Baltic J. Modern Computing, Vol. 5 (2017), No. 4, 329-350
http://dx.doi.org/10.22364/bjmc.2017.5.4.01

Normalization of Domain Modeling in Enterprise Software Development

Saulius GUDAS, Andrius VALATAVICIUS

Vilnius University, Institute of Mathematics and Informatics, Vilnius, Lithuania
Akademijos St. 4, LT-2021 Vilnius, Lithuania

saulius.gudas@mii.vu.lt, andrius.valatavicius@mii.vu.lt

Abstract. Normalization has become traditional in database design theory and practice. One disadvantage of the model-driven development is that usage of concepts normalization, and functional dependency in the enterprise software engineering is limited to only one stage of system development life cycle (SDLC) - the database design stage. The provided research of these concepts motivate normalization of the entire SDLC. The main part of the paper is devoted to the normalization of the enterprise modeling stage, which is based on the perceived causality of the target domain. The concepts of management functional dependency (MFD) and management transaction (MT) introduced for capturing causal dependencies within the target domain. The first step is the discovery of MFD of business activities. MT is an initial specification of MFD, which gives a basis for enterprise model normalization using the detailed frameworks. Enterprise model normal forms ENF1 – ENF5 defined and illustrated.

Keywords: normalization, domain causality, functional dependency, management transaction, knowledge-driven transformation, enterprise model, normal forms

1. Introduction

The concept of normalization is known in distinct research and technology areas, for instance, in statistics, mathematics, domain modeling, process modeling, ontological domain modeling, automatic knowledge representation, comparison and retrieval, data modeling (associated with signal processing), service normalization, domain ontology modeling (normalization of large ontologies), workflow normalization (based on the Petri Nets). Understanding of normalization in distinct subject domains are divergent, however, some generic features are observable. Normalization process seeks to transform the initial system (an *empirical* model of the subject domain) to the normal form (a *normalized* model).

The practical need for normalization within the area of information systems engineering (ISE) and enterprise software systems engineering need has already emerged (Fong, 2015), (Eessaar, 2014), (Gudas, 2012, 2015, 2016), (Allard et al., 2010), (Kecheng et al., 2001), (Date, 1999), (Kent, 1983).

New normalization theories - Normalization Process Theory (NPT), Normalized Systems Theory (NST) and Norm Analysis method (NA) have emerged in different, however, co-related areas of organizational management, organizational modeling, and requirements engineering (Van Nuffel et al., 2009), (Murray et al., 2010), (Mannaert et al., 2009, 2011), (Eessaar, 2014), (Linden et al., 2012), (De Bruyn et al., 2012), (Tan et al., 2004), (Chong and Liu, 2000).

One of the key features is that the normalization procedure uses the already discovered causal dependencies inside the particular subject domain. For instance, normal forms of database models are based on the concept of functional dependency (Date, 1999), (Kent, 1983). Normalization in data modeling stage of SDLC is important to eliminate data anomalies using functional dependencies of attributes. Normalization is a formalized procedure in the database design theory and practice. Recently in OLAP systems, new types of data dependencies are discovered - Conditional Functional Dependencies (CFDs) and Association Rules (ARs) (Allard et al., 2010). However, data base design is only one of stages of IS development life cycle.

Normalization and modularity analysis are applied to some aspects of the real world domain modeling, e.g. normalization of the enterprise processes (workflows in Pankratius et al. (2005) and health care processes (Murray et al., 2010), functional features (Osis, 2004), and domain ontologies (Rector, 2003), (Özacar et al., 2011). These approaches are relevant to applying normalization for the domain modeling stage of SDLC. Normalization of the workflow modeling starts with pre-normalization step when the workflow is represented as a Petri net. Pre-normalization of source model simplifies transformations into normal forms (1NF, 2NF, and 3NF). Normalization is based on the dependencies of components (places or transitions) (Pankratius and Stucky, 2005).

Model-driven domain analysis in the context of software development in (Osis, 2004) is an example of the internal modeling for the software engineering needs. Functioning Cycle - a closed path of cause-and-effect relations among functional features is a key construct of domain modeling approach in (Osis, 2004). The purpose of the normalization process in the area of ontological modeling is the modularization of the domain related concepts, focused on the creation of the concept structures (generalization or aggregation hierarchies) and is based on empirical criteria and experience (Özacar et al., 2011), (Rector, 2003).

The article presents a generalized view to the *normalization* from the perspective of the internal modeling paradigm. The internal modeling paradigm is a prerequisite for discovering causal dependencies within the real world domain (i.e. enterprise). The transferring of the perceived causal dependencies, which are essential in the particular real world domain to all the subsequent SDLC stages, starting with enterprise/business process modeling stage, is the main condition of the knowledge-based development. The captured domain information (i.e. CIM level model or business process model) can be transformed using normalization rules, which aimed to the identification of the real world domain causal dependencies, conceptualization and transferring across the rest stages of SDLC without loss of essential information. The presented systematic approach for knowledge-based software engineering framed by the normalized SDLC (Gudas, 2012, 2016).

The goal of paper is to motivate the enterprise modeling normalization procedure and to define enterprise model normal forms, which are based on the discovered causal dependencies within the real world domain (i.e. enterprise). Normalization of the enterprise model (EM) is considered as a knowledge-driven transformation of the already developed business process models into an enterprise management model (EMM).

The structure of the paper is as follows. In section two the assumptions of the approach towards normalization of IS development are presented, and the principal scheme of the IS normalized development life cycle LC is discussed.

Section 3 is the analysis of the normalization and functional dependency understanding in diverse subject domains. The key concepts of the enterprise internal modeling approach defined in Section 4: functional dependency, business function, management functional dependency, management transaction (MT), Detailed Value Chain Model (DVCM), and Elementary Management Cycle (EMC). Section 5 includes a description of the normalization steps and definitions of the Enterprise model Normal Forms (ENF). Finally, conclusions summarize the presented approach towards normalization of the knowledge-based software development.

2. The premises of the IS development normalization

Our approach towards normalization of domain modeling in the information systems development is based on the assumptions as follows:

IS development is perceived in the context of the internal modeling paradigm, i.e. organizational system or enterprise is considered as a white box. The internal modeling is aimed to the identification of *a deep structure* and *essential patterns of behavior* (*laws)* of the subject domain (Gudas, 2012), (Dietz, 2006). Analysis of “normalization” and “functional dependency” concepts is accomplished from the internal modeling perspective. As of assumption 1, an internal modeling paradigm is applied for IS subject domain modeling, i.e. for enterprise modeling (Gudas et al., 2016), (Gudas, 2016). Internal modeling of enterprise domain focuses on deep properties of the enterprise management activities, pursue to reveal the content of the causal dependencies and transactions (functional dependencies, data/knowledge dependencies). "The deep structure of an information system comprises those properties that manifest the meaning of the real-world system the information system is intended to model." (Wand, Weber, 1995). The internal modeling of business enterprise considers an enterprise as a goal-driven complex system (a self-managed system) with predefined meta-structure of activities, and predefined types of internal transactions, obligatory to manage and control enterprise (Gudas et al., 2016).

Functional dependency (FD) is a key property of a real world domain from the internal modeling perspective, i.e. the functional dependencies between elements (components, entities, attributes, values, etc.) of domain models are conceptualization of the real world causal dependencies, and reveals deep knowledge of mutual influences, causal links between elements of the subject domain (Wand, Weber, 1995). For instance, functional dependencies between attributes of entities in data model capture the meaning of a real world domain dependency as perceived by the analyst (data base designer). Empirically some functional dependency could be discovered, captured and identified, or missed in some particular domain model due to mistake or due to the objectives of modeling. To systemic result, the f*unctional dependency (FD)* concept is required for discovering and representing of the domain causality.

Generally, normalization means the transformation of the initial (*empirical)* model of the domain to the “norm”-defined model (*normalized* model). Model transformation is supported by: a) knowledge of deep properties (causality) of a domain, b) knowledge of "norm" (criterions and procedure of normalization).

The transaction is a key concept for discovering of deep properties (causality) of the subject domain. The transaction is an essential concept in enterprise architecture on different layers: business strategy layer, business process layer, business process/enterprise layer, application layer, and software components layer. The content of transactions on the EA layers is different; it corresponds to the viewpoint (semantics) of the definite layer. On business management layer in enterprises, there are several interpretations for transactions; however, in business management frameworks a transaction (e.g. Action workflow approach, Deming's PDCA cycle, transactional workflows) is a closed loop sequence of goal-driven activities (i.e. value oriented transactions) as in (Medina-Mora et al., 1992), (Deming, 1993), (Porter, 1985), (Georgakopoulos et al., 1995), (Rummler et al., 2010). On business process layer, the enterprise transaction in (Dietz, 2006), (Papazoglou, 2003) or the management transaction (MT) in (Gudas et al., 2016), (Gudas, 2016) is a single indivisible logical unit of work (however, it is a complex process) comprising a closed loop sequence of information transformation steps. A primary reason of MT emerging is a management functional dependency (MFD); it causes collaboration of activities that are needed for achieving some enterprise goal (Gudas, 2012). MT is a closed loop sequence of goal-driven information transformations (comprising a management function Fj) focused on the control of enterprise process Pi (Gudas, 2012, 2016). On application layer, the transaction is defined a closed loop sequence of information exchange that is treated as a unit for the purposes of satisfying a request.

The essence of this premise is that a conceptual structure of the transactions in all EA layers or SDLC stages are the same – transaction is a single indivisible logical unit of work, transaction is a cyclic process, transaction is a closed loop sequence of steps (processes, actions, activities, works, transformations, procedures, and other) (Gudas et al., 2016). The semantics of transaction (and internal elements) correspond to the viewpoint (semantics) of the definite EA layer or SDLC stage. The conceptual structure of the generalized transaction corresponds to the conceptual structure of the control system with the feedback loop (fig.1):

$$T\ (Q) = \{(S_1,\ldots, Sn), (M_1,\ldots, Mn), Rs, Feedback\} \qquad (1)$$

Here: T- transaction, a single indivisible logical unit; Q – goal, objective, criteria, requirement, rule, etc. (depends on the layer or stage); Si - process, activity, information transformation, application, procedure, and other (depends on the layer or stage); Mj – flow, message, and other (depends on the layer or stage), Rs – a sequence relationship; Feedback – a constraint, it is necessary to establish a closed loop of S, and in this way to create a single unit.

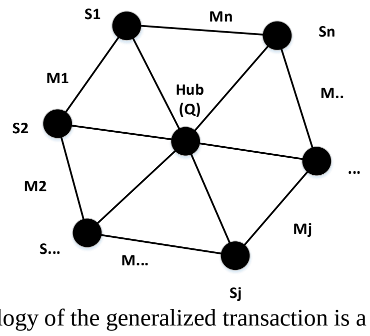


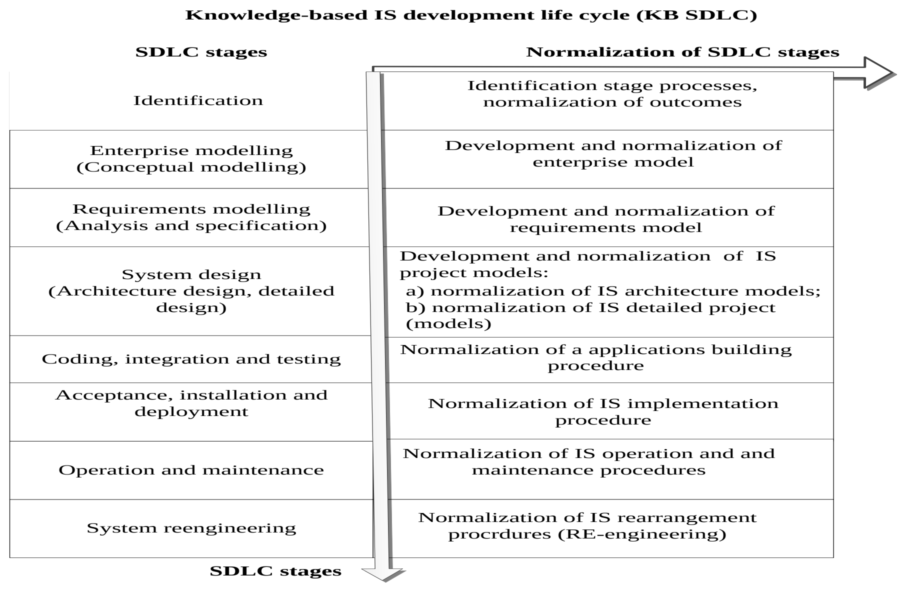
Fig. 1. Topology of the generalized transaction is a wheel graph

.

The topology of the generalized transaction is a wheel graph (fig. 1). In the graph theory, a wheel graph is obtained from a cycle graph C_{n-1} by adding a new vertex called *a Hub* that is connected to all the vertices of cycle graph C_n (Bondy et al., 2008):

A notable disadvantage of the model-driven development is that usage of concepts *normalization*, and *functional dependency* in the IS engineering (i.e. enterprise software engineering) is limited to only one stage of SDLC - the database design stage. The provided research of these two concepts reveals an idea to normalization of the entire IS development life cycle (SDLC). Based on our analysis, we believe that *a transaction* is the appropriate concept for normalized representation of processes on EA layers and stages of SDLC, which expresses the essential management and control requirements and restrictions.

The normalized IS development life cycle (SDLC) was introduced in (Gudas, 2012). A prototype is the two dimensional RUP model with pre-defined standard phases (Inception, Elaboration, Construction, Transition) on every stage of SDLC. Currently, normalization in the software development is limited to only to single stage of SDLC - the database design stage. Theoretically reasonable to consider that the functional dependencies (i.e. perceived causal dependencies) of the particular real world domain) ought to be explored on all stages of SDLC, not limited only to the database design.


Knowledge-based IS development life cycle (KB SDLC)

SDLC stages	Normalization of SDLC stages
Identification	Identification stage processes, normalization of outcomes
Enterprise modelling (Conceptual modelling)	Development and normalization of enterprise model
Requirements modelling (Analysis and specification)	Development and normalization of requirements model
System design (Architecture design, detailed design)	Development and normalization of IS project models: a) normalization of IS architecture models; b) normalization of IS detailed project (models)
Coding, integration and testing	Normalization of a applications building procedure
Acceptance, installation and deployment	Normalization of IS implementation procedure
Operation and maintenance	Normalization of IS operation and and maintenance procedures
System reengineering	Normalization of IS rearrangement procrdures (RE-engineering)

Fig. 2. The principle scheme of the IS normalized development LC

Thus, our premise for defining of the normalized information system development life cycle (in fig. 2) is assumption as follows: the model-driven development is the transferring of the functional dependencies (FDs), which is essential in the target domain, to all the subsequent SDLC stages, starting from the enterprise management modeling (business process modeling) stage (Gudas, 2012). Also, along with already known and used *data FD* the new types of functional dependencies are defined as follows: Functional dependency of the enterprise management activities - management

FD (MFD); Functional dependency of the requirements specification components (Requirements FD); Functional dependency of the system architecture components (System architecture components FD); Functional dependency of the data model components (Data FD); Functional dependency of the software system components (Applications FD).

The normalized SDLC is a two-dimensional model of the IS engineering process:

$$\text{NSDLC} = (\text{LC stage, Normalization}) \quad (1)$$

Normalization is considered here as a knowledge-based transformation when a content of every stage of IS DLC is reconstructed using functional management dependency (MFD) related criterions and by purpose and content of the particular SDLC stage. A key concept to understand normalization as knowledge transformation process is a functional management dependency (MFD), which is introduced in the enterprise management modeling theory (Gudas, 2012, 2016).

3. Normalization and functional dependency in diverse subject domains

The significance of normalization in the enterprise IS (enterprise software) engineering is not perceived yet. Understanding of normalization in distinct subject domains (statistics, mathematics, ontological modeling, data base design, etc.) is divergent, yet it has some generic features. In relational database theory, "normalization" is closely related with the concept "functional dependency" (Codd, 1971), (Kent, 1983). A functional dependency is a key concept in normalization and data integrity constraints analysis (Rissanen, 1977), (Carlson et al., 1982). However normalization is not well defined in the context of entire IS development life cycle, except the data base design stage.

Understanding of normalization and functional dependency concepts is systematized in this section by summarizing diverse subject domains.

3.1. Normalization

Understanding of *normalization* and *required knowledge for normalization* in diverse subject domains briefly:

- *In statistics*, one of the definitions of normalization is adjusting values measured on different scales to a notionally common scale. A subject domain is a real world - values of attributes of real world processes or objects. *Required knowledge: statistics normalization method, Subject of normalization: measured values (data). Criterions of transformation: dependencies of values.*

- *In sociology*, normalization involves the construction of an idealized norm of conduct (Taylor, 2009). *A subject domain is a real world: society, social behavior.* **.** *Required knowledge: essential properties of the domain objects/processes; and normalization method. The subject of normalization: a content of behavior (social processes). Criterions of transformation: dependencies of properties.*

- *In mathematical logic and theoretical computer science,* one of the definitions of normalization consider the transformation of a system to an irreducible term (a normal form) (Baader, Nipkow, 1999)). *Subject domain: an abstract world (abstract content) as a set of objects and their (essential) properties. Required knowledge: (essential) properties of objects and a normalization method (rules applied to transform objects). The subject of normalization: a content of abstract structures (structure of models). Criterions of transformation: dependencies of properties.*

- *In data modeling,* (typically associated with signal processing) data normalization is the process of reducing data to its canonical form. Data can be normalized to provide a limited range of values within a norm. *Subject domain: data (various types of data). Required knowledge: a set of data values and normalization method. The subject of normalization: relations of data items (structure of model). Criterions of transformation: dependencies of properties.*

- *In ontology modeling*, normalization of large ontologies (the domain level ontologies) is considered as a decomposing the ontology into independent disjoint skeleton taxonomies restricted to be simple trees. Normalization is required to achieve explicitness and modularity in the domain. The purpose of the normalization of an ontological model is the modularization of the domain related concepts, focused on the creation of the concept structures (generalization or aggregation hierarchies) and is based on empirical criteria and experience (Rector, 2003). *Subject domain: real world domain. Required knowledge: important objects and their properties; normalization method. The subject of normalization: a content of domain (structure of models). Criterions of transformation: ontological dependencies of properties.*

- *In relational database design,* normalization is a sequence of data model transformation steps and is defined by Normal Forms (1NF, 2NF, 3NF, etc.) (Codd, 1971), (Kent, 1983). We draw attention to the reciprocal relationship between the data model and the real world. Changed functional dependencies (different specifications of the data model) between the same data (attributes of entities) "generate" other potentially possible (permissible) physical situations in the real world. Such an interconnection is important point of normalization - understanding (and revealing) the causal relationships of the RW domain justifies (determines) the permissible interfaces between the model elements. *Subject domain: data sets (semantic data) and data values. Required knowledge: functional data dependencies, and normalization method; Subject of normalization: relationships of attributes (structure of data model). Criterions of transformation: functional dependencies of attributes.*

- In workflow modeling (WFM), normalization is defined by three normal forms of WFM in (Pankratius, Stucky, 2005). WFM normal forms are defined for workflow models represented as a Petri net after pre-normalization step. Pre-normalization simplifies transformations into normal forms. The 1NF is intended to flatten out hidden sub-workflow specifications to atomic components; it is aimed at the identification of redundant flow specifications. The second normal form (2NF) is intended to eliminate multiple occurrences of isomorphic sub-nets with identically labeled components (places or transitions). The third normal form (3NF) is aimed to eliminate multiple occurrences of isomorphic sub-nets with semantically identical labels for places or transitions. *Subject domain: real world domain processes and flows. Required knowledge: dependencies of components (places or transitions), and normalization method. The subject of normalization: the content of domain (structure of WFM). Criterions of transformation: dependencies of components.*

3.2. Functional dependency

Normalization and functional dependency are related concepts, i.e. in data base design normal forms are defined regarding functional dependencies. Functional dependency (FD) is a property of a real world, i.e. the functional dependencies between elements (components, entities, attributes, values, etc.) of subject domain models reveals causal links between elements in the real world domain (Wand, Weber, 1995).

A summary of a *functional dependency* (FD) characterization in diverse subject domains from the perspective of required domain knowledge is as follows:

- *In mathematics***,** two variables x and y are tied by a functional dependence, if for each value of one of them it is possible to receive by the certain rule one or some values of another. *Subject domain: a set of variables. Required knowledge: a deep understanding of subject domain entities (objects), their attributes (variables) and relations;*

- *In ontological modeling* analysis of the existential dependence of **c**omposite objects defines existential dependence (EDG) as follows: composite objects are existentially dependent objects in the sense of (EDG) since they require the existence of proper parts (OD_SEF, 2015). *Subject domain: a set of composite objects. Required knowledge: a deep understanding of subject domain entities (composite objects) and relations);*

- *In enterprise modeling* functional dependency of enterprise management activities (for instance, finance management, human resource management, procurement, etc.) is the sequence of required essential information interactions between internal components of definite activity (within management activity steps), that are required for implementation of that particular enterprise management activity (i.e. enterprise management function) (Owens, 2013), (Gudas, 2012). Subject domain: a system of enterprise management activities: enterprise management functions and enterprise processes. *Required knowledge: a deep understanding of causal relationships between activities, understanding of internal informational dependencies of activities, understanding of obligatory steps within each activity (enterprise management function and enterprise process).*

- *In relational data base design,* a functional dependency defines a functional relationship between attributes. A set of attributes X in relation R is said to functionally determine another set of attributes Y, also in R, (written X →Y) if, and only if, each X value is associated with precisely one Y value; R is then said to satisfy the functional dependency X → Y. Second and third normal forms are defined regarding functional dependencies. In relational database design, the several equivalent axiomatizations of FDs are given by Armstrong (Armstrong, 1974). For instance, the one definition of FD is based on properties of reflexivity ($X \rightarrow X$); augmentation (if $X \rightarrow Z$ then $X + Y \rightarrow Z$) and pseudo transitivity (If $X \rightarrow Y$ and $Y + Z \rightarrow W$ then $X + Z \rightarrow W$). *Subject domain: real world domain, entities, and their attributes. Required knowledge: a deep understanding of subject domain entities and essential relations.*

Summing up, normalization procedure exploits the deep knowledge of domain: first, the normalization procedure is based on the functional dependencies of the target domain, in other words, normalization requires revealing causation within the target domain. The content of *normalization* and *functional dependency* (FD) concepts in the particular subject domain is determined by knowledge of deep structure and dynamics of target domain – is based on the perceived causal dependencies (laws of behavior). So, the deep knowledge of the subject domain is a knowledge of *the laws of behavior* (e.g. knowledge of the obligatory technological links, or management and control work sequence) within the domain. Nevertheless, do not forget, that correlation does not imply causation.

Generalization of understanding of normalization and functional dependency is aimed to the deployment of these concepts in the information systems engineering (ISE). "The deep structure of an information system comprises those properties that manifest the meaning of the real-world system the information system is intended to model." (Wand, Weber, 1995).

The approach towards the generalization of the normalization definition in the IS engineering is based on the premises as follows:

1. Analysis of “normalization” and “functional dependency” concepts should be accomplished in the context of internal modeling paradigm when a subject domain (i.e. organizational system or enterprise) is real world domain considered as a white-box. The target of internal modeling is an identification of a deep structure and essential patterns of behavior (laws) of the subject domain (Gudas, 2012), (Dietz, 2006).

2. Functional dependency (FD) is a property of a real world, i.e. the functional dependencies between elements of the ISE models (components, entities, attributes, values, etc.) reveal (correspond, correlate) causal dependencies between elements of a subject domain (Wand, Weber, 1995).

For instance, functional dependencies between attributes of entities in data model capture the meaning of an application domain as perceived by an analyst (data base designer). In common, functional dependency could be revealed and identified. However, it could be missed in case of an analyst mistake or due to objectives of modeling.

3. Generally, normalization is transformation of the initial model (an empirical model) of the subject domain to the “norm”-defined model (a normalized model), and is based on two kinds of knowledge: a) Knowledge of deep properties of some subject domain, b) Knowledge about “norm” development (knowledge about criterions and procedure of normalization).

4. Normalization in the enterprise domain modeling

Summarizing, the concept of functional dependency expresses the deep properties of the subject domain being examined. Generalization of the concept “functional dependency” in the IS engineering is aimed to overcome_limitations of subject domain systems analysis (business process modeling), requirements specification and enterprise software design methods. Limitations of model-driven development (MDD) are related to the model transformation gaps between SDLC stages: business modeling, requirements specifications, and software design models (IS a project model).

4.1. Deep knowledge of the subject domain

Identification of a functional dependency in particular subject domain requires definite knowledge about internal interactions within that particular domain (the type of systems). Identification of functional dependency requires revealing the interactions of the domain elements (objects, processes, entities) and inter-dependencies of their properties (variables, attributes). Therefore, identification of the functional dependency in the particular subject domain is a knowledge intensive process; consequently, the internal modeling approach is urgent. For example, normalization of data model could be considered as the knowledge-driven transformation, because the predefined knowledge of the data functional dependencies in that subject domain being examined.

Understanding of the functional dependency in the enterprise management modeling is closely related to the understanding of business management transaction and business function (Gudas, 2016). We accept the proposition of J. Owen "Processes steps that are not business functions have no logical foundation or integrity." and *"Business functions are the core activities of an enterprise. All other activities and data are derived from business functions"* in (Owen, 2013) about the fundamental importance of understanding business functions.

However, we are focused on the modeling of business functions on the more detailed level. The aim is revealing of information-data-knowledge transformations within the management activities. Our approach towards enterprise model normalization is based on the premises about two kinds of required knowledge: knowledge about the deep structure of the subject domain, and knowledge about subject domain dynamics based on the deep causal dependencies.

In the enterprise internal modeling approach (Gudas, 2012), the concept of management functional dependency (MFD) for capturing causal dependencies of the business management activities have been introduced. Management Functional Dependency (MFD) is a primary causal dependency of business activities required by strategic plans or operational capabilities. MFD is captured by domain analyst and represented on the enterprise modeling layer as the *management transaction (MT).* Every MT should identify two parts of management activity - definite management function (F) and definite enterprise process (P), and the information interactions (K) between these two parts of management activity (Gudas, 2012).

4.2. Knowledge of subject domain dynamics

The first step of enterprise management modeling is discovering of MFD within the problem domain and conceptual representation of MFD by the management transaction (MT) (see fig. 3). Finally, a problem domain is considered as the Detailed Value Chain Model (see fig. 3), comprising a system of the management transactions (Gudas, 2012). The Detailed VCM (DVCM) refines the causal dependencies of business activities and represents the informational content of each MFD as the management transaction (MT). The definite MTji = ((Fj x Pi), Ki, Kj) includes the management function Fj and enterprise process Pi, and feedback control flows Ki and Kj between Fj and Pi. Here Ki is a flow of state attributes I, Kj – a flow of controls j (see fig. 4).

The next step is exposing the deep knowledge of target domain – an internal structure of the MTs is obtained. The internal structure of MT by definition is the Elementary Management Cycle (EMC) (see Fig. 4).

An example of concrete MT: ***MT ((Order fulfillment) = ((Fj – Order fulfillment management) x (Pi – Build and ship product), Ki – Orders received, Kj - Product shipment invoice).***

The next step is internal modeling of the identified management transactions. Herein a deep structure of management transaction MTji is defined as the elementary management cycle (EMCji). The EMCji includes the enterprise management goal (G), enterprise process Pi(G), and the management function Fj(G) with a predefined internal structure as follows: the information transformation steps T(G) = (T_1, ..., T_n), the management information flows K(G) = (K_A,..., K_V) between the steps T(G), and a set of influences S(G) of the management goal (G) focused on the process Pi(G), steps T(G), and flows K(G) (Fig.4). The elementary management cycle (EMCji) is defined as follows:

EMCji = (G, Pi(G), Fj(T(G), K(G), S(G))); (3)

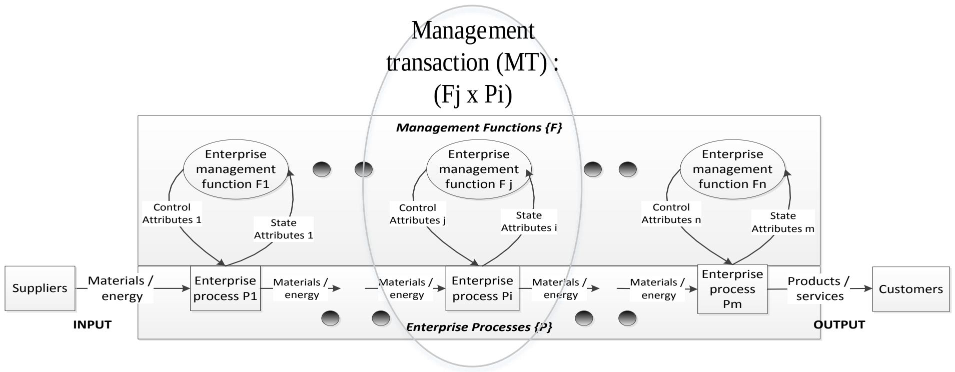


Fig. 3. Enterprise domain is represented as Detailed Value Chain Model (DVCM); it includes the captured management transactions (MT)

The concept "management information" comprises all types of information flows used in management interactions, i.e. it includes data, information, goals, rules, directives, constraints, etc. The management information flows S show an impact of goal (G) on the EMC steps T and include the rules and directives for identification (or modification) of the content of steps T (sub-functions of the management function F), i.e. the logic of information transformations within the EMC steps (T) is goal-dependent. Management information (S) directed from goal G to flows K defines the rules and directives for identifying (or modifying) of the content of flows K.

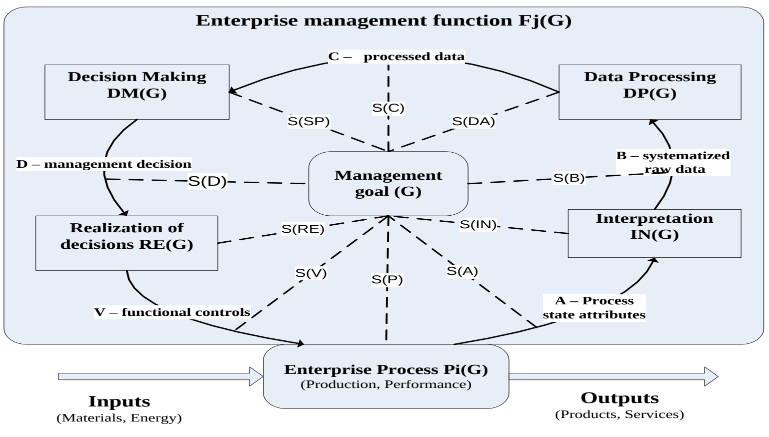


Fig. 4. The internal structure of management transaction (Fj x Pi) is the Elementary Management Cycle (EMCji)

Herein a deep structure of management transaction MTji is adopted for needs of IS engineering, and is defined in Fig. 4 as the Elementary Management Cycle (EMCji) - the internal steps of EMC are limited to four, and the semantics is defined concretely (Gudas et al., 2005), (Gudas, 2012).

Four types of management information transformation steps identified in EMC: IN – interpretation (data gathering and systematization), DP – data processing, DM – decision making, RE – realization of decisions (controls), impact to process Pi. Management information flows between steps are classified as follows: A – state attributes of Enterprise Process Pi(G), B – systematized data, C – processed data, D – management decisions, V – functional controls for Enterprise Process Pi(G):

EMC = (G, P(G), F(IN(G), DP(G), DM(G), RE(G), A(G), B(G), C(G),D(G), V(G), S(G))) (3)

The Elementary Management Cycle (EMC) is considered as the typical unit of enterprise management from the information point of view. Interactions between the enterprise process Pi and steps (IN, DP, DM, and RE) of the enterprise management function Fj are an illustration of the concept "management functional dependency" (MFD).

Management Functional Dependency (MFD) is primary in the sense that MFD is a reason (cause) of some particular management transaction. Enterprise management functional dependency (MFD), is identified at the top level of domain modeling as management transaction (MT) (see fig. 3) and is defined here in detail as EMC (see Fig. 4). From the viewpoint of IS, engineering needs the set of the internal steps of management transaction MTji is limited here to five semantically different FDs as presented in the Fig. 4 (Gudas, 2012). In our case MFD consists of a consistent series of functional dependencies (FD) that are conceptualized as EMC elements:

MFD = {FD1, FD2, FD3, FD4, FD5} (4)

Here:

FD1 = (A → IN(G) → B) – interpretation step IN is conceptualization of functional dependency of A – attributes of the enterprise process (technological process) state, and B – output attributes of interpretation;

FD2 = (B → DA(G) → C) – data processing step DA is conceptualization of functional dependency of B, and C – data processing step output (IN output B is the DP input C);

FD3 = (C → SP(G) → D) – decision making step DM is conceptualization of functional dependency of C, and D – decision making output (DP output C is the DM input D);

FD4 = (D → RE(G) → V) – decision implementation step RE is conceptualization of functional dependency of D, and V - decision implementation output (DM output D is the RE input V);

FD5 = (V → P(G) → A) – enterprise process (technological process) implementation step is a conceptualization of the functional dependency of V, and A – enterprise process state attributes (RE output V is the controlling impact on the enterprise (technological) process).

Thus, in our case enterprise management functional dependency (MFD) consists of functional dependencies FD1, FD2, FD3, FD4 and FD5 which make a closed loop:

MFD = {FD1 = (Si → IN(G) → Sj); FD2 = (Sj → DA(G) → Sn); FD3 = (Sn → SP(G) → Sm); FD4 = (Sm → RE(G) → Sk); FD5 = (Sk → P(G) → Si)} (5)

The MFD concept is used when some subject domain is perceived as a set of self-managed goal-driven activities, which are conceptualized as management transactions (see fig. 3). The presented example of *MFD(Order fulfillment)* revealed the perceived causal dependencies between real world activities which should be created (organized) and managed within the manufacturing enterprise:

MFD(Order fulfillment) = (FD1 = (A(Orders received)→ IN(Complete order) → B(Verified orders), FD2 = (B(Verified orders) → DP(Submit order) → C(Credit requests), FD3 = (C(Credit requests) → DM(Check credit) → D(Approved orders), FD4= (D(Approved orders) → RE(Scheduled orders) → V(Scheduled orders), FD5 = (V(Scheduled orders) → Pi (Build and ship product) → (A (A(Orders received)))) (6)

Notice, that *MT(Order fulfillment)* in the text above is a conceptualization of the perceived domain causality *MFD(Order fulfillment).*

5. Normalization of enterprise model

An enterprise model is a set of business process models acquired by analysts and experts in the initial SDLC stage of conceptual domain modeling. Traditionally business process models (BPMs) are constructed by few analysts in the context of black-box approach (external modeling paradigm) as systems of (input, process, output) components. Commonly, business process modelers use BPMN, ARIS or some other BPM language and the following elements: activities (*processes)* related using information and material *flows* and, at times, by pointing out their relationships with *events* or *organizational sub-units*. The, therefore, content of different BPMs could be overlapping to some extent, also could be some gaps of connectivity between BPMs. That is way initially emerged a set of business process models and, in sum, enterprise model is an empirical model (a set of empirical business process models).

5.1. Normalization steps

Normalization of EM is a sequence of transformations of an empirical enterprise model (EM) to get an enterprise management model (EMM). In our approach normalization procedure of EM is knowledge-driven transformation, which is based on the two predefined knowledge structures: a management transaction (MT) and an elementary management cycle (EMC). The main normalization steps of enterprise modeling are depicted in figure 5.

Definition. *Normalization of the enterprise model (EM) is a knowledge-driven transformation of the acquired empirical information (perceived domain knowledge or already developed business process models) into an enterprise management model (EMM). EM normalization procedure is based on the internal modeling of the management functional dependencies (MFDs) perceived within enterprise domain.*

The aim of enterprise modeling normal forms is to ensure the domain causality representation by the enterprise model (regardless of the business process modeling notation). We notice that BPMN was not selected for conceptual representation because it is not convenient for depicting cyclical processes (e.g. such as MT and EMC).

Can be two primary sources of the domain knowledge (empirical information): a perceived domain composition by observation, or already formed before empirical enterprise/business process models (fig. 5).

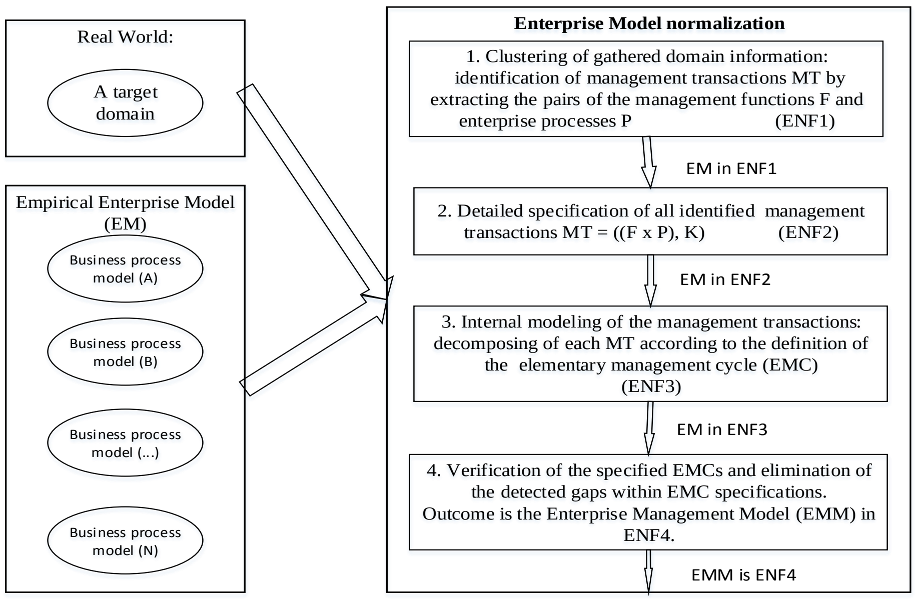

Real World:
A target domain
Empirical Enterprise Model (EM)
Business process model (A)
Business process model (B)
Business process model (...)
Business process model (N)
Enterprise Model normalization
1. Clustering of gathered domain information: identification of management transactions MT by extracting the pairs of the management functions F and enterprise processes P (ENF1)
EM in ENF1
2. Detailed specification of all identified management transactions MT = ((F x P), K) (ENF2)
EM in ENF2
3. Internal modeling of the management transactions: decomposing of each MT according to the definition of the elementary management cycle (EMC) (ENF3)
EM in ENF3
4. Verification of the specified EMCs and elimination of the detected gaps within EMC specifications. Outcome is the Enterprise Management Model (EMM) in ENF4.
EMM is ENF4

Fig. 5. Normalization of the enterprise modeling

The normalization steps of enterprise model are as follows (Fig. 5):

1. Transformation of empirical information (the perceived elements of the target domain or elements of already formed before empirical enterprise models) into pre-normalized form (ENF1, and ENF2), defined as Detailed Value Chain model (DVCM) in our approach (steps 1 and 2 in Fig. 5):

Clustering of the gathered empirical domain information by dividing the elements "activities" into two types "enterprise management function F", and "enterprise process P", and extracting pairs (F x P) of F and P. In this way the management transactions MT start identifying.

A closed loop within all identified management transactions MT is specified, i.e. the bi-directional information flows between enterprise management functions F and a corresponding enterprise processes P is identified. At this moment an initial enterprise model in two steps is transformed and depicted as a Detailed Value Chain Model (Gudas, 2012).

2. Development of the internal model of management transactions MT. The structure of the internal model of MT depends on the problem being addressed; it is defined here as an Elementary Management Cycle (EMC) (see step 3 in Fig. 5):

Decomposition of management transactions MT by the definition of EMC into a definite set of lower level components. Thus, each MT specified on the lower level is a closed loop of EMC steps and connecting information flows, which are influenced by management goal (see fig. 4).

3. Revision of results (step 4 in Fig. 5): correction of the detected gaps within EMCs and overlapping of different EMCs. Finally, a normalized enterprise model – enterprise management model (EMM) is obtained as a set of verified EMCs.

5.2. Definitions of the Enterprise model Normal Forms (ENF)

Definition: *An enterprise model (or business process model) is in the first normal form (ENF1) if all management transactions are identified (i.e. the pairs of management functions (F) and enterprise processes (P) are identified).*

The captured domain information (see fig. 5) is overlooked and clustered according to the definition of the management transaction MT. All elements denoting "domain activities" (i.e. processes) are divided into the two types (Fig. 3): P – the enterprise processes (material transformations) and F - the enterprise management functions (information/data/knowledge transformations). The pairs of interacting enterprise activities (F x P) are recognized and depicted in the model. This results in ENF1 of the enterprise model.

This is the first step of reconstruction of the empirical EM into the Detailed VCM (fig. 3), which by definition meets the second normal form (ENF2) of the enterprise model.

Definition: *An enterprise model (or business process model) is in the second normal form (ENF2) if all management transactions {(F x P)} between enterprise processes (P) and enterprise management functions (F) are specified, i.e. essential management functional dependencies (MFDs) are conceptualized.*

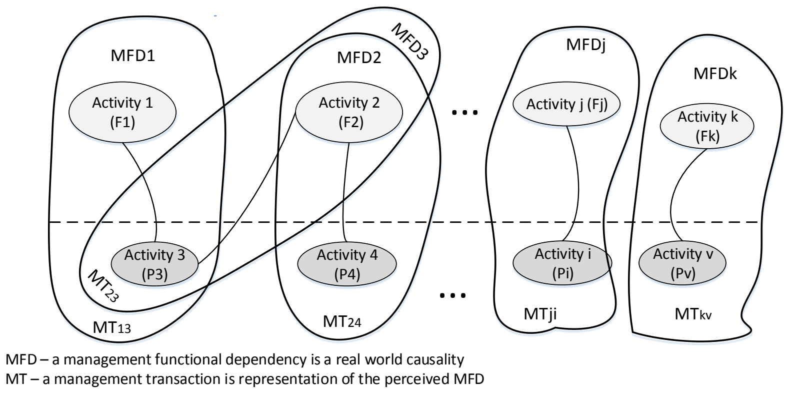


Fig. 6. An abstract enterprise model after clustering meets ENF1 – the pairs of two types of interacting enterprise activities (F x P) have been recognized in the target domain

This is a final step of transformation of the empirical *(clustered)* EM into a Detailed Value Chain Model - essential MFD are perceived in the target domain and represented as management transactions {(F x P)} (Fig. 3).

Ending normalization in the ENF2 you have to finish specification of all identified MT by naming a feedback loop information flows between activity types F and P of management transactions {(F x P)} as depicted in fig. 3. I do not provide a detailed specific example of the limited scope of the article.

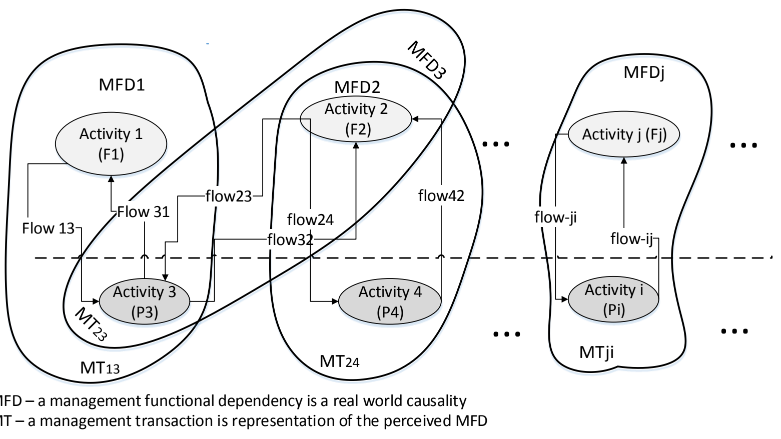


Fig. 7. Illustration of ENF2: all identified management transactions (MTs) specified as the closed loop systems MT_{13}, MT_{23}, MT_{24},.., MTji,…MTkv.

An example of enterprise model in ENF2 is presented in fig.7. All identified in ENF1 management transactions are specified as a closed loop interaction of F and P: MT_{13}= ((P3, F1), Flow31, Flow13); MT_{23}= ((P3, F2), Flow32, Flow23); MT_{24}= ((P4, F2), Flow42, Flow24); MTji= (Pi, Fj), Flow-ij, Flow-ji),…

Let us say that concrete example of MT_{13} is ***MT(Order fulfillment)*** defined in the text above.

Definition: *An enterprise model (or business process model) is in the third normal form (ENF3) if a deep structure of all management transactions MT is specified as the Elementary Management Cycle (EMC).*

The internal structure of each management transaction MTji = {(Fj x Pi), Ki, Kj} is specified as EMC: components of management transactions (activities and flows) are classified into types by expertly attributing them to a corresponding EMC step (IN, DP, DM or RE) and corresponding type of EMC flow (A, B, C, D). Consequently, management transactions MTji are normalized according to EMC structure (see Fig. 4).

An example of ENF3 of management transaction MT_{13} (see Fig. 7) of the abstract target domain is presented in Fig. 8. The internal elements of the Activity1 (F1) have been found and identified as follows: a goal of management G1, activities Activity11 (F11), Activity 12 (F12), Activity13 (F13), and Activity14 (F14), and new flows Flow-b, Flow-c, Flow-d, Flow-e, as well identified impacts S (A) of goal G1 to internal elements of $EMC_{13.}$

Let us say that concrete example of MT_{13} is *MT(Order fulfillment)* defined in the text above. The internal structure of MT_{13} can be determined correctly if the structure of the *MFD (Order fulfillment)* is already known. Suppose the analyst do not have a prior knowledge of *MFD (Order fulfillment)* and did not notice within target domain the business activity *Check credit* and flow *Credit requests,* so the internal model of MT_{13} on this stage of normalization got incomplete:

EMC(Order fulfillment) = (G1(Quality of service); P3(Build and ship product); F1(Order fulfillment management)= (IN(Complete orders), DP(Submit orders), *RE(Schedule orders); A(Orders received), B(Verified orders), D(Approved orders), E(Product shipment invoice); S(G1)).*

By this method, an internal structure of all management transactions MT13, MT23, MT24,..,MTji,…MTkv must be captured and specified as the elementary management cycles (EMC) to obtain ENF3 of the initial enterprise model.

Definition: *An enterprise model (or business process model) is in the fourth normal form (ENF4) if verification and revision of enterprise management functions structure are performed and all gaps or overlapping of EMCs are eliminated.*

Enterprise models (in ENF3) are analyzed and corrected: EMC "gaps" are identified, which stands for identification of certain cases of mismatch to the theoretical EMC structure. Carrying out an additional analysis of the enterprise domain, missing elements of the definite EMC (enterprise management function) are uploaded to model (fig. 9).

Afterward, taking the hierarchical structure of the enterprise management model into consideration (considering that components of EMC are complex and comprise hierarchical structures), the normalization procedure can be applied to the EMC steps (IN, DP, DM, RE).

The abstract management transaction (corresponding to MFD1 in the Fig. 7) is specified as the EMC1 = (IN, DP, DM, RE, G, P) in the Fig. 8. This is an example of enterprise model in the ENF3, but some gaps in this particular management transaction model (a part of Enterprise management model) remains in this step of normalization procedure (see Fig. 8) because of the incompleteness of the initial empirical EM provided in the Fig. 6.

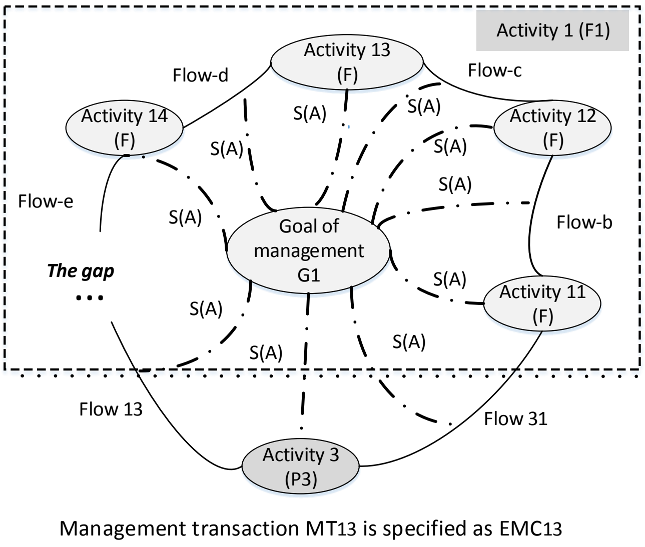


Fig. 8. An example of ENF3.: the management transaction MT_{13} have been specified as the EMC_{13}

The internal model EMC_{13} of management transaction MT_{13} in fig. 9 have been verified and corrected than the gaps are eliminated (new activities Activity, X and Activity Z, have been added), and identifiers are assigned to newly added flows (Flow-e, Flow-w, S(A), …, S(A)).

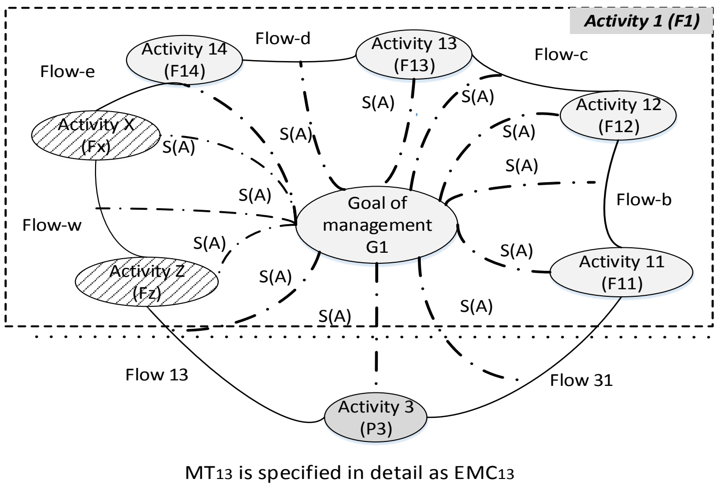


Fig. 9. An example of ENF4: the internal model EMC_{13} of management transaction MT_{13} have been verified and corrected.

Let us say that concrete example of MT_{13} is *MT(Order fulfillment.* The internal structure of MT_{13} can be determined correctly if the structure of the *MFD (Order fulfillment)* is already known. There were corrections made, and now *MT13(Order fulfillment)* internal model is in ENF4:

EMC(Order fulfillment) = (G1(Quality of service); P3(Build and ship product); F1(Order fulfillment management)= (IN(Complete orders), DP(Submit orders), DM(Check credit), RE(Schedule orders); A(Orders received), B(Verified orders), C(Credit requests), D(Approved orders), E(Product shipment invoice); S(G1).)

All internal models EMC of all specified before management transactions MT have to be verified and corrected to obtain ENF4 of the enterprise model.

Definition: *An enterprise model (or business process model) is in the fifth normal form (ENF5) if it is in normal form ENF4, and normalization of procedural components of the elementary management cycles (EMCs) is fulfilled to the normal form ENF4.*

According to the definition, the procedural elements (transformations IN, DP, DM, RE, see fig. 4) of EMCs have to be decomposed, in this way, the lower level EMCs of each transformation (IN, DP, DM, RE) are created. Next, normalization procedure should be carried out for each lower level EMC. This corresponds to the application of ENF3 and ENF4 procedures to normalize these (lower level) business process models of the transformations (IN, DP, DM, and RE).

Summarizing, normalization of the domain model applying normal forms (ENF1 – ENF5) is considered as knowledge-driven model transformation, which is based on the concepts of MFD and MT. Two domain knowledge structures - Detailed Value Chain Model (DVCM) and Elementary management cycle (EMC) are used for capturing and specification of domain knowledge. The principle scheme of the enterprise model normalization provided in figure 10. In our approach MT is defined in detail as EMC framework; however, there may be a different MT detailing.

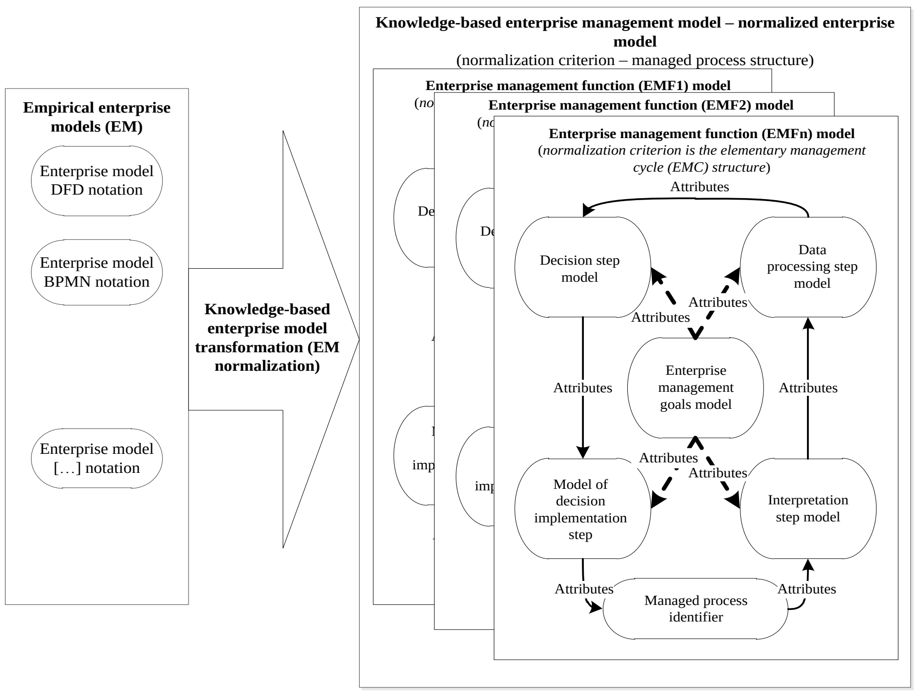

Empirical enterprise models (EM)
Enterprise model DFD notation
Enterprise model BPMN notation
Enterprise model […] notation
Knowledge-based enterprise model transformation (EM normalization)
Knowledge-based enterprise management model – normalized enterprise model
(normalization criterion – managed process structure)
Enterprise management function (EMF1) model
Enterprise management function (EMF2) model
Enterprise management function (EMFn) model
(normalization criterion is the elementary management cycle (EMC) structure)
Attributes
Decision step model
Data processing step model
Enterprise management goals model
Model of decision implementation step
Interpretation step model
Managed process identifier

Fig. 10. The principle scheme of the enterprise model normalization.

In this way, each enterprise management function F is described as EMC and expertly revised until a theoretically correct managed process (F x P) is achieved by identifying missed enterprise management components. These new (missed in empirical models) components identified by analysis of the enterprise domain or by additionally questioning the expert.

6. Conclusions

One disadvantage of the model-driven development is that usage of concepts normalization, and functional dependency in the IS engineering (i.e. enterprise software engineering) is limited to only one stage of SDLC - the database design stage. The provided research of these two concepts reveals an idea to normalization of the entire IS development life cycle (SDLC). Normalization and management functional dependency are key concepts for enhancement of the model driven development methods and technologies towards knowledge-based engineering. The purpose of the normalized IS development life cycle is given background for systematic, knowledge-based software engineering, which is based on the deep knowledge of subject domain (enterprise).

The presented approach of enterprise model normalization is a possible way for normalizing the first stages of SDLC – enterprise/business process modeling. Enterprise model normalization is a knowledge-driven transformation of the acquired set of business process models (BPM) into an enterprise management model (EMM). Enterprise model normalization based on the discovering of management functional dependencies within the problem domain, and conceptualization using MT, DVCM and EMC frameworks. Presented definitions of enterprise model normal forms (ENF1 – ENF5) are described briefly using the example of an abstract subject domain.

The knowledge-based software development should maintain the transferring of the real world functional dependencies across SDLC stages, starting with the enterprise modeling (business process modeling) stage. Based on our analysis, we believe that the management transaction expresses the essential managerial and control requirements, and restrictions, consequently is the appropriate concept for creating normalization procedures of SDLC stages.

Discovery and transferring of domain causal dependencies using such concepts as the management functional dependency, the management transaction, and normalization forms the basis for enhancement of model-driven software engineering.

Acknowledgements

This research inspired by the seminars of the Software Engineering Department of the Vilnius University Institute of Mathematics and Informatics. I would like to express my gratitude to the colleagues for the attention and remarks that allowed the development of the ideas in this paper.

References

Allard, P., Ferré, S., Ridoux, O. (2010) Discovering Functional Dependencies and Association Rules by Navigating in a Lattice of OLAP views. In: Kryszkiewicz M., Obiedkov S. (Ed.), *Conference on Concept* Lattices *and Their Applications (*CLA*)*, 199– 210.

Armstrong, W.W. (1974) Dependency structures of data base relationships. *Information Processing 74*, North-Holland Pub. Co., Amsterdam, 580– 583.

Baader, F. Nipkow, T. (1999). *Term rewriting and all that*. Cambridge University Press.

Bondy, A., Murty, U.S.R. (2008) *Graph theory*, Springer.

De Bruyn P., Van Nuffel D., Verelst J., Mannaert H. (2012) Towards Applying Normalized Systems Theory Implications to Enterprise Process Reference Models. In: Albani A., Aveiro D., Barjis J. (eds) *Advances in Enterprise Engineering VI. EEWC 2012*. Lecture Notes in Business Information Processing, vol 110. Springer, Berlin, Heidelberg.

Carlson, C.R., Arora A.K., Carlson M.M. (1982) The application of functional dependency theory to relational databases. *The Computer Journal*, Vol. 25, No. 1, 68– 73.

Chong, S., Liu, K. (2000) A Semiotic Approach for Modeling and Designing Agent-Based Information Systems Based on Roles and Norms. In: *Proceedings of AOIS-2000 at CAiSE*00,* iCue Publishing, Berlin, 552– 567.

Codd, E.F. (1971) Further normalization of the data base relational model. In: *Courant Computer Science Symposia 6': Data Base Systems*, Prentice-Hall, Englewood Cliffs, N.J.

Date, C.J. (1999) *An Introduction to Database Systems* (8th ed.), Addison-Wesley.

Deming, W.E. (1993) *The new economics for industry, government and education*. MIT Press, Boston.

Dietz, J.L. (2006). The deep structure of business processes, *Communications of the ACM*, 49(5), 58– 64.

Eessaar, E. (2014) On Applying Normalized Systems Theory to the Business Architectures of Information Systems, *Baltic Journal of Modern Computing*, Vol. 2, No. 3, 132– 149.

Fong, J. (2015) *Information Systems Reengineering, Integration and Normalization*, Springer.

Georgakopoulos, D., Hornick, M. Sheth A. (1995) An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases* 3, Kluwer Academic Publishers, Boston, 119– 153.

Gudas S., Lopata A. (2016) Towards internal modeling of the information systems application domain. *Informatica*, 2016, Volume 27, Issue 1, 1– 29.

Gudas S. (2016) Information Systems Engineering and Knowledge-Based Enterprise Modelling: Towards Foundations of Theory. *Springer Proceedings in Business and Economics, Editors: Androniki Kavoura et al. Strategic Innovative Marketing*, Springer, 481- 497.

Gudas S. (2015) Normalization of Domain Modelling in Information Systems Development. In: *7th International Workshop „Data Analysis Methods for Software Systems"*, Druskininkai, Lithuania, December 3-5, Vilnius, 20 -21.

Gudas, S. (2012) *Foundations of the Information Systems' Engineering Theory* (Lithuanian), Vilnius University, Vilnius.

Gudas, S., Skersys, T., Lopata, A. (2005) Approach to Enterprise Modeling for Information Systems engineering, *Informatica*, 2005, 16(2), 175– 192.

Kent, W. (1983) A Simple Guide to Five Normal Forms in Relational Database Theory, *Communications of the ACM* 26(2), 120– 125.

van der Linden, D., Mannaert, H., De Bruyn, P. (2012) Towards the Explicitation of Hidden Dependencies in the Module Interface. In: *ICONS 2012: The Seventh International Conference on Systems*, 73– 78.

Kecheng, L., Sun, L., Dix, A., Narasipuram, M. (2001). Norm Based Agency for Designing Collaborative Information Systems, *Information Systems Journal*, 11, 229– 247.

Mannaert, H., Verelst J., (2009) *Normalized Systems. Re-creating Information Technology Based on Laws for Software Evolvability*, Koppa.

Mannaert, H., Verelst, J., Ven, K. (2011). Towards evolvable software architectures based on systems theoretic stability. *Software:* Practice *and Experience* 42(1), 89–116.

Medina-Mora, R., Winograd, T., Flores, R., Flores, F. (1992). The action workflow approach to workflow management technology. In: *CSCW 92 Proceedings*, 281– 288.

Murray, E., Treweek, S. Pope, C., et al. (2010) Normalization process theory: a framework for developing, evaluating and implementing complex interventions, *BMC Medicine* 8(63). `http://www.biomedcentral.com/1741-7015/8/63`

Osis, J. (2004). Software development with topological model in the framework of MDA. In: *Proceedings of the 9th CaiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CaiSE'2004*, 1, Riga, Latvia: RTU, 211 – 220.

Owens, J. (2013) The function model the foundation for all business models. *Orbus Software*, 2013. `http://www.orbussoftware.com`

Özacar, T., Öztürk, Ö., Ünalır M. O. (2011) ANEMONE: An environment for modular ontology development. *Data & Knowledge Engineering* 70(6), 504-526.

Pankratius, V., Stucky, W. (2005) A Formal Foundation for Workflow Composition, Workflow View Definition, and Workflow Normalization based on Petri Nets. In: Hartmann S., Stumptner M. *(*Ed.) *The Second Asia-Pacific Conference on Conceptual Modeling (APCCM2005)*, University of Newcastle, Newcastle, Australia. Conferences in Research and Practice in Information Technology, 43, 79 – 88.

Papazoglou M. P. (2003) Web Services and Business Transactions (2003). In World Wide Web: *Internet and Web Information Systems*, 6, Kluwer Academic Publishers, 49–91.

Porter, M.E. (1985) *Competitive Advantage*. The Free Press. New York.

Rector A. (2003) Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. *Proceedings of international conference on knowledge capture K-CAP'03*, ACM press, New York, NY, USA, October 23–25, 2003, Sanibel Island, Florida, USA, 121– 128.

Rissanen, J. (1977) Independent Components of Relations. *ACM Transactions on Database Systems*, 2(4), 317– 325.

Rummler G.A. Ramias A.J., Rummler, R. (2010) *White Space revisited: creating value through process*, John Wiley & Sons Inc.

Tan, S. Liu, K. (2004) A Semiotic Approach to Organisational Modeling Using Norm Analysis. In: *6th International Conference on Enterprise Information Systems*, Porto, Portugal, 2004, 1-15.

Taylor D. (2009) Normativity and Normalization, *Foucault Studies*, No 7, 45– 63.

Van Nuffel, D., Mannaert, H., De Backer, C.,Verelst, J. (2009) Deriving normalized systems elements from business process models. In: *Proceedings of the Fourth International Conference on Software Engineering Advances (ICSEA 2009),* K. Boness, J.M. Fernandes, J.G. Hall, R.J. Machado, and R. Oberhauser, Eds. Los Alamitos, USA: IEEE Computer Society, September 2009, 27–32.

Wand, Y., Weber, R. (1995) On the deep structure of information systems. *Information Systems Journal,* 5(3), 203– 223.

Wand, Y., Storey, V.C., Weber, R. (1999) An Ontological Analysis of the Relationship Construct in Conceptual Modeling, *ACM Transactions on Database Systems,* Vol. 24, No. 4, 494–528.

Received October 6, 2017, accepted October 16, 2017