A Conceptual Framework for Analysing Enterprise Engineering Methodologies

Antonia Albani[*,a], David Raber[a], Robert Winter[a]

[a] Institute of Information Management, University of St. Gallen, Unterer Graben 21, CH-9000 St. Gallen, Switzerland

Abstract. *For the design and engineering of enterprises, several methodologies are available that successfully address certain aspects of design problems in enterprises or certain domains. In real-world design problems it is essential to choose the right means to reach the desired ends. Often it is not apparent which methodology is best chosen in order to reach desired ends. Additionally, real-world design problems often require several such methodologies to be combined because multiple aspects have to be covered and/or the problem combines characteristics of several domains. In order to allow for a systematical understanding and comparison of methodologies and for a facilitation of their composition (if necessary), we propose a general conceptual framework. The framework allows analysing the essential concepts and constituent parts of enterprise engineering methodologies. The resulting analysis supports decisions making concerning which methodology or which combinations of methodologies to apply to the given design problem. To demonstrate its usefulness, we first analyse the concepts and building blocks of two design and engineering methodologies on that basis. Second, we show how these two methodologies, which are based on very similar concepts—as resulted from the analysis by applying the conceptual framework—can be combined in order to derive at a complete solution for a given design problem.*

Keywords. Enterprise Engineering • Conceptual Framework • Design Methodologies

Communicated by Manfred Reichert. Received 2013-01-07. Accepted after 2 revisions on 2015-03-09.

1 Introduction

Enterprise Engineering is a discipline that studies enterprises from an engineering perspective (Albani et al. 2011). This means that enterprises are considered to be purposefully designed and implemented systems. Such systems can be re-designed and re-implemented if there is a need for change. The problems that need to be solved in such a context while addressing the enterprise in a holistic way are of complex nature and multifaceted. Since most enterprise solutions will include not only task and people components, but also information technology (IT) components like software systems and IT infrastructure components, Enterprise Engineering and Information Systems Engineering have different foci, but many commonalities. While the focus in Information Systems Engineering is set on the *content* of information and communication (see Fig. 1), the *intention* of collaboration and cooperation between social individuals, by means of entering into and complying with commitments, is set on top of the content in Enterprise Engineering. Solutions of complex enterprise problems concern both aspects and need to be constructed in a systematic way. Here is where the emerging discipline of Enterprise Engineering is considered to be a suitable vehicle. It deals adequately with the challenges that modern enterprises face nowadays (Dietz et al. 2013).

As stated in Albani and Dietz (2010), '[t]he mission of the discipline of Enterprise Engineering is

* Corresponding author.
E-mail. antonia.albani@unisg.ch
This project is partly supported by the Swiss National Science Foundation (SNSF) and the Swiss Commission for Technology and Innovation (CTI).

to combine (relevant parts from) the organisational sciences and the Information Systems sciences, and to develop theories and methodologies for the analysis, design, and implementation of future enterprises'. Fig. 1 illustrates the influences of Information Systems Sciences and Organisation Sciences on the discipline of Enterprise Engineering.

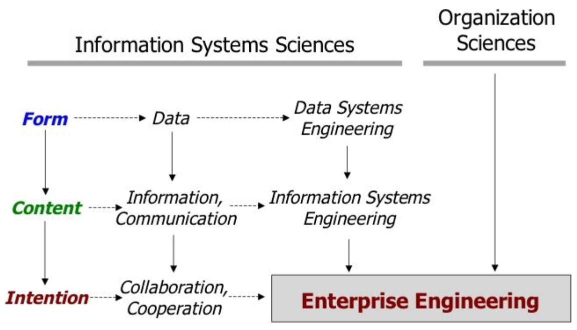

Information Systems Sciences
Organization Sciences
Form
Data
Data Systems Engineering
Content
Information, Communication
Information Systems Engineering
Intention
Collaboration, Cooperation
Enterprise Engineering

Figure 1: Enterprise Engineering (Albani and Dietz 2010)

Albani and Dietz (2010) explain the relationships in Fig. 1 as follows: 'As the content of communication was put on top of its form in the 1970's, the intention of communication is now put on top of its content. It explains and clarifies the organisational notions collaboration and cooperation, as well as authority and responsibility. It also puts organisations definitely in the category of social systems, very distinct from Information Systems.'

Since Enterprise Engineering considers enterprises as designed and engineered complex systems, and because there are many similarities with Information Systems design and engineering, the Design Science Research (DSR) approach for Information Systems can provide a basis to solve Enterprise Engineering problems. Dietz et al. (2013) state in the journal article 'The discipline of Enterprise Engineering' that DSR is already quite widely accepted, notably in the Information Systems area cf. (Hevner 2007; Hevner et al. 2004; March and Smith 1995), but builds also the scientific foundation for justifying research in the discipline of Enterprise Engineering.

Several methodologies[1] are available and have been applied in practice to aid enterprise design and engineering. Examples of such methodologies, among many others, are Business Engineering (Österle and Winter 2003), Design and Engineering Methodology for Organisations (DEMO) (Dietz 2006), Business Process (Re)Engineering (Davenport and Short 1990; Hammer and Champy 1993), PICTURE (Becker et al. 2007) or ARIS (Scheer and Schneider 2005). Such methodologies successfully address certain aspects of enterprise problems, e. g. modelling the essential business transactions in order to decide about splitting or allying of enterprises (by applying DEMO), modelling the business processes (by applying Business Process (Re)Engineering or PICTURE), or integrating business aspects as well as IT aspects (by applying Business Engineering). They aim at building and testing various kinds of designed artefacts (e. g. software systems, procedures/project plans, but also reference models or reusable methods) as solutions to certain enterprise design problems or even better to classes of similar enterprise design problems.

Since design and engineering methodologies are often widely varying with regard to foundations, goals, and processes, Hevner (2007) introduces a general design research framework comprising three cycles—relevance, design and rigor cycles, which should be present and clearly identifiable in every piece of design science research (see Fig. 2.).

In the *relevance cycle* the requirements of the application domain are defined and introduced

[1] The terms 'method' and 'methodology' may cause some confusion since the usage of the terms may differ. There are also differences in usage between North America and Europe. In this article we use the term 'methodology' as it is referred in Mingers (2001) by 'a methodology'. He states, '[. . .] *a methodology* is more general and less prescriptive than a method. It is a structured set of guidelines or activities to assist in generating valid and reliable research results. It will often consist of various methods or techniques, not all of which need be used every time. It can be difficult to precisely delineate the boundaries between method and methodology at one end [. . .], or between methodology and a general research approach [. . .] at the other.'

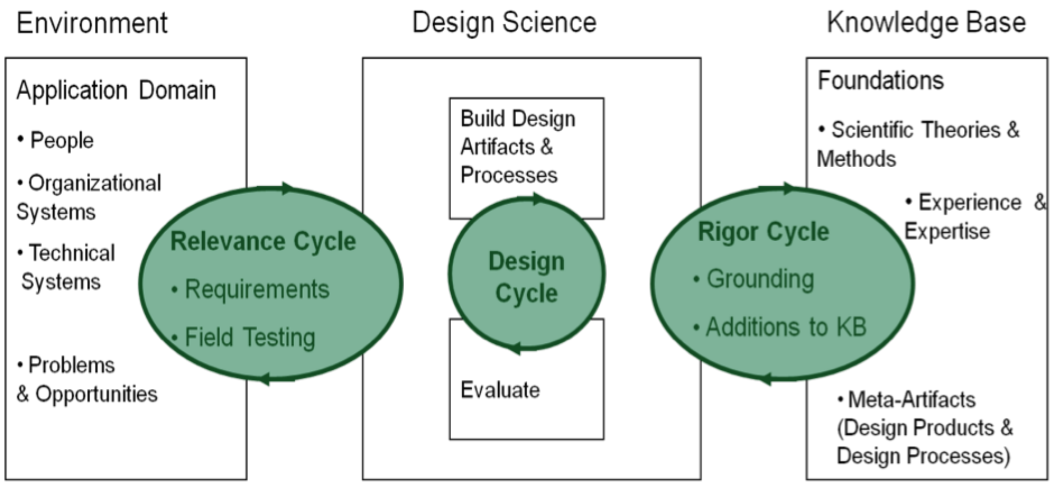


Figure 2: Design Science Research Cycles (Hevner 2007)

into the design process. Additionally, the resulting research artefacts are established in the environment (e. g. by field-testing) in order to demonstrate their problem solving utility. In the *rigor cycle* domain experience and expertise as well as existing generic design products and processes are introduced into the design process in addition to scientific theories and methods. Additionally, new generalisable knowledge derived from the design process is added to the knowledge base for reuse. The *design cycle*, which is essentially a solution search procedure, iterates between the core activities of building and evaluating design artefacts. '[. . .] multiple iterations of the design cycle [. . .][are necessary] before contributions are output into the relevance cycle and the rigor cycle.' (Hevner 2007).

Since Hevner's framework, as introduced above, is generally applicable for any constructional problem and result type in design science research, it might be a starting point to frame specific Enterprise Engineering research proposals.

In order to identify respective goals and context of a design problem one needs to choose the right means to reach the desired ends. Often it is not apparent which methodology is best chosen in order to reach desired ends. Additionally, due to the complex and holistic nature of enterprise design and engineering problems often one methodology alone does not give full support to reach a desired end. Real-world design and engineering of organisations often requires several methodologies to be combined because multiple aspects have to be covered and/or the problem combines characteristics of several domains. This fact is also addressed by Mingers (2001) in his article *Combining IS Research Methods: Towards a Pluralist Methodology*. He motivates the need for a multimethod approach as follows: 'Different paradigms each focus attention on different aspects of the situation, and so multimethod research is necessary to deal effectively with the full richness of the real world.'

However, in order to decide about which methodology to choose or which methodologies to combine their essence and their constituents needs to be understood well. As also stated by Hevner, '[. . .] practical utility alone does not define good design science research. It is the synergy between relevance and rigor and the contributions along both the relevance cycle and the rigor cycle that define good design science research' (Hevner 2007). The methodology chosen, or the ones to be combined need not only to be of practical relevance, but also theoretically sound.

Since enterprise engineering methodologies are quite different in nature, there is a need for a conceptual basis allowing to systematically *analyse* them. A clear understanding of the methodologies allows for choosing the right methodology or combination of methodologies to reach desired ends. This paper aims to contribute to such conceptual basis, i. e. to propose a *conceptual framework for analysing Enterprise Engineering methodologies*.

After having discussed background work in Sec. 2, the conceptual framework is proposed in Sec. 3. It is based on Chmielewicz's (1994) conceptualisation of research in social sciences, the design science research artefact types as proposed by March and Smith (1995) and the current results of the on going discussions on theories—explanatory and design theories—(Baskerville and Pries-Heje 2010; Gregor 2006; Gregor and Jones 2007). In order to evaluate the proposal's utility, we use it to analyse two exemplary methodologies: *Design and Engineering Methodology for Organisations (DEMO)* (Dietz 2006) in Sec. 4; and *PICTURE* (Becker et al. 2007) in Sec. 5. Section 6 elaborates on the advantages of a possible composition of both methodologies based on the resulting analysis of the methodologies with the proposed framework. Since the proposed framework has the potential to be used for many other design and engineering methodologies, section 7 discusses contributions as well as limitations and presents an outlook on further research.

2 Background Work

Hevner et al. (2004) state that much of the work performed by Information Systems practitioners deals with design. Common design artefacts produced by design science researchers are *constructs, models, methods* and *instantiations* (March and Smith 1995). Hevner et al. describe these artefacts as follow: '*Constructs* provide the language in which problems and solutions are defined and communicated [. . .]. *Models* use constructs to represent a real world situation—the design problem and its solution space [. . .]. *Methods* define processes. They provide guidance on how to solve problems, that is, how to search the solution space. [. . .] *Instantiations* show that constructs, models, or methods can be implemented in a working system' (Hevner et al. 2004).

As stated by Winter (2008) it is important to understand the artefact types of design science research not as separate concepts, but as an interdependent system. Winter (2008) refers to Chmielewicz's (1994) conceptualisation of research in social sciences, which may serve as a foundation to explain such dependencies. Chmielewicz differentiates between ontological facts, theoretical statements, technological statements, and normative statements. The respective artefact types are foundational concepts, cause-effect relations, means-ends relations, and object-value relations (judgments). To illustrate this taxonomy, the pyramid metaphor seems appropriate: Applicable ontology and meta models constitute the foundation for theoretical statements. Valid theories should constitute the foundation for effective technology, i. e., statements about which means *can* be used to achieve certain goals. Finally, normative statements represent judgments, i. e., statements about which techniques *should* be used to achieve certain goals, and which goals *should* be pursued.

The relationships between the design science research artefact types and Chmielewicz's conceptualisation as proposed by Winter (2008) are illustrated in Fig. 3 in the two left columns.

As stated by Winter (2008),

- foundational concepts correspond to constructs,
- means-end relations correspond to models and methods, and
- concrete choices correspond to instantiations.

This analogy can also be extended to the disputed role of theories as design research artefacts: Opposed to both March and Smith (1995) and Hevner et al. (2004), who do not consider theories as key artefacts of design science research, several design science research authors clearly state that theories need to be considered (Goldkuhl 2004; Venable 2006a,b; Walls et al. 1992).

While these authors obviously refer to Kerlinger's (1964) concept of theory that emphasises examining and predicting a phenomenon, more recently the concept of *design theories* has attracted

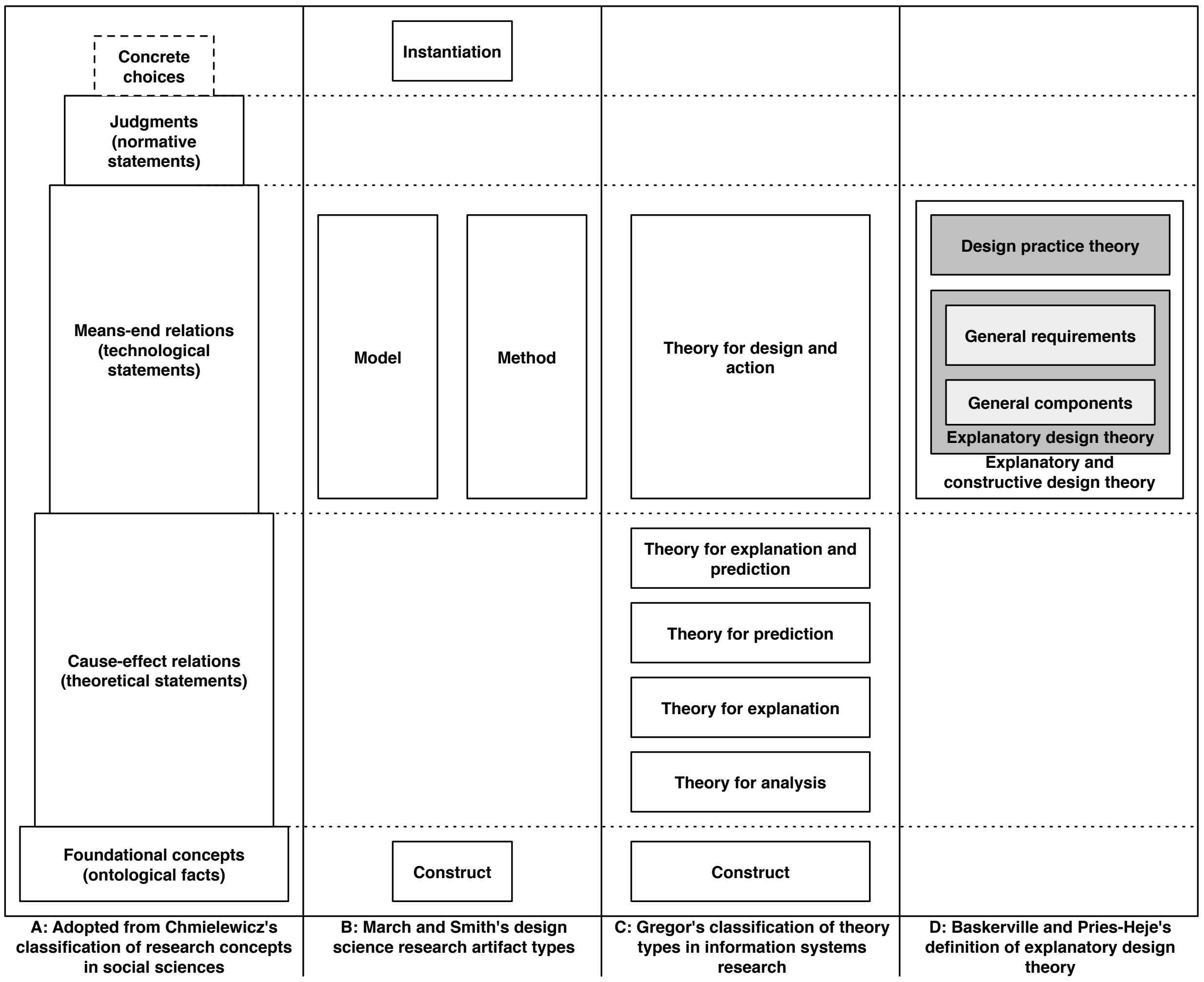

Concrete choices
Instantiation
Judgments (normative statements)
Design practice theory
Means-end relations (technological statements)
Model
Method
Theory for design and action
General requirements
General components
Explanatory design theory
Explanatory and constructive design theory
Theory for explanation and prediction
Theory for prediction
Cause-effect relations (theoretical statements)
Theory for explanation
Theory for analysis
Foundational concepts (ontological facts)
Construct
Construct
A: Adopted from Chmielewicz's classification of research concepts in social sciences
B: March and Smith's design science research artifact types
C: Gregor's classification of theory types in information systems research
D: Baskerville and Pries-Heje's definition of explanatory design theory

Figure 3: Classification of Research Concepts in Social Sciences Versus Design Science Research Artefact Types

a lot of attention (Gehlert et al. 2009; Gregor 2006; Gregor and Jones 2007; Iivari 2007; Venable 2006a). A classification of contributions related to design science design theories can be found in (Fischer et al. 2010).

However there is still an on going discussion regarding the term *design theory*. Gregor (2002, 2006) elaborates on the term *theory* and distinguishes five different types of theory important for the discipline of Information Systems: *Theory for analysis, theory for explanation, theory for prediction, theory for explanation and prediction* and *theory for design and action* (see Fig. 3, column C). While the first four types of theory are in line with Kerlinger's traditional theory understanding in the social sciences, theories for design and action are different. The distinctive feature of design theory, according to Gregor (2006), is that it gives explicit prescriptions (e. g. methods, techniques, principles of form and function) for constructing an artefact. Based on this specific feature, design theories can be understood as means-end relations as shown in Fig. 3, as opposed to the traditional understanding of theory as representing cause-effect relations. Gregor (2006) further states that any type of theory must be represented physically in some way (e. g. in words, mathematical terms, symbolic logic, diagrams) and that all of the primary constructs in the theory should be well defined (see construct element in column C in Fig. 3). Even if there is quite some literature dealing with the topic of design theory in Information Systems, there is no complete agreement about the characteristics and components of design theories (Baskerville and Pries-Heje 2010). As early as 1992, Walls et al. (1992) introduced the components of an Information Systems design theory and proposed a design theory for Vigilant Executive Information Systems. Among others, Gregor and Jones (2007) extended the work of Walls et al. by introducing additional structural components of a design theory. Analysing several methodologies to design theory, Baskerville and Pries-Heje (2010) identified a number of shared assumptions among them and discussed the problems and issues in the delineation of design theory. Based on these insights they proposed to partition design theory into an explanatory and a practice part, designated as *explanatory design theory* and *design practice theory*, respectively. Both parts together constitute an *explanatory and constructive design theory.* Looking at design theories from an explanatory point of view, they further identified two key elements being common in many works in design theory: *requirements* and *components* with their embodied relationships that explain the solution. The requirements specify the reasons for components; respectively components are justified by requirements. This explanatory design theory as introduced by Baskerville and Pries-Heje is '. . . a general design solution to a class of problems that relates a set of general components to a set of general requirements' (Baskerville and Pries-Heje 2010). In column D of Fig. 3 the methodology of Baskerville and Pries-Heje is put in relation with the other methodologies mentioned above. The *explanatory design theory* provides functional explanations of artefacts. This means explaining *why* a solution has certain components in terms of the requirements stated in the design. The *design practice theory* gives explicit prescriptions on *how* to design and develop an artefact by means of models and methods. Even if the design theory as proposed by Baskerville and Pries-Heje (2010) has an explanatory part, this part can not be understood as an explanatory theory in Kerlinger's understanding. However the design practice theory can be understood in the same way as the theory for design and action as introduced by Gregor (2006). Both parts, the explanatory design theory as well as the design practice theory, are concerned with a means-end relation: while the former is explaining that very relationship, the latter provides guidelines on how a certain means reaches a desired end. Integrating these components, the explanatory and constructive design theory of Baskerville and Pries-Heje (2010) is related to means-end relations as introduced by Chmielewicz (1994).

Having illustrated the on going discussions on the relevant artefact types (e. g. design theories) and on the understanding of design theory in DSR, the need for a conceptual bases by means of a framework in Enterprise Engineering research becomes evident.

Several frameworks for analysing or validating methodologies exist in literature. Worth mentioning are the framework of Hackathorn and Karimi (1998) for comparing Information Engineering Methods and the dynamic framework of Iivari et al. (2001) for classifying Information Systems methodologies and approaches. They both aim at contributing to the same goal as the framework presented in this paper. Namely better understand the core of the different methodologies. However, the domains of investigation are different. While Hackathorn and Karimi (1998) focus on current methodologies and tools for applying information engineering to construct the Information Systems architecture for an organisations, Iivari et al. (2001) focus on methodologies and approaches for developing Information Systems. The framework presented in this paper however is aiming at a better understanding of Enterprise Engineering methodologies, where Information Systems development is just one part of it. While the domain focus differs in the different frameworks, also the characteristics, which are analysed, are of different nature. Hackathorn and Karimi (1998) analyse the methodologies by means of two dimensions, namely the goals dimension (strategic, tactical and operational) and the conceptual-to-practical dimension (conceptual foundations and practical results). Based on these two dimensions Hackathorn and Karimi (1998) suggest roles for planners and developers and two processes that assure that organisational goals and Information Systems architecture are compatible. Iivari et al. (2001) analyse the methodologies by a set of foundational features that are shared by subsets of methodologies and approaches. The framework is organised as an inheritance structure in which each Information Systems development approach inherits the paradigmatic assumptions of the paradigm it represents (e. g. object-oriented). This results in a four-tiered framework with the tiers being paradigms, approaches, methodologies, and techniques.

Different from the frameworks just introduced, the conceptual framework presented in this paper provides a structure to analyse and therefore better understand the concepts and constituent parts of Enterprise Engineering methodologies by focusing on the theoretical grounding of the methodologies and on the means provided to reach desired ends. In addition to gaining an understanding if a certain Enterprise Engineering methodology is based on explanatory and/or predictive theories and if so, on which theories it is based, it also clarifies why a methodology has certain components in terms of the requirements stated in the design. Further, it clarifies how an artefact is designed and developed by means of the given models and methods. The aim of the framework is not to classify the methodologies into groups with similar focus (as is the case in Hackathorn and Karimi (1998) and Iivari et al. (2001)), or to analyse the quality of the methodologies by means, e. g. of its effectiveness and adoption in practice. Example frameworks for analysing the quality of methodologies are *A conceptual modelling quality framework* by Nelson et al. (2012), *The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods* by Moody (2003) or *Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework* by Moody et al. (2002). Further, quality frameworks have also been developed in the area of modelling languages, where e. g. Frank (1998) introduced a framework for evaluating the quality of modelling languages, arguing that ‘evaluating and comparing modelling languages is a prerequisite for progress in the field of conceptual modeling’ (Frank 1998).

3 Conceptual Framework

As we have seen in section 2, several artefact types have been proposed, however, even if these artefacts are strongly related among each other, the relationships have not been made explicit so far. In order to better understand existing design and engineering methodologies, a conceptual framework will help putting the relevant artefacts into relationship. The framework we propose in this research is illustrated in Fig. 4.

The conceptual framework is composed of five major artefact types: (1) concepts and constructs, (2) explanatory and/or predictive theories, (3) explanatory and constructive design theories, (4) models and methods and (5) concrete design solutions.

The basic notions that are needed in a research methodology are brought together by means of *concepts* and *constructs*. Dietz’s (2006, pp. 35–38) illustration of concept, object, sign and their relationships, which is based on the meaning triangle from semiotics, helps us to explain the notions of concepts and constructs as used in the conceptual framework. ‘A *concept* is a subjective individual thing. It is a thought or mental picture of an object that a subject may have in his or her mind’ (Dietz 2006, p. 37). ‘An *object* is an observable identifiable individual thing’ (Dietz 2006, p. 37). ‘A *sign* is an object that is used as a representation of something else. A well-known class of signs is the symbolic signs, as used in all natural languages’ (Dietz 2006, pp. 36–37). The relationships in the meaning triangle are as follow: A concept *refers* to an object; a sign *denotes* an object; and a sign *designates* a concept. The notion of *construct* as used in the conceptual framework seem to correspond with the notion of sign as defined by Dietz, since ‘constructs provide the language

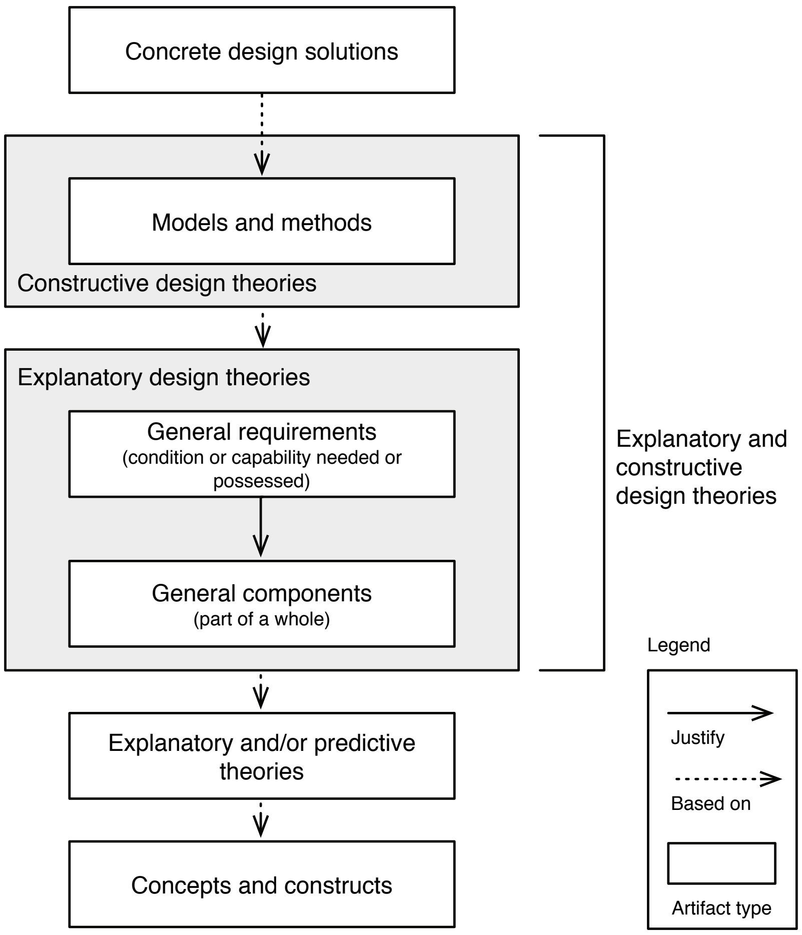

Concrete design solutions
Models and methods
Constructive design theories
Explanatory design theories
General requirements
(condition or capability needed or possessed)
General components
(part of a whole)
Explanatory and constructive design theories
Explanatory and/or predictive theories
Concepts and constructs
Legend
Justify
Based on
Artifact type

Figure 4: Conceptual Framework for Analysing the Essential Concepts and Building Blocks of Enterprise Engineering Methodologies

in which problems and solutions are defined and communicated' (Hevner et al. 2004). Concepts and constructs constitute the foundation of the framework on which the other artefacts are based.

As mentioned by Kerlinger, '[. . .] the basic aim of science is to find general explanations of natural events. Such general explanations are called theories.' (Kerlinger 1964, p. 10). He defines theory as follows: 'A theory is a set of interrelated constructs (concepts), definitions, and propositions that presents a systematic view of phenomena by specifying relations among variables, with the purpose of explaining and predicting the phenomena' (Kerlinger 1964, p. 11). This is what we call the traditional view of theory. But since the two goals, explaining and predicting, are not always achieved together, we follow the recommendation of Gregor (2006) and introduce the artefact type *explanatory and/or predictive theories* in our conceptual framework in Fig. 4. It integrates three types of theories, namely theory for explanation, theory for prediction, theory for explanation and prediction (shown in Fig. 3 column C). This artefact type reflects our understanding of *kernel theories* as introduced by Walls et al. (1992). It may provide the theoretical grounding for design theories.

We agree with Baskerville and Pries-Heje (2010) that a design theory has an explanatory and a practical (constructive) part and that the explanatory part consists of *general requirements*, *general components* and their interrelationships. In the constructional framework in Fig. 4, the *explanatory design theory* and the *constructive theory* are introduced in accordance with the definitions of design theory proposed by Baskerville and Pries-Heje (2010). Together they constitute the *explanatory and constructive design theory* (Baskerville and Pries-Heje 2010).

While the explanatory design theory explains *why* a component is being constructed into an artefact, the constructive design theory explains *how* an artefact needs to be constructed. The *how* is achieved by means of *models* and *methods* (see Fig. 4). However, not every model and/or method constitutes a constructive design theory. Only models and methods which are founded in theory, either in the explanatory and/or predictive theory or in the explanatory design theory, and which are sufficiently generalised to solve a class of design problems rather than a singular design problem, can be considered as constructive design theories. Models and methods which do not constitute a constructive design theory are e. g. the ones provided by TOGAF (The Open Group 2009) or ITIL (Office of Government Commerce 2005). They are based on defined concepts and constructs, but not explicitly and directly grounded on a theoretical foundation. An example methodology providing models and methods, which clearly constitute a constructive design theory, is DEMO (Dietz 2006). DEMO is based on an explanatory design theory (called 'Performance in Social Interaction' or PSI for short). The explanatory design theory is based on explanatory and/or predictive theories as e. g. Systemic Ontology (Bunge 1979), just to mention one of them. The next section will explain this grounding of DEMO in more detail.

The last artefact type in the constructional framework is *concrete design solutions,* shown on top of Fig. 4. Concrete design solutions are actually instantiations of those models and methods that have been applied for solution construction. They correspond to the instantiations as introduced by March and Smith (1995). If based on constructional design theory, concrete design solutions are theoretically sound. That means they are based either on an explanatory design theory, and/or on an explanatory and/or predictive theory. Such solutions do not only satisfy the relevance, but also the rigor aspect of proper DSR. The judgments, or normative statements as introduced by Chmielewicz (1994), are not part of the constructional framework. The reason is that decisions about adequate means and right goals are not generalisable in a way that they could be part of the framework and such decisions should be taken when *applying* the framework.

The conceptual framework as proposed above contributes to the analysis of existing design and engineering methodologies for organisational systems. It is not intended for a specific Information Systems 'school', as e. g. design theory opponents, kernel theory pragmatists, or kernel theory fundamentalists in DSR (as categorised in Fischer et al. (2010)). It rather allows for a better understanding of the essential concepts and building blocks of the methodologies of different schools contributing to a better selection of an appropriate methodology and, if necessary, a suitable and feasible composition of different methodologies in order to reach the desired end.

4 Design and Engineering Methodology for Organisations

DEMO (Dietz 2006) is a methodology in the area of Enterprise Engineering that allows to reveal the *essence* of an enterprise by means of supporting the construction of conceptual enterprise models. By the essence is understood that the models completely abstract from all realisation and implementation issues as e. g. in the models it is not yet defined if an actor role is realised by means of a human being or by means of IT; in the models we find only actor roles and not any assignment of a specific human to a defined actor role—this are all realisation and implementation choices, which are not part of the DEMO models. A complete overview of the DEMO methodology and its underlying concepts is available in the book of Dietz (2006).

DEMO has three important capabilities, which allow for designing the essence of an enterprise.

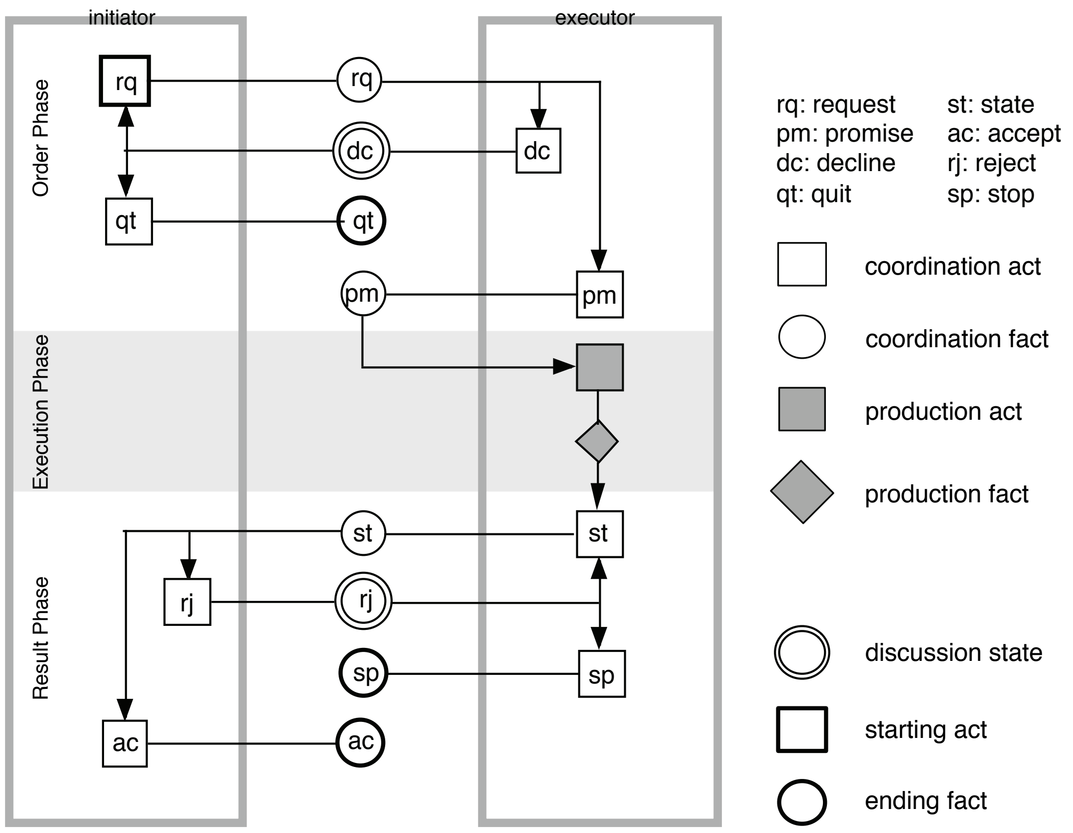


Figure 5: Standard Transaction Pattern (Dietz 2006)

First, in order to disentangle the essential knowledge of the operation and construction of an enterprise, Dietz (2006, pp. 89–98) introduces among others the concept of *transaction patterns*. All *essential* acts in an enterprise—coordination acts as well as production acts—are instantiations of one universal pattern, called a *transaction* (see Fig. 5).

Such transactions always involve two actor roles, the initiator and the executor, and are aimed at achieving a particular result. Actor roles are the essential unit of authority and responsibility in an enterprise and they perform two kinds of acts, *production* acts and *coordination* acts. The *initiator* starts and completes a transaction, and the *executor* performs the production act. In the order phase the initiator and the executor try to reach agreement about the intended result of the transaction, i. e. the production fact that the executor is going to create as well as the intended time of creation. In the execution phase this product is created by the executor, and in the result phase both actors try to reach agreement about the fact that has been produced. The result of every successful transaction is the creation of a production fact.

Second, in order to take into consideration all possible exceptions that may occur in the coordination of production acts, different types of transactions are provided in DEMO. The *standard transaction pattern* as shown in Fig. 5 contains two kinds of exceptions: decline and reject. The so-called *basic transaction pattern* is comprised only of the request, promise, state, and accept coordination acts. The *complete transaction pattern* handles all possible exceptions that may occur in a transaction and is constituted by the standard pattern and four *cancellation patterns*. Cancellation patterns concern the revocation of a request act, promise act, state act, or accept act. With the complete transaction pattern DEMO ensures that all *exceptions* concerning the coordination of a production act are taken into consideration.

Third, DEMO allows for a drastically reduction in complexity. By applying the complete transaction pattern a first substantial reduction of complexity in understanding the construction and operation of enterprises is achieved. A second reduction of complexity is attained in distinguishing between three different human abilities, which are involved in the activities they perform: the forma, informa and performa abilities (Dietz 2006, pp. 105–114). The three distinct human abilities play a role in the performance of coordination acts and production acts and the design of these abilities results in different types of transactions, which allow to reduce the complexity of the corresponding enterprise models. The distinction mentioned is explained here by means of the production acts: The forma ability is concerned with the form aspects of information in terms of information transmission and storage. This type of production acts is known as *datalogical acts*. Transactions that contain a datalogical act are called *datalogical transactions*. Datalogical transactions deal with the documentation of information in the enterprise and only take into account the form of information. The informa ability states that information can be reasoned, computed or deduced. Those activities are known as *infological acts* and the corresponding transactions are called *infological transactions* if they include this type of production act. Actors performing infological transactions bring changes to information and knowledge by performing infological production acts. The performa ability concerns making decisions, judgments, or creating material things such as products. This is what Dietz calls *ontological acts* or *ontological transactions* respectively. The ontological transactions concern the essence of the enterprise. It consists of actors who directly contribute to the goals and functions of an enterprise by performing ontological production acts.

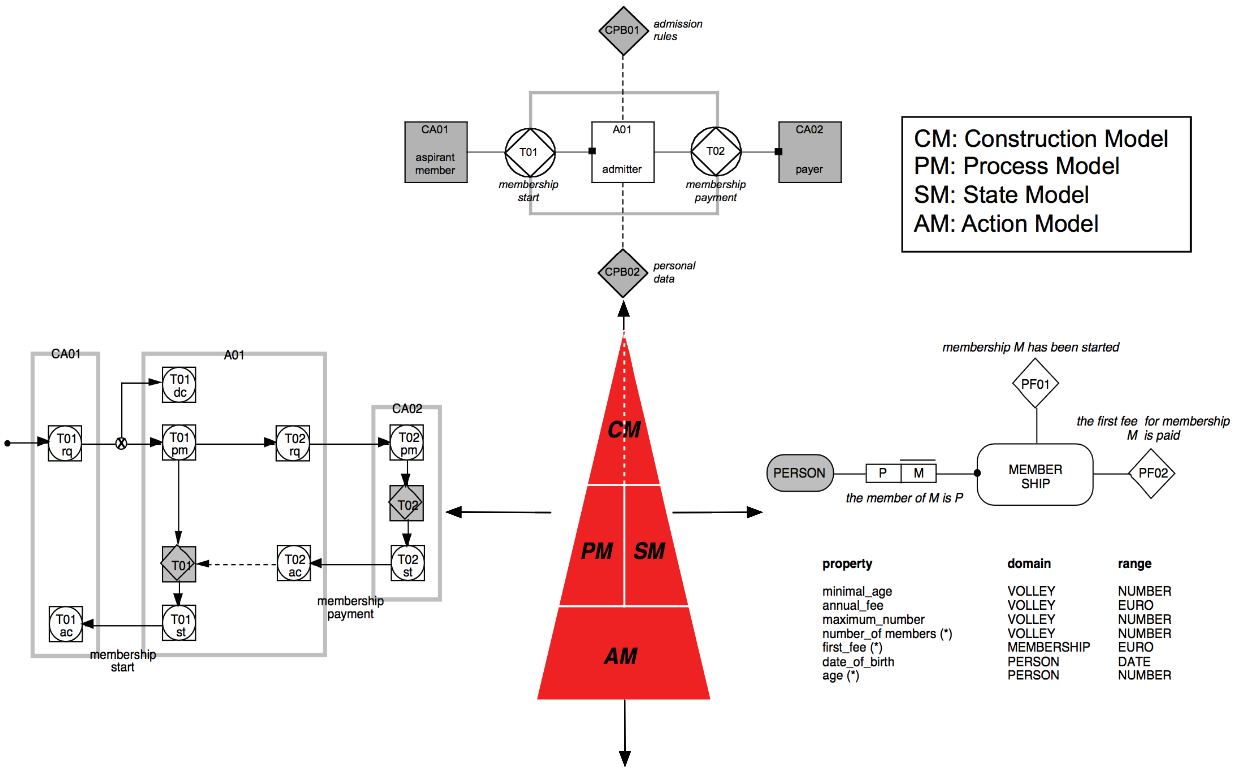
An Enterprise Ontology modelled with DEMO is comprised of four related types of aspect models: the *construction model*, the *process model*, the *action model* and the *state model* (see Fig. 6).

The *construction model* of an enterprise specifies its composition, its environment, and its structure. The composition and the environment are both a set of actor roles. The interactions between actor roles are shown by means of their transaction types. The *process model* shows how the distinct transactions are interrelated. In principle, it specifies for every included transaction type the process steps allowed to be taken. The *state model* is the specification of the state space (i. e. the set of allowable states) of both the production world and the coordination world of an enterprise. It therefore contains the conceptual model of all facts that are produced and all facts that are used. It consists of specifying the object classes, fact types, result types, as well as existential laws that hold. The *action model* defines the guidelines or procedures actors follow in order to act responsibly. These guidelines or procedures are called action rules.

When applying the DEMO methodology to enterprises, *instantiations* of the different models—construction, process, action and state model—are created. Those instantiations are the essential ontological models of an enterprise. These models are generic enterprise models, which can be implemented within specific organisational contexts in order to reach defined goals.

Having introduced the basics of the DEMO methodology, we now use the conceptual framework, as introduced in section 3, to highlight the essential building blocks and concepts of the DEMO methodology. The result is shown in Fig. 7.

DEMO finds its roots in the scientific fields of Language Philosophy, as e. g. Social Action Theory, in particular the Theory of Communicative Action (Habermas 1984), or Speech Act Theory (Searle 1969), and in Systemic Ontology (Bunge 1979). These *explanatory and/or predictive theories* are the kernel theories on which the DEMO methodology is based on (see Fig. 7). The basic concepts of DEMO, summarised by Dietz in the *Performance in Social Interaction (PSI) theory* (Dietz 2006, pp. 81–125), constitute the *explanatory design theory*. The general requirements specify the reason for the general components.



property	domain	range
minimal_age	VOLLEY	NUMBER
annual_fee	VOLLEY	EURO
maximum_number	VOLLEY	NUMBER
number_of members (*)	VOLLEY	NUMBER
first_fee (*)	MEMBERSHIP	EURO
date_of_birth	PERSON	DATE
age (*)	PERSON	NUMBER

```
on requested T01(M) with member(new M) = P
  if age(P) < minimal_age or #members(Volley)=maximum_number(current_year) → decline T01(M)
  ◊ age(P) ≥ minimal_age and #members (Volley) < maximum_number(current_year) → promise T01(M)
  fi
no
```

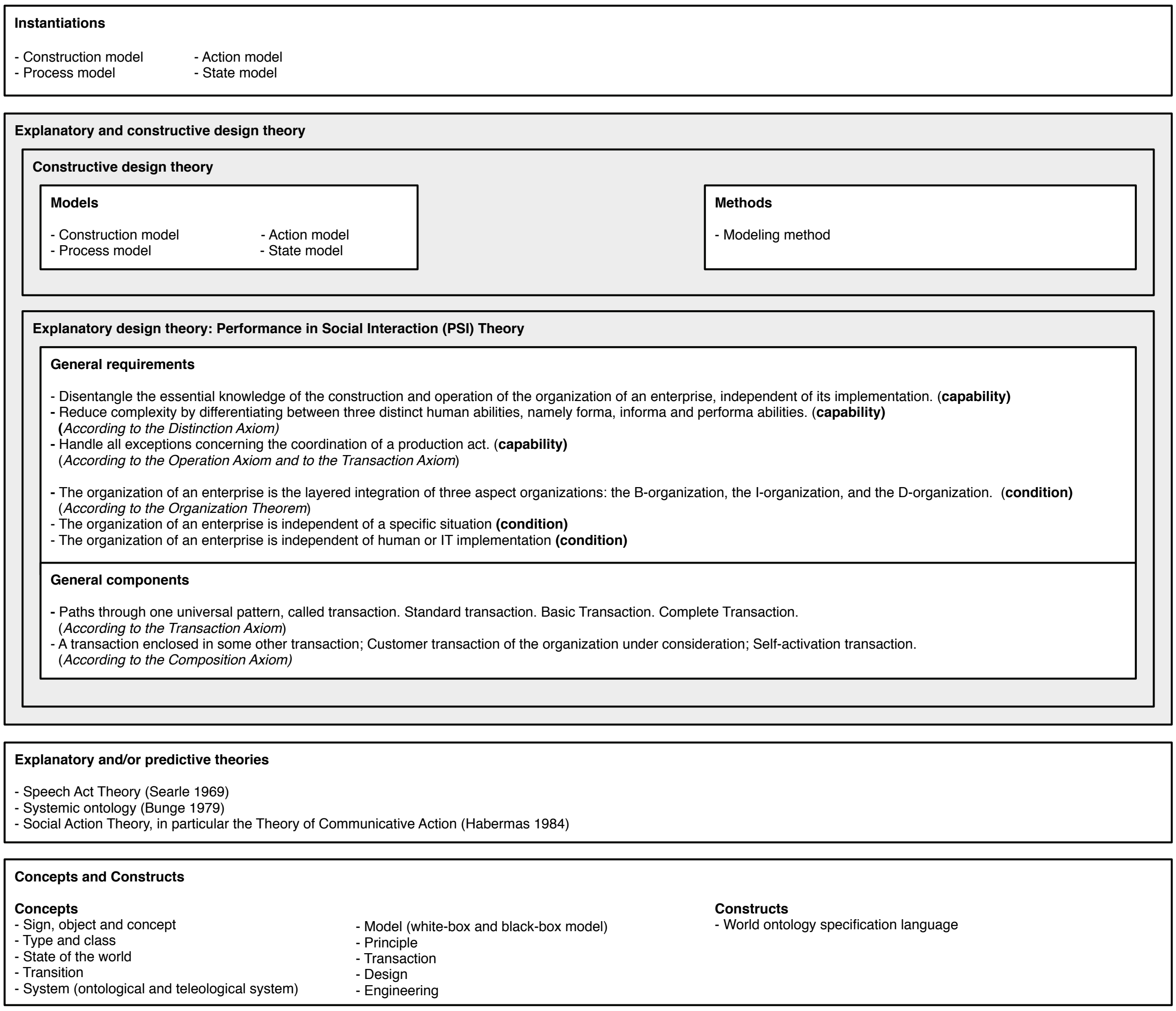
Figure 6: DEMO *Aspect Models (Dietz 2006)*

That means that for example the components *basic transaction*, *standard transaction* and *complete transaction* are justified by the requirements of *disentangle the essential knowledge of the construction and operation of the organisation of an enterprise* and *handling all exceptions concerning the coordination of a production act*. The paths trough such a universal pattern called a transaction, are essential for the DEMO methodology. Explicit prescriptions on *how* to design and develop the DEMO artefacts by means of construction, process, state and action models are given in the modelling method of DEMO.

5 PICTURE

PICTURE is a domain specific modelling method, which, in contrary to general purpose modelling methodologies, is created to solve problems within a particular area of concern and feature specific vocabulary of said domain (Frank 2010). In fact, PICTURE has originally been developed to mainly address reorganisation projects in public administrations (Becker et al. 2007). It aims at efficient modelling of the entire process landscape of an organisation, thereby taking the specific legal and political constraints within the public administration domain into account. The PICTURE method consists of the actual modelling language and the procedure model which guides the application of the language (Becker et al. 2007). The application of the PICTURE method is supported by means of a web-based tool.

The modelling language consists of 24 domain specific process building blocks from the area of public administrations. A process building block represents a defined set of activities within an ad-



Instantiations

- Construction model
- Process model
- Action model
- State model

Explanatory and constructive design theory

Constructive design theory

Models

- Construction model
- Process model
- Action model
- State model

Methods

- Modeling method

Explanatory design theory: Performance in Social Interaction (PSI) Theory

General requirements

- Disentangle the essential knowledge of the construction and operation of the organization of an enterprise, independent of its implementation. (**capability)**
- Reduce complexity by differentiating between three distinct human abilities, namely forma, informa and performa abilities. (**capability)** (*According to the Distinction Axiom)*
- Handle all exceptions concerning the coordination of a production act. (**capability)** (*According to the Operation Axiom and to the Transaction Axiom*)

- The organization of an enterprise is the layered integration of three aspect organizations: the B-organization, the I-organization, and the D-organization. (**condition)** (*According to the Organization Theorem*)
- The organization of an enterprise is independent of a specific situation **(condition)**
- The organization of an enterprise is independent of human or IT implementation **(condition)**

General components

- Paths through one universal pattern, called transaction. Standard transaction. Basic Transaction. Complete Transaction. (*According to the Transaction Axiom*)
- A transaction enclosed in some other transaction; Customer transaction of the organization under consideration; Self-activation transaction. (*According to the Composition Axiom)*

Explanatory and/or predictive theories

- Speech Act Theory (Searle 1969)
- Systemic ontology (Bunge 1979)
- Social Action Theory, in particular the Theory of Communicative Action (Habermas 1984)

Concepts and Constructs

Concepts

- Sign, object and concept
- Type and class
- State of the world
- Transition
- System (ontological and teleological system)
- Model (white-box and black-box model)
- Principle
- Transaction
- Design
- Engineering

Constructs

- World ontology specification language

Figure 7: Application of the Conceptual Framework to DEMO

ministrational process. Examples of such building blocks are listed in Tab. 1.

Since the process building blocks of PICTURE are domain specific, they apply the domain vocabulary, have a specific meaning, and are defined on a specific level of abstraction. This simplifies modelling and facilitates understanding of process models. Each process building block can be further described using a set of attributes. In Fig. 8, the building block *incoming document* is depicted, which is described by the four attributes *document*, *source*, *source medium* and *software system*.

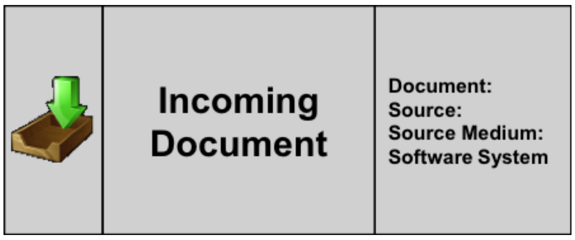


Figure 8: Sample Process Building Block with Attributes (adapted from Becker et al. 2007)

The 24 process building blocks of PICTURE create the basis for modelling processes. In PICTURE, processes can consist of several sub-processes.

Table 1: Process Building Block Examples (adapted from Becker et al. 2007)

Process Building Block	**Definition of the Process Building Block**
Incoming Document	A document, which arrives from an internal or external source.
Create Document	A new document is generated.
Print Document	A document is outputted with a printer.
Formal Assessment	A proposal is formally assessed and a decision is reached.
Enter Data into IT	Facts or documents are manually entered into an IT system.

For each sub-process a single organisational unit is responsible. By linking different sub-processes together, the complete process is visualised while the change between different organisational units is also captured. Sub-processes consist of building blocks that are aligned as a sequential flow, forming the activities executed within a business process. While constructs like parallel processes or logical branches cannot be modelled directly, sub-processes can be used to simulate such 'complications'. Logical branching for example is captured in PICTURE using sub-processes that represent process variants.

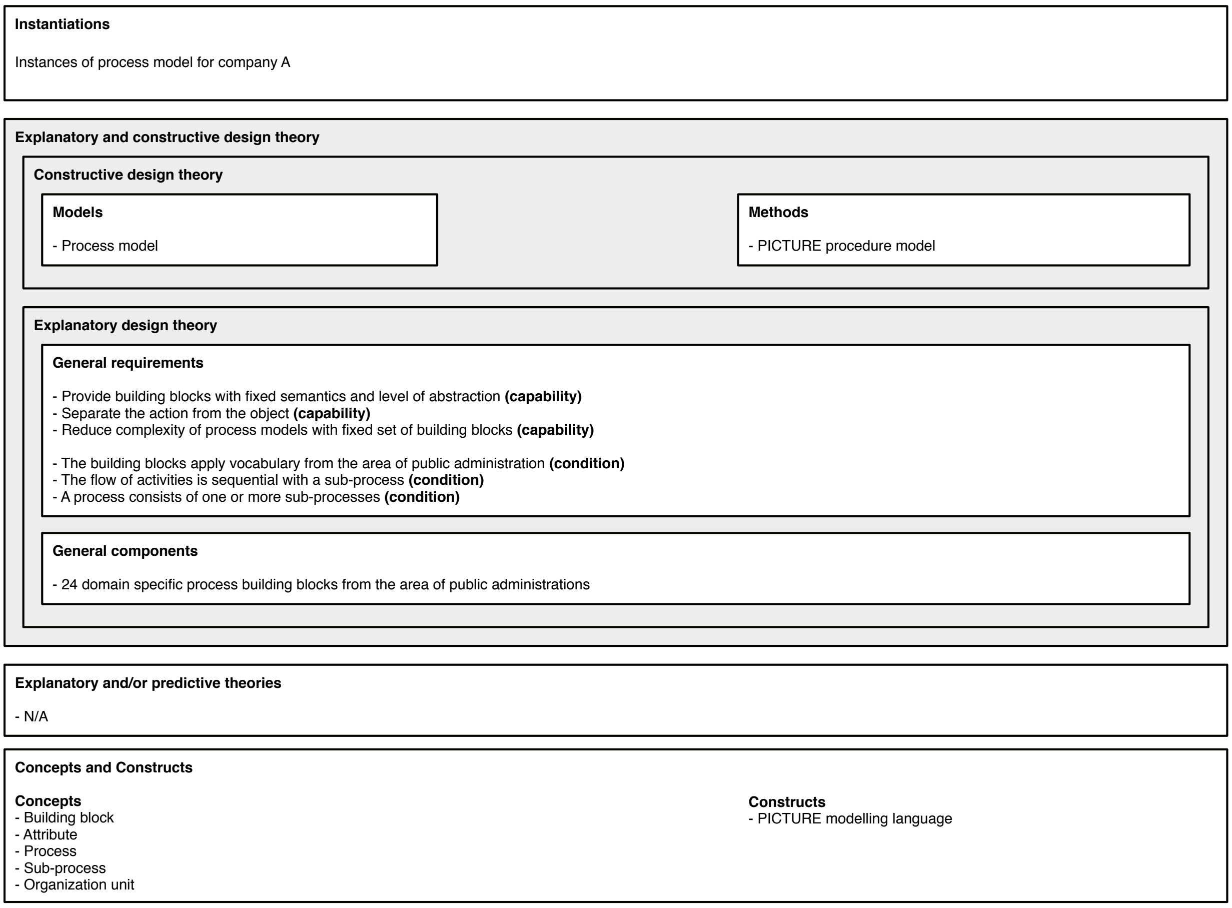
The PICTURE procedure model guides the application of the PICTURE modelling language. This procedure model is composed of three steps (Becker et al. 2007). Step one represents the *preparation of modelling*. In this step, the PICTURE method is adapted to the specific characteristics and objectives of the project. Basically, the attributes that need to be captured in the modelling step and which are important to the project are defined. After preparation, the *actual modelling* starts in the second step. The complete process landscape within the scope of the project is modelled using the building blocks. Process owners identify required processes. In the third step, process models are *analysed and used*. Benefits of these models are increased process transparency and proper documentation of the process landscape. Analyses include identifying potential for process improvements and automation through the use of IT.

To analyse the PICTURE method we apply the conceptual framework as introduced in section 3. As shown in Fig. 9, PICTURE is neither based on explanatory nor predictive theories. Nevertheless concepts and constructs of PICTURE are clearly defined. The domain specific process building blocks are central for the PICTURE method and are justified e. g. by the general requirements of *providing building blocks with fixed semantics and levels of abstraction* as well as *reduce complexity of process models with fixed set of building blocks.* How the building blocks are applied is defined in the PICTURE procedure model.

6 Advantages of Composing DEMO and PICTURE

In this section we show by means of an example from the energy sector why DEMO and PICTURE where chosen as methodologies to document the meter reading process. Based on the application of the framework to both methodologies the advantages can be shown. However, the selection of the two methodologies at this stage was based on the author's methodological knowledge. In the future however, when the framework is applied to more methodologies, the resulting analysis should provide the bases for selection.

Both methodologies DEMO and PICTURE are based on a solid foundation. While DEMO and PICTURE both have clearly defined concepts and constructs as seen in Fig. 7 and Fig. 9 under 'Concepts and Constructs', DEMO is even based on theory (see 'Explanatory and/or predictive theories' in Fig. 7). Both methodologies aim at contributing to the same problem class, namely achieving transparency and reducing complexity as listed under 'General requirements' of both


Instantiations

Instances of process model for company A

Explanatory and constructive design theory

Constructive design theory

Models

- Process model

Methods

- PICTURE procedure model

Explanatory design theory

General requirements

- Provide building blocks with fixed semantics and level of abstraction **(capability)**
- Separate the action from the object **(capability)**
- Reduce complexity of process models with fixed set of building blocks **(capability)**

- The building blocks apply vocabulary from the area of public administration **(condition)**
- The flow of activities is sequential with a sub-process **(condition)**
- A process consists of one or more sub-processes **(condition)**

General components

- 24 domain specific process building blocks from the area of public administrations

Explanatory and/or predictive theories

- N/A

Concepts and Constructs

Concepts
- Building block
- Attribute
- Process
- Sub-process
- Organization unit

Constructs
- PICTURE modelling language

Figure 9: Application of the Conceptual Framework to PICTURE

framework applications in Fig. 7 and Fig. 9 (*Reducing complexity by differentiating three distinct human abilities* listed for DEMO and *Reduce complexity of process models with fixed set of building blocks* listed for PICTURE). The reduction of complexity is achieved by applying *generic artefacts* (transaction patterns in case of DEMO and domain specific process building blocks in case of PICTURE—both listed under 'General components' in the frameworks in Fig. 7 and Fig. 9). These generic artefacts can be implemented for or adapted to a specific situation at hand. Regarding their result types, however, both methodologies differ. DEMO focuses on modelling the essence of the organisation of an enterprise by means of its construction model, which specifies the composition, environment, and structure of an enterprise. This fact is listed in Fig. 7 as a 'capability' of the methodology under 'General requirements'. The composition and the environment are both described by a set of actor roles, which interact by means of transactions. A transaction is a path through a pattern of coordination acts needed to execute a production act, which is a *generic artefact* and listed in the framework in Fig. 7 under 'General components'. The process model of DEMO shows how the distinct transactions are interrelated, focusing on the coordination between actor roles. DEMO does not provide any concepts for describing how the production acts are performed or implemented. While DEMO aims at revealing the essence of an organisation by means of generic organisational models, which are completely independent of their implementation, PICTURE provides process building blocks (listed as 'General components' in Fig. 9), which represent specific sets of activities within a process allowing for an efficient documentation of the implemented process landscape. The composition of DEMO and PICTURE results in essential constructional models of the organisation (resulting from applying DEMO), with a detailed description of how to implement the processes (resulting from applying PICTURE).

The value of choosing the two methodologies lies in the extension of the implementation independent DEMO models—which focus on the construction of an organisation by means of modelling the essential business transactions, its actor roles, action rules and state space of the organisation, without dealing with implementation specific aspects e. g. of the processes—with the implementation specific process variants of PICTURE, which are based on *generic process building blocks*.

One can however argue that other methodologies, such as Business Engineering (Österle and Winter 2003), or ARIS (Scheer and Schneider 2005), would also fit for combining e. g. with DEMO. But due to the application of the framework in order to analyse and understand the methodologies better, we could see that both DEMO and PICTURE are a) based on clearly defined concepts and constructs, b) use *generic artefacts* (transaction patterns and domain specific process building blocks) to reduce complexity and c) focus on different aspects of modelling the organisation (DEMO focuses on revealing the essential business transactions and actor roles, and PICTURE focuses on an efficient documentation of the implemented process landscape). From the presented frameworks we understand that both approaches deliver *generic design results* for parts of the whole problem solution such that a composition of the two approaches is superior to the application of just one of them, resulting in a more complete solution. In order to really exclude other methodologies from combining them with e. g. DEMO, they should be analysed as well by means of our conceptual framework. However, this would go beyond the scope of this paper, where one focus is to show the applicability of our framework.

Our example is taken from a publicly funded project of the Swiss Commission for Technology and Innovation, which is concerned with process performance measurement of the meter-to-cash process for electrical utilities. The meter-to-cash process describes the business process starting with the reading of energy consumption data of customers and ending with the customer's payment. A first task of the project was a detailed documentation of the meter-to-cash process in order to enable further performance analyses in later stages of the project. It was required to abstract from implementation specific details, which differ for any company in the energy sector, in order to illustrate the essential process steps in the meter-to-cash process.

Since the meter-to-cash process is very complex and may be implemented in many ways within one company (dependent of the specifics of the different customers), a methodology for understanding the essence of the organisation of the billing company, responsible for the meter-2-cash process, was necessary. For this reason the DEMO methodology was chosen. It allowed for concentrating on the essential business transactions, responsible actor roles, the environment by means of business partners such as customers, electricity provider or network operator, without diving into the details of the implemented processes. An extract of the resulting DEMO models—focusing on the meter-reading sub-process—is shown in Fig. 10 (the complete diagrams can be found in the appendix).

Fig. 10 shows the Actor Transaction Diagram (ATD) with its corresponding transaction results in Tab. 2. Together the ATD and the Transaction Result Table (TRT) constitute the Construction Model (CM) of the meter-to-cash example.

The *meter reading transaction (B-T11)* with the two corresponding actor roles are highlighted in Fig. 10. The meter reading is either initiated by the *meter reading manager* actor role, in case of a timely repeating (e. g. monthly) *meter control (B-T10)*, or by the *final meter reading manager* actor role in case of an *unscheduled meter reading completion* request, e. g. due to an address change or a contract end. While Fig. 10 shows the business

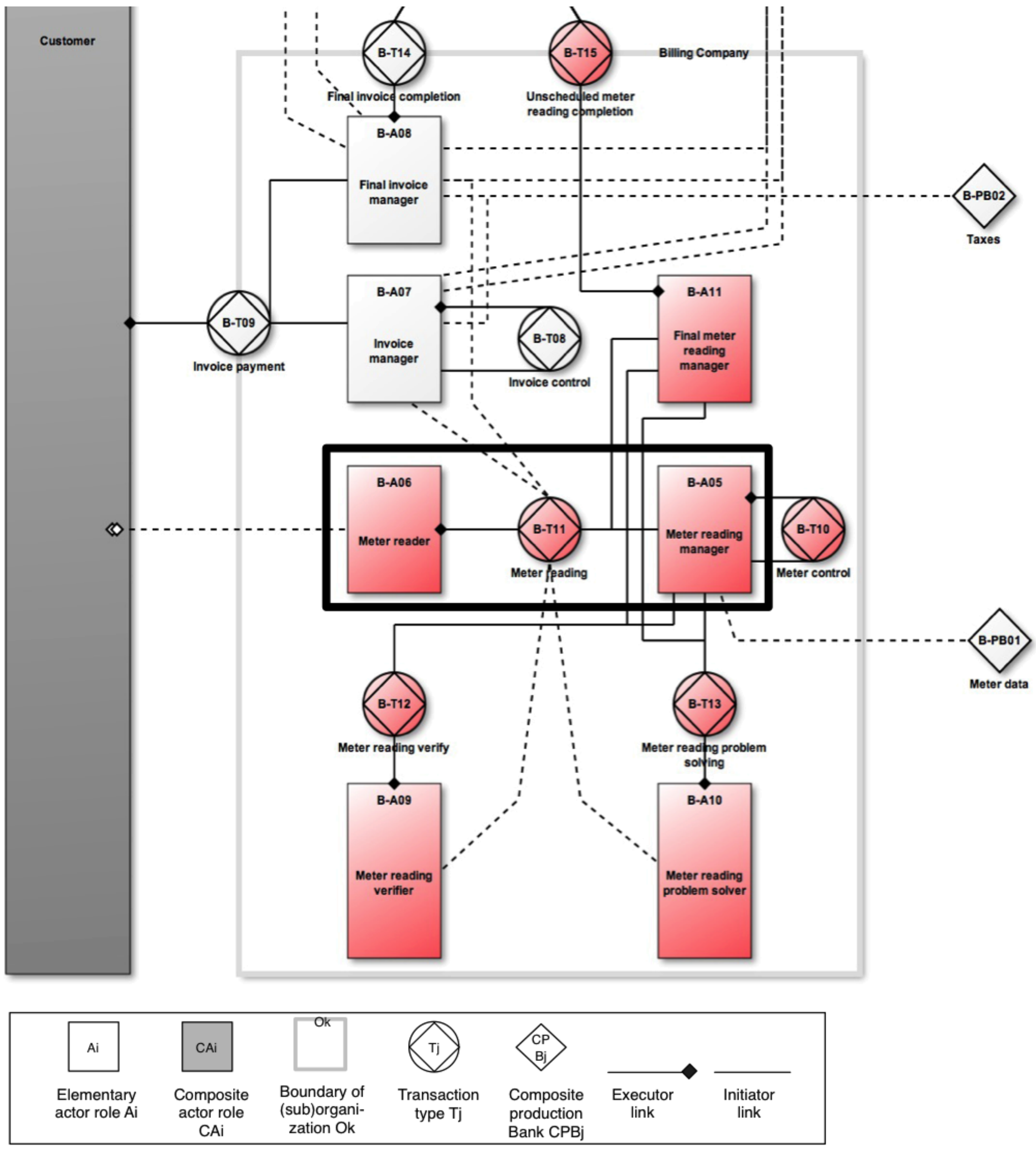


Figure 10: Extract of the Actor Transaction Diagram of the Meter-to-Cash Example

transactions, Fig. 11 details the single transactions by means of the process model, which is based on the transaction pattern as introduced in Fig. 5 and illustrates the coordination acts among the actor roles. When a *meter control request* has been received by the *meter reading manager, 1..n meter reading requests* are sent to the according *meter readers*. The *meter readers* promise the execution of the meter reading, execute it and state that meter reading is provided (*B-R11: meter reading R provided*) (see Tab. 2). As soon as the meter reading manager has accepted the stated

Table 2: Extract of the Transaction Result Table of the Meter-to-Cash Example

transaction type	**result type**
B-T10 meter control	B-R10 meter control performed
B-T11 meter reading	B-R11 meter reading R provided
B-T12 meter reading verify	B-R12 meter reading R verified
B-T13 meter reading problem solving	B-R13 problem P of meter reading R solved

transaction, a request for verification of the results is initiated (*meter reading verify (B-T12)*). In case of resulting problems the *meter reading problem solving* transaction *(B-T13)* is initiated. When all mentioned transactions are finalised and accepted by the meter reading manager, the *meter control* transaction *(B-T10)* can be finalised (see Fig. 11).

As we can see from the models introduced, the DEMO models do not provide any information about different implementation variants of the meter-to-cash process, it only visualises the coordination acts needed in order to execute the production acts, such as e. g. meter control performed, or meter reading provided. To illustrate this we look at the meter reading transaction. DEMO does not provide any implementation information about how the meter reading is done; if the reading is done by an employee of the meter reading company, if the reading is done by the customer itself or even if the meter reading is done by an electronic device, submitting the data automatically to the meter reading company. Since it is known that several implementations (by means of people and technology) exist and that those variants differ in time and costs, an additional methodology needed to be selected to extend the DEMO models with specific information about implementation variants. This was achieved by applying the PICTURE method to model variants of sub-processes of the meter-to-cash process. PICTURE was selected as an adequate method after having analysed it with the conceptual framework as shown in section 5. Due to the concentration on processes and the use of generic components (building blocks), which allowed to abstract from process details and to reuse existing knowledge available for such kind of people intensive administration processes, PICTURE was selected for extending the DEMO models with implementation specific details, allowing for modelling several variants of the generic meter-to-cash process.

Fig. 12 depicts these three variants of the meter reading sub-process. An employee of the company, a smart meter device, or the customer himself can perform the reading of electricity consumption data and its transmission to the electrical utility provider. In the first variant (as depicted left in Fig. 12), an employee of the utility receives the task to read data from various meter devices on a specific route. After getting all required information in the headquarters, the employee visits every meter device on the specified route and records the consumption data using his hand-held device. Each reading is verified. When the route is completed the employee returns to his headquarters and transmits the data to the IT system for further processing. The second variant (shown in the middle of in Fig. 12) represents a fully automated meter reading performed by a smart meter that regularly sends consumption data to the IT system. Thirdly, (shown right of Fig. 12), meter reading can be performed on a self-service basis by the customer himself. In this variant, the customer reads the electricity consumption data from the meter and sends it to the utility by using e. g. an online form.

As shown in Fig. 12 on the right, the first process step of all three variants corresponds to the request of the meter reading transaction (B-T11) in DEMO. This is independent of its implementation, i. e. if it is executed by people or technology. The 'read consumption data' step of all three variants

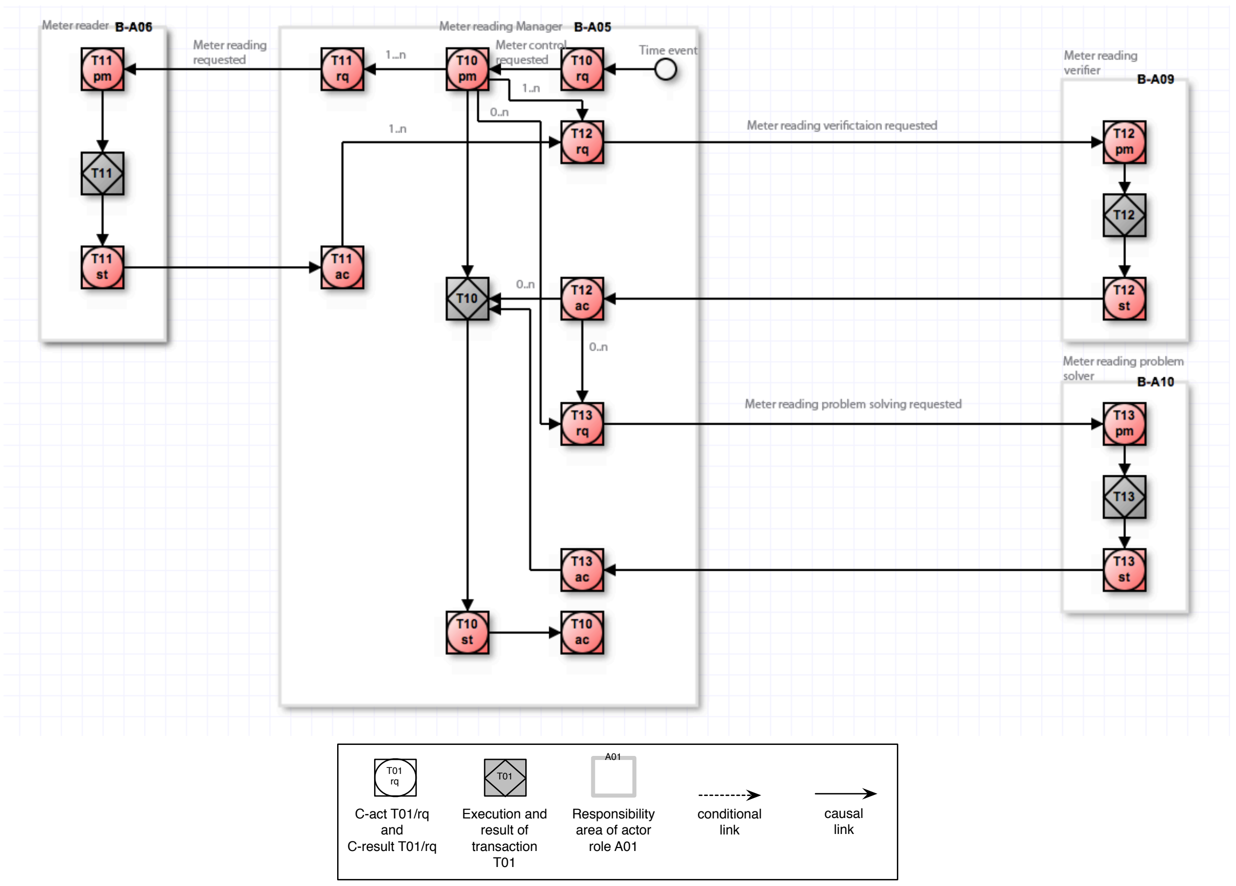

Meter reader B-A06
Meter reading requested
Meter reading Manager B-A05
Meter control requested
Time event
Meter reading verifier B-A09
Meter reading verifictaion requested
Meter reading problem solver B-A10
Meter reading problem solving requested
T11 pm
T11
T11 st
T11 rq
T11 ac
T10 pm
T10 rq
T10
T10 st
T10 ac
T12 rq
T12 ac
T13 rq
T13 ac
T12 pm
T12
T12 st
T13 pm
T13
T13 st
1..n
0..n
C-act T01/rq and C-result T01/rq
Execution and result of transaction T01
Responsibility area of actor role A01
conditional link
causal link

Figure 11: Extract of the Process Model of the Meter-to-Cash Example

corresponds to the execution of the meter reading transaction, while the last process step in all three variants corresponds to the state of the meter reading transaction. While these main process steps are implementation independent and are therefore modelled in DEMO, all other process steps modelled in PICTURE are relevant to a specific way of implementing the meter reading process. While DEMO helps in modelling the complete process steps and abstract from implementation, PICTURE extends the DEMO models by implementation specific steps, but still on a higher level of abstraction by means of the generic and reusable building blocks.

7 Conclusions and Future Work

In this paper we have proposed a conceptual framework for analysing existing Enterprise Engineering methodologies. The framework allows to study said methodologies regarding their a) underlying concepts and constructs, b) foundation based on explanatory and/or predictive theories, c) conditions or capabilities needed or possessed, d) general components provided and e) applied methods and resulting models. The analysis of methodologies based on the conceptual framework permits to better understand to which ends the methodologies provide which means. Having a clear understanding of the means-end relationship allows for choosing the appropriate methodologies to solve desired problems.

The necessity for the conceptual framework arises from the fact that several classifications of DSR artefacts exist, but the relationship between the existing artefacts is mostly missing. Each artefact type contributes to understand specific aspects

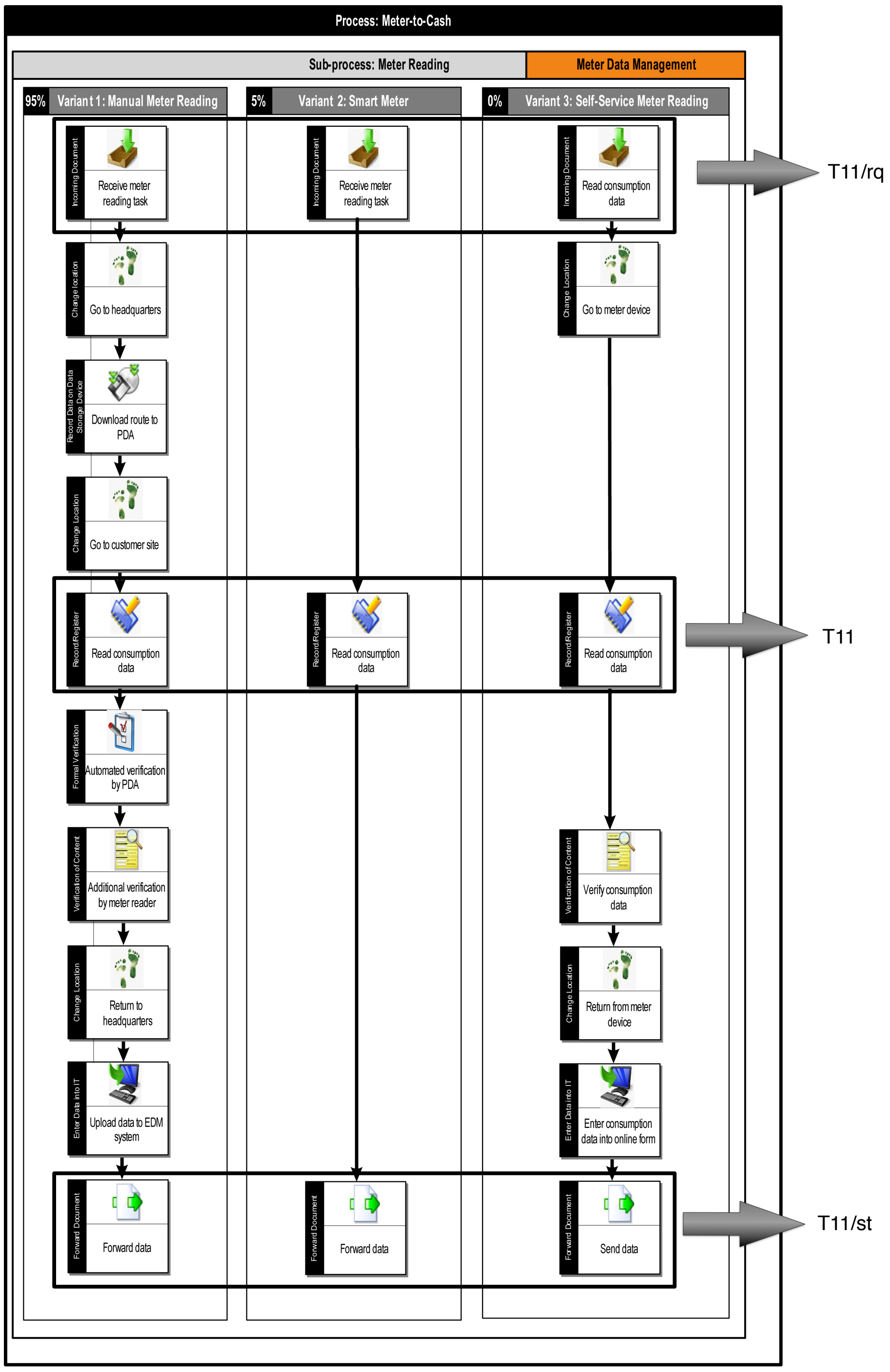


Figure 12: PICTURE Variants of the Meter-to-Cash Processes

of the methodologies, but without relationships a sufficient understanding of the means-end relationships is not possible. The contribution of the work presented in this paper is that we put the existing artefacts in relationship, resulting in the *Conceptual Model for Analysing Enterprise Engineering Methodologies.* In order to arrive at this artefact we followed the DSR approach. By means of the rigor cycle we based our framework on existing and established classifications of research concepts in social sciences as well as DSR artefact types. The resulting conceptual framework is therefore rigorously derived and is our first contribution to the knowledge base of the framework of Hevner (2007).

Following the relevance cycle we applied the derived conceptual framework to two different methodologies, DEMO and PICTURE. Both methodologies are used in praxis to solve real world problems and aim at contributing to the same problem class, namely achieving transparency and allowing for processes optimisation, by means of generic artefacts, which can be implemented or adapted to specific situations at hand. The result types of the two methodologies however differ. While DEMO focuses on designing the essence of an enterprise, independent of its implementation, PICTURE allows for an efficient documentation of the implemented process landscape for specific situations. The analysis of DEMO and PICTURE allowed for a selection of the two in order to solve a real world problem resulting in combining the modelling of the essence of an enterprise, independent of its implementation, with implementation details of the essential processes for specific situations. By applying the conceptual framework to both DEMO and PICTURE we followed the relevance cycle of Hevner (2007). This resulted in our second contribution to the knowledge base of Hevner (2007), namely the analysis of two relevant methodologies by means of the conceptual framework.

Further, based on the analysis of DEMO and PICTURE by means of the conceptual framework, we could demonstrate the advantages of applying the two methodologies for an extract of a case study in the energy sector. Whereas DEMO was selected to model at higher level of abstraction the essential business transactions relevant to the meter-to-cash process, PICTURE models were produced focusing on the process details of the different variants within the meter-to-cash process. Such a detailed analysis of the meter-to-cash process, including its sub-processes, actor roles and information involved, provides the basis for further improvements and related changes in the enterprise.

With this very specific example in the energy sector we gained important information on where and how the DEMO transactions can be extended with realisation and implementation aspects. These findings may be generalised in future research such that they can be applied in any project where DEMO is used for Enterprise Engineering.

In order to establish the conceptual framework as an instrument for analysing methodologies for designing and engineering enterprises, more complete studies need to be conducted illustrating even further the advantages of using that instrument. As future work we therefore plan to study different goals and contexts in organisational change projects and to use the conceptual framework in order to identify the right means to reach desired ends. Consequently several existing Enterprise Engineering methodologies need to be analysed using the proposed conceptual framework and possible applications of methodologies need to be defined looking at the strengths of the analysed methodologies. By doing so we will not only better understand existing methodologies, but also be able to refine the presented conceptual framework. A further task in future research is the elaboration of a domain map. Such a map should allow for identifying methodologies that pursue similar goals within comparable contexts independently of their proposed solution designs. This should avoid the application of methodologies having a completely different focus. The domain map should support some pre-selection. In contrast to the selection of the two methodologies presented in this paper that was based on the author's methodological knowledge, decisions of this kind should rather be based on a decision supporting domain map that is part of Enterprise Engineering's research knowledge base.

References

Albani A., Dietz J. L. G. (2010) Preface. In: Albani A., Dietz J. L. G. (eds.) Advances in Enterprise Engineering IV. Springer, Berlin, Heidelberg

Albani A., Dietz J. L. G., Verelst J. (2011) Preface. In: Albani A., Dietz J. L. G., Verelst J. (eds.) Advances in Enterprise Engineering V. Springer, Berlin, Heidelberg

Baskerville R. L., Pries-Heje J. (2010) Explanatory Design Theory. In: Business & Information Systems Engineering 2(5), pp. 271–282

Becker J., Pfeiffer D., Räckers M. (2007) Domain Specific Process Modelling in Public Administrations – The PICTURE-Approach. In: Wimmer M. A., Scholl J., Grönlund A. (eds.) Electronic Goverment: 6th International Conference, EGOV 2007, Proceedings Vol. 4656. Springer, Berlin and Heidelberg, pp. 68–79

Bunge M. A. (1979) Ontology II: A World of Systems. Treatise on Basic Philosophy Vol. 4. D. Reidel Publishing Company, Dodrechte, The Netherlands

Chmielewicz K. (1994) Forschungskonzeptionen der Wirtschaftswissenschaften, 3rd ed. Poeschel, Stuttgart

Davenport T. H., Short J. E. (1990) The New Industrial Engineering – Information Technology and Business Process Redesign. In: Sloan Management Review 31(4), pp. 11–27

Dietz J. L. G. (2006) Enterprise Ontology – Theory and Methodology. Springer, Berlin, Heidelberg

Dietz J. L. G., Hoogervorst J. A. G., Albani A., Aveiro D., Babkin E., Barjis J., Caetano A., Huysmans P., Iijima J., Van Kervel S. J. H., Mulder H., Op 't Land M., Proper H. A., Sanz J., Terlouw L., Tribolet J., Verelst J., Winter R. (2013) The discipline of Enterprise Engineering. In: International Journal of Organizational Design and Engineering 3(1), pp. 86–114

Fischer C., Winter R., Wortmann F. (2010) Design Theory. In: Business and Information Systems Engineering 52(6), pp. 383–386

Frank U. (1998) Evaluating Modelling Languages: Relevant Issues, Epistemological Challanges and a Preliminary Research Framework. Institut für Wirtschaftsinformatik, Universität Koblenz, Koblenz

Frank U. (2010) Outline of a Method for Designing Domain-Specific Modelling Languages. Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik

Gehlert A., Schermann M., Pohl K., Krcmar H. (2009) Towards a research method for theory driven design research. In: Hansen H. R., Karagiannis D., Fill H. G. (eds.) Business Services: Konzepte, Technologien, Anwendungen; Proceedings der 9. Internationalen Tagung Wirtschaftsinformatik. Österreichische Computer Gesellschaft, pp. 441–450

Goldkuhl G. (2004) Design Theories in Information Systems – A Need for Multi-Grounding. In: Journal of Information Technology Theory and Application 6(2), pp. 59–72

Gregor S. (2002) Design theory in Information Systems. In: Australasian Journal of Information Systems 10(1), pp. 14–22

Gregor S. (2006) The Nature of Theory in Information Systems. In: MIS Quarterly 30(3), pp. 611–642

Gregor S., Jones D. (2007) The Anatomy of a Design Theory. In: Journal Of The Association For Information Systems 8(5), pp. 312–335

Habermas J. (1984) The theory of communicative action: Reason and the rationalization of society. Beacon, Boston

Hackathorn R. D., Karimi J. (1998) A Framework for Comparing Information Engineering Methods. In: MIS Quarterly

Hammer M., Champy J. (1993) Reengineering the Corporation – A Manifesto for Business Revolution. HarperCollins Publishers, New York

Hevner A. R. (2007) A Three Cycle View of Design Science Research. In: Scandinavian Journal of Information Systems 19(2), pp. 87–92

Hevner A. R., March S. T., Park J., Ram S. (2004) Design Science in Information Systems Research. In: MIS Quarterly 28(1), pp. 75–105

Iivari J. (2007) A Paradigmatic Analysis of Information Systems As a Design Science. In: Scandinavian Journal of Information Systems 19(2)

Iivari J., Hirschheim R., Klein H. K. (2001) A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches. In: Journal of Management Information Systems 17(3), pp. 179–218

Kerlinger F. N. (1964) Foundations of Behavioral Research; Educational and Psychological Inquiry. Holt Rinehart and Winston, New York

March S. T., Smith G. F. (1995) Design and Natural Science Research on Information Technology. In: Decision Support Systems 15(4), pp. 251–266

Mingers J. (2001) Combining IS research methods: towards a pluralistic methodology. In: Information Systems Research 12(3), pp. 240–259

Moody D. L. (2003) The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods. In: Ivan (ed.) Proceedings of the 11th European Conference on Information Systems (ECIS), pp. 1327–1336

Moody D. L., Sindre G., Brasethvik T., Solvberg A. (2002) Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework. In: Spaccapietra S., March S. T., Kambayashi Y. (eds.) 21st International Conference on Conceptual Modeling – ER

Nelson J. H., Poels G., Genero M., Piattini M. (2012) A conceptual modeling quality framework. In: Journal of Software Quality Control 20(1), pp. 201–228

Office of Government Commerce (2005) Introduction to ITIL. Stationary Office, London

Österle H., Winter R. (2003) Business Engineering. In: Österle H., Winter R. (eds.) Business Engineering – Auf dem Weg zum Unternehmen des Informationszeitalters Vol. 2. Springer, Berlin etc., pp. 3–19

Scheer A.-W., Schneider K. (2005) ARIS – Architecture of Integrated Information Systems. In: Bernus P., Mertins K., Schmidt G. (eds.) Handbook on Architectures of Information Systems Vol. 2. Springer, Berlin, Heidelberg, pp. 605–623

Searle J. R. (1969) Speach Acts, an Essay in the Philosophy of Language. Cambridge University Press, Cambridge MA

The Open Group (2009) TOGAF Version 9 – The Open Group Architecture Framework (TOGAF). Van Haren, Zaltbommel

Venable J. (2006a) A Framework for Design Science Research Activities. In: Emerging Trends and Challenges in Information Technology Management Vol. 1. Idea Group Publishing, 2006 Information Resources Management Association International Conference, pp. 184–187

Venable J. (2006b) The Role of Theory and Theorising in Design Science Research. In: Chatterjee S., Hevner A. (eds.) Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006). Claremont Graduate University, Claremont, CA, pp. 1–18

Walls J. G., Widmeyer G. R., El Sawy O. A. (1992) Building an Information System Design Theory for Vigilant EIS. In: Information Systems Research 3(1), pp. 36–59

Winter R. (2008) Design Science Research in Europe. In: European Journal Of Information Systems 17(5), pp. 470–475

Appendix

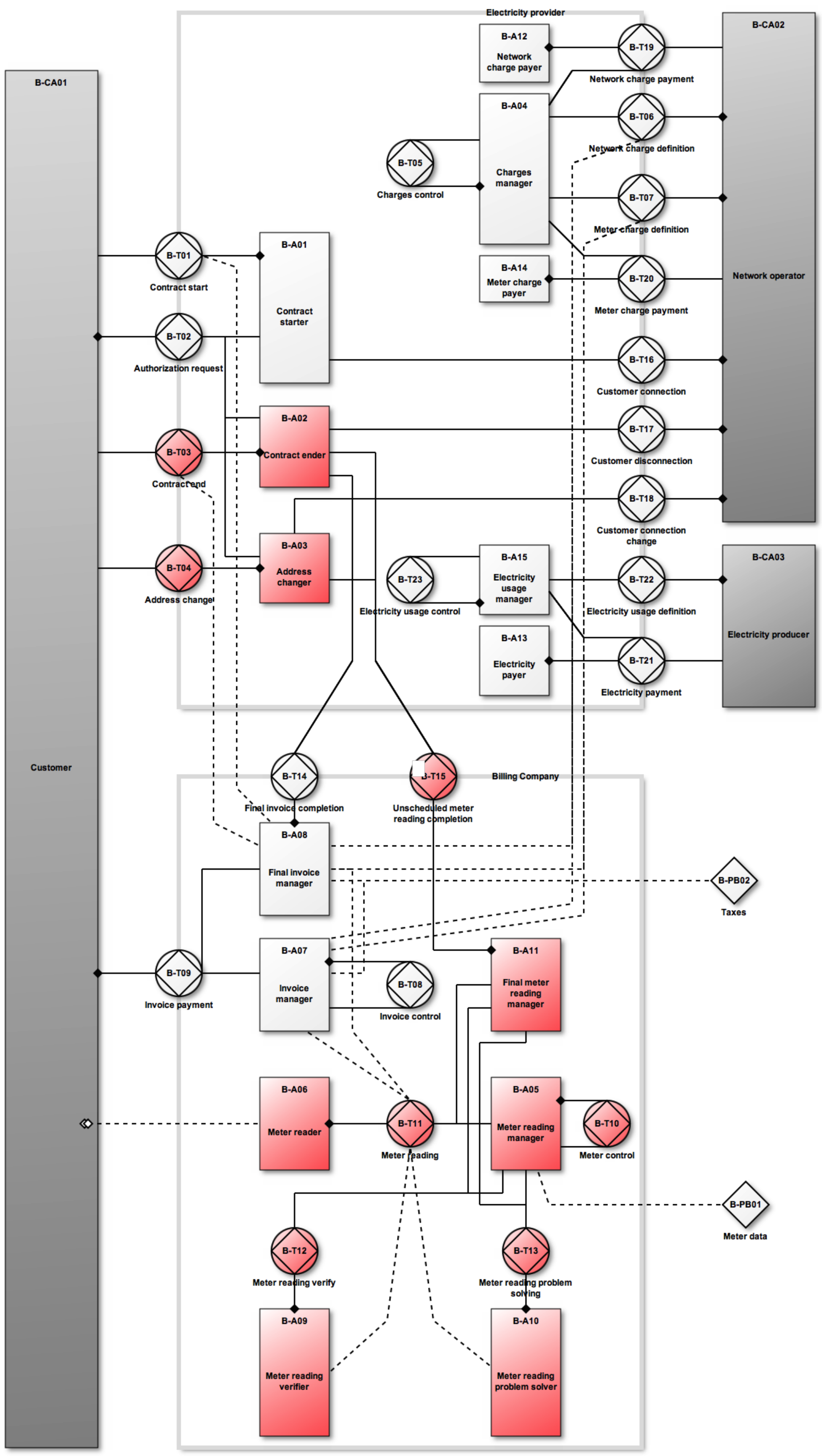


Figure 13: Actor Transaction Diagram of the Meter-to-Cash Example (transactions relevant to the meter reading process are highlighted in red)

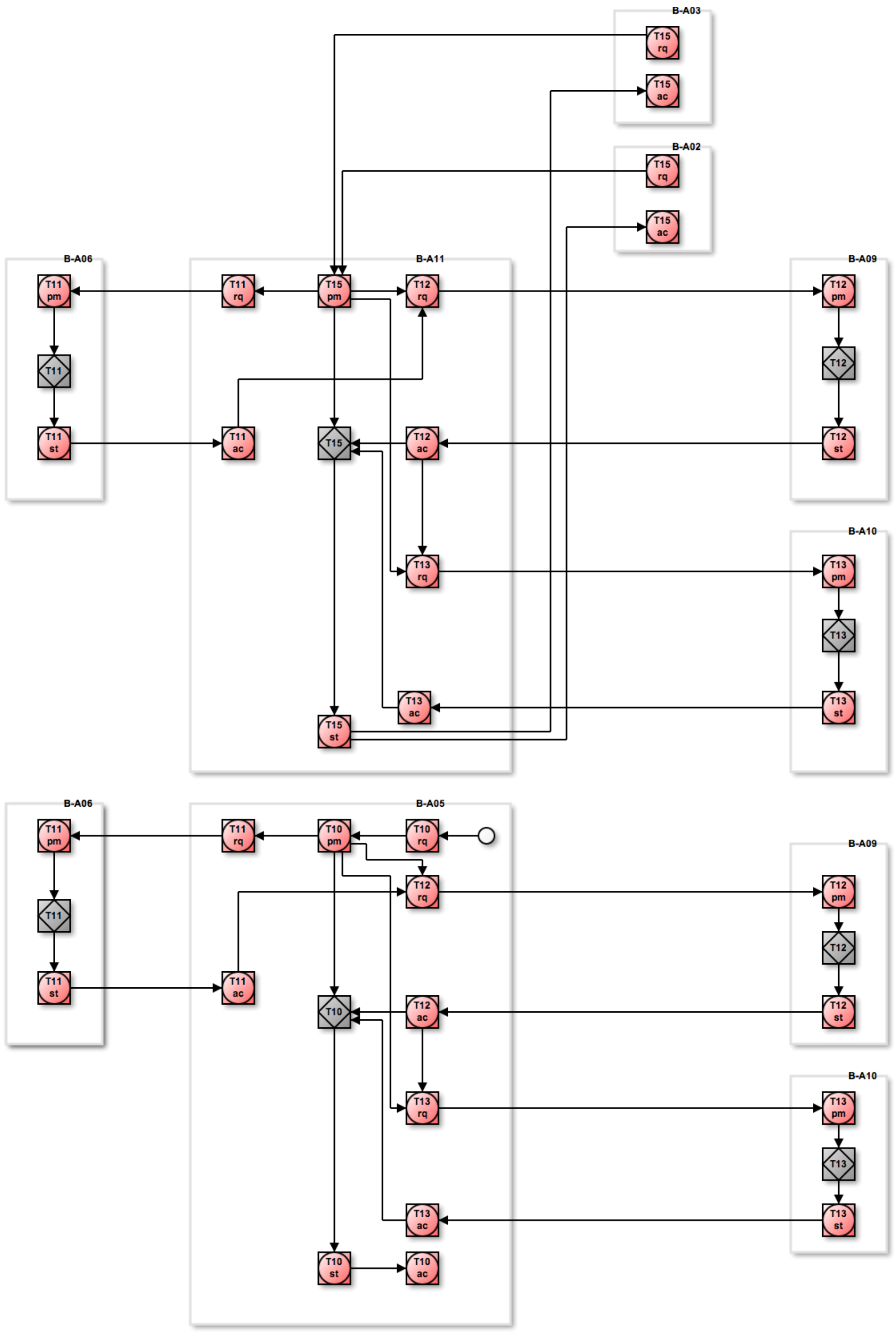


Figure 14: Meter Reading Process Model of the Meter-to-Cash Example

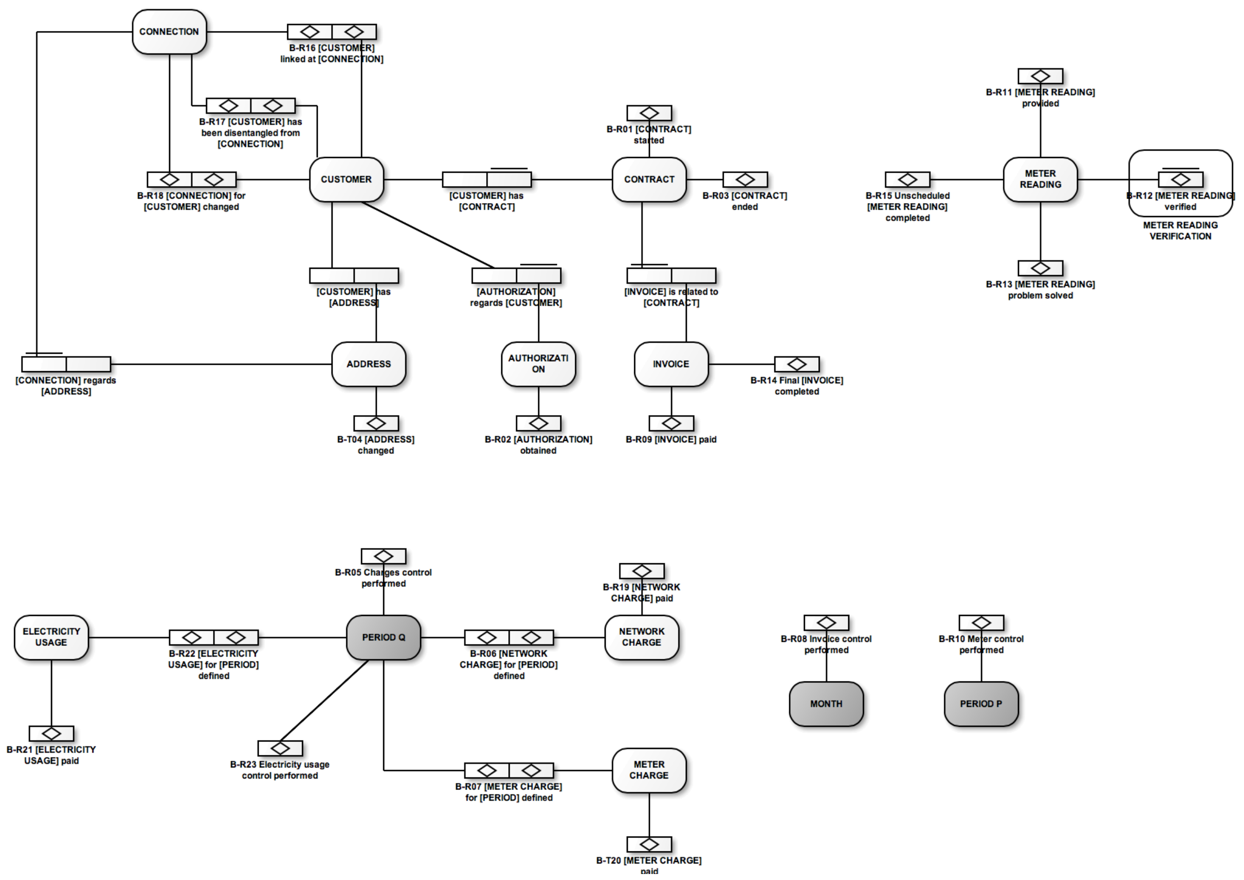

CONNECTION
B-R16 [CUSTOMER] linked at [CONNECTION]
B-R17 [CUSTOMER] has been disentangled from [CONNECTION]
B-R18 [CONNECTION] for [CUSTOMER] changed
CUSTOMER
[CUSTOMER] has [CONTRACT]
B-R01 [CONTRACT] started
CONTRACT
B-R03 [CONTRACT] ended
[CUSTOMER] has [ADDRESS]
[AUTHORIZATION] regards [CUSTOMER]
[INVOICE] is related to [CONTRACT]
[CONNECTION] regards [ADDRESS]
ADDRESS
AUTHORIZATION
INVOICE
B-R14 Final [INVOICE] completed
B-T04 [ADDRESS] changed
B-R02 [AUTHORIZATION] obtained
B-R09 [INVOICE] paid
B-R11 [METER READING] provided
B-R15 Unscheduled [METER READING] completed
METER READING
B-R12 [METER READING] verified
METER READING VERIFICATION
B-R13 [METER READING] problem solved
B-R05 Charges control performed
B-R19 [NETWORK CHARGE] paid
ELECTRICITY USAGE
B-R22 [ELECTRICITY USAGE] for [PERIOD] defined
PERIOD Q
B-R06 [NETWORK CHARGE] for [PERIOD] defined
NETWORK CHARGE
B-R08 Invoice control performed
B-R10 Meter control performed
MONTH
PERIOD P
B-R21 [ELECTRICITY USAGE] paid
B-R23 Electricity usage control performed
B-R07 [METER CHARGE] for [PERIOD] defined
METER CHARGE
B-T20 [METER CHARGE] paid

Figure 15: Object Fact Diagram of the Meter-2-Cash Example