Evaluating the Expressiveness of Domain Specific Modeling Languages using the Bunge-Wand-Weber Ontology

Jörg Becker  
University of Münster
Philipp Bergener  
University of Münster
Dominic Breuker  
University of Münster
Michael Räckers  
University of Münster

{ joerg.becker | philipp.bergener | dominic.breuker | michael.raeckers }@ercis.uni-muenster.de

Abstract

Business Process Management is becoming an ever more important aspect for organizations alongside with Business Process Diagrams as a tool to describe business processes. So far process modeling has been mainly performed with generic process modeling languages. These approaches have however limitations when it comes to the needs of specific problem domains or automated process analysis. Semantic building block based languages (SBBL) aim to overcome those limitations by integrating domain semantics in the modeling language. However, this class of languages is only useful if they exhibit the same expressiveness as generic languages. In this paper we strive to answer this question by comparing the expressiveness of the SBBL language PICTURE with ARIS as a generic language based on the Bunge-Wand-Weber ontology, showing that PICTURE has hardly construct deficits compared to ARIS while showing less construct redundancy and construct overload in its constructs.

1. Introduction

Business Process Management is moving more and more in the focus of organizations. Business Process Diagrams (BPD) enables them to get a transparent overview over the relevant extracts of the organization. BPDs are used to create clarity about the logical sequence of activities in an organization. They are also applied to describe the resulting products and services, the required resources and data, as well as the involved organizational units. They have been discussed in Information Systems (IS) literature as a tool for a variety of purposes, especially for overall performance evaluation of an organization [22]. However, a semantic analysis of BPDs is necessary to identify relevant information for managerial decision making [10, 11].

So far process modeling has mainly been performed with generic (general-purpose) languages [1, 20]. These modelling languages, such as Activity Diagrams (AD), Business Process Modelling Notation (BPMN), or Event Driven Process Chains (EPC), are flexible instruments to describe diverse processes in many different domains. However, several projects show that it is hard to deal completely with special needs and purposes of a single application domain [6] especially in case of process analysis.

The semantic building block-based approach has its roots in the ontological research on the foundations of conceptual modeling [8, 19, 31, 34]. In this context semantics of a certain application domain is integrated within the concept of Business Process Management on a meta-level. One goal while developing this new concept of modeling was to realize a strong guidance of users and a limited degree of freedom within the modeling activities. We evaluated the concept in terms of usability and comparability [3, 5], however one remaining question is:

Is the class of SBBL as expressive as generic modeling languages, i.e. is it possible to capture same or even more information?

To answer this question in a next step of evaluation cycle the remainder of our paper is as follows: First we introduce the class of SBBL and present an instantiation of that class, PICTURE. Subsequently we present an analysis framework by Bunge, Wand and Weber and then map PICTURE and ARIS as proxy of generic modeling languages on that framework. We chose ARIS as evaluation concept as from our point of views the scope of ARIS and especially EPC is close to PICTURE in comparison to other generic concepts like e.g. BPMN. Finally we will compare the results of our mapping and present our conclusions.

2. Semantic building block-based modeling

2.1. Characteristics of semantic building block-based modeling

The main modeling constructs of SBBLs are semantic building blocks, also called process building...
blocks (PBB). Each of them comprises a specific set of reoccurring tasks that can be found in the problem domain. To specify the individual building blocks, a domain ontology is required. It should cover all relevant aspects that should eventually be representable in the BPDs. By deriving the building blocks from this ontology, it can be ensured that all of them are semantically disjoint. As PBB designed in this way offer exactly one way to describe a specific fact found in the problem domain, ambiguities in the representation of this fact are avoided. Every PBB is identified by a unique domain statement which describes its meaning. This could be, for instance, “Print document” or “Forward document”. To further specify the meaning of PBB, an individual set of attributes can be assigned to each of them. An attribute is also derived from the domain ontology. A controlled range of possible values ensures that no ambiguities occur. Like the PBB also attributes as well as their values are identified by a unique domain statement.

The use of SBBLs is limited with regard to the application domain and the modeling purpose. As PBB comprise the semantics of the application domain, facts outside of this domain cannot be represented by using this language. Additionally, utilizing an SBBL in a modeling project requires the SBBL to support the particular project goals. With clearly defined building blocks and attributes, the level of abstraction and detail is fixed. Thus, the use of models created with the SBBL is limited to the goals the creators of the language had in mind during the process of creation.

2.2. Reasons to use a semantic building block-based language

General purpose modeling languages exhibit several disadvantages. The lack of a formal foundation causes the BPD created with them to be ambiguous [30]. This hampers the analyzability of the BPD and thereby increases the cost utilizing them. Furthermore, ambiguities increase the risk of making mistakes during the analysis, which might as well result in wrong and costly decisions, especially if the organizations core processes are reorganized. But not only is the analysis of BPDs an expensive task. As both domain experts and method experts are required to document the organizations processes, the creation of BPDs can involve significant efforts too [4].

SBBL reduce the cost of analyzing BPDs. With semantically enriched building blocks, SBBL are capable of being analyzed automatically. As the meaning of these constructs is known at the design time of the SBBL, it is possible to define analysis procedures before a BPD is created. Furthermore, the unambiguously and mutually exclusively defined building blocks considerably reduce the variations between the constructed models. Thus, it is easier to anticipate how specific situations found in the problem domain will be represented in the models, which allows a machine to search for them. Thereby efforts for performing complex and long-winded manual analysis can be saved. For example, an algorithm could identify media breaks by searching for every occurrence of a PBB “Enter data into IT”.

2.3. The PICTURE methodology

PICTURE is a methodology from the SBBL-type and was specifically designed to be used in public administrations. Its main purpose is to capture the complete process landscape of a public administration to assess reorganization potential, especially regarding the introduction of E-Government measures. Currently, it consists of 24 PBB, and several attributes, which are used to create BPDs. A List of these PBB can be found in Figure 1. Each type of PBB has its own specific semantics. Based on this semantics, we can group the PBB in three basic groups:

- Event related (E-)PBB. These PBB do not contain any Transformation but only indicate that something has happened. These are the PBB which are used to describe interfaces (e. g. Document comes in).
- Transformation related (T-)PBB: In these PBB an action is performed on processed objects (e. g. Create Document).
- Decision related (D-)PBB: These PBB decide on the further process flow and thus allow different resulting states (e.g. Perform a formal investigation).

PICTURE comprises four different views to create complete BPDs. In the following we will give a rough overview of these views. A more detailed description is e.g. given in [6]. The process view describes the operational structure of the administration in the form of single activities and the hence evolving processes. At the same time the process view integrates all other views by recording “who” carries out single activities, “with what” they are carried out and “what” is edited respectively produced. The central element of the PICTURE method and thus also of the process view are the PBB. At the next structural level PBB are composed to sub processes. Thereby a sub process is understood as a sequence of activities (PBB) which are carried out within one organization unit by one administrative employee and which contribute to the performance of a task of the complete process. Sub processes contain attributes, too. Here it is for instance recorded how often the sub process is carried out per year (number of cases) and who is responsible for its execution.

Within sub processes the modeling of process building blocks is done strictly sequential. This is due to the reason that one sub process only comprises those activities which one single administrative employee carries out. To represent alternative process flows it is possible that several alternative operational variants exist for one sub process, e.g. due to a decision (acceptance respectively rejection). For representing such a situation, PICTURE offers two different constructs. One the one hand, attributes can be used, like the above described attribute Input Channels, where different cases can be represented by entering percentages. On the other hand, it is possible to define sub process variants. Such a sub process variant describes the alternative execution of the sub process from start to finish.

Sub processes are composed to processes. A process is characterised by providing exactly one benefit or service for the customers of an administration. Examples for such processes are Issuing an identity card or Extending the parking permit.

In the organisational view the organisational structure of the administration is represented in a hierarchical composition of the different organisational units and positions. The organisation units are the basic elements of the organisation view. The organisational units are responsible for the execution of process aspects within the processes. Hence, in PICTURE sub processes are assigned to organisational units.

The business object view contains information concerning the necessary input (e.g. applications) and the corresponding produced output or possible intermediate products (e.g. statements or notifications) of an administrative process.

The resource view shows which work equipment is needed for providing an administrative service. That is, for example, software applications like MS Office or specialised procedures as well as hardware (printer, scanner) or judicial information like laws. The resource view contains element types for representing these non-organisational work supporters as well as sources and targets of the business objects.

3. The Bunge-Wand-Weber ontology

Business process modeling aims, just like most of the other modeling techniques used in the information systems field, at describing a relevant section of the real world to serve a specific purpose. As these techniques proliferated over time, the need to evaluate and compare them arose. The development of methods to examine the appropriateness of modeling languages was a result of this need. Such approaches can be found, for instance, in [21], who present a framework to evaluate software development methods. An survey on different approaches can be found in [29]. The most
famous attempt however, applied by several researchers in the last two decades, was developed by Wand and Weber [32, 34-38]. They have adapted an ontology originally developed by Bunge [9] and adapted it to be used in information modeling. Their ontology is deemed to be sufficient and necessary to describe every aspect of the real world [18]. By mapping constructs of a modeling language to constructs of the ontology and vice versa, it allows the assessment of the modeling language expressiveness.

The BWW ontology has proven its usefulness in the evaluation of several different modeling languages. Researchers applied it to dataflow diagrams [33], entity relationship diagrams [38], languages from the upper Case-Tools [16], the unified modeling language (UML) [13] and the open modeling language (OML) [23] as well as for object oriented system analysis in general [24]; beside its use for the assessment of Nijsen’s Information Analysis Method (NIAM) [39], as a theoretical basis to define modeling languages for off-the-shelf information system requirements and its application to the analysis of enterprise system requirements [28], reference models were evaluated with it [14]. Also, the expressiveness of the ARIS modeling framework was measured with its aid [17]. Our study is based on this work.

Despite its excessive use in literature, the applicability of the BWW ontology is not undisputed. An evaluation using the BWW ontology demonstrates only the correspondence between two formal systems, which is meaningless if the quality of the reference system is unknown. Thus, the lack of a formal proof for its appropriateness is criticized [40, 41]. However, the BWW ontology presents one of the most utilized model to evaluate conceptual modeling techniques including a proven track record to evaluate process modeling. While there are some alternative approaches, no other ontological model seems to be as widely adopted as BWW [27]. Hence, we will use BWW as ontological basis in this research. Nevertheless, the results of an ontological analysis using the BWW ontology have to be critically reviewed in the end.

The BWW ontology consists of different constructs which are listed in table 1. For a detailed description, see [16, 34-38].

4. Research methodology

The aim of this paper is to compare the expressiveness of languages from the SBBL-class with traditional business process modeling methodologies. To achieve this goal, we analyze one methodology from each class using the BWW ontology and subsequently compare the results. PICTURE serves as a representative for the SBBL-class, while the ARIS methodology represents the traditional modeling languages. We are aware of the problem that BPMN is coming up more and more to a standard in BPM, however, many concepts integrated in BPM are not in the core focus of pure process modeling. So we did not compare to BPMN in this phase, as for example the subsequent translation in executable workflow elements via BPEL and other workflow-related ideas are not in the scope of PICTURE. Further we refer to a Study from Gartner [7] who see ARIS in a leading position in the area of business process management.

Employing the BWW ontology to evaluate modeling languages requires a rigorous methodology to ensure comparability, completeness and objectivity. Several shortcomings of ontological analysis have been indentified in [26]. Thus, we follow the methodology proposed by these authors to overcome such problems.

First, a meta model is needed for the BWW ontology and the modeling language that is to be evaluated. Both meta models should be created using the same notation. On the one hand, this facilitates the understanding of the ontological constructs, and on the other hand, it improves the comparability, because the mapping between the modeling language and the ontological constructs of the BWW ontology, originally defined in a set theoretic notation, is reduced to a comparison of two conceptual models [12]. A meta model of the BWW ontology has been created by [12, 25]. It expresses the complete ontology in extended entity relationship modeling (eERM) notation. The meta model is structured by clustering it into logically connected parts. Our analysis is based on this work. A meta model for the PICTURE language can be found in [2].

Second, two mappings must be created to compare the modeling languages with the BWW ontology. The representation mapping starts with the elements of the BWW ontology meta model and aims at identifying corresponding elements in the modeling language meta model. Beginning with the entity type thing, the analysis proceeds with examining the BWW ontology meta model cluster by cluster and assigning suitable elements from the modeling language meta model to the individual constructs. Once the representation mapping is complete, the interpretation mapping has to be created. Apart from the fact that this analysis starts with the modeling language meta model, the procedure is similar to that of the representation mapping. Only the direction of the mapping is changed.

The creation of mappings between language elements and the BWW ontology is a very subjective process. Therefore we applied a two-stage approach to create our mappings. In the first step two of the authors...
created the mappings of PICTURE to the BWW ontology independent of each other. In a second step, these mappings were consolidated with the help of a third author thus resolving the discrepancies between the mappings from step one.

Third, the results of the mapping process must be evaluated to assess the expressiveness of the modeling languages in regard to the BWW ontology. The expressiveness can be characterized based on the mapping using four ontological inconsistencies [36].

1. **Construct overload** means that an individual element from the modeling language corresponds to multiple concepts from the BWW ontology. This indicates that the meaning of an element could be understood in several ways, which might lower the understandability of the models.

2. **Construct redundancy** means that an individual concept from the BWW ontology corresponds to multiple elements from the modeling language. When construct redundancy occurs, users of a modeling language could be confused about what the appropriate modeling language element is to express a certain aspect of the problem domain. This is a possible source of variations that should be avoided. However, construct redundancy is not necessarily a problem. If the BWW ontology is too abstract, the different modeling language elements could express distinct subtypes of a single ontological concept. If the elements do not express the concept completely, construct deficit is the result.

3. **Construct excess** means that an individual element from the modeling language has no corresponding concept in the BWW ontology. As these modeling elements do not express any aspect of the problem domain that is deemed to be important, this indicates that the element could be left out.

4. **Construct deficit** means that an individual concept from the BWW ontology has no corresponding element in the modeling language. This indicates that the modeling language lacks the ability to express certain aspects of the problem domain.

Given the results of the analysis of the modeling methodologies ARIS and PICTURE, they can be compared with each other. These results are used to assess advantages and shortcomings of SBBLs compared to traditional business process modeling languages. The authors claim that SBBL are easier to handle due to the limited freedom during the modeling process as well as unambiguously defined language elements. With clearly defined language elements, PICTURE should exhibit less construct overload and construct redundancy than ARIS does while not increasing construct deficit. Thus, our propositions are as follows:

(P1) PICTURE exhibits less construct overload and redundancy than ARIS.
(P2) PICTURE does not exhibit more construct deficit than ARIS.

5. Evaluation

5.1. Mapping PICTURE to the BWW Ontology

To evaluate the methodologies, in a first step we created a mapping of PICTURE to the BWW ontology and vice versa according to the process described above. We used the meta model of the BWW ontology provided in [12].

We will start the mapping with the BWW construct of a thing. As a thing defines a real-world phenomenon on the instance level we cannot find a corresponding element in the process view of PICTURE since processes here are depicted on the type level. However, the other views of PICTURE exhibit constructs which can be mapped to a thing: Organizational units and positions (organizational view), resources (resource view) and processed objects (processed object view).

A **Class** from the BWW ontology corresponds to the 24 different types of PBB in PICTURE. Furthermore the categories of resources and processed objects can be mapped to classes.

The BWW construct of a process corresponds to the process in the PICTURE process view, which depicts a whole process model. Based on our classification of PBB as given in section 2, we can map those different types of PBB to BWW constructs. E-PBB can be mapped to states since they do not perform any actions. T-PBB can be as transformation in the sense of BWW. Furthermore, they also contain information about the state of things after the transformation in their attributes. Hence, a sequence of two PBB (only state vector of first PBB) can be mapped to an event in BWW.

D-PBB can be mapped to the ontology construct of a state law. They check if things like processed object are in a law or unlawful state from a functional perspective (e. g. if an application is complete). If corrective actions are necessary to bring a thing back to a lawful state, this can be depicted as a process variant in PICTURE. Such a variant contains the necessary PBB sequence (events) to correct the state of the thing. Therefore, it can be mapped to the BWW construct of a lawful transformation.
In the context of PICTURE a system from the BWW ontology can be seen as a whole administration depicted in PICTURE. However, PICTURE does not have an explicit construct for this concept. Hence, the system environment is depicted in PICTURE using external organizational units. Accordingly, an external event can be seen as E-PBB which is triggered by an external organizational unit. All other events in PICTURE are internal events.

A stable state is depicted in PICTURE either by PBB at the end of a process or by the special PBB type interruption of work, which means that the current process remains in its state until an external event causes a change of state.

Since a T-PBB in PICTURE acts upon things like processed objects a PBB type can be seen as a property in general of those things. Furthermore attributes in PICTURE can be mapped to attributes.

Regarding the level structure PICTURE exhibits three fixed levels, the process level, the sub process level and the PBB level.

The process view in PICTURE does not exhibit constructs for system composition or subsystem. However, the organizational hierarchy, the processed object hierarchy and the resource hierarchy can be mapped to a system composition, and accordingly organizational units, processed object categories and resource categories can be mapped to subsystems.

5.2. Comparing ARIS and PICTURE

To evaluate the modeling methodologies, we will compare our mapping and a mapping of ARIS to the BWW concepts created by Green and Rosemann [17].

PICTURE clarifies process models by supporting the modeling of things and classes. Looking at the process view of ARIS, one can observe that there is no possibility to describe things and the classes they belong to in it. Hence, model users may be confused about what things of the problem domain the process works on [17]. This construct deficit is partially overcome by integrating the data, output or organizational view, but still, not all the things an activity works on can be expressed. PICTURE exhibits the same weakness in its process view, but the integration of the three other views solves the issue well. Incorporating things from these views, it is known in every step of a process what is done (process view), who is doing it (organizational view), what is required to accomplish the task (resource view) and what the object is that this activity works on (processed object view). Furthermore, information about classes of things a PBB works on can be directly inferred from the semantically enriched PBB.

PICTURE does not have redundant views. Comparing the function view and the process view of ARIS, it is obvious that the function view is completely redundant. Every BWW construct expressed by the function view is already covered by the process view [17]. Therefore, this construct overload compromises the ontological clarity of the models created using these views. In PICTURE, all views are defined in a way that there is no overlap between them. While the organizational, resource and processed object view all represent things, classes, system compositions and subsystems, they deal with different aspects of the problem domain. Thus, they represent sub concepts of the more general BWW constructs mentioned above, which ensures ontological clarity by avoiding construct redundancy. The same holds true for all the views of ARIS except the function and process view.

PICTURE avoids construct redundancy in the representation of decisions. As it can be seen in table 1, state laws as well as lawful transformations are expressed in ARIS using connectors together with their surrounding function and event types. As these patterns can include an arbitrary number of function and event types and connectors can be combined in any way to construct complex logical formulas with them, there exists an infinite number of patterns that may represent state laws and lawful transformations. PICTURE simplifies this by encapsulating state laws within D-PBB and lawful transformations within variants. Thus, there is less construct redundancy in PICTURE due to the fact that single language elements with defined domain meanings are used rather than complex combined elements from ARIS.

PICTURE avoids construct redundancy and overload by fixing the level of abstraction. To express the hierarchy of processes in ARIS, individual function types can be re-interpreted as whole processes to build a level structure on them. This results in construct overload, as the function type can represent transformations, properties in general and processes. Consequently, construct redundancy results because the BWW construct process is expressed using either an ARIS process or a function type [17]. As PICTURE enforces a certain level of abstraction by employing processes, their sub processes and the PBB, no such problem arises when using this methodology. However, since PBB allow inferring information about the class of the thing processed by the PBB, the PBB adds some construct overload in this area. While both aspects described in this paragraph produce redundancy either in ARIS or PICTURE, which is clearly problematic from an ontological point of view, it can be argued whether modelers or model users consider them problematic as well.
Regarding construct deficit, we will first evaluate the process views of both methodologies. Apart from minor differences, one can easily see that they cover almost the same ontological constructs. However, the process building blocks of PICTURE allow describing classes. Thus, PICTURE exhibits less construct deficit in this view. Comparing the additional views of both languages is only partially feasible. As PICTURE does not focus on data modeling, the data view of ARIS has no correspondence within PICTURE. Also the output view of ARIS is not part of PICTURE, but is contained within the processed object view to some extent. The resource view of PICTURE has no counterpart in ARIS. Only the organizational views of both methodologies share most of their key concepts. While almost all BWW constructs expressible in the ARIS organizational view are expressible in the PICTURE organizational view as well, PICTURE does not allow to represent couplings. Due to this construct deficit the expressiveness of PICTURE in the sense of the BWW ontology is lower than that of ARIS at this point.

<table>
<thead>
<tr>
<th>BWW construct</th>
<th>PICTURE equivalent</th>
<th>PICTURE view</th>
<th>ARIS equivalent</th>
<th>ARIS sub-model</th>
</tr>
</thead>
<tbody>
<tr>
<td>THING</td>
<td>Organizational unit Position Resource Processed Object</td>
<td>Organizational view Organizational view Resource view Processed Object view</td>
<td>Organizational unit Position User Product Catalogue Product Model Bill of Materials</td>
<td>Organizational view Organizational view Organizational view Output view Output view Output view</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>T-PBB types attributes</td>
<td>Process view All views attributes</td>
<td>Function type, Entity type All views</td>
<td>Process / function view Data view</td>
</tr>
<tr>
<td>CLASS</td>
<td>PBB Type Resource category Processed object category</td>
<td>Process view Resource view Processed object view</td>
<td>Entity type</td>
<td>Data view</td>
</tr>
<tr>
<td>KIND</td>
<td></td>
<td></td>
<td>Specialization / Generalization</td>
<td>All views expect process view</td>
</tr>
<tr>
<td>STATE</td>
<td>E-PBB</td>
<td>Process view</td>
<td>Event type</td>
<td>Process View</td>
</tr>
<tr>
<td>STATE LAW</td>
<td>D-PBB</td>
<td>Process view</td>
<td>Function type ➔ Connector ➔ Event type</td>
<td>Process View</td>
</tr>
<tr>
<td>EVENT</td>
<td>PBB ➔ PBB</td>
<td>Process view</td>
<td>Event type ➔ Function type ➔ Event type</td>
<td>Process view</td>
</tr>
<tr>
<td>PROCESS</td>
<td>Process</td>
<td>Process View</td>
<td>Process Function Type</td>
<td>Process view / Function view</td>
</tr>
<tr>
<td>TRANSFORMATION</td>
<td>T-PBB</td>
<td>Process view</td>
<td>Function type</td>
<td>Process view</td>
</tr>
<tr>
<td>LAWFUL TRANSFORMATION</td>
<td>Process Variant</td>
<td>Process view</td>
<td>Event type ➔ connector ➔ function type</td>
<td>Process view</td>
</tr>
<tr>
<td>COUPLING BINDING MUTUAL PROPERTY</td>
<td>Relationship type Role</td>
<td>Data view Organizational view</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 6. Conclusion

Starting with the assertion that SBBL are more efficient for business process modeling due to their unambiguously defined language elements and the reduction of freedom modelers have during the modeling process, the BWW ontology was used to compare PICTURE, a methodology based on SBBL, with ARIS, a generic modeling methodology. The first proposition was that, if PICTURE actually exhibits fewer ambiguities than traditional methodologies like ARIS, it should reduce the construct redundancy and construct overload found in an analysis using the BWW ontology. While restricting the choices a modeler can make when creating business process models helps to ease the modeling process and to improve the efficiency, it must be ensured that all relevant aspects of the problem domain are still expressible. Therefore, our second proposition was that PICTURE should not exhibit more construct deficit than ARIS, since this would mean that PICTURE would lack some expressiveness.

Within the analysis, it could be shown that PICTURE improves some problematic aspects of the ARIS methodology. First, the lacking possibility to express the concept of a thing and a class is encountered by integrating three supplementing views on the PBB level. This way, for every activity within a process, it can be seen what is done with which business object, which organizational unit is responsible for the activity and what resources are necessary. Second, the views of PICTURE are unambiguously defined. In contrast, the function view of ARIS is ontologically redundant, since all its information is already expressed within the process view. Third, a significant amount of redundancy can occur in ARIS when decisions are represented. As | BWW construct | PICTURE equivalent | PICTURE view | ARIS equivalent | ARIS sub-model |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM COMPOSITION</td>
<td>Organization hierarchy Resource hierarchy Processed object hierarchy</td>
<td>Organization view Resource view Processed object view</td>
<td>Cluster Organizational hierarchy Product model hierarchy</td>
<td>Data view Organizational view Output view</td>
</tr>
<tr>
<td>SYSTEM ENVIRONMENT</td>
<td>External organizational unit</td>
<td>Resource view</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSYSTEM</td>
<td>Organizational unit Resource category Processed Object Category</td>
<td>Organization view Resource view Processed object view</td>
<td>Cluster Organizational Object Product modeling Object</td>
<td>Data view Organizational view Output view</td>
</tr>
<tr>
<td>LEVEL STRUCTURE</td>
<td>Process Sub process</td>
<td>Process view</td>
<td>Function/event type decomposition Organizational modeling levels Product modeling levels</td>
<td>Process view / Data view Organizational view Output view</td>
</tr>
<tr>
<td>EXTERNAL EVENT</td>
<td>E-PBB with external organizational unit</td>
<td>Process view</td>
<td>Start event type</td>
<td>Process view</td>
</tr>
<tr>
<td>STABLE STATE</td>
<td>E-PBB at the end of a process PBB: Interruption of work</td>
<td>Process view</td>
<td>End event type</td>
<td>Process view</td>
</tr>
<tr>
<td>INTERNAL EVENT</td>
<td>PBB → PBB</td>
<td>Process view</td>
<td>Event type → Function type → Event type</td>
<td>Process view</td>
</tr>
<tr>
<td>WELL-DEFINED EVENT</td>
<td>PBB → PBB</td>
<td>Process view</td>
<td>Event type → Function type → Event type</td>
<td>Process view</td>
</tr>
</tbody>
</table>
connectors are used to build up arbitrarily complex logical functions, many different redundant representations can be produced. PICTURE encapsulates decisions within D-PBB which produce different variants of processes. Thereby construct redundancy is reduced and the representation is simplified. Fourth, the fixed level of abstraction used within PICTURE renders a re-interpretation of other language elements as processes, like it can be done with the function type of ARIS, unnecessary, thereby reducing construct overload.

Apart from these differences, the representation of processes is similar in both methodologies. As the explicit representation of data or product structures is not in the scope of PICTURE, it does not have views comparable to the ARIS data or output view. Thus, the construct deficit resulting from the absence of corresponding views within PICTURE can be ignored. However, there is still a small amount of construct deficit resulting from the missing possibility to express the BWW construct coupling.

7. References


