Wand and Weber's Decomposition Model in the Context of Business Process Modeling

So far there are no generally accepted criteria for assessing the quality of a decomposed business process model. This is surprising, since the benefits of decomposing process models are well-known in literature. The paper at hand analyzes in how far the decomposition model of Wand and Weber can be used in the context of business process modeling. The decomposition conditions are interpreted for process models while it becomes obvious that they should not be used in an isolated way for judging the quality of a decomposition.

DOI 10.1007/s12599-012-0229-1

The Authors

Dr. Florian Johannsen (✉)
Prof. Dr. Susanne Leist
Department of Management Information Systems
University of Regensburg
Universitätsstraße 31
93053 Regensburg
Germany
Florian.Johannsen@wiwi.uni-regensburg.de
Susanne.Leist@wiwi.uni-regensburg.de

Received: 2011-10-31
Accepted: 2012-06-19
Accepted after three revisions by Prof. Dr. Buxmann.
Published online: 2012-09-05

This article is also available in German in print and via http://www.wirtschaftsinformatik.de: Johannsen F, Leist S (2012) Das Dekompositionsmodell nach Wand und Weber im Kontext der Prozessmodellierung. WIRTSCHAFTSINFORMATIK. doi: 10.1007/s11576-012-0334-2.

1 Outline of the Problem

Business process modeling is nowadays commonly accepted as an essential task within business process management efforts (Becker et al. 2010, p. 4). According to Becker et al. (2010, p. 4), business process models serve as a basis for decisions on IT-investments, process reengineering efforts, or also e.g., the design and implementation of information systems.

Nevertheless, as Mendling et al. (2007, p. 51) point out, the understandability of a process model is related to its size. The more activities a process model has, the harder it becomes to understand it (Mendling et al. 2010, p. 130). In contrast, an abstract process description is not appropriate when relevant details are missing, e.g., the derivation of requirements for an information system (Bobrik et al. 2007, p. 88). A well-known approach for creating process models that are manageable and understandable, but also contain all the information needed to serve the different purposes occurring in an enterprise (e.g., process improvement), is to decompose them into modules or subprocesses (Gruhn and Laue 2007, p. 13; Smirnov et al. 2011, p. 498; Reijers and Mendling 2008, pp. 20–21; Reijers et al. 2011, p. 881). Thus a process model is decomposed into subprocesses or model levels, differing in detail (Reijers et al. 2011, pp. 881–882).

Whereas the benefits of decomposing process models are obvious, the question what actually characterizes a "good" decomposition has been given little attention to date (Burton-Jones and Meso 2006, p. 40; 2008, p. 771). Burton-Jones and Meso (2006, p. 40) consider the process of decomposition to be "much more an art than a science". According to Reijers et al. (2011, p. 882), decomposition in process models is usually reached in an "ad-hoc" fashion. While several suggestions on how to decompose a model into subprocesses can be found in literature (e.g., La Rosa et al. 2011; Reijers et al. 2011), generally accepted guidelines which supports the user in assigning subprocesses to certain model levels are missing yet (Reijers et al. 2011, p. 882).

In that context, Wand and Weber's model for a good decomposition (Weber 1997, pp. 147–170), which has been developed for information systems, seems promising for establishing criteria to judge the quality of a decomposed process model (Recker et al. 2009, p. 355). The good decomposition model is a set of formal conditions that define criteria for the specification of well-defined systems and subsystems (Burton-Jones and Meso 2002, p. 102). It is part of the Bunge-Wand-Weber (BWW) ontological models, which have their roots in the information system discipline (Recker et al. 2009, p. 336; Fettke and Loos 2007, p. 11).

We have selected the BWW ontology for this investigation for four reasons. Firstly, the BWW ontology has been given considerable attention regarding the ontological analysis of modeling languages (e.g., Recker et al. 2005, 2007; Green and Rosemann 2000; Recker and Indulska 2007; Rosemann et al. 2009). In that context, it provides a theoretical foundation for the evaluation of modeling languages. Secondly, the basic character as well as the comprehensive scope of the BWW ontology allows a wide applicability (Recker et al. 2005). It covers all aspects of systems modeling (Burton-Jones and Meso 2002, p. 102) and, in particular, the decomposition model can therefore potentially provide a useful basis for evaluating the decomposition of process models. Thirdly, it has already been successfully used to analyze and evaluate decompositions in object-oriented analysis (see Burton-Jones and Meso 2002, 2008). Fourthly, the decomposition model has

not been analyzed in the context of business process modeling yet (Reijers and Mendling 2008, p. 23). Finally we are motivated by the fact that, by our research, we derive a set of propositions that can be empirically evaluated in subsequent works. In the long term, it is our aim to define guidelines for decomposing a process model which are based on theoretical foundations, like the decomposition model, and on corresponding empirical validations. With the study at hand, we therefore interpret the decomposition model for business process modeling in a first step.

Recker et al. (2009) have shown that modeling languages differ regarding their ontological completeness as well as their clarity. The interpretation of the decomposition conditions (Weber 1997, pp. 153–163) will thus be heterogeneous for different modeling languages (e.g., EPC, BPMN, or Petri-Nets). To guarantee the necessary level of detail for our investigation and to avoid confusion (especially regarding the heterogeneous terminology in business process modeling), we have in this paper restricted our research to extended event-driven process chains (eEPCs) (e.g., Scheer et al. 2005).

Based on our interpretation of the decomposition conditions, requirements a decomposed process model should meet can be derived. It is our purpose to find out if the decomposition model can help to create a better understanding of what actually characterizes a good decomposition in business process modeling, and thus can bring structure into the fuzziness of the whole subject.

The structure of the paper is as follows: In the next section, we present basics regarding eEPCs and decomposition of processes. Section 3 deals with the BWW models and highlights the relevance of the decomposition model for the research at hand. In Sect. 4, we reflect the decomposition conditions and interpret them for eEPC models. Thus we generate conditions indicating whether a decomposition of an eEPC model can potentially be considered as "good" or not. Finally we present an outlook and an agenda for further research.

2 Conceptual Basics and Related Work

2.1 Event-Driven Process Chains

Event-driven process chains (EPCs) emerged in the early 1990s and have gained considerable attention in business process modeling (Scheer et al. 2005, p. 119). Tool support and user acceptance play a key role regarding their popularity in practice (Mendling 2008, p. 17). The EPC is part of the ARIS framework and can be interpreted as a set of event types, function types, and connectors (Mendling et al. 2010, p. 130; Scheer et al. 2005, pp. 127–129). An eEPC model may be specified through additional information, for example data object types, which are provided by the different views of the ARIS framework (Scheer et al. 2005, pp. 120–124). In this regard, literature speaks of extended event-driven process chains (eEPCs) (see Scheer et al. 2005, p. 119).

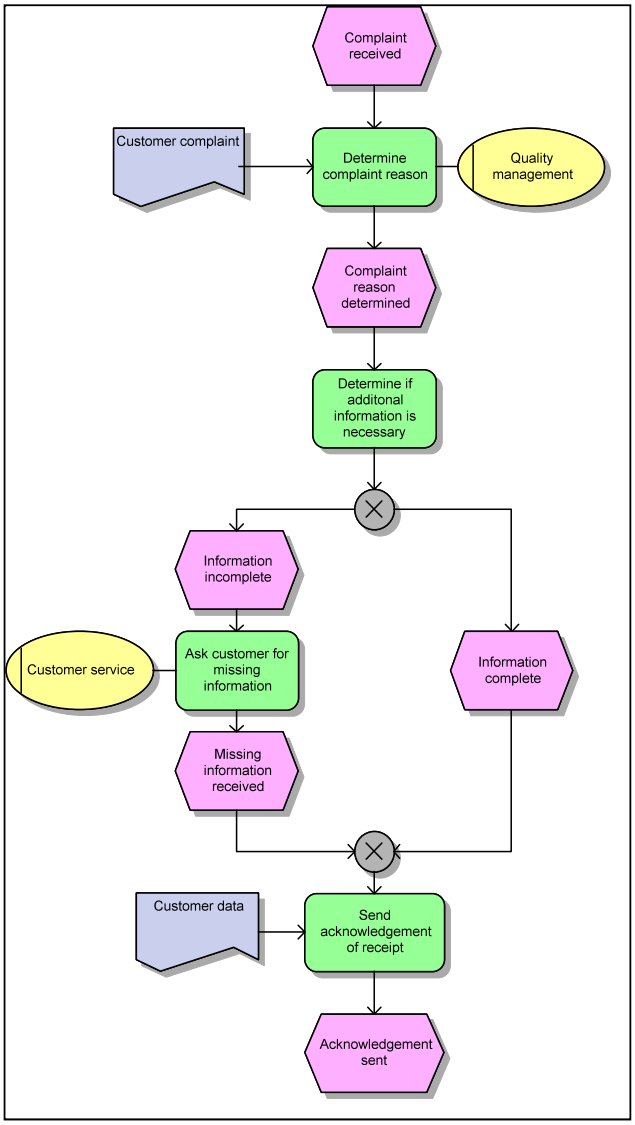

Fig. 1 Excerpt from a "customer complaint management" process model

An example of an eEPC representation can be found in **Fig. 1**. As already described by Mendling et al. (2010, p. 130), the tasks to be executed (so-called function types) are symbolized by green rectangles, while the event types (symbolized by red hexagons) represent the state before and after executing the function type. Responsibilities for performing the function types are indicated by constructs of the ARIS organization view (e.g., Scheer et al. 2005, pp. 120–121), in our case yellow ovals that stand for organizational units. "Routing rules" (Mendling et al. 2010, p. 130) are defined by logical connectors, which are symbolized by gray circles on a graphical level (Scheer et al. 2005, p. 127).

The example (**Fig. 1**) shows a subprocess "sending acknowledgment of receipt" of a complaint management process at a German fleet management com-

pany and describes the following procedure: As soon as the customer complaint has been received, the reason for the complaint is identified by the quality management department. If additional information is necessary for processing the complaint, the customer service asks the customer for the missing information. The customer is then sent an acknowledgement of receipt, as soon as the required information is complete.

2.2 Decomposition in Business Process Models

Regarding the quality of business process models (e.g., Gruhn and Laue 2007; Mendling et al. 2007; Vanderfeesten et al. 2007a, 2007b), "decomposition" can improve the understandability of a process model and reduce the probability of errors simultaneously (Mendling et al. 2010, p. 130).

In the literature, the term "decomposition" is used in several publications and in different domains (e.g., systems theory, conceptual modeling, business process modeling). In systems theory, "decomposition" is defined as "breaking down a complex system into smaller, relatively independent units" by Paulson and Wand (1992, p. 174). Mesarovic et al. (1970) for example decompose a holistic system into a multilevel, hierarchical structure. In conceptual modeling, the term functional decomposition is often used for describing the breakdown of a process into smaller pieces (Yourdon 1989).

In business process modeling, further expressions such as "modularization" (Reijers et al. 2011, p. 883; La Rosa et al. 2011, p. 617) are used alongside with "decomposition" (Mendling et al. 2010, p. 130; Vanhatalo et al. 2007, p. 43; 2008, p. 102) for describing the derivation of several smaller models that represent subprocesses in the sense of Mendling et al. (2010, pp. 130–131).

The decomposition itself can be accomplished in two ways (see Davis 2001, p. 242; Davis and Brabänder 2007, p. 256). We will refer to these as *decomposition approach I* and *II* in the following:

On the one hand, the process designer may start by creating a high-level process model (Davis and Brabänder 2007, p. 256). In further modeling steps additional levels of details are added (Davis and Brabänder 2007, p. 256). Therefore the process designer establishes subprocesses which describe certain high-level concepts in a more detailed way (Davis and Brabänder 2007, p. 256) (*decomposition approach I*). The subprocesses are arranged within a level structure (see Davis 2001, p. 244; Davis and Brabänder 2007, p. 257).

On the other hand, the process designer may create a detailed process model right from the start (Davis 2001, p. 242; Davis and Brabänder 2007, p. 256). More abstract views are then generated by abstracting from the details (Davis and Brabänder 2007, p. 256) respectively splitting the "detailed process model into a group of related, understandable subprocesses" (Reijers et al. 2011, p. 882) (*decomposition approach II*). The subprocesses are assigned to certain model levels (see Reijers et al. 2011, p. 882). It should be mentioned that Reijers et al. (2011, p. 882) use the term "modularization".

In the following, we give a brief overview of existing work concerning the process of decomposing a business process model:

- A common concept referred to is "block-structuring" (Reijers et al. 2011, pp. 891–892; La Rosa et al. 2011, p. 616), which means building "blocks" around the nodes of a process model. Thus a detailed process model is required (*decomposition approach II*). These "blocks" represent subprocesses. This approach can also be found in La Rosa et al. (2011, p. 618), subsumed under the "vertical modularization pattern".
- A further approach according to Reijers et al. (2011, p. 892) uses graph clustering to find nodes that are strongly "connected" to each other and may form a subprocess. In that context, also the "label similarity" (Reijers et al. 2011, pp. 892–893) of nodes can be helpful for clustering the nodes. Thus nodes holding a similar label are assumed to address common topics in a process model, and are therefore grouped into subprocesses (Reijers et al. 2011, pp. 892–893). Again a detailed process model has to be established first (*decomposition approach II*).
- The basic idea of delineating subprocesses that give details for specific tasks within an abstract model representation is also known as "disaggregation" (e.g., Malone et al. 1999; vom Brocke 2006; Heinrich et al. 2009) or "hierarchical decomposition" (Davis 2001, p. 242; Davis and Brabänder 2007, p. 256). In this context, Malone et al. (1999, p. 428) introduce the "Process Compass" to support the vertical disaggregation of a process into subprocesses. A generic process model is therefore decomposed into "parts" of the process considered (see Malone et al. 1999, pp. 428–429) while additional information is provided (*decomposition approach I*).
- La Rosa et al. (2011, pp. 619–620) mention further approaches for splitting up a process model by means of "horizontal" and "orthogonal" modularization. By "horizontal modularization" La Rosa et al. (2011, p. 619) address the derivation of several small models – that all have the same level of abstraction – from one original model. This idea is also found under the term "horizontal segmentation" (Davis 2001, p. 233; Davis and Brabänder 2007, p. 249) or "specialization" (Malone et al. 1999, p. 428), for example. Thus subprocesses are created by specifying a general process regarding a common object (e.g., "loan type") (see La Rosa et al. 2011, p. 620; Malone et al. 1999, pp. 427–428; Davis 2001, pp. 231–232; Davis and Brabänder 2007, pp. 247–248). When modularizing process models in an "orthogonal way" (see La Rosa et al. 2011, pp. 619–620), subprocesses are built around specific "concerns" (e.g., exception handling) of a company. In this way subprocesses are generated which address specific functions needed over and over again within different process models (e.g., "identity verification") (La Rosa et al. 2011, pp. 619–621). Building subprocesses around certain concerns of a company has also been promoted by Österle (1995), who suggested building subprocesses around "services", "business objects", or "specific customer segments". The subprocesses can be defined during modeling (*decomposition approach I*) or on the basis of a detailed process model (*decomposition approach II*).
- Another approach to reduce complexity is known as "process model abstraction" (e.g., Smirnov et al. 2011, 2010a, 2010b) in literature. Following this idea, a detailed process model is given (*decomposition approach II*), while certain parts of the model are aggregated to generate more abstract views

Table 1 Criteria for a "good" decomposition

Approach	Sources	Criteria for a "good" decomposition
Block-structuring	e.g., Reijers et al. 2011, pp. 891–892; La Rosa et al. 2011, p. 616	A "good" decomposition is given if each "block" (subprocess) is characterized by one incoming control-flow arc and one control-flow arc leaving the block (single entry single exit component – SESE component) (e.g., Vanhatalo et al. 2007, p. 43; 2008, p. 100).
Graph clustering	e.g., Reijers et al. 2011, pp. 892–893	The nodes within a cluster (subprocess) are more firmly connected to each other by arcs than to any node outside the cluster. Following the "label similarity" criteria according to (Reijers et al. 2011, pp. 892–893), the nodes within a cluster are supposed to be similar regarding their labels.
Disaggregation/hierarchical decomposition/vertical modularization	e.g., Malone et al. 1999; Davis and Brabänder 2007, p. 256; La Rosa et al. 2011, p. 618	As La Rosa et al. (2011, p. 618) mention, an indicator for a good decomposition is whether the subprocesses form "single entry single exit components" (SESE components) (e.g., Reijers et al. 2011, p. 891) or not. Davis and Brabänder (2007, p. 258) suggest that a decomposition has between three and seven levels.
Horizontal modularization/horizontal segmentation/specialization/orthogonal modularization	e.g., La Rosa et al. 2011, pp. 619–621; Davis and Brabänder 2007, p. 249; Malone et al. 1999, p. 428	According to Davis and Brabänder (2007, p. 252) the subprocesses are delineated properly, when they have "limited connections to other parts of the process" (expressed by arcs) and are linked by events. Thus each subprocess should deal with a "common object" (e.g., loan type), "concern", or "theme" (e.g., order handling).
Process model abstraction	e.g., Smirnov et al. 2011, 2010a, 2010b	The abstraction of a model has been done properly when activities that are similar to each other regarding activity properties (e.g., data object or role) (see Smirnov et al. 2011, pp. 499–504), regarding their labels and ancestors in a meronomy tree (Smirnov et al. 2010b) or regarding behavioral relations (Smirnov et al. 2010a) have been aggregated for example.

for specific user-groups. Smirnov et al. (2011) aggregate activities by building clusters of activities based on a distance measure which considers properties of an activity (e.g., data objects or roles). In addition, Smirnov et al. (2010a) also propose a meronomy-based approach considering semantics for aggregating activities. Furthermore, behavioral profiles of a process model can be used for deriving an abstract model structure (see Smirnov et al. 2010b). The idea of considering the execution of a process to decide which information should be presented to a user can also be found in the slider approach of Polyvyanyy et al. (2008).

While the above mentioned approaches are very specific, there is no universally accepted recommendation which of these is to be used regarding a certain project situation within a business process modeling project. Furthermore, business process modeling still lacks a generally accepted theory for decomposition.

2.3 Criteria for Judging a "Good" Decomposition

Based on the approaches introduced (Sect. 2.2), criteria to judge the quality of the decomposition can be derived. The following **Tab. 1** gives a brief overview.

While it becomes obvious that criteria can be defined regarding specific approaches, a generally accepted suggestion for judging the quality cannot be given.

3 The Decomposition Model of Wand and Weber in the Context of Business Process Modeling

3.1 Basics of the BWW Models

The BWW models comprise three concepts: the representation model, the state tracking model and the good decomposition model (Weber 1997, pp. 85–86; Rosemann and Green 2002, p. 78).

The central construct in the BWW representation model is a "thing" (Rosemann and Green 2002, p. 81; Weber 1997, p. 34; Rosemann et al. 2004, p. 115). As Weber (1997, p. 34) states, "the world is made up of things" (humans, IT-systems, etc.). A thing possesses properties (hair color, processing power, etc.) which are represented by attributes (Weber 1997, p. 34; Rosemann and Green 2002, p. 81). The state of a thing is expressed by the "values" that are assigned to its attributes (Weber 1997, p. 37). In that context, different types of properties have to be distinguished (see Weber 1997, pp. 35–37). A brief overview concerning property types can be found in Green and Rosemann (2000, p. 75), for example. According to Weber (1997, p. 41), a thing goes through changes throughout its lifetime which is documented in the history of the thing. A change of state is achieved by a transformation (Weber 1997, p. 41). The things themselves are grouped within a system, with each system having its own environment (Weber 1997, pp. 44–45). A complete overview of the constructs of the representation model is given in Green and Rosemann (2000, pp. 75–76),

while a detailed explanation can be found in Weber (1997, pp. 33–54). In addition, Rosemann and Green (2002) have developed a corresponding meta model for the BWW representation model, to ease understanding and communication of its basic ideas (Rosemann and Green 2000, p. 622).

The decomposition model is an ontological model that is based on the concepts of the abovementioned representation model and has been developed for assessing "the goodness of information system representations" (Weber 1997, p. 169). For this aim five conditions have been proposed, namely the (1) minimality condition, the (2) determinism condition, the (3) freedom from losslessness condition, the (4) minimum coupling condition as well as the (5) strong cohesion condition (see Weber 1997, pp. 153–163).

As the decomposition model of Wand and Weber is investigated in this paper, the definition of "decomposition" according to Weber (1997, p. 46), who defines decomposition regarding systems and subsystems, is transferred in the context at hand. Adapting the ideas of Weber (1997, pp. 46–48) for this paper, the "decomposition of a process" can be defined as a set of subprocesses, while the process "equals the union of the compositions"[1] of the subprocesses in the set. The decomposition is represented in a level structure of subprocesses (e.g., Davis and Brabänder 2007, p. 257), while the process or the subprocesses are depicted as corresponding process models.

3.2 Relevance of the Good Decomposition Model for the Research at Hand

As mentioned in Sect. 2.2, literature provides suggestions supporting practitioners in designing a decomposed business process model. But as Reijers et al. (2011, p. 882) or Davis and Brabänder (2007, p. 267) mention, there is no "unique way" to decompose or modularize a process model. Thus different approaches for decomposition (Sect. 2.2) may be used by a business analyst to derive subprocesses. While it has been shown that decomposition actually reduces the complexity of a model (e.g., Reijers et al. 2011, pp. 884–890; Burton-Jones and Meso 2008, pp. 756–801), the user's perception of a "good decomposition" is rather subjective. An objective basis for evaluating a decomposed model is missing.

The use of existing metrics for process model quality (e.g., Gruhn and Laue 2007; Mendling et al. 2007; Vanderfeesten et al. 2007a, 2007b, 2008a) seems doubtful, since they were developed for non-decomposed models. It is still an open issue whether using them on subprocess models and aggregating the corresponding values reflects the perceived complexity of a holistic model or not (Reijers et al. 2011, p. 882). In general, metrics focusing on decomposed models and enabling an evaluation of alternative designs still need to be developed (Reijers et al. 2011, p. 883). While it seems obvious that criteria can be derived regarding specific approaches for decomposition or modularization (Sect. 2.3), a general means for judging the quality of a decomposition, regardless of the decomposition approach used, does not exist. Consequently, generally valid criteria need to be established.

Therefore we follow a different approach in this paper and investigate how Wand and Weber's decomposition model can be transferred to business process modeling. The conditions of the decomposition model are based on an ontological model (BWW), which has already been successfully used for the evaluation of modeling languages (e.g., Recker and Indulska 2007; Rosemann et al. 2009; Recker et al. 2005, 2007; Green and Rosemann 2000). Since the BWW ontology is characterized by a wide scope, covering all facets of systems modeling (Burton-Jones and Meso 2002, p. 102), the decomposition conditions provide an extensive perspective to the "discipline" of decomposition. While the decomposition criteria introduced in Sect. 2.3 are strongly related to the decomposition approach used (Sect. 2.2), the decomposition model of Wand and Weber introduces conditions that are independent from specific approaches (Sect. 2.2) and are universally applicable. The conditions of Wand and Weber focus on the result of the decomposition. As Burton-Jones and Meso (2006, p. 40) mention, the good decomposition model of Wand and Weber (see Wand and Weber 1989, 1990; Weber 1997) is the only general theory of decomposition that has been proposed for information systems.

Burton-Jones and Meso (2002, 2006, and 2008) conducted an empirical investigation and demonstrated the benefits of the decomposition conditions for gaining understandable UML diagrams. In addition, Burton-Jones and Meso (2008) found out that high quality decompositions do not only ease the understanding of conceptual models, but additionally directly affect an individual's understanding of a domain considered. However, whether the positive results obtained by Burton-Jones and Meso (2002, 2006, and 2008) also hold true for business process models, needs to be investigated thoroughly. A respective investigation has not been carried out yet (Reijers and Mendling 2008, p. 23). If the decomposition model can be transferred to business process modeling, metrics for guiding the decomposition of a business process model in the sense of Reijers et al. (2011, p. 883) may be derived in further steps.

It has to be mentioned that the interpretation of the decomposition conditions for UML diagrams by Burton-Jones and Meso (2002, 2006, and 2008) seems rather subjective. The link between their specification of the decomposition conditions and the ontological analysis of UML is not obvious. This link would have been helpful for assessing how far their interpretation harmonizes with the original ideas of Wand and Weber (1989). In addition, the authors focus on conceptual models (UML diagrams) describing an information system and not business process models.

3.3 Different Terms and Definitions

Wand and Weber developed their decomposition model for information systems. It is built on clear definitions of an information system and its components (Weber 1997, pp. 33–54). The focus is on things and their relations (Weber 1997, p. 34). While Wand and Weber also consider transfer functions and transformations which change the state of a thing (Weber 1997, pp. 49–52), the decomposition model – in its origin – neglects the view on processes and functions. In addition, the terms Wand and Weber use to describe an information system and its components differ from those common in business process modeling with

[1] Since a user may consider a composition of selected subprocesses within the set only, the union of all compositions is necessary for representing the holistic process.

Table 2 Comparison of terms and definitions

BWW	eEPC	Additional comments
Thing (instance)		Whereas the thing defined by (Weber 1997, p. 37) is a real-world phenomenon on the instance level, the corresponding construct in the eEPC, the "business relevant object", is always modeled on a type level.
	Organizational unit, position, user, product catalogue, product model, bill of material (types) (see Green and Rosemann 2000, p. 81) To summarize all these kinds of objects of an eEPC diagram we introduce the term "business relevant object" (see Green and Rosemann 2000, p. 81)	
State of a thing (instance)		The state of a thing is described by all values of attributes which are assigned to the thing (Weber 1997, p. 37). The input state defines the state of the thing, before any transformation or transfer function is conducted. The output state defines the state of a thing, after the transformation or transfer function has been performed. Business relevant objects are further specialized as data objects (client, supplier, product, order, certificate, etc.). They are the fundamental part of the data view, and represent entity or relationship types in a corresponding entity relationship model (Scheer et al. 2005, p. 132). Therefore they are modeled on type level. Attributes describe the properties of data objects types, but are not displayed in the eEPC diagram.
	Data object type	
State variable		A state variable is an attribute which describes one property of a thing.
	Event type	The event type indicates a circumstance at a definite point in time, and is represented by changes of a business relevant object (Scheer 2001, p. 127). The event type either depicts the change of state for a business relevant object, after the function has been conducted, or the trigger for performing the following function (Scheer et al. 2005, p. 127).
Transformation, transfer function		The transformation depicts the change of the state of the system triggered by an internal or external event (Weber 1997, p. 50). The transfer function depicts the change of the totality of inputs of a system to output (Weber 1997, p. 51).
	Process, function type (see Green and Rosemann 2000, p. 81)	

the eEPC. An ontological analysis is necessary for transferring the decomposition conditions to business process modeling (Sects. 4.1 to 4.5). To this aim, a short introduction to important terms of both areas (BWW and eEPC), which have similar meanings or are related to one another, will be given first (**Table 2**). The terms presented in **Table 2** will be of crucial importance when specifying the decomposition conditions in Sect. 4.

The eEPC describes a process which is displayed as a diagram (the process model) and is further decomposed to several function types (Green and Rosemann 2000, p. 82). It is important to mention that Wand and Weber argue on the instance level. Thus a thing represents a real-world phenomenon with specified values for all its attributes. In contrast, the eEPC describes processes on a type level (Green and Rosemann 2000, p. 82). A function type is a classification of functions on instance level which conducts the same operation on an instance of a certain data object type. Therefore the function type is described by the operation and the data object type, while the data object type is specified by its attributes and the corresponding domains. As a consequence, values of attributes are disregarded on a type level.

The emphasis on "things" or "function types" implies another important distinction. Whereas Wand and Weber decompose the system into subsets of "things", the decomposition in eEPC defines subsets of function types. Subprocesses and the process within a decomposition are related, while, in most cases, part-of-relations and sometimes specialization-relations are given (see Malone et al. 1999, pp. 427–432).

Similar to Wand and Weber, the execution of a function in an eEPC results in a change of the values of the attributes of one or more data objects. The execution of the function is represented with at least three modeling constructs (event type → function type → event type) and can be interpreted with the change of the state of the thing. But as already mentioned the eEPC diagram represents its constructs only on type level. In addition, attributes of the data object type are not displayed. Therefore it is not visible in an eEPC diagram which values of the attributes of the data object type were changed. Even though the sequence of "event type → function type → event type" implicitly includes the data object type, as well as the attributes on which the operation of the function type is conducted, the data object type is often explicitly pictured in an eEPC diagram.

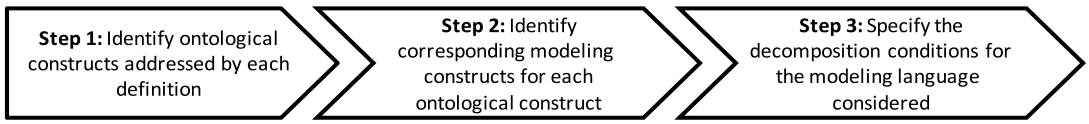

Fig. 2 Procedure for specifying the decomposition conditions

3.4 Procedure for Specifying the Decomposition Conditions

The introduction of divergent terms – used by the eEPC and the BWW models – in Sect. 3.3 indicates major differences between these concepts. In this regard, a deeper analysis of the decomposition conditions has to be done for transferring the conditions to eEPCs. Therefore we consider the results of the ontological analysis of eEPCs that was already performed by Green and Rosemann (2000, 2001). **Figure 2** presents the steps we adhere to for deriving our interpretation.

Step 1: At first, the decomposition conditions are introduced as they were originally defined. Then we analyze which ontological constructs of the BWW representation model are addressed by each condition.

Step 2: The completeness of the eEPC regarding the BWW constructs was investigated by Green and Rosemann (2000, 2001). We refer to the results of this representation mapping respectively clarity analysis (Rosemann et al. 2004, p. 116) to identify corresponding modeling constructs (of the eEPC) regarding the ontological constructs that were identified in step 1.

Step 3: Based on these findings, we derive an interpretation of the decomposition conditions for a decomposed eEPC process model.

By that procedure, we assure that our interpretation of the decomposition conditions for business process models sticks close to the basic ideas of Wand and Weber. To mitigate subjectivity, this procedure was conducted by two researchers independently. Afterwards the results were compared and the final interpretation (Sect. 4) was derived.

4 The Decomposition Conditions

In this section, we introduce our interpretation of the decomposition conditions for eEPCs. We stick to the procedure presented in **Fig. 2**. As mentioned above, the decomposition model (Wand and Weber 1989, 1990; Weber 1997) comprises five conditions: (1) minimality, (2) determinism, (3) freedom of losslessness, (4) minimum coupling, and (5) strong cohesion.

4.1 Minimality

Step 1: According to Weber (1997, p. 153) a decomposition "is good only if for every subsystem at every level in the level structure of the system there are no redundant state variables describing the subsystem".

In simple words: this means that every subsystem in a decomposition should be described with the minimum number of attributes which are needed to specify the states and events occurring in the subsystem (Weber 1997, p. 153). In this definition, four ontological constructs of the BWW representation model are mentioned: *system*, *subsystem*, *level structure*, and *state variable*.

Step 2: Following the ideas of Green and Rosemann (2000, p. 81) as well as Weber (1997, p. 33), the "system" and the "subsystem" are a set of organizational constructs or constructs of the output view "acting on" each other (Weber 1997, p. 42). For example, a "salesperson", a "receiving clerk" as well as a "product item" can form a "system" (Rosemann and Green 2002, p. 84). These "things" (e.g., salesperson) take part in a specific business process, while they can stem from different systems or subsystems. According to Weber (1997, p. 45), a subsystem itself is a "system whose composition and structure are subsets of another system". The subsystems are arranged within a level structure (Weber 1997, p. 46), which is indicated by corresponding model levels and decomposition indicators according to Green and Rosemann (2000, p. 81).

A further construct of the minimality condition as defined by Weber (1997, p. 153) is the notion of a "state variable". An example for a state variable within a (sub-)system is the "order status" for a "customer" who is a thing in that context. According to Green and Rosemann (2000, pp. 81–82) as well as Hoffmann et al. (1993, p. 6), event types in the EPC are used for representing the values of corresponding state variables (attributes). Attention has to be given to the notion of an "event", which is different for eEPCs and the BWW representation model (Green and Rosemann 2000, p. 82). According to Keller et al. (1992, p. 10), an "event" in an eEPC diagram represents a state of an information system. In the BWW representation model an "event" describes an "ordered pair" of states instead, while the change of state is performed by a transition (Weber 1997, p. 41).

A state variable (attribute) is needed, and it is therefore not "redundant", in the sense of Weber (1997, p. 153), when there is a change of "state" for that variable (attribute) throughout the history (Weber 1997, p. 42) of the (sub-)system. This means the value of the state variable (attribute) must at least change once during the lifetime of the (sub-)system. But as Weber (1997, p. 154) indicates, a state variable (attribute) may also be "needed", when it enables a change of state for another variable (attribute), while its own state never changes.

Step 3: Regarding eEPC models we conclude: The data object types in a data model representing "things" (Weber 1997, p. 34) that take part in a business process model are characterized by attributes, while an event type within an eEPC describes the occurrence of a "state" for those attributes (Keller et al. 1992, p. 11; Hoffmann et al. 1993, p. 6). If an event type within an eEPC model refers to an attribute of a "thing" (Weber 1997, p. 44) that takes part in a business process while its value never changes during process execution, this event type is "redundant" in the sense of Weber (1997, p. 153). During runtime such "events" are never reached and are irrelevant for the continuation of the process according to Scheer et al. (2005, p. 127). Thus their existence violates the minimality condition. Several aspects need to be consid-

ered when analyzing a process model regarding minimality in the sense of Weber (1997, p. 153).

- Initially the business analyst should refer to the data model holding data object types which represent "things" in the sense of Weber (1997, p. 44). This approach is supported by Hoffmann et al. (1993, p. 6), emphasizing that potential values of the attributes, represented in the attribute domain of a data object type, indicate which event types are needed for an eEPC model. Consider a "handling of customer order" process as an example which deals with the processing of a customer order. The customer is characterized by an attribute "order status". The value of the "order status" may be "waiting to be served" at the beginning and change to "order completed" during process execution. Since the state changes and the customer is a "thing" dealt with in the "handling of customer order process", the attribute (state variable) is "needed" in the sense of Weber (1997, p. 153). Thus within an eEPC process model on the type level (Green and Rosemann 2000, p. 82), two event types (labeled as "customer waiting to be served" and "order completed" for example) would be modeled to express this circumstance. A corresponding function type (labeled as "process customer order" for example) indicates the "transformation" (Green and Rosemann 2000, p. 81) in the eEPC model.
- Furthermore attributes whose values never change throughout the lifetime of a process but are necessary for other attributes to change their values (Weber 1997, p. 154) are to be considered. In such a case the event types of an eEPC model referring to the former attributes are relevant for the process model as well and "not redundant" in the sense of Weber (1997, p. 154). It has to be mentioned that the identification of such event types is rather complex on a type level, and requires deep knowledge of the real-world situation to be modeled. Such an event type could refer to the existence of an "efficient CRM system", for example, which is supposed to outlive the existence of a "handling of customer order process".
- Finally the business analyst should be aware that also event types that do not refer to certain data object types of a data model may be necessary for an eEPC model (Hoffmann et al. 1993, p. 6). According to Hoffmann et al. (1993, p. 6), such event types (e.g., "dataset is locked") can be essential for specifying a (sub-)system more precisely, and thus are "non-redundant" in a process model as well. However, the identification of corresponding event types is rather subjective and needs some sort of consensus of the model users in the sense of Schütte and Rotthowe (1998, pp. 245–246).

After having identified "redundant" (see Weber 1997, p. 153) event types within a process model, the focus should now be on modeling constructs that are related to those event types. In case the eEPC model holds event types that are "redundant" (as explained above), the corresponding function types representing a transition (that will never occur during process execution) become "redundant" regarding the eEPC model, too. In addition, the function types may have relations (information flow, organization flow, etc.) to constructs of other ARIS views in a process model (see Scheer et al. 2005, pp. 120–132), which may become redundant and thus violate the minimality condition as well. Take **Fig. 1** as an example. If the event types "information incomplete" and "missing information received" turn out to be "redundant" in the sense of Weber (1997, p. 153), also the function type "ask customer for missing information", the symbol representing the organizational unit "customer service" as well as the corresponding organizational flow (e.g., Scheer et al. 2005, p. 123) become redundant. Therefore, referring to the representation of a process as an eEPC model, not only event types and function types but also other constructs, stemming from different views of the ARIS framework, must be analyzed, whether they are needed any longer (see Weber 1997, p. 153).

In that context, two aspects need to be considered: Firstly, this approach needs a data model considering the relevant attributes. Secondly, identifying attributes whose values change at least once during the lifetime of the process always bears uncertainty and requires deep knowledge of the real-world situation to be modeled on type level (Green and Rosemann 2000, p. 82).

In general, the avoidance of redundancy is also demanded by authors such as Mendling et al. (2010, p. 130) or Schütte and Rotthowe (1998, p. 246), for example.

4.2 Determinism

Step 1: Weber (1997, p. 154) defines the determinism condition as follows: "*For a given set of external (input) events at the system level, a decomposition is good only if for every subsystem at every level in the level structure of the system an event is either (a) an external event, or (b) a well-defined internal event*".

The definition addresses two major constructs of the BWW representation model that have not been mentioned yet: *external event* and *internal event*.

Internal events occur by transformations that are internal to the system and not triggered by things in the environment (Weber 1997, p. 50). They are supposed to be well-defined. If knowledge concerning the prior state is available, it is then possible to predict the subsequent state that will arise (Weber 1997, p. 155). External events, however, lead to a change of state of a system because of environmental components (Weber 1997, p. 50).

Step 2: As becomes obvious by the research of Green and Rosemann (2000, pp. 81–82), a change of state is represented by the triple "*event type* → *function type* → *event type*". The function type indicates the transition (Green and Rosemann 2000, p. 81). In that context, also the BWW construct "state law" has to be considered. The "state law" assures that only states of a thing are considered that are realistic (Recker et al. 2009, p. 361). As Green and Rosemann (2000, p. 81) indicate, the connectors (AND, XOR, OR) of an eEPC are used to constrain changes of states to those that are realistic in the sense of Weber (1997, p. 40). Since internal events should be well-defined, it is to be demanded that no uncertainties occur on the state a process may take after the execution of the function.

In addition, the definition of Weber (1997, p. 154) speaks of "external events". These lead to a change of state within a system because of environmental influences, for example a server crash at a supplier (Weber 1997, p. 154). According to Green and Rosemann (2000, p. 81), external events are represented in an eEPC as start event types. Assuming that the process is a subsystem of the information system, an external event can therefore be the "arrival of a customer order" for example, as long as the customer is not part of our subsystem (see Weber 1997, p. 43). In that case, the "customer order" may

trigger a "receive customer order" subprocess. From that point of view several subprocesses can be designed while each subprocess is triggered by its "environment" (customer, supplier, etc.). Another subprocess, "request handling", may thus resolve from a "customer request".

Step 3: Therefore the following main results regarding eEPC models can be derived:

First of all, conditions specifying which events follow a function at a decision node during process execution (XOR, OR) need to be precisely defined. These conditions are represented by an appropriate labeling of the event types of an eEPC model, for example (see Davis 2001, p. 121). In that context, Decker and Mendling (2009, p. 779) recall that the label of an event type only represents "informal information".

Authors such as Mendling et al. (2010, p. 130), Kindler (2006), Davis (2001, p. 122), or van der Aalst et al. (2002) recommend avoiding OR-connectors, even though not in the context of decomposition. It has been shown that the use of OR-Splits may result in process designs in which the states following the performance of a function are hard to foresee and may cause complications during process execution (see van der Aalst et al. 2002; Mendling et al. 2010, p. 130).

Thus, a good decomposition is characterized by a thorough specification of conditions enabling a proper execution of the process. Actually the accurate specification of conditions within an eEPC model is not a quality characteristic usually associated with "decomposed process models" in special. Nevertheless, this requirement results from a thorough specification of the determinism condition (see Weber 1997, p. 154) for the context at hand. As a supplementary support for a business analyst, decision tables (see Balzert 2009, pp. 386–404) may be used to mitigate uncertainty regarding the state a process takes during execution.

Secondly, subprocesses are built around external events (Weber 1997, p. 154) which are represented as start event types in the corresponding subprocess models (Green and Rosemann 2000, p. 81). If we assign customers or suppliers to the "environment" in the sense of Weber (1997, p. 43), a similar idea can be found in Österle (1995, pp. 93–94), for example. Following Decker and Mendling (2009, pp. 780–781), the condition triggering the instantiation of the process is to be made explicit in the label of a start event type, while single (one start event triggers instantiation) and multi event triggers (multiple start events trigger instantiation) can be differentiated in that context.

4.3 Freedom of Losslessness

Step 1: According to Weber (1997, p. 155) "*a decomposition is good only if every hereditary state variable and every emergent state variable in a system is preserved in the decomposition*". Properties of a thing must not get lost during decomposition (Weber 1997, p. 155).

The major constructs of the BWW representation model that are addressed in this definition and have not been mentioned yet are: *hereditary* and *emergent state variable*. While a hereditary property is possessed by a "thing" (as well as at least one of its components) (Weber 1997, p. 37), an emergent property represents a property which only becomes obvious when considering the "thing" (including all lower level systems) as a whole (Weber 1997, p. 37). The idea behind the principle is that no meaning is supposed to get lost when a system is decomposed (Weber 1997, p. 158).

Step 2: The freedom of losslessness condition as described by Weber (1997, pp. 155–158) is more of an issue with database decomposition (see Burton-Jones and Meso 2006, p. 42) than it is with the decomposition of business process models. According to Green and Rosemann (2000, p. 81), there are representation mechanisms for properties in eEPCs in general, but not for hereditary properties in particular (Green and Rosemann 2000, p. 81; Recker et al. 2009, p. 339).

Regarding our work at hand, the "IT expertise" of a company can be seen as a hereditary property in the sense of Weber (1997, p. 37), for example, since it is possessed by both the enterprise itself and the subset of that company's IT-employees. The attribute used for expressing this property (see Weber 1997, p. 34) must be attached to both the company and the IT-employees in a corresponding data model. The fact that a property actually addresses a hereditary property must be derived from the context.

Regarding emergent properties, Weber (1997, pp. 156–158) admits that their presence is not always obvious, since they are not possessed by a lower level system or subsystem. They can only be recognized when a deeper understanding of the real-world situation is gained (Weber 1997, p. 156). In the context of a "customer order handling process", an emergent property could be a "customer's level of satisfaction" with a company based on multiple interactions with that company in the past. The customer's level of satisfaction is determined by her/his interaction with different contact points of that company such as the "order processing", "customer service", or "after-sales support" for example. As Weber (1997, p. 156) mentions, an "emergent property" usually is related to "lower-level properties", for example customer's expectations regarding product maintenance in the "after sales support". But the emergent property is not readily apparent when considering singular subprocesses only. Green and Rosemann (2000, p. 81) give further examples for an emergent property such as "cost type" or "cost rate".

In the works of Moody and Flitman (2000, p. 464) dealing with data model decomposition and Burton-Jones and Meso (2006, p. 44) focusing on the decomposition of UML diagrams, the specification of "hereditary" and "emergent" state variables falls short. This can be explained, since Weber (1997, p. 156) broadens his definition of the freedom of losslessness condition by demanding "not to lose properties" of a thing that is decomposed in general. In that context, he addresses all types of properties (see Weber 1997, pp. 35–37) that should be preserved during decomposition, while the preservation of emergent and hereditary properties is most challenging.

Step 3: Regarding the specification of the freedom of losslessness condition for an eEPC model, the following conclusion can be drawn:

- During the decomposition no property should get lost according to Weber (1997, p. 156). Properties are expressed by attributes (Weber 1997, p. 34) in a corresponding data model. For evaluating an eEPC process model regarding freedom of losslessness all attributes (state variables) that are "non-redundant" in the sense of Weber (1997, p. 153) (Sect. 4.1) must be preserved in the subprocess models of a decomposed eEPC model. In Sect. 4.1, it has been stated that the attributes

can be related to event types, while a change of state for an attribute is indicated by the triple "event type → function type → event type" (Green and Rosemann 2000, p. 81). Thus all "non-redundant" attributes must be considered by a corresponding arrangement of event types, respectively function types (in the sense of Green and Rosemann (2000, p. 81)). This approach matches the decomposed eEPC model with a data model, describing the things and their properties represented by attributes.

- However, if an original eEPC model exists that has been decomposed, a matching between these models (before and after the decomposition) can be performed without considering the data model. Thus the user has to analyze whether (1) all function types representing tasks to perform the original process can be found in corresponding subprocesses, (2) all possible "states" of the original process (indicated by event types in eEPCs) are captured in the decomposed process model and (3) all relations (e.g., organization flow, information flow) between the tasks to be performed and constructs from the other ARIS views have been preserved. Consider **Fig. 1** for example. If a subprocess is built around the triple (event type "information incomplete" → function type "ask customer for missing information" → event type "missing information received"), information is lost when the responsibility of the "customer service" is no longer depicted within the decomposed eEPC model.

By considering the approaches mentioned above, a business analyst can verify that no meaning has been lost during decomposition (Weber 1997, p. 158), and the business logic has been preserved in a semantic sense (Reijers et al. 2011, p. 884). At this point, similarities to Schütte and Rotthowe (1998, p. 245) become obvious, since the authors demand a consensus "regarding the presented problem". In addition, relations can be drawn to the "semantic quality" in the sense of Krogstie et al. (1995, p. 90). Thus it is demanded that all relevant properties of the real-world are considered in the model (Krogstie et al. 1995, p. 90).

4.4 Minimum Coupling

Step 1: According to Weber (1997, p. 161) "a decomposition has minimum coupling iff the cardinality of the totality of input for each subsystem of the decomposition is less than or equal to the cardinality of the totality of input for each equivalent subsystem in the equivalent decomposition".

Thus the minimum coupling condition requires a "minimum cardinality" of the "totality of the input" (Weber 1997, p. 161). The condition compares two different decompositions of a system – which must have the same components to be considered as equivalent – and identifies the strength of coupling (Weber 1997, p. 161). The intention is to minimize "the total action of all environmental things on each subsystem in the decomposition" (Weber 1997, p. 159).

Coupling, *subsystem*, and *input* are major constructs addressed by this definition. The "input" of a system, as already mentioned, consists of those states which arise within a system by the actions of things in the environment (Weber 1997, p. 49). The notion of "subsystem" has already been explained in Sect. 4.1. "Coupling" means that things (e.g., employees) are associated with one another (Weber 1997, p. 42).

Step 2: First, there is no modeling construct for coupling in the process view (Green and Rosemann 2000, p. 81), and there is no clear indication on how to model "input" either. In addition, the definition of the boundaries (comprising "things" such as organizational units, products, etc.) that make up a system or subsystem in the sense of Weber (1997, p. 44) does not match up with the primary ideas of the eEPC (Green and Rosemann 2000, p. 83).

Step 3: Specifying this condition for the paper at hand is challenging. Only the sense of the condition – "minimizing the total action of all environmental things on each subsystem in the decomposition" (Weber 1997, p. 159) – can be transferred. Assuming that the process is a subsystem of the information system, the decomposition of the process into subprocesses should be designed in such a way that there is minimal interaction (Vanderfeesten et al. 2007a, 2007b, p. 179) between them. The interactions between subprocesses in eEPCs are described by external event types (they are the start event types according to Green and Rosemann (2000, p. 81) and refer to an attribute) and by input data object type(s). Transferring the idea of Wand and Weber to process modeling, an input data object type is specified, because an environmental data object type "acts on" it. The specific action is described by an environmental function type. Therefore, according to Wand and Weber, an input data object type is at the same time an output data object type of an environmental function. Whereas Wand and Weber try to minimize the number of input states of the thing (which are affected by an environmental thing) in order to reduce interaction between the subsystems, within an eEPC model it is only possible to reduce the number of input data object types and external event types. With this interpretation the coupling condition had to be adapted for eEPCs and is slightly different from the original definition by Weber (1997, p. 161). This is due to the fact that modeling with eEPCs is only possible on type level, and additionally, attributes of an input data object type cannot be specified in an eEPC. Therefore a one-to-one mapping of Wand and Weber's idea, namely to count the number of all input states of the thing, is not possible. Instead only the numbers of input data object types and external event types can be minimized regarding an eEPC model. This difference with regard to the original definition of Weber (1997, p. 161) can possibly lead to misinterpretations. But from our point of view it is the only possible way to transfer the minimum coupling condition to process modeling with eEPCs.

Following this idea, subprocesses are "coupled" when they share common data object types (Vanderfeesten et al. 2007a, 2007a, p. 182) or external event types. In the following, we try to exemplify our interpretation of the condition (**Fig. 3**). In the example, we have a set of three subprocesses resulting from a decomposition: subprocess 1, subprocess 2, and subprocess 3[2]:

As we can see, the subprocesses interact with each other by exchanging data object types. "Data object type 2" serves as input for function type "D" in subprocess 2. "Data object type 3" instead is the output of function type "F", and serves

[2] For reasons of comprehensibility we omitted the intermediate event types.

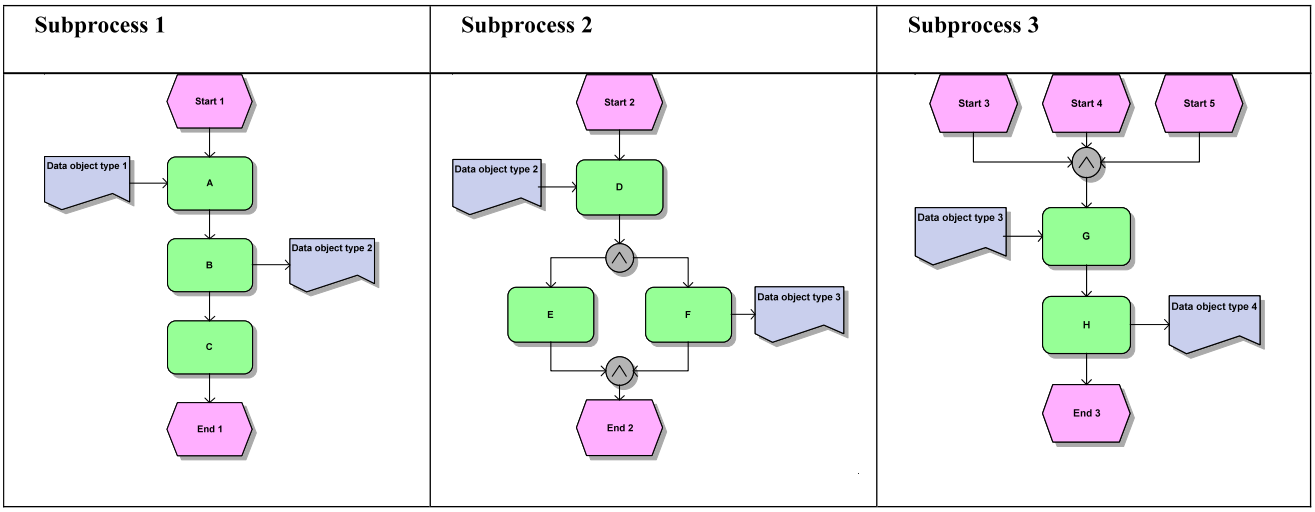

Fig. 3 Example for the minimum coupling condition

as input for function type "G" in subprocess 3. The example demonstrates that all subprocesses will have at least one external event type, which is the start event type. Following the idea of Weber (1997, p. 158), the interaction between subprocess 2 and 3 can be minimized by merging them. In addition, this would reduce the number of external event types too, while the event types "start 3", "start 4" and "start 5" are merged with "end 2". At that point, it has to be mentioned that – following the ideas of Weber (1997, p. 162) – merging the subprocesses and reducing coupling might involve having weak internal cohesion within the result.

We assume that our example represents the complaint management process (as mentioned in Sect. 2.1), and subprocess 2 aims to consolidate the information received from the customer on the one hand, and the information received from the corresponding internal department of the fleet management company on the other hand. In subprocess 3, the complaint is actually handled. As a precondition the contract with the customer must be valid ("start 3"), the information about the complaint must be complete ("start 4"), and information about the customer (e.g., basic claims data) must be available ("start 5"). The three event types result from performing the function types of subprocess 2. By merging subprocesses 2 and 3, the three external event types would replace the event type ("end 2") and thus become internal event types.

In the example above (**Fig. 3**), merging subprocesses 1 and 2 would be appropriate regarding our specification of the "minimum coupling condition" as well, since these interact by exchanging "data object type 2".

Designing subprocesses without having an interchange of data object types between them is not reasonable in all situations. For example, imagine a "complaint" (data object type) regarding our complaint management process. The complaint may be used by a lot of different function types of that process (e.g., "send acknowledgment of receipt" or "send solution"). In such a situation, it is hard to design the subprocesses that have "minimum coupling" and therefore little interchange of data object types. In the worst case, a decomposition with the least coupling degree comprises only one resulting process model (Weber 1997, p. 161).

Weber (1997, p. 161) mentions that a fair basis for comparability needs to be established. Using this condition in our research, it is therefore necessary that the alternative decompositions represent the same real-world phenomena (Burton-Jones and Meso 2006, p. 39) and consider the same tasks to be performed.

The following conclusion can be drawn: Subprocesses of a process have "minimal coupling" if each of them possesses less input data object types and external event types than in any other comparable decomposition of the same process. Due to the lack of ontological expressiveness of the eEPC regarding the constructs of the BWW representation model, our interpretation comes close to the original ideas of Weber (1997, p. 161), but cannot be adapted directly. Since coupling cannot be addressed by separate modeling constructs in the process view (Green and Rosemann 2000, p. 81; 2001, p. 32), we focus on input data object types and external event types to explain the "interaction" phenomenon (Weber 1997, p. 160).

4.5 Strong Cohesion

Step 1: The last condition of the model according to Weber (1997, p. 163) concerns strong cohesion. In this context the definition of Dromey (1996, p. 42) is referred to. Therefore "a set of outputs is maximally cohesive if all output variables affected by input variables are contained in the same set, and the addition of any other output to the set does not extend the set of inputs on which the existing outputs depend and there is no other output which depends on any of the input set defined by the existing output set" (Dromey 1996, p. 42).

The constructs of the BWW representation model that are highlighted by this definition, and have not been mentioned yet, are *output* and *cohesion*. The output characterizes a state that arises in a thing outside the system considered (Weber 1997, p. 51), while cohesion is seen as the "dual" to coupling (Weber 1997, p. 161).

Step 2: Regarding the ontological analysis done by Green and Rosemann (2000, p. 81) there is no separate construct for expressing "cohesion" in eEPCs, while the modeling of "output" is likewise unclear. Thus, the specification of the condition is quite challenging once again.

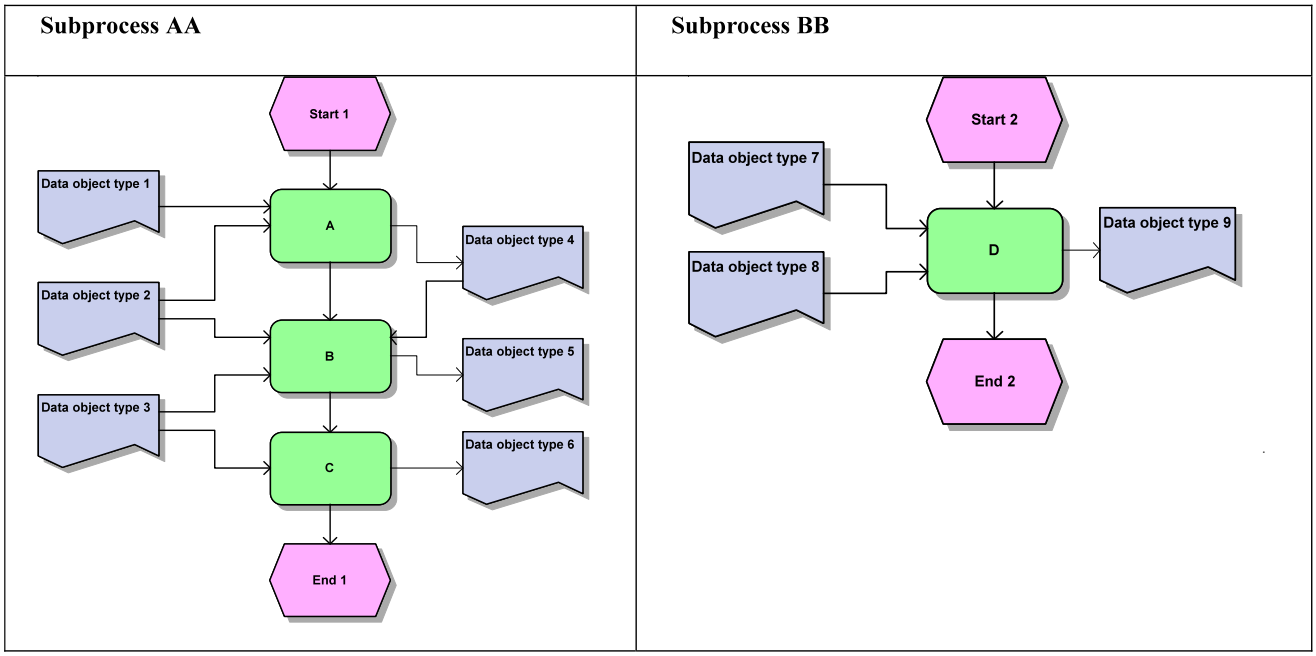

Fig. 4 Example for the strong cohesion condition

Step 3: Literature has not explicitly dealt with "strong cohesion" in business process models. However Reijers (2003), Reijers and Vanderfeesten (2004) and Vanderfeesten et al. (2008b) develop metrics for measuring cohesion. A strong emphasis is once again placed on data objects which are processed by operations within a function (Vanderfeesten et al. 2008b, p. 421). But due to the above-mentioned ontological deficiencies (eEPCs are modeled on type level, and the state variables are not visible because the attributes of data object types in eEPCs are disregarded) only an interpretation of Wand and Weber's primary ideas can be made. A look at the following example can give a more detailed explanation:

Let us imagine that data object type 1 is the "customer", data object type 2 is the "product", data object 4 is the "customer order" and data object type 5 is the "purchase order". To achieve the "purchase order" the "bill of material" is needed (data object type 3). In addition, based on the "bill of material", material is ordered independently from a customer order (for example all C-materials).

In **Fig. 4** we visualize two subprocesses. As can be seen, all the output data object types in "subprocess AA" depend on the input data object types. Since for the eEPC it is not possible to visualize the different attributes of the data object types, we have to assume that all attributes of an output data object type depend functionally on its input object types. In other words, the possibility that a subset of attributes of an output data object type does not depend functionally on its input data object types is to be excluded from the consideration. This is, of course, a restriction regarding strong cohesion. If this assumption is not valid, additional models have to be designed (e.g., Vanderfeesten et al. 2008b), since the eEPC lacks means for expressing such constellations.

Based on this supposition, "data object type 4" depends on the processing of the input data object types "1" and "2", while "data object type 4", "data object type 3", and "data object type 2" are needed for generating "data object type 5" in **Fig. 4**. Again "data object type 3" is needed for generating "data object type 6". Therefore function types "A", "B", and "C" are merged within one subprocess "AA". We can depict the relation between these data object types in a diagram that is based on the ideas of Scheer et al. (2005, p. 125). It becomes evident that all the data object types in subprocess "AA" are interrelated with one another (**Fig. 5**). A separate subprocess "BB" deals with the data object types "7", "8", and "9" (**Fig. 5**). We assume that these data object types are independent from those in the subprocess "AA". That is why they are considered in an own subprocess.

Things are different, if "data object type 3" would be needed for creating "data object type 9". In that case, the function types could be merged within one subprocess only, since all data object types created as output (data object type 4, 5, 6, 9) would depend on the set of input data object types (data object type 1, 2, 3, 7, 8).

However, the basic idea of our interpretation is the following: Strong cohesion within a subprocess is given, if all function types transforming a set of input to output are captured within a subprocess. Each input data object type within this subprocess cannot be found as input in any other subprocess at the same model level.

4.6 Summary Regarding the Conditions and Critical Acclaim

Table 3 summarizes our findings. A major challenge is to be seen in the ontological deficiencies of the eEPC regarding the BWW models (Green and Rosemann 2000, pp. 81–83; 2001, p. 32).

Minimality Condition This condition mainly focuses on event types within an eEPC model, but has an impact for other modeling constructs, too, as was shown in Sect. 4.1. This is emphasized by literature in business process modeling, demanding the avoidance of redundant modeling constructs for business process models that were not decomposed into subprocess models as well (Mendling et al. 2010, p. 130; Schütte and Rotthowe 1998, p. 246). Admittedly, the requirement of redundant-free models is

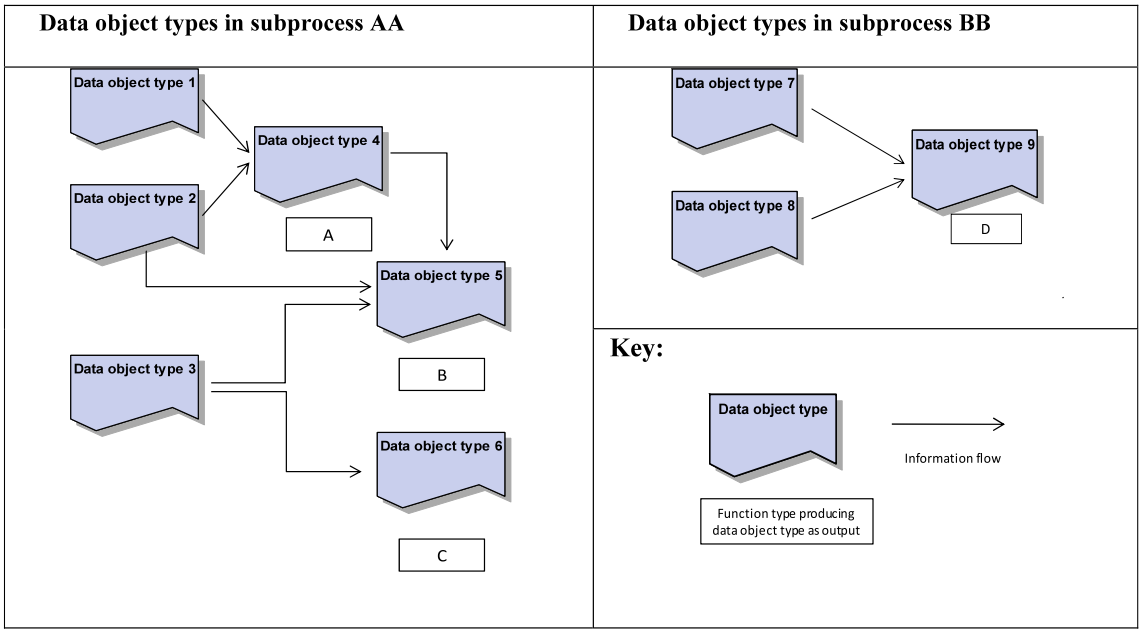

Fig. 5 Relation between data object types

Table 3 Summary of the conditions

Decomposition conditions	Main focus for process modeling
Minimality (Sect. 4.1)	To fulfill the minimality condition, the decomposed eEPC process model should not hold any event types that are "redundant" and thus indicate "states" that never occur during process execution. Event types used for representing states of state variables (attributes) on a type level that are not needed for the continuation of a process on an instance level (see Scheer et al. 2005, p. 127) are generally to be avoided. Function types and modeling constructs from other ARIS views (e.g., of the organizational view, data view, output view) related to these must be reflected upon their necessity regarding this fact as well.
Determinism (Sect. 4.2)	To fulfill the determinism condition, the conditions specifying which event(s) follow(s) the execution of a function during process execution need to be precisely defined. Rules have to be established and the event types to be labeled appropriately. OR-connectors should be avoided. In addition, subprocesses are built around external events.
Freedom of Losslessness (Sect. 4.3)	To fulfill the condition of freedom of losslessness, no information must get lost during decomposition. The event types being related to attributes describing properties play a decisive role and should be preserved during decomposition. Function types associated to these event types together with modeling constructs from other ARIS views must be reflected upon – whether they can be found in corresponding subprocess models of a decomposition or not – as well.
Minimum coupling (Sect. 4.4)	For fulfilling the condition of "minimum coupling", each subprocess of a process must have less input data object types and external event types than in any other comparable decomposition of the same process.
Strong cohesion (Sect. 4.5)	Strong cohesion within a subprocess is given, if all function types transforming a set of input to output (data object types) are captured within a subprocess. Each input within this subprocess cannot be found in any other subprocess at the same model level.

a controversial issue, especially for process models (e.g., Becker 1995, p. 145). The purpose of the process model should always be considered. Thus the condition is not rigorously applicable.

Determinism Condition The requirements are neither new nor do they only arise in the field of decomposition. Similar recommendations are also made by Decker and Mendling (2009), Mendling et al. (2010, p. 130), van der Aalst et al. (2002), or Österle (1995), for example. Whereas the first requirement can be controversially discussed regarding the purpose of the process model, the second requirement supports delineating subprocesses.

Freedom of Losslessness Condition: While properties are to be preserved during decomposition, emergent properties are not always obvious, as Weber (1997, pp. 156–158) admits. The relation of properties to modeling constructs of an eEPC model was explained in Sect. 4.3. Nevertheless, the decision whether the business logic (Reijers et al. 2011, p. 884)

and the initial meaning (Weber 1997, p. 158) of a model are preserved, requires a consensus of the model users and designers. Therefore this aspect bears subjectivity, while similarities to the principle of "construction adequacy" (Schütte and Rotthowe 1998, pp. 245–246) or "semantic quality" (Krogstie et al. 1995, p. 84) are apparent.

Minimum Coupling and Strong Cohesion Condition The original ideas of Wand and Weber for both conditions cannot be directly transferred to eEPCs. In how far these interpretations actually support a business analyst in the context of eEPC modeling (and, in comparison to the original ideas, whether they are more or less useful for the decomposition), requires empirical evidence. Nevertheless, they provide clear instructions for the business analyst and their fulfillment can be easily proven. Therefore they enable the business analyst to improve a decomposed process model.

In summary, the following conclusions can be drawn. The conditions *minimality*, *determinism*, and *freedom of losslessness* address aspects that can be found in existing works on model quality, although not in the spotlight of decomposition (e.g., Mendling et al. 2010, p. 130; Schütte and Rotthowe 1998, p. 246). In addition, the conditions, as they were specified above, do not necessarily deal with aspects inherent to decomposition (e.g., precise specification of conditions regarding event types). Otherwise, the attention that is paid to the abovementioned aspects regarding process model quality additionally stresses the importance of the conditions. Since it is possible to derive requirements from the conditions that increase model quality (**Table 3**), the conditions definitely support the business analyst in decomposing process models. This holds true although some conditions such as minimality and freedom of losslessness (as we have seen in Sects. 4.1 and 4.3) are challenging to fulfill when modeling with eEPCs. The decomposition conditions "minimum coupling" as well as "strong cohesion" provide clear advice on how to evaluate the decomposition based on the structure of the decomposition and the design of the subprocesses. But the main contribution of the conditions lies in their interrelationships. The most obvious interrelationship concerns "minimum coupling", which tends to merge subprocesses, and "strong cohesion", which tends to delineate subprocesses. The "minimality condition" suggests avoiding redundant event types, whereas the "freedom of losslessness condition" prevents the loss of event types related to relevant attributes. Additionally the "determinism condition" supports the creation of subprocesses around external events, while the "minimum coupling" condition proposes to merge subprocesses to reduce interaction between them. When applying the decomposition conditions in combination, the business analysts are able to balance the single effects each condition has on the process model. Therefore the decomposed model is not imprinted by the effects of one condition (e.g., minimality) only. To substantiate this contribution empirical evidence is needed.

5 Outlook, Limitations and Further Research

While literature provides several approaches for performing the decomposition of business process models (e.g., Reijers et al. 2011; La Rosa et al. 2011; Smirnov et al. 2011, 2010a, 2010b; Reijers and Mendling 2008; Vanhatalo et al. 2007, 2008; Malone et al. 1999; vom Brocke 2006), it is still unclear how to evaluate the quality of a decomposition in an objective way. In literature, Wand and Weber's decomposition model is mentioned for that purpose (Recker et al. 2009, p. 355; Reijers et al. 2011, pp. 882–883; Reijers and Mendling 2008, p. 23), while a corresponding investigation has not been done yet. The potential that is assigned to the decomposition model for establishing criteria to judge the quality of a decomposition (Recker et al. 2009, p. 355) was a major motivation for the research at hand. Our intention for this paper was to specify the decomposition model (Weber 1997) for business process modeling. In doing so, we intended to find out whether the conditions can be helpful for judging the quality of a decomposition or not. Thus this study does not aim to derive further approaches to perform the decomposition, but to find means for evaluating the quality of the results gained.

However, a one-to-one mapping of the decomposition condition for eEPCs is not possible. One reason for this are the ontological deficiencies of the eEPC which, in consequence, lacks a level of detail. Whereas the decomposition model is defined on instance level and regards values and attributes of the things, the eEPC focuses only on types and neglects values and attributes of data objects. A further aspect concerns the emphasis on the structure of an information system (thing) or on the behavior (process). Since we intended to remain close to the original ideas of the BWW models, we built our argumentation on existing results of the ontological analysis for eEPCs and interpreted the conditions for the study at hand.

Despite the different areas of application considered (information systems and business process modeling), we have seen that some of the ideas that are presented in the decomposition conditions are not completely new or surprising. In addition, not all conditions are inherent for the discipline of decomposition, but can also be used for models that were not decomposed. Apart from this, we derived requirements from the decomposition conditions which are not always easy to fulfill, but definitely support the business analyst. In addition, we found out that the main contribution of the decomposition conditions is to enable business analysts to balance the single effects each condition has on the process model.

Another finding by applying Wand and Weber in the context of business process modeling is the increasing significance of the data view. The question arises whether the capabilities of the eEPC (and many other process modeling languages, too) are suitable for the decomposition of the processes or whether the eEPC should be enhanced by further constructs specifying the data view in more detail. In that context, even the development of additional model types to capture data object types in more detail (e.g., Vanderfeesten et al. 2008b, p. 422) and to allow a better mapping of the decomposition conditions could be considered.

Although the conditions strongly emphasize the data view, the intention of a process designer and semantics fall somewhat short. In addition, the conditions neglect characteristics of the process (e.g., the characteristics of the control-flow) which will also have an impact on a good or bad decomposition (see the techniques which are discussed in Sects. 2 and 3.2). Therefore, the potential of the decomposition model for creating an evaluation base for decomposed models (respectively establishing guidelines for de-

composition) has to be further investigated. However it does not seem to be advisable to use the decomposition conditions in isolation to assess the quality of a decomposition.

In further research, we intend to verify our interpretation of the decomposition conditions empirically. Subsequently we intend to specify the conditions for other modeling languages to see whether differences exist to eEPCs. Empirical studies will have to show in how far these conditions can be understood and handled by business analysts for evaluating their own business process models. By means of the collected feedback we hope to be able to specify the conditions more precisely regarding the needs of a practitioner. In a final step, we intend to derive guidelines for a good decomposition based on Wand and Weber's decomposition model.

References

Balzert H (2009) Lehrbuch der Softwaretechnik, 3rd edn. Spektrum, Heidelberg

Becker J (1995) Strukturanalogien in Informationsmodellen: Ihre Definition, ihr Nutzen und ihr Einfluß auf die Bildung von Grundsätzen ordnungsmäßiger Modellierung (GoM). In: König W (ed) Wirtschaftsinformatik 95: Wettbewerbsfähigkeit, Innovation, Wirtschaftlichkeit. Physica, Heidelberg, pp 133–150

Becker J, Thome I, Weiß B, Winkelmann A (2010) Constructing a semantic business process modeling language for the banking sector. Enterprise Modelling and Information Systems Architectures 5(1):4–25

Bobrik R, Reichert M, Bauer T (2007) View-based process visualization. Lecture Notes in Computer Science 4714:88–95

Burton-Jones A, Meso P (2002) How good are these UML diagrams? An empirical test of the Wand and Weber good decomposition model. In: Proc 23rd international conference on information systems (ICIS), pp 101–114

Burton-Jones A, Meso P (2006) Conceptualizing systems for understanding: an empirical test of decomposition principles in object-oriented analysis. Information Systems Research 17(1):38–60

Burton-Jones A, Meso PN (2008) The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. Journal of the Association for Information Systems 9(12):748–802

Davis R (2001) Business process modeling with ARIS. Springer, London

Davis R, Brabänder E (2007) ARIS design platform: getting started with BPM. Springer, London

Decker G, Mendling J (2009) Process instantiation. Data & Knowledge Engineering 68(9):777–792

Dromey G (1996) Cornering the chimera. IEEE Software 13(1):33–43

Fettke P, Loos P (2007) Ontological evaluation of Scheer's reference model for production planning and control systems. Journal of Interoperability in Business Information Systems 2(1):9–28

Green P, Rosemann M (2000) Integrated process modeling: an ontological evaluation. Information Systems 25(2):73–87

Green P, Rosemann M (2001) Ontological analysis of integrated process models: testing hypotheses. Australasian Journal of Information Systems 9(1):30–38

Gruhn V, Laue R (2007) Approaches for business process model complexity metrics. In: Abramowicz W, Mayr HC (eds) Technologies for business information systems. Springer, Heidelberg, pp 13–24

Heinrich B, Henneberger M, Leist S, Zellner G (2009) The process map as an instrument to standardize processes: design and application at a financial service provider. Information Systems and E-Business Management 7(1):81–102

Hoffmann W, Wein R, Schweer A-W (1993) Konzeption eines Steuerungsmodells für Informationssysteme – Basis für die Real-Time-Erweiterung der EPK (rEPK). IWI-Heft 106, Institut für Wirtschaftsinformatik, Universität Saarbrücken

Keller G, Nüttgens M, Scheer A-W (1992) Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). IWI-Heft 89, Institut für Wirtschaftsinformatik, Universität Saarbrücken

Kindler E (2006) On the semantics of EPCs. Data & Knowledge Engineering 56(1):23–40

Krogstie J, Lindland OI, Sindre G (1995) Towards a deeper understanding of quality in requirements engineering. In: Proc 7th conference on advanced information systems engineering (CaiSE '95), pp 82–95

La Rosa M, Wohed P, Mendling J, ter Hofstede AHM, Reijers HA, van der Aalst WMP (2011) Managing process model complexity via abstract syntax modifications. IEEE Transactions on Industrial Informatics 7(4):614–629

Malone TW, Crowston K, Lee J, Pentland B, Dellarocas C, Wyner G, Quimby J, Osborn CS, Bernstein A, Herman G, Klein M, Donnell EO (1999) Tools for inventing organizations: toward a handbook of organizational processes. Management Science 45(3):425–443

Mendling J (2008) Metrics for process models – empirical foundations of verification, error prediction, and guidelines for correctness. Springer, Heidelberg

Mendling J, Reijers H, van der Aalst W (2010) Seven process modeling guidelines. Information and Software Technology 52(2):127–136

Mendling J, Reijers HA, Cardoso J (2007) What makes process models understandable? Lecture Notes in Computer Science 4714:48–63

Mesarovic MD, Macko D, Takahara Y (1970) Theory of hierarchical, multilevel, systems. Academic Press, New York

Moody DL, Flitman AR (2000) A decomposition method for entity relationship models: a systems theoretic approach. In: Proc 1st international conference on systems thinking in management, pp 462–469

Österle H (1995) Business Engineering – Prozesse- und Systementwicklung – Band 1: Entwurfstechniken, 2nd edn. Springer, Heidelberg

Paulson D, Wand Y (1992) An automated approach to information systems decomposition. IEEE Transactions on Software Engineering 18(3):174–189

Polyvyanyy A, Smirnov S, Weske M (2008) Process model abstraction: a slider approach. In: EDOC 2008, pp 325–331

Abstract

Florian Johannsen, Susanne Leist

Wand and Weber's Decomposition Model in the Context of Business Process Modeling

Whereas the benefits of decomposing process models are obvious, the question what actually characterizes a "good" decomposition of a business process model has been given little attention to date. In addition, the process of decomposition itself is considered as being an "art" in literature. Our approach for achieving a "good" decomposition is Wand and Weber's decomposition model for information systems. As a first step in our investigation we aim to explore in how far the decomposition model can be adapted for business process modeling at all. The potential this model might bear for evaluating decompositions of process models has been promoted in literature quite often, while a corresponding investigation is still missing. We address this gap by the following research. In the long term, we intend to establish guidelines for decomposing business process models in a structured way.

Keywords: Decomposition, Process model, Event-driven process chain

Recker J, Indulska M (2007) An ontology-based evaluation of process modeling with Petri nets. Journal of Interoperability in Business Information Systems 2(1):45–64

Recker J, Indulska M, Rosemann M, Green P (2005) Do process modeling techniques get better? A comparative ontological analysis of BPMN. In: Proc 16th Australasian conference on information systems, Sydney

Recker J, Rosemann M, Krogstie J (2007) Ontology- versus pattern-based evaluation of process modeling languages: a comparison. Communications of the Association for Information Systems 20(48):774–799

Recker J, Rosemann M, Indulska M, Green P (2009) Business process modeling: a comparative analysis. Journal of the Association for Information Systems 10(4):333–363

Reijers HA (2003) A cohesion metric for the definition of activities in a workflow process. In: Eighth CAiSE/IFIP8.1 international workshop on evaluation of modeling methods in systems analysis and design, pp 116–125

Reijers HA, Mendling J (2008) Modularity in process models: review and effects. Lecture Notes in Computer Science 5240:20–35

Reijers HA, Vanderfeesten ITP (2004) Cohesion and coupling metrics for workflow process design. Lecture Notes in Computer Science 3080:290–305

Reijers HA, Mendling J, Dijkman RM (2011) Human and automatic modularizations of process models to enhance their comprehension. Information Systems 36(5):881–897

Rosemann M, Green P (2000) Integrating multi-perspective views into ontological analysis. In: Proc 21st international conference on information systems, Brisbane, Australia, pp 618–627

Rosemann M, Green P (2002) Developing a meta model for the Bunge–Wand–Weber ontological constructs. Information Systems 27(2):75–91

Rosemann M, Green P, Indulska M (2004) A reference methodology for conducting ontological analyzes. Lecture Notes in Computer Science 3288:110–121

Rosemann M, Green P, Indulska M, Recker JC (2009) Using ontology for the representational analysis of process modeling techniques. International Journal of Business Process Integration and Management Decision 4(4):251–265

Scheer AW (2001) ARIS – Modellierungsmethoden – Metamodelle – Anwendungen. Springer, Berlin

Scheer AW, Thomas O, Adam O (2005) Process modeling using event-driven process chains. In: Dumas M, van der Aalst W, Hofstede AT (eds) Process-aware information systems. Wiley, New Jersey, pp 119–146

Schütte R, Rotthowe T (1998) The guidelines of modeling – an approach to enhance the quality in information models. Lecture Notes in Computer Science 1507:240–254

Smirnov S, Dijkman R, Mendling J, Weske M (2010a) Meronymy-based aggregation of activities in business process models. Lecture Notes in Computer Science 6412:1–14

Smirnov S, Weidlich M, Mendling J (2010b) Business process model abstraction based on behavioral profiles. Lecture Notes in Computer Science 6470:1–16

Smirnov S, Reijers HA, Weske M (2011) A semantic approach for business process model abstraction. Lecture Notes in Computer Science 6741:497–511

van der Aalst WMP, Desel J, Kindler E (2002) On the semantics of EPCs: a vicious circle. In: EPK 2002 – business process management using EPCs, Trier, pp 71–80

Vanderfeesten I, Cardoso J, Reijers HA (2007a) A weighted coupling metric for business process models. In: Proc of the CAiSE 2007, pp 41–44

Vanderfeesten I, Reijers HA, Mendling J, van der Aalst WMP, Cardoso J (2008a) On a quest for good process models: the cross-connectivity metric. Lecture Notes in Computer Science 5074:480–494

Vanderfeesten I, Reijers HA, van der Aalst WMP (2008b) Evaluating workflow process designs using cohesion and coupling metrics. Computers in Industry 59(5):420–437

Vanderfeesten ITP, Cardoso J, Mendling J, Reijers HA, van der Aalst WMP (2007b) Quality metrics for business process models. In: Fischer L (ed) BPM and workflow handbook 2007: future strategies, pp 179–190

Vanhatalo J, Völzer H, Koehler J (2008) The refined process structure tree. Lecture Notes in Computer Science 5240:100–115

Vanhatalo J, Völzer H, Leymann F (2007) Faster and more focused control-flow analysis for business process models through SESE decomposition. Lecture Notes in Computer Science 4749:43–55

vom Brocke J (2006) Design principles for reference modeling – reusing information models by means of aggregation, specialisation, instantiation, and analogy. In: Fettke P, Loos P (eds) Reference modeling for business systems analysis. Idea Group, Hershey, pp 47–76

Wand Y, Weber R (1989) A model of systems decomposition. In: Proc 10th international conference on information systems, Boston, pp 42–51

Wand Y, Weber R (1990) Toward a theory of the deep structure of information systems. In: Proc international conference on information systems, Copenhagen, Denmark, pp 61–71

Weber R (1997) Ontological foundations of information systems. Coopers & Lybrand, Queensland

Yourdon E (1989) Modern structured analysis. Prentice Hall, Englewood Cliffs