# Mining processes, resource consumption and witnesses for task completion from logs

## Deliverable D3.2

**FFG – IKT der Zukunft**
**SHAPE Project**
**2014 – 845638**

**Table 1** Document Information

| | |
|---|---|
| Project acronym: | SHAPE |
| Project full title: | Safety-critical Human- & dAta-centric Process management in Engineering projects |
| Work package: | 3 |
| Document number: | 3.2 |
| Document title: | Mining processes, resource consumption and witnesses for task completion from logs |
| Version: | 1 |
| Delivery date: | 01 October 2015 (M2) |
| Actual publication date: | ———— |
| Dissemination level: | Public |
| Nature: | Report |
| Editor(s) / lead beneficiary: | WU Vienna |
| Author(s): | Saimir Bala, Cristina Cabanillas, Jan Mendling, Axel Polleres |
| Reviewer(s): | Claudio Di Ciccio, Tudor Ionescu |

## Contents

# 1 Introduction

The Deliverable 3.2 of the SHAPE project[1] reports work performed under Task 3.2 *Gather resource information about available resources from enterprise repositories* of Work Package 3 *Automatic Discovery of Processes and Resources and Constraints from Unstructured Data*. Its main aim is to further progress on mining document repositories by identifying the next steps that must be done towards a novel process discovery technique that makes use of unstructured and structured data. Concomitant to this deliverable is the milestone **(M2)** *Mining processes, resource consumption and witnesses for task completion from logs*.

The previous deliverable D3.1 Cabanillas et al. [2015] provided an overview on text mining and process mining methodologies from a general perspective as well as a mining approach for a specific type of processes, namely, *project-oriented processes*, also known as *project mining*. This deliverable delves into the applicability of such methods on real project data and studies potential extensions to increase their application scope. In particular, the deliverable explains how:

- The project mining technique introduced in D3.1 has been implemented and integrated into the Business Process Management Systems (BPMSs) Camunda[2].
- The project mining approach can be improved by annotating the information obtained from the Version Control Systems (VCSs) logs according to several criteria in order to increase the precision of the description of the activities mined. For example, text mining techniques can provide insights from the messages attached to SVN commits.
- The process mining technique described in D3.1 can be extended to take into account not only process activities performed by one single human resource but also collaborative activities performed by a team.

The rest of the deliverable is organized as follows. Section 2 describes the implementation of project mining as a library that can be run in the Camunda BPMS engine. Section 3 gives details on the future directions for enriching and improving our project mining technique. Section 4 outlines our idea to mine team compositions from event logs for collaborative work in processes. Finally, Section 5 closes this deliverable and points out the next steps to be carried out.

[1] https://ai.wu.ac.at/shape-project/
[2] http://camunda.org/

# 2 Implementation of a project mining technique in Camunda

In D3.1 Cabanillas et al. [2015] we introduced a project mining technique that works as follows.

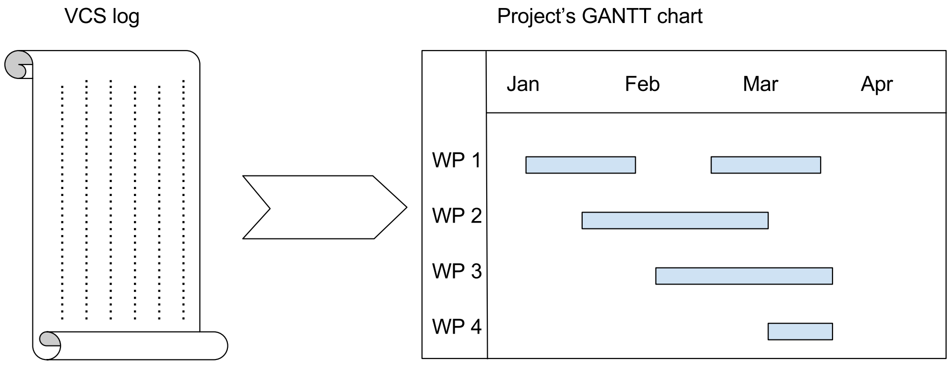
- First, it takes as an input a VCS log from the projects repository.
- Then it extracts the events from the commits. At the same time, it builds project's tree structure from the paths that are present in the log.
- The next step associates the events to each of the objects in the tree structure.
- Afterwards the events are aggregated into activities by using a threshold based technique described in Baier et al. [2014]. Furthermore, activities starting time is estimated by taking into account the commit frequency of the contained events
- The obtained activities are further aggregated or collapsed to respond users' need for a coarser or finer grained view on the work, respectively.
- As a final step, a coverage indicator is computed that shows how efficiently the time available for a task is used by actual work.

Figure 1 clarifies the above description by showing in *a)* Figure 1a, the input and the output of project mining; and *b)* Figure 1b, the approach to obtain a Gantt chart from a VCS log, as a five-step process.

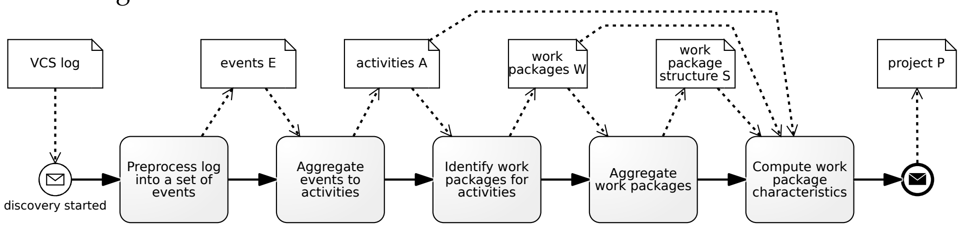
The approach has been implemented in Camunda, an open-source, extensible BPMSs. The software has been integrated into the architecture that is presented in Deliverable D6.1 Ionescu [2015], which provides further details about the general problem and a conceptual solution. Specifically, D6.1 presents an architecture based on three types of processes as depicted in Figure 2:

1. The first process is the main business process. Users execute activities as long as the process runs and generate artifacts. Artifacts are stored in a VCS.
2. The second process is the adaptation process. This process is able to perform changes in the main process to adapt to the evolution of the organization.
3. The third process is the mining process. As long as the main process runs, the mining process analyzes the artifacts that are being produced and eventually comes out with new insights on the running main business process. When a new insight is ready and shows a potential or existing risk, the adaption process takes over to correct the behavior of the main process.

As said, the mining process makes use of external data, such as the VCS logs and possible additional sources of data. These sources are denoted as KB in the




**(a)** Visualizing work as a Gantt chart




**(b)** Steps for mining project-oriented business processes from the VCS log

**Figure 1** Illustration of the project mining approach

picture and they may describe rules, regulations or behavioral patterns that the process must comply with. At the current state, the process mining task runs the project mining algorithm introduced in D3.1. In this architecture, the project mining technique helps to get useful insights on project-oriented business processes. To this end, it analyzes the artifacts that are produced during the lifetime of the project. The procedure has two main steps: *i) Learn* - runs an instance of the miner, and *ii) Inform* - communicates a new insight on the process to the adaptivity process.

Camunda is able to run services that are endpoints in a server. However, this has the disadvantage that we cannot make use of the visualization part which was implemented as a graphical user interface (GUI) component in Java. Such component allowed for user interaction with the elements of the directory (see Cabanillas et al. [2015]).

To achieve our goal of running the entire project mining approach in Camunda, we have separated the application logic part from the visualization part. Regarding the latter, we had the choose whether to adopt a representation of a Gantt chart that is exportable in a Web exchange format, or to make our component a library that can be called inside Camunda by a direct invocation. The first option would allow an external web application to interpret the encoded information and draw a Gantt chart accordingly. However, we opted for the second option because

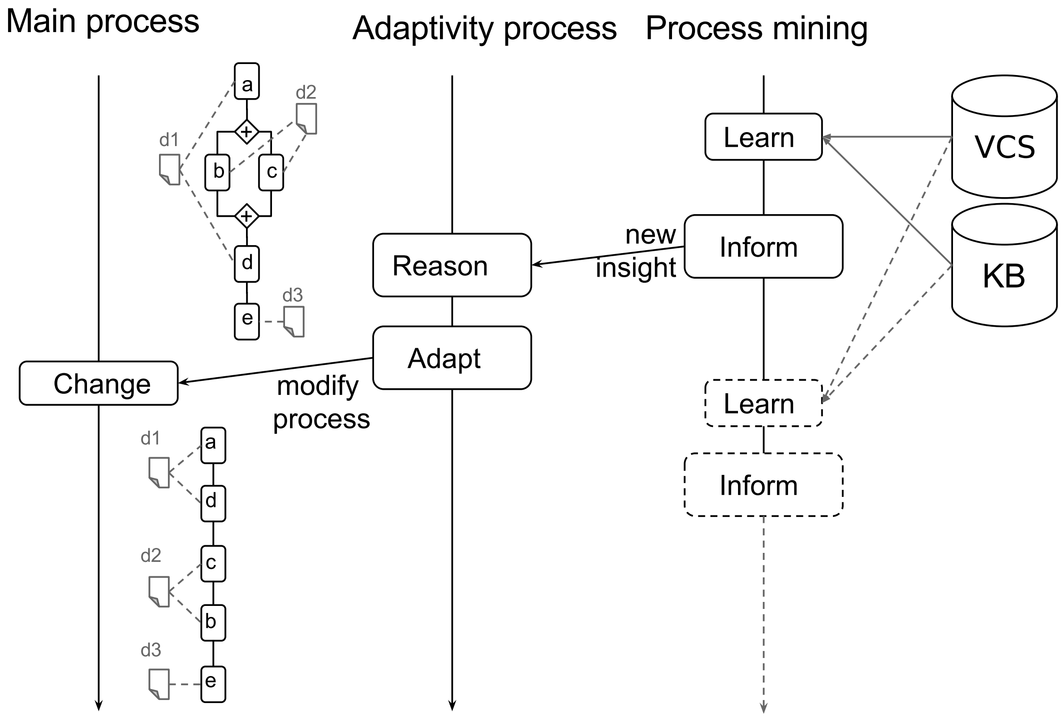



**Figure 2** Three process architecture from Deliverable D6.1

it does not require external services, thus allowing for a more integrated software environment. The choice was also guided by the design of the architecture shown in Figure 2 and discussed in more detail in deliverable Ionescu [2015]. It makes use of internal processes rather than calling them as external services. Therefore, we wrapped our project mining technique into a service. Nonetheless, the first option is still achievable with little modification.

As described above, the project mining technique takes as input a VCS log and a project-dependent threshold, and returns a representation of the Gantt chart. In order to still be able to exploit the aggregation functionalities available with the GUI-based implementation (see Figure 3), we compute all the sub-events aggregations for each node of the project tree and use an *isAggregated* flag to indicate whether a node is expanded or collapsed by the user. All the information is stored as a payload on each tree node. This makes it possible to avoid recomputing on user's end. Granularity scaling can be done by hiding the events and showing only the activities of a user-chosen node when a user decides to aggregate on the node, and vice-versa when the user decides to collapse the subtree of the node.

A node representation in Java is an object with the attributes shown in Listing 1.

```
private String value;
private boolean isAggregated;
```

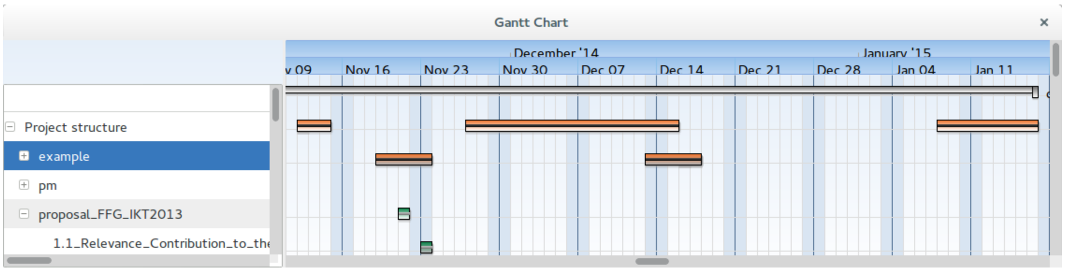



**Figure 3** The Gantt chart visualization from the GUI-based version of the project mining technique. This picture shows a part of the project tree structure on the left part and the corresponding activities or events on the right. The nodes `example` and `pm` are collapsed and therefore activities (*aggregation* step) are shown for them. On the contrary, the node `proposal_FFG_IKT2013` is expanded and fine granularity events are shown for it and its children.

```
    private Activity activity;
    private List<Event> eventList;
    private Node parent;
    private List<Node> childList;
```

**Listing 1** Java object that models an event

An event $e$ is implemented as a Java object whose attributes are shown in Listing 2.

```
    String id;
    DateTime start;
    DateTime end;
    String fileID; //where did it happen
    String type; // whether it's a Modify, Delete or Add
    String author;
    String commitID;
    String comment;
```

**Listing 2** Java object that models an event

```
[[e1 e2 e3] [e4 e5] [e7 e8 e9]]
```

**Listing 3** An example of a project structure representation through a tree with events

Listing 3 illustrates the data structure that has been chosen for an *activity* object, specifically, a list of lists. The one in the listing is composed of 9 events that are grouped into 3 different chunks. This representation allows to express which events are aggregated from the aggregation step of the project mining algorithm. At the same time, such representation is easily exportable in formats like

JavaScript Object Notation (JSON) or XML[3].

The program outputs a Gantt chart as a Plain Old Java Object (POJO). A POJO has interesting properties such as *a)* it can be persisted into a database (e.g. using Hibernate[4] or Java persistence API[5]) and *b)* is also translatable into other formats, such as JSON[6]. In this case, we output a POJO that makes use only of object and list entities .

```
+-[0] MiningSVN
 +-[1] lib
   +-[2] license4j-1.4.0.jar  A D
 +-[1] src
   +-[2] db
    +-[3] jdbc
      +-[4] TestMySqlConnection.java  A M D
      +-[4] ConnectionFactory.java  A M
   +-[2] gui
    +-[3] nebula
      +-[4] TreeConnectorExample.java  A D
      +-[4] examles
        +-[5] GanttSectionExample.java  A
+-[0] README.md  A M M M
```

**Listing 4** An example of a project structure representation through a tree with events

Listing 4 shows a part of the output of the program with its own Git change log (`MiningSVN.log`) as input file. This is a simplified printout of the real output tree, and shows only the type of the events that have occurred for each node, leaving out the details on events and activities. The letters `A`, `D` and `M` stand, respectively, for `Added`, `Deleted` and `Modified`. For instance, the file `README.md` has been added once and modified three times.

This version of our technique can also be run as a standalone component from the command line. An example call that takes as input the file `MiningSVN.log` and uses a threshold of 7 days to aggregate is encoded as shown in Listing 5.

```
   java -cp project-mining.jar main.ProjectMiner -f MiningSVN.
   log -t 7
```

**Listing 5** Command line call to the project mining algorithm

[3] http://www.w3.org/XML/
[4] http://hibernate.org/
[5] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[6] http://www.json.org/

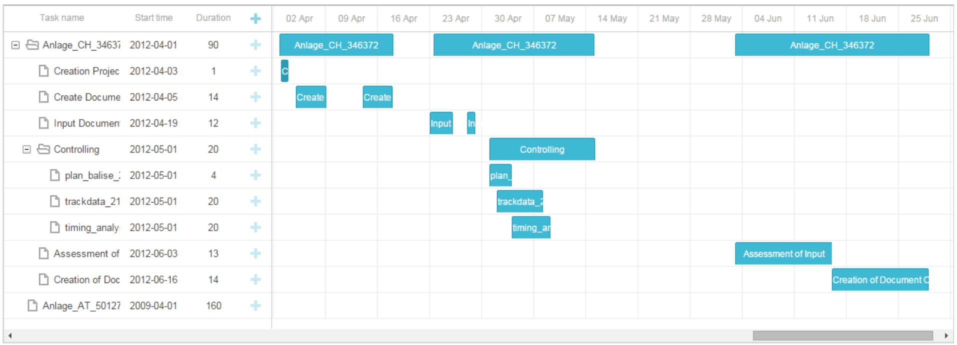



**Figure 4** Visualization of the Gantt chart in a browser

The library was included in the context of the deliverable D6.1 Ionescu [2015]. Figure 4 shows a Javascript implementation that uses the dhtmlxGantt[7] library to display the results of the project miner in a web browser.

The tool is open source and can be tried under the link https://github.com/s41m1r/MiningCVS/tree/application-logic/MiningSVN.

# 3 Project mining with annotations

This section outlines further developments on the approach for mining project-oriented business processes presented in D3.1 (Cabanillas et al. [2015]) and published in Bala et al. [2015]. As already outlined as future work in the previous deliverable, we aim at improving our mining technique in order to fully exploit the data that we can obtain from the software repository logs. We have identified 7 directions to get further insights on the process being discovered, which serve for annotating and classifying the work into activities in a more precise way. The following subsections describe each of the 7 points we are currently focusing on for current and future development.

## 3.1 Activity labels

Following up on mining project repositories, the first thing we need to investigate is how to identify business process activities from the data available in Subversion (SVN). The comments entered by users in the commits can help towards this purpose, but still, there is the need to understand to which business process activity a text refers. Difficulties grow when considering real-world cases where often users do not follow any given pattern when it comes to documenting their work

[7] http://dhtmlx.com/docs/products/dhtmlxGantt/

progress. Even though guidelines are available such as the ones in Mendling et al. [2010], companies often use their own naming conventions. Therefore, a first challenge that we would need to face in SHAPE is the *activity identification* from the natural language text of the comments. For this purpose, entity-recognition algorithms (Etzioni et al. [2005],Ritter et al. [2011]) can be firstly used in order to extract only the interesting part of the sentence. Afterwards, the extracted sentences can be refined into verb-object labels by applying automatic refactoring from Leopold et al. [2010].

The work of Mendling et al. [2015] raises the attention for the semantic aspects of text labels and natural language descriptions. The authors identify 25 challenges for semantic process modeling. Figure 5 shows seven of the 25 challenges that are specifically related to the semantic of activity labels.

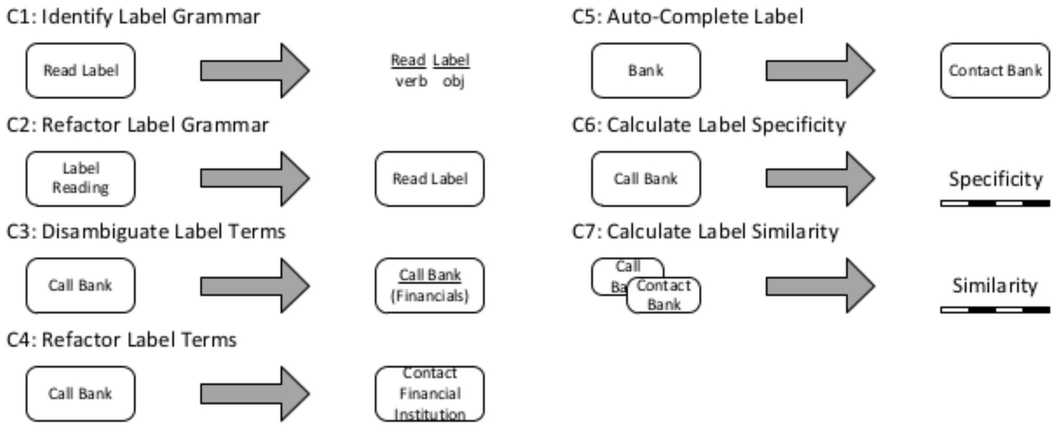



**Figure 5** Challenges related to the semantic of labels. Picture from Mendling et al. [2015]

## 3.2 Resource and resources type

We can mine resources and resource types out of document repositories. Given a VCS log, resources are present in the commit events. However their role remains unknown, as log files do not store any additional information about the resource besides the ID, which often consists of merely an email address. Therefore, we want to tackle the question: *how can we understand resource types?* Roles can be classified into two different types

- *Organizational roles*. For instance, Software Analyst or Project Manager. Such roles are typically defined in an organizational model describing the "static" structure of the company.

- *Functional roles*. These roles are more dynamic than the aforementioned roles, since they may change for different projects, e.g., a writer or a reviewer when preparing a project deliverable.

Organizational roles have received much interest from literature (Leitner et al. [2013], Baumgrass and Strembeck [2013] and process mining of the organizational perspective techniques have been implemented (Zhao and Zhao [2014], Song and van der Aalst [2008]). In recent work Schönig et al. [2015] present a novel approach that uses declarative mining and is able to deal with agile processes.

Efforts have also been made towards mining the functional roles of resources from software repositories. Anvik and Murphy [2007] compare two approaches that make use, respectively, of the source repository logs and of carbon-copy lists (CC:), comments, and bug resolver information from reports in bug networks. The evaluation was carried out by project experts from the Eclipse Platform. Results show an improvement with respect to a previous approach by Anvik et al. [2006] that took into consideration only the software repository files. The work of Yu and Ramaswamy [2007] takes into account the interaction between users of a VCS in order to group them into clusters. This allows distinguishing, for example, between *core* members, who interact often; and *associate* members, who interact more rarely. Tests on ORAC-DR [8] and MediaWiki[9] showed that this approach has a predictive accuracy (percentage of correctly classified samples) of at least 94% and a coverage (percentage of a developer roles that can be predicted by the rule) of at least 80%.

### 3.2.1 Roles of resources based on file

VCS repositories can also be used to identify roles. There is typically a connection between the type of the produced artifact and the role of the user who creates or accesses it. Policies, handbooks and guidelines can be further used to gather more insights on which resource can or may produce which artifact. For instance, the phrase *"The project manager is in charge of writing the meeting minutes"* gives information on both the artifact (i.e. `minutes.doc, minutes.txt, etc`) and the role of the resource who produces it. Furthermore, by knowing a-priori which resource can produce what artifact, it is possible to understand whether a role violation is going on (i.e. some resource is working on a task (s)he was not assigned to).

[8] http://www.oracdr.org/oracdr
[9] https://www.mediawiki.org/wiki/MediaWiki

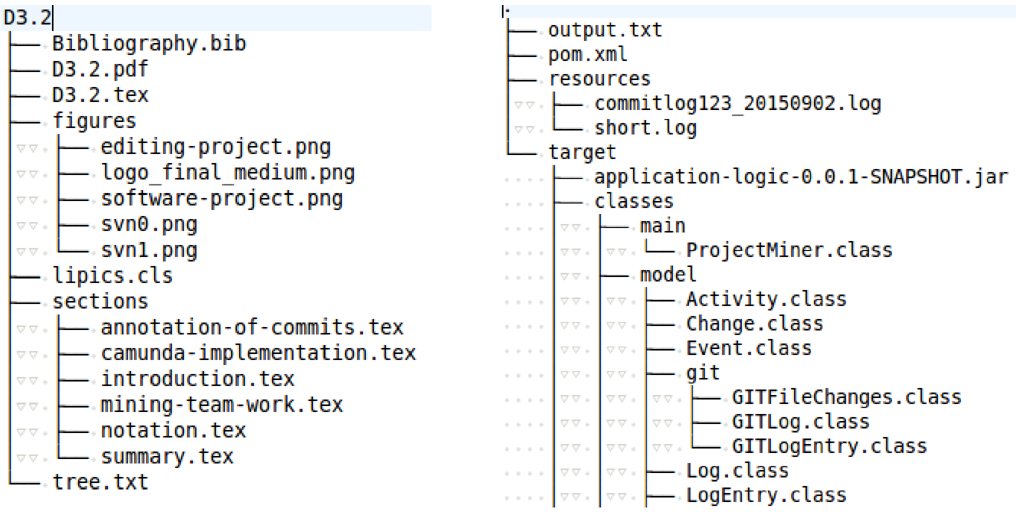
### 3.2.2 Annotating which is the highest node

Another annotation challenge is to map each resource to the right hierarchy level (s)he belongs to. This data can be easily understood from the VCS, but requires the project structure to respect the real access hierarchy according to the resource role. In this case we are able to discriminate between resources that operate on a higher level on the project tree and those who operate in depth. Such separations give us more information about the role of the resource. In fact, more resources such as managers or project leaders operate on the top of the tree hierarchy and they have more access and more visibility on the entire project. On the contrary resources such as programmers usually commit their changes in depth and have less access to the higher levels of the project hierarchy.

## 3.3 Understanding change weight

An important issue in project discovery is to understand the type of change events that the project undergoes. The challenge that we want to address in future work can be formulated as the question *how big is the change?*. This translates to understanding what type of events are actually happening in the project repository. It is possible to gather more insights into the project evolution if we look not only in which part of the project directory the changes happened, but also how much modification occurred in single files. This can be done by measuring the effective change of the files themselves. Many VCSs such as Git[10] or SVN[11] have *diff* commands incorporated that allow the user to see the differences between two different versions of a document. We plan to use such information for retrieving the file change, measured in terms of Lines of Code (LoC) or megabytes. Furthermore LoC is also a complexity metric, which in turn may lead to more changes. In this sense, complexity can also be related to tasks/effort. Rothenberger and Hershauer [1999] argue that LoC can be an appropriate measure for development effort. Especially when: i) the subject domain does not change; ii) the code is written in the same programming language; and iii) the software development company follows a rigid methodology. This is normally the case with SHAPE, where engineers follow a handbook and have a defined working methodology. Nevertheless, approaches to achieve better insights that go beyond LoC measures have also been addressed by Canfora et al. [2007], who are able to infer changes between two document versions even when the LoC count does not change.

[10] https://git-scm.com/
[11] https://subversion.apache.org/




**(a)** An editing project structure layout

**(b)** An excerpt from a software development project structure

**Figure 6** The structure of a project reflects also its type

## 3.4 Project depth

The project tree gives further information on the resources as well as on the type of project. For example, software development projects tend to present a very deep tree structure. Figure 6 shows the LaTeX layout of writing this manuscript and an excerpt from the project mining software, respectively on the left side and on the right side. It is easy to see how the structure is almost linear in the writing project (Figure 6a) as the maximum tree depth is 2, and how it drastically changes including deep-nested files up to 5 levels of depth, when it comes to a software project (Figure 6b). We can exploit the information that comes from the project depth to characterize certain types of work that strongly affect the tree depth. We plan to combine this information with other sources of evidence, such as filenames, for discovering the type or work.

## 3.5 Text recognition

It is possible to gather information on the meaning of the various parts of the project by also looking at common best practices that users adopt when using a VCS. For instance, SVN best practices[12] recommend the idea of a project root folder. Figure 7 illustrates that. A "project root" contains exactly three subdirectories: `/trunk`, `/branches`, and `/tags`. A repository may contain one or more

[12] https://svn.apache.org/repos/asf/subversion/trunk/doc/user/svn-best-practices.html

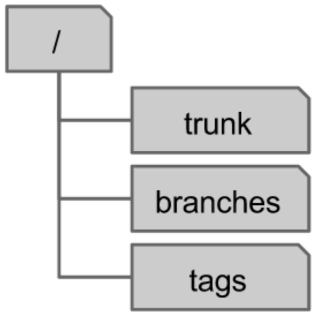



**Figure 7** Recommended repository layout for SVN projects. The trunk, tags, and branches trio is sometimes referred to as "the TTB directories".

project roots. The *trunk* is the directory under which the main project development occurs; *branches* is the directory in which various named branches of the main development line can be created; and *tags* is a collection of tree snapshots that are created, and perhaps destroyed, but never changed. Typically, a change in the repository reflects a single purpose: the fixing of a specific bug, the addition of a new feature, or any other particular task with a specified goal. Furthermore, it is usual for several projects to use clear guidelines and handbooks. Such handbooks suggest, for instance, how names should be used for the different artifacts that are produced within the project. Using this information, we can associate a meaning to the changes that are made to the project. This may lead to an improved activity recognition.

# 4 Mining teamwork from event logs

In D3.1 Cabanillas et al. [2015] we introduced an approach to mine the organizational perspective of business processes to represent resource-aware process models from the information stored in event logs Schönig et al. [2015]. In short, the approach discovered resource assignment rules describing (i) the conditions that the resources of the organizations must meet in order to participate in the process activities, and (ii) resource-related conditions that constrain the execution order of some activities, e.g., some types of resources may be forced to execute activities in a specific order, whereas other types of resources are not subject to such restrictions. These rules were extracted by confronting the information from past executions and the organizational model of the company (i.e., the positions, roles, organizational units, etcetera). That work was afterwards extended to include a post-processing step that simplified the generated process model by pruning out unnecessary, redundant resource assignment rules Schönig et al. [2015].

The aforementioned approach focused on activities that were performed by one single resource, i.e., individual work. We are currently working on extending the technique to also take into consideration activities that are collaboratively performed, i.e., which involve the participation of several resources. The goal is to mine the characteristics of the team members in such collaborative activities, e.g., to discover a team composed of a resource with a certain role $R$, a resource with expertise on a specific software application $S$ and a resource from a certain organisational unit $U$. Similarly to the previous approach, the framework to discover the characteristics of the team members is constituted by the workflow resource patterns Russell et al. [2005], specifically the following creation patterns:

- *Direct Distribution*, i.e., the ability to specify at design time the identity of the resource that will execute a task.
- *Role-Based Distribution*, i.e., the ability to specify at design time that a task can only be executed by resources that correspond to a given role.
- *Organisational Distribution*, i.e., the ability to offer or allocate activity instances to resources based on their organisational position or organisational units and on their relationship with other resources.
- *Capability-Based Distribution*, i.e., the ability to offer or allocate instances of an activity to resources based on their specific capabilities.

Following the same approach as for individual work mining, the mining of team compositions will consists of three steps:

1. A pre-processing phase will filter the teams found in the event log to later process only those that are frequent, i.e., the combination of team members that occur at least a certain amount of times according to a established threshold. The rest of member combinations will be considered exceptions and will not be taking into account for the characterization of the teams that are allowed to execute a certain collaborative activity.
2. The aforementioned creation patterns will be checked within the different teams for each activity in order to find out which rules apply for the team members, i.e., how the team allowed to execute an activity is characterized. This characterization is not accurate, since it is only able to determine the different creation patterns that occur within a team, but not how they are distributed among the team members. This implies that if we have, e.g., a team composed of three members and we discover three rules $direct(activity, Peter)$, $role(activity, Coordinator)$ and $capability(activity, speaksEnglish)$, at this point we do not know whether there must be one person with each of these characteristics, or there must be several members meeting one or several conditions.

3. A two-step post-processing phase will increase the precision of the information about the team composition by specifying the distribution of the discovered characteristics among the team members in order to bridge the previous gap.

# 5 Summary and future work

This deliverable sets the field for future developments on extracting process knowledge from unstructured and semi-structured data. We have identified and explained seven possible new insights that we can obtain from combining data from repositories with semantic annotation from text. This document also contains a novel mining technique for resource-perspective mining. This approach allows for discovering teamwork. As last, we have also modified the original GUI based application into a Java library that can be run inside Camunda.

For future work we aim at a publication on improving the project discovery technique using user comments from commits. We plan on using the points that we identified earlier in this deliverable as a starting base for the next dissemination. We are also working on the three-step approach for mining team compositions that we outlined in Section 4, which we plan to submit as a journal publication in the following weeks.

# References


John Anvik and Gail C Murphy. Determining implementation expertise from bug reports. In *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*, pages 2–2. IEEE, 2007.

John Anvik, Lyndon Hiew, and Gail C Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370. ACM, 2006.

Thomas Baier, Jan Mendling, and Mathias Weske. Bridging abstraction layers in process mining. *Information Systems*, 46:123 – 139, 2014. ISSN 0306-4379. http://dx.doi.org/10.1016/j.is.2014.04.004. URL `http://www.sciencedirect.com/science/article/pii/S0306437914000714`.

Saimir Bala, Cristina Cabanillas, Jan Mendling, Andreas Rogge-Solti, and Axel Polleres. Mining project-oriented business processes. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, *Business Process Management*, volume 9253 of *Lecture Notes in Computer Science*, pages 425–440. Springer International Publishing, 2015. ISBN 978-3-319-23062-7. 10.1007/978-3-319-23063-4_28.

Anne Baumgrass and Mark Strembeck. Bridging the gap between role mining and role engineering via migration guides. *information security technical report*, 17(4): 148–172, 2013.

Cristina Cabanillas, Jan Mendling, Axel Polleres, and Saimir Bala. Requirements for process, resource and compliance rules extraction from text. Technical Report 1, 2015.

Gerardo Canfora, Luigi Cerulo, and Massimiliano Di Penta. Identifying changed source code lines from version repositories. In *MSR*, volume 7, page 14, 2007.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165 (1):91–134, 2005.

Tudor Ionescu. Workflow-processing and verification for safety-critical engineering – conceptual architecture. Deliverable D6.1 (M2), 2015.

Maria Leitner, Anne Baumgrass, Sigrid Schefer-Wenzl, Stefanie Rinderle-Ma, and Mark Strembeck. A case study on the suitability of process mining to produce current-state rbac models. In *Business Process Management Workshops*, pages 719–724. Springer, 2013.

Henrik Leopold, Sergey Smirnov, and Jan Mendling. Refactoring of process model activity labels. In ChristinaJ. Hopfe, Yacine Rezgui, Elisabeth MÃ©tais, Alun

Preece, and Haijiang Li, editors, *Natural Language Processing and Information Systems*, volume 6177 of *Lecture Notes in Computer Science*, pages 268–276. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13880-5. 10.1007/978-3-642-13881-2_28.

Jan Mendling, Hajo A Reijers, and Wil MP van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.

Jan Mendling, Henrik Leopold, and Fabian Pittke. 25 Challenges of semantic process modeling. *IJISEBC*, 1(1):78–94, 2015.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145595.

Marcus A Rothenberger and James C Hershauer. A software reuse measure: Monitoring an enterprise-level model driven development process. *Information & Management*, 35(5):283–293, 1999.

Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *CAiSE*, pages 216–232, 2005.

Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. A Framework for Efficiently Mining the Organisational Perspective of Business Processes. *Decision Support Systems*, page Under review, 2015.

Stefan Schönig, Cristina Cabanillas, Stefan Jablonski, and Jan Mendling. Mining the organisational perspective in agile business processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 37–52. Springer, 2015.

Minseok Song and Wil MP van der Aalst. Towards comprehensive support for organizational mining. *Decision Support Systems*, 46(1):300–317, 2008.

Liguo Yu and Srini Ramaswamy. Mining cvs repositories to understand open-source project developer roles. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 8. IEEE Computer Society, 2007.

Weidong Zhao and Xudong Zhao. Process mining from the organizational perspective. In *Foundations of Intelligent Systems*, pages 701–708. Springer, 2014.