Computer Science and Information Systems 16(1):75–104 https://doi.org/10.2298/CSIS180226038J

Reconstructing De Facto Software Development Methods

Marko Janković[1], Slavko Žitnik[1], and Marko Bajec[1]

[1] Faculty of Computer and Information Science, University of Ljubljana,
Večna pot 113, 1000 Ljubljana, Slovenia
{marko.jankovic, slavko.zitnik, marko.bajec}@fri.uni-lj.si

Abstract. Software development is a complex process that requires disciplined engineering approaches. Empirical studies show that companies still don't document their development practice, or if they do, these are not up-to-date and do not reflect how they really develop software. The main objective of this paper is to propose an approach that can help companies in documenting their real development practice. Comparing to existing approaches that require substantial effort on the side of project members, our approach extracts information on development practice directly from software repositories. Five companies have been studied to identify information that can be retrieved from software repositories. Based on this, an approach to reconstruct development practice has been developed. The approach has been evaluated on a real software repository shared by an additional company. The results confirm that software repository information suffice for the reconstruction of various aspects of development process, i.e. disciplines, activities, roles, and artifacts.

Keywords: Software development method; Software repository; Development practice; Development method; Development project.

1. Introduction

Software development requires a systematic and disciplined approach to assure the quality of the process and its results, i.e. the software that we develop 1. This has been recognized already in the early beginnings of the software development era and has led to the construction of many software development methods.[1] Over the years, it then turned out that there is no ideal development method that could fit to all kinds of projects, even in the context of a single organization. How suitable a particular development method is, actually depends on many factors, ranging from project and organization characteristics to the characteristics of the development team. These findings have been identified by many researchers, e.g. [2–9].

[1] In this paper, we use the term software development method to denote the work that we do to structure, plan, control and perform the development of an information system. With the term method we cover all important aspects of the development lifecycle, i.e. activities to be carried out, artefacts to be developed, techniques to be used, roles to be assigned etc.

One of the research fields that emerged as a result of the aforementioned problems, is method engineering. Researchers in this field devoted a lot of effort to find suitable solutions. One of them is the so called situational method engineering, which is a process of constructing development methods specifically attuned to the needs of projects [10]. Such development methods would either be composed of fragments of other development methods or created by tailoring the development method that is generally used and known to the organization. Unfortunately, there are several obstacles that hinder the application of situational method engineering in practice [11, 12]. One is for instance that we need somebody who is capable of applying the method engineering process (must be familiar with various development methods, method fragments etc.) and what is even more challenging, we need enough time before starting the project so that this person can do the job. In real settings, where projects are almost always run in very tight schedules, this is rarely the case [13–15].

Problem statement. Based on our experience from introducing the situational method engineering process in practice 16 and from related research findings (e.g. 17), companies see as beneficial if they are able to document and monitor actual work on development projects and compare it with their prescribed methods. In this way, they can detect deviations if they occur. Doing this manually is however, very time-consuming and perceived by developers as an unnecessary burden. An approach is thus needed that does not require more than just a minimal effort from developers.

Objective. The objective of our research is to solve the above problem by reconstructing information about the project performance from the data that is captured in software repositories. We assume that software repositories contain enough data to reconstruct at least the main method elements (i.e. disciplines, activities, artifacts). Moreover, the objective is to support post-development analysis to learn how the project was performed and to possibly identify its positive and negative aspects in relation to its outcome, and also during the project performance, so as to detect situations that might lead to project failures.

Contribution. The main contribution of the research presented in this paper is the approach that facilitates the reconstruction of the development method elements from software repositories. In contrast to existing approaches that only enable the reconstruction of disciplines (e.g. analysis, design, development) or focus on the development phase only, our approach enables the reconstruction on a more detailed level, including additional development method elements.

Outline. The paper is organized in ten sections. In Section 2, we explain how the research was performed (research design), giving also a brief information on the participating companies. In Section 3 and 4, we describe the suggested approach with Section 3 focusing on the analysis of the software repository content and Section 4 on the reconstruction of the development method elements. Section 5 covers the evaluation and its results, which are discussed in Section 6. Section 7 provides the information on threats to validity. In Section 8, we position our research within related work and, finally, Section 9 concludes the paper.

2. Research Method

This section gives a brief description of the research design. Our research was motivated by the following research question:
Does the information stored in software repositories suffice for the reconstruction of basic characteristics of the software development methods that were de facto used in the corresponding software development projects?

2.1. Data Collection Procedure

Data were first collected from five software companies whose business is software development (see Table 1 for their profiles). The companies shared their software repositories with us (limited to selected projects only) and provided their personnel (project managers) for qualitative analysis. For the evaluation step, an additional company joined the research. Its data (software repository) and personnel were used to validate the research findings.

Table 1. Profiles of participating companies.

Company	Company profile
Marand d.o.o.	Company with around 100 employees that develop innovative and easy to use healthcare IT products.
Comtrade d.o.o.	Large company with over 500 employees. They develop IT solutions for different industries, including government, financial institutions, healthcare, telecommunication providers.
Ekipa 2 d.o.o.	Company with over 200 employees. They are focusing on development of entertaining mobile apps and games.
Optilab d.o.o.	Small company of about 30 employees. They develop complex information systems for clients from the financial sector, utilities and healthcare.
Adacta d.o.o.	Company, with over 350 professionals that provides support to 400 regional and international clients. They are specialized in developing and implementing business IT solutions and business consulting.

2.2. Research Approach

To answer the research question, the following approach was used:

- **Step1**: **analysis of software repository content**: the purpose of this step was to find out what kind of supporting tools the participating companies are using within software development and, more importantly, what kind of attributes they capture in software repositories. For further research, we assumed that attributes which we found in all repositories (from the involved companies), are generic and could be

thus found also in any other software repository (i.e. from any other software development company).

- **Step 2**: **development of the algorithms for automatic reconstruction of development method elements from software repositories**: the purpose of this step was to develop algorithms (and tools support) that will allow us to reconstruct the development method elements from software repositories. The objective was to reconstruct the development method elements that represent valuable information for project managers and other project team members. Using semi-structured interviews with project managers from the participating companies, we identified the main development method elements of their interest. For each of these elements we then developed algorithms for their reconstruction.
- **Step 3**: **evaluation**: the findings from the first step and the algorithms developed in the second step were evaluated by involving another company in the research. At first, we checked whether our assumption about generic attributes holds in their case and then employed our algorithms to reconstruct the selected development method elements from the repository on the recently finished project. Finally, we discussed the accuracy and usefulness of the reconstructed software development elements with their project manager.

In the rest of the paper, each of these steps is described in more detail.

3. Analysis of Software Repository Content

As a first step we asked companies to provide us access to their development environments or just give us snapshots of their software repositories. This was not easy to get due to the privacy and security issues, but eventually we got enough data to get a good picture on what they use and collect. As we expected, the companies were quite similar in terms of the type of tasks for which they were using computerized support (e.g. revision control, issue/bug tracking, document management, etc.) but differed to a certain extent in the actual tools they were using. Among the tools that we found, the most common were:

- Jira, Bugzilla or DevTrack for issue/bug tracking (ITS)
- Subversion or Git for revision control (RCS)
- Sharepoint or LogicalDOC as a document management system (DMS)

Additionally, some companies were using tools for other tasks, such as for managing code reviews (e.g. Cruicable, Reitveld) or for managing team collaboration (e.g. Slack, Confluence, Skype). But since these tasks did not have computerized support in all companies and in some cases data cannot be obtained due to the privacy issues, we did not analyze them further.

For each company and tool, we then examined what kind of data they actually store in their databases or logs. In Fig. 1, you can see the set of attributes that we were able to find in software repositories of all five participating companies.

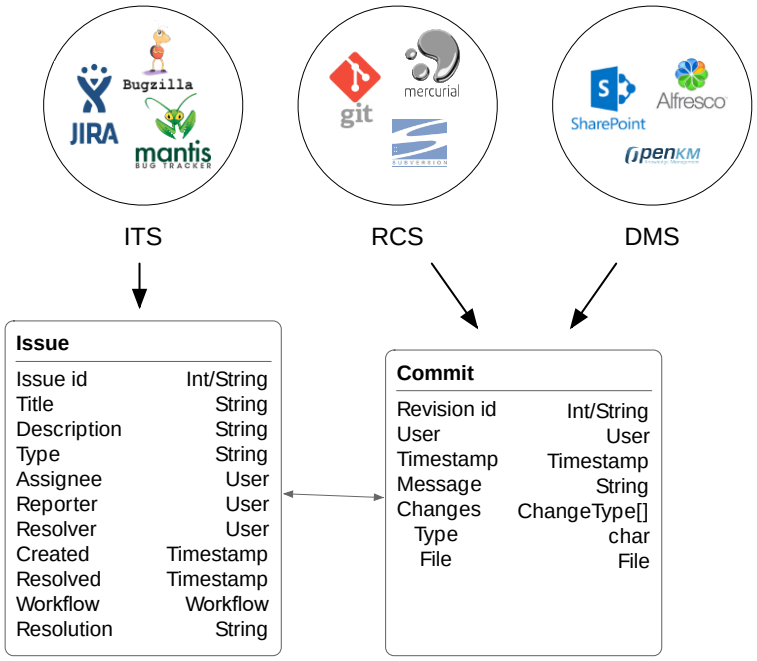

Fig. 1. Selected attributes from systems comprising the project's software repository.

The next step was to check how well the data from different tools comprising software repositories can be linked together. Remember that a software repository is not a standalone physical database but rather represents a logical view on several databases and logs from different systems. In our study, we found that important logical connections exist between issues and commits, which both carry important information and are kept track of in software development. While an issue represents a problem or associated tasks that need to be carried out in order to solve a specific problem, a commit refers to changes on specific files that are a result of solving the problem or task and are put back into the repository. Issues and commits are however managed in different systems and thus not necessarily linked. The link can be established if the commit message (Fig. 1) carries enough information so that we can identify which issue it is connected with. For unlinked commits and issues, techniques such as Frlink can be used 18. Let us also note that linking commits with issues is a good development practice that has been practiced in open source community for a long time 19 and should be enforced by the companies that want to raise the quality of their development processes.

Another challenge for establishing connections between data collected through various systems into a software repository, is to link user accounts created in these systems that refer to the same user. This is almost always the case, as software repositories usually comprise tools of different vendors and the single-sign-on option is not available. For this purpose, various existing entity resolution and identity merge algorithms can be used. In our case, we use the one published by Goeminne and Mens 20.

4. Algorithms for Automatic Reconstruction of Software Development Method Elements

As written in the introduction, our objective is to enable reconstruction of more detailed information about the development methods used on projects than existing approaches enable, and to do this without any substantial involvement of developers. The existing approaches [21-24] focus on the reconstruction of disciplines (i.e. they are able to tell how much effort was spent for analysis, design etc. or how these disciplines were following one another) or they go into details on the development phase only (i.e. by focusing on the issue lifecycle). In contrast, our goal is to focus on the whole project, and not just the development phase and to reconstruct more than just the disciplines. In this section, we first describe the meta model that was constructed in cooperation with the participating companies and then describe how this information can be reconstructed from the repositories. The steps for reconstructing the project specific software development method are shown in the Fig. 2. In the figure each step has a link to the section in which it is explained in more details.

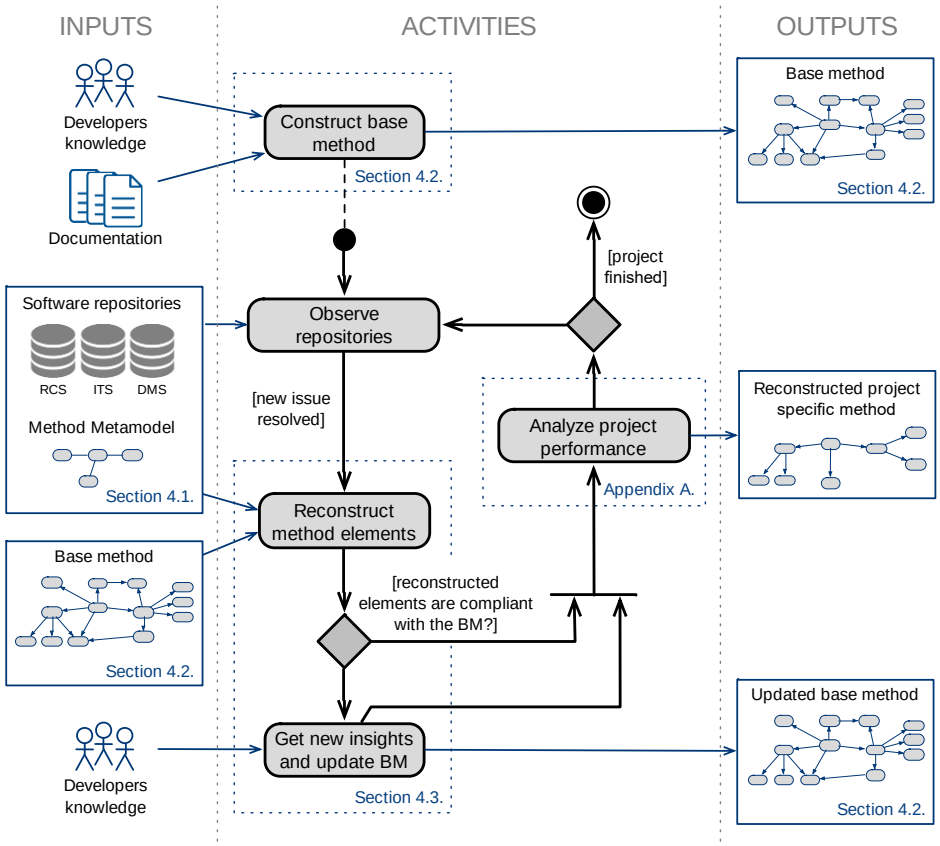

Fig. 2. Diagram showing the steps used during the reconstruction of the project specific software development method. BM is used as acronym for “base method”.

4.1. Metamodels of Software Development Methods

Development methods can be described by a number of different concepts, including phases, disciplines, deliverables, activities, techniques, examples, roles, experiences, life cycles, tools, etc. A number of different metamodels exist that underpin existing development methods 25. For the purpose of our research, we followed the SPEM metamodel 26. Its basics, including the separation of method and process concepts, were briefly described to the participating companies. Afterwards, the companies were asked to identify the main method meta elements that they would be interested in reconstructing from the software repositories. Linking data from software repositories to SPEM metaelements has also been done by others [27, 28]. The final selection, that was influenced also by the data that is actually captured in software repositories, included the following method meta elements:

Disciplines: a discipline presents a set of activities that are closely related in the sense that they all contribute to the same overall goal (e.g. Analysis, Design, Implementation).

Activities: an activity presents a general unit of work assignable to a specific performer (Develop a use case, Design GUI, etc.). As a result of an activity, different deliverables (artifacts) can be produced.

User roles: a user role is responsible for performing activities and producing the required artifacts (e.g. Developer, Analyst, Architect, etc.). Note that several project members can be assigned to a single user role.

Artifacts: an artifact is a result produced as part of performing a particular activity (e.g. Source code, Unit test).

The metamodel is represented in Fig. 3. Each *discipline* consists of one or many *activities* while an *activity* belongs to exactly one *discipline*. An *activity* can be connected with none, one or many *artifacts* and an *artifact* with one or many *activities*. For each *activity*, there is exactly one *user role* assigned, but for a particular *user role* there might be several *activities* that the *user role* is responsible for. The recursive relationship on the meta element *Activity* designates that *activities* are dependent on each other, which is due to the fact that they need to be performed in a certain order.

Fig. 3 also indicates the relationships of the metamodel elements with software repository concepts. The three most important concepts, originating from a software repository, are a *file*, an *issue*, and a *user*. The concept *file* designates a physical file that is stored in a revision control system or document management system, the concept *issue* represents an issue from an issue tracking system, and finally the concept *user* denotes user accounts from any of the software repository systems. The meaning of the relationships among metamodel elements and software repository concepts is as follows:

Artifacts are in relationship with *files* (stored in software repository) that represent the artifact. Each *artifact* can be related to none, one or several *files* while a file belongs to exactly one *artifact*.

Each *issue* requires a certain amount of work to be done. This work might be represented as an *activity*. A good development practice is that each *issue* is

connected to exactly one *activity* while *activities* might resolve several *issues* at once.

Users represent project team members with user accounts in a software repository. Several *users* can be assigned to a particular *user role* and vice versa, a particular *user* can play more than just a single *user role* in the observed project.

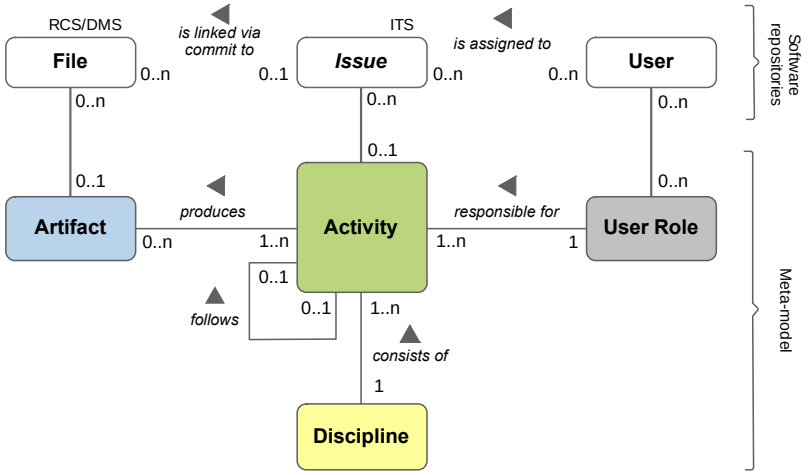

Fig. 3. Software development method metamodel and its connection with software repository concepts.

4.2. Construction of a Base Method

One of the concepts that plays an important role in our approach is the so called base method. With the base method, we denote the set of method elements that are typically used in a particular company when performing development projects. Taking into account the metamodel described in Section 4.1, the base method of a company includes its typical disciplines, activities, user roles, artifacts, and relations among them. The intensity of individual activities and the sequence of their performance is however not described with the base method as this depends on each particular project settings and its characteristics. In our approach, the base method represents the baseline to which we compare the reconstructed method elements and detect deviations.

The construction of a base method is a preliminary step before the reconstruction of the project specific development method. The simplest way to do this is by analyzing documentation of past projects or by acquiring this information from project managers. In some cases, companies even keep their software developments methods documented, in which case these documents can serve as a good starting point for the construction of the base method.

Fig. 4 represents an example of a base method constructed according to the existing documentation for one company that participated in the evaluation. Using the documentation, we were able to construct the company's base method consisting of 5

disciplines, 15 activities, 23 artifacts, 8 user roles, and the relations among them. During the construction we have also captured 14 rules, which define the presence of a relation between method elements based on the project characteristics. (e.g. Financial Calculation - If the project is small then Financial Calculation is not required; If the project is medium or large then Financial Calculation is required). More details about the construction are presented in the Section 5.3.

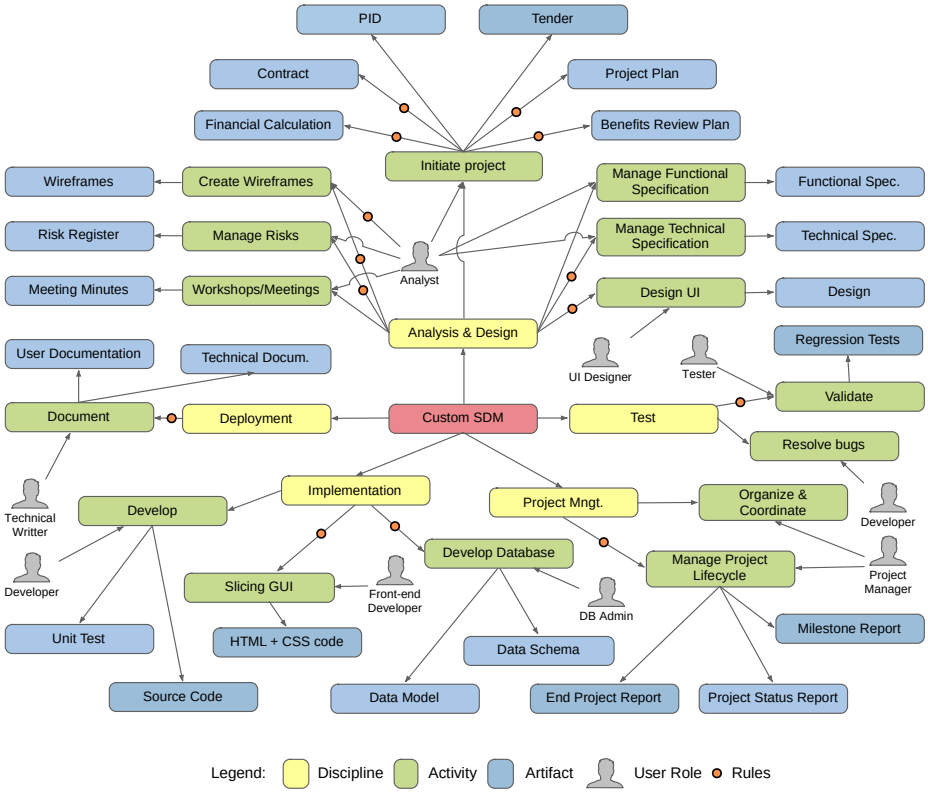

Fig. 4. The company's base method draft manually created out of the company's documentation. It comprises 5 disciplines, 15 activities, 23 artifacts, 8 user roles, and 14 rules. Examples of a rule: (a) Financial Calculation - If the project is small then Financial Calculation is not required; If the project is medium or large then Financial Calculation is required, or (b) If the project is small then Benefits Review Plan is not required; If the project is medium or large then Benefits Review Plan is optional.

4.3. Reconstruction of the Software Development Method Elements

Once the base method is defined, we use data from software repositories to reconstruct activities, artifacts, user roles, and disciplines. In this section we describe how this is

done. In the guidelines that follow, we take into account the findings on the data that software repositories include and how well this data is linked (see Section 3).

Reconstructing Artifacts

Artifacts are reconstructed from files that were committed to the repository. This is done immediately after an issue is being resolved. We check the repository and retrieve all files that were committed as a consequence of resolving this issue. Then we infer from the files (by checking their names, file types and if necessary also the file content) which artifacts from the base method they represent. Techniques that can be used for matching files to artifacts are many. Machine learning algorithms are useful when we have data to learn from, i.e. software repositories from past projects. In this case, we create a classifier which we then use for matching. If this is not available, we acquire additional information from the company employees so that matching can be done. The algorithm used in such cases is described at the end of this section.

Reconstructing Activities

After we reconstruct the artifacts as a result of resolving an issue, we go further and check which activities are connected to these particular artifacts. We do this by checking the base method where these relationships are defined. In most cases, activities are connected to one artifact only, thus the reconstruction of the correct activity is not a problem. The involvement of developers is only required in rare cases, a) when these relationships are not one to one (an activity produces several artifacts and vice versa, several activities might be responsible for the creation of one particular artifact) or b) when we have no artifact which we could use to infer the corresponding activity. This happens when we deal with an issue that did not commit any file to the repository. In such cases, we involve the developers to tell which activity is connected to that particular issue.

User Roles and Disciplines

Similarly, we also reconstruct user roles and disciplines. Since we already know the activity name, we simply check the base method to retrieve also the names of the associated user roles and disciplines. Furthermore, for each retrieved user role, we also retrieve the users (user accounts) that were assigned to this particular activity.

Algorithm for Reconstructing Artifacts and Activities

For matching files to artifacts and issues to activities we use the algorithm that is described below.

For the algorithm to work, the first step is to go through the base method and capture keywords that best describe each artifact. In addition, we capture file types that represent the format in which a specific artifact is usually created. Next, we capture keywords for the activities without artifacts.

Example: Keywords and file types for the artifact "Functional specification"

```
Keywords := {requirement, functional requirement,
non-functional requirement, usability, scalability…}
File types := {doc, docx, rtf, txt, pdf};
```

Such manual acquisition of this information is only required if we do not have any data to learn from. If this data is available, i.e. we have access to software repositories of finished projects, then techniques, such as Bag of Words, TF-IDF or similar can be used to automatically acquire this information.

The algorithm to match files to artifacts and issues to activities is as follows: for each resolved issue *I*, we find all connected commits *C*. For each such commit *C*, we classify each committed file *F* to an artifact *A* from the base method *BM*. We try to do that based on the file types. If several artifacts (artifact list *AL*) contain the same file type, we calculate individual artifacts weights. An artifact weight *w* for an artifact *A* tells what is the likelihood that the file *F* represents the artifact *A* from the artifact list *AL*. The higher the weight, the higher the likelihood. The weight *w* is calculated as a sum of TF-IDF values. The TF-IDF metric ((tfidf=tf(K,A)*idf(K,AL))) is calculated as a product between frequency of the keyword *K* in the file *F* ($tf(K,F)=f_{K,F}$) and the logarithm of the ratio between the number of all artifacts *A* in the artifact list *AL* and the number of these artifacts from the artifact list in which the keyword *K* appears ($idf(K,AL)=\log(|AL|/|K \in AL|)$). Once the committed files have been successfully classified to an artifact, we check in the base method which is the activity that is responsible for the delivery of this artifact.

If the issue under analysis cannot be connected to any commit, and thus we cannot identify the corresponding activity over connected artifacts, then we reconstruct the activity directly from the base method by employing a very similar approach (see lines 23-33 in Algorithm 1). Instead of searching for artifact keywords in committed files, we search for activity keywords in the issue title and description. These are two attributes that we can find in all issue tracking systems. For clarity reasons (to avoid duplicate lines), this part is not shown in the algorithm.

The algorithm is represented below. It uses three data structures:

Matrix Weight_KA: a two-dimensional matrix that tells for each keyword *K* and artifact *A* what is the likelihood that *K* represents *A*. The likelihood of *K* representing *A* is calculated using TF-IDF.

List Artifacts: a list of artifacts from the *BM* that contain a specific file type in their file types set.

List Activities: a list of activities from the *BM* that are in *BM* linked to the artifacts reconstructed during the classification of files linked to a specific issue.

Algorithm 1: Algorithm to reconstruct Artifacts and Activities

```
1  INPUT:
2  List ResolvedIssues; //ordered by resolved time ASC
3  BaseMethod BM;
4
```

```
5  OUTPUT:
6  // files are classified to artifacts
7  // issue are classified to activities
8  // relations between activities and artifacts
9
10 Matrix Weight_KA;
11 Set IssueArtifacts;
12 for each issue I in ResolvedIssues do
13   for each commit C in I.connectedCommits do
14     for each file F in C.files do
15       if F.type == IgnoreFileType then next file;
16       List Artifacts = artifactsByFileType(F.type, BM)
17         If Artifacts.size == 0 then
18           // new type -> ask project member, update BM
19         else if Artifacts.size == 1 then
20           Classify(F, Artifacts[0]);
21           IssueArtifacts.add(Artifacts[0]);
22         else
23           resetAndPopulateMatrix(Weight_KA, Artifacts);
24           for each artifact A in Artifacts do
25             for each keyword K in A.keywords do
26               Weight_KA[K,A]:= tf(K,F)*idf(K,Artifacts)
27
28           Artifact, Value = max(sumByA(Weight_KA))
29           if Value != 0 then
30             Classify(F, Artifact);
31             IssueArtifacts.add(Artifact);
32           else
33             // ask project member and update BM
34
35   if IssueArtifacts.size > 0 then
36     List Activities = findActByArtifacts(IssueArtifacts, BM);
37       if Activities.size == 1 then
38         Classify(I, Activities[0])
39       else
40         // ask proj. member (bad practice, new knowledge)
41   else
42         /*Issue without artifacts. Same as in lines 23-33,
43       but this time we use keywords from issue title and
44          description and as a list all Issues without
45       artifacts*/
```

As a part of the research described in this paper, we developed a computerized tool (iSPRToolset) that facilitates the application of the proposed approach in a company. The tool supports the following tasks:

a) The analysis and linkage of the data from tools that comprise software repositories
b) The creation of a base method
c) The automatic reconstruction of activities, artifacts, user roles, and disciplines using the algorithm described in Section 4.3.4
d) Various visualizations of the development method elements of the timeline

For more details about the tool check http://ispr.jmlabs.eu and the Appendix.

5. Evaluation

To make the evaluation unbiased, we invited an additional software company to join the project. In this way, we did not know what tools this company is using to facilitate development, neither what information it stores in its software repository. The evaluation comprised the following steps:

(1) Analysis of the company's software repository
(2) Construction of the company's base method draft
(3) Reconstruction of the development method elements
(4) Analysis of the project performance

The aim of the first step was to find out whether our assumptions about a) typical attributes that could be found in a software repository and b) linkage between repository data (issues, commits, user accounts) hold for this particular company.

In the second step, the goal was to create a draft of the company's base method.

The step three was dedicated to the evaluation to what extent specific development method meta elements can be reconstructed from the company's software.

Finally, the purpose of the fourth step was to evaluate how useful the reconstruction approach can be if used for controlling the project performance.

In the following, we report on the evaluation findings.

5.1. Profile of the Company

The company that we analyzed in the evaluation, develops e-business solutions for Health, Insurance and Telco industries and employs about 50 people. They develop software by following a combination of agile and traditional approaches. The decision to involve this particular company into our research was based on the following reasons:

- The company was willing to provide all the necessary information about the project that seemed appropriate for the evaluation.
- The project team members were allowed and willing to commit required time for the purpose of the evaluation.

- The company already had a prescribed and documented development method in place – i.e. guidelines for software development. This was useful as we could use it as starting point for the construction of the company's base method.

5.2. Analysis of the Software Repository Content

For the selected project, we imported data from three different tools that the company was using during the project. These were Jira (used as issue tracking system), SVN (used as revision control system), and LogicalDoc[2] (used as document management system). The first step was to retrieve issues, commits, and users (user accounts). For a summary report on the data collected see Table 2.

Table 2. Collected data.

Attribute	Value
# of issues retrieved (ITS)	186 – 13[a]
# of commits retrieved (RCS + DMS)	379 + 166
# of all files	3578
# of users (employees + stakeholders)	15 + 2

[a]In case of Jira, 13 issues were excluded as they were duplicates of other issues, could not be resolved, or were of the following type *meta task*.

On the next step, we tried to link commits from SVN and LogicalDoc to Jira Issues. At first, we did that by extracting issue IDs from commit messages using regular expressions, such as for example `"\b"+JiraProjectKey+"\b){1}[^\w[0-9]]+\d+"`. A similar approach was also used by others [29, 30]. In this way, we were able to link roughly 70% of all commits with corresponding issues and about 60% of issues with corresponding commits. These results alone were already promising, as we linked majority of the commits with issues and vice versa. To improve these results, we could have used the approach as suggested in 18, but we rather decided for a manual check via developers so that we also learned how consistent they are in using supporting tools.[3] Together with their help, we were able to link additional 139 commits and 54 issues. 34 (6.2%) commits and 18 (10.4%) issues were left unlinked.

By analyzing unlinked commits, we found 9 of them were made to restructure and move the repository to a new location. Additional 12 were related to specific changes, such as *upgrade library*, *add user as developer in pom.xml*, *fix typo*, *import files*, etc. The last 13 unlinked commits that were left were all found to be connected with project management activities and the creation/modification of various related documents.

Similarly, by analyzing unlinked issues we found that they mainly presented system administration activities, such as *increase RAM in test environment*, *update from java 6*

[2] In LogicalDoc, a commit is perceived as a new version of a file (check-in).

[3] It is important to note here that companies should require from their developers to link commits with issues otherwise important information is missing. The open-source community seems to be aware of that – in MongoDB and Hibernate, for example, over 90% of all commits from 2014 are linked to at least one issue from Jira 19.

to java 7, *install SSL certificate*, as well as activities that did not result in the creation of any artifact (e.g. *setup development environment*).

Regarding the resolution of users via user accounts, we had no problems, since the users were using the same usernames for all the systems. On this project, 17 different people participated, out of which 15 were employees and 2 stakeholders. For details on the results of linking the data see Table 3.

Table 3. Percentage of issues linked to commits and vice versa.

Attribute	Value
% of commits linked to issues (regex)	68.3
% of issues linked to commits (regex)	59.5
% of commits linked to issues (regex + manually)	93.8
% of issues linked to commits (regex + manually)	89.6

At the end of this step, we also checked how well the development method meta model corresponds to the company and its expectations from the development method reconstruction. The company's CIO was fine with the selected development method meta elements.

5.3. Construction of the Base Method Draft

When we asked the company to tell us how they usually develop software (i.e. do they have any predefined steps, deliverables, techniques, user roles etc. that project team members need to follow) they gave us a documentation in which they defined basics of their development method. This included the description of project disciplines and corresponding activities. For each activity, the documentation also provided a description of the activity goals and associated artifacts. Each activity was further linked with user roles responsible for its performance. The described development method also differentiated among different types of projects. Based on the project size, these were divided into three groups: small, medium and large. All this information was written in a series of word files and available to all employees. We constructed the base method by analyzing the provided documentation. We did that together with one of the company's project managers. The base method draft is depicted in Fig. 4. In the next sections, we describe how this base method served us to reconstruct the development method elements that were used on the observed project.

5.4. Reconstruction of the Project Software Development Method Elements

The most important part of the evaluation was to check how well can we reconstruct development method elements of an observed project by analyzing the data from the corresponding software repository.

To have a "golden rule", i.e. to be able to measure how accurate is the reconstruction, we asked the person that acted as the manager of the observed project, to help us manually reconstruct the development method elements that were used on the project.

For each issue, the manager identified connected commits and their files and based on that concluded what artifacts they represent.[4] In case an issue was found that had no connected commits (this happens when during the resolution of an issue no files are created), the project manager was asked to tell what activity this issue was about. Similarly, for the commits and related files that were not identified over issues, the project manager was asked to classify them into the artifacts they represent. From activities, we then inferred disciplines and user roles.

In the next step, we used this information as the baseline against which we compared the results obtained with our algorithm. To measure the quality of the reconstruction, we used the precision and recall measures, which are known from information retrieval and pattern recognition. The results are shown in tables below. To fairly judge the quality of the algorithm, we also compared manually and automatically classified file versions into artifacts and issues into activities. The reason for this is that some artifacts are created gradually, through many versions, and are thus connected to many issues. Consequently, it wouldn't be enough to limit the comparison on the artifacts only, as these might reconstruct well only for some of the commits.

Table 4. Precision and recall of the automatically classified file versions to artifacts and issues to activities.

	Manually classified	Automatically classified	Precision	Recall	F-measure
File version	5945	5908	0.997	0.991	0.994
Issues	173	155	0.98	0.88	0.93

Table 5. Precision and recall of the automatically reconstructed development method elements compared to the manually retrieved development method elements.

Method element	Manually retrieved	Automatically retrieved	Precision	Recall	F-measure
Artifact	15	12	1.00	0.8	0.89
Activity	12	10	1.00	0.83	0.91
Discipline	5	5	1.00	1.00	1.00
User role	8	7	1.00	0.88	0.94

To achieve good results with our reconstruction algorithm, it is crucial that artifacts are reconstructed with as high precision and recall as possible, as the reconstruction of other method elements depend on this.

As you can see from the results, the classification of file versions to artifacts yielded very good results (Table 4, row 1). The reason for this is that artifacts are quite different in terms of their names, content and formats in which they are created. Thus, we were able to differentiate among them with a high confidence. The results also show that file versions did not influence much on the classification accuracy. This is an important

[4] This was not that time consuming, as the company uses a special directory structure in revision control system and document management system to store files that belong to a certain result or artifact. This way the project manager could conclude already from the place in the directory structure what artifacts an observed commit's files most probably represents.

finding as it supports iterative reconstruction of development method elements, i.e. step by step through the project performance.

The files that were misclassified or were left unclassified, were not many and in most cases due to one of the following reasons: (a) the file was of an unknown type, i.e. not defined in the base method – in our case these were mainly fonts of file type woff, eot, tff…, (b) the file content couldn't be parsed – these were pdf files that contained images/scans and at the same time couldn't be classified based on keywords in their filenames and file paths, and (c) files that included keywords which are more typical for some other artifact – in our case this happened for files that represented meeting minutes (e.g. on one particular meeting they were discussing a lot on the functional specification, so this word occurred many times in the meeting minutes, and thus the file was classified as an artifact "functional specification" rather than "meeting minutes").

Good results in terms of precision and recall were obtained also for the classification of issues to activities (Table 4, row 2). In most cases, we were able to correctly identify activities that corresponded to issues by classifying files these issues created.

Table 5 shows results of reconstructing the development method elements. It compares manually retrieved development method elements with the automatic reconstruction. As you can see, all the automatic reconstructions were correct (100% precision) which is not surprising as the classification of files got such a high accuracy. The recall for activities and artifacts were however not that perfect on the first sight (0.8, 0.83, respectively). Several activities and artifacts were missing in the automatic reconstruction. The explanation that we got from the project members revealed that the missing elements were all newly introduced and thus couldn't be found in the base method. Let us emphasize however that these results are based on the fully automatic reconstruction, i.e. without any involvement of the development team. In other words, the results, presented in tables above, could be improved if the developers were asked for additional information in cases when classification or reconstruction couldn't be done.

5.5. Checking Project Performance

The algorithm for reconstructing development method elements can be used also during project performance. In this case, we reconstruct development method elements one by one, every time an issue is resolved, and check for their compliancy with the base method. There are several benefits of doing this during project performance. If the reconstructed development method elements are not compliant with the base method, the project manager is notified about that and can react by asking responsible project team members for clarification. In case no suitable argumentation is given, a bad practice was obviously detected and can also be prevented. However, if those responsible for the deviation can argument why they declined from what was expected, the base method can be supplemented by capturing new knowledge in terms of new development method artifacts or rules that bind project characteristics with some specific development method element. Additionally, if the project is being checked during its execution by reconstructing development method elements after each resolved issue, the information about activities and disciplines can be visualized on a timeline diagram which gives an interesting insight into the current state of the project – this is only possible if company

store information about the time planned and spent on the level of issues. It can even help to detect situations that might represent risk for the project. For example, if the majority of issues that are being resolved are still connected to activities and disciplines that should already be finished then it could be that we are at risk that the project will be late. Some of these analyses, produced with the iSPRToolset for this particular project are shown in the Appendix.

For the purpose of the evaluation, we simulated the project realization by passing the issues to our approach, as they were appearing chronologically (ordered ascending by the resolved date-time attribute) during the analyzed project. For each issue we have reconstructed the development method elements. The Fig. 5 shows the reconstructed development method. What is worth to mention is that we improved the base method by three new development method elements (two artifacts and one user role), which were detected during the reconstruction and present a new knowledge about development practice. This happened when the algorithm was not able to classify a file into any of existing artifacts or an issue into any of existing activities and we thus asked the project manager for an explanation. He explained that these development method elements are important but we obviously failed to capture them when we were creating the base method draft. Final assessment given by the project manager was that he would like to have our approach and iSPRToolset implemented and available for future projects he will be working on.

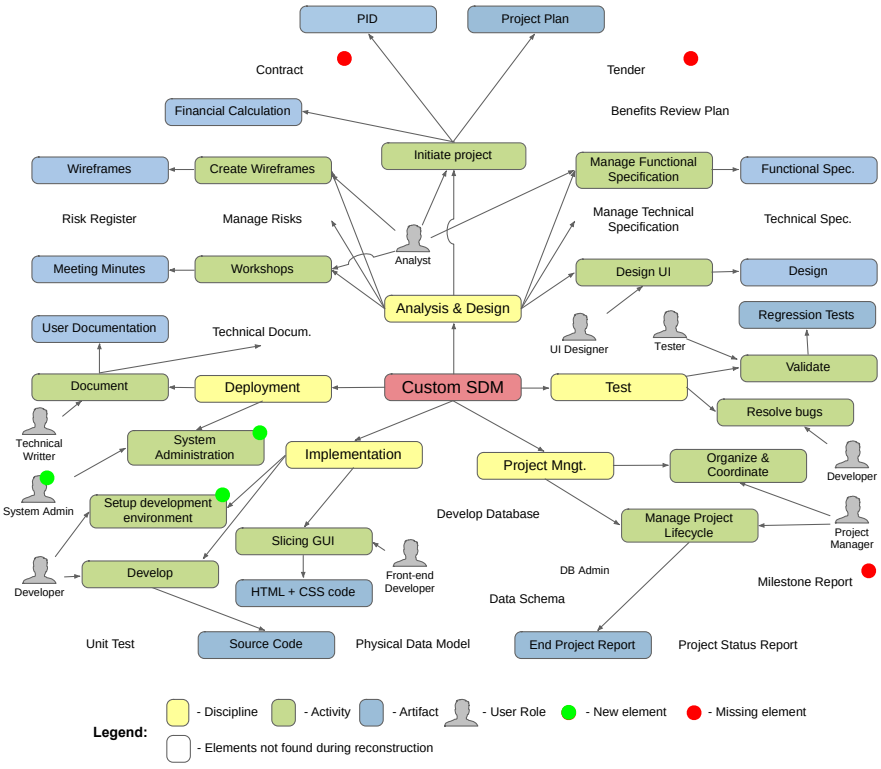

Fig. 5. Reconstructed software development method.

6. Discussion

The approach presented in this paper has some limitations that we need to acknowledge. First, it does not reconstruct all possible development method elements but only some of them, i.e. disciplines, activities, artifacts, and user roles, to be exact. It also does not reconstruct the workflow which would tell how exactly the activities followed one another. Furthermore, it depends on the quality of the data captured in software repositories. Finally, it requires some effort (although not substantial) from the project team members in order to work optimally.

On the other hand, to our knowledge, it achieves much more than existing approaches, which we are aware of. First, it helps the company in capturing and maintaining its base method, i.e. the method that really reflects how the company is developing software. Next, it helps the company to conduct development projects in a way that they are performed consistently and in line with what the company prescribes with the base method. Also, it helps to detect deviations from the base method as project is performed. Finally, it only requires small input from the development team.

To further validate the usability of the approach, semi-formal interviews were conducted with seven project managers of the participating companies. We asked them the following questions:

a) How do you perceive the suggested approach in terms of its complexity? Could it be introduced in your company? Do you think it would be accepted by your employees?
b) What do you expect the main benefits would be of using such approach in your organization?
c) What would you suggest to make the approach more useful?

The feedback that we received was generally positive. They all agreed the approach is simple enough to be adopted in their organizations. Since it doesn't require any substantial effort from developers or changes that developers would need to introduced in their everyday practice, the acceptance of the approach is also expected to be high.

As the main benefit, they emphasized the following possibilities offered by the approach: a) to do retrospective on finished projects, b) to observe project performance on the fly and identify steps that does not fit their regular practice (i.e. decline from their base method), c) to keep base method up-to-date, and d) to give more emphasis on methodological aspects of their development activities (as a side effect). Finally, as a suggestion for improvement they were all consistent that it would be very useful if we were able to reconstruct also workflow information, i.e. how exactly activities and their smaller counterparts (tasks) were performed during an observed project.

7. Threats to Validity

There are multiple threats to validity that face this research, we will address them in the context of construct validity, internal validity, external validity and reliability.

With respect to construct validity, we had to address the fact that we rely upon data that are created and annotated by project members and are stored in software repositories. To improve construct validity, we have validated the data and results with project members and constructed the reconstruction approach based on the insights that we got from five companies. The threat we face here is also that project members might not remember all the details from the project and there is no explicit evidence that the project has been performed as it was reconstructed. We presume that data in software repositories are unbiased and that project members possess enough information about how project has been conducted.

From an internal validity point of view, we do not face any threats, since our main goal was to show that using data from software repositories, we can reconstruct a development process and method followed on the project. In our evaluation, we analyzed an already completed project, hence the data should not be biased.

An external validity issue we face is that we evaluated our approach only on one case study, hence it is hard to justify how generalizable our results are. However, the approach to reconstruct method elements and perform different analyses is straightforward: if all the required data are available, it is reasonable to assume that reconstruction can also be done on other development projects. Among different organizations, the main difference, when using the proposed approach, is in the base method, which is specific to the organization and should be defined based on the company's development practice. Another threat we face is that the data are not of such a good quality as required. For example, commits and issues might not be linked to that extend as required. In our case study we were able to link 93.8% of all commits with an issue. However, this might not be achievable on other projects, since it is up to the development culture and rules inside a particular organization.

In terms of reliability the accuracy of the annotated data can be a concern as it can produce biased results. In case of the reconstruction this would give spurious results, but reconstruction would still be successful. So this threat is more related to the accuracy of the reconstructed method. To mitigate this threat, the reconstructed method was validated with project members.

8. Related Work

There are several works that can be considered related to our research. These can be grouped into four categories, according to the research fields they come from:

- Method engineering
- Software repository mining
- Software process discovery
- Software process mining
- Other related approaches

Note that there is some overlapping among these fields (specifically among the last three fields), in terms of approaches and techniques that they use. Different names that

they carry are more a result of the fact that they come from different research groups and times.

8.1. Method Engineering

Method engineering is an approach to create software development methods that are specifically attuned to organizations. In general, the idea lies in the conceptualization, and construction of new methods and tools (or in the adaptation of existing ones), so that they best fit requirements of a certain organization. The research on method engineering has a long tradition. A good introduction to the field can be found in 31. Based on the method engineering principles, a specific direction has emerged, called situational method engineering. As the name implies, situational method engineering deals with developing new methods or adapting existing ones on-the fly, i.e. to meet specific project situations. In the literature, a number of situational method engineering approaches were suggested [16, 32, 33]. For an excellent review see 10.

Method engineering and specifically situational method engineering works are related to our research in general, as they share the same motivation, i.e. to help software development companies develop software in more disciplined way. In both cases, development methods are the subject of research with a difference that situational method engineering approaches require much more human effort to properly work. As reported in 14 and 16, this is considered one of the main reasons why situational method engineering approaches hardly penetrate to practice.

8.2. Software Repository Mining

The analysis of software repository data is a research discipline that deals with the analysis of rich software repository information to get valuable insights about the development process and software itself. For a survey, see 34. The works that are directly related to our research, are for instance 35, 36, or 37. Here, the authors employ various statistical models, such as Latent Dirichlet Allocation and Latent Semantic Indexing to cluster unstructured and unlabeled textual data (commit log comments, source code, documentation, mailing lists, etc.) into topics. Although results show that this can be done efficiently, the topics do not convey much information on the underlying development method, except maybe the main activities, if they can be inferred from the topics.

In this group, we also include works that deal with software process recovery. Existing approaches mainly apply supervised and unsupervised techniques, such as bag-of-words, summary statistics, topic analysis, and Bayesian classifiers to recover the development process from a variety of artifacts that were created by developers and can be obtained from software repositories [21, 22]. These approaches allow for the recovery of the so called Unified Process Views, which illustrate how the relative emphasis on different disciplines changes over the course of the project. Detailed information on activities, user roles, and artifacts are out of scope of these works.

8.3. Software Process Discovery

In the field of software process discovery 38, their main objective is to automatically derive a formal model of a process from the data that was collected during the execution of a process. Several approaches have been suggested on how this can be done, for example [39–41]. What these works have in common, is that they only use information from revision control systems and not also from other systems that usually comprise a software repository.

8.4. Software Process Mining

In the field of software process mining, the authors linked data from different software repositories and apply process mining techniques to derive a process map and identify inefficiencies, imperfections, and enhance existing process capabilities [23, 42]. They have used data from software repositories to perform different control and organizational analyses. However, as part of their analyses, they only focus on the processes on the level of code review or bug life cycle. The same goes for other work in this field that employ process mining techniques to recover valuable information [24, 43, 44]. Here, the authors focus on the reconstruction of software processes on the level of disciplines or on the level of issues, but do not consider activities, user roles and artifacts, as we do in our research.

8.5. Other related approaches

There are also other research areas related to our research such as Organizational patterns, which also can be used to capture software development methods [45, 46, 47]. Mainly organizational patterns are still captured and documented manually, but some of the researchers are trying to use data from software repositories to detect bad practices (anti-patterns) [27, 28]. These approaches could benefit from our research since they could analyze the behavior and patterns on the higher level, level of activities.

9. Conclusion and Future Work

Software development is a complex and creative task, whose sophisticated results are increasingly influencing our daily lives in various ways. Due to the nature of this work, it is important that each company has a method in place to manage, control, and guide the work of software developers and project managers. Otherwise, confusion may ensue, leading to project failures, low quality of the developed software and higher maintenance costs.

To manage software projects, companies often use different supporting tools, such as issue tracking systems, revision control systems, document management systems, code review tools, and others. Their main goal is to support the work of developers. Each tool

per se contains a lot of information and valuable knowledge on how a project has been performed in practice. However, to obtain an even better overview of the development process as a whole, the information from these tools can be linked. Linked data can then be used to reconstruct what really happens behind those projects and eventually to learn, among others, why some projects go well and others do not.

In this paper, we described how the data from different software repositories (issue tracking system, revision control system, document management system) can be used to reconstruct valuable information on the project performance with only a little involvement of the developers. The aim of the paper was to demonstrate that using and linking the data from tools comprising software repositories, allows us to reconstruct the development method in more details than existing approaches do. Furthermore, the aim was to show that it is possible to capture the actual ways of working in an organization, in a form of a base method, which can be constantly kept up-to-date without any significant involvement of developers.

To identify the information that can be retrieved from software repositories we have cooperated with five companies, which shared their data with us. Based on the findings we have developed an approach to reconstruct development practice. We have evaluated the approach on a real software repository shared by an additional company. The results show that software repository information suffice for the reconstruction of various aspects of development process, i.e. disciplines, activities, roles, and artifacts.

As part of our future work we plan to gather data from other software repositories and include it into the process of reconstruction. With this we expect to rise the reconstruction accuracy and level of details reconstructed. We also plan to use the approach on other software projects to see how it performs in real-time manner (monitor, control, guide).

Acknowledgments. We thank Professor Martin Pinzger for his comments on the draft of this paper. We also wish to acknowledge financial support for this project by the Slovenian Research Agency ARRS within the research program P2-0359.

References

1. David P. and Parnas L.: Risks of Undisciplined Development. Communications of the ACM, Vol. 53, No. 10, 25-27. (2010) doi: 10.1145/1831407.1831419.
2. Hardy C. J., Thompson J. B. and Edwards H. M.: The use, limitations and customization of structured systems development methods in the United Kingdom. Information and Software Technology, Vol. 37, Issue. 9, 467-477. (1995) doi: 10.1016/0950-5849(95)97291-F.
3. Fitzgerald B.: An empirical investigation into the adoption of systems development methodologies. Information & Management, Vol. 34, 317–328. (1998) doi: 10.1016/S0378-7206(98)00072-X.
4. Huisman M. and Iivari J.: The individual deployment of systems development methodologies. Lecture Notes in Computer Science, 134–150. (2002).
5. Hansson C., Dittrich Y., Gustafsson B. and Zarnak S.: How agile are industrial software development practices? Journal of Systems and Software, Vol. 79, Issue 9, 1295-1311. (2006) doi: 10.1016/j.jss.2005.12.020.

6. Gonzalez-Perez C. and Henderson-Sellers B.: A work product pool approach to methodology specification and enactment. Journal of Systems and Software, Vol. 81, Issue 8, 1288-1305. (2008) doi: 10.1016/j.jss.2007.10.001.
7. Petersen K. and Wohlin C.: A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of Systems and Software, Vol. 82, Issue 9, 1479-1490. (2009) doi: 10.1016/j.jss.2009.03.036.
8. Clarke P. and O'Connor R. V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology, Vol. 54, Issue 5, 433-447. (2012) doi: 10.1016/j.infsof.2011.12.003.
9. Ivarsson M., Gorschek T.: Practice selection framework. Int. J. Soft. Eng. Knowl. Eng., Vol. 22, No. 01, 17-58. (2012) doi: 10.1142/S0218194012500027
10. Henderson-Sellers B., Ralyté J., Ågerfalk P. J. and Rossi M.: Situational Method Engineering. Springer. (2014) doi: 10.1007/978-3-642-41467-1.
11. Kuhrmann M., Méndez Fernández D., and Tiessler M.: A mapping study on the feasibility of method engineering. J. Softw. Evol. and Proc., 1053–1073. (2014) doi: 10.1002/smr.1642
12. Ter Hofstede A. H. M., Verhoef T. F.: On the feasibility of situational method engineering. Information Systems, Vol. 22, Issue 6, 401-422. (1997) doi: 10.1016/S0306-4379(97)00024-0.
13. Fitzgerald B.: The use of systems development methodologies in practice: a field study. Information Systems journal, 201-212. (1997) doi: 0.1046/j.1365-2575.1997.d01-18.x.
14. Fitzgerald B., Russo N. L. and O'Kane T.: Software development method tailoring at Motorola. Communications of the ACM, Vol. 46, Issue 4, 65-70. (2003) doi: 10.1145/641205.641206.
15. Coleman G.: An Empirical Study of Software Process in Practice. In Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 315c-315c. (2005) doi: 10.1109/HICSS.2005.86.
16. Bajec M., Vavpotič D. and Krisper M.: Practice-driven approach for creating project-specific software development methods. Information and Software Technology, Vol. 49, Issue 4, 345-365. (2007) doi: 10.1016/j.infsof.2006.05.007
17. Gupta M., Sureka A., Padmanabhuni S. and Asadullah A. M.: Identifying Software Process Management Challenges: Survey of Practitioners in a Large Global IT Company. In 12th IEEE Working Conference on Mining Software Repositories, 346-356. (2015).
18. Yan Sun, Qing Wang and Ye Yang: FRLink: Improving the recovery of missing issue-commit links by revisiting file relevance. Information and Software Technology, Vol. 84, 33-47. (2017) doi: 10.1016/j.infsof.2016.11.010
19. Jankovic M. and Bajec M.: Comparison of software repositories for their usability in software process reconstruction. In proceedings of IEEE 9th International Conference on Research Challenges in Information Science, 298–308. (2015) doi: 10.1109/RCIS.2015.7128890
20. Goeminne M. and Mens T.: A Comparison of Identity Merge Algorithms for Software Repositories. Sci. Comput. Program, Vol. 78, 971-986. (2013) doi: 10.1016/j.scico.2011.11.004
21. Hindle A.: Software process recovery: Recovering process from artifacts. In 17th Working Conference on Reverse Engineering, 305-308. (2010) doi: 10.1109/WCRE.2010.46
22. Hindle A., Godfrey M. and Holt R.: Software process recovery using recovered unified process views. In IEEE International Conference on Software Maintenance, 1-10. (2010) doi: 10.1109/ICSM.2010.5609670
23. Gupta M. and Sureka A.: Mining Bug Report History for Discovering Process Maps, Inefficiencies and Inconsistencies. In Proceedings of the 7th India Software Engineering Conference, 1-10. (2014) doi: 10.1145/2590748.2590749

24. Poncin W., Serebrenik A. and Van den Brand M.: Process mining software repositories. In 15th European Conference on Software Maintenance and Reengineering, 5-14. (2011) doi: 10.1109/CSMR.2011.5
25. Hug C., Front A., Rieu D., and Henderson-Sellers B.: A method to build information systems engineering process metamodels. Journal of Systems and Software, Vol. 82, No. 10, 1730-1742. (2009)
26. OMG, Software & Systems Process Engineering Metamodel Specification (SPEM), Tech. rep. (Apr. 2008). URL http://www.omg.org/spec/SPEM/2.0
27. Pícha P., Brada P., Ramsauer R., and Mauerer W.: Towards Architect's Activity Detection Through a Common Model for Project Pattern Analysis. In Proceedings of 2017 IEEE International Conference on Software Architecture Workshops. (2017).
28. Pícha P. and Brada P.: ALM Tool Data Usage in Software Process Metamodeling. In Proceedings of 2016 42th Euromicro Conference on Software Engineering and Advanced Applications. (2016).
29. Fischer M., Pinzger M. and Gall H.: Populating a Release History Database from version control and bug tracking systems. In proceedings of the International Conference on Software Maintenance, 23-32. (2003) doi: 10.1109/ICSM.2003.1235403
30. Sliwerski J., Zimmermann T. and Zeller A.: When Do Changes Induce Fixes? In proceedings of the International Workshop on Mining Software Repositories, 1-5. (2005) doi: 10.1145/1082983.1083147
31. Brinkkemper S., Lyytinen K. and Welke R. J.: Method engineering: principles of method construction and tool support. In Selected Papers from the International Conference on ''Principles of Method Construction and Tool Support''. (1996) doi: 10.1007/978-0-387-35080-6
32. Ralyté J., Deneckère R. and Rolland C.: Towards a generic model for situational method engineering. In Proceedings of the 15th Conference on Advanced Information Systems Engineering, 95-110. (2003)
33. Karlsson F. and Ågerfalk P. J.: MC Sandbox: Devising a tool for method-user-centered method configuration. Information and Software Technology, Vol. 54, Issue 5, 501-516. (2012) doi: 10.1016/j.infsof.2011.12.009
34. Woosung J., Eunjoo L. and Chisu W.: A Survey on Mining Software Repositories. IEICE Transactions on Information and Systems, 1384-1406. (2012)
35. Savage T., Dit B., Gethers M. and Poshyvanyk D., TopicXP: exploring topics in source code using latent dirichlet allocation, In IEEE International Conference on Software Maintenance, 1–6, 2010. doi: 10.1109/ICSM.2010.5609654.
36. Chen T. H., Thomas S. W., Nagappan M. and Hassan A. E., Explaining software defects using topic models, In 9th IEEE Working Conference on Mining Software Repositories, 189–198, 2012. doi: 10.1109/MSR.2012.6224280.
37. Hindle A., N. Ernst A., Godfrey M. W. and Mylopoulos J.: Automated topic naming. Empirical Software Engineering, Vol. 18, Issue 6, 1125-1155. (2013) doi: 10.1007/s10664-012-9209-9
38. Cook J. E. and Wolf A.: Automating process discovery through event-data analysis. In 17th International Conference on Software Engineering, 73-73. (1995) doi: 10.1145/225014.225021
39. Kindler E., Rubin V. and Schäfer W.: Incremental workflow mining based on document versioning information. Unifying the Software Process Spectrum, no. 3840 in Lecture Notes in Computer Science, 287-301. (2006)
40. Akman B. and Demirors O.: Applicability of process discovery algorithms for software organizations. In 35th Euromicro Conference on Software Engineering and Advanced Applications, 195-202. (2009) doi: 10.1109/SEAA.2009.87

41. Duan B. and Shen B.: Software process discovery using link analysis. In IEEE 3rd International Conference on Communication Software and Networks, 60-63. (2011) doi: 10.1109/ICCSN.2011.6014218
42. Gupta M.: Process Mining Software Repositories to Identify Inefficiencies, Imperfections, and Enhance Existing Process Capabilities. In Companion Proceedings of the 36th International Conference on Software Engineering, 658-661. (2014) doi: 10.1145/2591062.2591080
43. Rubin V., Gunther C. W., Van Der Aalst W. M. P., Kindler E., Van Dongen B. F. and Schäfer W.: Process mining framework for software processes. In proceedings of the international conference on Software process, 169-181. (2007)
44. Lemos A., Sabino C., Lima R. and Oliveira C.: Using process mining in software development process management: A case study. In proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1181-1186. (2011) doi: 10.1109/ICSMC.2011.6083858
45. Coplien O. J. and Harrison B. N.: Organizational Patterns of Agile Software Development, Prentice Hall. (2004)
46. Sulaiman Khail W. and Vranić V.: Treating Pattern Sublanguages as Patterns with an Application to Organizational Patterns. In Proceedings of 22nd European Conference on Pattern Languages of Programs. (2017)
47. Frťala T. and Vranić V.: Animating Organizational Patterns. In Proceedings of 8th International Workshop on Cooperative and Human Aspects of Software Engineering. (2015)

Appendix A. Project Performance Analyses

In this appendix, we provide examples of visualizations that have been created during the evaluation phase based on the information created with the iSPRToolset (http://ispr.jmlabs.eu). The aim is to show that there are other insights on the project performance that we get by following the proposed approach and might be beneficial for project managers.

The reconstructed method elements in combination with other information available from the software repositories (e.g. worklogs – each user logs hours spent working on a specific issue) provide a basis for different project analyses. All figures in this section are created using the data provided by the participating company.

Using the information about the worklogs allows us to observe on a daily basis how much time (effort) was spent per discipline, activity, or user role. This tells us, for instance, for which part of the project the most time was spent. Furthermore, we also have a possibility to analyze the total time spent for a particular discipline, activity, or user role. The daily intensity of particular discipline/activity/user role is presented on the right-hand side of Figures 6, 7, and 8.

Before acquiring a project, companies typically prepare a tender for which they also estimate the time that will be needed for a particular activity on the project in order to estimate the total costs of the project. The comparison of the actual and estimated time allows us to detect for which disciplines/activities/user roles developers spent more time than expected. This information is very important to project managers and can help them to make better estimations on new projects. In our case, the observed company, in order to prepare a tender and to estimate the project costs, does an internal financial calculation as part of which they also estimate the time needed for a particular task on the project. All this information is documented in an Excel file, which is classified to the Financial Calculation artefact from the base method. We used this information to gather

information about the estimated time for each activity and consequently also for each discipline. The comparison of the estimated and actual time is presented on the left-hand side of Figures 6, 7, and 8. It shows what portion of time was planned for a particular discipline/activity/user role and what portion was actually spent. With this comparison, project manager can identify project activities (and roles) that required more time than was expected.

It is a rule in the observed company that at the beginning of each project they also prepare a project plan, which includes the timeline of a project - often presented with a gantt chart. From the gantt chart we gathered the information about the timespan of a particular task and when it was planned to be resolved. We used this information to visualize how the expected timeline deviated from the actual one. The information from the gantt chart is integrated into the actual timeline of a project and is presented with light blue rectangles in Figures 6 and 8.

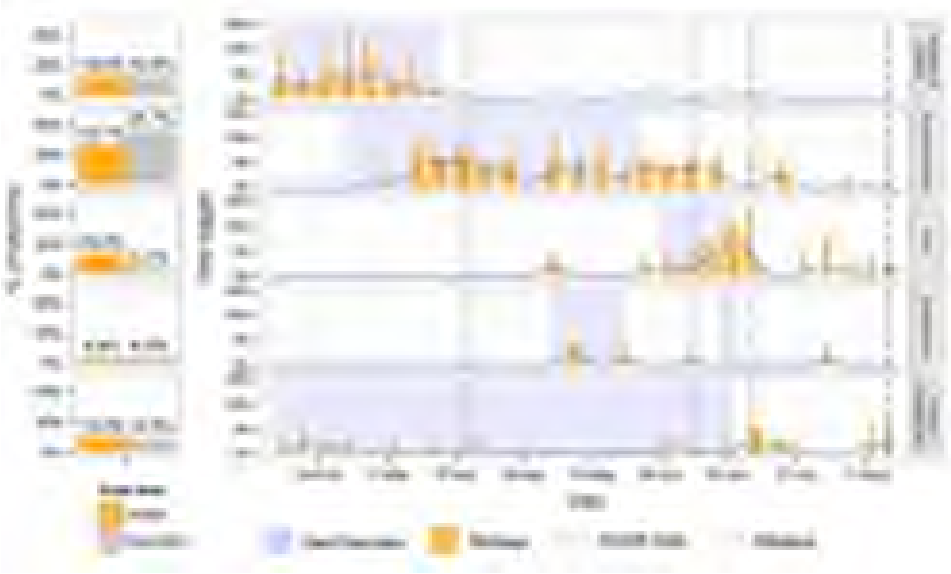

Fig. 6. Intensity of particular disciplines.

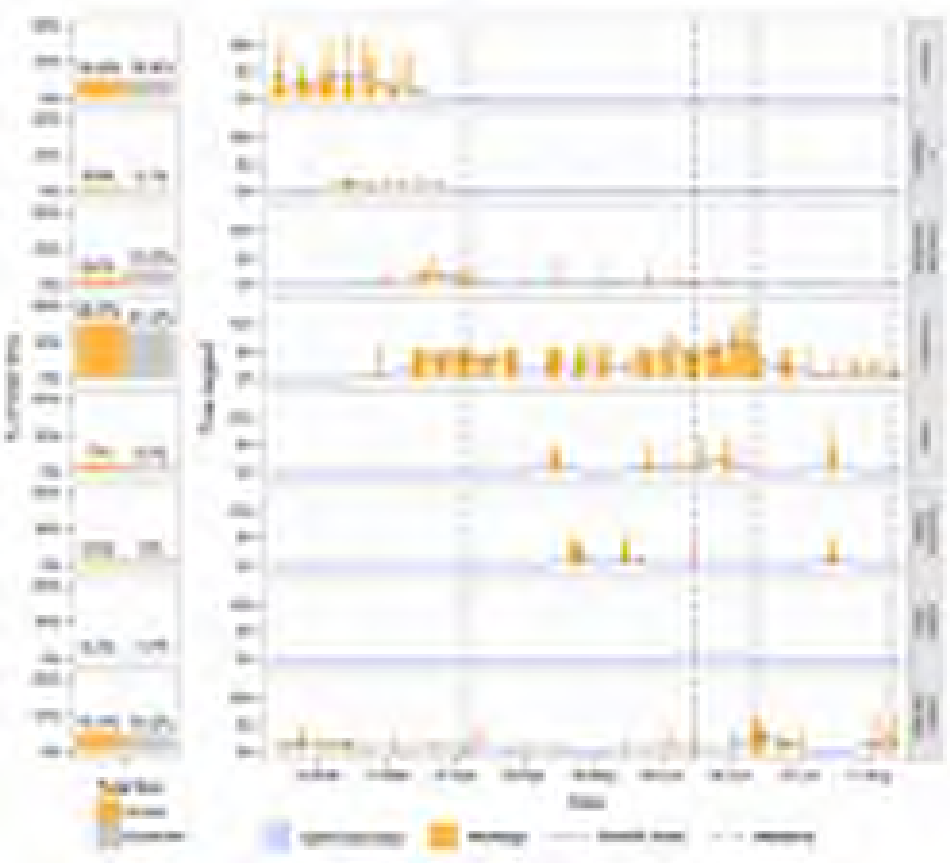

Fig. 7. Analysis on the level of user roles.

Fig. 8. Analysis on the level of activities

Marko Janković is a reseacher at the Faculty of Computer and Information Science, University of Ljubljana. He has been researching mainly in the field of mining software repositories and software development methods. His work has been published in journals and presented on several international conferences. Professionally he is also working on projects related to the Internet of things, big data, and data analysis.

Slavko Žitnik, PhD, is an assistant at the University of Ljubljana. His bibliography counts more than 20 items with a book chapter and a patent application. He has been researching mainly in the field of information retrieval and information extraction from textual sources. He designed an end-to-end information extraction system that uses an ontology and represents the extracted entities, relationships and coreferences against it. In the field of natural language processing he achieved first place at the BioNLP 2013 challenge in extracting relationships from text to build gene regulation network. Professionally he is also working on projects related to the Internet of things, big data, social networks and semantic data analysis.

Marko Bajec is a Full Professor and head of the Laboratory for Data Technologies at the Faculty of Computer & Information Science, University of Ljubljana. His research interests primarily focus on IT and data Governance include software development methods, IT/IS strategy planning, data management and manipulation. His work has been published in many high-ranked journals. In his career, he has led or coordinated over 30 applied and research projects. For his contribution in transferring knowledge to industry he received several awards and recognitions.

Received: February 26, 2018; Accepted: October 11, 2018