2017

The Global Crisis as Digital Transformation Motivator: from Lifecycle Optimization to Efficient Implementation Series

Sergey V. Zykov
Higher School of Economics; MEPHY, szykov@hse.ru

Ramis Gabeydulin
Higher School of Economics; State Research Institute of Aviation Systems, rgabeydulin@hse.ru

The Global Crisis as Digital Transformation Motivator: from Lifecycle Optimization to Efficient Implementation Series

Full Paper

Sergey V. Zykov
National Research University Higher School of Economics,
National Nuclear Research University MEPHI
szykov@hse.ru

Ramis Gabeydulin
National Research University Higher School of Economics,
State Research Institute of Aviation Systems
rgabeydulin@hse.ru

Abstract

It is generally known that software system development lifecycle (SSDL) should be managed adequately. The global economy crisis and subsequent depression have taught us certain lessons on the subject, which is so vital for digital transformation, for Industry 4.0. The paper presents the adaptive methodology of enterprise SSDL, which allows to avoid "local crises" while producing large-scale software. The methodology is based on extracting common ERP module level patterns and applying them to series of heterogeneous implementations. The approach includes a lifecycle model, which extends conventional spiral model by formal data representation/management models and DSL-based "low-level" CASE tools supporting the formalisms. The methodology has been successfully implemented as a series of portal-based ERP systems in ITERA oil-and-gas corporation, and in a number of trading/banking enterprise smart applications for other enterprises. Semantic network-based air traffic planning system, and a 6D-model-driven nuclear power plant construction support system are currently in progress.

Keywords

Global Crisis, Industry 4.0, Methodology, Enterprise Software development lifecycle, DSL.

Crisis in Software Development

The “crisis” phenomena occurred in software development relatively long ago, approximately since 1960-s (Tomaiko 1989; Zykov 2016). Software product lifecycle, which the industry had just started moving toward, was anarchic in many ways, since a uniform, systematic approach had been absent. At that time, software development did not allow precise variation of the “project triangle” parameters. In fact, the software had been developed in an “artisan” way, with a build-and-fix approach as the core “methodology”. Thus, systematic approach to SSDL, and responsibility for the deliverables should be required. The following decade revealed that the software development process started becoming rather a science than an art; however, it had not become a production yet, due to imperfect technologies. The era of unique, “hand-made” software projects from certain gifted programmers had passed away. Large software R&D centers appeared, one of the most well known examples of which is the Software Engineering Institute of the Carnegie-Mellon University (www.sei.cmu.edu/). The value of software, as compared to hardware, had increased tremendously. Mission-critical software systems appeared.

However, the software crisis, which started in the 60-s, lasted much longer and had a deeper nature than that in material manufacturing industries. Lacking “universal” methodology, the so-called “silver bullet” for software development, explicitly indicated that the crisis has not been overcome, and that the “depression” started. To conquer the crisis, we need to optimize the SSDL by systematically approaching all of its processes. The methods of software engineering (SE) methods and tools can help in this case, since the discipline approaches software development issues in a systematic way. The SE approach is chiefly oriented on “serial production” of large-scale, complex, high quality, architecturally heterogeneous, interoperable software systems (Schach 2011; Sommerville 2016). The other architectural aspects include portal-based software systems, remote services, etc. The system quality is measurable, by the following “dimensions”: reliability, security, fault tolerance, ergonomics, usability, reuse, documentation, maintainability. Heterogeneous software systems imply versatile architectures, data bases and warehouses, as well as structure degree (Kalinichenko and Stupnikov 2009).

SDDL Methodology

We recommend following a distinct and logically sound process of step-by-step software functional specification, including conceptual modeling, analysis and design (with software operational parameter monitoring), prototyping, implementation and maintenance. Analyzing the SSDL, we arrive to a conclusion that maintenance is its most specific phase, as compared to material production. Software reuse processes are essentially iterative and incremental. Software changes are more serious and radical than the changes in material objects. The software on the whole also has got a number of fundamental differences as compared to material production. For example, the SSDL is often essentially shorter than a material object lifecycle, since software ages much faster, and its retirement is economically preferable to maintenance. The accumulated individual/team experience in software development does not always result in system quality increase due to rapid changes of complex platforms. Software development has both similarities and essential differences as compared to material production. However, industrial production of large and complex software systems with high quality, reliability, efficiency, usability and other operating parameters is possible under a rigorous science and technology basis of SE methods and tools.

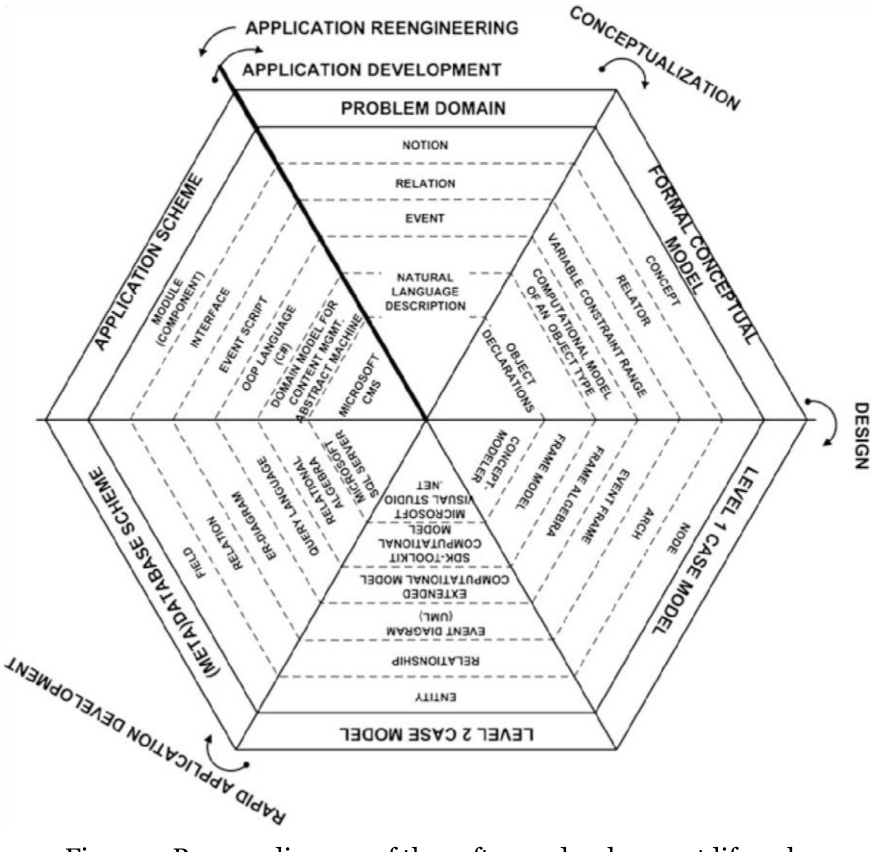

Figure 1. Process diagram of the software development lifecycle

Every SDDL stage can be optimized, including requirement analysis, product specification, design, implementation, maintenance and retirement. The SSDL optimization methodology is based on close integration of models and supporting enterprise-level methods and service tools. The models for problem domain and computing environment are built on rigorous formal theories, while those for other lifecycle stages are more heuristic and pragmatic. Therewith, the family of the software service tools used contains both traditional CASE and "lower" level instruments, which integrate the model and the software components in heterogeneous software systems. The generalized methodology of integrated software system development provides iterative bidirectional component-wise development of open, expandable heterogeneous software systems in global environment, allowing data consistency and integrity control.

During the SSDL, the transformation of the heterogeneous information systems are transformed from problem domain concepts to mathematical model data entities. Further, by means of original software toolkit (Fowler 1997; Zykov 2007; Zykov 2010; Zykov 2015a; Zykov 2015b; Zykov 2017; Zykov and Kukushkin 2013), the model is transformed into a complex semantic network and object-relational warehouses, managed by an abstract machine. Finally, we arrive to a well-formed layout of software system component interfaces and to an internet portal superstructure. The development levels are detailed in terms of entities, relationships, content definition/manipulation languages, and software tools.

Thus, the SSDL methodology is supported by a family of formal object models for content representation/management. The models incorporate fundamentals and methods of finite sequences, variable domains, semantic networks, and other theories (Barendregt 1984; Birnbaum, Forbus, et al. 2005; Guha and Lenart 1990; Curry and Feys 1958; Lenat and Reed 2002; Roussopulos 1976; Scott 1981; Wolfengagen 1999). Thus, the central aspect of mathematical and conceptual modeling, analysis and design of the software systems is shifting the SSDL paradigm from object-oriented to pure object approach, i.e. from IT to computing. Computing is a relatively new scientific subject, adequately representing complex, heterogeneous, changeable, and interactive problem domains in terms of objects and their environment (Wolfengagen 2010).

Currently, the multinational enterprises possess large, geographically distributed infrastructures, aimed at the same business goals. Each of the enterprises has accumulated a tremendous and rapidly increasing data burden, comparable to an avalanche. In certain cases, the data bulk exceeds petabyte size, and it tends to double every five years. Undoubtedly, management of such data is a serious challenge. The problem becomes even more complicated due to heterogeneous nature of the stored data, which varies from well-structured relational databases to non-normalized trees and lists, and to weak-structured multimedia data. The technology presented in the paper is focused at more efficient heterogeneous enterprise and uniform data management procedures. The technology involves a set of novel mathematical models, methods, and the supporting software engineering tools for object-based representation and manipulation of heterogeneous enterprise data.

An essential feature of the general ESS development framework is its two-way organization. The approach provides reverse engineering possibility both for ESS in general, and their components in particular. The practical value of the approach is provided by the verifiability of heterogeneous ESS components at the uniform level of the problem domain model, which is practically independent upon the hardware and software environment of the particular component. Therewith, a major theoretical generalization is a possibility of mathematically rigorous verification of the heterogeneous ESS components by a function-based model (Curry and Feys 1958; Scott 1981). The ESS engineering models are oriented at a promising "pure" objects approach, which is a strategy of .NET and Java technologies, where any program entity is an object. An essential benefit of the approach suggested is a possibility of adaptive, sequential "fine tuning" of ESS heterogeneous component management schemes in order to match the rapidly changing business requirements. Such benefit is possible due to the reverse engineering feature of the integrated general iterative framework of ESS development. The reverse engineering is possible down to model level, which allows rigorous component-wise ESS verification. Thus, conventional reengineering and verification can be enhanced by flexible correction and "optimization" of the target ESS in strict accordance with the specified business requirements. This is possible due to the suggested model-level generalization of the iterative, evolutionary ESS development framework.

Another benefit of the suggested ESS development framework is a possibility of building a "catalogue of templates for heterogeneous ESS", which is based on an integrated metadata warehouse, i.e., a "meta-snapshot" repository. Thus, the software development companies get a solution for storing relatively stable or frequently used configurations of heterogeneous enterprise software systems. The solution potentially allows avoiding the integration problems of "standard" ESS components and/or combinations, which have been obtained previously. The approach allows serious software engineering project savings for clients, provided the ESS developer's "meta-snapshot" repository already stores a similar or an analogous integrated solution to the system required. The above consideration clears the way for "meta-snapshot" repository development, which stores the chronological sequence of ESS solutions as a tree with the "baseline" and slight variations of ESS "branches". This is similar to version control CASE tools. The approach allows a reasonable selection of most valuable deliverables of the ESS lifecycle phases, and organization of similar solution "cloning". Therewith, the "clones" may be created for digital transformation of both different client enterprises and different companies of a single enterprise.

Semantic-oriented search mechanisms assist in revealing ESS "meta-snapshot" repository components, which provide the closest matching to the new requirements. The approach potentially allows terms-and-cost-effective and adequate transforming of the existing ESS components in order to match the new requirements with minimum corrections effort and, consequently, with minimum labor expenses. Therewith, the global perspective it becomes possible to reuse certain ESS components for current or new clients. Selection criteria for such "basic" components may be percentage of reuse, ease of maintenance, client satisfaction, degree of matching business requirements etc.

ITERA Oil-and-Gas Group: a Portal-Based Solution

The suggested methodology has been practically approved by development of Internet and Intranet portals in ITERA International Group of Companies. During the design stage, problem domain model specifications are transformed by the innovative ConceptModeller SDK to UML diagrams, then by Oracle Developer/2000 integrated CASE tool – to ER diagrams and, finally, into target IS and content storage schemes.

Using the suggested data model, the architectural and interface solution has been customized for enterprise resource management IS with content personalization for a wide spectrum of user and administrator types. To provide the required industrial scalability and fault tolerance level, the integrated Oracle design and implementation toolkit has been chosen to support UML and business process reengineering. A set of models have been constructed including problem domain conceptual model for enterprise content dynamics and statics as well as a model for development tools and computational environment in terms of state-based abstract machines, which provide integrated object-based content management in heterogeneous enterprise portals. For the model set, a generalized development toolkit choice criteria set has been suggested for information system prototyping, design and implementation. A set of SDKs has been implemented including ConceptModeller visual problem oriented CASE-tool and the CMS. According to the approach, a generalized interface solution has been designed for Internet-portal, which is based on content-oriented architecture with explicit division into front-end and back-end sides. Portal design scheme is based on a set of data models integrating object-oriented methods of management of data and metadata (or knowledge).The major implementations of portals in ITERA Group were: CMS for network information resources, official Internet site, and enterprise Intranet portal.

Distributed Trading Company: a Smart Domain-Driven Messaging System

A trading corporation used to commercially operate a proprietary Microsoft .NET-based message delivery system for information exchange between the headquarters and local shops. The system was client-server based. The client included a local database and a Windows-based messaging service, while the server side consisted of a Web service and central database. The operation/maintenance challenges were: complex client-side code refactoring; costly error localization/reduction; inadequate documentation; decentralized configuration monitoring/management for remote shops. To solve the problems mentioned, an approach based on domain-driven development (Evans 2003) and Domain Specific Languages (DSL) has been suggested. The approach included problem domain modeling and DSL development for managing objects of the problem domain. The DSL-based model helped to conquer problem domain complexity, to filter and to structure the problem-specific information. It also provided a smart, uniform approach to data representation and manipulation. We used an external XML-based DSL, which extended the scope of the enterprise application programming language. The methodology instance included the following steps:

DSL scope detection, problem domain modeling, DSL notation development, DSL restrictions development, and DSL testing.

The approach was client-side focused, since this is the most changeable and challenging task. The lifecycle model is iterative, and it the solution is based on a redesigned architecture pattern. The Windows service is a constant part of the smart application, which contains a DSL parser. The DSL parser input is a current message transfer map. The DSL scope included message transfer rules/parameters, and adding new types of messages. Different shops may have different configuration instances, which make the client side message processing/transfer structure. The next methodology stage was building semantic model of the objects handled by DSL. We got three types of the objects: messages, message transfer channels and message transfer templates. DSL describes object metadata, i.e., configurations and manipulation rules. Templates were core elements of the model, and channels were links between template instances. Templates and channels together make message maps. DSL described the maps, i.e. the static part of the model, while messages referred to its system dynamics and store the state.

Templates defined actions with messages, i.e. transform or route them. Templates were grouped into the IMessageProcessingPattern interface. The typical smart routing templates were: content-based router, filter, receiver list, aggregator, splitter, and sorter. We also produced a number of domain-specific templates for system reconfiguration, server interaction, etc. Channels were used for smart message management. In the graph of map messaging, templates are represented as nodes, while channels are arcs between certain templates. In our case, two types of channels were implemented: "peer-to-peer" channel and error messages channel. Based on DSL class model and implementation, messaging maps were built, which were later used by parser to generate system configuration. At this stage, DSL syntax and semantics were built. Each messaging map, generally, a script, was instantiated by a file. Messaging map was built as an XML document, which defined smart system configuration and contained templates for routing, message processing, transfer channels and their relationships. While parsing a messaging map, the parser created channel objects based on DSL channel descriptions. Then it configured the smart messaging system by creating message processing objects in a similar way. Finally, the parser instantiated the I/O channels, and creates the relationships between channels and message processor. The resulting DSL-based smart system configuration was functionally identical to the initial, C#-based one. Thus, the DSL-based refactoring resulted in a smart enterprise trade management system with transparent configuration and a standard object-based model (routing templates, channels, etc.). The DSL developed solved the problem of smart messaging management. Since changes are chiefly localized within the transfer configuration /map, the change management has been dramatically simplified. The DSL-based methodology instantiation assisted in conquering complexity, made the proprietary system an open, scalable, and maintainable solution. The approach can easily be customized to digitally transform a broad class of similar proprietary systems.

Air Traffic Planning System

The problem is to develop a unified access to the actual Air Traffic Planning data stored in the Russian Main Air Traffic Management Centre. Currently there are a lot of different types of automation systems for flight plans processing in Regional Air Traffic Flow Management Positions across the Russian Federation. The input data for all such systems around the world are short text telegrams containing the flight description (for example, departure and destination airports, departure time, airways and fixes, etc.). The Main Centre also has its own system (Gabeydulin, Zubkova and Goryachev 2010). Each system processes its portion (the Main Centre processes all the plans) of flight plans on its own rules, based on its own mathematical model of flight performance. Of course, these calculations differ. To support the centralized air traffic flow management all participants should use the same information. The first attempt was the web-application (Gabeydulin, Zubkova, Goryachev and Muchinsky 2012)(Figure 2). However, it is based on a legacy TAXXI Baikonur technology. The old technology involved component-based visualized assembling of the server side of the application. The ready-made VCL library components from Borland had been integrated with proprietary TAXXI components. The client side was TAXXI Communicator, i.e. an XML browser or a thin client. The TAXXI technology was severely limited by Microsoft Windows framework, which was the only possible basis for both client and server side smart applications. According to the Federal Target Program Modernization of Unified Air Traffic Management System of the Russian Federation, the Main Air Traffic Management Centre was going to create a new, smart system for remote access. The proposed web-based architecture is to be based on COTS solutions: Oracle DBMS, Java technologies, Apache HTTP server.

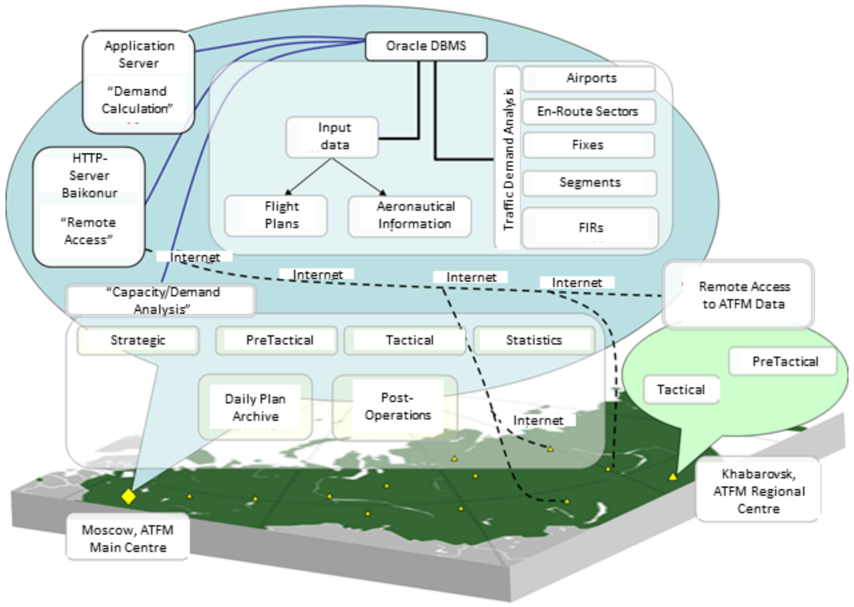

Figure 2. Russian Air Traffic Flow Management System Architecture

The practical application of this solution is the global enterprise-scale smart system, which is to provide a uniform and equal up to date information access to all international air traffic participants. The similar globalization processes are underway in Europe and the U.S.A. The suggested smart approach based on patterns and components is going to solve the issues of the architecture-level update and of migration to smart applications in Russia. The methodology is also going to simplify the digital transformation of the global air traffic management software.

Nuclear Power Plant: Approaching a 6D-Model Based Implementation

Another challenging aspect of the digital transformation is related to high level template-based software reengineering for nuclear power plants (NPP). Each stage of the NPP lifecycle is mapped into a set of business processes, where not only people, but also enterprise systems (CRM, SCM, ERP, PLM etc.) are interacting. Identifying operation sequences, the systems form business process automation standards. For example, workflow mechanisms can assist in building enterprise standards on electronic documents validation and approval. During a certain NPP lifecycle, the enterprise systems acquire information on it. Finally, each of the enterprise systems reveals certain NPP aspects: design, technology, economics etc. Thus, the systems together describe NPP as a large-scale object. Heterogeneous nature of such data objects, and a huge number of units, make NPP a very complex information object. Under digital transformation, a major competitiveness criterion in nuclear power industry is a set of electronic manuals, which helps to assemble, troubleshoot, repair NPP etc. Such manual set provides smart information models of the NPP, which allow getting information on the object without directly contacting it. Such a versatile description, combined in a single smart data model, is often referred to as a 6D model, which includes 3D-geometry, time and resources for operating the plant. Since mechanisms for information searching, scaling, filtering and linking, should provide complete and non-contradictory results, these smart models should have well-defined semantics. The uniqueness of data entry assumes information model data acquisition by the enterprise systems throughout the lifecycle.

While a single smart information model can be derived out of a single system, the 6D smart model should combine information models of a number of systems. The methodology for building a smart 6D model suggests portal based system integration, which can reside on a "platform" capable of entire lifecycle support (such as Siemens Teamcenter or DS V6). The further smart information model development assumes monitoring system state changes and their influence to the other parts of the system. This helps to instantly respond to critical issues in NPP construction, which can be used for smart decision making. Among major nuclear industry challenges, there is a concept of a typical optimized nuclear reactor. The digital transformation idea is selecting typical invariant units for rapid "template-based" development of a set of slightly varying versions. Applying this digital transformation to the 6D smart information model of the nuclear reactor is a promising approach to pattern based component development of NPP series.

Conclusion

SSDL management is a challenge in case of large-scale distributed heterogeneous smart applications. To address this challenge, a uniform SSDL smart management methodology is suggested, which includes models, methods and supporting CASE-level tools. The methodology implementations for digital transformation of a few large-scale governmental and commercial enterprises have proved essential project terms and cost reduction, and industrial quality level of the smart heterogeneous applications. Implementation of the methodology allowed to develop a unified ESS, which integrates a number of heterogeneous components: contemporary Oracle-based ERP modules for financial planning and management, a legacy HR management system, and a weak-structured multimedia archive. Other implementations and work-in-progress areas include: smart air traffic planning system, smart messaging system for a trading enterprise, smart systems for NPP and banking. Each of the implementations is a domain-specific one, so the system cloning process is not straightforward, and it requires certain analytical and CASE re-engineering efforts for digital transformation. In most cases, the approach reveals patterns for building similar smart implementations in series, which results in substantial term-and-cost reduction of 30% and more. These series can be applied for digital transformation of subsidiaries and enterprises, for implementation Industry 4.0 scenarios.

REFERENCES

Barendregt H.P. 1984. *The lambda calculus (revised edition)*. Studies in Logic, North Holland, Amsterdam.

Birnbaum, L., Forbus, K., et al. 2005. "Combining analogy, intelligent information retrieval, and knowledge integration for analysis: A preliminary report" in *Proceedings of ICIA 2005*. McLean, USA.

Curry, H.B., Feys, R. 1958. *Combinatory logic*, Vol.1, North Holland, Amsterdam.

Evans, E. 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison Wesley.

Fowler, M. 1997. *Analysis Patterns: Reusable Object Models*, Addison Wesley.

Gabeydulin, R.K., Zubkova, I.F., and Goryachev, D.I. 2010. "Algorithms and Software of the Automated Air Traffic Planning System of the Main Air Traffic Flow Management Centre", *Journal of Civil Aviation High Technologies* (159:9), pp. 121-127.

Gabeydulin, R.K., Zubkova, I.F., Goryachev, D.I., and Muchinsky, A.V. 2012. "Implementation of a remote function for airspace use analysis", *Journal of Civil Aviation High Technologies* (184:10), pp. 100-107.

Guha, R., and Lenat, D. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

Kalinichenko, L. and Stupnikov S. 2009. "Heterogeneous information model unification as a pre-requisite to resource schema mapping" in *Proceedings of ITAIS 2009*. Springer, pp. 373-380.

Lenat, D., and Reed, S. 2002. "Mapping Ontologies into Cyc" in *Proceedings of AAAI 2002 Conference Workshop on Ontologies for the Semantic Web*. Edmonton, Canada.

Roussopulos, N.D. 1976. *A semantic network model of databases*. Toronto Univ.

Schach, S.R. 2011. *Object-Oriented and Classical Software Engineering (8 ed.)*. McGraw-Hill.

Scott, D.S. 1981. *Lectures on a mathematical theory of computations*. Oxford University Computing Laboratory Technical Monograph. PRG-19.

Sommerville, I. 2016. *Software Engineering, 10th Edition*. Pearson.

Tomaiko, J.E. 1989. "Twenty-year Retrospective: The Nato Software Engineering Conferences" in *Proceedings of the 11th International Conference on Software Engineering*, Vol.I, p.96.

Wolfengagen, V.E. 1999. "Event Driven Objects" in *Proceedings of CSIT'99*. Moscow, Russia, pp. 88-96.

Wolfengagen, V.E. 2010. *Applicative Computing. Its quarks, atoms and molecules.* Moscow: JurInfoR.

Zykov, S.V., 2007. "Enterprise Content Management: Bridging the Academia and Industry Gap" in *Proceedings of i-Society 2007*. Merrillville, IN, USA, Vol.I, pp. 145-152.

Zykov, S.V. 2010. "ConceptModeller: A Frame-Based Toolkit for Modeling Complex Software applications" in *Proceedings of the International Multi-Conferences on Complexity, Informatics and Cybernetics (IMCIC 2010)*. Orlando, FL, U.S.A., Vol.I, pp. 468-473.

Zykov, S.V. 2015a. "Enterprise Applications as Anthropic-Oriented Systems: Patterns and Instances" in *Agent and Multi-Agent Systems: Technologies and Applications*. *Proceedings of the 9th International Conference KES-AMSTA 2015*. Sorrento, Italy, Springer Series on Smart Innovation, Systems and Technologies, Volume 38, pp. 275-283.

Zykov, S.V. 2015b. "Human-Related Factors in Knowledge Transfer: A Case Study" in *Agent and Multi-Agent Systems: Technologies and Applications*. *Proceedings of the 9th International Conference KES-AMSTA 2015*. Sorrento, Italy, Springer Series on Smart Innovation, Systems and Technologies, Volume 38, pp. 263-274.

Zykov, S.V. 2016. *Crisis Management for Software Development and Knowledge Transfer*, Issue 61: Springer Series in Smart Innovation, Systems and Technologies. Switzerland : Springer International Publishing.

Zykov, S.V. 2017. "Crisis Response and Management" in *Encyclopedia of Information Science and Technology*, Fourth Edition, pp. 1396-1406.

Zykov, S.V., Kukushkin, A. 2013. "Using Web Portals to Model and Manage Enterprise Projects", *International Journal of Web Portals* (4:5), pp. 1-19.