### Project Details

- **Project number:** 258109
- **Project Acronym:** FastFix
- **Project Title:** Monitoring Control for Remote Software Maintenance
- **Call identifier:** FP7-ICT-2009-5
- **Instrument:** STREP
- **Thematic Priority:** Internet of Services, Software, Virtualization
- **Start date of the project:** June 1, 2010
- **Duration:** 33 months

---

### D3.4: 1st Prototype of the User Profiler

- **Lead contractor:** TUM
- **Author(s):** Tobias Roehm, Dennis Pagano, Sergio Zamarripa, Maria del Carmen Calle, Walid Maalej
- **Submission date:** June 2012
- **Dissemination level:** Public

---

Project co-funded by the European Commission under the Seventh Framework Programme

© S2, TUM, LERO, INESC ID, TXT, PRODEVELOP
Abstract: This document summarizes how the user profiler component of FastFix works, how it is implemented and how to execute it. It is a supplement to the source code of the first version of the user profiler which can be accessed from the FastFix repository\textsuperscript{1}.

\textsuperscript{1}https://repository.fastfixproject.eu/svn/fastfix/trunk/
## Contents

1 Introduction ................................................. 4  
   1.1 About This Document .......................... 4  
   1.2 Motivation and GeneralOverview .............. 4

2 Implementation Details .................................... 6  
   2.1 Overview ........................................ 6  
   2.2 Frequency, Duration, Age - Model (FDA) ..... 6  
   2.3 Running the User Profiler ...................... 7

3 Limitations and Future Work .......................... 9  
   3.1 Implementation ................................ 9  
   3.2 Evaluation .................................... 9

Bibliography .............................................. 10
1 Introduction

1.1 About This Document

This document is a supplement to the source code of the first version of the user profiler component of FastFix. It describes briefly the concepts behind the user profiler and how it is executed. First, we describe why the user profiler is necessary within FastFix and how it is used (this chapter). Second, we summarize how the user profiler is implemented, the important concepts used, and how to get the user profiler running (Chapter 2). Fourth, we describe limitations of the current implementation and future work (Chapter 3).

As this document is a source code supplement, it is rather short and abstract. For more detailed information about the user profiler we refer to D3.2 [1] and D3.2e [3].

The source code of the user profiler component can be found in FastFix repository\(^1\) in bundle eu.fastfix.server.user.profiler.

1.2 Motivation and General Overview

An important success factor of software applications is their support of user needs. The detailed information about user behavior elicited by the FastFix sensors can be used to analyze user behavior, detect usability issues, derive ways to improve the support of user needs, and consider user behavior as cause of bugs.

Because lots of user behavior information is generated by the FastFix sensors, this information has to be aggregated before it can be used efficiently. The user profiler aggregates the behavior of individual users over time and creates a profile for each individual user. For example, it contains information how often a user performed a certain action or how often a user visits a certain application part. It operates on all user-related data, which is a subset of all events generated by the context system.

Each user profile stores information how often (frequency), how long (duration) and when (age) a specific user \(u\) performs a certain action such as clicking on a button or entering

\(^1\)https://repository.fastfixproject.eu/svn/fastfix/trunk/
text. The information is combined with the current artifact as context, i.e. it is stored if
a user enters text on view Login or on View Invoice.

The user profiler allows specific and general queries about the experience of a certain
user with a certain part of the application. Examples of specific queries are how often
a user entered text on form Login or how often the user issued the CreateInvoice command
from the Service component of the application. Further, examples for general queries are
how often the users entered text so far (regardless of the application part) or how often
the user issued the CreateInvoice command in total. Our hypothesis is that these queries
help to determine the experience and knowledge of a user, e.g. if he or she is using an
application feature for the first time. Different users can be compared based on these
individual profiles.

Information provided by the user profiler will be used in several FastFix components
and scenarios. First, the error report generation component will profile information to
filter relevant events based on frequency and FDA information. Second, the error replay
component will use profile information to highlight and filter relevant events based on
frequency and FDA information during error replay. Third, the pattern mining component
will use profile information to take frequency and FDA information into account when
mining for behavior patterns indicating errors. Fourth, the event correlation component
will use profile information when reasoning about user behavior that might indicate or
cause errors, such as infrequent actions.
2 Implementation Details

In this chapter we summarize how the user profiler is implemented and how a user profile looks like. Further, we describe how the user profiler can be executed.

2.1 Overview

The user profiler component is implemented in package eu.fastfix.server.user.profiler within the FastFix server. It registers at the server context bus. When it receives an event, it updates the user profile for the user, event type and artifact of the current event. Figure 2.1 summarizes the working mode of the user profiler.

![User Profiler Overview Diagram]

Figure 2.1: User Profiler Overview

2.2 Frequency, Duration, Age - Model (FDA)

Each individual user profile is implemented using the FDA Model [2]. More specifically, for each combination of user id, event type and artifact id, the frequency, the duration and a
timestamp are stored. Frequency denotes the number of occurrences of this combination. Duration denotes the sum of the duration of all occurrences. And timestamp denotes the last occurrence of this combination and is used to derive the age. An experience value is calculated from these data using the following formula.

Let $E$ be the set of events, $A$ the set of artifacts and $U$ the set of users. We define the experience of user $u \in U$ with an event $e \in E$ regarding an artifact $a \in A$ formally as

$$experience(u, e, a) := \frac{freq(u, e, a) \cdot dur(u, e, a)}{age(u, e, a)},$$

with $freq \geq 0$, $dur > 0$, and $age > 0$ for all $e \in E$, $a \in A$ and $u \in U$.

Frequency, duration and last timestamp are stored in the database for each combination of user id, event type and artifact id. Age and experience value are calculated when a query is made based on the values retrieved from the database. Table 2.1 shows the database table holding the user profile information.

<table>
<thead>
<tr>
<th>User Id</th>
<th>Event type</th>
<th>Artifact Id</th>
<th>Freq</th>
<th>Dur.</th>
<th>Last timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ButtonClick</td>
<td>LoginView.LoginButton</td>
<td>6</td>
<td>6</td>
<td>03.05.12, 13:00 h</td>
</tr>
<tr>
<td>1</td>
<td>ButtonClick</td>
<td>LoginView.AboutButton</td>
<td>2</td>
<td>5</td>
<td>03.02.12, 13:00 h</td>
</tr>
<tr>
<td>1</td>
<td>TextInput</td>
<td>LoginView.UsernameField</td>
<td>8</td>
<td>12</td>
<td>08.02.12, 09:13 h</td>
</tr>
<tr>
<td>2</td>
<td>Button Click</td>
<td>LoginView.LoginButton</td>
<td>237</td>
<td>311</td>
<td>07.05.12, 18:12 h</td>
</tr>
</tbody>
</table>

Table 2.1: User Profiles in Database (Underlined captions denote primary key)

### 2.3 Running the User Profiler

In this section we briefly describe how the user profiler can be executed and how to inspect the created profiles.

#### System Requirements

The following infrastructure is required to run the user profiler:

- MySQL installed (e.g. MAMP or LAMP)
- Java 1.5 installed

Further, the FastFix server has to be configured to be able to communicate with MySQL by setting the correct database credentials (port, username, password).
Starting User Profiler

The user profiler is started by starting the FastFix server and ensuring that the bundle eu.fastfix.server.user.profiler is incorporated and running. The user profiler is included in all FastFix releases from release 1.1. on.

Additionally to the bundle eu.fastfix.server.user.profiler the following bundles and their required bundles should be included and started:
eu.fastfix.server.communication, eu.fastfix.server.context.core.

Please note that when only the FastFix server is started as described here, nothing will happen as the user profiler is triggered by context events that arrive from the FastFix client. Hence, the FastFix client and at least one sensor has to be started additionally.

Inspecting User Profiles

The part of the user profiles that is stored in the database can be inspected easily by viewing the contents of table ExperienceProfileData in database FastFix* of the used MySQL instance. The table has the schema as described in Table 2.1.
3 Limitations and Future Work

3.1 Implementation

In the following, we summarize limitations of the current implementation of the user profiler. These issues are also points to work on during the next months.

- User identification and sessionization
  Currently, users provide a user id manually when starting the MOSKitt sensors. Future work will be to develop and implement a mechanism to identify users automatically. Thereby it is not necessary to elicit the specific username. But the method should be able to identify a user along several runs of the target application.

- Duration calculation
  Currently, all events are atomic events that occur at a certain point in time and have no duration.
  Future work will be to implement duration calculation for non-atomic events. For example, an event type that denotes inactivity of a user and has the duration that is delimited by event $e_1$ occurring before and an event $e_2$ occurring after the inactivity.

- Comparison of user profiles
  The information collected by the user profiler resembles a profile for each individual user.
  Future work will be to compare users based on this profile and to find groups of similar users using clustering algorithms.

3.2 Evaluation

The user profiler component is currently implemented as described in this document. In order to assess the correctness of its user model and its usefulness for FastFix scenarios, an evaluation is still necessary. Our hypothesis is, that the user profiler (as described in this document and D3.2e [3]) is appropriate to capture information about users and
D3.4: 1st Prototype of the User Profiler

useful in several FastFix scenarios. Also, the FDA model has been used in other contexts successfully. For example, Maalej [2] used it in the domain of tool integration.

In order to evaluate the user profiler, we plan the following activities:

- **Data collection during FastFix trials**
  The user profiler will be deployed in the FastFix trials and will collect real user data during the trials. After the trials, we will analyze the data collected, compare them with real users, and assess if the model describes real users well.

- **Empirical evaluation study**
  We will implement the exploitation of user profile information as described in Section 1.2. Then we will evaluate if the claims about user profiler hold and what improvements can be made by using profile information.
Bibliography

