9-2017

Method For Information Systems Automated Programming

Ivan Stanev
University of Sofia, instanev@fmi.uni-sofia.bg

Maria Koleva
University of Sofia, mkoleva@fmi.uni-sofia.bg

METHOD FOR INFORMATION SYSTEMS AUTOMATED PROGRAMMING

Research full-length paper

Track N°08

Ivan Stanev, University of Sofia, Sofia, Bulgaria, instanev@fmi.uni-sofia.bg

Maria Koleva, University of Sofia, Sofia, Bulgaria, mkoleva@fmi.uni-sofia.bg

Abstract

The suggested method is based on experience gained during the development of three large state administration programs, namely Bulgarian e-Customs (BeC), Bulgarian e-Health (BeH), and Bulgarian e-Government (BeG). Both technological and organizational problems during the realization of these programs are identified. A Common Platform for Automated Programming (CPAP) is developed in order to address and solve the identified problems. A method for information systems (IS) automated programming based on KBASE (Knowledge Based Automated Software Engineering) is proposed. Widely used international standards for system specification and generation are selected, including Business Process Model and Notation (BPMN), Case Management Model and Notation (CMMN), Unified Modelling Language, Decision Management and Notation (DMN), Ontology Web Language (OWL), natural language (NL) combinatorial dictionaries, Contextual design (CD). CPAP Development environment architecture is designed including server area, data base area, security area, Enterprise service bus, and specialized NL- based, context- based, event- based, process- based, message- based, service- based, object- based, rule- based, and ontology-based tools. The advantages of several KBASE CPAP prototype realizations are presented.

Keywords: e-Customs, e-Health, e-Government, Common Platform for Automated Programming, Knowledge Based Automated Software Engineering, automated programming, BPMN, CMMN, DMN, OWL, natural language processing, knowledge processing, contextual design, context-based technique, event-based technique, process-based technique, message-based technique, service-based technique, object-based technique, rule-based technique, ontology-based technique.

1 Motivation

The process of Enterprise Information Systems (EIS) development is still less effective and more resource consuming than the software industry and their clients would require.

This has been demonstrated by two facts that persist in software development in recent years, namely: (1) the software industry experiences a permanent shortage of IT professionals; (2) there is a turbulent discussion of what methods and technologies shall be used for the various IT projects.

These facts unambiguously show that automated programming shall be considered the main solution to existing problems. As a result, the demand for software specialists will gradually reach an acceptable level. Automation will certainly reduce the development and maintenance cost of the software products, improve their quality and reduce their design and development time.

Less obvious, however, are the decisions that need to be made when selecting a method, techniques, and a platform for automated programming.

2 Background and related work

This article is part of a series of works related to the elaboration of the KBASE Method, which includes: (1) ***Analysis of the problems*** in the development of EISs, namely Bulgarian e-Customs ([20]), Bulgarian e-Health ([19]), and Bulgarian e-Government ([22]); (2) The ***idea of the KBASE*** method ([16]); (3) ***Architecture prototypes*** of EISs (BeC [21], BeH [19], BeG [23]) in the KBASE context; (4) ***A Common platform for automated programming*** (CPAP, [18]).

The KBASE method is developed as result of the domain area problems analysis, based on 85 software products developed by the KBASE team. Most of the KBASE realization techniques used to automate or improve the programming process are partially or fully implemented and verified in one or more of these developments.

Data demonstrating the benefits of the technologies selected for the KBASE method are obtained from project records. EISs developments are performed in the following roles: **(1)** ***researcher*** (21 research prototypes); **(2)** ***contracting authority*** (53 projects implemented mainly under the BeC, BeH and BeG programs); **(3)** ***contractor*** (established a high-tech software company which designed and developed 11 EISs in partnership with leading companies in the software development industry).

In this section are presented some problems and successful solutions in building EISs (sub-section 2.1) and knowledge processing (KP) systems (sub-section 2.2). The results of the cloud computing platforms comparative analysis are presented in sub-section 2.3. The choice of the Rational Unified Process (RUP, [6]) as a base of the KBASE method is justified in sub-section 2.4.

2.1 Related work in the Enterprise IS area

This section presents the results achieved during the realization of three large state administration programs – Bulgarian e-Customs, Bulgarian e-Health, and Bulgarian e-Government. They are detailed in [19], [21], and [23].

Each of the three programs at present has more than 15 business modules, 5000 workstations in intranets, and between 500 000 and 3 000 000 end users in internet. They are an integral part of the trans-European solutions developed and operated by the EU Commission, EU member states and partner countries.

The technological development of BeC, BeH and BeG spans over a long period of time that could be partitioned generally in three stages.

Stage 1 by the end of 2006 - first version of the information systems is developed following the (**1**) RUP, UML, JEE, XML standards; (**2**) object oriented, multi-tier, web-based, distributed software architecture; (**3**) UML, JEE XML, EJB; Application Server, DB Server technologies. The first modules using SOA are developed and the centralization of computing resources is started. No active services are available to business and citizens, the interoperability concept is not working smoothly in practice, and a common software architecture is still to be elaborated.

Stage 2 by the end of 2013 - include the reengineering of these information systems following SOA and CC concepts. The single window technology is introduced together with centralized identification and authorization management. The Enterprise service bus is established. A centralized service repository is put in place but the implementation of reusable components remains relatively low and changes to the system require a considerable amount of time and effort. The interoperability concept based on state registers is realized. The underlying infrastructure is virtualized but remains insufficient to cover the needs of the software developments. Selected software development process is difficult to be implemented in practice.

Stage 3 by the end of 2020 - include the creation of Strategies, and Roadmaps of BeC, BeH and BeG. The suggested architectures are similar to those proposed in [19], [21], and [23]. The realizations of the proposed architectures is started.

During the realization of these programs two types of ***problems*** are recognized: (**1**) ***organizational***, such as: lack of single access point with a centralized authorization solution, inefficient infrastructure use, low level of standardization of processes and objects, insufficient IT staff to develop new systems and support existing ones, low competence of business staff for business processes description and functional testing, lack of guidelines for collection and interpretation of information, huge resources are allocated to the development of inefficient business models, low competence for requirements gathering and elicitation, chaos in requirements definition, low level of business specific activities quality; and (**2**) ***technical***, such as: big volume of not digitized data, low quality of digitized data, low level of semantic interoperability, implementation of reusable components to a minimum, resulting in frequent rewrite of existing functionality, lack of scalability, low level of software development automation, lack of flexibility and instruments for fast information systems adaptation to rapidly changing legal base, lack of centralized identification, relatively low quality of the developed software products, lack of standardization of processes and objects, low test coverage.

2.2 Related work in the Knowledge Processing area

This section presents the results achieved in the Knowledge Processing area. A detailed description is presented in [16].

Some of the useful results are: (**1**) prototyped ***3 specification techniques*** (formal language, graphical user interface, natural language processing); (**2**) prototyped ***5 AI techniques*** (non-formal specification, code generation, self-verification, self-monitoring, self-tuning); (**3**) prototyped ***4 KBASE components*** (knowledge processor, product generator, problem solver, knowledge base manager); (**4**) prototyped ***5 technological processes*** (SOA interpretation process, knowledge engineering process, domain customization process, system customization process, runtime operation process).

2.3 Cloud Computing Platforms comparative analysis results

The ***industrial platforms*** Amazon AWS, Microsoft Azure, Google App Engine; VMWare vCloud, IBM Bluemix, HP Helion, and Oracle OCPaaS are analyzed in [17].

As a result of this analysis a ***set of requirements*** ([17]) for the KBASE platform CPAP is prepared including: (**1**) CPAP should combine techniques for automated programming from SOA, CC, the KBASE method and the Method for Automated Programming of Robots ([15]); (**2**) CPAP should provide for incomplete and incorrect specification of the problem to be solved using language and tools familiar to the end user or “man in the street”; (**3**) The platform should enable knowledge acquisition and knowledge interpretation (e.g. knowledge based systems, ontologies and fuzzy sets) for automated removal of deficiencies and inaccuracies in the specification and for software generation; (**4**) The platform should ensure automated software generation from complete and accurate specification; (**5**) CPAP should allow the use of automated techniques such as Contract Testing and Quality of Service Testing to fine tune the software, to check on software performance and integration with third party components; (**6**) Emphasis should be given to the automated generation of user identification components.

The Software Architecture of the Common Platform for Automated Programming is designed in [18] to cover the above mentioned requirements.

2.4 Arguments for selection of RUP as base of KBASE

Several methods for management of the software engineering (SE) process are studied. The selected methods can be organized in three groups: (**1**) methods covering only the management aspects of the SE process, including "shell" methods such as: **PRINCE2**, or Projects IN Controlled Environments ([2]), developed by the UK Cabinet Office; **TEMPO**, or the TAXUD Electronic Management of Projects Online, developed by the European Commission Directorate General Taxation and Customs Union; **PMBOK**, or Project Management Body of Knowledge ([14]), developed by the US Project Management Institute; (**2**) methods covering partially the management and technological aspects of rapid SE processes (the **Agile** family of methods, [1]); (**3**) methods covering to a great extent the management and technological aspects of SE processes such as the Rational Unified Process (**RUP**, [6]), developed by Philippe Kruchten, Ivar Jacobson and others at Rational Corporation.

The ***Shell group*** of methods does not describe in details the roles, objects and actions in the SE process. This is often the reason for serious misunderstanding between clients and developers, as well as between different developers. The low level of standardization makes these methods inappropriate for automated programming.

The ***Agile group*** of methods has two important problems: (**1**) the short and often incomplete design reduces to minimum the possibilities for reuse; (**2**) as a results of the rapid and imprecise development process, the lifetime of the realized products is too short. Thus the Agile group of methods is not appropriate for the realization of EISs.

Only the ***RUP group*** of methods is suitable for EIS development. However RUP hinder a smooth development process due to the following problems: (**1**) roles, objects, and actions are defined in detail, thus making the software process too heavy and difficult to customize; (**2**) the development team training time is too long and expensive; (**3**) the complexity of the process makes it difficult to be followed by the client. These problems could be solved after simplification of RUP and extension with automated programming and knowledge processing methods and techniques.

2.5 Requirements for KBASE realization

As a result of the existing platforms and methods analysis the following important ***problems*** could be identified: (1) IS domain area ***low level of standardization*** (both IS components and development processes); (2) ***Low quality of IS design*** caused by the gap between clients and developers professional language and culture; (3) IS realized with ***inefficient development methods***; (4) ***Low level of reusability*** of IS components due to partial design and early implementation; (5) IS deployment after ***low level of Contract and Quality of Service testing*** due to selected inappropriate test cases, and very limited test coverage.

The standards, technologies and tools used for the development of BeC, BeH, and BeG demonstrate limited capabilities to solve the relevant problems. For this purpose new development method and technological solution need to be sought with regard to their evolution in order to guarantee that these systems meet the following ***requirements***: (1) have a high degree of standardization; (2) allow the use of cloud technologies for the integration of new hardware and software resources into a common structure for collaborative use; (3) allow intelligent data and knowledge processing; (4) ensure interoperability with partner information systems; (5) provide high quality automation of software programming.

3 KBASE Method definition

The KBASE method is defined partially in [16] (concept, key terms, platform, and 5 case studies). The KBASE term definitions are extended in this article with the description of the KBASE framework (section 3.1), disciplines (section 3.2), specification language (section 3.3) and development environment (section 3.4).

3.1 KBASE Framework

The KBASE ***Framework*** (presented in *Figure 1*) is an integrator of the software development ***methods*** (**M**) RUP and KBASE. This Framework supports the ***technologies*** (**T**) Software Engineering (**SE**), Service Oriented Architectures (**SOA**), Automated Programming (**AP**), and Knowledge Processing (**KP**). These methods and techniques are realized in the context of the ***platforms*** (**P**) Cloud Computing (**CC**) and Common Platform for Automated Programming (**CPAP**).

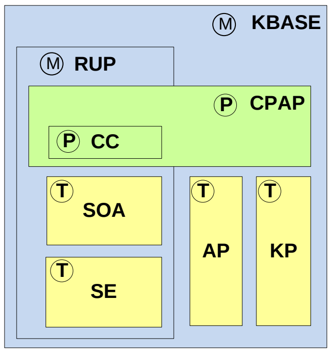

Figure 1. KBASE Framework

In this Framework the techniques SE and SOA are realized for development of EISs in the context of RUP. The CC platforms typically integrate the components and applications of these techniques.

The techniques SE and SOA could be enriched and/or upgraded in all possible combinations by AP and KP. The platform CPAP enriches and/or upgrades CC for the realization of all programming techniques. The method KBASE enriches and/or upgrades RUP for the realization of all programming techniques and their interpretation in the presented platforms.

This Framework is designed to: (1) ***generate*** EIS from formal and non-formal design specifications; (2) ***complete and determine*** the fuzzy and incomplete non-formal specifications (when required) using KBASE knowledge base, and ontologies; (3) ***verify*** the complete and determined specifications through comparison with domain area models defined beforehand; (4) ***generate*** EIS by pre-designed, pre-developed and pre-tested reusable components and services; (5) ***integrate*** the pre-defined components and services with new ones; (6) ***standardize*** the EIS specification, EIS components, EIS generation process and EIS integration process.

The KBASE method realizes the automated programming of EIS in three large groups of actions:

1. **Specification of developed IS**

This process includes the actions outlined in *Table 1* following the RUP and KBASE methods. These actions represent a simplification of RUP to KBASE.

2. **IS specification validation**

In this group are validated new IS components (such as dictionary articles, ontology objects and relations, semantic and syntactic constructs, and objects of the specific IS). The objects which are not validated successfully are returned to knowledge engineers for more detailed and more precise specification.

3. **IS generation**

This group covers the generation of the software components required for the IS operation. Most actions are performed automatically by the CPAP tools. In case of complex tasks or need for further information about the domain area, the knowledge engineers intervene and support the process with their expertise.

3.2 KBASE Lifecycle Disciplines

The automated programming process modifying RUP is presented in *Table 1* below. The disciplines names are presented in the "*Discipline*" column. The work performed within each discipline is described in the "*Discipline action*" column. The specification technique/s used for the relevant action is indicated in the last column "*Specification technique*" and further described in section 3.3. The resulting model is presented in the "*Model*" column.

Discipline	Model	№	Discipline action	Specification technique
management	SDP	1.	project management and realization	SPEM
management	SDP	2.	configuration and change management and realization	SPEM
management	PQP	3.	quality management and realization	SPEM
management	TM	4.	test management and realization	BIT
management	IM	5.	infrastructure management and realization	event based
requirements	SRS	6.	gather and analyze relevant legal base	context based, NL based, ontology based
requirements	SRS	7.	build organization hierarchy	object based
requirements	SRS	8.	define functional areas	context based
requirements	SRS	9.	build roles hierarchy for each functional area	object based
requirements	SRS	10.	develop software architecture concept	event based
requirements	SRS	11.	gather functional requirements	context based, NL based, ontology based
requirements	SRS	12.	gather non-functional requirements	context based, NL based, ontology based
requirements	SRS	13.	categorize, classify and prioritize functional requirements	context based
requirements	SRS	14.	categorize, classify and prioritize non-functional requirements	context based
analysis	BM	15.	selection of control structure (business process / state engine / multi-agent system)	process based / object based / event based
analysis	BM	16.	control structure description	process based / object based / event based
analysis	BM	17.	high level description of control structure services (service type, input, output and system objects, interfaces, interactions, messages, reports, security, actors, preconditions, post conditions)	service based, rule based
analysis	DatM	18.	identify primary data objects - documents, forms, reports, messages	object based, rule based
analysis	DatM	19.	develop data objects hierarchy	object based
analysis	BM	20.	select candidate services	event based
analysis	BM	21.	select services	event based
analysis	BM	22.	refine control structure with candidate services to control structure with services	event based
analysis	UCM	23.	define use cases for each service	service based
analysis	UCM	24.	define input and output data objects for each service and select data objects storage options	object based, rule based
analysis	UCM	25.	define input and output messages for each service and select messages storage options	object based
design	SA	26.	develop software architecture component diagrams	event based
design	SA	27.	define system architecture components	event based
design	SA	28.	define interfaces of system architecture components	interface based
design	SA	29.	develop GUI navigation tree	object based
design	DesM	30.	prepare class diagrams of the services	service based

Discipline	Model	№	Discipline action	Specification technique
design	DesM	31.	define attributes and operations of classes	service based
design	DesM	32.	develop sequence diagrams of the services	interface based
design	DatM	33.	refine object data model into relational data model	object based
design	DatM	34.	relational data model normalization (if neces-sary)	object based
design	DesM	35.	develop statechart diagrams of the services	interface based
design	DesM	36.	develop statechart diagrams of important data objects	interface based
design	DatM	37.	update Data model	object based, rule based
design	CM	38.	prepare package diagrams	event based
design	OWLM	39.	associate use case, component, class, package and infrastructure diagrams	ontology based
generation	CM	40.	code generation	all
generation	CM	41.	code implementation	all
generation	CM	42.	code integration	all
exploitation	all	43.	deployment	all
exploitation	all	44.	exploitation	all
exploitation	all	45.	enhancement	all
exploitation	all	46.	recycling	all

Table 1. KBASE lifecycle disciplines

Abbreviations used in Table 1: **BIT** - Built-in-Test, **BM** – Business model, **CM** – Code model, **DatM** – Data model, **DesM** – Design model, **IM** – Infrastructure model, OWLM - Web Ontology Language model, **PQP** – Project Quality Plan; **SDP** – Software Development Plan, **SA** – Software Architecture, **SPEM** - Software & Systems Process Engineering Metamodel Specification ([9]); **SRS** – Software Requirements Specification, **TM** – Test model, **UCM** – Use case model.

It has to be noted that: (1) the ***pre-project*** is performed by implementing the disciplines and actions to a limited extent; (2) actions could ***iterate*** in order to achieve better results; (3) the "***Management***" discipline represents horizontal activities related to planning, specification and realization; (4) the "***Code model***" term is used instead of the standard RUP "Implementation model" with no changes in the contents and notation; (5) the "***infrastructure diagram***" term is used instead of the UML standard "deployment diagram" to better convey the purpose with no changes in the notation; (6) the "***Infrastructure model***" term is used instead of the standard RUP "Deployment model" to incorporate the development, testing, staging, production and management environments; (7) "***Service***" is used for services of three different control structures, namely business process, state engine and multi-agent systems.

For the purpose of the KBASE method presented in this paper, the following simplifications/ extensions to RUP are proposed: **(1)** the RUP discipline Business Modeling is removed since low productive, and highly resource consuming; **(2)** the RUP discipline Analysis and Design is separated in two KBASE disciplines – first Analysis, and second Design, in order to reduce the gap between domain area experts and analysts on the one hand and between analysts and designers on the other hand; **(3)** the RUP discipline Implementation is replaced by Generation; **(4)** the RUP horizontal disciplines Configuration and Change management, Project management, Testing and Environment (№1-5) are combined in one Management discipline to optimize project realization; **(5)** the KBASE Requirements discipline (actions №7-12) uses the Contextual design models which improve the quality of primary information collection; **(6)** the KBASE Analysis discipline (actions № 15 – 17) uses the process-based, object-based and event-based specification techniques instead of the RUP Use Case Specification in order to reduce the resources for UML model completion; **(7)** the KBASE Design discipline (action №39) uses ontologies for automation of the development process; **(8)** the KBASE Generation discipline (actions №40-42) uses automated support of models and documents; **(9)** the Test management and realization action (№4) uses structured Use Case Specifications for better compliance between use cases and test cases, improvement of the quality and efficiency of the test process; **(10)** the Test management and realization action (№4) uses the Built-in-Test method ([4]) for automated testing; **(11)** the actions from the RUP Deployment discipline are incorporated in the KBASE Exploitation discipline (action №43).

3.3 KBASE Specification Language

The following ***Specification Techniques*** (ST) are selected to implement the actions of the method described in *Table 1*, as well as to create ontology objects static and dynamic behavior.

(1) ***Natural Language based ST*** (standard used - Natural Language Combinatorial Dictionary, NLCD). The specification technique is used to describe domain area ontology objects and behavior, the legal base, functional and non-functional requirements. The Combinatorial Dictionary contains information for the syntax functions, semantic-syntax functions and word-order zone of each lexeme. A Linguistic Processor verify the NL description of domain area objects and behavior. If the description is successfully verified they are integrated in the domain area ontology using the morphological NL processor, semantic-syntax NL processor, and business rules.

(2) ***Context based ST*** (standard used – Contextual Design, CD, [5]). CD models are used to improve the structuring of information about the domain area. The specification technique is used to describe domain area ontology objects and behavior, describe functional area, functional and non-functional requirements. A Context Interpreter verifies the specified objects and behavior. If the specifications are successfully verified they are integrated by a Context Manager in the domain area ontology, or used for generation of new program code.

(3) ***Event based ST*** (standard used – Case Management Model and Notation, CMMN, [12]; Business Process Model and Notation, BPMN, [10]; Unified Modeling Language, UML, [11] (Component Diagram, Infrastructure Diagram, and Package Diagram)). The specification technique is used to describe complex Information Systems (IS), realized as event based Multi-Agent Systems, including their Software Architecture, Infrastructure Model, and behavior. An IS Interpreter verifies the specified models and behavior. If the specifications are successfully verified, they are used for generation of IS program code.

(4) ***Process based ST*** (standard used – BPMN). The specification technique is used to describe domain area candidate business processes. The specified candidate processes are used for identification of candidate services and selection of services. After the selection of services, candidate processes are transformed in processes. A Process Interpreter verifies the specified processes. If the specifications are successfully verified they are integrated by a Process Manager in the domain area ontology, or used for generation of new program code.

(5) ***Message based ST*** (standard used – BPMN, UML (Sequence Diagram)). The specification technique is used to describe the Server to server (S2S) direct communication, including structure of messages, typical message objects, data types, data structures, and sequence diagrams describing the S2S valid and invalid interactions. A Message Interpreter verifies the specified messages, message objects and interactions. If the specifications are successfully verified they are integrated by a Message Manager in the domain area ontology, or used for generation of new program code.

(6) ***Service based ST*** (standard used – UML (all diagrams)). The specification technique is used to describe domain area services including their input and output objects, classes, attributes, methods, behavior, etc. A Service Interpreter verifies the specified objects and behavior. If the specifications are successfully verified they are integrated by a Service Manager in the domain area Service Repository, or used for generation of new service program code.

(7) ***Object based ST*** (standard used – UML (State Engine)). The specification technique is used to describe domain area ontology objects and their behavior, including object states, state transitions, transition conditions, and transition operations. An Object Interpreter verifies the specified objects and behavior. If the specifications are successfully verified they are integrated by an Object Manager in the domain area ontology, or used for generation of new program code.

(8) ***Rule based ST*** (standard used – Decision Model and Notation, DMN, [13]). The specification technique is used to describe domain area simple and complex rules applicable to class attributes and diagrams transitions/ connectors/ associations, ontology establishment and restructuring, decision making, and knowledge management. A Rule Interpreter verifies the specified rules and behavior. If the specifications are successfully verified they are integrated by a Rules Manager in the domain area ontology, or used for generation of new program code.

(9) ***Ontology based ST*** (standard used – Web Ontology Language, OWL, [24]). The specification technique is used to describe formally taxonomies and classification networks, essentially defining the structure of knowledge for various domains. An Ontology Interpreter verifies the specified ontologies. If the specifications are successfully verified they are integrated by an Ontology Manager in the domain area ontology, or used for generation of new program code.

3.4 KBASE Development Environment

The CPAP model is developed taking into account the features of leading CC technological environments, standards defining SOA and CC, as well as the KBASE method. CPAP components for classic, combined, automated and ontology programming, organized in layers and packages, are presented in [18].

CPAP-DE architecture is presented in *Figure 2* below. ***Actors*** (presented as) represent: (1) the end user, responsible to specify the problem to be solved, to provide the input data, and to analyze the calculated results (2) external CPAP correspondents interacting through server-to-server (S2S) communication; and (3) knowledge engineers (KE) in the business and IT domain areas responsible to design and implement new data objects, ontology primitives, ontology structures, Graphical User Interfaces (GUI), messages, interactions, rules, etc.

CPAP-DE Components are: (1) **GUI Display** (presented as) conducts the dialogue between CPAP and CPAP Actors. The key requirements for GUID realization are the automation of its design, development, and testing process, as well as its rapid reengineering; (2) **Editor** (presented as) is a software component which ensures the design or modification of GUIs, the specification and modification of components. The simple specification techniques and availability of rich libraries with predefined reusable objects are the key requirements for the realization of the Editors; (3) **Compiler** (presented as) is a software component which ensures the transformation of non-formal to formal user specification. The key requirement for the Compiler functionality is the capability of identifying the insufficient, incomplete, or fuzzy parts of user specifications and for producing a fully defined formal specification; (4) **Manager** (presented as) is a software component responsible for the organization, management and fault tolerant performance of every particular CPAP-DE process. Its role is to ensure the resources required for the performance of the process, to organize the communication with other managers, to control the prioritization of the performance of required services, as well as to identify and, if possible, to correct the own runtime errors. Well-defined self-training and self-organization capabilities are the main requirements for these managers; (5) **Generator** (presented as) is a software component responsible for the generation or modification of other software components through formal specifications. Considerable reduction of development and testing activities performed by CPAP-DE experts after component generation is the main goal of its functionality; (6) **DB** (presented as) is a software component responsible for storing data, knowledge, services and components. (7) **Servers** (presented as) manage the execution of CPAP components at service and process level.

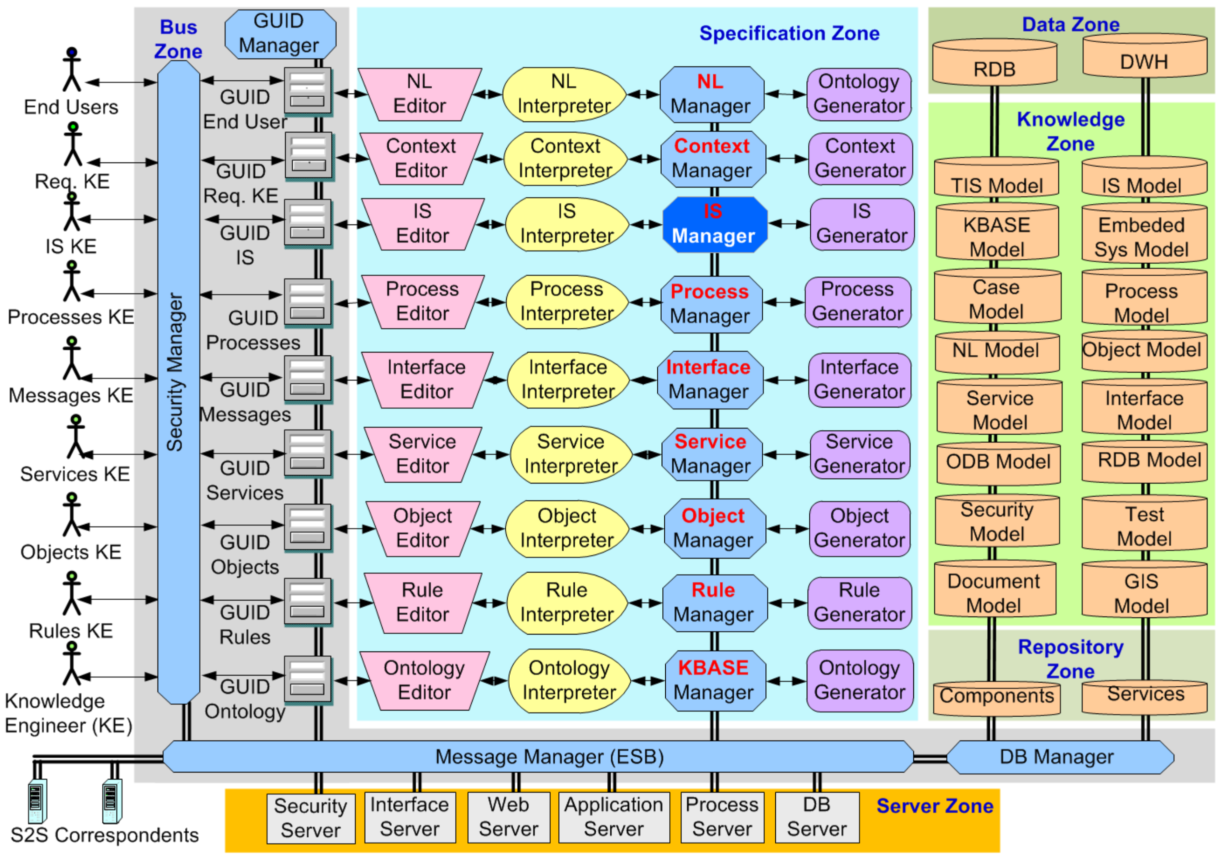

Figure 2. DE Architecture

Abbreviations used in Figure 2: **DB** – Data base, **DWH** – Data warehouse, **ESB** – Enterprise service bus, **GIS** – Geographical information system, **GUID** – Graphical user interface display, **KE** – Knowledge engineer, **IS** – Information system, **NL** – Natural language, **RDB** – Relational data base, **Req.** – requirements, **S2S** – Server-to-server, **TIS** – Template information system.

CPAP-DE Zones are the following: (1) The ***Specification Zone*** (including Natural Language based, Context based, Event based, Process based, Message based, Service based, Object based, Rule based, and Ontology based specification instruments) is responsible for compiling to internal representation the high level IS specifications, their verification, and the generation of IS code;

(2) The ***Bus Zone*** organizes all CPAP-DE interactions via the Security Manager, Message Manager, DB Manager, the couples GUI Manager – GUIDs, and S2S communication channel – namely interactions between users (End Users and Knowledge Engineers), between the Specification Zone instruments and users, between the Specification zone and all DB Zones (Data, Knowledge and Repository), between all zones and CPAP-DE Server Zone, automatic communication between CPAP-DE Servers and partner Servers; (3) The ***Server Zone*** (including Security, Interface, Web, Application, Process, and DB Servers) manages and realizes the whole CPAP computational process; (4) The ***Data Zone*** is the CPAP operational (RDB), and analytical (DWH) data store; (5) The ***Knowledge Zone*** is the knowledge store, containing all CPAP ontologies and knowledge models; (6)The ***Repository Zone*** stores all CPAP active units, including components, services, etc.

4 Results evaluation and future work

Different combinations of the solutions incorporated in KBASE and CPAP-DE are evaluated in a large number of applications and prototypes. Some of the important results demonstrating the problem solving and efficiency improvement capabilities of KBASE and CPAP-DE are presented in *Table 2* below.

These improvements are either quantitative or qualitative. Relevant improvements for the specific application are marked with √ at their intersection as applicable.

A detailed description of the Module for Automated Programming of Robots (MAPR), Intelligent Product Manual (IPM), Built-in-Test Adaptive Document Display (BIT), and Information Objects Manager (IO.Man) is provided in [16]. BeC, BeH and BeG are presented in section 2.1 of this paper.

Type	Improvement	MAPR	IPM	BIT	IO.Man	BeC	BeH	BeG
quantitative	product specification time reduced	√				√		
quantitative	programming time reduced	√	√	√	√	√		
quantitative	COTS (Commercial off-the-shelf) components integration time reduced			√		√	√	√
quantitative	testing time reduced	√	√	√	√	√		
quantitative	COTS components testing time reduced			√		√		
quantitative	IT team reduced	√			√	√		
qualitative	adaptive to various domain areas	√	√	√	√			
qualitative	adaptive to different end users	√	√					
qualitative	adaptive presentation to various standards		√			√	√	√
qualitative	adaptive presentation to different media	√	√			√	√	√
qualitative	real time code synchronization		√					√
qualitative	real time documents synchronization		√			√	√	√
qualitative	prevent emergency system failure			√	√	√	√	√
qualitative	real time performance improvement			√	√			√

Table 2. Improvements achieved in the realized KBASE applications

Quantity improvements are related to considerable decrease of the necessary time and effort for software development (modelling, implementing, testing, customization) and modification as well as for the development of the integrated platform.

Quality improvements are in three aspects: (1) improvement of product adaptation to the domain area (adaptation to different users, standards, media, etc.); (2) improvement of product stability including prevention of system failure and failure solving; (3) quality of service improvement including product response time reduction, increased real-time product and product documentation update and synchronization, etc.

Another effect of introducing the KBASE technology and tools is the change of IT development team profile. Although the application of the domain area ontology requires more business analysts in the team, the introduction of automation in the implementation process considerably reduces the required number of implementers and testers.

Measurements on the MAPR, IPM, BIT and IO.Man projects are provided in [16]. Measurements in the enterprise information systems domain is presented in Table 3 below. The reference projects cover the three domains presented in section 2.1, namely e-customs, e-health, and e-government.

The problem number in the header row corresponds to the problems described in sub-section 2.5. The first column indicates the specific realization, name, owner, and results for each reference project (noted as "Rn"). For each problem are presented two reference projects, R1 being the one realized following the RUP software development process, and R2 – following the method proposed in this paper. Some of these projects have the same scope while using different realization technologies (e.g. reference projects for problems 3 and 5). In this case R1 is developed by a contractor and R2 is developed by a team of students from a software engineering master program.

The KBASE solution to each problem, successfully demonstrated in R2 is presented in the bottom row.

Problem №	**1**	**2**	**3**	**4**	**5**
R1 realisation	**manual** GUI programming	**non-structured** interviews	**UML Use Case** described processes	implementation **after 20 % design**	**non-structural** use cases
R2 realisation	**automated** GUI programming	**structured** interviews (data flow diagrams and mock-ups)	**BPMN** described processes	implementation **after 100% design**	**structural** use cases
R1 name	EMS1	BTMS1	TIS Hospital Care	eVote	BTMS1
R2 name	EMS2.1	BTMS2	TIS Hospital Care	eID	BTMS1
R1 owner	BCA	BCA	MoH	MTITC	BCA
R2 owner	BCA	BCA	UoR	MTITC	UoR
R1 results	role BA - **2** role GUI Programmer - **12** Design time (p/m) - **4** Impl. Time (p/m) - **24**	functional test errors - **n** change request (DesM) - **n** change request (Prod) - **n**	classes DesM (same granularity) - **630**	reusable components - **5%** functional change requests - **n**	number of Use Cases - **250** number of Test Cases - **2000** test coverage - **25%**
R2 results	role BA - **4** role GUI Programmer - **3** Design time (p/m) - **12** Impl. Time (p/m) - **6**	functional test errors - **0.7** x **n** change request (DesM) - **0.5** x **n** change request (Prod) - **0.2** x **n**	classes DesM (same granularity) - **245**	reusable components - **32%** funct. change requests - **0.5** x **n**	number of Use Cases - **1350** unmber of Test Cases - **1590** test coverage - **87%**
KBASE solution	new full set specification techniques	introduced context based ST	introduced event, process, and object based ST	replace RUP by KBASE design method	introduce structural Use Case algorithmization style

Table 3. Results evaluation

Abbreviations used in Table 3: **BA** – Business analyst, **BCA** – Bulgarian Customs Agency, **BTMS** – Bulgarian Transit Management System, **DesM** – Design model, **EMS** – Excise Management System, **MoH** – Ministry of Health, **MTITC** – Ministry of Transport, Information Technologies and Communications, **TIS** – Template information system, **UoR** – University of Ruse Software Engineering Master's program.

Future work is related to: (1) development of a new Unified Specification Language to integrate and optimize the selected specification techniques in the KBASE method; (2) build a second generation of nine industrial specification tools; (3) build a second generation of industrial Common Platform for Automated Programming; (4) develop new specification techniques (e.g. image processing, cluster analysis etc.).

5 Conclusion

The analysis performed in this paper identifies 5 important problems related to the software development process. They are presented in section 2.5. In order to solve these problems the suggested KBASE framework integrate the methods RUP and KBASE, the technologies Software Engineering, Service Oriented Architectures, Automated Programming, and Knowledge Processing, and the platforms Cloud Computing and Common Platform for Automated Programming.

Based on the KBASE framework the following solutions are proposed: (**1**) ***Nine Specification Techniques*** based on different combinations of nine international standards for IS description are introduced to improve the level of standardization of ongoing IT processes and objects; (**2**) ***Contextual Design Method*** is introduced to improve the systematic collection of the initial information; (**3**) ***Process-based,***

object-based and event-based specification techniques are introduced as more efficient replacement of large part of the RUP Use Case Specification; (**4**) ***Automated programming*** is introduced to allow code generation after completion of the design model, which considerably increases components reusability level and the quality of the developed product; (**5**) ***Built-in-Test method*** is introduced to improve test coverage and the efficiency of the development process.

Acknowledgment

The presented work has been partially funded by the National Scientific Research Fund, Contract No. 02/13/12.12.2014.

References

[1] Abrahamson, P., O.Salo, J.Ronkainen, J.Warsta (2002). Agile software development methods: Review and analysis (Technical report). VTT. 478.
[2] Axelos Limited (2017). Managing Successful Projects with PRINCE2. The Stationery Office
[3] Bauer, F.L. (1968), Report on a conference sponsored by the NATO Science Committee. Garmisch. Germany. 7th to 11th October 1968
[4] EU IST-1999-20162 Development and Applications of New Built-in-Test Software Components in European Industries. Software Architecture. 2003
[5] Holtzblatt, K. and H.Beyer (2016). *Contextual Design, Second Edition: Design for Life*. Morgan Kaufmann Publishers. San Francisco.
[6] Kruchten, P. (2000). The Rational Unified Process: an Introduction. Addison-Wesley.
[7] Liu, F. et. all. (2011). *NIST Cloud Computing Reference Architecture*. Gaithersburg: National Institute of Standards and Technology Special Publication 500-292. US Department of Commerce
[8] Mell, P. and T.Grance (2011). *"The NIST Definition of Cloud Computing".* Gaithersburg: National Institute of Standards and Technology NIST Special Publication 800-145 US Department of Commerce.
[9] Object Management Group (OMG) (2008). *Software & Systems Process Engineering Metamodel Specification* (SPEM) v.2.0. http://www.omg.org/spec/SPEM/2.0/
[10] OMG (2014), *Business Process Model And Notation v.2.0.2*. http://www.omg.org/spec/BPMN
[11] OMG (2015). *Unified Modeling Language v2.5*. http://www.omg.org/spec/UML/
[12] OMG (2016-1). *Case Management Model And Notation v1.1*. http://www.omg.org/spec/CMMN
[13] OMG (2016-2). *Decision Model And Notation v1.1*. http://www.omg.org/spec/DMN
[14] Project Management Institute (2013). A Guide to the Project Management Body of Knowledge Fifth Edition.
[15] Stanev I. (2012). *"Method for Automated Programming of Robots",* Knowledge Based Automated Software Engineering. Cambridge Scholars Publishing. Cambridge.
[16] Stanev, I. and K.Grigorova (2012). *"KBASE Unified Process",* Knowledge Based Automated Software Engineering. Cambridge Scholars Publishing. Cambridge.
[17] Stanev, I. and M. Koleva (2015-1), "KBASE Technological framework – Requirements". In: *Proceedings of 17th International Conference on Semantic Interoperability and Integration* (ICSII 2015).

[18] Stanev, I. and M.Koleva (2015-2), "A Common Automated Programming Platform for Knowledge Based Software Engineering". In: *Proceedings of 17th International Conference on Semantic Interoperability and Integration* (ICSII 2015).

[19] Stanev, I. and M.Koleva (2016-1). "Bulgarian Health Information System based on the Common Platform for Automated Programming". In: *Proceedings of 10th Mediterranean Conference on Information Systems* (MCIS 2016).

[20] Stanev, I. and M.Koleva (2016-2). "Bulgarian e-Customs based on the Common Platform for Automated Programming – Requirements". In*: Proceedings of 55th Annual Science Conference of University of Ruse.*

[21] Stanev, I. and M.Koleva (2016-3). "Bulgarian e-Customs based on the Common Platform for Automated Programming – Technological Framework". In: *Proceedings of Proc. 55th Annual Science Conference of University of Ruse.*

[22] Stanev, I. and M.Koleva (2016-4). "Bulgarian e-Government Information System Based on the Common Platform for Automated Programming – Requirements". In: *Proceedings of 10th Jubilee International conference Information Systems & Grid Technologies* (ISGT 2016).

[23] Stanev, I. and M.Koleva (2016-5). "Bulgarian e-Government Information System Based on the Common Platform for Automated Programming – Technical Solution". In: *Proceedings of 10th Jubilee International conference Information Systems & Grid Technologies* (ISGT 2016).

[24] W3C (2012), *Web Ontology Language v.2,* https://www.w3.org/standards/techs/owl#w3c_all