3-5-2015

Supporting Knowledge Elicitation and Analysis for Business Process Improvement through a Modeling Tool

Florian Johannsen

Hans-Georg Fill

Supporting Knowledge Elicitation and Analysis for Business Process Improvement through a Modeling Tool

Florian Johannsen[1] and Hans-Georg Fill[2]

[1] Chair of Business Engineering, University of Regensburg, Germany
florian.johannsen@wiwi.uni-regensburg.de
[2] Research Group Knowledge Engineering, University of Vienna, Austria
hans-georg.fill@univie.ac.at

Abstract. Business Process Improvement (BPI) is a high priority topic for modern enterprises. However, due to the distributed knowledge, conducting BPI projects has become challenging in times of inter-organizational business networks. For supporting the elicitation, analysis, and sharing of knowledge by practitioners, we describe how semi-formally described domain knowledge on BPI and knowledge on problem-solving techniques is transformed into an implementation-oriented representation in the form of a modeling tool. Thus, we revert to the FDMM formalism (Formalism for Describing ADOxx Meta Models and Models) that permits to bridge the gap between semi-formal meta models and those that are executable on the ADOxx meta modeling platform.

Keywords: Business Process Improvement, Meta Modeling, Formalization.

1 Introduction

In today's enterprises, the improvement of business processes ranks among the top priorities CEOs have to deal with [1]. However, conducting BPI projects has become more and more challenging. Information technology (IT) causes high market transparency, which leads to rapidly changing customer requirements [2]. A company thus has to continuously analyze the "Voice of the Customer" (VOC) (cf. [3]) to meet consumers' current needs, which is a resource-intensive process. Further, companies increasingly create "value" (cf. [3]) in business networks [4]. To prevent such BPI projects from falling short of their initial aim, employees from all cooperating partners in the network need to be involved. The proper documentation and communication of emerging knowledge on business processes is thus recognized as a critical success factor for BPI initiatives (cf. [5]). To deal with these challenges, methodologies and techniques of knowledge management can be drawn upon [6]. In that context, employees' knowledge on potential process improvements can be elicited, shared across an organization and ideally be internalized into everyday work practices (cf. [7]). Elicitation thereby refers to obtaining information required for solving problems [8]. This may either be accomplished through organizational or technology-oriented approaches that can be particularly adapted for these purposes (cf. [9]). In a

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Johannsen, F.; Fill, H. (2015): Supporting Knowledge Elicitation and Analysis for Business Process Improvement through a Modeling Tool, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 752-766

previous work, a BPI roadmap was developed in cooperation with BPI experts to enable the goal-oriented creation of solutions for optimizing process performance (cf. [10]). One major goal of this roadmap is to support knowledge elicitation (cf. [11]) of improvement potentials among employees. The roadmap comprises a manageable set of BPI techniques (cf. [12]). To codify the emerging knowledge and results when using the roadmap, a modeling language in the form of a meta model was specified [10]. Meta models permit to structure a domain and define how model instances can be created [13]. Model instances allow to document and communicate the parts of process-related knowledge that can be made explicit and provide a basis for conducting further analyses such as queries and the creation of reports (cf. [14]).

So far, the meta model defined for the BPI roadmap specified the structure of the BPI techniques and their relations to each other. Although this was done using UML class diagrams, the level of detail used is not sufficient for realizing an appropriate technical implementation as a modeling tool. However, a graphical modeling tool is favorable to efficiently support employees in eliciting and analyzing knowledge in an effective and efficient manner, thus avoiding the bottleneck in capturing and validating knowledge that has been a traditional shortcoming of expert and decision support systems [15]. Further, a tool is essential for analyzing the information captured in the models using machine-based processing, or for feeding the information into other systems such as groupware platforms or business intelligence applications.

A preliminary step for implementing the BPI roadmap as a software prototype is to formally specify it using an adequate formalism (cf. [16]). Formal specifications help to better understand customer requirements, to uncover contradictions in the functional design and to provide an uncomplicated transfer from specification to implementation [16]. We therefore pose the following research question: *How can the semi-formal representation of the BPI roadmap as a meta model be adequately transferred into a formal specification that serves as a sound base for its implementation as a graphical modeling tool?*

In knowledge engineering, it has long been discussed how knowledge representations that are primarily oriented towards supporting the communication between domain experts and technical experts (e.g., meta models) can be further formalized to enable machine processing [17]. For example, through Model-based and Incremental Knowledge Engineering (MIKE) smooth transitions can be achieved from semi-formal representations over formal representations to implementation-oriented representations [18]. The implementation-oriented representation can then be executed.

In the following, we will describe how such transitions can be achieved in the context of BPI. For this purpose, we refer to the procedure as depicted in Fig. 1, which is similar to the MIKE development process (cf. [18]) and follows the design science approach or the principles of engineering, respectively (cf. [19], [20]).

Therefore, requirements on a BPI roadmap and a supporting modeling tool were defined in a first step (cf. [10]). Based on these, a concept of the BPI roadmap was established and evaluated in cooperation with practitioners. In the *design* activity, a corresponding semi-formal meta model was specified. This paper focuses on the subsequent step of formalizing the meta model (*formalization* activity). The formalization facilitates the derivation of an executable implementation-oriented representation

(*development* activity). The graphical modeling tool, supporting the use of the BPI roadmap, represents the final artifact received via the *deployment* activity.

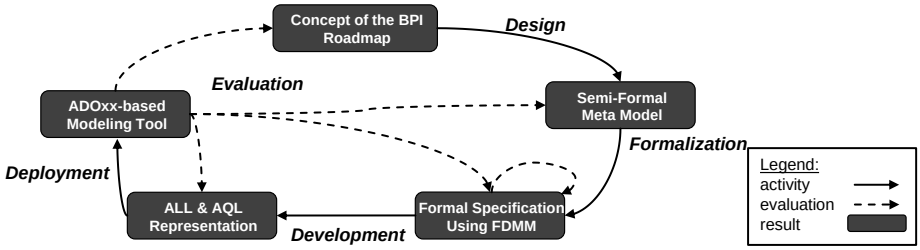

Fig. 1. Procedure for the Development of Tool Support for Knowledge Elicitation and Analysis in BPI initiatives

The purpose of this paper is to show how the constructs of the semi-formal meta model of the BPI roadmap can be formalized (*formalization* activity) enabling the transition to implementation-oriented representations that can be directly deployed on the openly accessible ADOxx meta modeling platform to generate a modeling tool.

The contribution of the paper is threefold: first, it demonstrates how a semi-formal concept for conducting BPI projects can be transferred to an implementation-oriented representation or a running prototype, respectively. That way the use of a formal specification for achieving an exact design without contradictions and thus speeding up the subsequent implementation is particularly emphasized. Second, the paper highlights how the peculiarities of a "roadmap" solution for BPI (cf. [10]) are reflected in its formal specification. Third, the benefits of formalisms for validating functional specifications become obvious from the example of the BPI roadmap.

The structure of the paper is as follows: in the next section, foundations of BPI, the formalism used and the ADOxx meta modeling platform are provided. In section 3, the formalization of the BPI roadmap is introduced. This serves as a basis for the prototypical implementation (see section 4). Afterwards, the results of this research are discussed. The paper is concluded with a summary and an outlook.

2 Foundations

2.1 Business Process Improvement and the BPI Roadmap

In recent years, manifold BPI approaches were developed. The most prominent methodology was introduced by Harrington [21]. Ideas of this initial approach were taken up and further specified by Adesola and Baines [22], Vakola and Rezgui [23], or Lee and Chuah [24] in their BPI methods. Details on these approaches can be found in Zellner [12] for example. However, existing BPI approaches have methodological flaws hampering their usability [12]. Further, practitioners increasingly shrink back from using holistic BPI approaches which they perceive as over-dimensioned and hard to handle (cf. [1]). They prefer a manageable set of easy-to-use and well-

established BPI techniques instead (cf. [1]). To address this need, the so-called BPI roadmap was introduced (cf. [10]). The BPI roadmap is a logical arrangement of 11 proven and beneficial BPI techniques supporting all stages of a BPI project. It was evaluated in practice and can be used in the production as well as the service industry.

The roadmap works as follows (see Fig. 2): at the beginning of a BPI project, the process to be improved is visualized (SIPOC Diagram) and customers' as well as employees' requirements on the process are identified (CTQ/CTB Matrix). Afterwards, performance indicators which measure the process performance are defined and prioritized (Measurement Matrix). Then, process data is collected (Data Collection Plan). The collected process data is analyzed (Histogram, Scatterplot) and causes for lacking process performance are discussed (Ishikawa Diagram). Finally, solutions for process improvement are worked out (Affinity Diagram), implemented, and measures for a continuous process control are set up (Reaction Plan, Control Charts).

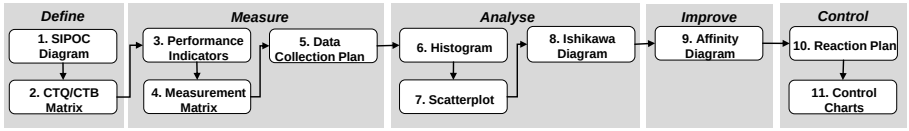

Fig. 2. The BPI Roadmap [10]

The BPI techniques of the BPI roadmap were transformed into 9 conceptual model types, and an integrated meta model was derived (see Fig. 1 – *design* activity). Fig. 3 shows an excerpt of this integrated meta model where the CTQ-/CTB-Model, a central model type of the BPI roadmap, is highlighted. In this model type, the verbally expressed requirements of internal (e.g., employees) as well as external customers on process performance are documented as the "Voice of the Customer (VOC)" or the "Voice of the Business (VOB)", respectively. The requirements are then condensed to core statements which serve as a basis for deriving quantifiable "Critical-to-Quality (CTQ)" and "Critical-to-Business (CTB)" factors. These determine the goals of a BPI project. An example is shown in Fig. 4. Several reports to query and analyze the results captured in the model types were defined (cf. [10] and Fig. 3).

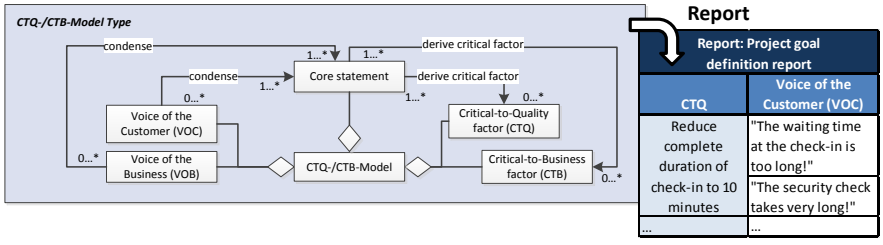

Fig. 3. Excerpt of the BPI Roadmap Meta Model and an Example for a Report

2.2 Modeling Methods and the FDMM Formalism

Modeling methods in general are composed of a modeling language, mechanisms, algorithms, and a modeling procedure that defines how the method makes use of the

language and algorithms to achieve results [25]. The components of a modeling method may come in various degrees of formalization [26]. Formalizations contribute in particular to the intersubjective understandability and/or the processing by different computer systems. For the implementation of the modeling tool, we focus on the formalization of a modeling language and according query mechanisms. The FDMM formalism, which was originally proposed in Fill et al. [27], permits to describe the syntax of meta models and models and the instantiation of models from meta models in a mathematical and thus unambiguous format. It provides the means to smoothly transition from semi-formal representations to implementation-oriented representations.

In literature, further formalizations for meta modeling approaches such as EMOF [28], [29] and KM3 [30] were introduced. However, both pursue the specification of software structures [31], which complicates their use in the BPI context. The same holds true for the Object Constraint Language (OCL) [32] which focuses on UML in particular. FDMM represents an easy-to-use formalism which aims at specifying meta models for different application domains without requiring specialized mathematical knowledge apart from set theory and first-order-logic. Therefore, it was chosen as a means for formalizing the meta model of the BPI roadmap. As FDMM is independent of a specific technical implementation, its constructs need to be mapped to an implementation-oriented, i.e., executable specification such as the ADOxx Library Language (ALL) [31]. In the following, we will briefly describe the very core concepts of FDMM as they have been used for method descriptions previously (cf. [33]). However, we refer the interested reader to the original publications on FDMM for further details (cf. [27]). Meta models MM are defined in FDMM as:

$$\mathrm{MM} = \langle \mathrm{MT}, \preccurlyeq, \mathrm{domain}, \mathrm{range}, \mathrm{card} \rangle \tag{1}$$

The set MT stands for model types that will be used as groupings or diagram types of elements in a meta model:

$$\mathrm{MT} = \{\mathrm{MT}_1, \mathrm{MT}_2, \dots, \mathrm{MT}_m\} \tag{2}$$

In every model type MT_i, a tuple of a set of object types O_i^T, a set of data types D_i^T, and a set of attributes A_i are contained:

$$\mathrm{MT}_i = \langle \mathrm{O}_i^T, D_i^T, A_i \rangle \tag{3}$$

The sets O^T, D^T and A are collections of all object types, data types and attributes of a model type. However, there may also be object types that exist independently of a model type:

$$O^T = \bigcup_j O_j^T, D^T = \bigcup D_i^T, A = \bigcup A_i \tag{4}$$

The relation $\preccurlyeq$ defines an ordering on the set of object types, O^T, i.e., $o_1^t \preccurlyeq o_2^t$ denotes that object type o_1^t is a subtype of object type o_2^t. To assign attributes to object types, a domain function is defined that maps attributes to the power set of object types:

$$\mathrm{domain} : A \to \mathcal{P}\left(\bigcup_j O_j^T\right) \tag{5}$$

In the next step, the range function maps attributes to the power set of all pairs of object types and their model types to data and model types. In this way, the values of attributes are constrained in the model instances that are to be defined later. In addition to standard attribute types such as integer or string, also references to other object

types in model types or to complete model instances, i.e., via model types, can be defined:

$$\text{range: } A \to \mathcal{P}\left(\bigcup_j \left(O_j^T \times \{\text{MT}_j\}\right) \cup D^T \cup \text{MT}\right) \tag{6}$$

To introduce the concept of cardinalities, the card function constrains the number of attribute values for a specific object type:

$$\text{card: } O^T \times A \to \mathcal{P}\left(\mathbb{N}_0 \times (\mathbb{N}_0 \cup \{\infty\})\right) \tag{7}$$

The sets O^T, D^T, A are defined to be pairwise disjoint. It is further required that there exists a corresponding domain function for any attribute which points to an object type of the same model type:

$$a \in A_i \Longrightarrow \text{domain}(a) \subseteq O_i^T \tag{8}$$

When instantiating a meta model MM, the instances are defined by the tuple:

$$\langle \mu_{\text{mt}}, \mu_{\text{O}}, \mu_{\text{D}}, \tau, \beta \rangle \tag{9}$$

Where the function μ_{mt} defines a mapping from model types MT to the power set of model instances mt

$$\mu_{\text{mt}}: \text{MT} \to \mathcal{P}(\text{mt}) \tag{10}$$

and the set mt comprises the union of all mappings of model types to model instances so that every element of the set of model instances mt corresponds to a model type:

$$\text{mt} = \bigcup \mu_{mt}(\text{MT}_j) \tag{11}$$

With the function μ_{O}, we map the object types of a particular model type to the power set of object instances O:

$$\mu_{\text{O}}: \bigcup_j \left(O_j^T \times \{\text{MT}_j\}\right) \to \mathcal{P}(O) \tag{12}$$

Thereby, the set O is the union of all object instances in the way that no object instance exists that does not have a mapping to an object type and a model type:

$$O = \bigcup_j \mu_{\text{O}}\left(O_j^T \times \{\text{MT}_j\}\right) \tag{13}$$

FDMM further introduces the concept of abstract object types. Any object type $o_t \in O^T$ may be denoted as an abstract type, which means that for all model types MT_i in which the object type o^t ($o^t \in O_i^T$) is contained, the object type can only be instantiated via the function μ_{O} through one of its subtypes:

$$\mu_{\text{O}}(o^t, \text{MT}_i) = \bigcup_{o_1^t \neq o_,^t o_1^t \preccurlyeq o^t} \mu_{\text{O}}(o_1^t, \text{MT}_i) \tag{14}$$

With the function μ_D, data types are mapped to the power set of data objects. In FDMM neither data types nor data objects are further specified. It is rather left to the user to further define the characterization of the used data types and their validity as necessary:

$$\mu_{\text{D}}: D^T \to \mathcal{P}(D) \tag{15}$$

Finally, to link the value of attributes to object instances, FDMM defines triple statements that are part of model instances via a mapping defined by the function β:

$$\tau \subseteq O \times A \times (D \cup O \cup \text{mt}) \tag{16}$$

$$\beta: \text{mt} \to \mathcal{P}(\tau) \tag{17}$$

FDMM further defines a number of correctness, disjointness and partitioning constraints that we do not list here for reasons of brevity (cf. [27], [33]).

2.3 The ADOxx Meta Modeling Platform

Different approaches exist for the implementation of modeling methods in modeling tools. Depending on the goals and purpose of the tool to be developed, the implementation can be accomplished from scratch, i.e., by using a common programming language such as Java or C++ or it can be reverted to specific libraries, services, or software applications. In so doing, the most extensive support is currently provided by meta modeling platforms such as MetaEdit+ [34], Eclipse GMF/EMF [35], GME [36] or ADOxx [37]. With meta modeling platforms, modeling tools can be realized with little or almost no programming effort. They thus permit to drastically shorten the development life cycle of modeling tools and supply a wide range of additional functionalities based on their underlying meta modeling languages [38]. For our purposes, we reverted to the ADOxx meta modeling platform which is provided for free at *www.adoxx.org* and was used successfully in previous projects by the authors. ADOxx is an industry-scale C++-based meta modeling platform which allows to define a modeling language and its graphical representation as well as mechanisms and algorithms using a set of domain specific languages [37]. These languages are ALL for the specification of the classes, relationclasses, attributes, and model types, the GRAPHREP language for the specification of the graphical representation of classes and relationclasses, the ATTRREP language for the specification of attribute visibilities and the ADOscript language for the specification of mechanisms and algorithms. Based on these definitions, ADOxx automatically generates modeling tools that provide graphical modeling editors, amongst further components for analyzing, simulating, evaluating, and transforming models.

3 An Implementation-Oriented Representation for Knowledge Elicitation and Analysis in BPI

With the foundations presented in the previous section, we can now apply FDMM to the BPI roadmap meta model. To exemplify the application of the FDMM formalism, a central model type is selected, namely the "CTQ-/CTB-Model" $\mathrm{MT}_{CTQ/CTB}$. The meta model as shown in Fig. 3 can be specified as follows:

$$\mathrm{MT}_{CTQ/CTB} = \langle O^T_{CTQ/CTB}, D^T_{CTQ/CTB}, A_{CTQ/CTB} \rangle \tag{18}$$

A model type is further specified by object types, data types, and attributes. Consequently, we define the following object types for the CTQ-/CTB-Model:

$$O^T_{CTQ/CTB} = \{\text{Abstract-Voice-of-Customer-Business-Class, VOC, VOB, Abstract-Critical-to-Customer-Business-Class, CTQ, CTB, Core-statement, Condense, Derive-critical-factor}\} \tag{19}$$

Afterwards, inheritance relationships can be determined. For example, the object types “VOC” and “VOB” represent subtypes of the object type “Abstract-Voice-of-Customer-Business-Class” that is defined as an abstract object type:

$$\begin{aligned} \text{VOC} &\preccurlyeq \text{Abstract-Voice-of-Customer-Business-Class} \\ \text{VOB} &\preccurlyeq \text{Abstract-Voice-of-Customer-Business-Class} \\ \text{CTQ} &\preccurlyeq \text{Abstract-Critical-to-Customer-Business-Class} \\ \text{CTB} &\preccurlyeq \text{Abstract-Critical-to-Customer-Business-Class} \end{aligned} \tag{20}$$

That way, attributes can be assigned to the subtypes more easily and relationships between all subtypes of an object type can be defined. Different data types are used in the CTQ-/CTB-Model. Whereas *“String”* represents a common data type, the “**Enum**” types are used to define data types with pre-defined, fixed values:

$$D^{T}_{CTQ/CTB} = \{\, String, \mathbf{Enum}_{quality\ dimension}, \mathbf{Enum}_{priority}, \ldots \} \tag{21}$$

For example, the type “$\mathbf{Enum}_{quality\ dimension}$” comprises the values “Quality”, “Time”, “Cost” and “Flexibility” which represent commonly accepted dimensions of process quality (cf. [39]). To each project goal (CTQ or CTB), one of these dimensions is assigned via an attribute “Quality-dimension”. By means of the type “$\mathbf{Enum}_{priority}$”, it is possible to prioritize the project goals later on:

$$\mathbf{Enum}_{quality\ dimension} = \{\text{Quality, Time, Cost, Flexibility}\} \tag{22}$$

$$\mathbf{Enum}_{priority} = \{\text{High, Medium, Low, Unspecified}\} \tag{23}$$

After that step, the attributes of the object types are to be defined. The attributes derived for the CTQ-/CTB-Model are focused again. Besides attributes for describing an object in more detail, e.g., “Name” or “Description”, also attributes to prioritize the project goals (e.g., “Priority”) are formulated. In addition, elements required for specifying the start- and endpoint of connections, e.g., the attributes “condense-to” and “condense-from”, are considered:

$$\begin{aligned} A_{CTQ/CTB} = \{&\text{Name, Description, Quality-dimension, Priority,} \\ &\text{condense-to, condense-from, derive-critical-factor-from} \\ &\text{derive-critical-factor-to,}\ldots\} \end{aligned} \tag{24}$$

In a final step, the attributes are to be specified by information on domain, range and cardinalities. When attributes are assigned to an abstract type, the attribute definitions are inherited by all subtypes. This is exemplified for the “Abstract-Voice-of-Customer-Business-Class” abstract type and the attribute “Description”. The intention is that instances of the “VOC” or “VOB” object types can be specified via textual descriptions using the attribute “Description”. The user is not forced, though, to attach a corresponding description which becomes obvious by the cardinality $\langle 0,1 \rangle$:

$$\begin{aligned} \text{domain(Description)} &= \{\text{Abstract-Voice-of-Customer-Business-Class}\} \\ \text{range(Description)} &= \{\text{String}\} \\ \text{card(Description)} &= \langle 0,1 \rangle \end{aligned} \tag{25}$$

Later on, the “CTQ” and “CTB” objects are referenced by the model type “Measurement Matrix Model”. This model type introduces the object type “CTQ-

Reference", amongst others. We use the "CTQ-Reference" object type to demonstrate how references from one object type to another object type work across model types:

$$\begin{gathered}\text{domain}(\text{Referenced-CTQ}) = \{\text{CTQ-Reference}\} \\ \text{range}(\text{Referenced-CTQ}) = \{(\text{CTQ}, \text{MT}_{CTQ/CTB})\} \\ \text{card}(\text{Referenced-CTQ}) = \langle 1,1 \rangle\end{gathered} \tag{26}$$

For connecting objects within a model type, connections were defined previously (19). For example, the VOCs and VOBs are to be condensed to so-called core statements in the CTQ-/CTB-Model. Therefore, a connection "Condense" can be used to visually assign VOCs and VOBs to corresponding core statements. The attributes "condense-from" and "condense-to" are used to express this connection:

$$\begin{gathered}\text{domain}(\text{condense-from}) = \{\text{Condense}\} \\ \text{range}(\text{condense-from}) = \{(\text{Abstract-Voice-of-Customer-Business-Class}, \text{MT}_{CTQ/CTB})\} \\ \text{card}(\text{condense-from}) = \langle 1,1 \rangle\end{gathered} \tag{27}$$

$$\begin{gathered}\text{domain}(\text{condense-to}) = \{\text{Condense}\} \\ \text{range}(\text{condense-to}) = \{(\text{Core statement}, \text{MT}_{CTQ/CTB})\} \\ \text{card}(\text{condense-to}) = \langle 1,1 \rangle\end{gathered} \tag{28}$$

To demonstrate the instantiation of meta models of the BPI roadmap via FDMM, we focus on the CTQ-/CTB-Model again.[1] The instantiation is defined as follows:

$$\mu_{MT}(\text{MT}_{CTQ/CTB}) = \{\text{mt}_{CTQ/CTB1}\} \tag{29}$$

Afterwards, the object types are instantiated:

$$\begin{gathered}\mu_O(\text{VOC}, \text{MT}_{CTQ/CTB}) = \{\text{VOC}_1, \text{VOC}_2\} \\ \mu_O(\text{VOB}, \text{MT}_{CTQ/CTB}) = \{\text{VOB}_1, \text{VOB}_2\} \\ \mu_O(\text{CTQ}, \text{MT}_{CTQ/CTB}) = \{\text{CTQ}_1\} \\ \mu_O(\text{CTB}, \text{MT}_{CTQ/CTB}) = \{\text{CTB}_1\} \\ \mu_O(\text{Core-statement}, \text{MT}_{CTQ/CTB}) = \{\text{CS}_1, \text{CS}_2\}\end{gathered} \tag{30}$$

To label the constructs, attribute values of the associated data types need to be instantiated. This is shown for the data type *"String"*:

$$\mu_D(\text{String}) = \{\text{'The waiting time is too long!', 'Reduce cycle-times'}, \ldots\} \tag{31}$$

These attribute values are then assigned to object instances. The following example allocates instances of the *"String"* data type (as created above) to instances of the object types "VOC" and "Core-statement":

$$\begin{gathered}(\text{VOC}_1 \text{ Name } \text{'The waiting time is too long!'}) \in \beta(\text{mt}_{CTQ/CTB1}) \\ (\text{CS}_1 \text{ Name } \text{'Reduce cycle-times'}) \in \beta(\text{mt}_{CTQ/CTB1})\end{gathered} \tag{32}$$

For the definition of edges in the CTQ-/CTB-Model, instances of the object types "Condense" and "Derive-critical-factor" have to be created. Whereas the "Condense"

[1] An instance of the model type "CTQ-/CTB-Model" – taken from the prototypical implementation – is shown in Fig. 4.

connection is used for assigning VOCs and VOBs to core statements, the "Derive-critical-factor" connection is used to relate core statements to CTQs and CTBs:

$$\mu_O(\text{condense, MT}_{CTQ/CTB}) = \{c_1, c_2, c_3, c_4\} \tag{33}$$

$$\begin{gathered}(c_1 \text{ condense-from } \text{VOC}_1) \in \beta(\text{mt}_{CTQ/CTB1}) \\ (c_1 \text{ condense-to } \text{CS}_1) \in \beta(\text{mt}_{CTQ/CTB1})\end{gathered} \tag{34}$$

$$\mu_O(\text{Derive-critical-factor, MT}_{CTQ/CTB}) = \{\text{dcf}_1, \text{dcf}_2\} \tag{35}$$

$$\begin{gathered}(\text{dcf}_1 \text{ derive-cf-from } \text{CS}_1) \in \beta(\text{mt}_{CTQ/CTB1}) \\ (\text{dcf}_1 \text{ derive-cf-to } \text{CTQ}_1) \in \beta(\text{mt}_{CTQ/CTB1})\end{gathered} \tag{36}$$

From the conceptual model types, textual reports can be generated, enabling an easy analysis and communication of the results of a BPI project. Based on the BPI roadmap, a set of 12 beneficial reports was specified for that purpose (cf. [10]). In the following, we exemplarily formalize the queries for the "Project goal definition report" focusing on project goals of a BPI initiative (see Fig. 3). The report is derived from the CTQ-/CTB-Model and analyzes which customer requirements (VOCs) are transformed into specific project goals (CTQs). The report is based on two queries (Q1 and Q2). First, all core statements connected with a specific CTQ via the connection "Derive-critical-factor" are queried (37). Then, the VOCs for each core statement are retrieved (38):

$$\begin{gathered}QS_1 = \mu_O(\text{Core-statement, MT}_{CTQ/CTB}) \\ Q_1 = \left\{ q \in QS_1 \;\middle|\; \begin{array}{l} \exists\, \text{d} \in \mu_O(\text{Derive-critical-factor, MT}_{CTQ/CTB}), \\ \exists\, \text{ctq} \in \mu_O(\text{CTQ, MT}_{CTQ/CTB}), \\ \exists\, m \in \mu_{MT}(\text{MT}_{CTQ/CTB}), \\ (d \text{ derive-cf-from } q) \in m \wedge \\ (d \text{ derive-cf-to } ctq) \in m \end{array} \right\}\end{gathered} \tag{37}$$

$$\begin{gathered}QS_2 = \mu_O(\text{VOC, MT}_{CTQ/CTB}) \\ Q_2 = \left\{ q \in QS_2 \;\middle|\; \begin{array}{l} \exists\, cond \in \mu_O(\text{Condense, MT}_{CTQ/CTB}), \\ \exists\, cs \in Q_1, \\ \exists\, m \in \mu_{MT}(\text{MT}_{CTQ/CTB}), \\ (cond \text{ condense-from } q) \in m \wedge \\ (cond \text{ condense-to } cs) \in m \end{array} \right\}\end{gathered} \tag{38}$$

4 Implementation via ADOxx

Based on the formal specifications that we outlined above, we could derive implementation-oriented representations in the form of ALL and ADOxx Query Language (AQL) code for the ADOxx meta modeling platform. The implementation-oriented representation contains knowledge already captured in the formal specification but enriches it by additional information required for implementation purposes, e.g., information on the modeling procedure or algorithms [18].

However, when transferring the specifications from FDMM to ADOxx, some specificities have to be taken into account (cf. [33]), for example that FDMM does not provide a first-order concept for expressing relations but rather uses a generic way of connecting instances of object types. In ADOxx on the other hand, it has to be decided whether a connection between object types should be implemented as a graphically represented relation – including the information what the graphical representation should look like based on defining statements in the GRAPHREP grammar [37] – or whether a reference attribute of the type INTERREF is to be used. Attributes of the type INTERREF permit to connect one object type instance to other object type instances or other model types. However, a graphical representation for these types of connections is not automatically provided by the platform. From the specifications in FDMM, we derive according ALL code that can be directly executed by the ADOxx platform. Likewise, also the UI-based ADOxx development toolkit could be used.

ALL code excerpts:

```
MODELTYPE \"CTQ-/CTB-Model\" from:none plural:\"CTQ-/CTB-Models\" pos:2
not-simulateable bitmap:\"db:"\\\\CTQ_CTB.bmp\"
       INCL \"VOC\"
       INCL \"VOB\"
       INCL \"CTQ\"
...
CLASS <CTQ> : <Critical to Customer-Business>                    1
...
  CLASSATTRIBUTE <GraphRep>
       VALUE "GRAPHREP
       PEN color:$00007f w:1pt
       FILL color:$eaea72                                        2
...
  ATTRIBUTE <Show quality dimension>
       TYPE ENUMERATION
       FACET <EnumerationDomain>
       VALUE "Yes@No"
       ...
```

By this transition, all model types that were previously defined in FDMM are mapped to corresponding model types in ALL. This is exemplified for the CTQ-/CTB-Model type in the ALL code excerpts above. Each object type in FDMM is transferred to a class or a relationclass depending on its graphical representation. In the example, the object type "CTQ" as defined in FDMM is represented as an equivalent class in ALL (marker **1**). The attributes in FDMM either become class attributes (e.g., "GraphRep") or attributes (e.g., "Show quality dimension") (marker **2**), depending on whether they are predefined in ADOxx or represent user-defined attributes.

For the generation of reports based on queries, the formal definition of the queries needs to be transferred to executable code as well. This is achieved on the ADOxx platform by using AQL. The purpose of AQL is to issue queries on model instances similar to the way in which SQL is used in the area of databases [37]. For example, to query all VOCs assigned to a certain CTQ (see equations (37) and (38)), the following AQL statements are specified, which can be directly executed by ADOxx as well:

```
I)   ({"Reduce complete duration of check-in to 10 minutes":"CTQ"}<-
     "Derive critical factor") AND (<"Core statement">)
II)  ({"Reduce   cycle-times":"Core   statement"}<-"Condense")   AND
     (<"VOC">)
```

Fig. 4 shows an exemplary screenshot for the CTQ-/CTB-Model.

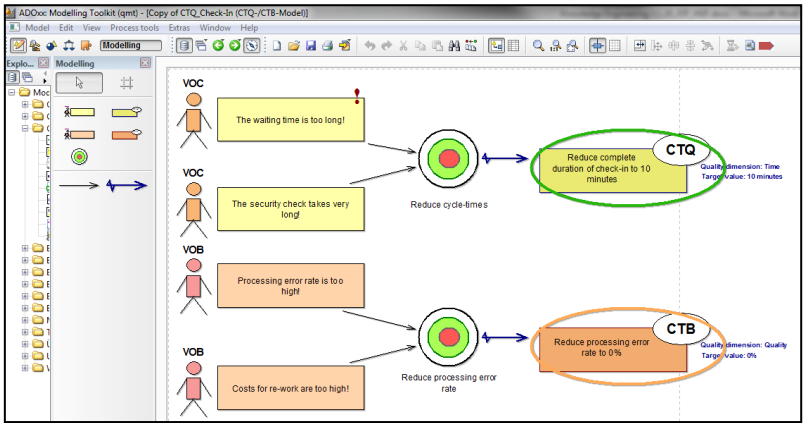

Fig. 4. Example of a CTQ-/CTB-Model Instance in ADOxx taken from the Prototype

5 Discussion

The development process as shown in Fig. 1 combines semi-formal (meta model), formal (FDMM), and executable (e.g., ALL) representations for realizing a graphical modeling tool which supports BPI initiatives. The meta model of the BPI roadmap structures the domain (cf. [13]) and captures knowledge on the problem-solving process (cf. [18]), namely the goal-oriented conduction of BPI projects. The formal specification of the meta model prepares the ground for its implementation. It transfers the knowledge on the problem domain and the problem-solving process into an implementation-oriented representation (cf. [18]), capturing mandatory information for creating the prototype. This includes ALL code as well as AQL expressions. As shown in section 4, the FDMM constructs can be directly mapped to ALL as well as AQL. The development of a graphical modeling tool is thus significantly accelerated.

A central benefit of the BPI roadmap is the logical interrelatedness between the BPI techniques. The results produced by a technique as output (e.g., CTQs and CTBs) are thus taken up and processed further by a subsequent technique. This peculiarity is reflected in the meta model by interrelated classes across model types (cf. [10]). In total, 11 corresponding references exist in the meta model of the BPI roadmap. This high degree of interrelatedness becomes obvious in the formal specification by the FDMM representation of references from one object type to another object type across model types (e.g., equation (26)). In this particular aspect the formalization of the BPI roadmap is different from those of other modeling methods. In addition, the precise specification of queries suggested for the BPI roadmap (cf. [10]) is a further differentiator from comparable formalization efforts in other application domains. The formulation of queries, expressed as formal expressions (see section 3), is a prerequisite for analyses and the machine-based generation of reports (cf. [40]) – a particular feature required in BPI initiatives to communicate results achieved (cf. [5]).

Main users of the BPI roadmap and the tool are team members engaged in BPI projects at a company. But the tool can also be applied in an academic setting to demonstrate the general functioning of BPI projects, making students potential users as well. Therefore, a variety of requirements were defined for the roadmap and the tool, respectively (cf. [10]). The FDMM formalization allows validating the conformity of the design specification with these functional requirements. For example, in the research at hand it could be assessed whether the value ranges for data types as defined meet user expectations or not. E.g., it could be decided whether the values "Quality", "Time", etc. were perceived as suitable for specifying the attribute "Quality-dimension". Thus, formalizations help to clarify ambiguous user requirements and to avoid misconceptions (cf. [16]). Mathematical procedures may be used for checking the consistency and syntactic correctness of the specification on FDMM level (cf. [16]). This is also highlighted in Fig. 1 (cycle at activity "formal specification using FDMM"). Once the prototype is established, additional evaluative conclusions can be drawn on the results produced throughout the development (see Fig. 1). For example, it may be judged whether the implementation-oriented representation is executable in an error-free manner with fast response times or not. An error-free execution was given for the prototype in this research confirming the correctness and the validity of the formalization. Further, the feedback of practitioners in a currently running usability study is expected to be highly valuable for evaluating the design of the BPI roadmap as a meta model.

6 Conclusion, Lessons Learned and Outlook

In our paper, we introduced a formalization of the BPI roadmap using FDMM. The purpose was to derive implementation-oriented representations to prepare the ground for the creation of a prototype. The BPI roadmap was initially developed to conduct BPI projects in a goal-oriented manner. To illustrate the functioning of FDMM, we chose a central model type, namely the CTQ-/CTB-Model.

In the project, we learned that despite the challenges when using FDMM it helps to bridge the gap between functional requirements derived from the problem domain and implementation-related aspects. Further, FDMM expects the user to precisely define attributes and corresponding values, and thus avoids uncertainty and ad-hoc decisions during implementation. Our research enables practitioners to benefit from a tool facilitating the efficient communication, documentation and processing of results and process knowledge emerging in BPI projects. Further, it became evident, that FDMM is beneficial for the development of software to come to an implementation-based representation of domain knowledge and problem-solving procedures for BPI. Scientists and practitioners are provided with means to significantly speed up development projects, building graphical modeling tools to support BPI initiatives. Based on a sound formalization, misconceptions may be avoided and the functional design can be validated (cf. [16]). This is extremely helpful considering the multitude of references between the model types of the BPI roadmap and the importance of reports in the BPI context. There are, nonetheless, some limitations to this study. First, due to page re-

strictions and reasons of clarity, the formal specification of the BPI roadmap was only exemplified for one model type. However, the basic principles of FDMM are identical for the other model types. Additionally, we built the examples around a central model type of the BPI roadmap to provide representative insights. Second, a usability test of the prototype is still running. While the first feedback received is promising, a concluding statement on its usability cannot be made yet. In future work, we intend to enhance the prototype by an interface to social networks for automatically retrieving customer statements (VOCs) from corresponding company pages (cf. [41]).

References

1. Davis, D.: 3rd Biennial PEX Network Report: State of the Industry – Trends and Success Factors in Business Process Excellence, Report (2013)
2. Greenberg, P.: The impact of CRM 2.0 on customer insight. Journal of Business & Industrial Marketing 25, 410-419 (2010)
3. Pande, P., Neumann, R., Cavanagh, R.: The Six Sigma Way – How GE, Motorola and other top companies are honing their performance. Mc Graw Hill, New York et al. (2000)
4. Mellat-Parast, M.: Supply chain quality management: An inter-organizational learning perspective. International Journal of Quality & Reliability Management 30, 511-529 (2013)
5. Seethamraju, R., Marjanovic, O.: Role of process knowledge in business process improvement methodology: a case study. BPMJ 15, 920-936 (2009)
6. Dalkir, K.: Knowledge management in theory and practice. Elsevier, Amerstdam (2005)
7. Nonaka, I.: The knowledge-creating company. HBR 85, 162-171 (2007)
8. Erdani, Y., et al.: Ternary grid as a potentially new technique for knowledge elicitation/acquisition. 2nd IEEE International Conference on Intelligent Systems, pp. 312-315.
9. Maier, R.: Knowledge management systems: Information and communication technologies for knowledge management. Springer, Berlin/Heidelberg (2007)
10. Johannsen, F., Fill, H.-G.: Codification of Knowledge in Business Process Improvement Projects. In: 22nd European Conference on Information Systems, Tel Aviv.
11. Shadbolt, N.R., Smart, P.R.: Knowledge Elicitation: Methods, Tools and Techniques (prepublication). In: Wilson, J.R., Sharples, S. (eds.) Evaluation of Human Work, CRC (2015)
12. Zellner, G.: A Structured Evaluation of Business Process Improvement Approaches. BPMJ 17, 203-237 (2011)
13. Karagiannis, D., Höfferer, P.: Metamodels in Action: An Overview. In: ICSOFT 2006 – 1st International Conference on Software and Data Technologies, pp. 27-36, Setúbal (2006)
14. Anaby-Tavor, A., et al.: Insights into enterprise conceptual modeling. Data & Knowledge Engineering 69, 1302-1318 (2010)
15. Gavrilova, T., Andreeva, T.: Knowledge elicitation techniques in a knowledge management context. Journal of Knowledge Management 16, 523-537 (2012)
16. Fraser, M.D., Kumar, K., Vaishnavi, V.K.: Strategies for incorporating formal specifications in software development. Communications of the ACM 37, 74-86 (1994)
17. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge engineering: principles and methods. Data & Knowledge Engineering 25, 161-197 (1998)
18. Angele, J., Fensel, D., Landes, D., Studer, R.: Developing Knowledge-Based Systems with MIKE. Automated Software Engineering 5, 389-418 (1998)
19. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28, 75-105 (2004)

20. Lipton, P.: Engineering and Truth. In: Philosophy of Engineering – Volume 1: Proceedings of a series of seminars held at The Royal Academy of Engineering, pp. 7-13 (2010)
21. Harrington, H.J.: Business Process Improvement – The Breakthrough Strategy for Total Quality, Productivity and Competiveness. McGraw-Hill, New York et al. (1991)
22. Adesola, S., Baines, T.: Developing and evaluating a methodology for business process improvement. BPMJ 11, 37-46 (2005)
23. Vakola, M., Rezgui, Y.: Critique of existing business process re-engineering methodologies: The development and implementation of a new methodology. BPMJ, 238-250 (2000)
24. Lee, K.T., Chuah, K.B.: A SUPER methodology for business process improvement. International Journal of Operations & Production Management 21, 687-706 (2001)
25. Karagiannis, D., Kühn, H.: Metamodelling Platforms. In: 3rd International Conference EC-Web 2002 – Dexa 2002, pp. 182-195, Aix-en-Provence (2002)
26. Bork, D., Fill, H.-G.: Formal Aspects of Enterprise Modeling Methods: A Comparison Framework. In: 47th HICSS, pp. 3400-3409. IEEE Press (2014)
27. Fill, H.-G., Redmond, T., Karagiannis, D.: FDMM: A Formalism for Describing ADOxx Meta Models and Models. In: ICEIS 2012, Poland (2012)
28. Poernomo, I.: The meta-object facility typed. In: ACM symposium on Applied computing, pp. 1845-1849, Dijon (2006)
29. Favre, L.M.: Formalization of MOF-Based Metamodels. In: Favre, L.M. (ed.) Model Driven Architecture for Reverse Engineering Technologies. Information Resources Management Association, pp. 49-79 (2010)
30. Jouault, F., Bézivin, J.: KM3: a DSL for Metamodel Specification. International Conference on Formal Methods for Open Object-Based Distributed Systems, pp. 171-185 (2006)
31. Fill, H.-G., Redmond, T., Karagiannis, D.: Formalizing Meta Models with FDMM: The ADOxx Case. In: Cordeiro, J., Maciaszek, L.A., Filipe, J. (eds.) Enterprise Information Systems 141, pp. 429-451. Springer Berlin Heidelberg (2013)
32. OMG: Object Constraint Language (OCL) – Version 2.4 (2014)
33. Fill, H.-G., et al.: A Formal Specification of the Horus Modeling Language Using FDMM. In: Alt, R., Franczyk, B. (eds.) Proceedings 11th Wirtschaftsinformatik, Volume 2, pp. 1165-1179, Leipzig (2013)
34. Tolvanen, J.-P., Kelly, S.: MetaEdit+: defining and using integrated domain-specific modeling languages. In: 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, pp. 819-820, ACM, New York (2009)
35. McNeill, K.: Metamodeling with EMF: Generating concrete, reusable Java snippets, www.ibm.com/developerworks/library/os-eclipse-emfmetamodel/ (access: 31.07.2014)
36. Ledeczi, A., et al.: The generic modeling environment. In: Workshop on Intelligent Signal Processing, IEEE Press (2001)
37. Fill, H.-G., Karagiannis, D.: On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. Enterprise Modelling and Information Systems Architectures – An International Journal 8, 4-25 (2013)
38. Kern, H., Hummel, A., Kühne, S.: Towards a comparative analysis of meta-metamodels. In: Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11, pp. 7-12. ACM, New York (2011)
39. Österle, H.: Business in the information age. Springer, Berlin et al. (1995)
40. Bräuer, S., et al.: Using a Generic Model Query Approach to Allow for Process Model Compliance Checking – An Algorithmic Perspective. In: Alt, R., Franczyk, B. (eds.) Proceedings 11th Wirtschaftsinformatik, Volume 2, pp. 1245-1259, Leipzig (2013)
41. Lehner, F., Fteimi, N.: Organize, socialize, benefit: how social media applications impact enterprise success and performance. In: 13th i-Know Conference, Graz, Austria (2013)