Knowledge life cycle management as a key aspect of digitalization

Eduard Babkin[1[0000-0003-2597-9043]], Tanja Poletaeva[1[0000-0002-8267-6316]] and Boris Ulitin[1[0000-0003-3774-2457]]

[1]National Research University – Higher School of Economics, B.Pechorskaya St., Nizhny Novgorod, Russia
{eababkin, tpoletaeva, bulitin}@hse.ru

Abstract. Digital transformation of organizations became a significant research and engineering challenge worldwide. Implementing such a transformation requires not only a change in the technical equipment of the enterprise, but also developing new methods of knowledge life cycle management which include extraction of individual, interpersonal or organizational knowledge to explicit machine-readable forms and their conscious application during enterprise reengineering. Successful accomplishment of these tasks vitally relies on a rigorous scientific theory and formal methods. This work presents a new approach to knowledge life cycle management of different forms of knowledge based on combination of ontology engineering and evolvable domain-specific languages.

Keywords: Digital transformation, Enterprise engineering, Ontology development, Domain-specific languages.

1 Introduction

During last decades we became witnesses of unprecedented advances in various domains of micro-electronics, communications and computer sciences. These advances have great impact on almost every aspect of economic and social structures. Such concepts as Industry 4.0 ([5], [56], [64]) Logistics 4.0 ([5], [42]), as well new models of government and public administration services ([4], [10], [52]) demonstrate new trends in development of organizational theory and business models based on digital technologies. That development assumes the wide-scope conversion process from mainly analog information into the binary machine-understandable languages, accompanied by the notion of digital innovation as "the concerted orchestration of new products, new processes, new services, new platforms, or even new business models in a given context" [27]. A large-scale combination of these digital innovations has a name of digital transformation, emphasizing crucial synergetic effects on an institutional level ([27], [29],[31], [32], [34], [52], [64]). Despite intrinsic processes of digital transformation in each application domain, common characteristic features are clearly visible: institutionalized disruptive changes of social and business domains,

intensification of networking and cooperation, emergence of complex cyber-physical systems ([27], [29], [32],[34], [52], [56]).

Radically new conditions of information processing, planning and strategic management of digitalized enterprises call to engineering of various flexible organizational forms on the solid ground of enterprise engineering principles: virtual enterprises [44], agile enterprises [36] and distributed autonomous organizations (DAO) [33].

Recent achievements in enterprise engineering ([14], [54], [65]) provide a design blue print as well as offer a certain set of methodological implications. During recent years within the enterprise engineering community the notion of Enterprise Architecture (EA) became manifestation of the systemic engineering approach to understanding and redesigning organizations ([26], [27], [56]). First of all, EA-based methods of digital transformation should lead to design of cohesive socio-technical systems because, as it is stated in [5], the digital transformation aim is not to replace humans in their works, but to avoid inaccuracies and to have faster processes where the information can be shared effortless and in real time.

Modern researches determine a complicated manner of relations and multiple perspectives on socio-technical systems ([5], [56], [64]). According to [64] vertical integration requires the intelligent cross-linking and digitalization of business units in different hierarchal levels within the organization. Such complexity makes practical implementation of digital transformation quite difficult in general and in particular domains as well. Recent analysis of Westerman et al. [67] shows that none of the 50 companies, most of which had a turnover of more than $1 billion, had successfully transformed all elements of EA. At the same time Hafsi et al. conclude that despite the ongoing research in academia, the benefits and the role of EA management in digital context are still a topic of lively discussions, and there is a gap in research on how to leverage EA for digital transformation [26]. It is concluded in [31] that the characteristics of the industry raise barriers for process innovations and effectively constrain application of EA for digital transformation. By a similar manner Oleśków-Szłapka and Stachowiak [42] point out significant problems of digitalization in Logistics 4.0, while Oliva and Kotabe determine significant barriers to knowledge management in startups [43].

Performed analysis shows, that successful implementation of digital transformation strategy vitally depends on further progress in liaison of EA practices and enterprise knowledge management in new contexts of virtual organizations and evolvable cyber-physical systems. Supporting the concept of digital transformation "as the third and ultimate level of digital literacy" [26] we have a strong reason to augment the notion of digital transformation by the concept of knowledge-based digital transformation as a new paradigm of organizational theory. That augmentation revives relevance of the pioneering work on design of inquiring systems by C.W. Churchman [11].

According to [66] during knowledge-based digital transformation enterprise modeling and knowledge management could combine their efforts to develop reference and reusable core enterprise ontologies and behavior representations as required by the smart, sensing and sustainable (S3) digital enterprises of tomorrow. Following the pioneering work of Fox and Gruninger [19] and the Enterprise Ontology of Uschold

et al. [62] several ontology-based modeling approaches were proposed, such as: DEMO [13] or MRO [63].

In the enterprise engineering studies of knowledge management within a digital enterprise an important problem arises which is proper extraction of tacit individual, interpersonal or organizational knowledge to explicit machine-readable forms and their conscious application during enterprise reengineering. In that context research multiple researchers show that tacit knowledge greatly influences behavior of enterprise ([38], [47]).

By our view in most applied engineering methods availability of structured externalized knowledge is usually only a requirement to construct models and necessary for enterprise models [44]. Specific features of tacit knowledge require development of new forms of its machine-based representation, and support mechanisms for knowledge life cycle management. In our studies a following principal research question was specified: which theoretical backgrounds facilitate design or integration of artifacts which comprise a unified solution to knowledge life cycle management fostering knowledge-based digital transformations? Fig.1 depicts such a research framework.

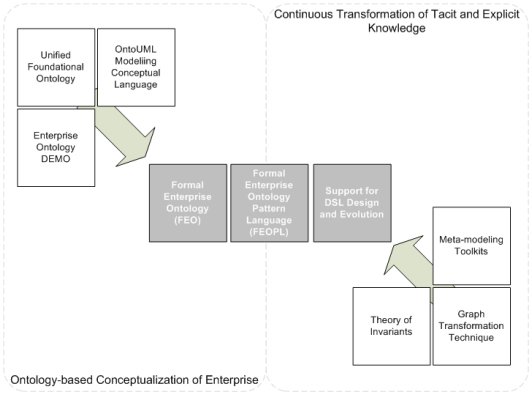

Fig. 1. The research framework of studies.

In order to pursue an answer we hypothesize that:

1. Implementation of the knowledge triad model [39], which facilitates mutual transformations of explicit and tacit knowledge, can be a practically achievable form of knowledge-based digital transformation if enterprise engineering methods are designed in accordance with social constructivism paradigm.
2. Combination of three elements becomes critical for developing these new enterprise engineering methods:
   a. methodology for extracting tacit knowledge;
   b. a constructivist view – aligned theory of comprehensive ontology-based knowledge modeling for proper conceptualization of enterprise;
   c. a theory and methodology for continuous transformation of tacit and explicit knowledge according the model of knowledge triad (using phronesis).
3. Following design artifacts may instantiate the elements a), b) and c):

- mathematical and psychological principles of repertory grids by G. Kelly [20] can be used for the purpose of reconstruction of a personal world view and developing a solid methodology for extracting tacit knowledge by application of factual approach to knowledge construction;
- a constructivist-based theory of knowledge modeling which combines advances of designing top-level ontologies and formal ontology of enterprise proposed by J. Dietz;
- an ontology-grounded theory of domain-specific languages with proper methods of their transformation and evolution, providing an explicit-tacit knowledge combination engine.

In that article we wish overview key results in developing artifacts for the elements b) and c) for comprehensive understanding of socio-technical systems, and providing a reliable decision support for digitalization.. These results include Formal Enterprise Ontology (FEO), Formal Enterprise Ontology Pattern Language (FEOPL) and a specific approach to supporting transformative evolution of ontology-based domain-specific languages.

Section 2 provides readers with necessary foundational information concerning knowledge management, enterprise engineering and domain-specific languages. Section 3 offers description of our research, the practical application of which results are shown in Section 4. In the conclusion we overview achieved results and determine directions for further investigation.

2 Foundational principles of knowledge-based digital transformation

2.1 Generic Paradigms of Knowledge Management

Hafsi et al. in [26] provide a direct connection between digital transformation and knowledge management as a specific organizational discipline that aims to acquire, transform, store, use and discard knowledge that is important in generating value for the organization. Oliva and Kotabe [43] consider the knowledge management as one of the key enterprise processes that supports the dynamic capabilities of emerging digital organizations. In the context of digital transformation Nonaka et al. [39] argue that the company needs to have organizational forms that achieve a dynamic synthesis of knowledge exploration and exploitation. Weichhart, Stary and Vernadat provide even a more radical viewpoint – the rate of new product introduction is a function of a firm's ability to manage, maintain and create knowledge [66].

We may distinguish several aspects of knowledge-based digital transformation. Interoperability becomes the first aspect because dynamic synthesis of knowledge exploration and exploitation during digital transformation raises grand challenges. For example, in [44] authors show unprecedented nature of these challenges for the case of mapping the Industry 4.0 elements to the European Enterprise Interoperability Framework. A detailed set of interoperability includes such elements as interoperability of models and processes, explicit knowledge, knowledge management systems.

Undoubtedly, as Weichhart, Stary and Vernadat noted in [66] with respect to semantic interoperability, the key element of that set is mutual ontological commitment on the basis of machine-readable shared ontologies. For instance, the Ontology of Enterprise Interoperability (OoEI) proposed by Naudet et al. [37] can give an example of ontology-based support for enterprise modeling.

We see the second aspect of knowledge-based digital transformation in a more precise stratification of knowledge onto different types. From the time of ancient Greece epistemology determines three subsets of knowledge: *techne* (the practical skill required to be able to create), *episteme* (context-independent knowledge), and *phronesis* (practical wisdom) ([66], [68]). Simultaneously in the modern knowledge creation theory, two types of knowledge are distinguished: tacit and explicit [39]. Polanyi [48] defines explicit or codified knowledge as the type of knowledge that can be effectively transferred through a formal language, and tacit knowledge as having a personal quality that makes its formalization and communication difficult. At the same time tacit knowledge can be shared, developed, and extended by physical collaboration [38]. In [47] the authors argue that the distinction between tacit and explicit knowledge is the key to understanding organizational knowledge. Nonaka and Nishihara even emphasize the importance of tacit knowledge over explicit knowledge, through an understanding that tacit knowledge is the foundation of all knowledge [38].

Distinction of two knowledge types supposes presence of a dynamic approach to the knowledge management ([40], [41]). In order to achieve deep understanding of such knowledge dynamics, which is very important for digital innovations and digital transformations ([43], [47]), some conceptualization of knowledge synthesis is required. To pursue that goal Nonaka et al. propose to combine traditional and modern taxonomies of knowledge within a unified conceptual framework of knowledge triad [39]. In that framework dynamic synthesis of knowledge is realized through the knowledge dialectics of tacit knowledge, explicit knowledge and phronesis. According that model of "knowledge triad" phronesis drives the conversion of tacit and explicit knowing. Practical evaluation of that framework in modern conditions of digital enterprises has been provided in [35], which confirms existence of four phases of the process of generating and converting knowledge phases: Socialization, Externalization, Combination, Internalization.

In that framework dynamic synthesis of knowledge is realized through the knowledge dialectics of tacit knowledge, explicit knowledge and phronesis. As it is stated in [39] it is the phronesis of the leaders with their practical wisdom that facilitates and propels new business models of dynamic fractal organizations. Taking such a holistic view point leads to the conclusion that modern foundations of knowledge management need "to synthesize the subjective and the objective, the personal and the organizational perspective" [66].

2.2 Foundations for a proper conceptualization of knowledge about enterprise

Being paired with generic principles of knowledge management Enterprise Engineering aims at developing a holistic systemic view on the construction and the operation

of enterprises [15]. However, there is no agreement about the best shared conceptualization of enterprises even in terms of a foundational organizational paradigm.

We strongly believe that it is the social constructivism paradigm which reflects the key characteristics of digital transformation and is becoming a prevailing approach in construction and evolution of organizational knowledge. According to the constructivist view, individuals actively participate in a construction of their own knowledge through interactions within complex social systems [25]. As a clear example of well-founded conceptualization and a constructivist view on enterprise knowledge evolution and management the enterprise ontology [13] can be distinguished. That approach includes the ontology-based concise, comprehensive, coherent, and consistent enterprise modeling language, and the corresponding modeling methodology (DEMO - Design & Engineering Methodology for Organizations). Providing a consistent set of micro-theories grounded in Language-Action Perspective (PSI – Performance in Social Interaction theory), the enterprise ontology represents a coordination viewpoint underlying other ontological theories of enterprises.

2.3 Definition and classification of ontologies

Ontology is a representational artifact, comprising a taxonomy as a proper part, whose representations are intended to designate some combination of universals, defined classes, and certain relations between them [3].

According to this definition, the following considerations can be deduced: (1) the ontology is a representational artifact = def. the scheme of a certain area; (2) the ontology contains concepts of a certain area, its properties and relations between them; (3) a proper part of relations are taxonomy-type relations.

Based on these considerations, the ontology can be represented as a triple(O, R, F), where $O = U \bigcup C$ is a set of objects, where $U = \{u_1, u_2, \dots, u_N\}, N \in \mathbb{N}$ (where $u_i, i = 1, N$ is a concept (universal) of a certain area an can be represented as a set of its attributes $u_i = \{attr_1, attr_2, \dots, attr_M\}, M \in \mathbb{N},, i = 1, N)$, C(class) is a set of $c_i, i = 1, K,\ K \in \mathbb{N}$, where c_i is an exemplar of some u_j, R is a set of relations between elements of O, and F is an interpretation function assigning values to the non-logical constants of the language [22].

From this point of view, the ontology can be naturally perceived as a graph (O, R), with a set of functions of constraints F.

On the other hand, the ontology is some kind of representation, created by the designer [23]. From this point of view, development of the ontology has always some certain goal, which affects the whole design process and its final result, the ontology itself. As a result, the following classification of ontology kinds based on their level of dependence on a particular task (or a viewpoint) can be identified [3, 22]: top-level ontologies, which describe very general concepts, independent of a particular problem or domain; domain (task) ontologies, which describe, respectively, the vocabulary related to a generic domain (task) and application ontologies, which describe concepts that depend both on a particular domain and a task, and often combine specializations of both the corresponding domain and task ontologies.

In the current research, we pay attention mostly to the enterprise ontologies, which refer to the Application ones. But as the Domain ontology contains only the necessary concepts of a subject area, Application ontology operates with a subset of these concepts necessary to achieve a certain goal. That allows us to consider the Application ontology as the reduced Domain ontology and can be used as a basis for development of DSM which in turn is used in the development of domain-specific languages.

2.4 Domain Specific Languages

Static conceptual structures of the enterprise ontology alone are not capable of maintaining mutual transformations of explicit and tacit knowledge during knowledge life cycle management. For that purpose, ontology should be fused with a specific mechanism for dynamic generating and converting knowledge. Evolvable domain-specific languages may be considered for that role.

A number of research results demonstrate suitability of using Domain-Specific Languages (DSL) for defining the context for different knowledge modeling and management tasks of modern companies ([16], [17], [22], [23], [46], [51], [53], [57]). For example, Sprinkle [57] describes the implementation of DSL for modelling logistic interactions within the organization. Pereira et al. [46] prove the effectiveness of DSL usage for the definition of the context of the resource allocation problem.

In the context of our studies frontiers of DSL application for extraction and transformation of tacit knowledge attract special interest, because as Colins states – language plays a role of a repository for tacit knowledge [12]. Indeed, domain-specific languages can be considered as a practical implementation of interactional expertise, which may be viewed also as an attempt to introduce the tacit dimension of linguistic knowledge [55]. In [69] ontologies and domain-specific languages were considered as among the primary tools for extraction and representation of explicit and tacit knowledge in the safety domain. Gross demonstrates application of visual domain-specific languages for grasping tacit knowledge in a complex domain of artistic lighting [21].

Formally a domain-specific language is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language, which is broadly applicable across domains, and lacks specialized features for a particular domain [18]. In [45] two parts of the DSL are identified: (1) a syntactic part, which defines the constructions of DSL; (2) a semantic part, which manifests itself in the semantic model. The syntactic part allows us defining the context for working with the second one, which defines meaning of DSL commands in terms of the target domain. The syntactic part itself contains the domain concepts and rules (abstract syntax), as well as the notation used to represent these concepts – let it be textual or graphical (concrete syntax).

A syntactic part of DSL can also be separated into two levels: the level of objects and the level of functions. The object-level is equivalent to the set of objects of the meta-model. The functional level contains operations, which specify the operational context for the objects. That two-level division of the DSL syntactic part provides the maximum correspondence between the ontological model of the target domain and

the DSL model, and the most convenient way of organizing conversions between them.

A semantic part of DSL is derived from the conceptual model of the target domain. According to Parr [45] we will call such a model as Domain-Semantic Model (DSM). DSM can be constituted by either just small pieces of domain knowledge (e.g. small taxonomies equipped with few rules) or rich and complex ontologies (obtained, for example, by translating existing ontologies). That gives respectively weak or rich and detailed representation of a domain [6]. In our research DSM becomes the bridge between the enterprise ontologies and DSL.

Static features of DSL are well studied and a lot of automated tools exists to design and exploit DSL in the enterprise practice. However, as a dynamic complex structure, any domain demonstrates the tendency to the evolution over time: new concepts may arise, while others unite into more general ones or become obsolete. In accordance with these changes, DSL should also support the possibility of evolution.

The simplest option in this case is to rebuild DSL whenever the domain model changes. But this process has several disadvantages. First of all, the process of DSL development is really time-consuming, since DSL contains internal and external parts, connected with the domain model and DSL syntax correspondingly. Secondly, DSL development, since DSL is a language, is often associated with the use of grammar tools that require special skills from developers. Finally, while a new version of DSL is being created, the domain can be changed again. Thus, the DSL changes may not be synchronized with the domain changes, making DSL not fully compliant with the needs of the end users.

3 Developing own unified solution

In that section we perform a synthesis of aforementioned foundations and describe own contributions to the theory and practice of knowledge life cycle management and knowledge-based digital transformations. At first, an ontology-based conceptualization of enterprise is described, which facilitates ontology-based description of cornerstone enterprise concepts. Secondly, an ontology-based methodology for continuous transformations and verification of DSL is given, which can play a role of the mechanism for continuous transformation of tacit and explicit knowledge.

3.1 Building an ontology-based conceptualization of knowledge about enterprise

In order to play a role of an ontological basis for knowledge-based digital transformation enterprise ontology should be fused with a corresponding foundational ontology because the core enterprise theory provided by DEMO is not fully axiomatized yet. Rephrasing the definitions made by Guizzardi in [22] with regard to enterprise modeling, the domain appropriateness and the comprehensibility appropriateness of an enterprise conceptual modeling language is guaranteed by the meta-model of this language representing a full axiomatization of enterprise ontology.Despite the plurality

of existing foundational ontologies, in our work we exploit the Unified Foundational Ontology (UFO) and its compliant conceptual modeling language OntoUML [22] in order to build and to represent an ontological theory of enterprise interactions with the basis on a solid theoretical framework.

Practical implementation of such fusion requires performing two intellectual tasks. At first hand, a consistent ontology-based conceptual modeling language with a strong referential semantics should be designed. Secondly, a set of design-oriented practices should be developed which impose relevant constraints on the modeling language application during solution of recurrent modeling problems of enterprises.

For the solution of the task of developing a conceptual modeling language, the OntoUML conceptual modeling language [22] was taken as a basis. Restating the DEMO enterprise ontology in terms of OntoUML allowed us to combine modeling elements of enterprise ontology and reference ontology (UFO), as well as to reveal some gaps and inconsistencies on the analysis of the DEMO foundations. Moreover, we added some additional ontological categories based on their relevance for the theory of enterprise ontology.

This work resulted in a fully axiomatized Formal Enterprise Ontology (FEO) [49]. This is the domain- and standard-independent ontological theory that provides a referential semantics for metadata. The UFO-C part (a foundational ontology of social entities) [24] guarantees a well-defined ontological foundation of FEO and expressiveness of the essence of an organization in the ontological categories of foundational ontology.

The OntoUML-based FEO language provides modeling primitives that reflect the conceptual categories and axioms defined by the whole ontological theory. For example, FEO includes the axioms in first order logic that supplement the forgoing definitions by relevant constraints and formally specify the notion of an ontological transaction.

The second task aimed at solving recurring conceptual modeling problems of enterprises by adapting a generic notion of ontological patterns to FEO. Ontology patterns [17] were considered as a promising approach to capture standard domain-specific solutions to recurrent problems of conceptual modeling [16]. In general, each pattern has to be dedicated to a particular type of modeling issues, provide a solution, be accompanied with the instructions about its applicability in a right situation, and be associated with the set of related patterns. A set of interrelated patterns comprises a certain pattern language which can be applied systematically depending on requirements of the modeled situation and the goals of the modeler.

Following the method proposed by Guizzardi in [22], we created a set of modeling constructs (ontology patterns) represented in OntoUML, and called it the Formal Enterprise Ontology Pattern Language (FEOPL) [50]. All patterns of that language inherit axiomatization of the FEO ontology, thus making the meta-model of the language isomorphic to this ontology.FEOPL patterns include the following:

- a Transaction pattern, intended (1) to specify the notion of transaction and (2) to tackle problems related to modeling of properties and an evolution of basic units of business processes;

- the pattern of Coordination actions and their resulting commitments, which puts together Actor Role, C-act, C-act Intention, C-act Proposition, Transaction, C-commitment and their interrelations;
- the pattern of Production Actions, which states that propositional contents of production actions are abstract representations of allowable/desired states of the production world of an enterprise;
- the pattern of Production Facts, which explicitly defines semantics of the notion of a production fact.

A more detailed description and analysis of the presented templates is contained in [50]. As patterns description shows, FEOPL facilitates formalization of rules and conditions for social coordination actions based on the information derived from domain ontologies. Moreover, the language meta-model correlated with both the upper level ontology and the FEO preserves real-world semantics in a broad sense. That is reducing the number of semantic conflicts in representation of enterprise domain.

Grounding FEOPL in the DEMO modelling language leverages application of formal methods for knowledge management because the FEOPL patterns define precise semantics for interrelation between lifecycles of social objects (including transactions, commitments and claims) and lifecycles of enterprise products.

The modelling power of FEOPL was investigated in application to modelling problems of test-bed case studies. The results of modelling confirmed relevance and efficiency of FEOPL application for modelling knowledge-based digital transformations.

3.2 Evolution of Domain-Specific Languages for Managing Knowledge

Having reliable tools of conceptualization in terms of formal enterprise ontology, we may consider further advancing of DSL design on that solid ground. Application of the ontology as a model of DSL guarantees that DSL is identical to the corresponding domain, thereby allows the users interacting with it more effectively.

In our approach [59] we apply the formalization of the semantic DSL level in a model-oriented manner as a combination (O, R) of some objects of the target domain and relations between them, where each object is a set of its attributes and tions $o_i = (Attr_i, Opp_i) = \begin{pmatrix} \{attr_{i_1}, attr_{i_2}, \dots, attr_{i_M}\}, \\ \{opp_{i_1}, opp_{i_2}, \dots, opp_{i_K}\} \end{pmatrix}, M, K \in \mathbb{N}, i = 1, N)$.

In these terms, the syntactic part of DSL can be represented as a subset of the semantic level, needed for representation of a certain problem situation. The very one difference is that the syntactic part may not absolutely reflect the semantic constructions but identify its own definitions (pseudonyms) for the semantic constructions, according to the user's needs.

As follows, the structure of the syntactic level can be formalized as a triple $(O_{syntax}, R_{syntax}, Alias_{syntax})$, where $O_{syntax} \subseteq O$ and $R_{syntax} \subseteq R$ are the subsets of objects and relations between them of the semantic DSL level respectively, and $Alias_{syntax}$ is a set of pseudonyms for objects' components (attributes and operations).

By a similar object-oriented manner Domain-Semantic Model can be derived from the corresponding FEO and can be represented as a seven-tuple:$DSM = (\mathcal{H}_C, \mathcal{H}_R, O, R, A, M, D)$, where

- $\mathcal{H}_\mathrm{C}$ and $\mathcal{H}_\mathrm{R}$ are sets of classes and relations schema. Each schema is constituted by a set of attributes, the type of each attribute is a class. In both $\mathcal{H}_\mathrm{C}$ and $\mathcal{H}_\mathrm{R}$ are defined partial orders for the representation of concepts and relation taxonomies;
- O and R are sets of class and relation instances also called objects and tuples;
- A is a set of axioms represented by special rules expressing constraints about the represented knowledge;
- M is a set of reasoning modules that are logic programs constituted by a set of (disjunctive) rules that enables reasoning about the represented and stored knowledge, so new knowledge not explicitly declared can be inferred;
- D is a set of descriptors (i.e. production rules in a two-dimensional object-oriented attribute grammar) enabling the recognition of class (concept) instances contained in O, so their annotation, extraction and storing is possible.

In our research, we focus only on the sets O and R. Consideration of other parts of DSM is beyond the scope of our study, since it determines more meta-characteristics of DSM itself, rather than the objects and connections between them, which are the most interesting for the further development of the DSL semantic model.

Finally, taking into account previously mentioned formalization of the ontology as a triple(O, R, F) [22], we can argue, that the ontology can be naturally perceived as a set (O, R), with a set of functions of constraints F. Such definition of the ontology guarantee, that an ontology can be completely transformed into DSM, which ensures complete consistency of all three models (ontology, DSM and DSL model) with each other.

Under these circumstances, we can tell about the complete ontology-based and model-oriented representation of the DSL structure. That representation corresponds to principles of model-driven engineering [7]. From this point of view, we can derive DSL semantics as result of transformation of DSM. Similarly, using transformation rules on entities and relationships between them from the DSM, the meta-model of a DSL can also be defined. Finally, reflecting DSL abstract syntax terms on the concrete visual icons or textual constructions DSL concrete syntax can be defined.

Such a hierarchical model-driven approach allows us not only to describe both levels of DSL in structured and unified manner but optimize the process of DSL development and evolution by introducing several syntactic DSL dialects on one fixed semantic level. Furthermore, the versification of DSL can be provided in a similar way on the semantic level as well as on the syntactic level, without need to re-create the whole DSL structure every time, when the changes are required. All these features open practical opportunities for proper reflection of transition between tacit and explicit knowledge of the users in a corresponding evolution of different DSL dialects with varying syntactic or semantic elements. Also, automation of DSL syntactic and semantic transformation using MDE principles forces traceability between different DSL dialects and allow us to use advanced methods of formal verification, as it will show later.

Combining the object-oriented model of the DSL structure with the formal definition of DSM on the basis of a single meta-meta-model, we can specialize a well-known semantic hierarchy of meta-models for our approach to model-oriented development and evolution of DSL (fig.2). In our case this hierarchy is separated into four layers, according to the stages of the DSL development. Each lower level is based on the model artefacts of the upper level A single M3 meta-meta-model determines common grounds for all meta- and models of the lower levels. This meta-level defines also notations in which concrete models will be defined and what rules for their transformations will be used.

We propose to create a DSL structure from the Domain-semantic model (DSM) through the so-called semantic projection mechanism. The semantic projection is an operation, which is conducted over DSM. Any semantic projection performs a certain model-to-model transformation (M2M) of DSM to some its fragment. Thus, semantic projection fully determines the semantic model of a particular dialect of DSL.

We suggest application of a group of model-to-model transformations for practical implementation of semantic projections and producing corresponding DSL artefacts. In this case the semantic model becomes an object-temporal structure, because it should be adopted according changes in DSM over the time, thereby defining a new object filling of the DSM.

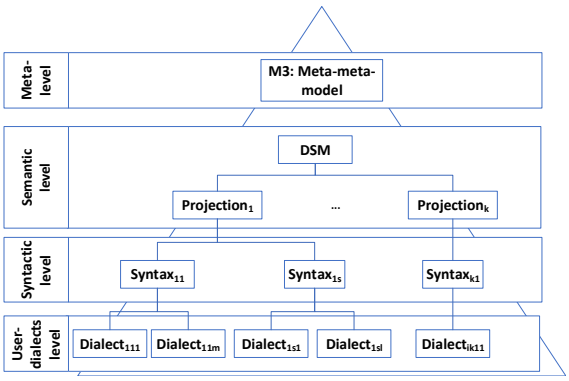

Fig. 2. The semantic hierarchy of projection-based DSL development.

After the semantic projection was performed, the syntactic level of DSL can be developed by a M2M transformation of the result of the corresponding projection. What is important, these DSL syntactic models are independent of each other and are determined by end-users in accordance with the adaptation of the semantic projection to their own tasks. Finally, created syntaxes are used by the end-users of DSL, who determine the set of DSL dialects within the single specific syntactic model.

Fig.3 shows differences between traditional approaches and our proposals. Traditional approaches start with the manual definition of the DSL concrete syntax which is followed by the translation of the syntax in terms of grammars. Consequently, every change in the target domain leads to the need to redefine the DSL concrete syntax and re-create the corresponding grammar. A similar process repeats in a case, when changes in DSL are caused by the end-users. As a result, outcomes of tradi-

tional approaches contain inconsistent dialects of DSL, which cannot be mapped among themselves due to differences in all levels of the DSL structure.

In order to provide such transformations, it is sufficient to adjust the system of matching rules between the components of the ontological model of the target domain and the components of the DSL model. One of the possible solutions to this problem can be the mechanism of graph transformations between the graph representation of the ontology and the DSL model. If we interpret the entities of our model as vertices of the corresponding graph and relations between them as corresponding edges, we can postulate, that any model can be described using graph-oriented manner. This fact logically results in the opportunity to describe model transformations and the corresponding dynamic operational semantics of DSL using graph transformations rules. It is important to note, that such advanced methods of graph transformation as Graph Grammars (TGGs) [28] enable specification of direct and inverse transformation rules facilitating bi-directional DSL transformation.

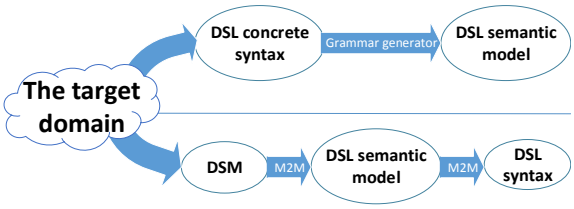

Fig. 3. The scheme of differences between traditional (top) and projection-based (bottom) DSL development approaches.

Practical implementation of such transformation can be achieved by using one of specialized graph-transformation languages such as ATL Transformation Language, GReAT (Graph REwriting and Transformation) [1], AGG (Attributed Graph Grammar), QVT. In our current research, we propose to use QVT, because this language allows us to describe the transformation rules from any original model into any target model, conducting a transformation at the level of meta-models. Using that instrument we demonstrated evolution of DSL in the railway allocation domain as an example of knowledge-based digital transformation [58]. In that case our method enabled evolutionary changes of syntax and semantics of DSL in response to changing the knowledge model of the users. Such changes frequently occur during modification of the business model of the railway services.

In the scope of automated transformation between different variants of DSL an important issue of transformation verification arises. If the verification succeeds, then we conclude that the model transformation is correct with respect to each pair (p, q) of properties (objects, relations) for the specific pairs of source and target models having semantics defined by a set of graph transformation rules. Otherwise, property p is not preserved by the model transformation and debugging can be initiated based upon the error traces retrieved by the model checker. That debugging phase may fix problems in the model transformation or in the specification of the target language.

In what follows, we offer the unified and highly automated approach, allowing developers to formally verify by model checking that a model transformation (specified

by meta-modeling and graph transformation techniques) from an arbitrary well-formed model instance of the source modelling language into its target equivalent preserves (language specific) dynamic consistency properties. In that approach the notion of invariants is specialized for a particular case of DSL verification.

In terms of the most general approach [8], the invariant is a property, held by a class of objects, which remains unchanged when transformations of a certain type are applied to the objects. From this point of view, invariant can be interpreted in two ways: (i) a set of objects, which leave unchanged during the transformation provided, (ii) an operation, which can be applied to several objects at the same time (e.g. operation RENAME, which change the name of the object, regardless of its type). Taking into account these ideas, invariants are separated into two classes: structural and functional (inductive and operational) invariants. In both cases invariants are defined on top of some transformation (transition) of the set of objects.

For example, consider an inductive invariant. Usually it determines that there is a strong correspondence between elements of two sets of objects, which are connected during some relation (transformation). Such definition is very close to the relational approach for model transformation definition, when the relationship between objects (and links) of the source and the target language are declared. That results in the insight, that the inductive invariant can be an effective mechanism for the definition of such model transformations and for the validation of the feasibility of obtaining one model by transforming another. We may conclude that the process of graph transformation resembles the search for various structural invariants in the source and the target with consequent application of corresponding graph transformation rules to them. Consequently, we can reformulate model transformations using the double-pushout approach (DPO) with injective matching for graph transformations and an invariant technique.

According to these principles, we can conclude, that validation of the model transformation correctness can be fully described through invariant mechanisms. Such definition allows us to automate the process of formal validation of the model transformation, reducing it to verifying the presence of invariants of both types among defined model (graph) transformations.

Since we describe the model transformation using the graph-oriented approach in QVT transformation language, the procedure to derive the OCL invariants need be implemented. With application of OCL invariants both problems can be solved using existing OCL verification and validation tools for the analysis of model transformations. With these inputs, verification tools provide means to automatically check the consistency of the transformation model without user intervention. Checking consistency enables the verification of the executability of the transformation and the use of all validation scenarios.

In our resent work [60] details of our approach to invariant-based transformation are provided together with the overview of an actual implementation of the verification algorithm for a case of transformation between different enterprise models.

4 Demonstration in a practical case

In that section we demonstrate practical application of our approach for the case of the railway station resource allocation domain. Partially aspects of this evolution were considered in our previous works [58, 59], therefore, here we will pay more attention directly to demonstrating the evolution of the language than to analyzing the process of its development and content.

4.1 DSM for railway allocation domain

The domain of railways services represents an interesting and significant case of dynamic management of knowledge and enterprise digitalization. In particular the context of the railway allocation problem can change frequently because of arrivals of new trains, or changing the priority of existing services. As a result, we have to have a clear and simple way to adapt new changes in terms of the proposed framework, responsible for finding the optimal resource allocation. In the process of DSL design for the railway allocation process, it's vital to identify all the types of resources in this domain.

There are three general resources for any railway station, each with specific attributes: railways, trains and service brigades. All of them are represented in the DSM for the corresponding domain, which is more complete in comparison with that considered in our previous work [59], since it contains the specification of the requirements (Skills) both for the Services and for the Brigades providing them.

After the DSM created, we can identify the semantic level of DSL, describing the DSL meta-model. For this purpose, M2M transformation rules can be used, as it was described in [58]. This is reasonable since both DSM and DSL meta-model are described in a model-oriented way. In addition, M2M transformations are independent from the notation of model definition, that allows us to describe DSM and DSL meta-model independently, in the most appropriate way. As a result, we will have the complete DSL meta-model, which can be used during the following DSL syntax definition. This definition includes two parts: definition of objects for DSL syntax, which are the equivalents to the objects, described on the semantic level of DSL, and grammar, describing the operations and correct terms for the future DSL syntax. In our case, we used the Backus-Naur form of grammar definition, because this form allows us to identify rules, based on the previously created objects, and automatically convert the resulting rules into an abstract, language-independent form.

As a result, the created DSL semantic and syntactic levels are wholly coherent and can be evolved using transformations in real time. In addition, such changes are provided separately, since the invariants on both levels are identified.

In order to demonstrate how the evolution of users' knowledge reflects into the transformation of DSL terms we analyze to states of the DSL: original one, derived from the DSM, and its further development using evolution tools.

4.2 The original DSL

As a starting point for our DSL development we consider, that DSL supports only basic constructions and operations on objects defined in the DSM (railways, trains and service brigades) and the relationship between them. As a result, the following structure of DSL terms exists (fig. 4and fig. 5). As you can see, these constructions are sufficient to perform the basic operations of the domain: creating objects and establishing links between them (fig. 6).

```
Trains
({id type priority length [services] wagonsToService})*
EndTrainsBlock;

Railways
   ({id type totalLength usefulLength [equipment]})*
endRailwayBlock;

Services
   ({name priority [skills] standartDuration [equipment]})*
endServiceBlock;

Brigades  ({id [ skills ] capacity})* endBrigadeBlock;
```

Fig. 4. DSL objects.

```
Put idTrain to idRailway with startNick;
```
This command allows to move Train with identifier idTrain to railway idRailway on start nick startNick

```
Relocate idTrain [from idOldRailway] to idNewRailway;
```
This command allows to move Train with identifier idTrain from railway with identifier idOldRailway to railway idNewRailway.

```
Move forward/back idTrain by countShifts;
```
This command allows to move Train within current railway forward or back by the count of shifts equals to countShifts.

Fig. 5. DSL functions.

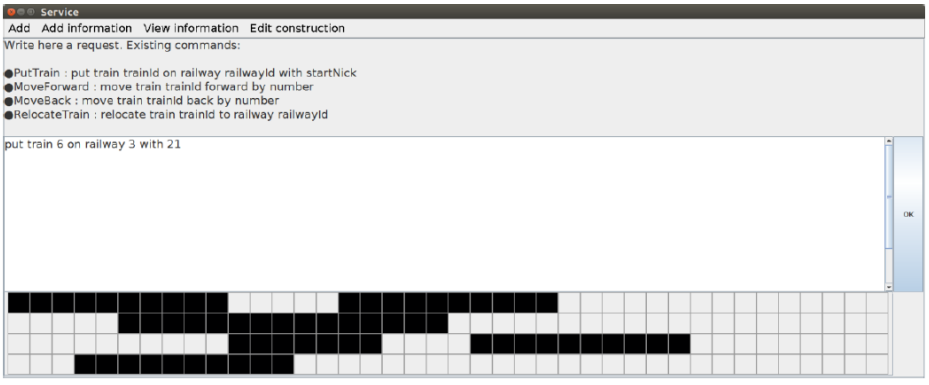

Fig. 6. Example of scenario in terms of the original DSL.

On the other hand, these commands do not reflect the time perspective of the domain, distributing only the set of resources available at a given moment in time between arriving trains.

In order to improve the quality of DSL and provide the user with the ability to plan resource allocation over time, it is necessary to make changes to the original DSL. As a result, we need to provide the user with the ability to change the design of the DSL, which creates a more complex version of the subject-oriented interface.

4.3 Subject-oriented GUI

Developing subject-oriented GUI we should take into account, that it pursues two goals: writing and executing scenarios in the current version of DSL, as well as making changes (evolution) into DSL.

As a result, the interface created contains two parts: the first one, responsible for the DSL scenario definition and processing, and another one, needed for evolution of DSL. The first part, which contains only a visual panel, representing all the DSL components needed for definition of DSL scenarios, was properly described in our previous work [61] and mentioned in previous section. In what follows, the second part of the interface (see fig. 7 and fig. 8) responsible for DSL evolution is more interesting for us.

This part of interface allows us to adopt DSL automatically whenever the evolution is provided. Such automation allows us to support DSL evolution by end-users without the need to re-compile the whole framework and to have special programming skills.

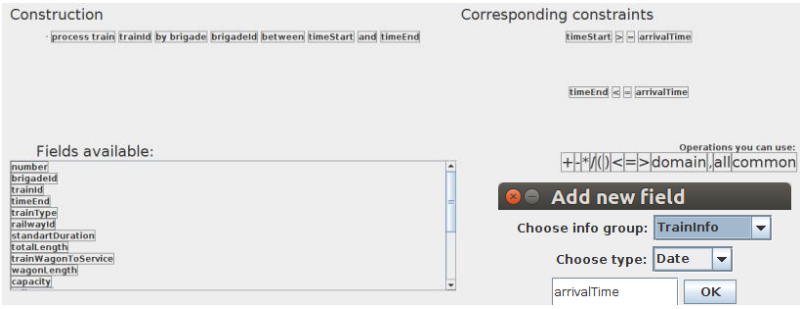

Fig. 7. Evolution of DSL implementation.

Fig. 8. Scenario with added command.

In order to design such evolution, the second part of the interface is used (fig. 7). This part contains three main components: the component to define/change a new/existing command of DSL, the component for definition of constraints, connected with the command and the component for definition of syntactic terms, related to the new command. All these components are identified in accordance with the structure of DSL: objects, which contain attributes and operations and relations between them. As a result, the created interface allows us to define the whole DSL structure and change it in real time without need to re-create the DSL manually.

4.4 Evolution of DSL

The first change we want to provide to the user is to add a time perspective to all objects. In fact, this means that each of the objects will have an additional attribute associated with time. For example, arrival time at the trains, the start and end time of servicing at the servicing brigades, etc.

From a formal point of view, this DSL development scenario is an example horizontal evolution. In more details, this classification of evolution was described in [58, 59].

In this case, using the evolution part of the interface, we add a new attribute *arrivalTime* to the *TrainInfo* object. For this purpose, the corresponding interface component can be used (fig. 7, right). Similarly, we add other time attributes to other objects. After making such changes, we can argue that the structure of DSL objects has changed. However, more importantly, we can reflect these changes at the DSM level, using the model-to-model (M2M) transformations, as it as in details described in our previous work [59].As a result, we can argue that in this case there was a transfer of tacit knowledge of users to explicit knowledge.

It is important to note that the changes made are immediately applied to the language and can be used in further evolution and scenarios.

For example, we can extend syntactic part of DSL by adding new command: *process train trainId by brigade brigadeId from timeStart till timeEnd*. This command uses existing objects for the DSL semantic level: a train and a brigade, but implements a new syntactic term and new attribute added in the previous case of evolution. In order to implement this command, the second part of GUI is used. First of all, the user should define a needed command, using the block of available fields of DSL objects. As a result, the following construction and constraints, related to this, are defined (fig. 7). Finally, the created term is compiled and added to the DSL, ready to use.

What is the most important, in this case, we only define new commands, without need to re-create the DSL structure and can use them in sceneries in real time. For example, the result of added command is represented in fig. 8. As follows, the approach proposed allows us to implement all types of DSL evolution in real time, correctly transforming new commands into DSL syntactic and semantic objects and terms.

Currently existing approaches, allowing also to allocate resources of railway station, are targeted to one concrete type of resources (for example, to brigades by Wang

et al. [66], or to trains by Chen et al. [9]). Furthermore, such approaches use static models of resource allocation and cannot be adopted according to new types of resources or solving models in real time. In comparison to existing approaches, the approach proposed is independent from the nature of the resources and can be adopted to any other domain.

The only limitation for our approach is the fact, that it can provide the opportunity to define only unidirectional transformation of DSL, according to changes in the domain model. This limitation can be explained by the fact, that languages of model transformations do not support bidirectional transformations, because symmetric transformation means using the opposite to the original operation (delete instead of add, etc.). However, such limitation can be resolved using the idea of closure operations necessary for organizing the DSL evolution [59].

5 Conclusion

Critical aspects of digital transformations, including the cross-institutional level of changes, dynamic nature of emerging business models and increasing importance of knowledge management strategies in the course of designing digital enterprises as inquiring systems, lead us to the conclusion that successful digital transformation requires application of a systemic engineering approach.

That article aimed at observing a complex phenomenon of digital transformation from the systemic viewpoint of enterprise engineering. Our attention was attracted to further progress in combination of enterprise engineering techniques and different knowledge types in order to facilitate knowledge life cycle management in the context of knowledge-based digital transformations because the practice of continuous defining, acquiring, disseminating, storing, applying, and assessing knowledge in organizations prepares people and potentializes internal changes.

Along that way several contributions were proposed in the ontology engineering. A new ontology modelling language of Formal Enterprise Ontology (FEO) was proposed which restates DEMO in precise terms of UFO. FEO gives a modeling language with precisely defined formal semantics provides an input for inference procedures and engines with a minimal information loss. Represented in OntoUML the FEOPL patterns fully inherit the FEO. In addition to a modeling power inherited from OntoUML, the FEOPL patterns enforce a correlated modeling of changes (the behavioral perspective of an organization) and objects undergoing these changes (the structural perspective of an organization).

We believe that proper combination of FEO and FEOPL with evolvable domain-specific languages facilitate continuous transformation of explicit and tacit knowledge. In our research, we explored an opportunity to provide the method of co-evolution of the ontology, used as a model of the subject area, and DSL. We proposed a formal ontology-based DSL structure together with a method of semantic projections. This method combines graph representation of the ontology and DSL with the set of rules, formulated in terms of an automated graph-transformation language. This mechanism has several advantages: the DSL designer does not need to know the se-

mantic domain(s), nor the relationship between the concepts of his/her DSL and the concepts of the semantic domain, and he/she can still be benefited from its analysis tools. We call semantic bridges to those general mappings between different domains from which DSL-specific semantic mappings can be automatically derived. Models can then cross these bridges to benefit from each world without being aware of how they were constructed.

In comparison to traditional approaches, the proposed projection-based method of DSL development is organized in the strong correspondence of the target domain. Such correspondence is provided by the consequent projections among different models in a semi-automatic way through M2M transformations: from DSM into a DSL semantic model and then into a syntactic model of the specific DSL dialect. In comparison with existing approaches to transformation verification like [2] and [30], which also use the ideas of automated model generation with subsequent correctness property checking, our approach doesn't depend on the modelling language and property chosen. Such independency follows from deriving invariants as stable logical structures from the model transformation rules. As a result, the verification procedure reduces to a simple check of two sets of OCL constraints between themselves.

Using our approach, we can define several DSL syntactic dialects over one specific DSL semantic model expressed in the form of FEO, which will be consistent and can be transformed between themselves without the redefinition of the DSL semantic models. Applicability of the proposed approach was demonstrated using a real-life example of co-evolution of the ontology and DSL in the railway transportation domain. Evaluation of the software prototypes has demonstrated that our approach to fusion practically enables continuous transformation of domain-specific languages in response to changes of the underlying enterprise ontology or knowledge of the users.

That example demonstrated an attractive feature of our method regarding the ratio of explicitly formulated knowledge. As fig.9 shows, evolution of DSL leads to increasing complexity of the user interface in terms of number of available elements and relations between these elements. As far as the user expresses knowledge about the subject area in terms of DSL more and more implicit knowledge can be reformulated explicitly.

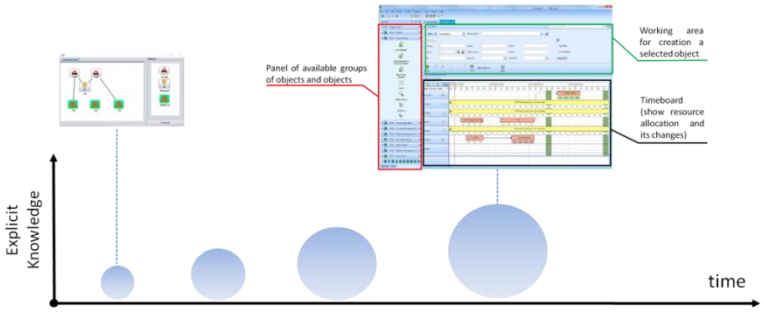

Fig. 9. Growth of explicit knowledge alongside using DSL.

Changing the focus to the second pillar of our approach, namely ontology-based methods of dynamical evolution of domain-specific languages, we also can recognize

some important directions of further research. In present kind our software prototypes require manual elaboration of user's insights and transformation of their tacit knowledge to the explicit knowledge via modification of ontology. It will be beneficial to adapt machine learning algorithms for automatic production of recommendations for ontology changes on the basis of intellectual analysis of users' interactions with a DSL. Another improvement includes design of more efficient model transformation algorithms for cases of complex domains.

In our vision achieved results and prospective plans clearly envisage importance and great potential of designing deep interconnections between such elements of enterprise engineering as enterprise ontologies and domain specific languages. We hope that results of such interconnections will facilitate efficient and effective knowledge-based digital transformations.

6 References

1. Agrawal, A., Karsai, G., Shi, F., 2003. Graph Transformations on Domain-Specific Models. In: International Journal on Software and Systems Modeling, pp. 1-43. Nashville: Vanderbilt University Press.
2. Akehurst, D., Kent, S., 2002. A relational approach to defining transformations in a meta-model. In Proc. Fifth International Conference on the Unified Modeling Language – The Language and its Applications, LNCS, vol. 2460, pp. 243–258. Springer.
3. Arp, R., Smith, B., and Spear, A.D., 2015. Building Ontologies with Basic Formal Ontology. The MIT Press.
4. Bani, M., De Paoli, S., 2013. Ideas for a new civic reputation system for the rising of digital civics: digital badges and their role in democratic process. In: ECEG2013–13th European Conference on eGovernment: ECEG.
5. Barreto, L.,Amaral, A., and Pereira, T., 2017. Industry 4.0 implications in logistics: an overview. Procedia Manufacturing, 13, pp. 1245-1252.
6. Bell, P., 2007. Automated Transformation of Statements within Evolving Domain Specific Languages. Computer Science and Information System Reports, pp. 172–177.
7. Beydeda, S., Book, M., 2005. Model-driven software development. Heidelberg: Springer.
8. Chandy, K.M., 1989. Parallel program design. In Opportunities and Constraints of Parallel Computing (pp. 21-24). Springer, New York, NY.
9. Chen, W., Dong, M., 2018. Optimal resource allocation across related channels. Operations Research Letters, pp. 397-401.
10. Chou, J., Hsu, S., Lin, C., Chang, Y., 2016. Classifying influential for project information to discover rule sets for project disputes and possible resolutions. Int. J. Project Manag. 34, pp.1706–1716.
11. Churchman, C. 1971. The Design of Inquiring Systems: Basic Concepts of Systems and Organization. Basic Books Inc., Publishers. NY, London.

12. Collins, H., 2012. Language as a repository of tacit knowledge. In The symbolic species evolved. pp. 225-239). Springer, Dordrecht.
13. Dietz, J. L.G., 2006. Enterprise Ontology: Theory and Methodology. Springer, 2006.
14. Dietz, J.L., Hoogervorst, J.A., Albani, A., Aveiro, D., Babkin, E., Barjis, J., Caetano, A., Huysmans, P., Iijima, J., Van Kervel, S.J. and Mulder, H., 2013. The discipline of enterprise engineering. International Journal of Organisational Design and Engineering, 3(1), pp.86-114.
15. Dietz, J.L.G., Hoogervorst, J.A.P., 2008. Enterprise Ontology In Enterprise Engineering, In Proceedings of the 2008 ACM symposium on Applied Computing, pp. 572-579.
16. Falbo, R.A., Barcellos, M.P., Ruy, F.B., Guizzardi, G., Guizzardi, R.S.S., 2016. Ontology Pattern Languages. Ontology Engineering with Ontology Design Patterns: Foundations and Applications. IOS Press.
17. Falbo, R.A., Ruy, F.B., Guizzardi, G., Barcellos, M.P., and Almeida, J.P.A., 2014. Towards an Enterprise Ontology Pattern Language. Proceedings of 29th Annual ACM Symposium on Applied Computing (ACM 2014), pp. 323–330.
18. Fowler, M., 2010. Domain Specific Languages. Addison Wesley.
19. Fox, M. S., Gruninger, M. 1998. Enterprise Modeling. AI Magazine, 19 (3), pp.109–121.
20. Fransella, F., Bannister, D., 1977. A Manual For Repertory Grid Technique. Academic Press.
21. Gross, J.B., 2005. Programming for artists: a visual language for expressive lighting design. In 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05), pp. 331-332. IEEE.
22. Guizzardi, G., 2005. Ontological foundations for structural conceptual models. Telematics Instituut Fundamental Research Series, The Netherlands.
23. Guizzardi, G., 2007. On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Proceedings of the 2007 conference on Databases and Information Systems IV, pp. 18-39. IOS Press, Amsterdam.
24. Guizzardi, G., Falbo, R.A., Guizzardi, R.S.S., 2008. Grounding software domain ontologies in the Unified Foundational Ontology (UFO): the case of the ODE software process ontology. In: XI Iberoamerican Workshop on Requirements Engineering and Software Environments, pp. 244–251.
25. Guizzardi, R.S.S., 2006. Agent-Oriented Constructivist Knowledge Management. Centre For Telematics and Information Technology PhD.-thesis series, The Netherlands.
26. Hafsi, M., Assar, S., 2016. What enterprise architecture can bring for digital transformation: An exploratory study. In 2016 IEEE 18th Conference on Business Informatics (CBI), vol. 2, pp. 83-89. IEEE.
27. Hinings, B., Gegenhuber, T., and Greenwood, R., 2018. Digital innovation and transformation: An institutional perspective. Information and Organization, 28(1), pp.: 52-61.
28. Königs, A., Schürr, A., 2006. Tool integration with triple graph grammars-a survey. Electronic Notes in Theoretical Computer Science, 148(1), pp.113-150.

29. Krimpmann, D., 2015. IT/IS organisation design in the digital age – A literature review. International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, 9(4), pp.1189–1199.
30. Küster, J. M., Abd-El-Razik, M., 2006. Validation of model transformations - first experiences using a white box approach. In: MoDELS Workshops, LNCS, vol. 4364, pp. 193–204.
31. Linderoth, H. CJ, Jacobsson M., and Elbanna A., 2018. Barriers for Digital Transformation: The Role of Industry. In Australasian Conference on Information Systems, vol. 48.
32. Loebbecke, C., Picot, A., 2015. Reflections on societal and business model transformation arising from digitization and big data analytics: A research agenda. Journal of Strategic Information Systems, 24(3), pp.149–157.
33. MacDonald, T.J., Allen, D., and Potts, J., 2016. Blockchains and the boundaries of self-organized economies: Predictions for the future of banking. In Banking beyond banks and money, pp. 279-296. Springer, Cham.
34. Mangematin, V., Sapsed, J., and Schüßler, E., 2014. Disassembly and reassembly: An introduction to the special issue on digital technology and creative industries. Technological Forecasting and Social Change, 83, pp.1–9.
35. Maravilhas, S., Martins, J., 2019. Strategic knowledge management a digital environment: Tacit and explicit knowledge in Fab Labs. Journal of business research, pp. 353-359.
36. Moreira, M. E., 2017. Agile Enterprise. Apress.
37. Naudet, Y., Latour, T., Guédria, W., and Chen, D., 2010. Towards a Systemic Formalisation of Interoperability. Computers in Industry. 61, pp.176–185.
38. Nonaka, I., Hirose, A., 2018. Introduction to the Concepts and Frameworks of Knowledge-Creating Theory. In Knowledge Creation in Community Development, pp. 1-15. Palgrave Macmillan, Cham.
39. Nonaka, I., Kodama, M., Hirose, A., and Kohlbacher, F., 2014. Dynamic fractal organizations for promoting knowledge-based transformation–A new paradigm for organizational theory. European Management Journal, 32(1), pp. 137-146.
40. Nonaka, I., Toyama, R., and Hirata, T., 2008. Managing Flow: A Process Theory of the Knowledge-Based Firm. New York: Palgrave Macmillan.
41. Nonaka, I., von Krogh, G., 2009. Perspective—tacit knowledge and knowledge conversion: Controversy and advancement in organizational knowledge creation theory. Organization Science, 20, pp.635–652.
42. Oleśków-Szłapka, J., Stachowiak, A., 2018. The Framework of Logistics 4.0 Maturity Model. In International Conference on Intelligent Systems in Production Engineering and Maintenance, pp. 771-781. Springer, Cham.
43. Oliva, F. L., Kotabe, M., 2019. Barriers, practices, methods and knowledge management tools in startups. Journal of Knowledge Management.
44. Panetto, H., Iung, B., Ivanov, D., Weichhart, G., and Wang, X., 2019. Challenges for the cyber-physical manufacturing enterprises of the future. Annual Reviews in Control.
45. Parr, T., 2012. Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. Pragmatic Bookshelf.

46. Pereira, M., Fonseca, J., and Henriques, P., 2016. Ontological approach for DSL development. Computer Languages, Systems&Structures, pp. 35–52.
47. Pérez-Luño, A., Alegre, J., and Valle-Cabrera, R., 2019. The role of tacit knowledge in connecting knowledge exchange and combination with innovation. Technology Analysis & Strategic Management, 31(2), pp. 186-198.
48. Polanyi, M., 1966. The tacit dimension. Chicago: University of Chicago Press.
49. Poletaeva, T., Babkin, E., Abdulrab, H., 2014. Ontological Framework Aimed to Facilitate Business Transformations, 1st Joint Workshop ONTO.COM co-located with 8th International Conference on Formal Ontology in Information Systems (FOIS 2014). Vol. 1301.
50. Poletaeva, T., Abdulrab, H., Babkin, E., 2016. From the Essence of an Enterprise Towards Enterprise Ontology Patterns. In Enterprise Engineering Working Conference, pp. 118-131. Springer, Cham.
51. Quirino, G.K., Nardi, J.C., Barcellos, M.P., Falbo, R.A., Guizzardi, G., Guarino, N., Bochicchio, M., Longo, A., Zappatore, M.S., Livieri, B., 2015. Towards a Service Ontology Pattern Language. In: 34th International Conference ER 2015. LNCS, vol. 9381, pp. 187-195. Springer International Publishing, Switzerland.
52. Reis, J., Espírito Santo, P., and Melão, N., 2019. Artificial Intelligence in Government Services: A Systematic Literature Review. In World Conference on Information Systems and Technologies, pp. 241-252. Springer, Cham.
53. Ruy, F.B., Falbo, R.A., Barcellos, M.P., Guizzardi, G., 2015. Towards an Ontology Pattern Language for Harmonizing Software Process related ISO Standards. In: 29th Annual ACM Symposium on Applied Computing, pp. 388-395.
54. Sandkuhl, K., Fill, H.G., Hoppenbrouwers, S., Krogstie, J., Matthes, F., Opdahl, A., Schwabe, G., Uludag, Ö. and Winter, R., 2018. From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. Business & Information Systems Engineering, 60(1), pp.69-80.
55. Schilhab, Theresa. Derived embodiment in abstract language. Springer, 2017.
56. Schmidt, R., Zimmermann, A., Möhring, M., Nurcan, S., Keller B., and Bär, F., 2015. Digitization–perspectives for conceptualization. In European Conference on Service-Oriented and Cloud Computing, pp. 263-275. Springer, Cham.
57. Sprinkle, J., 2016. A safe autonomous vehicle trajectory domain specific modelling language for non-expert development. Proceedings of the International Workshop on Domain-Specific Modeling, pp. 42-48.
58. Ulitin, B., Babkin, E. and Babkina, T., 2018. Ontology-based DSL development using graph transformations methods. Journal of Systems Integration, 9(2), pp.37-51.
59. Ulitin, B., Babkin, E., 2017. Ontology and DSL co-evolution using graph transformations methods. In: Lecture Notes in Business Information Processing Issue 295: Perspectives in Business Informatics Research, pp. 233-247. Springer.
60. Ulitin, B., Babkin, E., Babkina T., 2019. Automated formal verification of model transformations using the invariants mechanism. Lecture Notes in Business Information Processing Issue 365: Perspectives in Business Informatics Research, pp. 59-73. Springer.

61. Ulitin, B., Babkin, E., Babkina T., 2016. Combination of DSL and DCSP for decision support in dynamic contexts. Lecture Notes in Business Information Processing Issue 261: Perspectives in Business Informatics Research, pp. 159–173. Springer.
62. Uschold, M., King, M., Moralee, S. and Zorgios, Y., 1998. The Enterprise Ontology. The Knowledge Engineering Review, 13(1), pp. 31–89.
63. Usman, Z., Young, R. I. M., Chungoora, N., Palmer, C., Case, K. and Harding, J. A., 2013. Towards a Formal Manufacturing Reference Ontology. International Journal of Production Research, 51(22), pp.6553–6572.
64. Ustundag, A., Cevikcan, E., 2017. Industry 4.0: managing the digital transformation. Springer.
65. van Gils, B., Proper, H.A., 2018. Enterprise Modelling in the Age of Digital Transformation. In IFIP Working Conference on The Practice of Enterprise Modeling, pp. 257-273. Springer, Cham.
66. Wang, H., Wang, X., Zhang X., 2017. Dynamic resource allocation for intermodal freight transportation with network effects: Approximations and algorithms. Transportation Research Part B: Methodological, pp. 83-112.
67. Westerman, G., Bonnet, D., McAfee, A., 2014. The nine elements of digital transformation. MIT Sloan Management Review, 55(3), pp.1-6.
68. Yoon, K. S., 2012. Measuring the Influence of Expertise and Epistemic Engagement to the Practice of Knowledge Management. International Journal of Knowledge Management, 8(1), pp. 40–70.
69. Zhang, S., Boukamp, F. and Teizer, J., 2015. Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA). Automation in Construction, 52, pp.29-41.