until the assignment of a value to this attribute happens. Dual deep instantiation abandons the strict metamodeling confinements of deep instantiation and distinguishes between source and target potencies [78].

## 2.2 Business Process Modeling

In large parts of this book we deal with business process modeling. More specifically, we propose an artifact-centric modeling approach for multilevel business processes. In this context, variability and flexibility are important aspects, with the notion of behavior consistency being closely related. Orthogonal to multilevel business processes is the traditional notion of business process model abstraction.

### 2.2.1 Data- and Artifact-Centric Modeling

A *business artifact* [82] encapsulates, in a single object, a data model along with the corresponding business process model for working with the data, referred to as the object life cycle model. Object life cycles are commonly modeled using variants of finite state machines [47]. Object behavior diagrams [55], for example, employ Petri nets for the representation of object life cycles. Other work [66, 72] leverages the expressive power of the BPMN standard for artifact-centric business process modeling. The

guard-stage-milestone approach [48, 49], on the other hand, produces a more declarative representation of object life cycles.

Various existing approaches towards business process management put their emphasis on the data objects involved in a process. The object-process methodology [26] defines the notions of objects, processes, and states as the main modeling primitives. States describe objects and processes change the states of objects. The PHILharmonicFlows framework [57, 56] supports object-aware business process management and distinguishes between micro and macro process modeling, the former capturing the behavior of individual objects, the latter representing interactions between objects. Proclets [5, 6] are an object-oriented representation of business processes, where a proclet corresponds to an object that is attached with an explicit life cycle model. The proclet modeling approach especially emphasizes the interaction between objects, rather than considering objects only in isolation.

The UML standard describes model types that may be employed for artifact-centric business process modeling. In such a UML-based approach [30], a UML class diagram represents the data model of business artifacts and UML state machines typically represent the life cycle model of artifacts. UML-based artifact-centric business process models are accessible to methods for the formal verification of correctness [19]. In this book, we employ UML state machines for representing object life cycles.

### 2.2.2 Variability and Flexibility

Central to the notion of variability is the concept of *process variant*. Different process variants may exist for achieving the same goal. These process variants have the same underlying core process but may differ from the core process with respect to the exact type and sequence of conducted activities [98, p. 45]. Configurable process models make explicit the variation points between different process variants [105]. Questionnaire-based approaches may reduce the complexity of handling multiple configurable process variants and facilitate the tailoring of a configurable process model to the specific needs of individual users [58, p. 105]. The operational approach towards the management of process variants performs change operations on a base process model, allowing for the insertion/deletion and modification of process fragments at specified variation points [40, 41]. Business process families [37] adapt the principle of software product lines for the representation of business processes. A business process family comprises a reference business process model and a set of features. The features relate to elements

in the process model and serve as the basis for customization. Process owners may customize the reference business process by using different selections of features. A business process family may also be characterized as a “collection of processes meeting a common goal but in different ways” [102].

Real-world business situations often necessitate dynamic adaptations of business process models [99]. Change patterns may guide modelers through the adaptation of process models and instances, thereby ensuring correctness of the resulting adaptation [138]. Flexible approaches towards business process management allow for the quick implementation of new processes and on-the-fly adaptation of process instances [97]. Business processes may also be flexible by design, providing process owners with different choices [98, p. 59 et seq.]. Meta-processes may allow for the dynamic construction of business process models, resulting in a business process model that optimally fits the needs of the current situation [104]. For artifact-centric business process modeling, the representation of a process design entity along with the actual business process model supports the handling of flexibility [65].

### 2.2.3 Behavior-Consistent Specialization

The notion of behavior consistency realizes variability in data- and artifact-centric business process models. A life cycle model that is a behavior-consistent specialization of another, more general life cycle model is a variant of this more general life cycle model. Different frameworks for behavior-consistent specialization rely on various different modeling languages, for example, Petri nets [3], UML state machines [125], or object/behavior diagrams [108]. More recent work [141] has investigated the observation-consistent specialization of synchronization dependencies. In the context of process views, behavior-consistent specialization assists with the propagation of local changes to a central process model [67].

Two flavors of observation consistency may be distinguished, namely *observation consistency* and *invocation consistency* [108]. Observation consistency applies to situations where the specialized life cycle model is observable in the same way as the more general life cycle model. In this case, any execution of a specialized life cycle model must be a valid execution of the more general life cycle model, when disregarding the refinements and extensions of the specialized life cycle model. The notion of invocation consistency is stronger than observation consistency, additionally demanding that any sequence of activities that is valid in the more general life cycle model must also be a valid in the specialized life cycle model.

### 2.2.4 Business Process Model Abstraction

In business process modeling, abstraction commonly refers to the description of the same process at different levels of granularity. A process may thus be composed by several sub-processes. For example, the negotiation and signing phase sub-processes constitute the conclusion of a contract. Most business process modeling languages allow for the representation of such abstraction hierarchies. UML state machines allow for the nesting of states under composite states [88, p. 560]. BPMN allows for the representation of sub-processes, the notation allowing for both a collapsed and expanded presentation in order to hide unnecessary details from the user [16, p. 118].

Business process model abstraction is essential to handling complexity of large-scale models [119]. Typically, business process model abstraction refers to the reduction of complexity in business process models by grouping individual activities into sub-processes [118], thereby providing a more general view on the underlying business process. Business process model abstraction is part of the broader research field on well-structuredness of business process models [96]. This whole field is orthogonal to multilevel business process modeling: Rules for well-structuredness may be applied individually to each of the business process models at the different levels in a multilevel business process model.

## 2.3 Business Process Automation

Business process management systems support modeling and execution of business processes [27, p. 298 et seq.]. In particular, business process management systems ensure the valid execution order of the activities of a business process and provision the appropriate resources needed for the completion of these activities. Some of these activities may even be completed autonomously by the software system, without human intervention. An automated business process may be referred to as *workflow* [27, p. 298]. The automation of business processes requires a suitable representation language [94]. For example, BPEL is a widely-supported modeling language for business process automation [84, 62]. The modeling language YAWL also comes with an execution environment, its formal foundation in Petri nets making it accessible to formal verification [44]. The Genesys Orchestration Server [32] employs State Chart XML, an XML-based representation language for state machines. For artifact-centric business processes, the Siena system [23] was among the first prototypes (cf. [24]) for the execution

of artifact-centric business processes. The Siena system employs XML for the representation of business artifacts, similar to the approach for business process automation as presented in this book, which also adopts an XML representation in order to allow for the execution of multilevel business processes. The Barcelona system [43] supports the design and execution of business processes using guard-stage-milestone models.

## 2.4 Business Process Intelligence

Following a review of BPI literature, Felden et al. [31, p. 200] define BPI as “the analytical process of identifying, defining, modelling and improving value creating business processes”. In the chapter on business process intelligence of their comprehensive book about the fundamentals of business process management, Dumas et al. present tools and techniques for “intelligently using the data generated from the execution of the process” [27, p. 353]. Business process intelligence (BPI) comprises a multitude of tools and techniques related to the analysis, monitoring, control, and optimization of business process execution [36]. An important aspect of business process intelligence is the discovery of processes through the mining of event data [21]. Other authors [74] use the term “business process analytics” as an umbrella term for various techniques for the analysis of event log data generated during the execution of business processes.

A data warehouse may organize the base data for business process analysis. In that case, the data warehouse is then often referred to as process data warehouse [36] or simply process warehouse [64, 63]. Process models may serve as the starting point for the definition of the schema for such a data warehouse [126, 70, 69]. Other work [122] investigates how knowledge about the life cycle models of the analyzed business objects may enrich a data warehouse schema. Furthermore, business processes may also require access to the data in a data warehouse and adjust further processing accordingly, based on the contents of the data warehouse [121, 123].

In business process analysis, measures may either refer to the models of business processes or the execution thereof [106]. Business process models may be analyzed with respect to their complexity and understandability, for example. A popular measure of interest for process execution is the cycle time, that is, the amount of time needed to complete a process instance [27, p. 219 et seq.]. In this book, we focus on the analysis of measures related to business process execution, in particular cycle time.

## 2.5 Modeling Languages

Conceptual models describe the domain of an information system [86]. The Unified Modeling Language (UML) defines model elements for both static and behavioral modeling [88], state machine diagrams and activity diagrams being the most notable for the modeling of behavior. Protocol state machines define the legal execution order of the methods of a class [88, p. 535 et seq.]. In this book, we rely on UML state machine diagrams in order to model artifact-centric business process modeling, in conjunction with UML class diagrams for representing the data elements.

Conceptual data models translate into database schemas [28]. The Extensible Markup Language (XML), although introduced as a web technology, has gained prominence for the specification of semi-structured logical database schemas. Examples for available XML database management systems are BaseX<sup>1</sup> and eXist-db<sup>2</sup>. In this book, we employ an XML database for the storage of business artifacts.

For business process automation, modelers must translate business process models into executable models, or workflow models [94]. State Chart XML (SCXML) is a W3C proposed recommendation for the representation of state machines using XML [136]. We use SCXML in the logical representation of business artifacts since its model elements and their semantics are very similar to UML state machines, allowing for an easy translation of conceptual multilevel business process models that use UML state machines into a corresponding logical representation. We differ from the SCXML specification in employing the XPath data model as described by the candidate recommendation [129] and last call working draft [130] since it neatly integrates into XML and XQuery; the feature was dropped due to lack of implementation [129].

XQuery is the standard query, manipulation, and programming language for XML data [134, 128]. The XQuery Update Facility (XQUF) [128] introduces data manipulation operations for XML documents. The concept of node identity from the XQuery and XPath data model [135] allows for (a sort of) object-oriented programming style. In this case, XML elements may be regarded as objects and passed to functions under preservation of their identity. Manipulation operations on these elements then propagate directly to the database.

---

<sup>1</sup><http://basex.org/>

<sup>2</sup><http://exist-db.org/>



<http://www.springer.com/978-3-658-11083-3>

Multilevel Business Processes

Modeling and Data Analysis

Schütz, C.

2015, XXIV, 232 p. 42 illus., Softcover

ISBN: 978-3-658-11083-3