UDC 681.518:658.512

MODELS AND TOOLS FOR EFFECTIVENESS INCREASE OF REQUIREMENTS TRACEABILITY IN AGILE-SOFTWARE DEVELOPMENT

M.V. Tkachuk[+], *R.O. Gamzayev*[+], *H.C. Mayr*[*], *V.O. Bolshutkin*[*]

+ National Technical University "Kharkiv Polytechnic Institute", 61002, Frunze Str., 21, Kharkiv Ukraine
* Alpen-Adria University of Klagenfurt, A-9020, Universitaetsstr. 65-67, 9020 Klagenfurt, Austria
tka@kpi.kharkov.ua, rustam.gamzayev@gmail.com, mayr@ifit.uni-klu.ac.at, vladimir.bolshutkin@gmail.com

This paper presents the Agile-centered framework for advanced requirements traceability support which consists of following components: a procedure to build an advanced traceability matrix (ATM) using the elaborated time-oriented metric for quantitative estimation of project activities and developer's interests; a special CASE-tool to combine the functions of typical requirements management system and the functionality of integrated development environments (e.g., Eclipse). ATM provides assessment of relationships between requirements and software projects artifacts due to gathering and processing all developer's activities in time-oriented retrospective data model.

Ця стаття презентує Agile-орієнтований підхід для підтримки трасування вимог, який складається з наступних компонентів: процедури побудови розширеної матриці трасування вимог (АТМ) із використанням метрики, яка враховує час, для чисельного оцінювання активності та ступеня інтересу розробників проекту; спеціального CASE-засобу, що об`єднує можливості типових системи управління вимогами та функціональність інтегрованих середовищ розробки програмного забезпечення (наприклад, Eclipse). Матриця АТМ забезпечує можливість кількісної оцінки ступеню зв'язків між вимогами та програмними артефактами шляхом накопичення та обробки ретроспективних даних щодо дій розробників проекту.

1. Introduction: problem actuality and statement of work

Requirements management (RM) is one of the most critical and weak-formalized disciplines in modern software engineering (SE), because of permanent changes in business logic and /or in projects configuration (tools and technologies used). This is actually the real challenge for any new software system to be done "from scratch". From the other hand even already existing software systems (or legacy systems) also have to be modified permanently with respect to new customers needs.

Nowadays to overcome these problems several adaptive or flexible SE-methodologies called as agile software development (ASD) are elaborated and used [1]. The main concepts of any ASD-methodology supposed to meet requirement changes in more efficient way are the following:

organizing a closed collaboration between several actors involved in ASD: stakeholders, analysts, developers, end-users;

usage of an iterative approach for the project development;

providing requirements traceability in order to reflect all requirement changes into appropriate projects artifacts.

In order to support all activities needed for (1)-(3) it is necessary to elaborate some new models and tools to be used in ASD-projects for RM generally, and especially for efficient requirements traceability. In this paper we present an ASD-centered integrated modeling and technological framework for this purpose. Notably, that the concept of requirements traceability has several meanings, for example: tracing requirements from their sources, tracing low-level requirements to high-level requirements or tracing requirements to project artifacts used to implement them. In this paper we imply tracing requirements to project artifacts.

ASD methodologies are widely used due to new character of the software projects and due to permanent changes in the project dynamic. New business issues lead to high requirements volatility and necessity to respond to them. ASD implies refusal from complete bureaucratic documentation, but really needs support for automated classification of knowledge, in order to reduce the time for adaptation to requirements changes and to minimize risks related to them. Exactly because of these reasons in the modern SE-concepts process control methods called also as "software cybernetics" (e.g. in [2]) are used. On Fig. 1 the cybernetics-centered scheme is shown, which represent one of the most widespread ASD-methodology, namely Scrum method [3]. There are 2 feedback loops included in this control scheme: 1) daily process control, basing on sprint backlog (SB) and using some source code quality metrics, 2) iteration process control, basing on product backlog (PB) and using some requirements quality metrics. The SB and PB both collect the selected requirements to be met in final software product. In the control loop (1) usually some IDE, e.g. Eclipse is used by developers team, while in the control loop (2) an appropriate requirements management system (RMS) can be exploited by stakeholders (product owners - in Scrum) and by domain analysts.

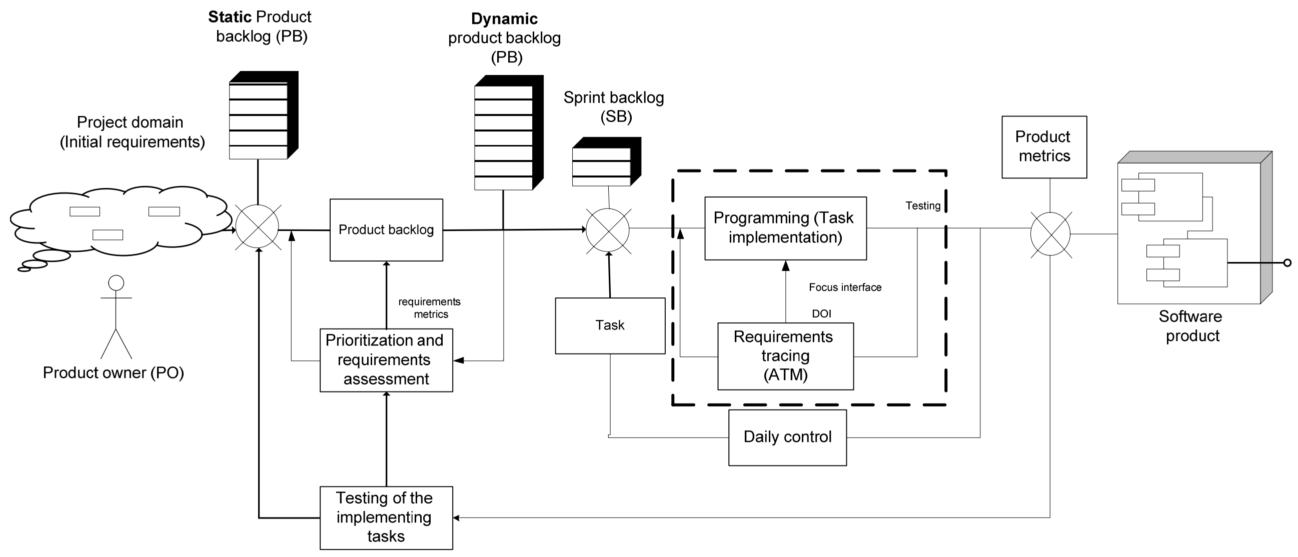

Static Product backlog (PB)
Dynamic product backlog (PB)
Sprint backlog (SB)
Project domain (Initial requirements)
Product owner (PO)
Product backlog
requirements metrics
Prioritization and requirements assessment
Task
Programming (Task implementation)
Testing
Focus interface
DOI
Requirements tracing (ATM)
Daily control
Product metrics
Software product
Testing of the implementing tasks

Fig. 1. Cybernetic-centered scheme of Scrum-methodology

Therefore it is clear that in order to organize a close cooperation between all projects participants on the conceptual level, and to support them in computerized way we need: (a) to reflect permanently all requirements changes in source code through an efficient traceability procedure, (b) to elaborate an appropriate CASE-tool which combines IDE and RMS functionality both. To solve this problem it is necessary to trace not only requirements changes and refinements but also to trace requirements to all relevant project artifacts (including source code). Many traceability models and techniques were proposed for this purpose, but the most widespread is a traceability matrix. Although this model of requirements tracing is quite simple, its usage causes some problems, which are considered more detailed below (see in Section 2).

We propose to solve these problems in the way to build so-called advanced traceability matrix (ATM), which allows to take into account all developers activities in time-oriented data model, with respect to project tasks context and developer's roles. Besides that we elaborate an integrated CASE-tool in order to combine the functions of typical requirements management system and the functionality of integrated development environments (e.g. Eclipse).

This paper is organized in the following way: in Section 2 some related works are briefly discussed, Section 3 introduces our approach to improve requirements traceability process in ASD basing on Scrum example, Section 4 describes the architecture and some special functionality of implemented CASE-tool prototype, Section 5 represents test case and results analysis, and finally, in Section 6 we come to conclusions and discuss possible future work.

2. Related works

Because of the considered problem's complexity it is necessary to analyze some activities already done in several interconnected research areas and application domains, which are briefly represented below.

Some approaches for requirements traceability representation and analysis. In the fundamental work [4] basic reference models for presentation and analyzing of relationships between several kind of project's artifacts and traceability links is proposed. These models are divided in two levels: with respect to needs of high-end users, and low-end users, and it is proved that appropriate traceability schemes should have different level of details. The paper [5] provides an approach for requirements traceability which is based on association rule mining, a special technology to be applied on large databases including sets of interconnected items. If a developer performs some programming task, all relevant project artifacts such as design documents, source code, multimedia files, etc. that might require corresponding changes, can be identified to be processed too. Examples of building several kinds of traceability matrixes are given in [6, 7], and in such works as, e.g. [8, 9], a cybernetics-centered approach for requirements management in general, and especially for requirements changes control in agile-development is proposed. On the other hand, some authors contend that exactly traceability matrix as a modeling tool is not useful for agile-project supporting, due to more informal and non-well documented approaches used in this case. The other well-known issue of traceability matrices is their big size, so it is difficult to read. Moreover, the paper [10] represents a matrix-less model to achieving requirements traceability that is equally applicable to both agile and traditional software development.

Task-focused user interface model. The task-focused interface is a type of a user interface which hides entire hierarchies of information, such as a file- system tree, and show only necessary subset of the tree that is relevant to the active task. This solves the problem of information overload when dealing with large hierarchies.

The main goal equals the goal of requirements traceability and may be treated as a way of representation traceability information.

The task-focused interface contains a mechanism which allows developer to specify the task being worked on and to switch between active tasks, a model of the task context such as a degree-of-interest (DOI) [11] function, a focusing mechanism to filter or highlight the relevant projects artifacts related to this task.

Formally, DOI function introduces the conception, but does not specify an algorithm for calculation. That is why we can use different metrics to calculate it. *DOI(r,f)* is a degree of the interest within working on the requirements r to artifact f, and it depends on the following elements:

$events_t$ - interaction events recorded while working on requirement r;

f – target artifact;

$relations$ – set of relations between artifacts.

The first implementation of the task-focused interface started as an open source project called Mylar (now Eclipse Mylyn). In [12] was elaborated an algorithm of DOI calculation which gave statistically significant improvement of programmer's productivity. To analyze quantitative efforts of using task-focused interface, an edit ratio metric was introduced. Edit ratio means the number of keystrokes in the editor over the number of structured selections made in the editor and views. Mylyn uses interaction history model that is shown at Fig. 2.

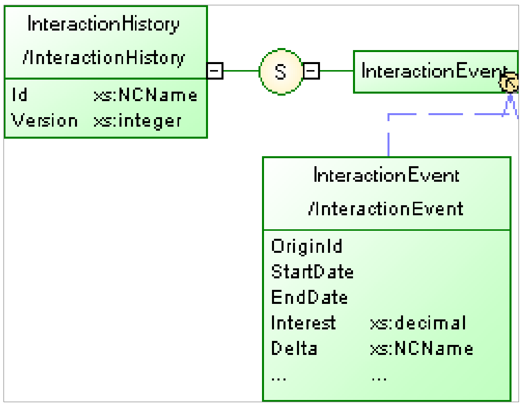


Fig. 2. Mylyn interaction history model

But this approach cannot be used to improve team work productivity because DOI is calculated only by previous experience of the developer and is not shared.

Existing integrated software solutions. There were attempts to integrate RM tool directly into IDE. The most significant example is RMS Borland CaliberRM that has plugins for most of Borland's IDE such as Borland JBuilder and Borland Delphi.

This integration implies introducing requirement management features inside generic IDE interface and makes learning curve plainer and improves effectiveness of IDE usage [13]. This feature is also very important when people roles in software have multiple development project. This situation is inherent in ASD, because of team multi-functionality. Similar integration solutions exist only for full-featured RMS and are heavy and expensive for small- and middle-size teams which typically are working on a gile-software projects.

There are a lot of different applications to improve development effectiveness and some of them consider whole project lifecycle and try to integrate different tools to be used in convenient way. The most advanced and popular tools are AccuBridge and Atlassian FishEye. Common feature is that these systems integrate tools developed by AccuRev and Atlassian correspondingly into development environment. Especially software configuration management (SCM) tools and issue tracking systems are the main components of such integration. AccuBridge provides an open SDK to make it possible for community developers implement connectors to extra tools. The core of AccuBridge is AccuRev SCM [14]. Atlassian FishEye is a tool to trace JIRA issues to related source code changes (commits). It is implemented as Eclipse plugin and also depends on Eclipse Mylyn [15].

Described tools do not solve problems which we have formulated. Moreover, RM is not considered in these tools at all as they are Agile-oriented and Agile methodologies do not strictly define this process. But some of those ideas are very interesting and may be reused and developed.

To summarize, we can say that chosen direction is perspective as there are a lot of related works and applications. Despite this, existing approaches have their shortcomings on solving problem we have formulated. So it makes sense to continue research in this direction

3. Proposed approach

We propose an approach based on integration of IDE with other development tools to make possible requirements traceability improvements.

At the highest level, our approach consists of the following logical components:

- automated statistic gathering;
- calculation of advanced traceability matrix (ATM) elements values for each developer;
- task-context initialization for new issues.

We are going to make traceability improvements by introducing the ATM that is built automatically using data gathered from different sources like IDE instances, RM solutions and other tools such as version control systems, issue tracking systems (ITS) etc. But for this model to be applied for real-world problems it is also necessary to support processing of ATM to make developers' work more convenient. Therefore, our idea is to take advantages of the focused interface and predict developers' interests on newly created tasks linked to requirements.

Thereafter, we will cover every logical components of our proposal in details.

Data Gathering and Storage. An important issue of any integration is choosing the approach to ensuring access to the data for all components. From this perspective, it is customary to distinguish data consolidation and data federalization. Unfortunately, at present there is no single accepted model of storage requirements. In different systems management requirements use different data models that often use the meta-model.

Therefore we have chosen an approach closer to the data federalization. This approach involves creating document-oriented database, recording which will include the URL of records from external storages. This will avoid duplication of data, but leave the possibility of finding detailed information about the required elements.

We elaborated time-oriented model to support traceability retrospective and to allow analysis of usage statistics. This model is based on the interaction history model and implies recording all interaction events to store them in the database.

Advanced Traceability Matrix. As mentioned above, the most widespread approach to RT is the traceability matrix. The formal model of requirements traceability matrix is very simple and can be written as follows:

$$M \subseteq R \times F \quad , \tag{1}$$

or

$$M : (R \times F \to \{0,1\}) \quad , \tag{2}$$

where $\boldsymbol{R}$ - the set of all requirements,

$\boldsymbol{F}$ may have different meanings, according to the purposes of constructing traceability matrix (usually, files, components, test cases). Later we will treat $\boldsymbol{F}$ as set of files.

This is a very simplistic model and it shows only whether there is a relationship between the task and the artifact but not the degree of relationships. However, without information on degree of relationship we cannot help developers to deal with information overloading. If we treat every file as important on single line change, we will get huge and useless traceability matrix.

So, our proposal is based on extending matrix domain:

$$M : (R \times F \to [0,1]) \quad , \tag{3}$$

We named such matrix ATM. This model leaves us a possibility to choose appropriate metric for ATM elements calculation and to define coefficient that shows border between important and unimportant files.

If we already have some metric or algorithm for calculating DOI, we can reuse it for ATM calculation. The only aspect is necessity of normalizing such DOI values to let ATM elements values be in interval [0,1].

For instance, for a set of requirements that are mapped into a set of tasks (done to meet the requirements), *DOI(t,f)* are calculated for each task and not for each requirement. We are assuming that every requirements could be divided into the set of the tasks that are assigned to the developers. In this case the elements of ATM matrix could be calculated with the following formula:

$$m_{rf} = \frac{\sum\limits_{t \in C_r} DOI(t, f)}{\sum\limits_{\substack{t \in C_r \\ f' \in F}} DOI(t, f')}, r \in R,\ f \in F \quad , \tag{4}$$

where $r \in R$ – requirement,

DOI(t,f) – developer's interest to file *f* in frames of activity t calculated on the basis of previous experience,

C_r – set of tasks related to requirement r,

$C \subseteq T \times R$ – relationship matrix between tasks and requirements.

Time-oriented metric. One of the simplest ways to define ATM calculation metric is to take into account a fact, that some files demand more time when working on some requirement. Using time-oriented metric we can distinguish between small cosmetic change and deep redesign and mark files with small changes as unimportant.

The degree of relationship between the file and the requirement can be calculated using the following metric:

$$m_{rf} = \sum_{k=1}^{\|A_r\|} \frac{\sum_{i=1}^{\|F_a\|} (t_{f_i}^{(i,2)} - t_{f_i}^{(i,1)})}{t_{a_k}^{(2)} - t_{a_k}^{(1)}}, r \in R, f \in F \quad , \tag{5}$$

where A_r – set of activities related to requirement r,
F_a – set of artifacts changed by activity a,
$t^{(1)}_{Ark}$ – start time of activity Ark ,
$t^{(2)}_{Ark}$ – end time of activity Ark ,
$t^{(i,1)}_{f}$ – time of i-th activation of file f ,
$t^{(i,2)}_{f}$ – time of i-th deactivation of file f.
Graphical interpretation of this metric is shown on Fig. 3.

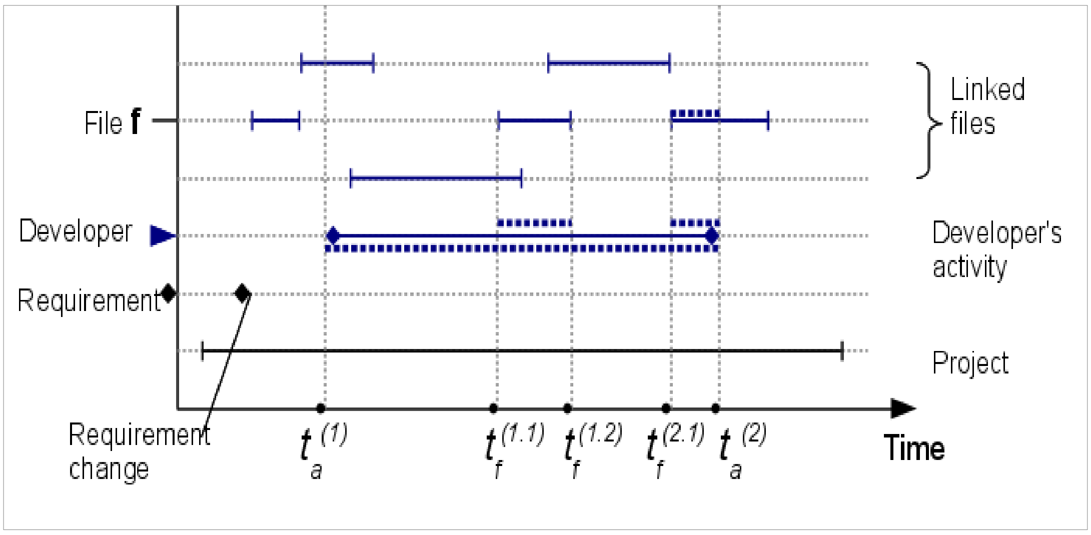


Fig. 3. Time-oriented metric interpretation

Of course this metric has its drawbacks: it cannot deal with situation when some file was opened but no work was actually done with it. But it is quite simple and applicable for preliminary validation of proposed approach.

Task-focused UI integration. By itself, the traceability matrix in the sense we are considering is a compilation of data about the links between requirements and project artifacts. However, to improve efficiency the data must be quickly and accurately processed. Standard traceability matrix can be very large, which may complicate its analysis.

To solve this problem, our approach involves the integration with the implementations of task-focused UI. In this case, the developer will be informed directly about the project artifacts, which require revising.

We assume that any activity that changes project artifacts in the process of developing is related to certain requirements and aims to meet them. This may be a realization of completely new demands or work on the existing requirements - adaptation to requirements' changes or fixing found inconsistencies.

4. Integrated CASE-tool: architecture and some implementation details

To get a proof-of-concept and to check a feasibility of proposed approach, models, algorithms and metrics, we have developed a tool called ReqMIT (Requirements Management Integrated Tool) . It is integrated into IDE and is able to gather retrospective data and predict future changes in files when requirements change. The prediction is done using ATM built on the gathered data.

In this section we describe software architecture of the elaborated CASE-tool, some of design solutions such as technologies.

System architecture. Designed architecture is service-oriented, to allow extension of current solution to add more features in future. Now we are integrating only IDE and ITS, but in software development many other tools can be used.

To integrate different types of tools that are used in development we decided to use the messaging approach to the integration.

Basic RMS features, such as CRUD (Create-Read-Update-Delete), are built-in in ReqMIT system. This allows us doing research on requirements traceability without dealing with other aspects of requirements management.

Our design solution has two databases: one for storing data related to developers, their roles and requirements; and second for retrospective information such as interaction events.

UML deployment diagram of elaborated software is shown at Fig. 4.

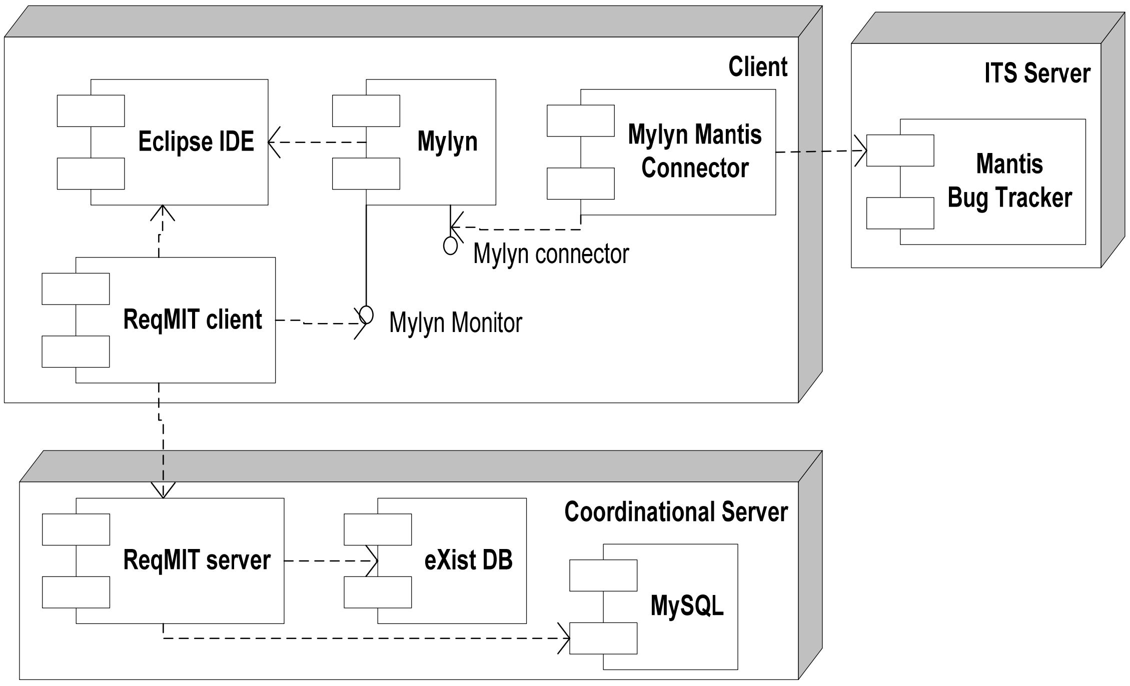


Fig. 4. Component structure of elaborated prototype

Chosen technologies. The proposed approach is independent of concrete tool. But we have chosen Eclipse IDE because it is one of the most popular development environments.

Another reason to choose Eclipse is that it has the most enhance Task-Focused UI implementation. This eases implementation, due to using the Mylyn Monitor API [Mylyn Integrator Reference], we can retrieve the user interaction data for their own analyses [16]. Now Mylyn is used in many aspects of the project: data gathering using Mylyn Monitor API, ITS access using Mylyn Connector API and interface focusing.

For implementation of message-driven architecture we use the integration framework Apache Camel[17]. Camel is an open cross-platform Java framework that allows performing application integration in a simple and understandable form and is able to work over multiple transport protocols. It is built on enterprise integration patterns [18]. Routes are described using declarative Java domain specific language.

We use JMS specification and its implementation Apache ActiveMQ to provide scalable and reliable messaging. As a database a pure Java XML database eXist is used because all event messages in system are transferred in XML form.

For testing purposes Mantis bug-tracker system is used but current implementation supports all ITS, which are supported by Mylyn. Currently Mylyn supports more than 50 ITS and allows programmers to implement other connectors [19]

5. Test-case and results analysis

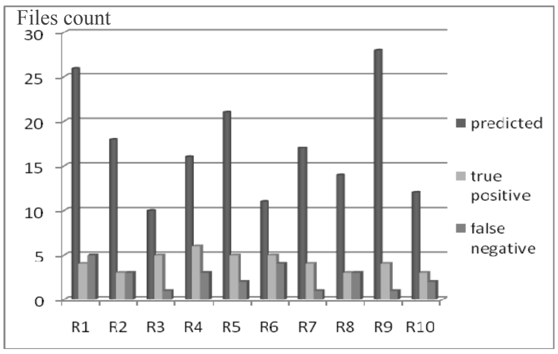
To validate our approach we have deployed the elaborated software and it was used several days by developers of real software project in e-learning domain. The goal was to gather retrospective data and analyze an applicability of proposed approach and metrics.

Measuring of traceability accuracy. To satisfy developers' needs, RT process must be accurate and tracing procedure should show only relevant files for current task. Algorithms efficiency measurement is done with precision and recall.

Precision *p* is the ratio of the positive prediction that was correct, while recall *r* is ratio of the positive cases that are retrieved. [20] This values could be calculated:

$$p = \frac{tp}{tp + fp} \;;\quad r = \frac{tp}{tp + fn} \;; \tag{6}$$

where *tp* – true positive, treated as interested and used;
fp – false positive, treated as interested but not used;
fn – false negative, treated as not interested but used.



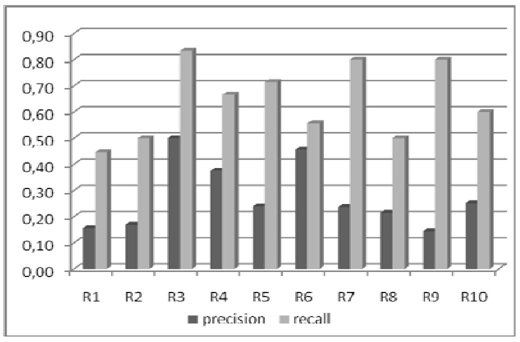


Fig. 5. a – predicted and used files for requirements; b – precision and recall for requirements

Here is shown some result data about predicted necessary documents and documents that were really needed while working on corresponding requirements. Average values for precision and recall is 0.43 and 0.61.

Our main hypothesis is that the proposed approach and CASE-tool improve requirements traceability process by increasing the programmer productivity. To validate this hypothesis we have adopted our data gathering and storage component to some kind of analysis framework that is able to record interaction history. It allows to 'play back' interaction to reproduce usage patterns and to gather statistics.

It is clear, that the system is useless without initial training on gathered data. Figure 6 shows precision and recall plot in time, dependent on count of processed activities (tasks and/or issues).

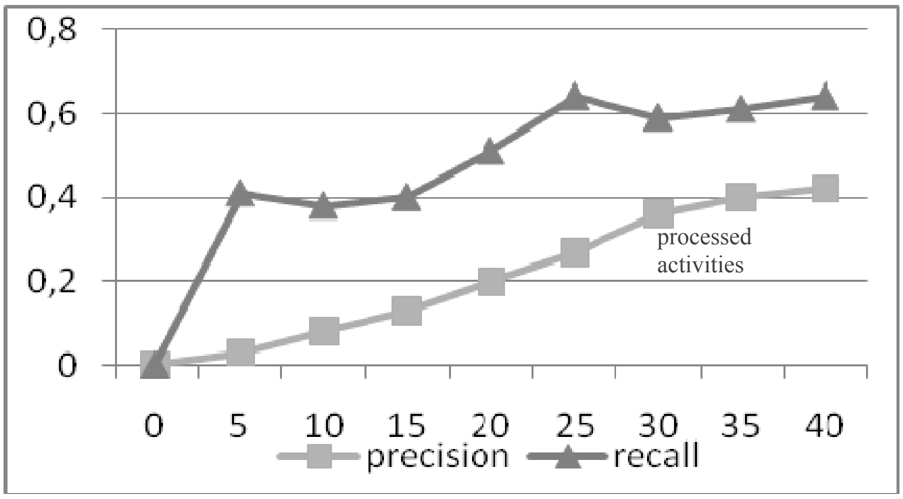


Fig. 6. Precision and recall ratio values

Quantitative measuring of efficiency improvement. We have made the experiment to provide quantitative analysis of the proposed approach effectiveness. We have used elaborated software in the real project and have gathered statistical data. As mentioned above, the edit ratio is the relative amount of edit versus selection events in any interaction history [11].

6. Conclusions and Future Work

In this paper we have analyzed the existing approaches of requirements traceability and we have developed data processing technology to improve requirements traceability by automating data gathering and have proposed an approach to build extended traceability matrix to support the task-focused developer interface.

Our approach is based on RM solutions integration with software development environments and plug-in support. Such integration and automation of requirements traceability process can significantly increase efficiency of software development, especially for Agile-centered projects development.

Our future work concerns final implementation of the proposed CASE-tool, as well as improving the algorithms for ATM calculation in order to make prediction more accurate. We also need to consider some issues connected with synchronization of the developer's activities with coordination server. Modeling part of our approach can be improved in the way to introduce structural and semantic relationships between requirements, artifacts and developers.

Acknowledgments

We have to thank students of the Department of Information Management Systems at the National Technical University "Kharkiv Polytechnic Institute" and "Interpak" company, who participated in testing of the ReqMIT tool.

1. *K. Beck*. (2001, Feb.) Agile manifesto homepage. [Online]. Available: http://agilemanifesto.org/
2. *S.D.Miller, R. DeCarlo, A. Mathur, and J. Cangussu,* "A control-theoretic approach to the management of software system test phase," Journal of Systems and Software; Special section on Software Cybernetics, vol. 11(77), p. 1486–1503, November 2006.
3. *Anderson D.J.* Agile Management for Software Engineering // Prentice Hall, 2003.
4. *Ramesh, B., & Jarke*, M. Toward Reference Models for Requirements Traceability. // IEEE Transactions on Software Engineering, 2001, 27(1), 58–93.
5. *Joern David, Maximilian Koegel, Helmut Naughton, Jonas Helming.* Traceability ReARMed // 33rd Annual IEEE International Computer Software and Applications Conference, 2009.
6. *Joao Paulo A. Almeida, Maria-Eugenia Iacob,* Pascal van Eck Requirements traceability in model-driven development: Applying model and transformation conformance // Published online: 3 August 2007 # Springer Science + Business Media, LLC 2007.
7. *Marco Lormans.* Managing Requirements Evolution using Reconstructed Traceability and Requirements Views // PhD thesis, IPA Dissertation Series 2009-03 Printed by Universal Press.
8. *Hong Xu, Pete Sawyer,* Ian Sommerville. Requirement process establishment and improvement from the viewpoint of cybernetics // The Journal of Systems and Software, issue 79 (2006), p. 1504–1513.
9. *Likoebe M. Maruping, Viswanath Venkatesh, Ritu Agarwal*. A Control Theory Perspective on Agile Methodology Use and Changing User Requirements // Information Systems Research Vol. 20, No. 3, September 2009, p. 377–399.
10. *Arbi Ghazarian*. A Matrix-Less Model for Tracing Software Requirements to Source Code // International Journal of Computers, Issue 3, Volume 2, 2008, p. 301-309.
11. *M. Kersten and G. C. Murphy*, "Mylar: a degree-of-interest model for ides," in Proceedings of the 4th international conference on Aspect-oriented software development, ser. AOSD'05. New York, NY, USA: ACM, 2005, p. 159–168.
12. *M. Kersten and G. C. Murphy*, "Using task context to im- prove programmer productivity," in Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering, ser. SIGSOFT '06/FSE-14. New York, NY, USA: ACM, 2006, p. 1–11.
13. *M. Tsuda, O. Ishikawa, S. Ohno, T. Harada, M. Takahashi, S. Kusomoto, and K. Inoue*, "Effectiveness of an integrated case tool for productivity and quality of software developments," IEICE TRANS. INF. & SYST, vol. E89D, no. 4, p. 1470–1479, 2006.
14. *P. Krill*. (2010, Jan.) Accurev offers agile alm suite. [Online]. Available: http://www.infoworld.com/d/developer-world/accurev-offers-agile-alm-suite-460
15. *J. Xie*. (2011, Jan.) Fisheye integrates jira issues to related code in your repository. [Online]. Available: http://www.atlassian.com/software/fisheye/tour/jira-issues-source-code.jsp
16. *YoungSeok Yoon, Brad A. Myers*. Capturing and Analyzing Low-Level Events from the Code Editor // Evaluation and Usability of Programming Languages and Tools (2011, Oct.)
17. *C. Ibsen*, Camel in Action - Greenwich: Manning Publications, 2010.
18. *G. Hohpe and B. Woolf*, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, 2004.
19. *Mylyn* extensions. [Online]. Available: http://wiki.eclipse.org/index.php/Mylyn_Extensions
20. *Dr. David L. Olson, Dr. Dursun Delen*. Advanced Data Mining Techniques Springer, 2008.