THE IMPORTANCE OF ONTOLOGY IN SOFTWARE ENGINEERING EDUCATION PROGRAMS

Zeynep Altan

Beykent University, Engineering-Architecture Faculty Software Engineering Department, Istanbul

zeynepaltan@beykent.edu.tr

Abstract: Since late 1970s (Yordon Structured Design, DeMarco Structured Analysis) and early 1980s (Structured System Analysis and Design Method -SSADM) to early 1990s (Booch Method), software developing methodologies have planned to model system building. As the real world problems became more complex and software industry became more complicated, software engineering education has been a necessity as a separate discipline. The Guide to the Software Engineering Body of Knowledge (SWEBOK) was published in 2004 after the description of a sequence of software engineering standards. These standards began to be specified in 1976 by IEEE Computer Society. On the other hand, the radical changes on the context of software problems arose object technology. The revolution of object technology took many years. As the consequence of different practices, a new software development approach Model-Driven Engineering (MDE) developed by Object Management Group (OMG) focused on models as the primary artifacts for development process. The processes have been implemented with transformations mapping information from one model to another. The integration of knowledge in different models depends on the existence of explicit declarative semantic models. Therefore, increasing diversity and complexity of information gave rise to increasing interests on ontologies. These formal domain models have been linked to each other on the Web. The linked ontologies provide shared terminologies for different applications. In 2013, the Knowledge Areas (KAs) in SWEBOK 3.0 have also been rearranged according to these complications of the real world problems. Formal and informal solution techniques, latest development methods and new technologies have been included to guide the software engineering education programs.

Keywords: Ontology, Semantic Web, Model Driven Engineering SWEBOK, Knowledge Area

Introduction

In 2009, Tim Berners Lee at TED conference explained the importance of Linked Data as follows[1]: "At the present day, the huge amount of data, in other words open data, must be combined each other as a different system or technology. It is required to give meaning to everything to obtain the required solution. This is the semantic web movement and Wikipedia is the first project of that movement". Knowledge based applications on MDE uses ontologies to share the information (Gruber, 1993; Uschold &Gruninger,1996; Parreiras& Gröner& Walter et.al. ,2013). Although the primary applications on ontology have been accepted as the research area of the artificial intelligence experts, it came into a part of industrial software engineering applications on the solution today's software problems. One of the knowledge areas of the Software Engineering education program on SWEBOK taxonomy (Bourgue & Fairley, 2014) is Mathematical Foundations KA and it supplies well understood solutions to the real world problems with an unambiguous logic. The search of the formal systems related with the completeness, in other words preciseness is the basis of discrete mathematics. Since the software problems for different applications requires distinct abstraction, Mathematical Foundations is a separate KA of the software engineering undergraduate programs. Software Engineering Models and Methods as another KA of the SWEBOK taxonomy defines the modelling as an abstraction of any software component, and this component has more than one abstractions. The aggregation of these abstractions compose the software model. When the developed model has been reused, coherence with the new context will be verified by the inferences constituted from simplifications (abstractions).

Ontology is a conceptualization corresponding to any domain as machine readable entities, attributes, and well understood rules, in other words axioms (Gruber, 2008; Tobias, 2011; Aßmann &Andreas&Christian, 2010). Various ontologies that have aimed at modelling the concepts support to reuse and to extend these independently developed ontologies. Information modelling as one of the modelling types of Software Engineering Models and Methods KA is an abstraction as semantical, in other words conceptual knowledge model and it includes the senses,

[1] https://www.ted.com/talks/tim_berners_lee_on_the_next_web?language=en

properties and constraints that formalize the real world outlook of knowledge. Finally, standard knowledge representation for semantic web can be determined by intelligent access to the heterogeneous and distributed information with better interaction of human-computer on the web. It is not possible to implement this using UML diagram that is one of the structural modelling types of the same KA (Software Engineering Models and Methods). Logic based knowledge representation used widely for ontology languages is represented by Description Logic (DL). (Sowa, 2008; Krötzsch&Simancík&Horrocks, 2014), and it belongs to the same KA of SWEBOK taxonomy. First order predicate logic supplies the modeling of all relationships among the objects as set elements. For example, World Wide Web Consortium (W3C) has been defined by DL syntax with the class/concept constructors of OWL2 (Web Ontology Language)[2]. Disambiguation of the ontological studies can also be obtained by semantic definitions which the complexity has been reduced. This is possible with theoretically well-defined reasoning algorithms and well explained formal properties. eScience, geographical investigations, engineering, medicine, biology, defense industry projects are some of the application domains that the ontologies have extensively been used. Software engineering education programs must be a separate discipline since the concepts summarized above. In this study, the reasons of teaching ontology in undergraduate software engineering programs due to the need to huge data for present day software products have briefly been explained.

Software Engineering Undergraduate Programs

The number of software engineering department in Turkey has been 16 in the 2015-2016 academic year. Three of them with the inclusion of Technology Faculty Departments are at Karadeniz Technical, Fırat and Celal Bayar Universities. Three of the foundation universities are held in North Cyprus (Doğu Akdeniz, Lefke and Yakın Doğu Universities), one is in Ankara (Atılım University), two are in Izmir (Izmir Ekonomi and Yaşar Universities) and one is in Mersin (Toros University). In Istanbul there are six Software Engineering Departments at the Universities Bahçeşehir, Beykent, Maltepe, Aydın, Işık and Sabahattin Zaim. Since the number of software engineering departments have rapidly increased in the last 10 years, it is important to prepare and to update the education programs of software engineering departments. The first study about the organization of software engineering undergraduate programs began in 1987 with the conference "The Conference on Software Engineering Education and Training - CSEET". These conferences devised by SEI (Software Engineering Institute) repeated with different periods, until they were completely complemented. Another important study to prepare the education programs is the project SWEBOK started in 1998. This project aims to the software engineering standards supported by IEEE (Altan, 2010). SWEBOK Guide released in 2004 was composed of 10 KAs. This guide has been expanded in parallel with the raise of problem sizes to appraise the software engineering on the world measurement. Table 1 shows all KAs with the reorganized 5 KAs. The undergraduate programs at software engineering departments must be updated in accordance with the new requirements since the comprehensive complexity of the software problems (Altan, 2015). Moreover, to distinguish software engineering undergraduate programs from computer engineering undergraduate programs, it is important to realize the KAs in SWEBOK guide.

Table 1 : SWEBOK 3.0 Knowledge Areas (KAs)

1	Software Requirements (2004)
2	Software Design (2004)
3	Software Construction (2004)
4	Software Testing (2004)
5	Software Maintenance (2004)
6	Software Configuration Management (2004)
7	Software Engineering Management (2004)
8	Software Engineering Process (2004)
9	Software Engineering Models and Methods (2004)
10	Software Quality (2004)
11	Software Engineering Professional Practice (2013)
12	Software Engineering Economics (2013)
13	Computing Foundations (2013)
14	Mathematical Foundations (2013)
15	Engineering Foundations (2013)

Since a special emphasis is put on the information sector both in our country and in European countries in accordance with Europe 2020 Targets, software engineers will increasingly be demanded persons. Moreover researches show that the software sector gets possession the most labor force when compared with other sectors. We also know that information and telecommunication technologies contributes the maximal productivity on all over the world. In this context, young graduates can employ at all domains from social media, business, and

[2] http://www.w3.org/TR/owl2-profiles

economics to education and production sectors to apply present-day technologies; besides they can develop new software products and new application domains. On the other hand, it is a fact that well-qualified labor supply is insufficient in Information Technology (IT) industry in Turkey. Although the sector employs 160,000 persons, more than half of them are the labor force with poor quality (Türkiye Bilişim Derneği, 2016). Furthermore, digital data will get around to 40 billion terabyte since the considerably growing data annually obtained from device, sensor, geographic applications, web and social media. Big data as a new concept arose from this huge increment beside the usual data sources, and the IT sector made an investment to the new technological developments to obtain consequences semantically. The fundamental reason of this research comes from the management of big data.

The Challenges on Software Engineering

The main problem of the software engineers is to produce working software artifacts according to the customer needs, in time and within the budget. This is the fundamental software quality criteria on the evolution of software engineering discipline. Software crisis gradually have been increasing since the first software products developed. In 1995, the Standish Group[3], a small IT Research Company, published "The CHAOS Report" about over 8,000 software projects. Each project was accepted to be successful if it met all the software product quality criteria. While only 16.2% of the projects were successful and 31.1% of projects were cancelled. 52.7% of them were challenged. Any project has usually been confirmed as canceled before delivered anything; in other words, before completion or never implemented. The challenged projects were either completed or operational with over-budget, over-schedule and/or estimate fewer featured. Standish Group 2015 CHAOS Report gives us 29%, 19%, 52% results for successful, cancelled and challenged projects sequentially. More than 50,000 projects around the world ranged all project sizes including re-engineering implementations give the similar results with the first CHAOS Report. During the 21 years software developing methodologies have changed to cope with the complex real world problems. Cancelled and challenged IT Projects are higher in Standish Group Reports than in other reports since the detailed evaluation criteria reflects the critical quality problems with software.

At the present time, the customers demand the products with more functionality delivering in shorter time. Moreover traditional software development processes starting with waterfall projects and the evolution of waterfall by time as prototyping, incremental, spiral and unified processes became a challenging issue for software engineering discipline. Too many requirements and scope changes, lack of management skills, much development costs than planned, lack of technical skills, unessential system anymore are other causes of unsuccessful software projects. By time three measurements on time, on budget, on satisfaction to succeed a project increased to six adding the measurements on target, on goal and on value. This is the success criteria of the Project Management Institute (PMI) that most of the project management processes have been doing from the PMBOK Guide (4. Edition)[4] since 2012. Team working and skill level of the team members are important issues for a software project. However, most organizations don't emphasize enough in increasing the skills of their people. The collection of basic behaviors about how people work together is another success factor nowadays software projects. On the other hand, the users must be involved in the project starting from the requirements elicitation at the all developing phases of the product[5]. Optimization and organization of the projects reduce project overhead and business capability increases[6].

Software development is the process of generating software through successive phases. This process includes not only the actual writing of code, but also the preparation of requirements, the design of what is to be coded, and the confirmation of what is developed to meet the objectives (Presman, 2014; Sommerville, 2011). Before system development methods came into being, the development of a new system or any product was often carried out by using the experiences and intuitions of the management and technical personnel. As the complexity of software products long ago made the need clear for some kind of development process, the waterfall model recorded in 1970 as the first public life cycle model. Since then, waterfall model describing linear and sequential development method has been a popular version of the software development life cycle for software engineering. Figure 1 shows the distinct goals of the waterfall for each development phases. Agile solutions have emerged as a perfect approach for the software development. These projects have firmly been shared with users and delivered continuously (Braude &Bernstein, 2010). Furthermore, the solutions of some software problems can be achieved by model based engineering frameworks, and the usage of frameworks supply the abstractions in a lower level than others.

[3] www.standisshgroup.com
[4] http://www.pmi.org/
[5] http://agilemanifesto.org
[6] www.chaostuesday.com

The basics of the software development processes must be given as individual courses in the software engineering undergraduate programs. The lectures can be carried out both in department electives and in core courses of the curriculum structure. This property distinguishes the computer engineering undergraduate program from software engineering education programs.

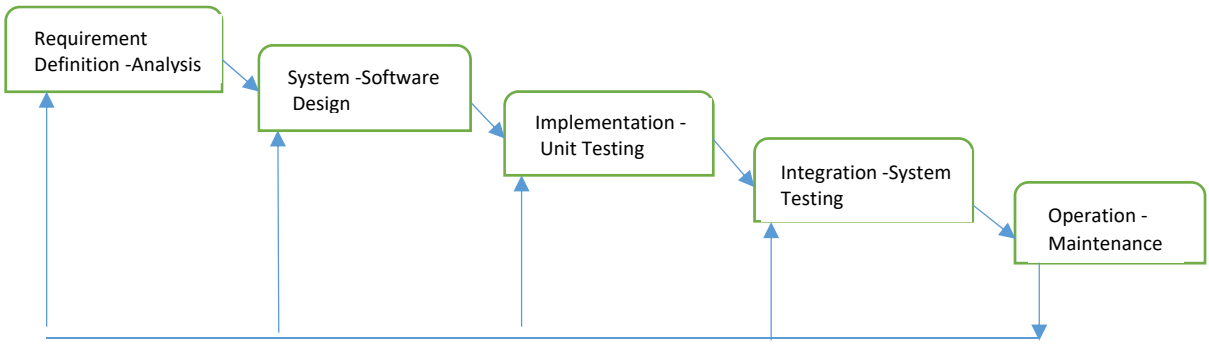
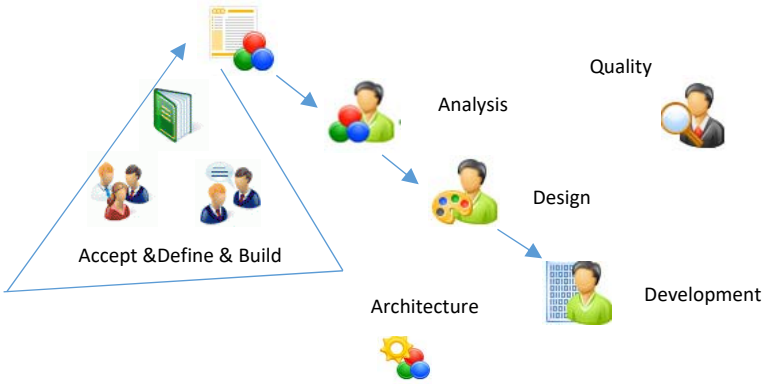


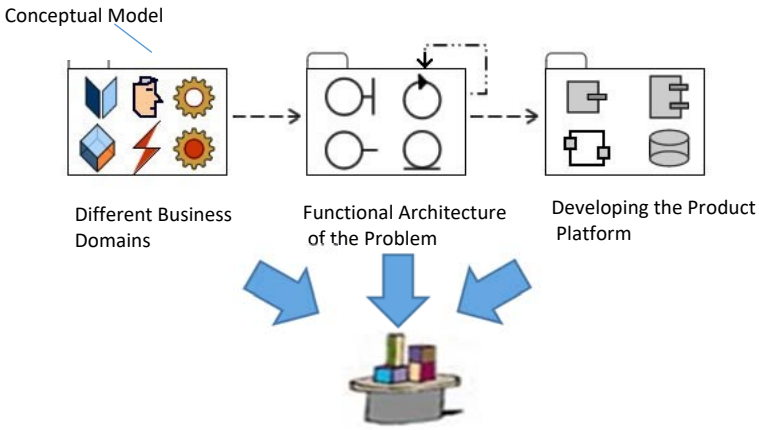
Figure 1: General overview of waterfall model

Model-Driven Software Development as a Software Engineering Paradigm

Model-Driven Software Development is a branch of software development approach based on the idea of developing software from domain-specific models (Perisic, 2014; Küster,2011). The primary aim of this development technique is to increase the productivity and maintainability of software. These can be achieved by raising the abstraction levels of physical system represented in a general purpose language. Such development processes are called as domain-specific models written in high level. Therefore developers can concentrate on application logic rather than the complexity of low-level implementation details. This is contrast to traditional software development practices where modelling is commonly used for documentation. Moreover, communication purposes and the final product broadly differs from the models representing the problem. We can not only distinguish the differences between procedural and declarative software engineering processes, we can also see the abstraction levels and the platforms to be integrated in Figure 2. The abstract syntax has been used at each abstraction level in Figure 2-b. The developers describe the problem and its solution at different abstraction levels using model based engineering.



a) General Overview to Agile and Phase Software Development Models



b) Model Driven Engineering Processes as Bottom-Up with work-units defined by their role on targeted artifact

Figure 2: Phase Based Modern Software Models and Model Driven Software Engineering

Model Driven Engineering (MDE) describes tasks directly from development flows. Therefore, a declarative and bottom-up approach is implied contrary to the activity based top down development processes. The effects are built-in and derived processes instead of textual and graphical developing phases of the activity based models. In Figure 3, we can clearly see the transition from conventional development processes to more effective processes. Domain-specific models constitute requirements-driven and architecture centric development approach (Figure 2-b) instead of document-based and code-centric problem solutions (Figure 2-a)

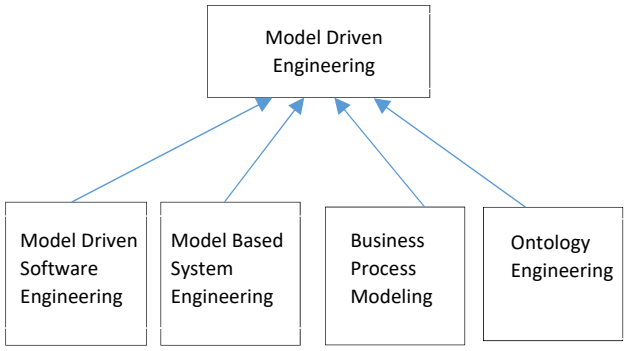


Figure 3: Models Relationships of Model Based Engineering

Model Driven Architecture (MDA)[7] developed by Object Management Group (OMG) is the special case of Model Based Software Engineering, and it is divided into three models as Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM) (Figure 4). CIM has to be used to clarify

[7] http://www.omg.org/mda/presentations.htm

the problem domain focusing on the environment and the specific requirements of the system. Briefly, the structural details of the product are hidden at this layer. Fine grained artifacts of this layer can be descripted as events and communications, objects and features, processes and activities and objects and agents. These are defined independently and followed as PIM. PIM is input to model the problem. At this phase, the developers define functional architecture of the system by specific technologies. All the information needed is symbolized with boundaries, controls, entities and services to describe the behavior of the system in a platform independent way by different platforms. The solution results as PSM. To constitute specific components of the developing system, PSM is applied to the convenient technologies. PSM combines the specifications in the PIM with the details that specify how the system uses a particular type of platform. Fine-grained artifacts of this layer is middleware, clients, servers, databases as specific platform characteristics. To get the well-defined software product, it is important not to confuse these three layers (Stahl & Völter, 2006).

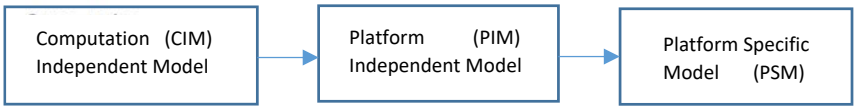


Figure 4: Architectural Layers of MDA with respect to abstractions

INCOSE (International Council on Systems Engineering) defined its 2020 vision as Model Based Systems Engineering, thereby announcing systems engineering as an independent discipline which works in conjunction with software engineering. This view approves the existence of formal description languages in domain-specific conceptual modelling in software engineering. Any description language as a text-based language, which acquires from discrete mathematics and first order propositional calculus, constitutes the conceptual modelling as an intermediate phase prior moving from the visual solution towards programming language solution (Altan, 2015).

Semantic Extraction of Ontologies

In his PhD theses Parreiras (2011) predicts that the software engineering domain will depend on the dependency between MDE and ontology technologies. In fact, real-world elements form the structure of models and language semantics comprise metamodels as a step of MDE. The second step is transformations between languages. UML (Unified Modelling Language) and OWL are two examples of the transformation languages. Ontology as an explicit, formal and declarative semantic model gives the integration of knowledge and services (Bunge, 1977). While UML has extendibility and modularity properties, these attributes has not been presented in most OWL languages. Ontology is also an urgency to develop contemporary software systems since the inclusion of huge knowledge. Basic introduction to knowledge and ontology is given in Sowa's book (2000). Logical languages, frame based languages and graph based languages are three different representation style of knowledge (Baclawski& Kokar& Kogut et.al, 2002). The first language expresses knowledge in terms of logical statements. An example of such knowledge representation is Knowledge Interchange Format (KIF). The second classification is similar to object oriented database languages. The last one includes semantic networks and conceptual graphs. Sowa as one of the pioneers of conceptual graph theory proposed to create a logical system to represent natural language semantics, and defined it being a graph representation for logic based semantic networks. It is also a compromise between a formal language and a graphical language. First order logic semantics has been used to translate the graphs. The modern frame-based systems denote the implementation of knowledge base (KB) systems and DL that can be applied in several areas cover the theoretical aspects (Baader et.al., 2003). We can integrate model driven development and semantic web using OWL ontologies, for example Protage[8]. Therefore, the resulted object design patterns will reduce the complexity and will increase the productivity. The templates which are the form of knowledge base also supply the reliability. A standard DL knowledge base consists of ABox and TBox components. The ABox contains extensional knowledge about the domain of interest. In other words, it is possible to assert the individuals with ABox axioms. TBox is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts. In other words it is possible to determine the relationships between objects with TBox axioms. The following example describes a simple domain ontology about online library records in terms of DL[9].

[8] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library#OWL_ontologies

[9] It has been accepted the class hierarchy of the ontology has previously been constituted.

Lecturer ⊑ ∃isMemberOf.University ⨅ ∃writes.Article ⨅ ∃teaches.Course	[TBox Axiom]
Course⊑ ∃ isArrangedby.Department	[TBox Axiom]
University ⊑ ∃isLocatedIn.City	[TBox Axiom]
University(Beykent,KavramVacationSchool,Sakarya),Department (computerEngineering)	[ABox axiom]
Article(computer science)	[ABox axiom]
Lecturer (Zeynep Altan), University (Beykent), City (Istanbul)	[ABox axiom]
isLocatedIn (Beykent,Istanbul), teaches(ZeynepAltan, computerEngineering)	[ABox axiom]
isMemberof (ZeynepAltan, Beykent)	[ABox axiom]
writes(ZeynepAltan, computerScience), Lecturer(Ali A.)	[ABox axiom]
isLocatedIn (Ali A., ¬Kavram), isLocatedin(Ali A., ¬Sakarya)	[ABox axiom]

Based on the KB above, any user may search all lecturers at Beykent University with the following DL query:

∃isMemberOf.{Beykent}

The answer of this request is *Zeynep Altan* in case of short KB. The conclusion also includes the lecturer *Ali A.* for the complete KB. We have to close the domain of a class with an example. We accepted that the class University is equivalent to the set of other defined individuals. Moreover, we can assert that university of the article is the same as university of the lecturer as the following:

course(?x) ∧university(?y)∧article(?z)∧ isMemberOf(?x,?y) →isArrangedby(?x,?z) →isLocatedin(?y,?z)

It is possible to reuse and extend these templates. When we define the super class, for example University class, it includes all universities in all cities of the studied countries. In other words, it includes a list of existing individuals of this type. This superclass and its rule is*MemberOf* is usable for other types of university domain. For example, for student relationships, Erasmus programs etc. Finally, the templates can be reused in other ontologies

Conclusion

Complex software systems play an important role in business and everyday life. Since the technologies and practices in the informatics area are increasingly developing, the education programs on software engineering need to include both technical and non-technical abilities with details for the industrial software engineering. Therefore an essential task in teaching software engineering is the incremental improvement and enhancement of courses. The complexity and evolution of software engineering education is an indisputable reality; therefore more transparent and traceable problem solving methods are valuable than traditional ones. MDE is one of the contemporary approximation to develop software products unlike traditional development techniques which tend to be code-centric. As a present day example, the usage of models at all levels of the software development life-cycle can be emphasized by MDA standard. This change in software engineering has impacted both the construction way of the software products and the teaching way the software engineering education. Moreover software engineering standards have a significant emphasis on body of knowledge that should be integrated in a software engineering curriculum. Besides, KAs in SWEBOK plays the most important role in updating education programs. In this paper, the impact of the MDE approach has been explained on software education. As a constituent of MDE, semantic web techniques enable new software engineering capabilities. It is possible to identify various ontologies based on all phases of software life cycle and their application scope. Because of the attractively of OMG's MDA software development approach, new KAs in SWEBOK must definitely be added to the software engineering undergraduate programs without delaying.

References

Altan, Z. (2010). Beykent Üniversitesi Yazılım Mühendisliği Lisans Programı, Akademik Bilişim (sf.573-580) XII. Akademik Bilişim Konferansı Bildirileri.

Altan, Z. (2015). Yazılım Mühendisliği Bilgi Alanlarındaki Gelişmelerin Eğitim programlarına Uyarlanmasının Önemi, Yönetim Bilişim Sistemleri Dergisi (sf.11-20).

Aßmann, U.; Andreas, B.A.; Christian W.(eds.) (2010). Reasoning Web. Semantic Technologies for Software Engineering, 6th Inter. Summer School. Lecture Notes in Computer Science, 6325.

Baader, F.; Calvanese,D.; McGuinness, D.; Nardi,D.; Patel-Schneider, P.(eds) (2003). The Description Logic Handbook Theory, Implementation and Applications, Cambridge Press.

Baclawski, K.; Kokar, M.K.; Kogut, P.A. et.al. Extending the UML Modified Modelling Language for Ontology Development Software and Systems Modeling 1(2):142-156 , 2002.

Bourgue, P.; Fairley, R.(eds.) (2014). SWEBOK version 3.0 Guide to the Software Engineering Body of Knowledge. IEEE Computer Engineering Society.

Braude, E.J.; Bernstein, E.B. (2010). Software Engineering Modern Approaches. John Wiley & Sons.

Bunge, M. (1977). Treatise on Basic Philosophy: Ontology I: The Furniture of the World. Springer.

Gruber, T. (1993). A translation approach to portable ontology specification. Knowledge Acquisition. (5:pp.199-220), Academic Press Ltd. London, UK.

Gruber, T. (2008). Ontology Encyclopedia of database systems. In: Liu, L.; Özsu, M.T. (eds), Encyclopedia of Database Systems. Springer-Verlag.
Krötzsch, M.; Simancík, F.; Horrocks, I. (2014). Description Logics. IEEE Intelligent Systems, 29(1). (pp.12-19).
Küster, J. (2011). Model-Driven Software Engineering Foundations of Model-Driven Software Engineering, IBM Research 2011.
Parreiras, F.S. (2011). Marrying Model-Driven Engineering and Ontology Technologies: The Two Use Approach. Ph. Degree Thesis, Koblenz-Landau University.
Parreiras, F.S.; Gröner, G.; Walter, T., et.al. (2013). Model-Driven Software Development. in Ontology Driven Software Development. Pan J.Z.,Stab S.,Aßmann U.,Ebert J., ZhaoY.(eds). (pp.21-50), Springer.
Perisic, B. (2014). Model Driven Software Development – State of The Art and Perspectives. Infoteh-Jahorina. (13).(pp.1237-1248).
Presman, R.S. (2014). Software Engineering Practitioner's Approach 8th Edition, McGraw-Hill.
Sommerville, I. (2011). Software Engineering 9. Edition. Pearson.
Sowa, J.F. (2008). The role of logic and ontology In language and reasoning. Theory and Applications of Ontology: Philosophical Perspectives, Poli, R. and Seibt, J. (Eds.), (pp. 231-263).
Sowa, J.F.(2000). Knowledge Representation Logical, Philosophical and Computational Foundations. PWS Pub.
Stahl, T.; Völter, M. (2006). Model-Driven Software Development, Wiley.
Tobias, W. T. (2011). Bridging Technological Spaces: Towards the Combination of Model-Driven Engineering and Ontology Technologies. PhDissertation. Universität Koblenz-Landau.
Türkiye Bilişim Derneği (TBD). (2016) .2015 Değerlendirme Raporu. Rapor 10.
Uschold, M.; Gruninger, M. (1996). Ontologies: Principles Methods and Applications, The Knowledge Engineering Review. 11 (2) (pp. 93–155), Cambridge University Press.

.

