

For what Purpose and by which Means Should we Describe Organizational Processes?

A Case Study from an Industrial Project

Thorsten Spitta¹, Juliane Kaup²

¹ University of Bielefeld, Germany

² GE Jenbacher, Austria

Abstract: The paper outlines a comparison of UML Activity Diagrams (AD) with the EPC 'blueprints' in a worldwide SAP-project of a large German enterprise. Our case study was performed in the firm's sales company in a North-European country. The paper designs artifacts for organizational process modeling, compared with the Bunge-Weber-Wand (BWW) ontology. They are matched against two widespread graphical notations, the EPC and the AD. Because of fewer drawbacks of the UML against the ARIS concept, the case study used the AD, extending the UML by a system view, which is demanded by the BWW ontology. A systems view appears essential in large projects in order to facilitate communication with the top management.

1 Introduction

Since Davenport's [Da93] and Hammer & Champy's [HC93] famous books the business and academic world noted a wave of attempts to reorganize enterprises and to model their business processes. The message of Hammer & Champy was clear: Substantial improvements in the firm's organization are mainly based on processes and less based on structural redesign. The main driver for successful redesigns of the flow of material, products, money and data is information technology [HC93, 35&44]. Implementation projects for new information systems should start with an analysis of business processes [Sc94], pursuing the goal of a *process-oriented organization* [Ma99]. This requires efficient and precise notations, which are well suited for communication between the organization's actors.

Graphical notations are generally preferred [BM06, 38], some are shown in table 1.

Table 1: Common notations for process description

Year	Name	Short	Reference
1962	Petri-Nets	PN	[AH00]
1993	Event-Driven Process Chains	EPC	[Sc94]
1997	UML Activity Diagrams	AD	[MG04]
2003	Business Process Management Notation	BPMN	[Wh04]

While the notations one to three are (graphical) *general purpose* languages at a first glance, the BMPN is explicitly designed as a *domain specific* language "readily understandable by all business users" [BP03, 14]. The BPML is structurally nearly equivalent to AD [Wh04] and is supposedly going to be unified with the UML as a specialization (see [OMG08]¹). The EPC is not called a *language* by its authors [Sc99], because it is mostly promoted with the development concept *architecture of integrated information systems* (ARIS) and named as a tool accordingly. Regarding the context of ARIS, the EPC also appears as a domain specific language for business processes only.

Petri Nets, EPC and AD have been analyzed in a number of evaluations on their suitability for business process modeling ([Aa99], [AH00], [Sö02], [Wo05], [Ru06]) and on their modeling power to gain syntactical and semantical correct models (see [BM06] for UML in general). While the first four references and other studies only investigate the syntactical power of the evaluated language, using *workflow patterns* as evaluation framework, we could only find one study according to the semantics, published by Bunge-Jones & Meno. They investigated the semantic power of UML for good decomposition in an empirical study [BM06].

However, evaluating the semantic power of a language requires more than formal patterns, it needs an *ontology* on which the grammar of the language is built [WW02]. An ontology is the theoretical notion of the basic constructs of any formal language. *Modeling* in Wand & Weber's words is "identifying the phenomena to be modeled and mapping the phenomena into the grammar's constructs" [WW02, 368]. The most cited ontology is called the *Bunge-Weber-Wand Ontology* (BWW) [WW90], which is also used in this paper. The ontology serves as a framework to evaluate the semantic power of a formal language, respecting from a philosophical view, that the ontology is believed to be a true axiomatic system only [Wy04].

The central question of our paper is: *Are ADs suitable for process modeling in large of-the-shelf software projects?* The main focus of those projects is the organization, not software technique. If the system to be implemented is SAP's ERP-system, known as R/3[®], the scenario of process descriptions is dominated by "blueprints", depicted as EPCs [CL99]. The blueprints play an important role for communication in project groups, between them and with other members of the organization. Our hypothesis in this context was: *The AD is better suited for handling and communication than the EPC.*

This is an important feature in *large* enterprises at least. Our case study was performed in an enterprise of nearly 30,000 employees worldwide. It is located in one headquarter, several production locations and selling units in 80 countries. Aside administering the orders of business customers, the selling units are also responsible for delivery and maintaining products. This means, that successful business of the enterprise is based on relative complicated material and data flows. These details may demonstrate, that it is worthwhile to regard only one case instead of performing an experimental design with statistical analysis.

¹ via the link www.bpmi.org

The paper proceeds as follows: Section 2 outlines those elements of the BWW ontology, which seem to be most important to our subject, the organizational flows of material and data, naming these subjects as artifacts of a framework. Section 3 discusses the competing frameworks ARIS and UML in behalf of the process notations EPC and AD and sketches some pragmatic extensions of the AD, realized by stereotypes. Section 4 shows and discusses some results from the case study, which could probably be generalized. Section 5 draws some conclusions and looks on further work to be done.

2 Elements for organization models

In this section we evaluate those parts of the BWW ontology which seem to be essential to describe organizations structurally and dynamically with regard to flows between their elements.

2.1 The BWW Ontology

First we outline the **static** basics of the BWW view of the world. The central construct is a *thing* possessing *properties*. Properties are visible as *states*, expressed in *state variables*. The values of these variables obey *laws*, expressing natural or artificial constraints deciding which states are *lawful* and which not. Only a small subset of the possible state space is considered as lawful.

The **dynamic** view of the BWW ontology is based on states and laws. Things cause *events* changing their own or other thing's properties. At least one state variable must be changed, but only events leading to lawful states are lawful. E.g. the event

'removal x items from thing *stock* with state *quantity* $q = 6$ '

is not lawful for $x > q$. The event in our example may be caused by a thing *stock-bookkeeping* which causes a *history* of the thing *stock*. *Stock-bookkeeping acts* on *stock*, with other words, the two things are *coupled*. Wand expresses it like this: "It is an ontological principle that every change is tied to things and every thing changes" [Wa96, 282]. Changes are modeled as *state transitions*.

There are two types of events, *internal* and *external* events. An internal event is caused by a thing itself, an external by another thing. The internal transitions of a thing can be completely specified as a *transition law* of the thing, which only may contain lawful states, also called *stable states*. *Unstable states* are not completely specifiable and may only be caused by external events.

Based on the fundamental view of statics and dynamics, Weber & Wand look bottom-up, defining a **system**. A *system* is a well defined set of interacting things. "Well defined" means, that all things of the system must be coupled. A system is surrounded by an *environment*, comprising things acting on the system or receiving from the system. Things are the only items of a system, which are able to receive events from or to produce events for things in the environment. Typical examples for things in the environment of an information system are *customer* and *supplier*.

2.2 Organizational artifacts

We cannot argue about a complex phenomenon like an organization [Mo96] without goals, for which purpose we want to construct models about it. The main goals of organizational modeling in the context of information systems are

1. understanding (*analyzing*)
2. changing or creating (*constructing*)
3. maintaining.

We derive the to-be artifacts for our modeling from these sequentially pursued goals. The models are a basic means for communication ([WW02, 363], [BM06, 38]).

The first goal is *understanding* the organization's structure and the role of the structure's elements in organizational processes, that is the dynamic behavior of the structural units. *Behavior* in this view is acting and deciding of people and operating of automata ([KR74, 184ff.], [Mi79, 17ff.]). We call those two elements of an organization an *actor*. Based on understanding, the next goal is successfully *changing* parts of the structure and of functions or *creating* new ones. For both steps, we need models of behavior, based on models of the thing's structure. In order to get a better overview, in the understanding step it may be tolerable to disregard the states of things. However, in the constructive step this would be counterproductive.

But on which things do actors act? We call this kind of thing a *resource*. Resources are things, whose value is their ability to produce other resources of higher value than the original resources (*surplus value*). This happens mostly by combining some of them, e. g. work and material to a higher-valued material. This transformation process is called *production*. In order to administrate production, transportation and storing materials, *data is required*, which is created, updated or retrieved. We intentionally use the notion *administration* and not "management" for those routine actions. The 'machine' executing parts of administrative actions is called *information system*.

An enterprise viewed as a system needs four input-types of resources, which are also inside the system. These are *people*, *material*, *data* and *money*. At least one type of resource is the output of a business system, the *product*, but also the other resource types may appear as output, e. g. money as profit or skilled people. A product can be seen as a generalization of the input types, because specialized products are composed of material, of data (e. g. artifacts like software), of manpower and knowledge (people), called *services*, or of money e. g. as loans in the case of a bank.

As there are many production steps, executed by many types of human or material actors (*machines*), creating new things is not limited to material. The majority of data things inside an enterprise system is created for management and documentation purposes. This fact is important for our perspective, because many *actions* of actors are *transitions* (see §2.1) of data. Transitions in a chain are *processes*. As main subject of organizational modeling we regard processes containing the flow of the resources material, money and data as *flow types*.

The last goal of our organizational view is *maintaining* durable resources like machinery, buildings, data and software. There are also *consumable* resources that disappear

in a production process, but they are not part of modeling information systems. Models regarding the maintaining goal are *documentations* about resources, containing their relevant *history*.

2.3 A framework for organization models

Merging the BWW Ontology with our short outline of relevant artifacts we can summarize, that for process modeling purposes the ontology seems to be sufficient to the understanding phase, at least. Table 2 shows a synopsis, showing the hierarchy of system → thing in the first row. Searching in the literature for comparable models was not successful. The only promising paper by Kettinger et. al [Ke97] outlines a framework of *activities*, based on consulting practices, not on things and their interactions.

Table 2: Mapping BWW terms and organizational analysis

BWW ontology	organizational pendant
system	organization
subsystem	organizational unit
thing	resource
	human actor
	machine (material or artificial actor)
property	attribute of data type
state	value of attribute
transition	resource flow
law	business rule
history	sequence of attribute's values
event	result of an actor's action
environment	actors, each behaving as an indivisible thing

After matching the generalized world of BWW and the specialized one of organizations, we have to examine, which level of detail is necessary in each phase of §2.2. Modeling is required in the first two phases, while phase three requires documentation only. The documents are models of the preceding phase.

The understanding phase needs *little* detail, the constructive phase *much*. However, too much detail is destructive for an organizational analysis ([HC93, 129ff.], [La05, sect. 2.7]), because the focus in the analysis phase is communication and understanding of the relevant economic context, which is the deep structure of an enterprise's processes. In this phase a lot of people are involved in potential disruptive changes. Their ability to see the relevant points should not be diverted by details. Moreover, details are expensive to exhibit² and preserve existing processes [HC93, 129].

² Long before the "process reengineering hype", the first author managed an off-the-shelf project in the Schering AG, Berlin – Germany in 1986. Schering introduced a new payroll software accounting monthly 26,000 people employed. The new software replaced a 15 year old self-made one. In the analysis phase, there were 172 different processes detected and modeled graphically for understanding purposes. The

The proceeding part of the paper focuses on the EPC and the AD, because of their practical relevance today [GR00, 78], [Sö02, 602].

3 Evaluation of graphical process notations

In §1 we cited some evaluations, most of them executed on the pattern level. In this section we examine the remaining notations of table 1 with regard to suitability for

- organizational overview of processes (understanding phase)
- ability of refinement to a more detailed view (construction phase)

after some remarks about grammatical deficiencies of modeling languages.

3.1 Overview and grammatical deficiencies

Evaluation of notations can be referenced to the following *grammatical deficiencies*: Construct *overload*, *redundancy*, *excess* and *deficit* [WW02], being categories for a notation's quality of mapping against an ontology.

Green & Rosemann [GR00] valued the ARIS framework as to suffer from some construct deficiencies, which could be overcome by changing some patterns of the framework. Soffer et al. [So01] also referred to ARIS and judged the *Object Process Methodology* (OPM) to be superior to ARIS, which seems to be similar to UML. Burton-Jones & Meso valued the UML to be partially defective, too [BM06], but preferred ADs among other process notations.

All four process notations of table 1 have one important construct deficit: They do *not* offer a genuine system view. Presumably, the creators of the frameworks ARIS and UML did not *intend* to give such a view. Scheer explicitly argues an information system to be that complex, that it only can be modeled in different perspectives [Sc99]. A system view would be needed in business projects of realistic size and should be incorporated into a process view [GR00, 83]. We will pursue this aspect later, after comparing the two notations EPC and AD in behalf of their usage in the first two phases.

3.2 Understanding phase

This section compares the process descriptions EPC and AD with regard to their efficacy in this phase and presents a proposal for a system view with the UML. On principle, it is possible to model a process either with an EPC or with a AD. The two notations can be mapped one into the other [LA98].

The EPC models a process as an alternating sequence of functions and events, based on some rules like the necessity of one or more start- and end-events. The AD shows in its simplest form a control flow of actions as well as a start- and end-event, guarded by the condition to be fulfilled. This means, an EPC has about double the number of

to-be-concept only needed 8 automatic batch processes and 15 computer-supported dialog processes. The problem in the constructive phase was *not* a finer granularity of the diagrams, but a detailed modeling of *data*, *access rights* on them and more than 2,000 *test cases*.

nodes than the corresponding AD. For a given number of functions (= actions) f we get in a simple sequential flow a number of nodes N :

$$N_{AD} = f + 2 \quad \text{while} \quad N_{EPC} = 2f + 1.$$

In large projects, there has to be depicted, administrated and understood a very large number of functions/actions. As every function *must* produce at least one output, which is in most cases the state of a data thing, the event nodes inside the process in the EPC are a construct overload weighting heavily. The charts to be communicated grow to unpracticable size. This prevents efficient and effective communication. Our hypothesis on this property of the EPC is:

Overloaded graphics are not *understood*, they are *believed*.

DeMillo et. al observed this phenomenon evaluating the correctness of large software systems and called it the *trust-me approach*, having consequences possibly like the *Titanic event*³ [ML79]. Examples in the next chapter will show this.

In the case of the system the authors were unable to model a system view with the EPC. The reference in [GR00, 83] to [LA98] could not be resolved. The *extended EPC* (eEPC) still has a construct deficit for "thing" as the original EPC.

The same construct deficit of the UML can be overcome. There are several proposals:

- The creators of the UML propose the *package* for a system [Ru05, 635]. This seems to be a very software-specific artifact.
- Oestereich et. al [Oe04, 207] oppose to this solution with the argument, that a package has no behavior. A system as a 'meta-thing' must have a behavior. They depict an *organizational unit*, being an element of an enterprise system, as a stereotype of an object.
- Larman models a precise interaction between environment and system with *interaction diagrams*, the interacting things regarding as elements of the system [La05].
- Reinhartz-Berger finally proposes another solution in extending the functions of the *use case diagram* by a stereotype *collaboration* [Re05].

We follow Oestereich's argument and generalize his proposal. For communication purposes a system must show a border to its environment and contain things showing a behavior. Remember §2.1: Things are the origin for transitions/events. The classical thing with behavior in the UML is the *object* (rectangle), resp. the class (specialized rectangle).

Problematic should be valued 'thing' in the case of the *activity*, because it is only implicit. The superstructure specification of the UML (v2.1.2) is silent to this topic⁴. In the case of a *class*, it is clear, that *methods* are those parts of the thing *class*, that describe its behavior. But what thing "behaves" in an activity? We had identified *people* and *automata* in §2.2 as active things of a system. Other resources are passive, like

³ "Errors should not occur? Shades of the ship that shouldn't be sunk." [ML79, 277]

⁴ see [OMG08], Part II – Behavior, 11 Actions and 12 Activities

data, material and money. The actors *people* and *automata* are obviously implicit behind the depicted activity diagrams.

That's why we draw a system as an object, showing activities (or actions) as elements, that also can be regarded as refinable *subsystems*. Only actions show a behavior in this refinement process. At every level, the modeler can switch from the static to a dynamic model, showing a process. This level will be very high in real projects, because important processes do not follow static organizational borders. Figure 1 shows the system model of a production enterprise, in which material&work, money and data are depicted as stereotypes of directed associations. Actors in the environment are not depicted like in use cases, because they are active things too, but from a system view they will not be refined. The stereotype is drawn as a special object.

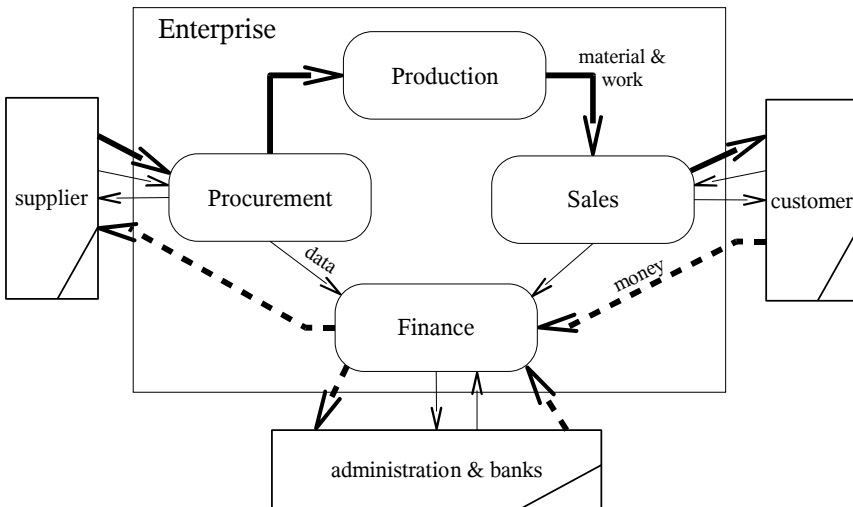


Figure 1: An enterprise as a system of different flow types

Figure 1 is not consistent to the actual UML 2.1 specification, because an object can not contain activities or actions. But our system view only serves communicative purposes. It is important to keep diagrams understandable without over-simplifying them. As every material and money flow is accompanied by data, parallel data flows are not drawn explicitly.

3.3 Construction Phase

As Soffer et. al have stated and the first author experienced in implementing several modules of SAP R/3⁸⁵, there is a lot of constructive detail in off-the-shelf implementations. In this phase, the guideline is no longer (perhaps intuitive) understanding, but a precise connection of the phase-1's to-be models to more detailed models. In this case UML seems to have fewer weaknesses than ARIS.

⁵ HR – human resource, FI – finance, AM – asset management, procurement of MM – material management.

As base for the further project, e. g. for customizing and finding access rights, precise data models have to be developed, connected to the data types, that are *created* in the processes. In order to detect responsibilities for data creation in the organization, the focus should lie on the routine business case data, not on the myriads of informative data. This point is due to later work, when the routine processes are clarified and their *testing* is designed. Moreover, an exact *specification* is necessary for data transferring programs, matching the old system's database to the new one. In the great majority of cases, this match is *not* 1 : 1.

Another area of detailed modeling are *decisions* in the processes, on which a project team has to decide how to automate them by the software's functions ("customizing") or to force human decisions in dialogues. In our case, we found a lot of such decision points, especially in deep refinement levels.

All this can be modeled either with ARIS or with UML. Both frameworks offer a view on data, but the connection between active things and passive (resources) differs considerably. In an AD, it is a common refinement, to expand a control flow for overview purposes to an object flow for further analysis and construction. The majority of passive things in an AD object flow are data types, e. g. a sales order or an invoice. As the EPC has no state concept for things, data has to be pasted on functions in the eEPC, which makes the process, already overloaded by events, even more overloaded. We now show some process views from our case study.

4 Activity Diagrams from a large SAP introduction

As mentioned in §1, our case study deals with the subproject *Sales* in a worldwide introduction of the SAP ERP-system. The following subprocesses of material and data flows had been the base for our empirical analysis:

1. Inquiry
2. Quotation
3. Order processing
4. Goods receipt
5. Material Picking
6. Production
7. Product staging
8. Shipping
9. Billing.

Figure 2 shows an overview process for the understanding phase. It is a to-be process, whose "actions" from the list above are all refined in separate subprocesses, some of them containing more subprocesses. In order to keep confidence we have simplified our examples in behalf of firm-specific details.

There is an interface to the subsystem *Finance*, which is an actor in the environment of the regarded system *Sales*, receiving the output of a data type *bill*. From the SAP system's view, the module FI (*finance*) is also outside the system containing our process, which is supported by the modules SD (*shipment an delivery*), parts of MM (*ma-*

terial management) and of PP (production planning). The corresponding EPC of the blueprint comprehends four printed pages.

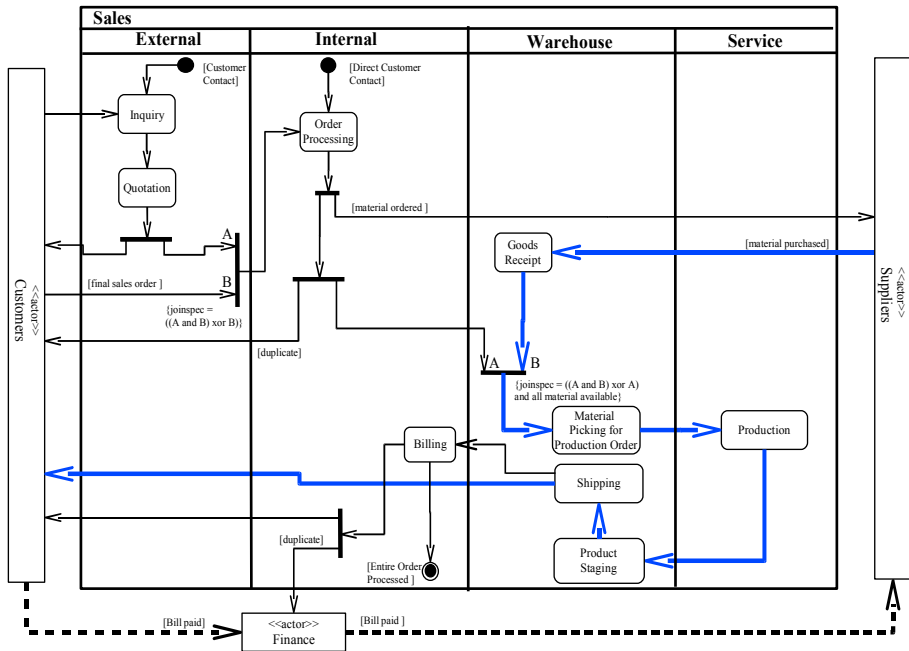


Figure 2: Sales department as a subsystem showing the sales process

The reader should look at the join of two flow types following the action *GoodsReceipt*. In the unification of the two flows, the data flow becomes implicit. This representation in an overview process seems to be well suited for communication, but relaxes a strong token concept.

The following figures 3 and 4 are refinements of the functions *Quotation* and *Billing*. Both processes show explicitly the data things, which to produce is the primary goal of the processes. It can be disputed whether an action, reading data only, causes an event, as there is no state changed. But doubtlessly data *creating* actions cause events.

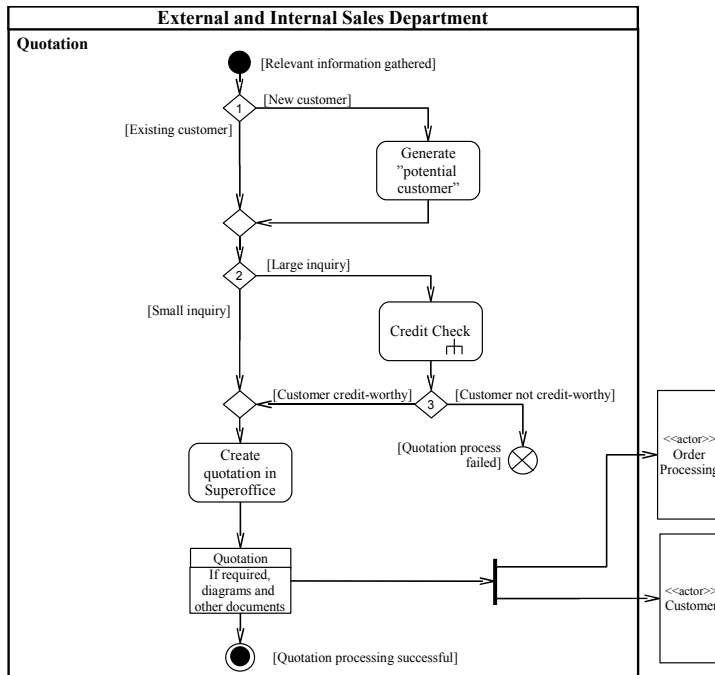


Figure 3: Process *Quotation* as a decision-dominated activity

The process *Quotation* shows the above mentioned dominance of decisions in some processes. We have numbered the three decisions in it, which probably are all subject to customizing for automatic processing. But before that, it has to be communicated and decided by the firm, how "large" should be measured (see the guard 'large inquiry' in figure 3).

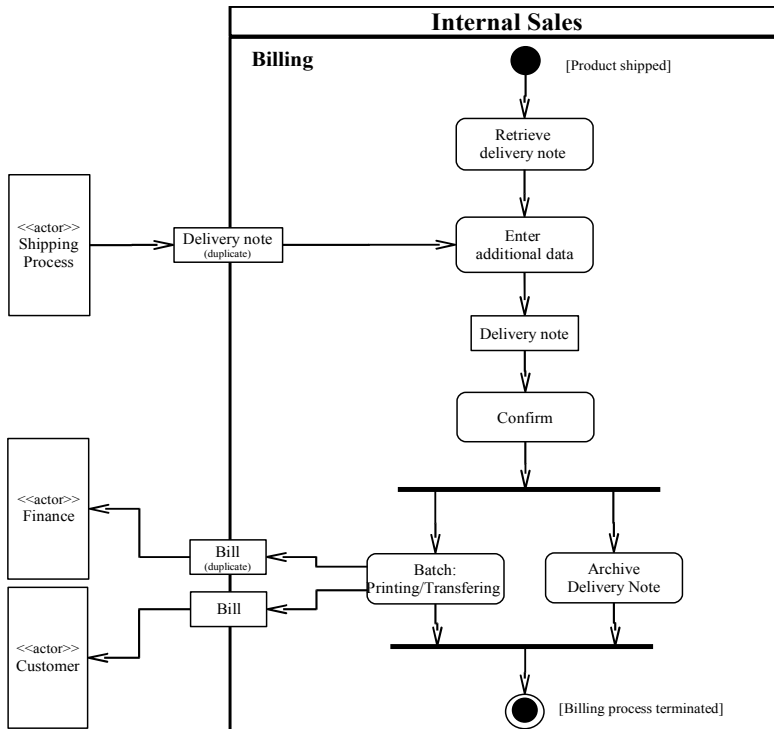
The process *Billing* (figure 4) outlines the intuitively understandable role of data objects in ADs. Two data types are created in the process: The *delivery note* changes its state – this must be explicitly shown [WW95, 212]. The *bill* is produced – this is the goal of the whole process.

The role of the diagrams in the communication processes inside the project team and with the top management was like this:

1. An overview diagram of the current state and a diagram similar to our figure 2 were discussed and decided on with the top management. Both diagrams were accepted as being well understandable.
2. Based on the to-be state validated in this way, the nine subprocesses (with two more sub-subprocesses) were depicted and successfully discussed.

This seemed to be a serious source for our hypothesis (see §1):

The AD is better suited for handling and communication than the EPC.

Figure 4: Process *Billing* as a data creating activity

5 Conclusion

Our case study has shown that ADs are well suited for the analysis phase of large projects. This happened and succeeded in an environment, dominated by EPC blueprints. Aside the probable superiority of the UML framework in the understanding phase it also seems to be more suited than ARIS in the further project steps. Results of activity diagrams can be applied and reused in the constructive phase. In this way redundant work and misunderstanding can be avoided.

Moreover, our extension of the UML grammar by a practicable system view seems to be a pragmatic proposal, not violating elementary UML constructs.

Further work should be done on empirical tests of EPC vs AD. This could probably only be laboratory experiments with students, because we cannot imagine any firm, conducting process modeling in a real world project twice.

References

- [Aa99] van der Aalst, W. M. P.: Formalization and Verification of Event-Driven Process Chains. *Information & Software Technology* 41(1999), 639-650.
- [Aa03] van der Aalst, W. M. P.; Hofstede, A, H. M.; Kiepuszewski, B.; Barros, A. P.: Workflow Patterns. *Distributed and Parallel Databases*, 14(2003) 1, 5-51.
- [ADO00] van der Aalst, W. M. P.; Desel, J.; Oberweis, A. (eds.): *Business Process Management: Models, Techniques and Empirical Studies*. LNCS 1806, Springer, Berlin et al., 2000.
- [AH00] van der Aalst, W. M. P.; Hofstede, A, H. M.: Verification of Workflow Task Structures: A Petri-Net-Based Approach. *Information Systems*, 25(2000) 1, 43-69.
- [BB03] Basu, A.; Blanning, R. W.: Synthesis and Decomposition of Processes in Organizations. *Information Systems Research* 14(2003) 4, 337-355.
- [BM06] Burton-Jones, A.; Meso, P. N.: Conceptualizing Systems for Understanding: An Empirical Test of Decomposition Principles in Object-Oriented Analysis. *Information Systems Research* 17(2006) 1, 38-60.
- [BP03] Business Process Management Initiative (BPMI): *Business Process Modeling Notation (BPMN) – Working Draft (1.0)*, August 2003. www.bpmi.org
- [CL99] Curran, T. A.; Ladd, A.: *SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management*. 2nd ed. Prentice-Hall, Englewood Cliffs, NY, 1999.
- [Da93] Davenport, T. H.: *Process Innovation – Reengineering Work through Information Technology*, Boston, MA: Harvard Business School, 1993.
- [De05] Delcambre, L.; Kop, C.; Mayr, H. C.; Mylopoulos, J.; Pastor, O. (eds): *Conceptual Modeling – ER 2005*. LNCS 3716, Springer, Berlin et al., 2005.
- [GR00] Green, P; Rosemann, M.: Integrated Process Modeling: An Ontological Evaluation. *Information Systems* 25(2000) 2, 73-87.
- [HC93] Hammer, M.; Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*, New York, NY, Harper Collins, 1993.
- [Ke97] Kettinger, W. J.; Teng, J. T. C.; Guha, S.: Business Process Change: a Study of Methodologies, Techniques, and Tools. *MISQ*, March 1997, 55-80.
- [KR74] Kast, F. E.; Rosenzweig, J. E.: *Organization and Management*, 2nd ed. McGraw Hill, Tokyo et al. 1974.
- [LA98] Loos, P.; Allweyer, T.: Process Orientation and Object Orientation – An Approach for Integrating UML and Event-Driven Process Chains (EPC). *Enterprise Distributed Object Computing Workshop, EDOC '98, Proceedings*, La Jolla /CA, 1998, 102-112.
- [La05] Larmann, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd ed. Prentice Hall, 2005.
- [Ma99] Malone, T.; Crowston, W.; Lee, J.; Pentland, B.; Dellarocas, C.; Wyner, G.; Quimby, J.; Osborn, C. S.; Bernstein, A.; Herman, G.; Klein, M.; O'Donnell, E.: *Tools for Inventing Organizations: Towards a Handbook of Organizational Processes*. *Management Science* 45(1999) 3, 425-443.
- [Mi79] Mintzberg, H.: *The Structuring of Organizations*. Prentice Hall, Englewood Cliffs /NY, London et.al 1979.
- [ML79] De Millo, R. A.; Lipton, R. J.; Perlis, A. J.: Social Processes and the Proof of Theorems of Programs. *CACM*, 22(1979) 5, 271-280.
- [Mo96] Morgan, G.: *Images of Organization*, 2nd ed. Sage, Thousand Oaks /CA, 1996.

- [Oe04] Oestereich, B.; Weiss, C.; Schröder, C.; Weilkiens, T.; Lenhard, A.: Objekt-Oriented Business Process Modeling with the UML (in German), dpunkt, Heidelberg 2003.
- [OMG08] Object Management Group: Unified modeling Language (UML) – Superstructure Specification 2.1.2. <http://www.omg.org/technology/documents/formal/uml.htm>; 2008-Jan-10.
- [Re05] Reinhartz-Berger, I.: Conceptual Modeling of Structure and Behavior with UML – The Top Level Object-Oriented Framework (TLOOF) Approach, in: [De05], 1-15.
- [Ru05] Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual, 2nd ed. Addison-Wesley, Boston et al. 2005.
- [Ru06] Russell, N.; van der Aalst, W. M. P.; Hofstede, A, H. M.; Wohed, P.: On the Suitability of UML 2.0 Activity Diagrams for Business Process modeling. Third Asia-Pacific Conference on Conceptual modeling (APCCM2006), Hobart, Australia. Vol 53, Stumptner, M.; Hartmann, S.; Kiyoki, Y. (eds.), pp. 95-104.
- [Sc94] Scheer, A. W.: Business Process Engineering – Reference Models for Industrial Enterprises. Springer, Berlin et al. 1994.
- [Sc99] Scheer, A. W.: ARIS – Business Process Frameworks, 3rd ed. Springer, Berlin et al. 1999.
- [Sö02] Söderström, E.; Andersson, B.; Johannesson, P.; Perjons, E.; Wangler, B.: Towards a Framework for Comparing Process modeling Languages. in: Pidduck, A. B. et al (eds.): CAISE 2002, LNCS 2348, 600-611.
- [So01] Soffer, P.; Golany, B.; Dori, D.; Wand, Y.: Modeling Off-the-Shelf Information Systems Requirements: An Ontological Approach. Requirements Engineering 6(2001), 183-199.
- [SW05] Soffer, P.; Wand, Y.: On the Notion of Soft-Goals in Business Process Modeling. Business Process Management Journal 11(2005 6), 663-679.
- [Wa96] Wand, Y.: Ontology as a Foundation for Meta-modeling and Method Engineering. Information & Software Technology, 38(1996), 281-287.
- [Wh04] White, S. A.: Process Modeling Notations and Workflow Patterns.
- [Wo05] Wohed, P.; van der Aalst, W. M. P.; Dumas, M.; Hofstede, A, H. M.; Russell, N.: Pattern-Based Analysis of UML Activity Diagrams, in: [De05], 63-78.
- [WW90] Wand, Y.; Weber, R.: An Ontological Model of an Information System. IEEE Trans. on Software Engineering, 16(1990) 11, 1282-1292.
- [WW95] Wand, Y.; Weber, R.: On the Deep Structure of Information Systems. Information Systems Journal, 5(1995), 185-202.
- [WW02] Wand, Y.; Weber, R.: Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda. Information Systems Research 13(2002) 4, 363-376.
- [Wy04] Wyssusek, B.: Ontology and Ontologies in Information Systems Analysis and Design: A Critique. Proceedings of the Tenth Americas Conf. on Information Systems, New York, August 2004, 4303-4308.