Acta Informatica Pragensia, 2015, 4(1): 30–43
DOI: 10.18267/j.aip.58
Peer-reviewed paper

Generated Report of the ORD BORM Model

Jakub Tůma[*], **Marek Pícka**[*], **Petr Hanzlík**[*]

Abstract

This contribution focuses on documentation of model-to-model transformation, respectively on converting model represented by BORM notation (Business Object Relation Modelling) to a HTML-based textual report. The transformation is from the management-level business process model into the operation-level business process model. The operation-level model should be textual and tailored for each individual participant. This article introduces a modular extension of OpenCASE software that utilizes OpenCASE knowledge base and API to generate textual reports automatically. This report is based on HTML language, because of the portability and easy distribution of the format. Such report is easy to understand even for business people without hard technical skills. We present both the transformation, and the modular extension on a case study.

Keywords: BORM, Transformation, Tool, Report, HTML, Case study.

1 Introduction

Models created in CASE tools are probably the most comprehensive description of system. However, these models, and relationships among them, can be difficult to understand. Even though BORM diagrams are generally easy to understand, it may still be challenging to fully comprehend the situations modelled, especially for non-specialists and businesspeople. A textual report describing BORM diagram can significantly contribute to better understanding of problem as a whole and increase the accessibility of information stored within models. A repository of CASE tool can be used as a source of data for generating such documentation. When a report is generated automatically from models, the safe consistency between the model and documentation is guaranteed. Another advantage is the possibility to reinstate documentation when source model changes.

We want to present our approach for flexible modelling of business processes both at the management and operations level. This approach consists of combining a suitable modelling method (BORM – Business Object Relation Modelling) and developing an original software tool (OpenCASE) to support it, as well as to perform automated model transformations.

This article focuses on transformation of source model created in CASE tool to textual output model and technologies used for implementation of this transformation. Input of this transformation is a model using BORM notation, output of the transformation is model represented as a HTML report.

[*] Department of Information Engineering, Faculty of Economics and Management, Czech University of Life Sciences, Prague, Kamýcká 129, 165 21 Praha 6, Czech Republic
✉ jtuma@pef.czu.cz, picka@pef.czu.cz, petr.hanzlik6@gmail.com

This research is based on the BORM method described in publications by (Knott, Merunka, Polak, 2000, 2006; Struska, Merunka, 2007; Struska, Pergl, 2009). The presented tool OpenCase (Pergl, Tůma, 2012) follows this modelling method, and uses it as input of transformation. Core transformations are based on the HOT (High Order Transformation) approach published by (Brambilla, Fraternali, Tisi, 2008). This transformation is presented on the case study which demonstrates the transformation from the management-level business process model into the operation-level business process model. More specifically the case study deals with the process of E-shop order goods.

2 Materials and methods

2.1 Methodology

In order to realize the proposed transformation, we have followed a stepwise research plan:

1. Define requirements for a suitable management-level business process model and operation-level business process model (BPM).
2. Describe how the transformation modelling method and the OpenCASE support these requirements.
3. Design a case study to illustrate the results.

2.2 Requirements for Management-Level and Operation-Level BPM

For purposes of this research, let us define that management level is focused on process orchestration, specifically:

- Terminology
- Logic of processes
- Relations of processes (transition, decomposition)
- Communication among participants
- Optimisation of the overall process

The language of models needs to support the aspects listed above. That is why a combination of *graphical* and *textual* language is usually used. The management level BPM specifies *terms* and their *relations* that are consequently manipulated in different ways:

- They need to be verified for correctness.
- They need to be communicated.
- They are used for reasoning.
- Various reports and statistics need to be calculated.
- They are often changed (they evolve).

This is why the management-level BPM needs to be some sort of **knowledge base**, not just a set of diagrams (graphical objects). Terms and their relations are necessary for full domain specification and to fulfil knowledge base.

By operational level, we mean concrete processes (operations) here, which are performed by assigned process participants (staff, systems). Systems (as participants) are software, thus subject of computer science and software engineering. For further discussion we will consider just human participants. We set following requirements on operation model:

- Language of the model is close to the language of participants.
- Model is accurate.
- Model contains just the necessary amount of details to perform operations required.
- Model is up to date and consistent with corresponding management-level model.

Staff is not supposed to be interested in the “big picture” at the operation level, they just need accurate instructions for performing their tasks, i.e. the management needs to answer their questions:

- What are the steps I should follow to successfully complete a task?
- How should I make decisions and select correct approach?
- What are the inputs I will get? From whom, how and when?
- What are the outputs I should produce? To whom shall I handle them, how and when?

To answer these questions, textual operation manuals are typically used, as operation-level staff does not have to understand graphical objects with abstract notations.

2.3 Business Object Relationship Modelling (BORM)

Business Object Relationship Modelling (Knott, Merunka, Polak, 2006; Polák, Merunka, Carda, 2003) is an object-oriented software engineering methodology, which has proven to be very effective in the development of business information and knowledge systems. Its effectiveness is achieved by unified and simple method for presenting all aspects of the relevant model. The BORM methodology makes extensive use of business process modelling (Knott, Merunka, Polak, 2000). BORM was designed as a method covering all phases of the software development. BORM focuses mainly on the first phases of the project also known as business analysis. BORM uses only limited, easily comprehensible group of concepts for every lifecycle phase. This makes it easier to understand even for the first-time users with almost no knowledge of business analysis.

Another fact that makes the BORM methodology more expressive is that it doesn´t need the division to static and dynamic views of the model and therefore does not bring a need of creation of different diagrams with a different viewpoints. BORM introduces the following types of diagrams:

- Business architecture diagram
- Object relationship diagram (ORD) (see Figure 1)
- Class diagram

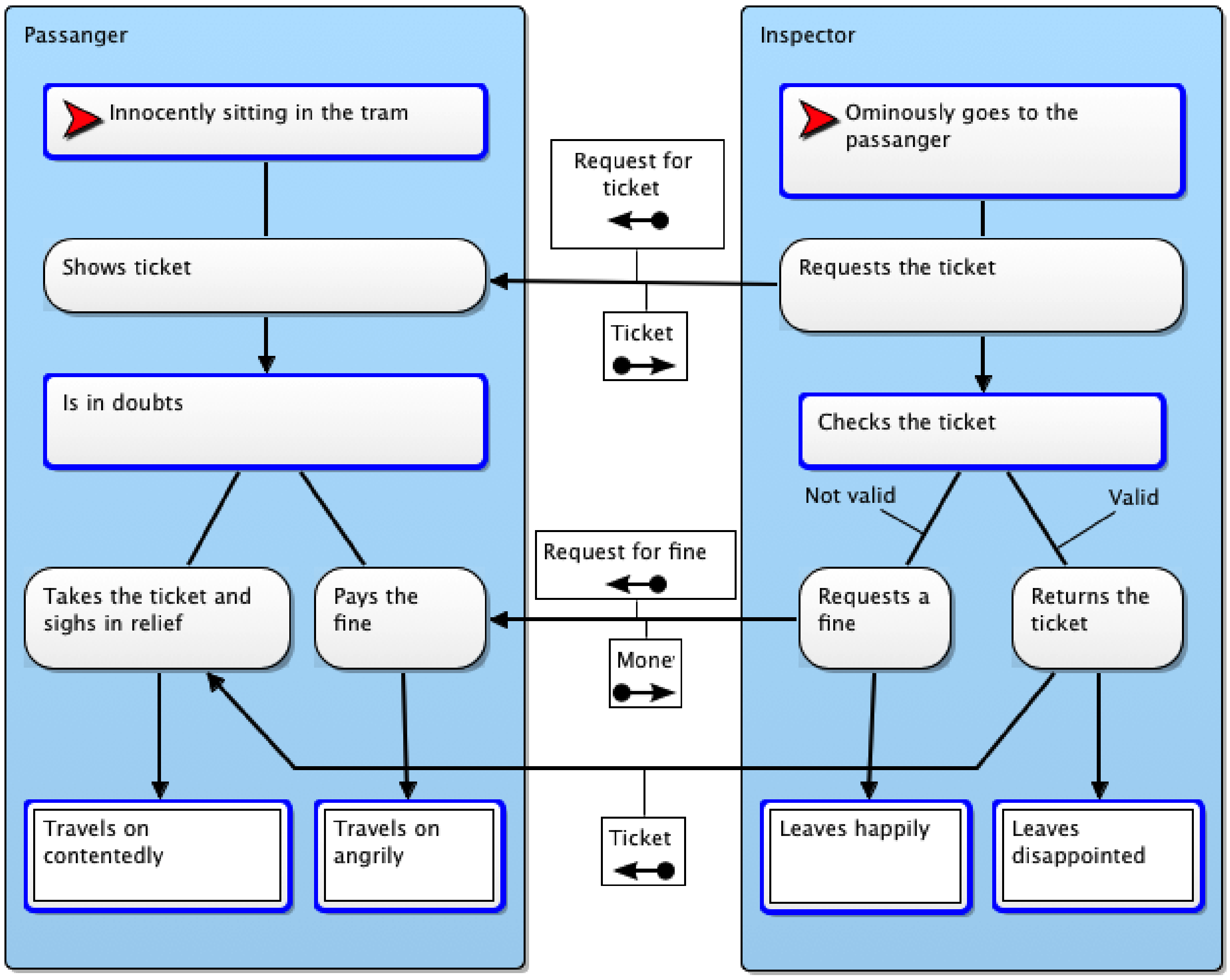


Fig. 1. *Example of Object Relationship Diagram (ORD). Source: (Pergl, Tůma, 2012)*

BORM represents every concept with the same symbols in the data structure, the communication or other diagrams. For visual presentation of the information BORM uses simple diagrams that contain only a necessary number of concepts and symbols. These concepts and symbols cover most of the needs for the initial description of the model and its processes. The following symbols belong to the symbols used in the initial description:

- Participant – an object representing the stakeholder involved in one of the modelled processes, which is recognised during the analysis.
- State – sequential changes of the participants in time are described by these states.
- Association – data-orientated relation between the participants.
- Activity – represents an atomic step of the behaviour of the object recognised during the analysis.
- Communication – represents the data flow and dependencies between the activities. Data may flow bidirectionally during the communication.
- Transition – connects the state-activity-state and represents changes of the states through activities.
- Condition – expresses constraint that holds for the communication or activity, (Polák, Merunka, Carda, 2006; Šplíchal, Pergl, Pícka, 2011).

2.4 OpenCASE

OpenCASE Tool (2014) is a CASE tool designed to support the research in the field of conceptual modelling and related ontologies. Snapshot of OpenCASE prototype is presented in the Figure 2. It is built upon the Eclipse Modelling Framework and it utilizes many of its advanced possibilities. Right now, we have implemented the BORM method's Business Architecture Diagrams and Object Relation Diagrams (Pergl, Tůma, 2012).

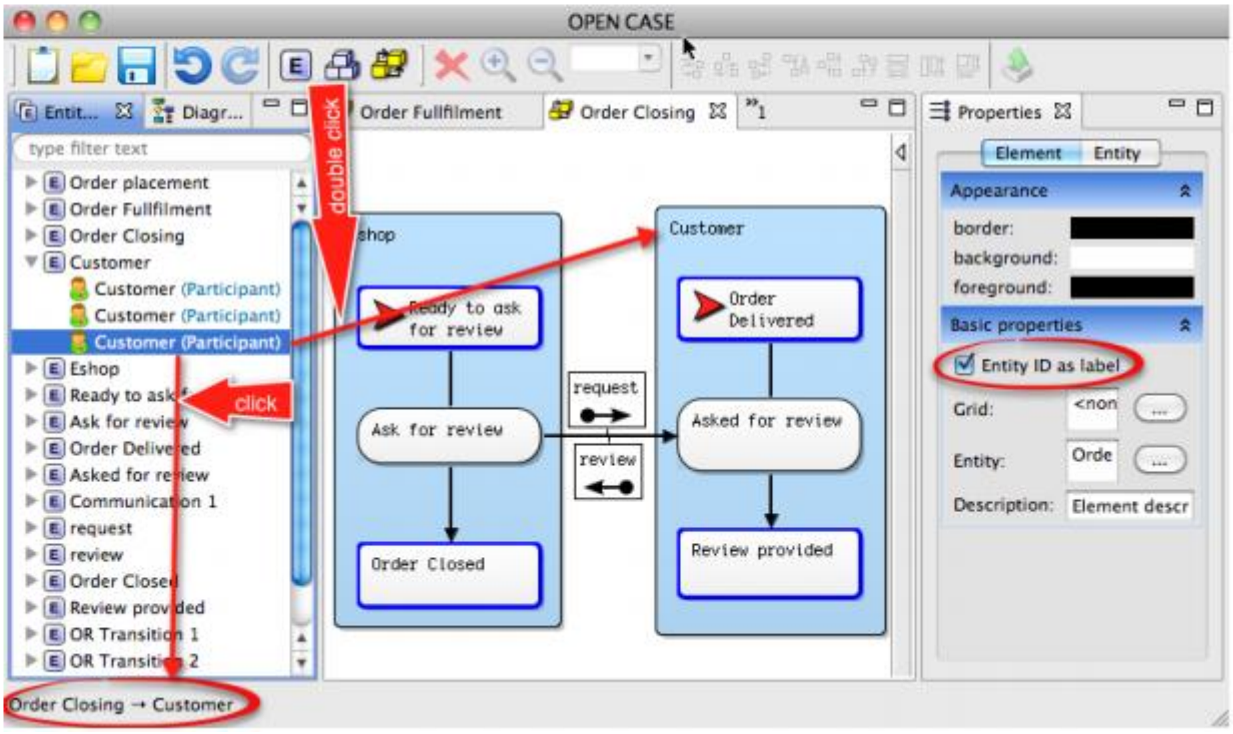


__Fig. 2.__ Snapshot of OpenCASE prototype. Source: (Pergl, Tůma, 2012)

3 Results and discussion

3.1 Documentation generating

The case study demonstrates the transformation from management-level business process model into operation-level business process model. As we specified in the requirements, operation-level model should be textual and tailored for each participant. This is where we utilize the OpenCASE as a knowledge base and its API to generate HTML page for each participant. This HTML output is similar to SBVR, (Cabot, Pau, Raventós, 2010; OMG, 2008; Booch, Rumbaugh, Jacobson, 1999).

The transformation is based on approach HOT (High Order Transformation) and this model generation has theoretical background in publication (Šplíchal, Pergl, Pícka, 2011). Modelling tool OpenCASE includes this transformation as an extendable module.

The higher order transformation phase addresses mapping (see Figure 3.), guaranteeing the coherence between the conforms to relationship of the two technical spaces. This task is performed in two subtasks:

1. a promotion transformation obtains the DSL metamodel by promoting the model M1 resulting from the model generation phase to metamodel (M2);

2. a manually defined HOT derives the M1T transformation by translating the M2T transformation, and eventually analyzing the structure of the DSL metamodel published by (Brambilla, Fraternali, Tisi, 2008).

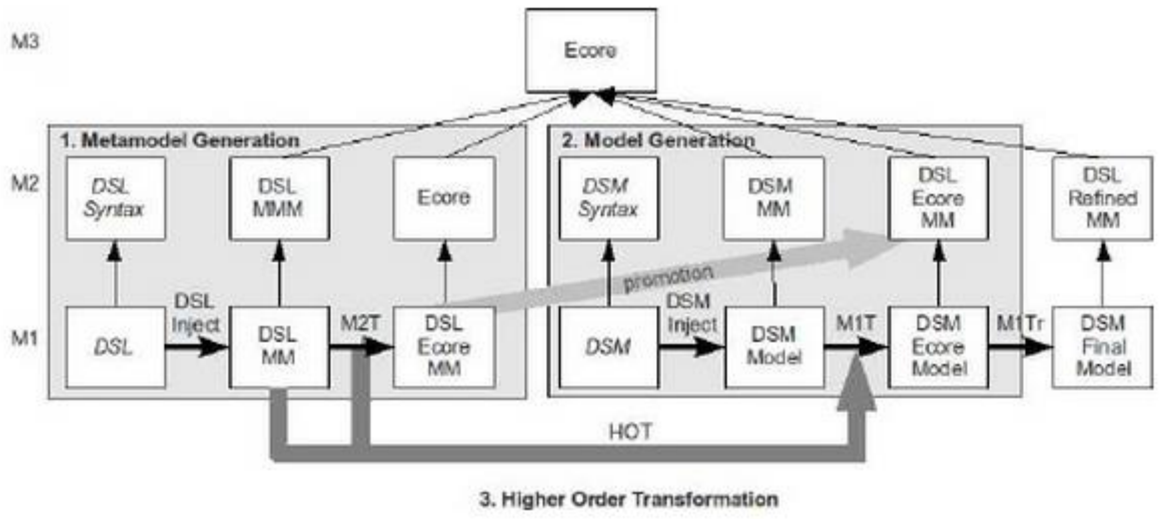


Fig.3. *Diagram of framework using HOT. Source: (Brambilla, Fraternali, Tisi, 2008)*

HOT approach was theoretically described by (Brambilla, Fraternali, Tisi, 2008) and cited in this paper above and shown in Figure 3. This HOT implementation is used for report generation of the ORD BORM Model.

A programing implementation of this mechanism is presented below, in the Snippet 1 and Snippet 2. These implementations are used for HTML report generation from ORD BORM Model.

```
(defn or-report [or-diagram]
 (let [name    (first (return or-diagram :entity :id))
       reports (map participant-report (participants or-diagram))
       notes (note-report (or-diagram))
       ]
   (list*
     (element :h1 {} (str "OR Diagram: " name))
     reports
     notes
     )))
```

Snippet 1: Main function for generating report from ORD diagram

```
(defn -write [this manager file]
 (with-open [writer (java.io.OutputStreamWriter. (java.io.FileOutputStream. file) "UTF-8")]
   (let [ords    (diagrams (.getProject manager) :ORDiagram)
    reports (mapcat or-report ords)]
  (emit (xhtml-page "XHTML Report" reports) writer :encoding "UTF-8"))))
```

Snippet 2: Export function for creating XHTML

Snippet number 1 shows the *main* function. This function is called OR-report, and it generates a semi model from the inner structure (metamodel of ORD BORM model) representing OR-diagram. This semi model is processed by the *write* function (see Snippet 2) which produces a XHTML report page. These functions have been implemented using function programming paradigm, which takes advantage of simplification of equational reasoning (Wadler, 1992). More information about functional programming can be found in Wadler (1992). DOM (Document Object Model) is used for transformation to HTML. DOM is tree structure of HTML transformation output.

To generate documentation (XHTML report) was implementated in functional language. Functional programming is simplicity of equational reasoning Wadler (1992) and more info about functional programming in publication (Wadler, 1992). DOM (Document Object Model) is used for generating or more precisely transformation to HTML. DOM is tree structure of HTML transformation output.

3.2 Case study

The case study is based on the processes order goods from e-shop. The case study was written with team cooperation based on the course information management. Course information management is part of the field Project management at the Czech University of Life Sciences, Prague. In order to demonstrate the principals of proposed BORM to HTML transformation and other concepts outlined in this paper, a part of the processes related to order goods from e-shop is presented in a simplified version.

The order goods process is carried out by cooperation among five participants: *Customer*, *Deliverer*, *Economics department*, *Order processor* and *Supplier*. This process is shown in details in Figure 4.

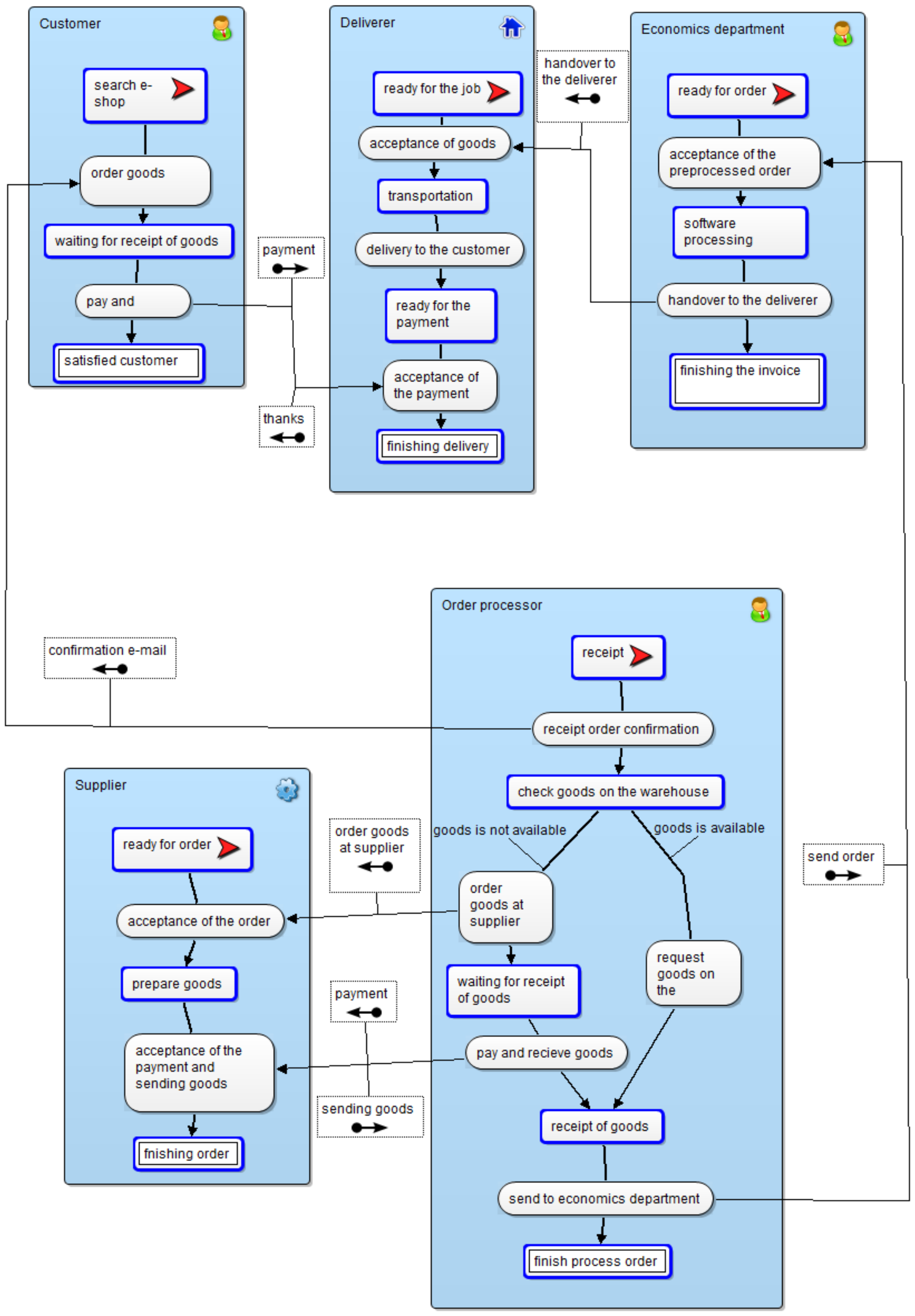
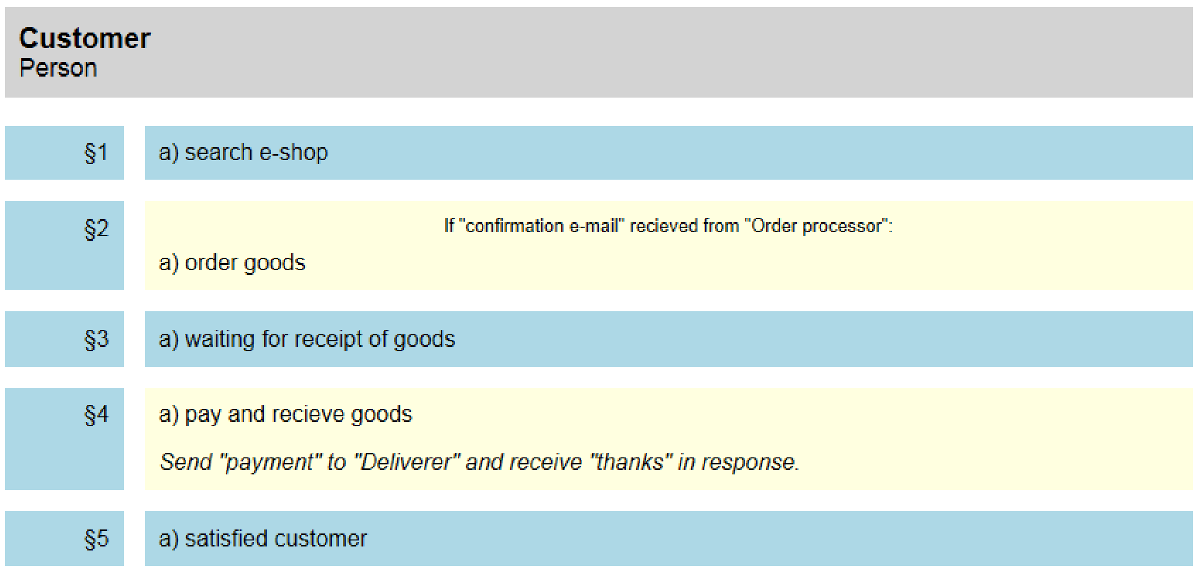


***Fig. 4.** Case study management process model in BORM: E-shop order goods. Source: authors.*

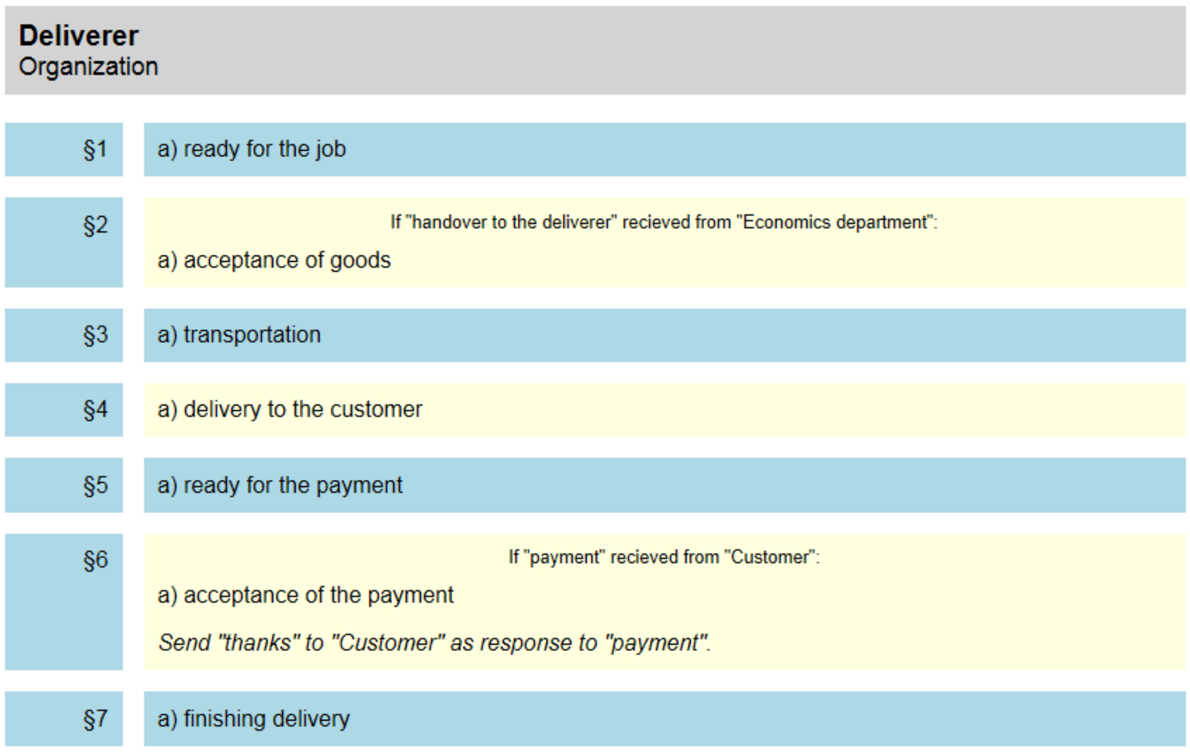
The HTML operation manuals are generated for each participant (Figures 5 – 9).



Customer
Person

§1	a) search e-shop
§2	If "confirmation e-mail" recieved from "Order processor": a) order goods
§3	a) waiting for receipt of goods
§4	a) pay and recieve goods *Send "payment" to "Deliverer" and receive "thanks" in response.*
§5	a) satisfied customer

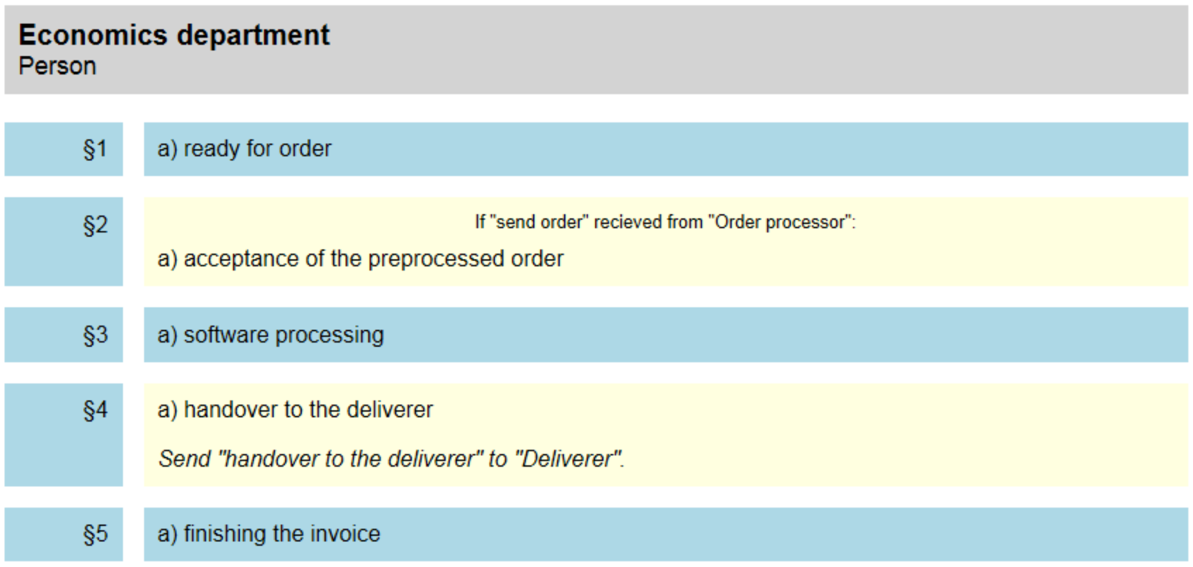
Fig. 5. *Operation model (manual) for participant Customer. Source: authors.*



Deliverer
Organization

§1	a) ready for the job
§2	If "handover to the deliverer" recieved from "Economics department": a) acceptance of goods
§3	a) transportation
§4	a) delivery to the customer
§5	a) ready for the payment
§6	If "payment" recieved from "Customer": a) acceptance of the payment *Send "thanks" to "Customer" as response to "payment".*
§7	a) finishing delivery

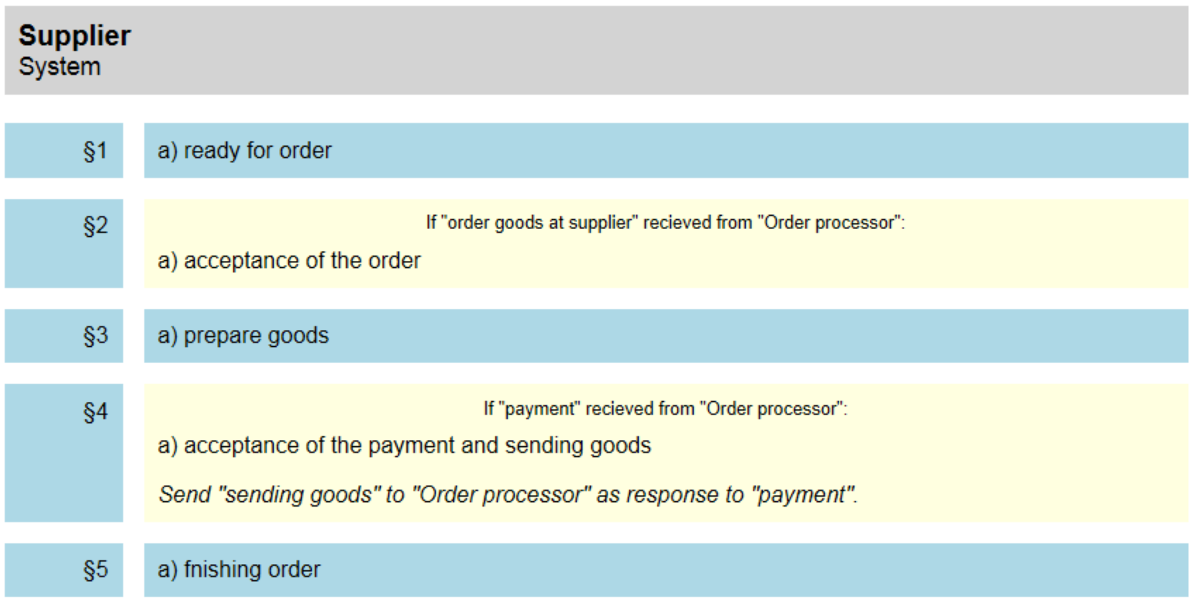
Fig. 6. *Operation model (manual) for participant Deliverer. Source: authors.*



Economics department
Person

§1	a) ready for order
§2	If "send order" recieved from "Order processor": a) acceptance of the preprocessed order
§3	a) software processing
§4	a) handover to the deliverer *Send "handover to the deliverer" to "Deliverer".*
§5	a) finishing the invoice

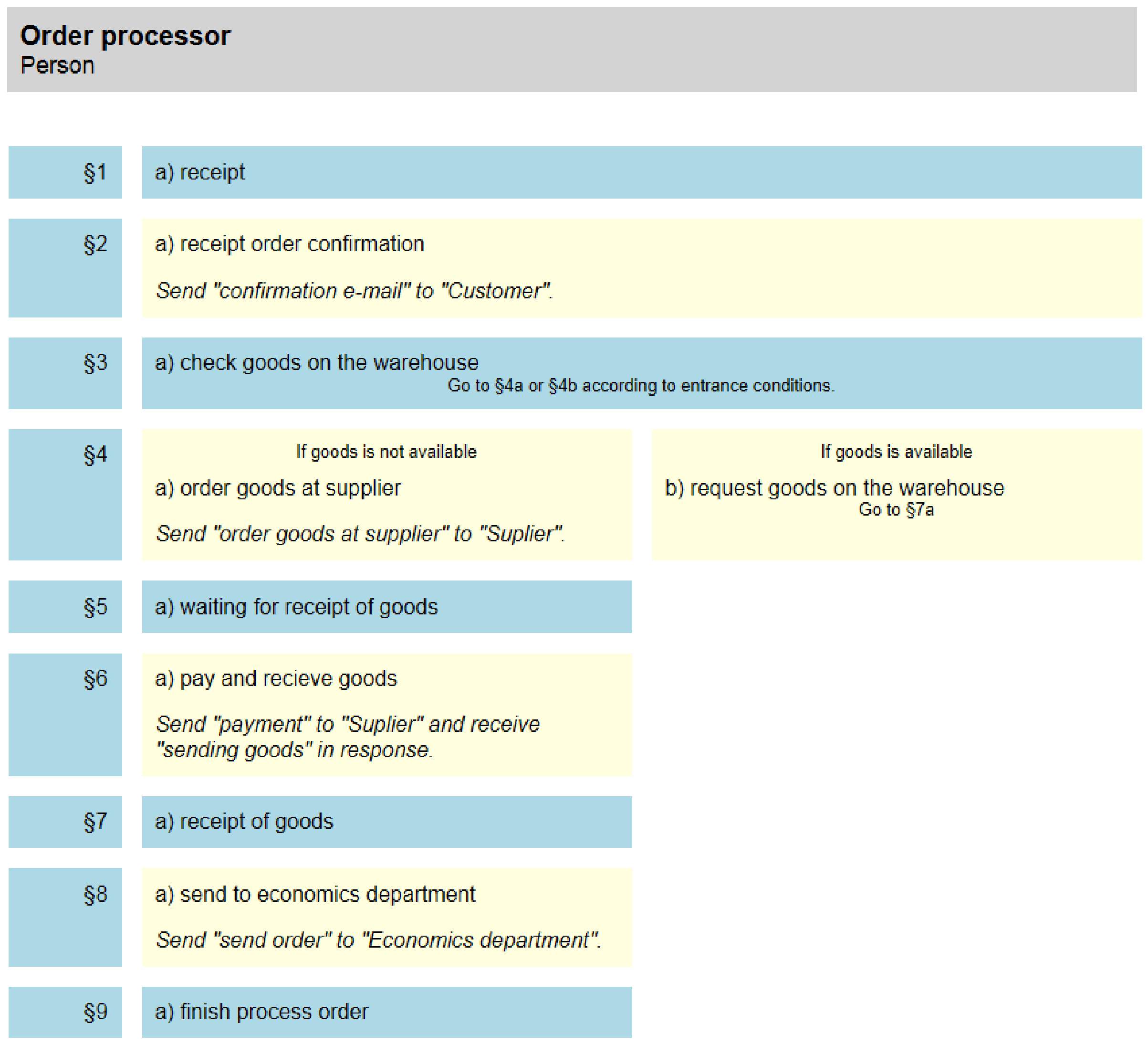
Fig. 7. *Operation model (manual) for participant Economics department. Source: authors.*



Supplier
System

§1	a) ready for order
§2	If "order goods at supplier" recieved from "Order processor": a) acceptance of the order
§3	a) prepare goods
§4	If "payment" recieved from "Order processor": a) acceptance of the payment and sending goods *Send "sending goods" to "Order processor" as response to "payment".*
§5	a) fnishing order

Fig. 8. *Operation model (manual) for participant Supplier. Source: authors.*


Order processor
Person

§1	a) receipt	
§2	a) receipt order confirmation *Send "confirmation e-mail" to "Customer".*	
§3	a) check goods on the warehouse Go to §4a or §4b according to entrance conditions.	
§4	If goods is not available a) order goods at supplier *Send "order goods at supplier" to "Suplier".*	If goods is available b) request goods on the warehouse Go to §7a
§5	a) waiting for receipt of goods	
§6	a) pay and recieve goods *Send "payment" to "Suplier" and receive "sending goods" in response.*	
§7	a) receipt of goods	
§8	a) send to economics department *Send "send order" to "Economics department".*	
§9	a) finish process order	

Fig. 9. *Operation model (manual) for participant Order processor. Source: authors.*

The transformation engine is handling correctly the branching, loops and hierarchical processes (process in a state). It uses paragraphs labelling to navigate the user along the process flow.

4 Discussion

There are currently two CASE tools that are able to operate with BORM ORD notation – CraftCase and OpenCASE. OpenCASE is a commercial product developed by CraftCase Company. OpenCASE on the other hand is a platform, which was developed and designed at the Faculty of Economics and Management, Czech University of Life Sciences, in cooperation with Faculty of Information Technology, Czech Technical University. It provides an open modular platform available for free for community of registered scholars and experts. It is based on generally available Eclipse plugin. Any interested developer can hence contribute to the development of the platform, and take part in experiments and research. Thanks to this background, OpenCASE can currently offer several state of art components, such as above presented transformation module.

5 Future works

Possible future works:

- *Listings,* like all input/output flows from/to a participant.
- *Calculation of metrics* (like numbers of states and activities in participants) that may be used for *complexity estimations* Struska and Pergl (2009), Struska and Merunka (2007).
- *Calculation of statistics,* e.g. about data flows and communications (which participants communicate the most/least, above/below average, etc.).
- *Semantics checks*: there is a starting state in every participant, at least one final state (According to the BORM, there may be exceptions to these rules, see (Gronback, 2009) for more details.),
- *Conceptual normalization* to be implemented in tool that is described by Molhanec (2011).
- Any further custom reporting / calculations / processing.

6 Conclusion

In this paper we presented our solution that generates HTML documentation from BORM diagrams and supports business process engineering in general. This is accomplished using a combination of suitable modelling method and notation (BORM) and software tool (OpenCASE). The key principle of the solution is that the modelled process is not just a diagram, but a whole knowledge base that may be used in operations, reporting, decision making and other areas. We presented one of consequences of this assumption: automatic generation of operations manuals.

The OpenCASE is created as an open platform for studying BORM method and business modelling in general. Apart from this, it is already a stable tool for effective drawing of BORM models and their management. It is implemented using open architecture based on Eclipse plugins, which makes it extensible and scalable. The presented extension of OpenCASE tool makes the BORM models more accessible for businesspeople who are typically not familiarised with state-based diagrams and business diagrams.

Acknowledgements

Many thanks to my supervisor Vojtěch Merunka and to the team leader Robert Pergl for a lot hints and mental support. This paper documents summarise previous and future developing process of dissertation thesis.

References

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modelling Language User Guide*. Massachusetts: Addison-Wesley Longman.

Brambilla, M., Fraternali, P. & Tisi, M. (2008). A Transformation Framework to Bridge Domain Specific Languages to MDA. In *MoDELS Workshops 2008*, (pp. 167-180). Berlin: Springer.

Cabot, J., Pau R. & Raventós, R. (2010). From UML/OCL to SBVR specifications: A challenging transformation. *Information Systems*, 35(4), 417-440.

Gronback, R. C. (2009) *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Massachusetts: Addison-Wesley Longman.

Knott, R. P., Merunka, V., & Polak, J. (2000). Process modeling for object oriented analysis using BORM object behavioral analysis. In *IEEE International Conference on Requirements Engineering* (pp. 7-16). Chicago: IEEE & ACM.

Knott, R., Merunka, V. & Polák, J. (2006). The BORM Method: A Third Generation Object-Oriented Methodology. In Liu, L. & Roussev B. (Eds) *Management of the Object-Oriented Development Process*. (pp. 337-360). Hershey: IGI Publishing.

Merunka, V., Milena, S., Arvilla, C., Plas, M. & Tuma, J. (2013). BORM-II and UML as accessibility process in knowledge and business modelling. In *22nd International Scientific Conference Agrarian perspectives* (pp. 155-163). Prague: Czech University Life Sciences Prague.

Molhanec, M. (2011). Some Reasoning Behind Conceptual Normalisation. In *Information Systems Development: Business Systems and Services* (pp. 517-525). Berlin: Springer.

OMG. (2008). Semantics of Business Vocabulary and Rules (SBVR) Specification, v 1.0 (formal/08-01-02). Retrieved from http://doc.omg.org/formal/08-01-02.pdf.

OpenCASE Tool. (2014). Retrieved from http://www.opencase.net.

Pergl, R., & Tůma, J. (2012). OpenCASE–a tool for ontology-centred conceptual modelling. In *Advanced Information Systems Engineering Workshops* (pp. 511-518). Berlin: Springer Heidelberg.

Polák, J., Merunka, V. & Carda, A. (2003). *Umění systémového návrhu: objektové orientovaná tvorba informačních systémů pomocí původní metody BORM*. Prague: Grada.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The Unified Modelling Language Reference Manual.* Massachusetts: Addison-Wesley Longman.

Šplíchal, P., Pergl, R. & Pícka, M. (2011) BORM Model Transformation. *Systémová integrace*, 18(2), 112-123.

Šplíchal, P. (2011) Model Transformation. In *20th International Scientific Conference Agrarian perspectives* (pp. 423-430). Prague: Czech University of Life Sciences.

Steinberg, D., Budinsky, F., Merks, E. & Paternostro, M. (2008). *EMF: eclipse modelling framework*. New York: Pearson Education.

Struska Z. & Merunka V. (2007) BORM points new concept proposal of complexity estimation method. In *9th International Conference on Enterprise Information Systems* (pp. 580-586). Berlin: Springer.

Struska Z. & Pergl R. (2009) BORM-points: Introduction and Results of Practical Testing. In *Lecture notes in business information processing: Enterprise information systems* (pp. 590-599). Berlin: Springer.

Wadler, P. (1992). The essence of functional programming, In *19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (pp. 1-14). New York: ACM.