CROSS-PLATFORM ENVIRONMENT FOR APPLICATION LIFE CYCLE MANAGEMENT

Elena Chebanyuk, Oleksii Hlukhov

Abstract: *"Application Lifecycle Management (ALM) integrates and governs the planning, definition, design, development, testing, deployment, and management phases throughout the application lifecycle" [OMG, 2006].*

This paper is devoted to designing of ALM for supporting all software development processes. A review of papers, making strong contribution for improving software development life cycle processes is represented. This review touches three branches of investigation, namely papers, related to: (1) improving of communication processes between stakeholders; (2) increasing effectiveness of some operations in software development life cycle processes; (3) developing fundamental methods and tools for performing different operations related to several software development life cycle. Then comparative analysis of such ALM environments as Visual Studio, Team Foundation Server, FusionForge, TeamForge, IBM Rational Team Concert, IBM Rational Software Architect, and IBM Rational Functional Tester, is performed. Comparison of different ALM environments' functionality lets to formulate requirements for designing cross-platform ALM environment.

Then the conceptual schema of cross-platform ALM based on Eclipse environment is proposed. All plugins' functionalities were properly tested. Collaboration of plugins for supporting several software development tasks is accurately defined. Data flows for different plug-ins collaboration are shown. These data flows are considered for several kinds of stakeholders roles.

In conclusion, recommendations for raising effectiveness of software development life cycle processes, using proposed cross-platform ALM environment, are presented.

Keywords: *software development life cycle, application life cycle management, software development life cycle process, requirement analysis, software designing, software testing, deployment, Eclipse, Team Foundation Server, FusionForge, TeamForge, IBM Rational Team Concert, IBM Rational Software Architect, IBM Rational Functional Tester, Mylyn, Javadoc, JUnit, STAN, FindBugs, Jubula, UML to Java Generator, Eclipce IDE, Data tools platform, windows builder, Eclipce color theme, RMF, UML Designer.*

ITHEA Classification Keywords: *D.2.1 Requirements/Specifications (D.3.1) - Elicitation methods (e.g., rapid prototyping, interviews, JAD, Methodologies (e.g., object-oriented, structured), Tools; D.2.5 Testing and Debugging - Testing tools (e.g., data generators, coverage testing); D.2.6 Programming Environments - Integrated environments; D.2.9 Management - Life cycle; D.2.11 Software Architectures; D.2.13 Reusable Software*

Introduction

According to OMG definition: "An Application Lifecycle is the continuum of activities required to support an enterprise application from its initial inception through its deployment and system optimization"[OMG, 2006].

"Application Lifecycle Management (ALM) integrates and governs the planning, definition, design, development, testing, deployment, and management phases throughout the application lifecycle" [OMG, 2006].

In software development process today incremental-iterative software development approaches is implemented.

New challenges for improvement effectiveness of software development life cycle processes causes to modifying existing techniques and tools for increasing of their effectiveness. In order to reach this goal application performance management (APM) tools are involved in software development life cycle process.

Application performance management (APM) tools offer these capabilities, enabling companies to diagnose problems quickly and improve service quality. For companies that are using Agile and DevOps processes, APM can help improve communication and expedite software delivery. It enables continuous monitoring and testing during all phases of software delivery, including production [IBM, 2015].

Certifying organizations like the International Standards Organization (ISO) have effectively worked on various models and suggested guidelines, procedures that may be adopted by IT vendors. Most of these models have focused on process improvements [Misra, 2017].

Related papers

To improve existing APM tools it is necessary to investigate software development life cycle processes and activities of stakeholders' collaboration.

It is a precondition of appearing many scientific papers and vendor solutions addressed to solve this topic. Such papers are divided on two directions, namely to improving collaboration between stakeholders and features of software development life cycle processes.

Consider result of researches, directed to investigating processes of improving collaboration between stakeholders.

Paper [Misra, 2017] proposes to estimate user capabilities depending on their roles in software development process. User capabilities are identified in two categories: IT users who are IT experts and involved in design, development, and implementation of SDLC driven projects, and, second, non-IT users who, despite having inadequate or no exposure to IT, contribute to SDLC driven projects. The framework is implemented through Unified Modeling Language (UML) based approach. Paper contains detailed analysis of stakeholder's activities in every software development life cycle process. These roles are primarily end-users, planners, and domain experts (IT and non-IT). For every user role it is defined which UML diagrams solve tasks of such kind of users the best.

During the early stage of IT acquisition, managing IT activities relating to operation, programming, and data collection were the major areas of concern. In later stages the focus was on establishing a unit to look after various types of applications over an extended lifecycle, despite change in technology.

Authors also note that most organizations use different life cycle models for different projects. However, it is difficult to ascertain the survivability of the system thus developed for its expected life cycle. It is argued that most of the models popularly coming under SDLC have limitations in delivering good result in a complex scenario, but are successful in a tightly specified domain. All software models under SDLC can be characterized as a problem solving loop, which may go through four distinct stages: status quo, problem definition, solution integration, post-acquisition assessment.

Thus, they propose recommendation for improving software development life cycle processes. Also challenges for increasing of software development life cycle processes effectiveness are formulated.

Software development life cycle processes are the brick from which software development life cycle is consists. Successful management of all software development life cycle process is a very complex task due to the next causes:

- When software requirements changes other software development artifacts are changed too;
- Tools, platform and software environments are changing very quickly too. Thus, time to study and obtain practical skills with new environments is needed;
- Changing of technologies in turn leads to changing of some actions in performing software development processes;
- Some vendors adopt classical schemes [OMG, 2006].

Consider papers, relating to challenges of designing effective ALM and improving technologies or tools for performing different ALM tasks.

The paper [Grichi, 2015] deals with the verification of reconfigurable real-time systems to be validated by using the Object Constraint Language (abbrev, OCL). Authors propose an extension of OCL, named Reconfigurable OCL, in order to optimize the specification and validation of constraints related to different execution scenarios of a flexible system.

Also a metamodel of the new ROCL is proposed with formal syntax and semantics. This solution gains in term of the validation time and the quick expression of constraints.

But papers lack recommendation about ROCL implementing to increase effectiveness of software development lifecycle processes. Also software tools, supporting designed OCL extension, were not described.

Paper [Chebanyuk and Markov, 2016] presents an approach, verifying class diagram correspondence to SOLID object oriented design principles, is proposed in this paper. SOLID is an acronym, encapsulating the five class diagram design principles namely: Single responsibility, Open closed, Liskov substitution, Interface segregation and Dependency inversion.

To check whether class diagram meets to SOLID, its analytical representation is analyzed by means of predicate expressions. Analytical representation describes interaction of class diagram constituents, namely classes and interfaces, in set-theory terms. For every SOLID design principle corresponded predicate expressions are proposed. Also criteria for estimation of analysis results are formulated. But paper lacks representing this approach in some restriction language.

Paper [Chebanyuk, 2014] presents a method of behavioral software models synchronization. Implementing this method behavioral software models, which are changed after communication with customer, are synchronized with other software models that are represented as UML diagrams. Method of behavioral software artifacts synchronization makes the Model-Driven Development (MDD) approach more effective. For synchronization of different behavioral software models, transformation approach in the area of Model-Driven Architecture (MDA) is proposed. Synchronization operation is executed using analytical representation of initial and resulting models. Initial behavioral software model is represented by UML Use Case Diagram. Resulting behavioral software model is represented as UML Collaboration Diagram. Analytical representation of UML Use Case diagram allows considering data flows. For this representation set-theory tool operations are used. As a Collaboration Diagram usually contains more information in comparison with Use Case one, method defines two types of Use Case diagram fragments. From the beginning Use Case diagram fragments that can be transformed directly to resulting diagram constituents are considered. Then the rest of Use Case diagram fragments are processed to represents rules of placement Collaboration Diagram messages. These rules help to designate data flows, represented in Collaboration Diagram, more accuracy. Method, proposed in this

article, can be used both separately and be a part of more complex transformation technics, methods and frameworks solving different tasks in MDA sphere.

Paper [Filho, 2016] presents an approach for resource identification, management, and service discovery in Service Oriented Architecture (SOA). The service identification process consists of a combination of top-down and bottom-up techniques of domain decomposition and existing asset analysis. In the top-down view, a blueprint of business use cases provides the specification for business services. In the bottom-up approach, the analyst departs from a service identifying the provider entity, where application and container it is located. The idea is to reach a context view from a service. In order to reuse a service, clients need to know much more than a simple service name or the address of the service provider. Developers need to see a service as an interface, including methods that they will invoke in order to execute the service and their necessary parameters. The lookup service can be seen as a directory service, where services are found and viewed.

The descriptors specifications include: *(i)* the service name (the entity type that provides the service); *(ii)* the path (URL) where the service is allocated; *(iii)* the scope informing if the service is local (in the container) or remote; *(iv)* name and type of the parameters; *(v)* a brief description of the service functionality; *(vi)* a return informing if any data type returns to the caller service; *(vii)* the keywords related to the service; and *(viii)* implementation, informing if the service is: implemented in Java, a Web service, or a legacy service encapsulated as a service in a component [Filho, 2016].

The approach emphasizes an architectural model that allows representation, description, and identification of services, and is explored as a metadata repository. It is focused not only on Web Services, but also in all services existing in big companies' applications, including currently developed services and legacy system services, highlighting the importance of reusing fine granularity services. The model includes discovery procedures to find and retrieve candidates for services composition and reuse. These procedures adopt a Case-Based Reasoning approach, in which the services are considered as cases kept and indexed in a repository. Case matching is carried out by means of text mining techniques that allow finding the most appropriate service candidate with the desired requirements for a particular task.

Authors propose to store services in local store and describe service in WSDL format. The next scheme of service preparation is proposed

The process starts with a description of a service required by a developer. This description includes a functional account of the service representing the developer experience, beyond the usual descriptors like *name* and *parameters*. The system searches for a service in CB, guided by a given description, and then it retrieves a list of the best matching services. If a service satisfies the developer necessity, than it is applied. Otherwise, the alternatives are: *(i)* to search in the Web, *(ii)* to adapt a case from the

retrieved case list, or *(iii)* to develop a new solution. In any case, the case base must be updated [Filho, 2016].

But the problem of remote user: obtaining information about service if its functionality or address is changed. One of the solutions is to deploy additional data storage for services identification.

As requirement analysis is a very important process consider papers, related improvement quality of operations, performed in it.

Paper [Shamra, 2014] proposes an approach of generation sequence or activity diagrams from requirements, presented in natural text. Requirements analysis process involves developing abstract models for the envisioned or the proposed software system. However, software requirements are captured in the form of Natural Language and, generating UML models from natural language requirements relies heavily on individual expertise. Thus, authors present an approach towards automated generation of behavioral UML models, namely activity diagrams and sequence diagrams. Initial information for transformation is lexical and syntactic analysis of NL statements that is grounded on patterns. Authors propose an idea to analyze requirement specification involving natural language patterns.

Patterns – grammatical knowledge or domain-specific prove helpful in improving the quality of analysis. Knowledge patterns are divided on three types: lexical patterns for indicating a relation; grammatical patterns, which are combinations of part-of-speech; and, paralinguistic patterns, which include punctuation, parenthesis, text structure etc. [Shamra, 2014].

The idea of approach is based on transforming the requirements statements to intermediary structured representations – frames. Frames are special structures for representing knowledge using slots to store knowledge in an Object Oriented manner and, are an efficient means for reasoning. Then frames are translated to behavioral UML models. Authors use peculiarities of constructing sentences in English language, namely parts of sentences, times and verb forms, as well as examples of frames for composing prepositions, active, and passive voice. For analysis of sentence Stanford Dependency parser is used [Stanford, 2015].

Knowledge also are stored in frames is then used to automatically generate activity and sequence diagram. Also authors use common in activity and sequence diagram elements notations.

Paper [Inoue, 2015] proposes an extension of goal graphs in goal-oriented requirements engineering. First it is necessary to understanding the relations between goals. Goals specify multiple concerns such as functions, strategies, and non-functions, and they are refined into sub goals from mixed views of these concerns. This intermixture of concerns in goals makes it difficult for a requirements analyst to understand and maintain goal graphs. In our approach, a goal graph is put in a multi-dimensional space,

a concern corresponds to a coordinate axis in this space, and goals are refined into sub goals referring to the coordinates. Thus, the meaning of a goal refinement is explicitly provided by means of the coordinates used for the refinement. By tracing and focusing on the coordinates of goals, requirements analysts can understand goal refinements and modify unsuitable ones.

Paper [Escande, 2013] proposed method of defining requirement priority and assigning it to different stakeholders. Requirements are proposed to be assigned to different levels of priorities. Then, they are grouped by level of priority. And it is defined to which stakeholder concrete requirement or group of requirements is assigned. But paper lack to proposition according to which criterion some requirement is corresponded to which priority level. And also there is no concrete recommendations how to distribute requirements between different types of stakeholders.

The work [Klimek, 2012] concerns gathering requirements and their formal verification using deductive approach. This approach is based on the semantic tableaux reasoning method and temporal logic.

Authors ground the necessity of developing of implementing formal methods and approaches for requirement analysis performing [Klimek, 2012].

Formal methods can constitute a foundation for providing natural and intuitive support for reasoning about system requirements and they guarantee a rigorous approach in software construction. The main motivation for this work is the lack of satisfactory and documented results of the practical application of deductive methods for the formal verification of requirement models [Klimek, 2012].

Temporal logic is a well established formalism for describing properties of reactive systems. It may facilitate both the system specifying process and the formal verification of non-functional requirements which are usually difficult to verify [Klimek, 2012].

The semantic tableaux method is quite intuitive and has some advantages over traditional deduction strategies. System requirements are gathered using some UML diagrams. Requirements engineering based on formal analysis and verification might play an essential role in producing the correct software since this approach increases reliability and trust in software. Deductive inference is always the most natural for human beings and is used intuitively in everyday life. A use case, its scenario and its activity diagram may be linked to each other during the process of gathering requirements. When activities and actions are identified in the use case scenario then their workflows are modeled using the activity diagram. The automation of this process is crucial and constitutes a challenge in the whole deductive approach [Klimek, 2012].

Diagram is decomposed on components. For every component it is proposed to use pattern allowing to describe in text operation, that match to it.

Also authors mentioned that temporal logic properties and formulas may be difficult to specify by inexperienced users and this fact can be a significant obstacle to the practical use of deduction-based verification tools.

Automatic transformation of workflow patterns to temporal logic formulas is proposed. These formulas constitute logical specifications of requirements models. The architecture of an automatic and deduction-based verification system is proposed. Authors expect that applying this concept results in the reduction of software development costs as some errors might be addressed in the software requirements phase and not in the implementation or testing phases. But analyzing complex software may cause too long formulas difficult for further processing [Klimek, 2012].

Paper [Teruel, 2011] proposes techniques for increasing of effectiveness communication processes between stakeholders. Authors represent three Goal-Oriented approaches, namely NFR framework, *i** and KAOS, are evaluated in order to determine which one is the most suitable to deal with this problem of requirements specification in collaborative systems. These Goal oriented approaches aimed to increase requirement modeling processes. *i** framework is complicated with several relations that are not presented in classical UML diagrams. KAOS goal model lets to define dependencies between goals. This classification was elucidated by *i** framework. Authors represented comparative analysis of considered frameworks and define their peculiarities.

*i** only provides a partial support for quantifying the relations among requirements when using contribution links. The *i** approach also fails in representing the requirements importance, giving no support to determine which requirements are more important than others [Teruel, 2011].

Nevertheless, the other two GO approaches also share this lack of representation of the importance of each requirement. KAOS also fails in the same features than *i** but, unlike this approach, KAOS obtains a lower (or the same) score in almost all features except for the Hierarchical Representation feature, thanks to its tree-based representation. Finally, the NFR framework is the less suitable approach, obtaining a very low score, because of both the lack of expressiveness to specify FRs and its lack of adaptability to represent Collaborative Systems Characteristics. Also approach how to analyze requirement specification to match requirements to proposed classification of goals and sub-goals do not identified [Teruel, 2011].

ANALYSIS OF EXISTING APPLICATION LIFE CYCLE PRODUCTS ON THE MARKET

There are many ALM tools available for tracking application changes. These range from dedicated ALM products that monitor an application from inception to completion, automatically sorting files into logical buckets as changes are noted, to simple wikis that require team members to record changes manually,

this section reviewed the most popular among them, namely: Visual Studio, IBM Rational Team Concert, FusionForge and Team Forge

1.1 Microsoft Visual Studio with Team Foundation Server

Microsoft Visual Studio (VS) is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services.

Microsoft Visual Studio has the next features:

- allow to write and debugg code;
- has a forms and data Designer;
- allow calculate code metrics;
- has plugins support;
- support source control via Team Foundation Server or Git.

Microsoft Visual Studio is Integrated with Team Foundation Server (TFS) proposing common Microsoft product that provides the next functions:

- source code management (either with Team Foundation Version Control or Git);
- reporting;
- requirements management;
- project management (for both agile software development and waterfall teams);
- automated builds;
- lab management;
- testing;
- release management capabilities.

TFS is often used on large enterprises. Free version of the product is an IDE (has limitations in code editor, etc.) with minimal possibilities (version control system, class diagram, metrics calculation, etc.) for life cycle support. In turn, the paid version allows you to not only make full use of the code editor, but also almost fully support the life cycle of the developed software [Microsoft, 2015].

1.2 IBM Rational Team Concert

Rational Team Concert (RTC) is a software development team collaboration tool developed by the Rational Software brand of IBM, who first released it in 2008. The software is available in both client

versions and a Web version. It provides a collaborative environment that software development teams use to manage all aspects of their work. It has the next features:

- supports all main paradigms of a software development;
- allow to create local source control;
- defect tracking;
- build Management;
- has a customized dashboard;
- allow to tracking changes in items;
- has a report system for fast-tracking of detected defects.

As well as Microsoft Visual Studio, IBM Rational Team Concert is focused on large enterprises and the base package only allows you to plan, schedule and monitor progress of work, version control and bug/feature tracking. In order to design, system requirements analysis, etc. you need not buy separate IBM Rational products [IBM b), 2015].

1.3 FusionForge

FusionForge (FF) is a free software application descendant of the forge (web-based project-management and collaboration software) originally created for running the SourceForge.net platform. FusionForge is licensed under the GNU General Public License, and is a fork/renaming of the code which was previously named GForge.It has the next features:

- provides version control by using GNU arch, Bazaar, CVS, Darcs, Git or Subversion)
- allow to bug/feature-tracking;
- has own messaging feature, that can be deployed to run a self-hosted forge;
- allow to create surveys for users and admins;
- plugins support;
- allow to create wiki for systems;
- has a task management system.

FusionForge well suited for medium and small teams. Environment allows automating some of the life cycle processes, flexibly configuring the environment for development team style (e.g. setting of fields to keep track of bugs). Of the downsides can highlight the need for a local server to work with the environment (in case of problems with the server team will not not have an access to the tasks and

repositories, due to lack of a desktop client), also environment does not provides tools for software design (e.g. UML support) [Fushionforge, 2016].

1.4 TeamForge

TeamForge (TF) is a collaborative revision control and software development management system. It provides a front-end to a range of software development lifecycle services and integrates with a number of free software / open source software applications (such as PostgreSQL and Subversion).It has the next features:

- has a revision control;
- software development management system;
- bug tracking system;
- allows you to track the progress of performed work;
- allows you to keep track of tasks after release (bug/feature completion tracking);
- allows to create any discussions for platform users;
- can be integrated in other software (Visual Studio, Microsoft Office, Eclipse and so on).

This environment is mainly focused on large companies that use additional software in the development of programs (for example Microsoft Office, Visual Studio, Outlook etc.) and its own servers (for deploying lifecycle environment). But, this environment, does not allow to build UML diagrams (does not support design process).

Table 1 represents results of comparison analysis of ALM environments [Teamforge, 2016].

As you can see from the Table 1 listed lifecycle management environments has both advantages and disadvantages. Disadvantages of considered ALMs are the next:

- need to be purchased (Visual Studio, IBM Rational Concert, TeamForge);
- not available on other operating systems (Visual Studio);
- lack of support of the design process (FusionForge, TeamForge, IBM Rational Concert (only in additional packages);
- lack of software testing tools (FusionForge, TeamForge, IBM Rational Concert (only in additional packages);
- lack of code editor (FusionForge, TeamForge, IBM Rational Concert (only in additional package));
- no desktop client (FusionForge, TeamForge);
- attachment only to the local server (FusionForge, TeamForge).

Table 1. Comparison of ALM environments

	VS	FF	TF	RTC
Version control	+	+	+	+
Requirements management	Only in TFS	+	+	+
Bug/feature-tracking	Only in TFS	+	+	+
Plug-in support	+	+	-	Can only integrates in others IBM's products
Code editor	+	-	-	+
Code debugger	+	-	-	+
GUI designer	+	-	-	-
UML support	Partially	-	-	In Rational Software Architect
Code generation from models	+	-	-	+
Task management	+	+	+	+
Unit Testing	+	-	-	In Rational Functional Tester
Functional testing	-	-	-	In Rational Functional Tester
Generation of developer's documentation	-	-	-	-
Cross-platform	-	+	+	+
Free to Use	Only Community version (Limited version)	-	+	-

Task and challenges

Analyzing review of papers and tools formulate requirements for designed ALM environment formulate requirements to future system:

- support software developing at all stages of the life cycle;
- it must be cross-platform;
- can be deployed on local computer;
- contain modules supporting stakeholders collaboration.
- support following functions:
  - task managing
  - defining requirements;
  - designing new software by using a UML diagrams;
  - implementing a new software by using Eclipse IDE, UML to code convertor etc.;
  - testing developed software;
  - maintenance developed software (automatic creation of developers documentation, bug tracking list);
  - and monitor changes in the project with the help of git version control).

Proposed approach

Designing good application life cycle management environment should consider possibilities to change performing of some processes by means of defining proper configuration of plugins.

To simplify software development lifecycle management we use Eclipse platform that anbles to plug a variety of free plug-ins, among which there are plug-ins for software lifecycle management.

The Figure 1 containing a graphical representation of plug-ins that can be used on the phases of a typical software development life cycle (SDLC).

Represent description of chosen plug-ins: functionality:

- UML to Java Generator extends functionality of UML Designer – allows to generate java code from UML Class Diagrams;
- WindowBuilder extends functionality of Eclipse IDE – allows to build GUI;
- Eclipse Color Theme extends the capabilities of the Eclipse IDE code editor and Data Tools Platform sql editor;
- JUnit extends functionality of Eclipse IDE – allows to write Unit tests to Java classes;

- STAN extends functionality of Eclipse IDE – allows to analyze code that was written in Eclipse IDE;
- FindBugs extends functionality of Eclipse IDE – allows to find possible bottlenecks, errors and bugs in code;
- Jubula extends functionality of Eclipse IDE – allows to write and test program that was written in Eclipse IDE;
- JavaDoc extends functionality of Eclipse IDE – allows to write developer's documentation for the written code in Eclipse IDE.

Data flow between environment components and software development artifacts is represented on Figure 2.

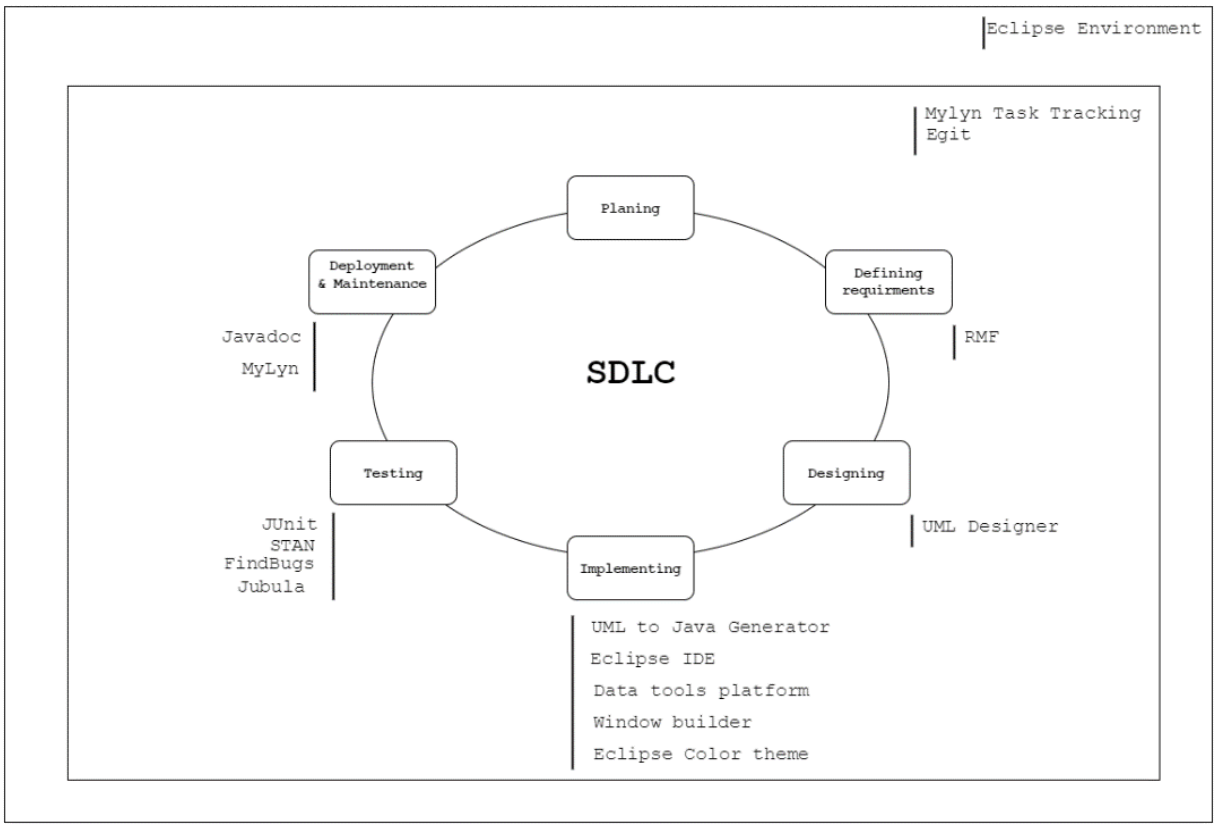

Figure 1. Plug-ins collaboration used to design a typical software development

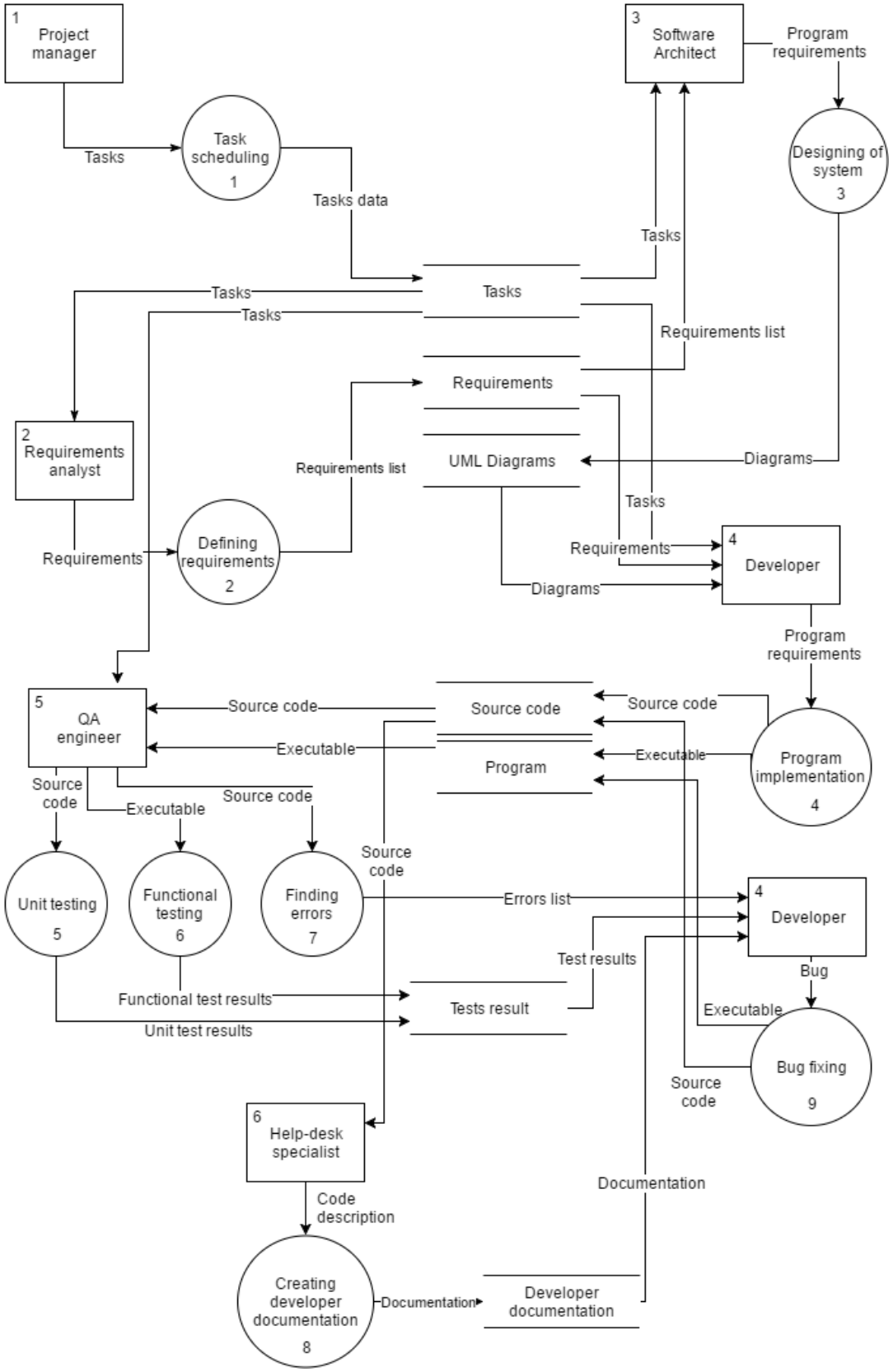

Figure 2. Data flow between ALM environment components

Represent Data flow between environment components

1. Project manager uses Mylyn for task scheduling;
2. From Mylyn Requirements analyst receive tasks for completion and by using RMF creates a requirements list;
3. Software architect receives tasks from Mylyn and Requirements list from EGit creates UML diagrams by using UML Designer;
4. Developer receives tasks from Mylyn, Requirements list and UML diagrams from EGit implements program in Eclipse IDE;
5. QA engineer receives tasks from Mylyn and source code from EGit creates Unit tests and perform Unit tests in JUnit;
6. QA engineer receives tasks from Mylyn and executable from EGit creates Functional in Jubula;
7. QA engineer receives tasks from Mylyn and source code from EGit perform Finding error process in STAN and FindBugs;
8. Help-desk specialist receives tasks from Mylyn and source code from EGit creates Developer documentation in JavaDoc
9. Developer receives error list, test results and developer documentation from EGit fixes bug in Eclipse IDE;
10. Tasks, Requirements, UML Diagrams, Source code, Program, Tests result and Developer documentation are stores in EGit.

Schema, representing matching with plug-ins and stakeholders' activities is represented in figure 3.

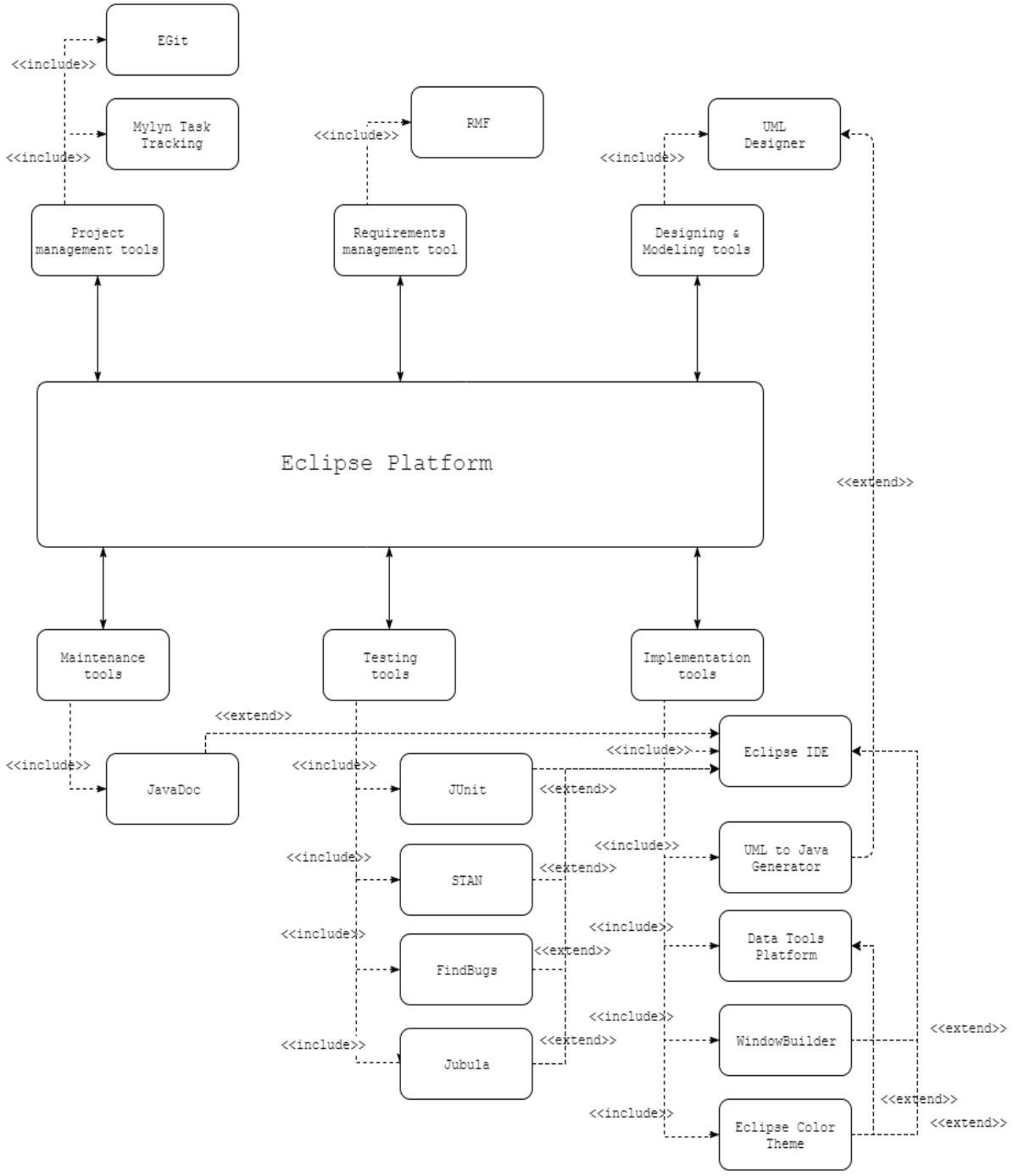

Figure 3. ALM environment and plug-ins interaction scheme

Work processes for each stakeholder role

Analyzing schemas on Figures 1-3 analyze work processes for each stakeholder role.

Project manager activities

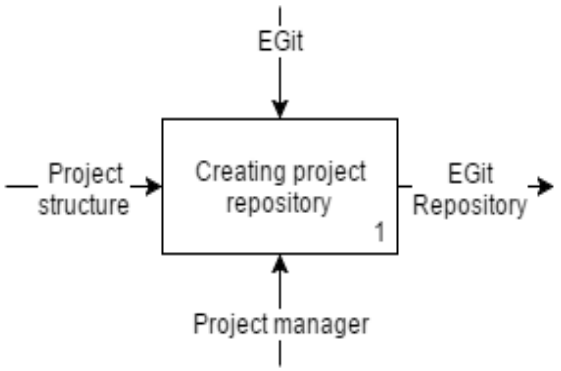

Figure 4. Creating project repository

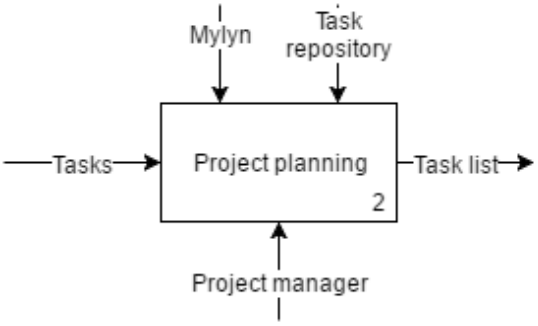

Figure 5. Project Planning

Requirement analyst activities

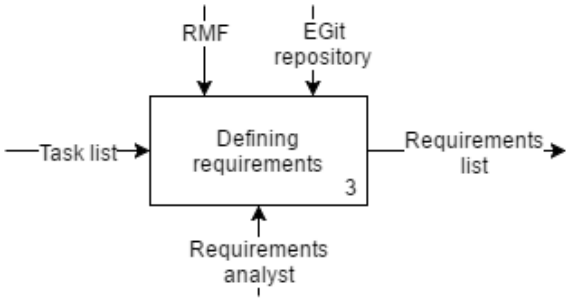

Figure 6. Defining requirements

Software architector activities

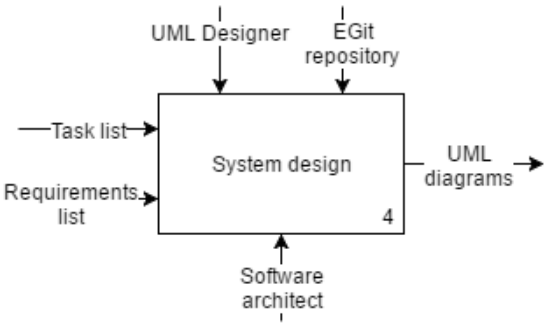

Figure 7. System designing

Software developer activities

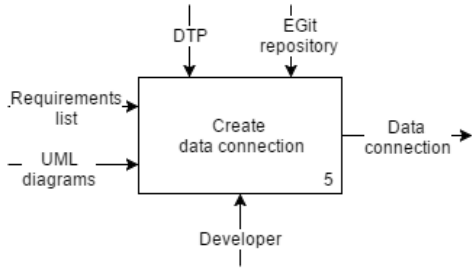

Figure 8. Create data connection

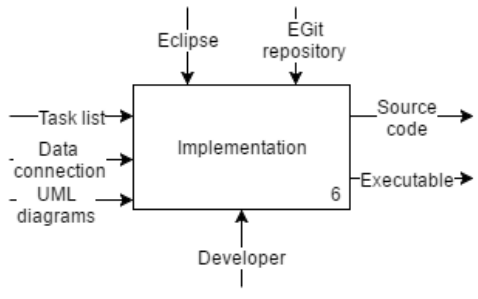

Figure 9. Upload(download) Implementation

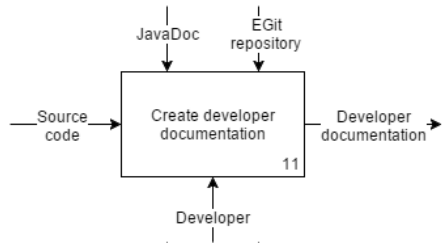

Figure 10. Create developer documentation

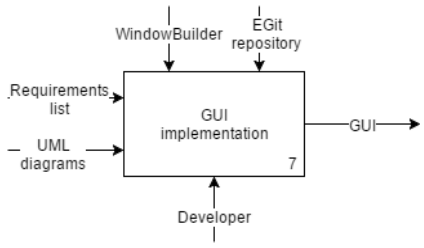

Figure 11. Upload(download) GUI Implementation

Activities of QA Engineer

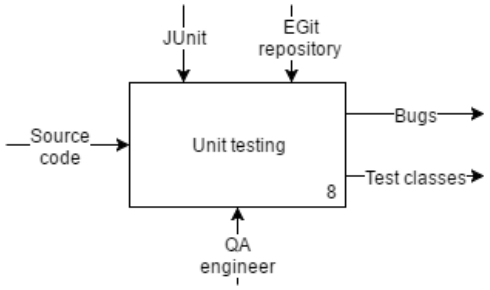

Figure 12. Unit testing

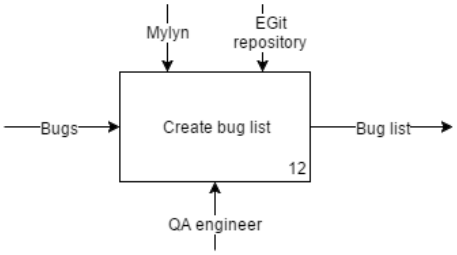

Figure 13. Create bug-list

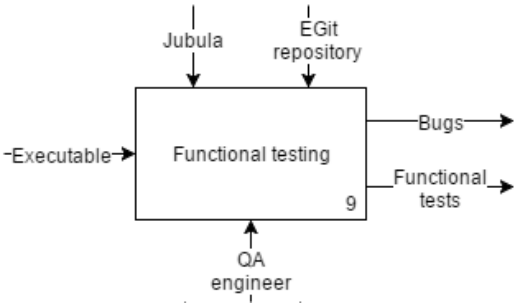

Figure 14. Functional testing

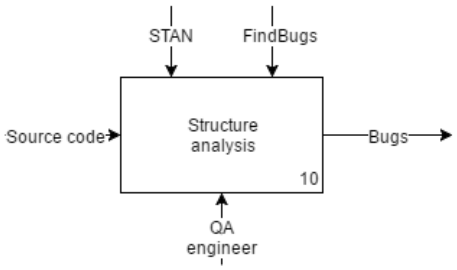

Figure 15. Structure analysis

Activities of Help-desk specialist

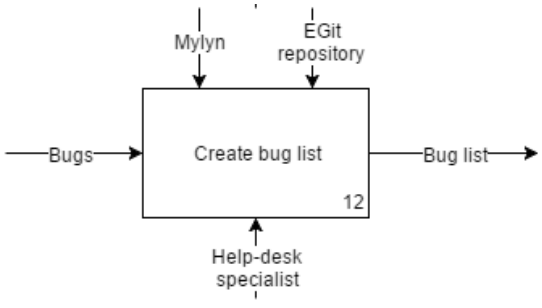

Figure 16. Create bug list

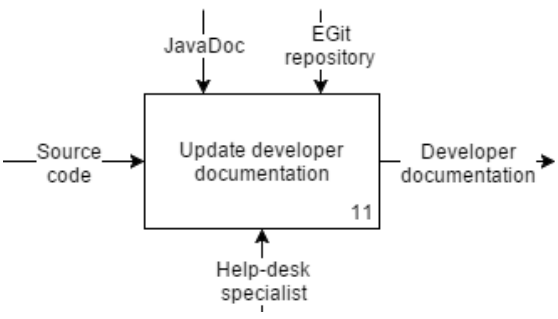

Figure 17. Update developer documentation

Conclusions

Cross-platform environment for supporting ALM that is based on Eclipse is proposed in this paper. It satisfies to all challenges, formulated above, namely cross-platform support; free; extensible by means of setting flexible plug-ins configuration; supporting extensible stakeholders activities and organizing different data flows between modules by means of adding new plugins; can be configured on local computer; also can be deployed on server for stakeholders interconnection. Represented activities according stakeholders roles are focused on actions, performed by stakeholders. In order to adopt software development life cycle processes to needs of specific enterprise sequence of actions, and consequently configuration of plugins can change. As eclipse provides supporting desktop, cloud and platforms IDEs, proposed ALM can be integrated with various programming languages, for example by means of creating developing perspectives [Eclipce, 2015].

Implementing of designed ALM will allow reducing the time required to develop new software, will increase the involvement of the team in the development process. In this turn, project managers will be able to track the progress of the project and identify the risks of disrupting the work schedule.

With the advent of such software environments in various industries, software developers and other stakeholders will spend less time on development, which in turn will reduce the cost of development.

Further research

To propose an ALM approach, facilitating requirement analysis and software designing; focusing on Model-Driven techniques, making accent on code generation techniques.

Bibliography

[Chebanyuk and Markov, 2016] Chebanyuk E. and Markov K. (2016). An Approach to Class Diagrams Verification According to SOLID Design Principles.In Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD, ISBN 978-989-758-168-7, pages 435-441. DOI: 10.5220/0005830104350441 http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=HASwCJGMcXc=&t=1

[Chebanyuk, 2014] Chebanyuk, Elena.2014. Method of behavioural software models synchronization. International journal Informational models and analysis. – 2014, №2 P 147-163 http://www.foibg.com/ijima/vol03/ijima03-02-p05.pdf

[Eclipce, 2016] https://eclipse.org/ide/

[Escande et al., 2013] Loup Escande E. and Christmann O. (2013). Requirements Prioritization by End-users and Consequences on Design of a Virtual Reality Software - An Exploratory Study.In Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE, ISBN 978-989-8565-62-4, pages 5-14. DOI: 10.5220/0004397900050014

[Filho et al., 2016] Filho A., do Prado H. and Ferneda E. (2016). A Metadata-based Architecture for Identification and Discovery of Services in SOA..In Proceedings of the 18th International Conference on Enterprise Information Systems - Volume 2: ICEIS, ISBN 978-989-758-187-8, pages 298-305. DOI: 10.5220/0005867702980305

[FusionForge, 2016] https://fusionforge.org/

[Grichi et al., 2015] Grichi H., Mosbahi O. and Khalgui M.. ROCL: New Extensions to OCL for Useful Verification of Flexible Software Systems. DOI: 10.5220/0005522700450052 In Proceedings of the 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 45-52 ISBN: 978-989-758-114-4

[IBM b), 2015] https://jazz.net/library/article/632

[IBM, 2015] http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=APW12347USEN

[Inoue et al., 2015] Inoue W., Hayashi S., Kaiya H. and Saeki M.. Multi-dimensional Goal Refinement in Goal-Oriented Requirements Engineering.DOI: 10.5220/0005499301850195 In Proceedings of the 10th International Conference on Software Engineering and Applications (ICSOFT-EA-2015), pages 185-195 ISBN: 978-989-758-114-4

[Klimek, 2012] Klimek R. (2012). Proposal to Improve the Requirements Process through Formal Verification using Deductive Approach.In Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE, ISBN 978-989-8565-13-6, pages 105-114. DOI: 10.5220/0004001901050114

[Microsoft, 2015] https://www.visualstudio.com/tfs/

[Misra, 2017] Harekrishna Misra Managing User Capabilities in Information Systems Life Cycle: Conceptual Modeling. International Journal of Information Science and Management Vol. 15, No. 1, 2017, 39-58 http://ijism.ricest.ac.ir/index.php/ijism/article/view/936

[OMG, 2006] ftp://ftp.omg.org/pub/presentations/ajw_alm/ALM.pdf

[Sharma et al., 2014] Sharma R., Gulia S. and Biswas K. (2014). Automated Generation of Activity and Sequence Diagrams from Natural Language Requirements.In Proceedings of the 9th International

Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE, ISBN 978-989-758-030-7, pages 69-77. DOI: 10.5220/0004893600690077

[Stanford, 2016] http://nlp.stanford.edu/software/stanford-dependencies.shtml

[Teamforge, 2016] https://www.collab.net/products/teamforge-alm

[Teruel et al., 2011] Teruel M., Navarro E., López-Jaquero V., Montero F. and González P. (2011). A COMPARATIVE OF GOAL-ORIENTED APPROACHES TO MODELLING REQUIREMENTS FOR COLLABORATIVE SYSTEMS.In Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE, ISBN 978-989-8425-57-7, pages 131-142. DOI: 10.5220/0003466301310142

Authors' Information

Elena Chebanyuk *– Assoc. Prof. of Software Engineering Department, National Aviation University, Kyiv, Ukraine,*
Major Fields of Scientific Research: Model-Driven Architecture, Model-Driven Development, Software architecture, Mobile development, Software development,
e-mail: chebanyuk.elena@ithea.org

Oleksii Hlukhov *– student of Software Engineering Department, National Aviation University, Kyiv, Ukraine.*