INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Process Mining of Enterprise Applications
based on OutSystems Agile Platform

Alexandre Luis de Bastos Coutinho Gonçalves Gouveia

Dissertation for the degree of
Master of Science in Information Systems and Computer Engineering

Committee

Prof.ª Dr.ª Maria dos Remédios Cravo (president)
Prof. Dr. Diogo R. Ferreira (supervisor)
Prof. Dr. Miguel Mira da Silva (co-supervisor)
Prof. Dr. Bruno Martins

November 2010

Acknowledgments

To my family that always supported me and made my academic path possible.

To Professor Diogo Ferreira and Professor Miguel Mira da Silva for their remarkable and precious guidance, suggestions, and availability to help that provided great value to this dissertation.

To Lúcio Ferrão of OutSystems for providing the initial assistance that allowed me to apply the process mining techniques studied in this dissertation to the OutSystems Platform.

To Carlos Mendes for the support given over the OutSystems server and published applications that were fundamental for this dissertation.

To Carlos Libório and Álvaro Rebuge of the process mining research group for the support and exchange of ideas.

To all my friends that accompanied me during the college years.

Abstract

Process Mining allows the analysis of business processes based on event logs. In this work, these event logs contain the events recorded during the execution of a certain application, where one can retrieve data from its operations, such as their identification, their status, the date and time they were executed and the connection they have to other operations. By analyzing a considerable amount of events through process mining algorithms, it is possible to discover behavioral patterns as well as the model for the process that drives application execution. If discrepancies between this model and the original behavior devised for the application are detected and are significant, an overall improvement to the application may be achieved through successive refinements. The purpose of this document is to illustrate how process mining can be applied in a specific application development platform, namely the Agile Platform of OutSystems.

Keywords: Process Mining, OutSystems Platform, Agile Development, Event Log, Process Models

Resumo

A Extracção de Processos permite a análise de processos de negócios baseados em logs de eventos. Estes logs contêm os eventos gerados durante a execução de uma determinada aplicação, dos quais é possível recuperar dados sobre operações efectuadas na aplicação, tais como o estado, os identificadores, a data e a hora em que foram executadas essas operações e as ligações estabelecidas com outras operações. Ao analisar uma quantidade considerável de eventos através de extracção de processos podem ser descobertos padrões de comportamento, assim como o modelo para o processo que controla a execução da aplicação. Se forem detectadas discrepâncias entre este modelo e o comportamento previsto da aplicação, e se estas discrepâncias forem significativas, uma melhoria global da aplicação pode ser conseguida através de refinamentos sucessivos. O objectivo deste trabalho é mostrar como a extracção de processos pode ser aplicada na Plataforma Ágil de desenvolvimento de aplicações da OutSystems.

Keywords: Extracção de Processos, Platforma OutSystems, Desenvolvimento Ágil, Log de Eventos, Modelos de Processos

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

An enterprise using information systems may have a significant concern about whether the systems being used are serving and will continue serving its best interests and goals. These systems, in order to support the business services that the enterprise provides, must be aligned with the management processes and the work of the people that are part of those services (Silva and Martins, 2008). Therefore management, people and technology are always related to one another.

Technology alone is not sufficient to deal with people's problems, since workers often do not know the full potential of the technology they are using and the enterprise can only take advantages from it if business decisions can be carried out properly. So one frequent problem in the enterprise is to know if people are doing their work as business expects them to.

Taking this into account, with the aim of presenting an approach to this problem, the motivation for this dissertation is to provide a study of the ability to retrieve usage information from applications developed in the OutSystems Agile Platform (OutSystems, 2008b). More specifically, this approach was based on Process Mining concepts and techniques (Weijters and Aalst, 2001; Aalst and Dongen, 2002; Aalst et al., 2003a,b, 2005) in order to analyse sets of events that were triggered by users interacting with those applications and, ultimately, to allow business models' improvements embedded on those same applications.

Commercial Business Intelligence (BI) tools look at aggregate data, such as frequencies, averages, utilization, service levels and focus on performance indicators to draw business analysis and conclusions. Although this is an important aspect for any enterprise, these tools do not enable the possibility to look to causal dependencies or bottlenecks. This may only be achieved by an analysis at a lower level of the execution logs of the applications. In this work this analysis will be performed by making use of process mining techniques.

1.2 Goals

An event log typically contains information about events referring to an activity and a case. The case being handled (also named as process instance) can be a customer order, a job application, an insurance claim, a building permit, etc. The activity, named

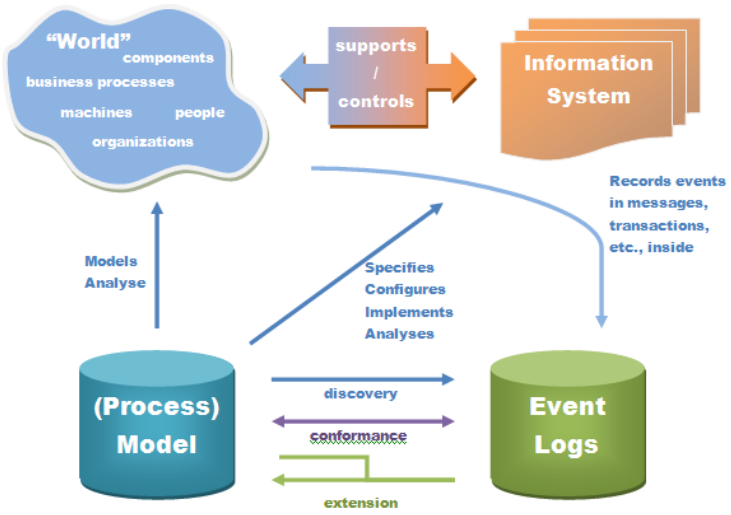


Figure 1.1: Process Mining relates the business world with its information systems and models through log discovery (adapted from Aalst et al. (2009))

task, operation, action, or work item is some operation on the case with a timestamp associated, indicating the time of occurrence. Moreover, when people are involved on these operations, event logs will characteristically contain information of the performer, executing or initiating the event.

The aim of Process Mining is to extract knowledge using specific techniques on logs where information systems record events from business contexts of the enterprise. By comparing the process models that specify those same systems with models that derived from that knowledge, it is possible to arrive to conclusions of compliance in actions or behaviors in those contexts, as Figure 1.1 depicts.

Throughout this discovery, management questions can be answered, such as:

- Who in the organization is responsible for the occurrence of certain events?
- How much time was spent between specific tasks in the process?
- How many people are typically involved in a case? Are they all required in order to complete the case?
- Do the given cases comply with the existing process model?

One of the great challenges of Process Mining lies in accessing event logs recorded by any enterprise application in order to retrieve and store the relevant information. The challenge has been acknowledged and partially met by a large framework supporting process mining research[1] – the ProM Framework (Dongen et al., 2005; Aalst et al., 2007a).

In the context of the work, the framework was used to analyse the event logs produced by the OutSystems Platform with the following goals:

- Study the possible methods to extract run-time data from applications developed in the OutSystems Platform.

[1] More information on www.processmining.org

- Develop preprocessing techniques to achieve a standard log format from the extracted data, with all the required fields.
- Identify the process mining techniques relevant for analyzing the extracted logs.
- Implement visualization capabilities for the extracted models.
- Compare the models developed in the OutSystems Platform with the extracted models and related information from event logs analysis.

All these goals aim to support the development cycle of the applications developed in the OutSystems Platform. By applying process mining techniques it will be possible to study the run-time behavior of these applications in new ways which, in turn, will foster the continuous development of these applications.

1.3 Document Structure

After the general introduction of the previous sections, the document is organized in the following manner:

- Chapter 2 provides information related to the OutSystems Platform and the type of development inherent to its applications.
- Chapter 3 details process mining techniques that were found to be of interest in analysing the event logs.
- Chapter 4 explains the approach that was adopted to retrieve relevant information from those logs.
- Chapter 5 demonstrates the proposed approach applied to an online application used in a real-world scenario.

The main contributions and future work are discussed in Chapter 6.

Chapter 2

The OutSystems Platform

Many researchers are developing new and more powerful process mining techniques. Software vendors are incorporating these in their BPM products but few of the more advanced process mining techniques have been tested on real-life processes (Aalst et al., 2007b). Before proceeding to process mining algorithms, the first step is to verify that the platform is able to record the necessary information. Using process mining techniques it is possible to extract a model that describes the behavior of the application. At the end, this model may be compared to the original behavior devised for the application with the final objective of determining whether the application behaves and is used as expected.

With the recent technology breakthroughs on information systems, new tools have been developed to offer methods of agile development (Theunissen et al., 2003; Abrahamsson et al., 2003). The development cycle of the applications from the OutSystems Platform rely on these, allowing a great flexibility when developing them, since they can be easily altered when necessary. Although the tools are based in other technologies, the final result can be related to the management of business processes, where the organization has the flexibility of adapting it to the current business situation. Despite agile development, these tools may not be aligned with the organization and may not allow to fully implement the business processes.

The following sections convey a better understanding about what kind of applications will be examined, detailing on how agile methodologies contribute to their development.

2.1 Agility in OutSystems Platform

OutSystems is a registered trademark and company distributing a platform based on agile development methodologies[1]. Agile Platform is currently used by software teams to rapidly develop and manage flexible web applications. On the 5.0 version of the platform (OutSystems, February 18, 2009), the possibility of business processes development using agile methodologies was fully integrated (Figure 2.1) by providing a rich and flexible user interface which allows an unified design, execution and management of both applications and processes, avoiding complex integration projects. An explicit launch of a process can be made using the built-in process APIs inside business logic flows, and an implicit launch can react to database events such as inserting or updating

[1] Agile Platform at www.outsystems.com/agile-platform

entity records. The publishing action deploys both processes and applications simultaneously, ensuring both are always synchronized. A workflow may be easily triggered when new data is inserted or changed, enabling task creation, email alerts, data update or web service triggering.

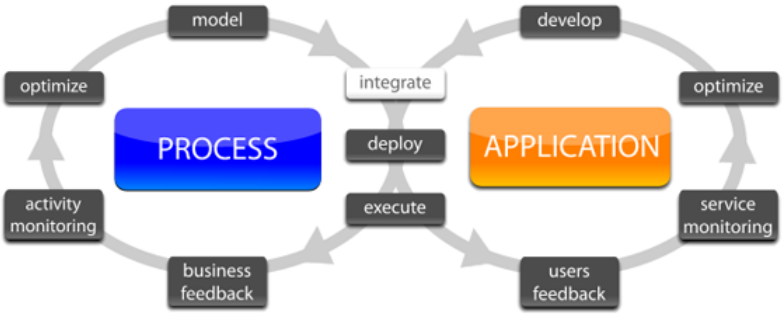


Figure 2.1: OutSystems brings the Application Development and Business Process Management life cycle together (OutSystems, February 24, 2010,F)

The Agile Platform is currently used for business applications and web sites by companies across industries including Telecom, Banking and Insurance, Pharma, Utilities, Consumer Goods and Government.

Agile Platform deals with the actual needs of rapid development and continuous change on the developed applications, concerning technological and business requirements of organizations. The underneath methodology is based on an adaptation of the SCRUM Agile Methodology (Rising and Janoff, 2000; Schwaber, 1995; OutSystems, November 11, 2008,J).

Project development with the OutSystems Agile Methodology has a sequence of iterations named Sprints. At the end of each one, a limited functional version of the system is ready for publication (i.e., deployment). These iterations of one to two weeks duration have analysis, development and testing activities where one or more features of the final system is completed. When all the iterations have been accomplished, the system has all the features available tested and approved. In the application life cycle, with the aid of the user's feedback, functionality improvement, error repairing and new features may be easily added without high risk or business impacts, since a built-to-change approach is being taken (OutSystems, 2008a).

2.2 Development Environment

The Agile Platform adopted a drag-and-drop programming style allowing the programmer to create an application without a single line of hand-written code, as can be seen in Figure 2.2.

From data modeling, to business logic definition and to interface construction, they are all visually specified on the platform, diminishing the misalignment between Business and Information Technology. Since learning business aspects is facilitated, this allows the technology's responsible staff to better handle and to understand what technology issues need to be dealt with. The flows of the screens and actions embedded on the application can be manually edited, also by drag-and-drop in the Service Studio, as depicted in Figure 2.3. Business users, testers and all the staff allowed to access

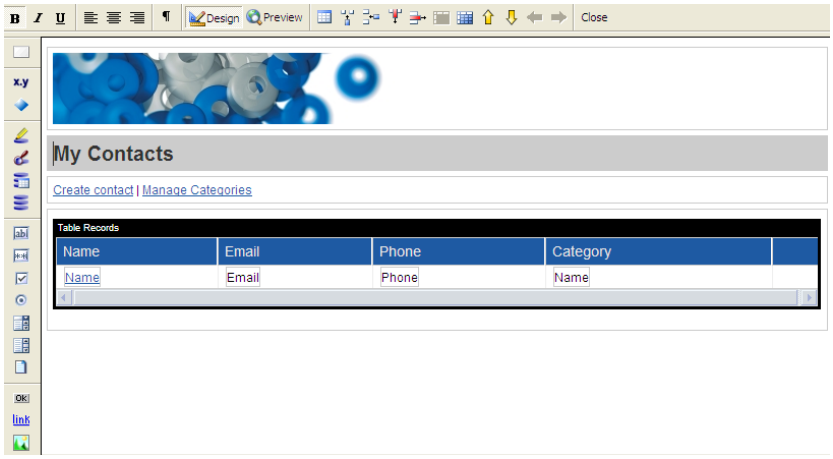


Figure 2.2: Service Studio's drag-and-drop screen editor environment

the running application webpage are able to pinpoint an area and leave their comments on it (OutSystems, 2009). These comments become available to project managers and developers for review, providing a rapid, unambiguous automated collaboration mechanism. This direct feedback's retrieval from a running web application is based on Embedded Change Technology (supporting the ITIL Incident Management process in a similar manner to the Issue Manager solution presented in the case study of (Ferreira and Silva, 2008).

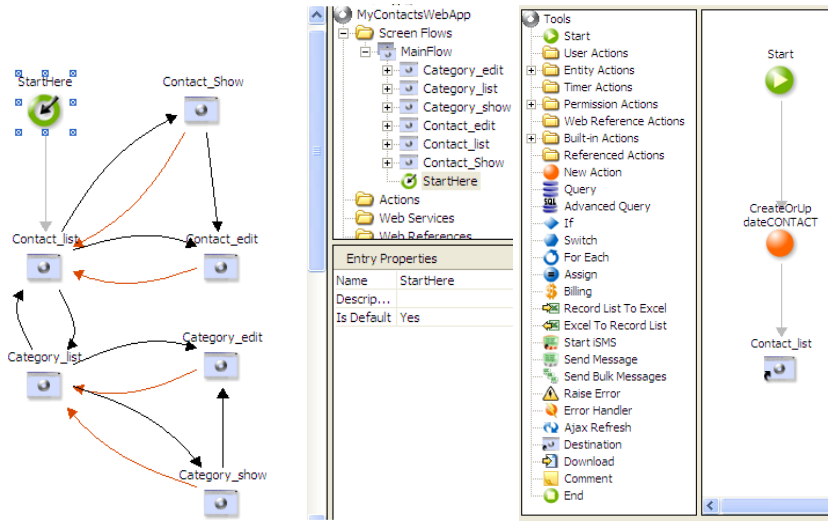


Figure 2.3: Service Studio's flows of screens and actions

All applications developed with the Agile Platform can be deployed as standard .NET or Java applications, not being closely coupled to a specific graphical user interface library or toolkit. Any web business application developed is fully open and standard. The generated code uses standards as ASP.NET, JSF, XML web services, cross-browser XHTML, JavaScript, XML, and Web 2.0 technologies like AJAX. This

allows the developed applications to easily interact with other systems and stay in-line with architectural guidelines, solving the portability problem of typical graphical user interfaces (Bishop, 2006).

To better understand the supporting technology, an overview of the Agile Platform is presented in Figure 2.4, comprising the main components briefly described below.

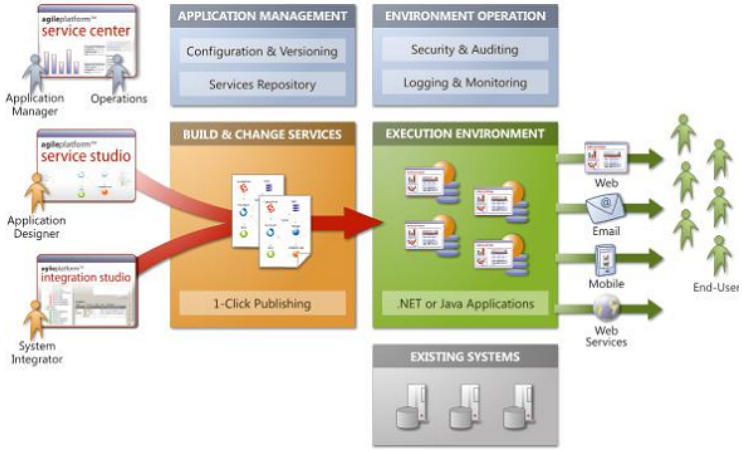


Figure 2.4: Agile Platform Overview (OutSystems, 2008b)

The **Integration Studio** is a desktop environment where developers create custom components to integrate external applications and databases. Its wizards help in the integration of databases, library APIs and third party applications such as SAP. Microsoft Visual Studio and Eclipse may assist in the creation of custom integration adapters which are published with one click and then can be reused several times in different web business applications.

The **Service Studio** is a desktop environment targeted at business-minded developers to rapidly assemble and change web business applications and business processes using visual models. The web application is defined within this studio, where web 2.0 User Interfaces, Business Logic, Databases, Integration Components, web Services, Security Rules, Scheduling activities and Business Processes can be modeled. Deployment and publishment is made by 1-Click Publishing, an automated mechanism encompassing consistency checking, component cataloging, version control, code generation and optimization, and distributed deployment.

The **Service Center** is a console for the operational management of the platform. It allows monitoring and auditing of the running applications to detect and isolate performance and quality issues, and to manage application server farms. Version control and configuration management is available for all web applications, services, integration adapters and other application resources including processes.

Thereby, useful information for process mining can be retrieved from monitoring tools. Figure 2.5 shows how the monitoring tools access specific tables on the database to produce the respective logs. In this particular case it shows the Screen Log that accesses the Screen table of the database in order to present the content of the attributes that are related to the user's interaction with the applications that are published in the server.

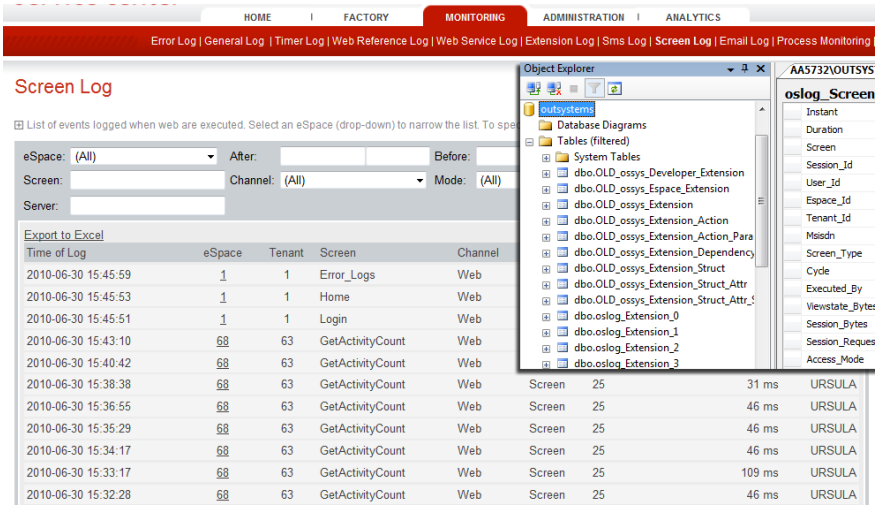


Figure 2.5: The Screen Log on the monitoring section of the Service Center contains the fields of the OutSystems database tables

2.3 Application development example

This section illustrates a simple application developed in OutSystems. The steps to develop this application can be found at the OutSystems website after the installation of the Agile Platform's Community Edition (version 5.1 (OutSystems, June 29, 2010))[2].

This particular application was built by loading employee records from an Excel Worksheet into a table on the database, as Figure 2.6 (a) depicts.

Once the loading is completed, just by dragging the employee entity to a web screen, a table is automatically created with the employee's attributes (Figure 2.6 (b)). The contents of this table are available in the web browser and can be searched via a search box embedded in the HTML as soon as the publish button is clicked.

New webpages for showing or editing individual records of the database's table can be automatically created by right clicking the employee's name in the list and establishing a linkage to those same webpages, as depicted in the (c) step of Figure 2.6. The connections, queries and the logic behind the operations that these web screens provide are also automatically established.

Figure 2.6 (d) shows an image widget added to the employee's edit webpage. When this is done the entity related to the employee's picture is automatically created. Any other new entities, such as the Note entity in Figure 2.6 (e), that are to be related with the employee's entity, just need to have an EmployeeId attribute. Once the two entities are dragged to the diagram, the connection between them is also automatically established, thus creating a full and extensible management application for employees.

Figure 2.7, on the other hand, presents the Screen Flow that is generated automatically throughout the development steps described and depicted on Figure 2.6. This flow shows all the relevant and only webpages, with their respective transitions between them, that users are allowed to visit in this application, therefore this would be the flow to find through process mining techniques after the users visited those webpages.

[2]The Community Edition is available at www.outsystems.com/download

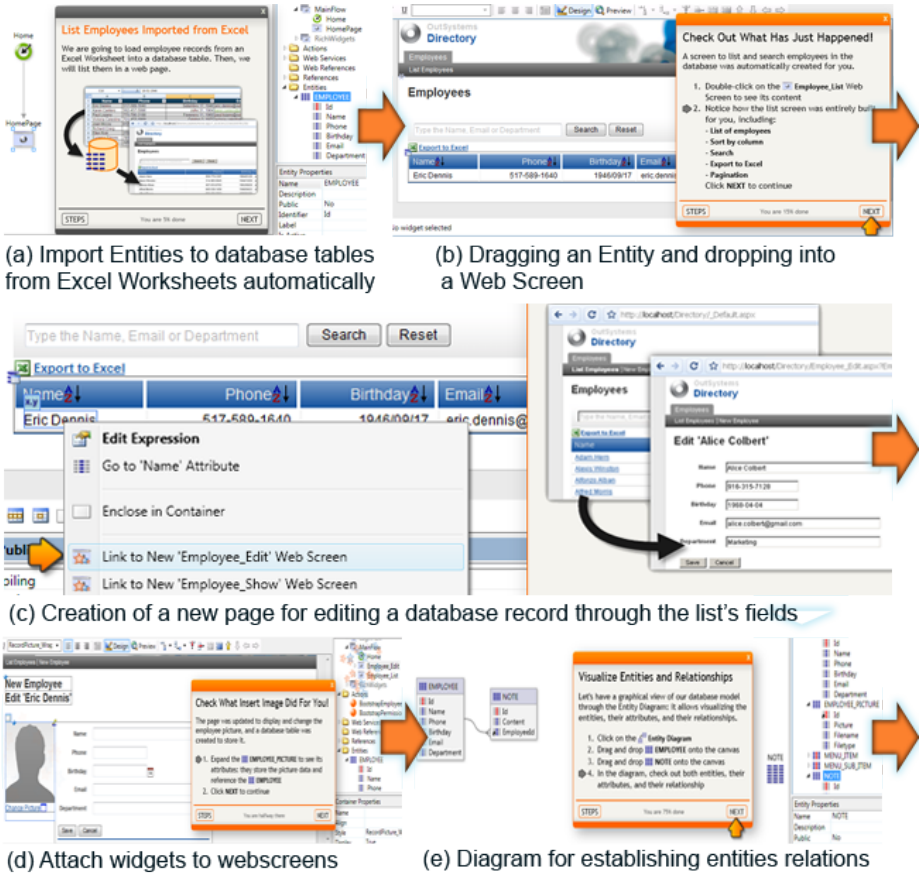


Figure 2.6: Simple application development with guiding steps in the Service Studio

This is the main goal of this work - to recover from event logs the behavior's model of the application in order to compare it with the flow designed in the Service Studio of the OutSystems Platform (as the one depicted in Figure 2.7).

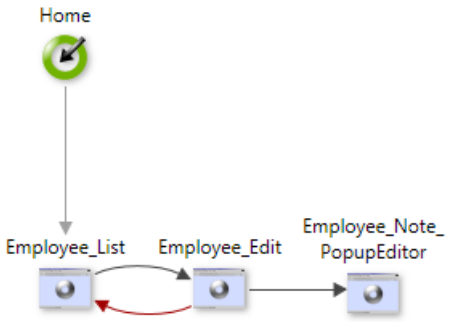


Figure 2.7: Screen Flow of the application depicted in Figure 2.6

2.4 Summary

In this chapter the OutSystems Platform main components were described as well as the way they relate to agile development. A sample application was illustrated in order to provide some insight into the applications that can be developed using this platform and the respective screen flow that is built linking the application's webpages.

Chapter 3 describes process mining techniques to provide a better understanding of the possible choices to analyze the Agile Platform's event logs.

Chapter 3

Process Mining Techniques

The term of Process Mining was introduced by Aalst et al. (2003b) for the method of extracting the description of a structured process from a set of events, where each event refers to a task (i.e., a real execution). In their work they referred to process discovery concepts (Cook and Wolf, 1998) and the idea of applying this method on the context of workflow management (Agrawal et al., 1998). There are several mining algorithms developed to date to facilitate the discovery of business processes or to check the conformance of run-time behavior (Rozinat, 2006) with respect to a given process model (using metrics to allow the comparison of the observed behavior in the event log and the control flow specified by the process model (Rozinat and Aalst, 2008)). The following sections contain process mining techniques that are related to the context of the event logs of this work. Section 3.5 presents the framework that allows the usage of these techniques by implementing those algorithms.

3.1 Control-Flow Mining Techniques

The goal of process mining is to collect data at run-time to support workflow design and analysis (Aalst et al., 2003a). Event logs are the base to extract information about processes. Instead of starting with a workflow design, the objective is to start gathering information about the workflow processes as they take place. In this type of mining, in order to record events they must conform to these rules:

- each event refers to a task (i.e., a well-defined step in the workflow);
- each event refers to a case (i.e., a workflow instance);
- events are totally ordered, where additional causality information can be extracted through each event's timestamp.

When a user of an application visits its webpages, it is possible to establish a workflow by connecting these visits as if they were events themselves. This is allowed due to the similarity of the context between workflow tasks and functionality provided by the application in its webpages. Each webpage refers to a screen action which is a well-defined step in the workflow (as a task), a session which is a workflow instance (as a case) and a timestamp which allows to order the entire sequence of visits to the webpages. The following subsections describe algorithms that can extract process models from such event logs.

3.1.1 Alpha (α) algorithm

The ProM α-algorithm plug-in implements the α-algorithm by constructing a Petri net that models the workflow of the process. Its objective is to deterministically infer relations between activities, thus ensuring that the log is complete and no further behavior may appear.

Process mining with this algorithm has known restrictions (Aalst et al., 2003b), namely difficulties in discovering:

- frequent and non frequent behaviors, where the latter is normally associated to noise;
- short-loops;
- duplicate tasks;
- implicit dependencies.

Further development has been made to improve this algorithm resulting in implicit dependencies detection (Wen et al., 2006) and the α++ algorithm, where short-loops can be identified (Medeiros et al., 2004a,b).

3.1.2 Heuristics Miner

Heuristics Miner operates over the control-flow perspective of a process model. It considers the order of the events within a case but disregards their order among cases of the same process. I.e., given two activities belonging to the same case, it is important to determine if one is followed by the other, but if they belong to different cases, their relation will not be relevant to the analysis. Weijters and Medeiros (2006) proposed three different mining perspectives that were adapted to the approach of this work (Section 4.3 depicts an example with these perspectives):

- The process perspective focuses on the control flow, i.e., the ordering of activities, in order to discover a good characterization of all possible paths.
- The organizational perspective focuses on the originator field, i.e., which performers are involved in performing the activities and how they relate with each other. This allows to structure the organization by classifying people in terms of roles and organizational units or to establish relations between individual performers (i.e., build a social network (Song and Aalst, 2008)).
- The case perspective focuses on properties of cases which can be characterized by their path in the process or by the originators working on a case. Nevertheless, cases can also be characterized by the values of the respective data elements.

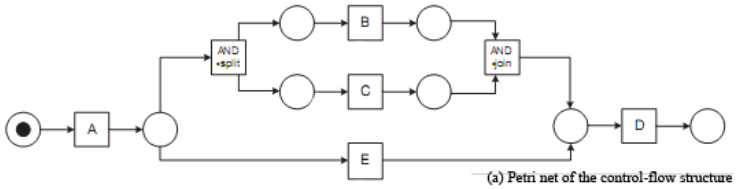
The algorithm has the following steps:

1. Dependency Graph elaboration, where the dependency relations between events are identified in order to avoid, for example, incorrect evaluation of noise where low frequent patterns can be present. This may be assisted with three thresholds:
   (a) Dependencies threshold, supporting which of the dependency relations between activities may be accepted;

(b) Positive observations threshold, aiding in the acceptance of dependency relations by measuring their frequency;

(c) Relative to best threshold, indicating the best dependency measure that has been detected.

2. AND/XOR-split/join and non-observable tasks (since they are not present in the event log the difficulty is aggravated). A translation of the generated Petri Net to a Causal Matrix is straightforward (Medeiros and Weijters, 2005; Medeiros et al., 2005) by setting for each activity an input and an output expression, as in the example depicted in Figure 3.1.



(a) Petri net of the control-flow structure

ACTIVITY	*INPUT*	*OUTPUT*
A	$\emptyset$	$(B \vee E) \wedge (C \vee E)$
B	A	D
C	A	D
D	$(B \vee E) \wedge (C \vee E)$	$\emptyset$
E	A	D

(b) Casual Matrix translated from (a)

Figure 3.1: Example of a Petri net and the respective translation to the Casual Matrix (Weijters and Medeiros, 2006)

3. Mining long distance dependencies, if the choice between activities is not determined within the respective node in the process model, further mining on the process model may be needed. A long distance dependency construct, where choices depend on previous ones, may require not solely a mining approach based on binary information, but also implicit dependencies detection (Wen et al., 2006) which may indicate strong sequence's relations of particular activities.

In the practical situation, with event logs with thousands of traces, low frequent behavior and noise, the Heuristics Miner allows to focus on all behavior in the event log, or only on the main behavior, showing a strong robustness to noise (Weijters and Medeiros, 2006).

3.1.3 Genetic algorithm

Genetic algorithms Medeiros and Weijters (2005); Medeiros et al. (2007) are adaptive search methods mimicking the process of evolution and therefore they are very robust to noise. Starting with a population of individuals, where each one is assigned a fitness measure indicating its quality Medeiros et al. (2005), if one individual is seen as a process model, its fitness may be seen as a function evaluating how well it reproduces the behavior in the log. Evolution is achieved by renewing the population with the fittest individuals and generating new individuals by combining parts of two or more individuals (*crossover* genetic operator) or by random modification of the individual (*mutation* genetic operator).

The crossover operation creates new individuals (off-springs) based on the fittest individuals (parents) in the current population. The goal is to generate a fitter population element based in the recombination of useful material in one of the parents. The mutation operation executes a random change on the details of a population element in order to insert new useful material in the population.

3.2 Sequence Clustering

In practical applications with large event logs, applying the above techniques may generate very complex models. In order to understand these models it is necessary to apply preprocessing techniques, such as sequence clustering, to study different behaviors separately (Veiga and Ferreira, 2009). With the sequence clustering algorithm, sequences are partitioned into a number of clusters or groups of similar sequences, each with an associated Markov chain (Silva and Ferreira, 2009). Each chain has a set of states with the respective transition probabilities between those states, where each state has a unique dependence with the previous state (first-order Markov Chain (Cadez et al., 2003)), or has multiple dependencies to a set of previous states (higher-order Markov Chains (Girolami and Kabán, 2005)).

The goal of sequence clustering is to extract the sequences of an event log (relating to specific cases) into groups where each group can be analyzed separately, thus generating simpler process models (Ferreira et al., 2007). This is particularly useful in the event logs of the OutSystems Platform in order to detect usage patterns within an application, or even within several similar applications that share the same webpages.

At the first stage, the sequences are randomly distributed into the clusters, to be later re-assigned to the cluster that can produce them with higher probability. Meanwhile the cluster models are re-estimated (within the Markov chain and supported by the transitions probabilities). After this first iteration, re-assignment of sequences and re-estimation of the cluster models are repeated until the algorithm converges, resulting in a set of Markov models that describe the behavior of each cluster.

Therefore within this context, sequence clustering allows to mine human activities, as presented in the case study of Ferreira (2009). By recording all the activities taking place in a daily work of a team (e.g.: interacting with an OutSystems' application) it is possible to sort out different user behaviors, thus providing insight into the underlying structure of those behaviors. In the case study of this work (Section 5.2.4) sequence clustering was applied to study these behaviors.

3.3 Social Network Miner

The tasks in an event log are commonly associated with their performers, i.e. the people or entities initiating or completing a given task. When this information is combined with the concepts from workflow management, it is possible to discover and analyse social networks (Aalst et al., 2005). If the events are ordered in time the log allows the inference of causal relations between activities and the corresponding social interaction. The hand-over of work from one performer to the next one can be discovered by using these inferences. The strength of their relation can be measured by the amount of times that work was transferred between them.

Social Network Analysis (SNA) refers to the collection of methods, techniques and tools in sociometry aiming and visualization at the analysis of social networks (Bernard

et al., 1990). There are several metrics presented in Aalst et al. (2005) in this context based on:

- Causality, depending on performers passing cases to one another.
- Joint cases, ignoring these causal dependencies to simply record how many times two individuals are performing activities for the same case. Cases can be characterized by their path in the process or by the originators (the individuals performing the tasks) working on a case (Song and Aalst, 2008).
- Joint activities, where the relevant issue is the frequency of users performing specific activities.
- Special events, assuming that events may not correspond only to the ending of an activity but also to its beginning, reassignment, scheduling, etc..

These metrics allow to derive sociograms which in turn enable the application of SNA techniques, such as betweenness, closeness, power, etc. Ethical and legal issues play a significant role in the practical application of process mining in general and SNA analysis in particular, since the behavior and privacy of the performers is exposed.

3.4 Performance

There is a constant pressure to improve the performance and efficiency of business processes. Fine-grained monitoring facilities are required in order to achieve this, such as Business Activity Monitoring (BAM), Business Operations Management (BOM), and Business Process Intelligence (BPI). However, the functionality offered by tools such as Cognos[1] and Business Objects[2] (Power and Kaparthi (2002) discuss more vendors of web-based Decision Support Systems and refer to DSSRecources.com for similar products) is limited to simple performance indicators such as flow time and utilization, but most of these systems do not focus on causal and dynamic dependencies in processes and organizations (Aalst et al., 2007b).

While organizational monitoring measures the organizational efficiency (e.g. idle times, workload analysis etc.), technical monitoring is used for performance measurement (e.g. system response time, system load etc.). Performance metrics can be extracted from workflow logs by tools such as the PISA tool (Muehlen and Rosemann, 2000). If the workflow log contains a timestamp for each event (timed logs), this information can be used to extract information about the performance of the process, such as bottlenecks in the process, flow times (mean, min and max), probabilities of occurrence of specific paths and, consequently, the performance of employees associated to the process (Aalst and Dongen, 2002).

3.5 ProM Framework

The techniques presented in the previous sections were brought together through a large framework known in the Process Mining field as ProM (Dongen et al., 2005; Aalst et al., 2007a). Allowing the interaction with a great number of plug-ins, which are nothing less than implementations of algorithms useful in process mining area, the

[1] www.cognos.com
[2] www.businessobjects.com

ProM Framework requires a standard structure for log files to use those same plug-ins. An event log under the ProM Framework has an XML structure using a specific schema known as WorkflowLog. To distinguish it from an ordinary XML file, the MXML file format was adopted with the respective Data Meta Model in Figure 3.2.

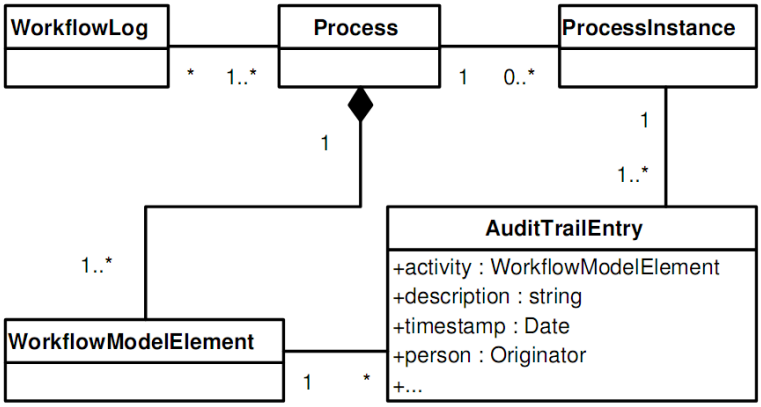


Figure 3.2: Mining Data Meta Model (Dongen and Aalst, 2005)

At the left side of Figure 3.3 is depicted an MXML with two process instances where each audit trail entry refers to one event that happened at a given point in time and is identified by the name inside the workflow model element. It should be noted that the timestamp and the originator of the event can be omitted, as there are ProM plug-ins to perform the analysis based only on the activity name and the event type (for example, at the right side of Figure 3.3 is the flow produced by the analysis of the MXML by the Heuristics Miner plug-in).

As it will be seen in Chapter 4, the relevant event logs provided by the OutSystems Platform are the ones related to the webpages that the users visit when they interact with the applications. These are screen logs, due to the fact that each webpage is represented by a screen transition, which in turn, is associated to one or several actions.

3.6 Summary

The process mining techniques referenced in this chapter, as well as the ProM Framework, were the base line to start this dissertation. Basically, the event logs of an application must be transformed to a common document format and then can be analyzed by different techniques, each generating a different kind of model.

Chapter 4 presents an approach that was based on the concepts of the process mining techniques and the framework mentioned in this chapter in order to illustrate a more specific analysis and results related to the context of the event logs of OutSystems' applications.

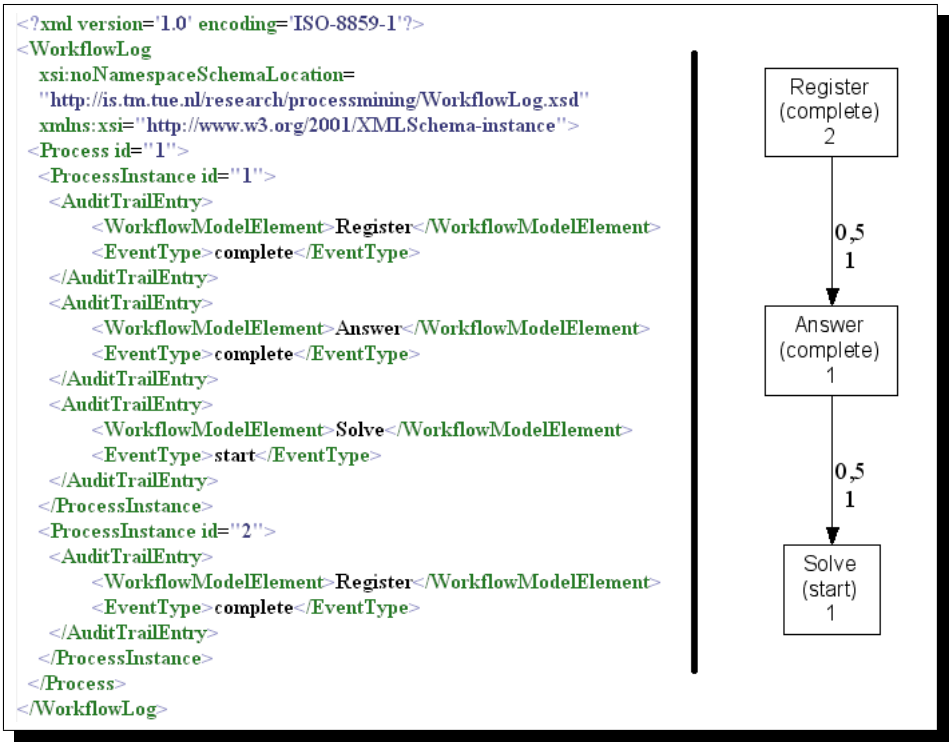


Figure 3.3: MXML document example (left) and the respective flow generated by the Heuristic Miner plug-in (right)

Chapter 4

Proposed Approach

Although based on the ProM Framework, the approach presented in this chapter is the result of an independent development for achieving additional knowledge through the use of specific techniques to analyze event logs related to the OutSystems' applications. In order to provide a better understanding on how these logs can be associated to the framework, table 4.1 illustrates the format of the screen logs.

ProM Framework Fields	Screen Log Fields
Process	Espace_Id (Application's Identifier)
Process Instance	Session_Id (Application's session)
Workflow Model Element	Screen (action referring to a webpage)
Timestamp	Instant (time and date of the visit)
Originator	User_Id (user's identifier)

Table 4.1: Fields comparison between OutSystems applications' logs and ProM Framework

4.1 Application Monitoring and Improvement

Generally, applications developed in the OutSystems Platform have the ability to be rapidly adapted by manual and semi-automatic interventions in the Service Studio to the characteristics of an ongoing business context, as Chapter 2 detailed.

Hence, by applying process mining techniques to the results of the end-users' interactions, one can contribute to the analysis and redesign of those applications, and improve them to optimally meet the needs of the business context that they are supposed to handle, as depicted by Figure 4.1.

The Agile Platform in its Service Center provides a monitoring section with several webpages where the user can access online logs of many types of events. But which of those types of events have information about the user's deployed application? Which of them may allow us to know how the user interacts with its own application? The next section presents a multitude of methods to gather the event logs and elucidates which information is relevant in those logs.

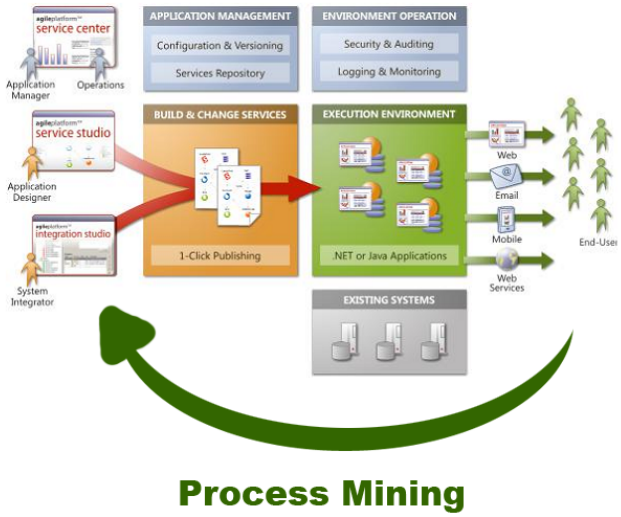


Figure 4.1: Process Mining of OutSystems Platform's applications

4.2 Sources of Event Logs

There are several possibilities for gathering run-time data from OutSystems' applications. One way is to gather information from the local machine where the application is published (e.g., platform's database). Another possibility is to use the monitoring tools provided by the platform itself (through the Service Center) which can be accessed locally or remotely. Although the monitoring tools are limited, it was clear that the Screen Log contained what was required, as table 4.1 depicted.

Service Center Screen Logs have some fields of interest, but more can be found through other methods, as shown in Figure 4.2.

The Screen information is also stored in the respective table on the SQL Server database, but access to the server must be granted. This table has all the fields of interest presented on table 4.1, such as the application's identifier, the time and date of the visit,the user's identifier and, most importantly, a screen action (e.g.: LoginPage, Home, Project_List) which partly reveals what the Web Screens of the application refer to.

The IIS logs enable a different alternative. The fields of interest can be selected through the IIS Manager. The session's identification can be retrieved, allowing to establish process instances in the MXML. The originator of a specific event is present whenever the event itself is not fully automated ("time_handlers"), but they lack what defines the WorkflowModelElements in the MXML. The screen action is not evident and far from the detail of the one stored in the database's table.

Another method consists in the use of an OutSystems application to access the platform's own logs. This application, the LogScreen application (Figure 4.2), grants a read only access to the same data that a database query would be able to retrieve. Therefore it was considered the best and secure way to achieve the log that has the Screen data.

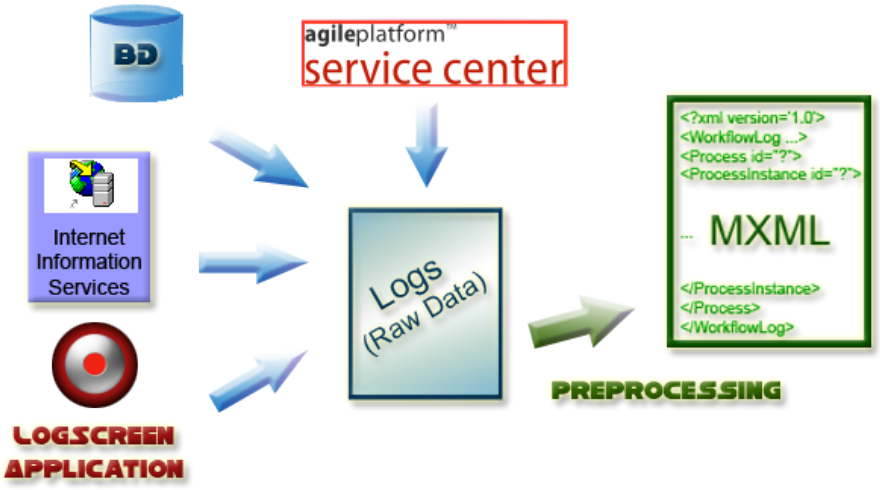


Figure 4.2: Information about user-interaction with an OutSystems' application can be gathered from several sources. By preprocessing the raw data (OutSystems' logs), it is possible to build a MXML document.

IIS logs have the advantage of easily gathering the relevant fields and of providing a quick access to them. But they also have the disadvantage of not relating the webpage visited to the screen action that the user invoked. Even with a great amount of attributes related to the an event, this results in a poor audit trail entry, like the one illustrated in Figure 4.3. The workflow model element is not distinguishable from the following ones, since they usually assume the same name even when the screen action is different. There is no reference to the originator, thus making harder establishing relations with other events.

```
<Process id="DEFAULT">
  <ProcessInstance id="-"><AuditTrailEntry>
    <Data>
      <Attribute name="ComputerName">K8DF</Attribute>
      <Attribute name="sIP">::1</Attribute>
      <Attribute name="csMethod">POST</Attribute>
      <Attribute name="uriQuery">-</Attribute>
      <Attribute name="Port">80</Attribute>
      <Attribute name="cIP">::1</Attribute>
      <Attribute name="csHost">localhost</Attribute>
    <Attribute name="timetaken">783</Attribute></Data>
    <WorkflowModelElement>
    /ServiceCenter/ServiceStudio.asmx</WorkflowModelElement>
    <EventType>unknown</EventType>
    <Timestamp>2009-12-22T14:53:23</Timestamp>
    <Originator>-</Originator>
  </AuditTrailEntry>
```

Figure 4.3: IIS Log when converted to MXML

4.2.1 LogScreen application

The LogScreen application was specifically developed to retrieve OutSystems' screen logs, which are the result of the users' interaction with the applications published on a specific server. Once the LogScreen is published in this server, it accesses the database through an SQL query (depicted in Figure 4.4) and places the result on a table designed in the Service Studio, resulting in an HTML document with all the relevant data of the events that occurred in the server's applications.

```
SELECT

ENLog_Screen.[TENANT_ID] , ENLog_Screen.[INSTANT] ,
ENLog_Screen.[DURATION] ,

ENLog_Screen.[SCREEN] , ENLog_Screen.[SESSION_ID] ,
ENLog_Screen.[USER_ID] ,

ENLog_Screen.[ESPACE_ID] , ENLog_Screen.[MSISDN] ,
ENLog_Screen.[SCREEN_TYPE] ,

ENLog_Screen.[CYCLE] , ENLog_Screen.[EXECUTED_BY] ,
ENLog_Screen.[SESSION_BYTES] ,

ENLog_Screen.[VIEWSTATE_BYTES] , ENLog_Screen.[SESSION_REQUESTS] ,
ENLog_Screen.[ACCESS_MODE]

FROM {Log_Screen} ENLog_Screen
```

Figure 4.4: LogScreen's SQL Query

In the Service Studio only one web screen has been made to show the result of this query. This web screen is simpler than the one presented in Section 2.3 because it only presents the table with the headers linked to the attributes of the database's table. The respective screen events that are stored in the database's table are directly transfered to the rows of an HTML table.

Figure 4.5 presents the generated HTML document which can easily be converted by an XQuery (an XML Query Language Boag et al. (2007)) to an MXML document. The column headers of the table that have relevance to the analysis refer to the fields depicted in table 4.1. The only field that is not mentioned in this table refers to the Duration column which presents the time taken to load the webpage.

4.2.2 Analysis

The following phases summarily describe the steps of the proposed approach for the discovery of behavioral models of applications developed in the Agile Platform (Figure 4.6):

1. Data Source identification: considering the storing data structures defined by OutSystems, the only source known as relevant to observe and study the user's application behavior is located on the Screen data structure. Here data is updated, in real time, when a user interacts with the running application.

Tenant_Id	Instant	Duration	Screen	Session_Id	User_Id	Espace_Id
47	2010-06-19 19:49:39	15	Contact_list	sqeir055f0jsij45csxbvyyg	0	52
47	2010-06-19 19:49:46	0	Contact_Show	sqeir055f0jsij45csxbvyyg	0	52
47	2010-06-19 19:49:47	0	Contact_edit	sqeir055f0jsij45csxbvyyg	0	52
47	2010-06-19 19:49:51	15	Contact_list	sqeir055f0jsij45csxbvyyg	0	52
47	2010-06-19 19:49:59	15	Contact_edit	sqeir055f0jsij45csxbvyyg	0	52
47	2010-06-19 19:50:04	0	Contact_list	sqeir055f0jsij45csxbvyyg	0	52

Figure 4.5: LogScreen Application webpage

Figure 4.6: Phases in analysis for validating tests and real-life examples

2. Preprocessing: data may be retrieved from either the OutSystems database, either the Service Center's Screen Log, or via IIS logs or LogScreen application. Each option must have a custom preprocessing and may need a supplement option or supplement definitions to provide all aspects and relevant data for a given event.

3. Application of process mining techniques: after achieving a proper MXML document, process mining algorithms may be applied via ProM plug-ins, or via an independent implementation of a chosen algorithm.

4. Observation and analysis: graphs in early stages may not produce a useful process model. However, by adding or complementing event logs with information related to user's visits, one may achieve a clearer understanding of the produced model.

5. Model and behaviors comparison: the original model developed in Service Studio may be hard to understand, since different webpages of the same application may be separated through various Screen Flows. By comparing these flows with the actual flow that one or more users have followed, a perception on how the application's Screen Flows are connected may be achieved.

For validating the developed solution, Section 4.3 presents an example of a simple application, while in Chapter 5, with a higher complexity level, it is presented an application used in a real case scenario.

4.3 Application Example

The following subsections focus on different ways to extract and display the relevant information from real event logs. A simple web application for managing contacts will be shown to exemplify the visualization possibilities of the information embedded in the retrieved log. This application development was similar to the one presented in Section 2.3. In fact, it was the result of one of the first OutSystems tutorials for creating tables and list data designed for an earlier version of the Agile Platform (version 4.2). Although it has a few more webpages than the former application, the structure is almost the same - it has a web screen presenting a list (that in version 5.1 could be the

result of importing an excel worksheet), an option to show each Contact's details on a new webpage by clicking a name's link of the list, an option for editing and deleting the Contact, and similar options to manage Categories (another entity related to the Contact's entity, as the Note entity is related to the Employee entity from example of Section 2.3).

The interaction sample that can be seen in Figure 4.7 consists of an updating of an existing contact's information followed by the creation of a new contact in the contacts list.

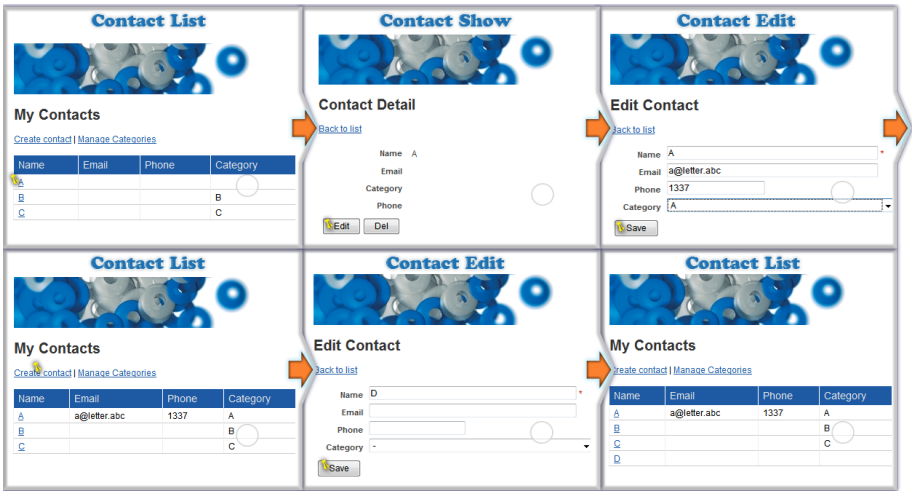


Figure 4.7: Interaction sample of a simple contacts manager OutSystem's application

This interaction was recorded in the database and the LogScreen application was able to generate the HTML document shown in Figure 4.5. Once the log is shown in this form, one can proceed to its preprocessing. The MXML depicted at the left side of Figure 4.8 is the result of this preprocessing.

4.3.1 Control flow

When a user visits a set of webpages from a certain application developed in the OutSystems Service Studio, a trace is left by all of his visits. This trace may be rebuilt by connecting the webpages in the order that the user has visited them. This is a Control Flow due to the dependency on the navigation choices that the user has made. Although this may seem generic at a first glance, different analysis may be produced with the same concept, such as:

- General control flow – where one single graph is produced with all the sessions and respective users that intervened in the application;
- Control flow separation by Process Instances – where multiple graphs are produced depending on the different sessions contained in the event log;
- User's control flow – where one graph per user that interacted with the application is produced.

The right side of Figure 4.8 depicts the control flow based on the MXML at its left side. In this example there is one unique session (the process instance identifier is the

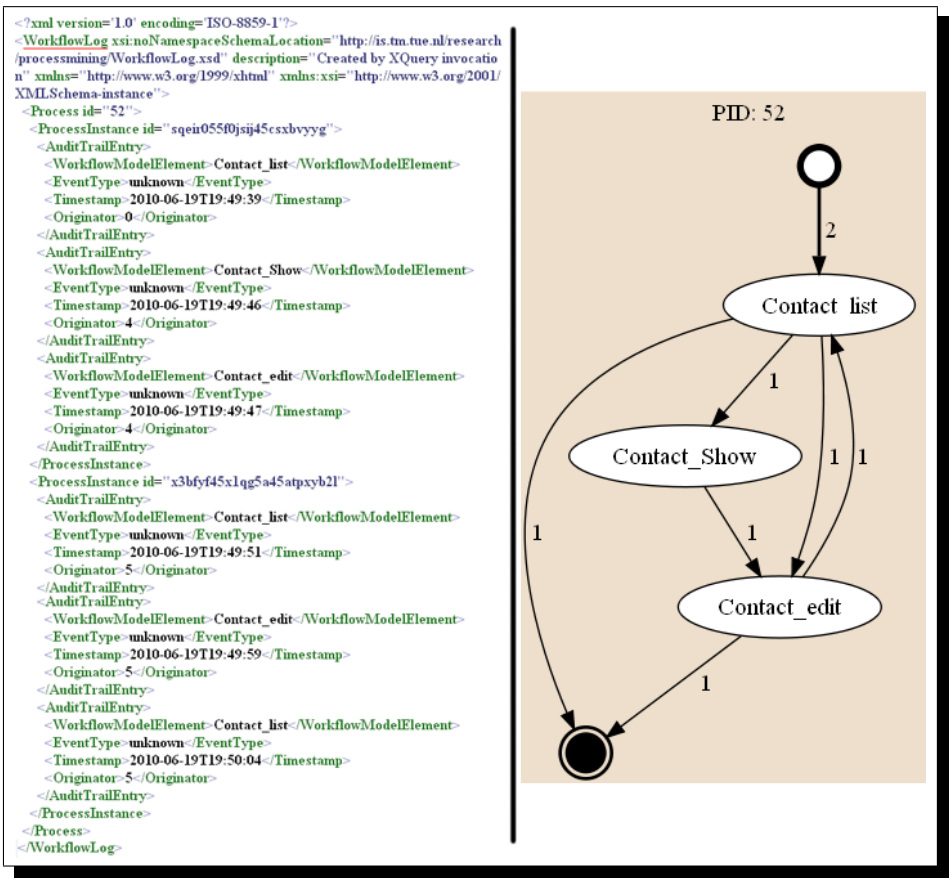


Figure 4.8: MXML generated from the LogScreen webpage (left) and the respective control flow (right)

OutSystems' session identifier "wxwvmg45ked4s325up34k1qu" which is, by default, a unique string of characters and numbers) and only one user (identified by the originator 0), which would produce the same flow on different analysis.

A different interaction with multiple users and different sessions can be simulated with the same MXML as depicted in Figure 4.9 (a). In this case, one user (originator 0) would enter the initial webpage (Contact_list) and another user (originator 4) in the same session would access one contact of the list and would edit it and save the new information (visiting only the Contact Show and Contact edit webpages).

On a different session, another user would just create a new contact accessing the edit and the contact's list webpages. Therefore the resulting control flow can be subjected to a session differentiation, depicted in Figure 4.9 (b), as well as a distinction of users' interactions through different graphs, as shown in Figure 4.9 (c).

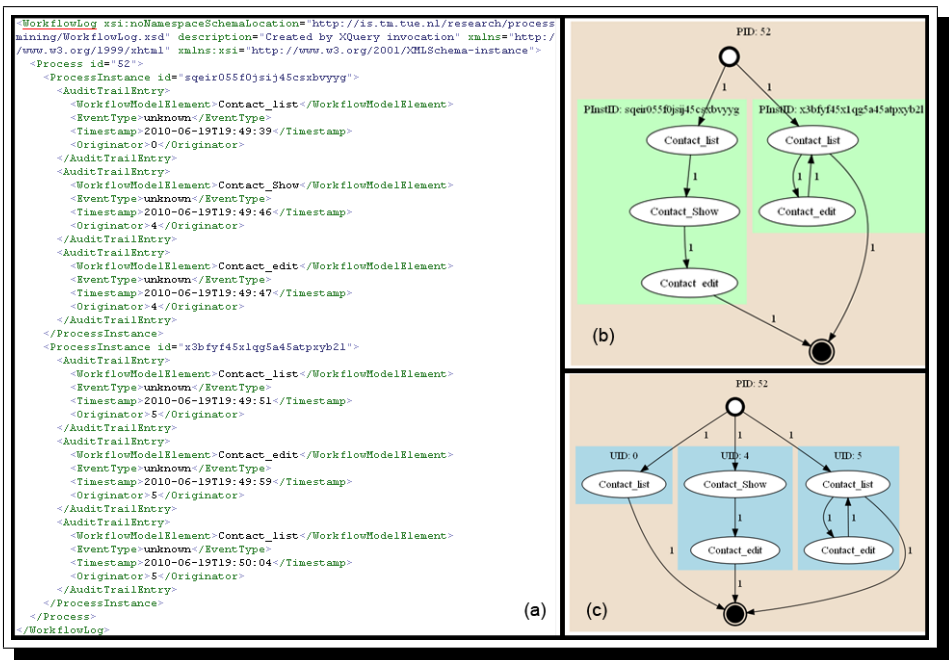


Figure 4.9: Multiple users (c) and sessions (b) can be distinguished by analysing the same MXML (a)

4.3.2 Social Network

The user's control flow gives a perspective over what each user visited in the entire set of the registered sessions. Although this seems quite complete for explaining a user's behavior, more users may have been on the same session interacting with the same or even different webpages [1] which affected the user's navigation. So, in order to obtain those who influenced a particular user's behavior in one certain session, the Social Network connects the users that visited a certain webpage in the same session. This way, an overall users' interrelation may be established by their interactions with the application.

The previous example depicted in Figure 4.9 shows that two users shared the same session and another one has visited two webpages in a different session of the same application. Thereupon, the network resulting from the log has only one connection between the two users that shared their session, as can be seen in Figure 4.10(left side).

Nevertheless, if multiple users share the same sessions of the same application, a complex network may be presented. Instead of simple unitary connections, referring to one session only, if several sessions are taken into account, it is probable that those users interact again, thus increasing the value of their connection. In other words, a connection's number depends on the sum of the sessions that two users interacted with each other. If they only interact in one session this number is 1, if they interact in three distinct sessions then the number is 3, and so on. This is depicted in the example of Figure 4.10(right side) as follows: user 0 and user 1 interacted in two different sessions and also interacted with user 2 in three different sessions, while user 1 only interacted

[1] The system usually interacts with the user in the same session, and it is viewed as the user with the identifier 0 by the OutSystems Platform.

with user 2 in two different sessions, thus user 1 was the one that interacted the less among these 3 users.

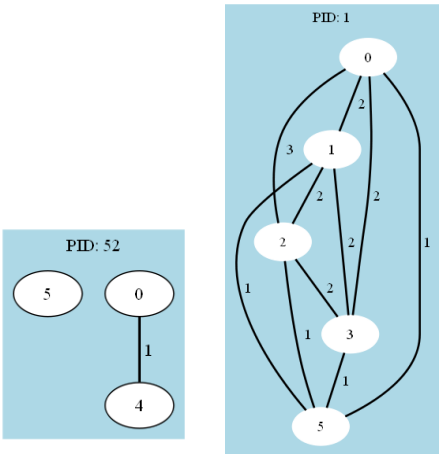


Figure 4.10: Social Network from the example of Figure 4.9 (left) and of an application with 5 users interacting in various sessions (right)

4.3.3 Performance

After acquiring the previous sections' flows or networks, a performance analysis can be done by comparing the time spent by users in each webpage. Each webpage, viewed as a node, may be visited by multiple users in the same or in different process instances (corresponding to different sessions).

Figure 4.11 depicts the analysis made on the previous example (Figure 4.9).

Name	Process	Visits	Loops	Users	NumberOfUserVisit	UsersVisits
Contact_list1	52	3	0	0 5 5	1st 1st 2nd	1 2
Contact_Show1	52	1	0	4	1st	1
Contact_edit1	52	2	0	4 5	1st 1st	1 1

Instance	EntranceTimes	ExitTimes	MinDuration	MaxDuration	AvgDuration
sqeir055f0jsij45csxbvyyg x3bfyf45x1qg5a45atpxyb2l x3bfyf45x1qg5a45atpxyb2l	19-06-2010 19:49:39 19-06-2010 19:49:51 19-06-2010 19:50:04	-User Swap to 4- 19-06-2010 19:49:59 -End Of File-	00:00:08	00:00:08	00:00:08
sqeir055f0jsij45csxbvyyg	19-06-2010 19:49:46	19-06-2010 19:49:47	00:00:01	00:00:01	00:00:01
sqeir055f0jsij45csxbvyyg x3bfyf45x1qg5a45atpxyb2l	19-06-2010 19:49:47 19-06-2010 19:49:59	-End Of Instance- 19-06-2010 19:50:04	00:00:05	00:00:05	00:00:05

Figure 4.11: Performance analysis on the example from Figure 4.9

If the flow has at least two webpages' visits (even if it is the same webpage), one can determine for each node:

- The process or the application that it belongs to.
- The number of visits that were made to the node.
- How many loops were made in the node (transitions to the same webpage).

- Which users visited the node and their respective instances.
- The order that different users visited the node (NumberOfUserVisit column in Figure 4.11).
- The total number of visits made by each user.
- Entrance timestamps, given directly by the given directly by the timestamp of the event.
- Exit timestamps, given by the next node's timestamp in the flow, albeit exceptions occur when:
  - The MXML file has been read completely, indicating an "end of file" message, as can be seen in the Contact_list node from Figure 4.11).
  - The next process is different from the current node's process, so a different application is about to be used. An "end of flow" message appears in these cases.
  - The next node belongs to a new instance, therefore indicating that the current session has ended and the user(s) may have ended a specific interaction with the application (as can be seen in the Contact_edit node from Figure 4.11).
  - The next node's user is different (there is a swap in the same session to another user). The first two WorkFlowModelElements of Figure 4.9 (a) and the social network (Figure 4.10) depicts this case by considering the interaction in the same instance of user 0 and 4. Therefore Figure 4.11 in the Contact_list node has no ExitTime for user 0 albeit it has an indication that another specific user continued to visit webpages inside that instance, thus occurring a swap. By comparing Figure 4.9 (b) and Figure 4.9 (c) there is a proof that this occurs.
  - The next node is the same as the current one, hence a "loop" message appears, which indicates that the exit timestamp is the entrance timestamp of the next line below.
- An entrance and exit timestamp subtraction, giving how much time the users have spent from the entire amount of visits on the node, namely:
  - the visit with the shortest duration, which is given by the minimum duration column from Figure 4.11.
  - the average duration of a visit to the node ("AvgDuration" column from Figure 4.11).
  - the visit with the longest duration, which is given by the maximum duration column from Figure 4.11.

Figure 4.12 depicts an example where 3 visits were made to the same node, therefore the calculations mentioned above consist of the sum and comparison of three durations. Other analysis based on this information may be performed, such as determining the shortest or longest path (considering time or number of webpages visited) of a process or instance, which users are predominant in the use of a certain application, which ones require always the assistance of another user to complete their session, or which

Figure 4.12: Durations example (time in minutes)

interact with the greater or lesser amount of users in different sessions (as social network analysis of Figure 4.10 depicted), and so on. The next section will focus on how the analysis and model extraction was achieved.

4.4 Implementation

Both the extraction of a control-flow model and the performance analysis described in the previous section are based on a MXML document. This document is created by applying an XQuery, such as the one depicted in Figure 4.13, to an HTML document provided by the LogScreen application (as shown in Figure 4.5).

The XQuery accesses directly to each one of the table's rows and builds immediately an AuditTrailEntry XML element, assigning the webpage's name, the timestamp and the user's identification to the respective field of the ProM Framework (according to 4.1). After acquiring a list of these entries, a new loop is made to verify and place each one of them on the block they belong, thus building the Process Instance and Process XML elements. These blocks when placed inside the WorkflowLog XML element allow the creation of a well-formed MXML document containing all the information of the desired event log document.

The source of the event log is chosen over a multitude of possible locations. The application was developed for firstly to ask for one of these locations, whether from the local OutSystems server (localhost - community edition), whether from a server on the World Wide Web, whether from a file saved on the local disk. Any source that doesn't use a local file must be an URL pointing to the LogScreen application published in the respective server. If the "new server" source is chosen then an option for accessing the desired application's event log is made available by inserting the identifier of that same application at the end of the URL, as Figure 4.14 demonstrates.

After selecting the HTML document's source, the document itself and an option for applying the XQuery to deal with the preprocessing are presented, as depicted at the top of Figure 4.15. With the following steps depicted on this figure, the graph model and the performance information are obtained.

The XQuery is invoked through a Java application that transforms the HTML document to the MXML document. Once this step reached, one may choose to build the social network or the control flow based on this document. The former produces a graph of the users' connections inside the same session, as explained in Section 4.3.2,

Query editor Execute

```
let $doc := gti:htmldoc("http://ursula.tagus.ist.utl.pt/LogScreen")
let $tb := $doc//tbody
let $pis := distinct-values(for $tr in $tb/tr return $tr/td[5]/text())
let $prs := distinct-values(for $tr in $tb/tr return $tr/td[7]/text())
let $lines :=
 for $tr in $tb/tr[td/@class]
 let $tds := for $td in $tr/td return $td
 return
 <p pr="{$tds[7]/text()}" pi="{$tds[5]/text()}">
 <AuditTrailEntry>
 <WorkflowModelElement>{$tds[4]/text()}</WorkflowModelElement>
 <EventType>unknown</EventType>
 <Timestamp>{replace($tds[2]/text(), " ", "T")}</Timestamp>
 <Originator>{$tds[6]/text()}</Originator>
 </AuditTrailEntry>
 </p>
let $df := <WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd"
description="Created by XQuery invocation">{
for $pr in $prs
 return
  <Process id="{$pr}">{
   for $pi in $pis
   let $pitemp :=
          for $l in $lines
          where matches($pr, data($l//@pr)) and matches($pi, data($l//@pi))
          return $l//AuditTrailEntry
   return if(empty($pitemp))
    then () else (
    <ProcessInstance id="{$pi}">{$pitemp}</ProcessInstance>)
 }</Process>
}</WorkflowLog>
return x:serialize($df, <options output="LogScreen.mxml" encoding="ISO-8859-1" indent="yes"/>)
```

Figure 4.13: XQuery applied an HTML document as the one depicted in Figure 4.5

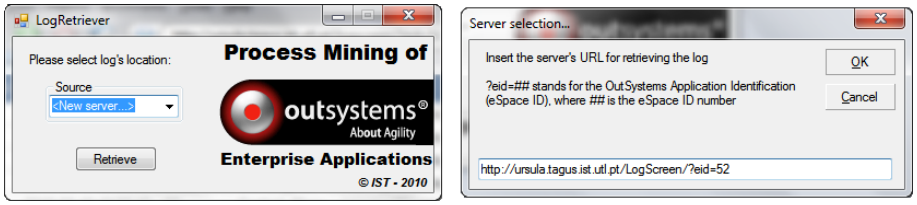


Figure 4.14: Source selection from the developed application

the latter is responsible for producing a flow graph depending on the options that are used to generate it, as referred in Section 4.3.1.

If the control flow option is selected another option is also enabled to specify a distinction of sessions or users, or a control over the flow's transitions, such as removing of loops to the same webpage or setting the maximum or minimum number of webpage transitions allowable in the graph.

The graph file has all the webpage connections, all the relevant information for separating users or sessions (according to what was selected in the previous step) and was intentionally made in one specific file format, the DOT format, from an open source graph visualization software [2]. The graphviz application is hence invoked for producing one image with the graph or graphs, or the performance analysis can be invoked instead. Or it can also be invoked after acquiring the image, because all relevant information was already processed.

[2] GraphViz – www.graphviz.org

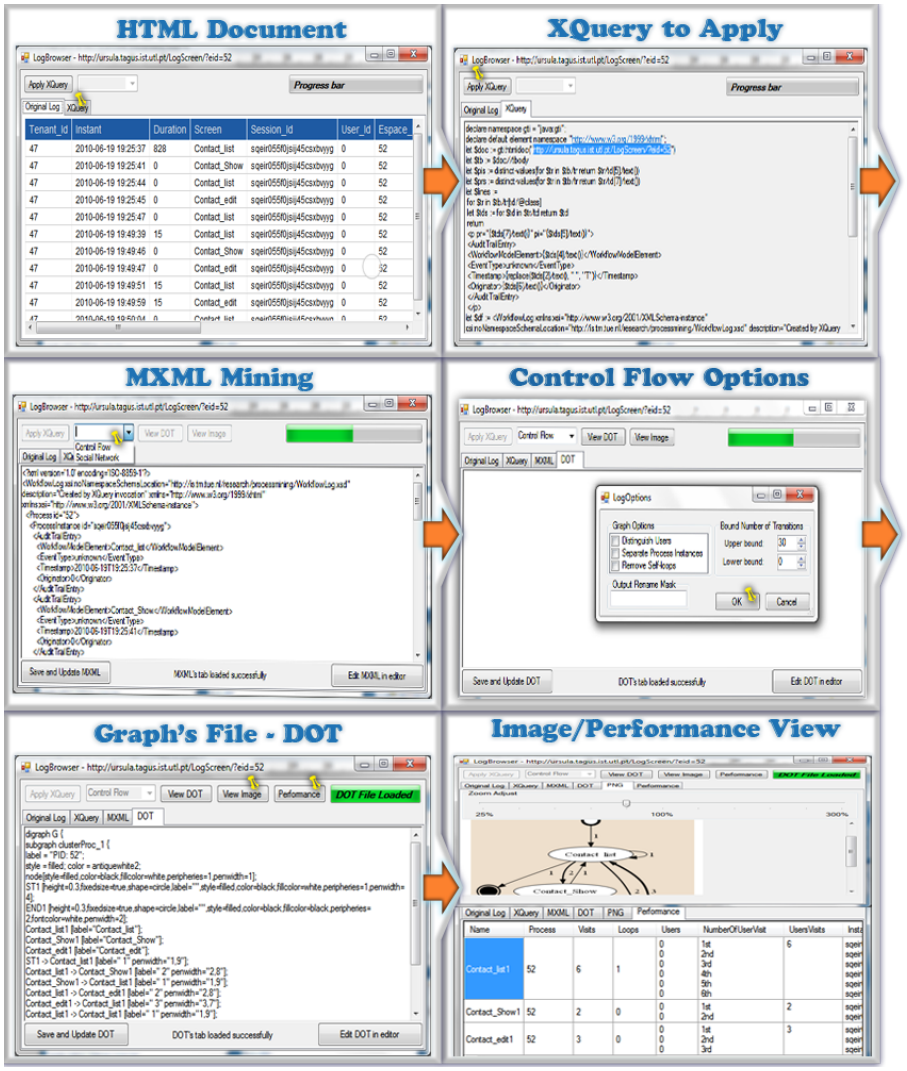


Figure 4.15: Steps for generating graphs and performance analysis based on a HTML document

In Section 5 a case study will be presented with a real-world OutSystem's application.

4.5 Summary

This chapter provided a description on how the event logs produced by the OutSystems platform can be analyzed through process mining techniques and adapted to this particular context. A brief explanation was given on how the developed application was implemented and what it allows to explore. In Chapter 5 a detailed example will fully cover every kind of analysis that is possible to perform with this application.

Chapter 5

Case Study

In the previous chapter, an approach to apply process mining techniques in the context of OutSystems applications was described. In order to complement the study carried out, this chapter provides a case study where this approach and the techniques described in Chapter 3 are applied to a real-world application – the Service Manager Application, detailed in the next section.

5.1 Service Manager Application

The Service Manager is a prototype application being actually used on an online OutSystems server which is based on a Ph.D research by Mendes (2010) on how to reduce the risks of implementing a catalog of IT services. This application allows to manage the life cycle of services provided by IT organizations and manage the requests associated to those services. It also allows to monitor the tasks performed by the registered users that complete those requests, hence providing a relevant degree of complexity for this study.

The application is based on the concept of service, where the service itself is considered to be an action due to its intangibility. A service can be as simple as an access granting to a particular functionality or providing support to a computer, application, printer, mobile phone, projects or training. Therefore the application purpose is to present a way to manage services considering the following characteristics of each one of them:

- A service is produced and consumed simultaneously by customers, therefore can not be inventoried, stored or reused.
- It is described in terms that are relevant to the customer.
- It responds to identified needs.
- It uses certain resources such as people, tools, information, infrastructures, etc.
- It follows a process.
- It adds value to those who benefit from it (the added value may be valid only for one agreed period although it may be renewed).
- It may contain tangible deliverables in their results.

A customer may require information, advice, amendments of specific services or access to an IT service, thus generating a service's request.

Thereby, in order to comply with these characteristics, the prototype application allows to:

- create/edit, publish, list and remove operations related to the life cycle of services;
- create/edit, comment, submit, budget, approve, reject, complete and close operations related to the life cycle of services' requests;
- create/edit, cancel and complete operations related to services' requests;
- create/edit, erase and resolve operations related to impediments in services' requests;
- manage services' requests in the context of projects and become integrated with skype.

The following sections provide an explanation on how the application works and its general structure, in order to later be analyzed by process mining techniques.

5.1.1 Usage

The application has several access profiles with specific permissions according to their roles. These permissions are reflected on the webpage's appearance and available operations. Figure 5.1 depicts an example of a user that is allowed to create requests - on the entrance webpage the left column presents a list of the requests in progress and the right column presents a search mechanism for those requests, a recent activity link list and useful links for that type of user.

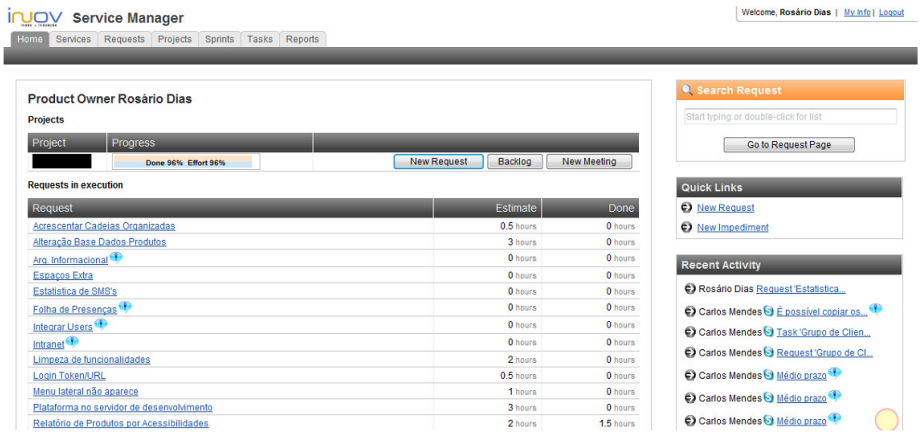


Figure 5.1: Home webpage of a user allowed to create requests

This means that different users with different permissions may have a completely alternate navigation through the Service Manager webpages due to the links that appear on those columns. In this way the levels of service are tailored to the needs of the user.

5.1.2 Development

As Figure 5.2 illustrates, this application is composed by several Screen Flows, where the "Common" Screen Flow is the starting point of the application. One user can follow through the Menu and navigate to different Screen Flows in the same session. This means that in order to understand a user's interaction related to all the transitions between webpages that were made in a specific session, one must open several of these Screen Flows and locate in them each webpage that was visited by the user.

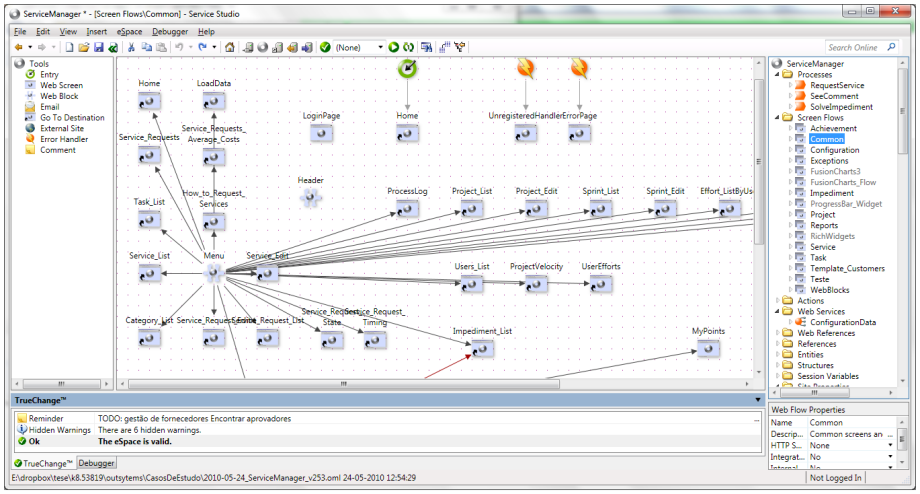


Figure 5.2: Service Manager in Service Studio - the common Screen Flow

Although this may seem practical at a first glance, the complexity usually increases in each new Screen Flow that is opened from the one containing the Menu, since the number of webpages and their connections also increase. In order to trace the entire interaction of a user, a great set of Screen Flows may be required to be opened and searched for specific webpages, which is a laborious and hardly practical study.

5.1.3 Retrieval of Event Logs

The LogRetriever, the developed application described in Section 4.4[1], was used to preprocess the event logs of the Service Manager Application. These event logs were transformed to the MXML format thus enabling the usage of the process mining techniques, described in Chapter 3, to analyse them and produce the respective results. This analysis is presented in Section 5.2. In Section 5.3, the LogRetriever algorithms are applied in order to probe further results and produce the type of graphs depicted in Figure 5.3.

[1]The manual was included in this document as Appendix: LogRetriever Manual

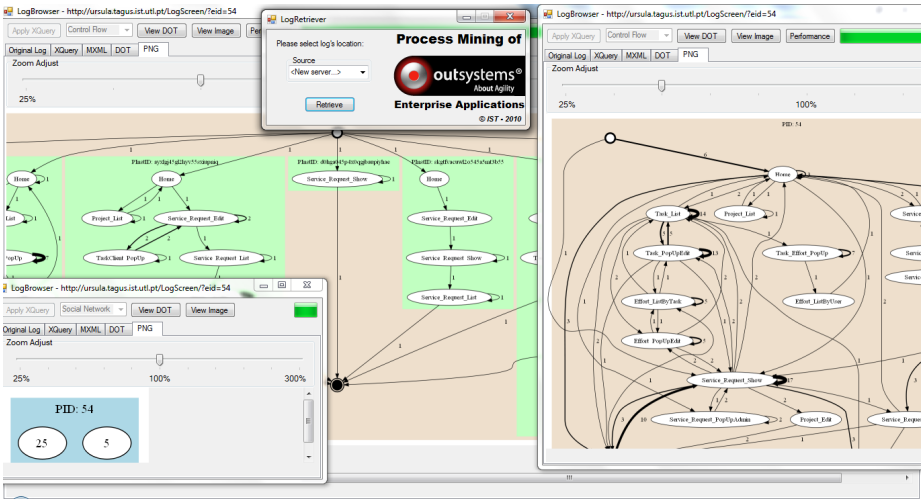


Figure 5.3: LogRetriever with several perspectives over the Service Manager event log

5.2 Analysis with ProM

The next subsections describe the application of selected techniques available in ProM to the events produced by OutSystems' applications.

5.2.1 Alpha (α) ++ algorithm analysis

The application of the Alpha ++ algorithm described in Section 3.1.1 revealed to be somewhat inconclusive. Generally it only depicts a small connection between some webpages while the rest are scattered without any relevant linkage. This happens because the alpha-algorithm is unable to capture some behaviors, most of them involving loops.

Figure 5.4 depicts a comparison between the ProM generated image and the one from the LogRetriever graph, showing that all the relations inferred between the webpages from the former are also present in the latter. Although the results seem to be insufficient to provide a relevant analysis of the event log, when using larger logs the connections start to reveal which webpages are closer to the beginning or end of the instances and also simpler flows give a perspective over the order of the webpage visits (from left to right).

5.2.2 Heuristics Miner analysis

On the other hand, the heuristics miner algorithm referred in Section 3.1.2, with the standard ProM input parameters, presents a much more detailed and resembling graph to the one achieved by the LogRetriever, as can be seen by comparing Figure 5.5 with Figure 5.6.

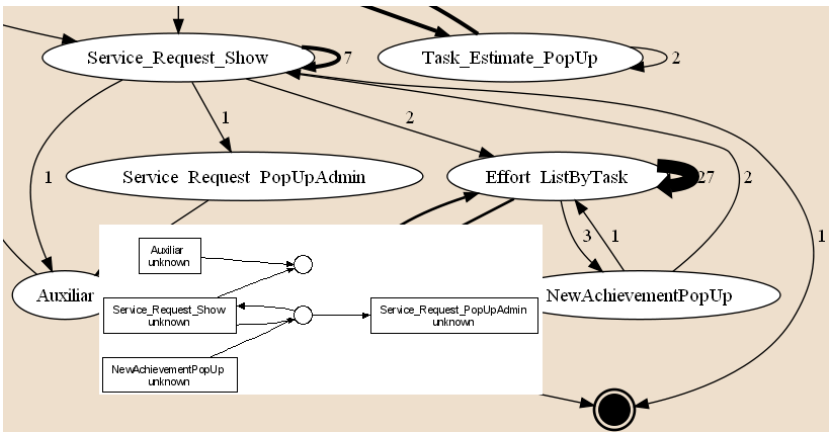


Figure 5.4: Alpha ++ algorithm output comparison

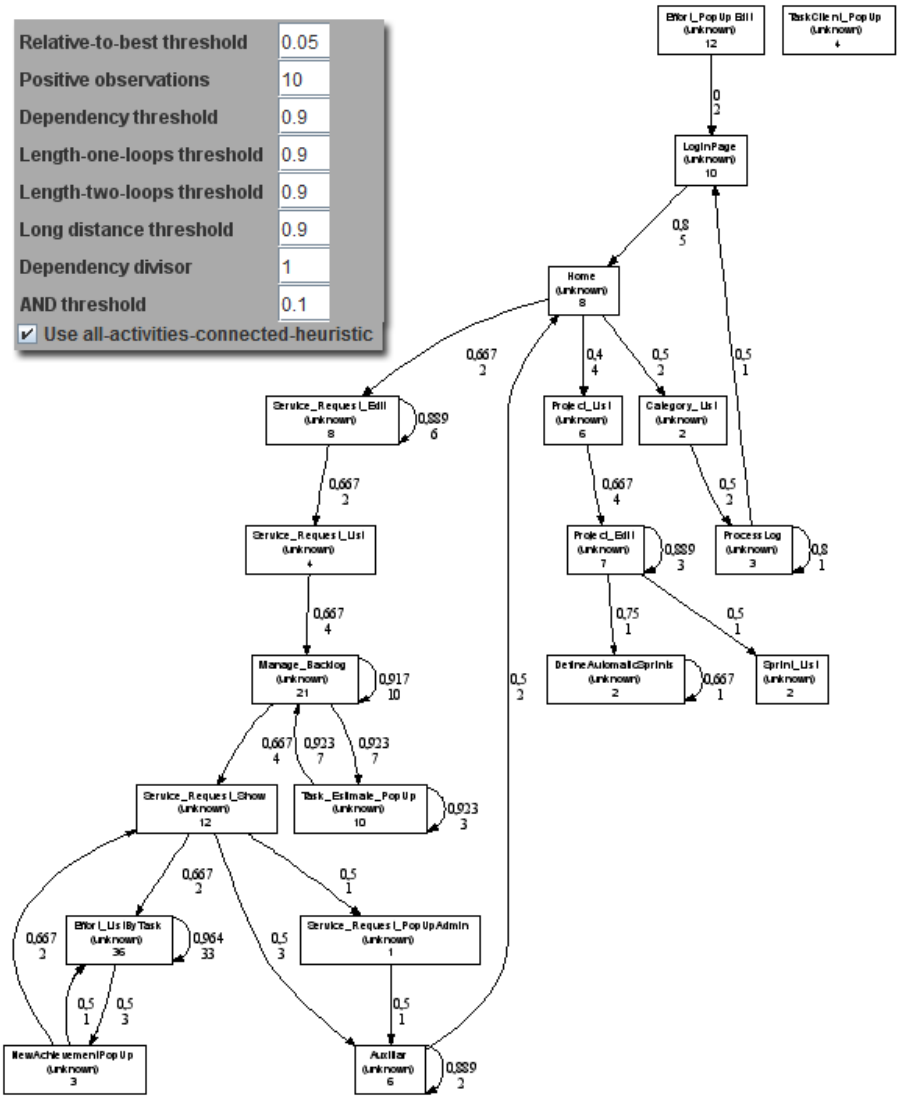


Figure 5.5: Heuristics Miner algorithm output (left) comparison with the general screen flow of LogRetriever (right)

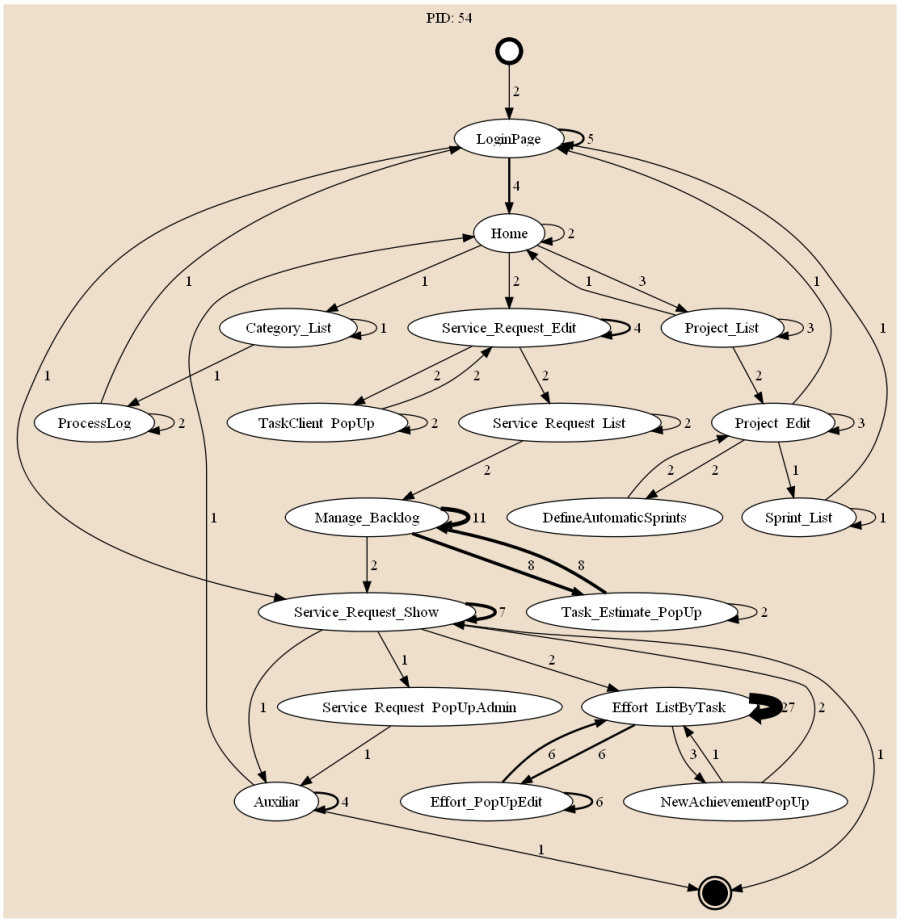


Figure 5.6: LogRetriever's General flow

The initial webpages are presented in the top of the graph, while the final ones are presented at the bottom. One can find that the causality between two given events of the process model is preserved by observing that they follow each other if the timestamps are compared. If one of these webpages is connected to another one from a different case (session/process instance), that will not be relevant to the first case. The LogRetriever uses a technique similar to Heuristics Miner to order the webpages in the graph. Both techniques provide sequences of webpage visits based on the several sessions (i.e., cases) present in the log.

The thresholds of the dependency graph are taken into account in the Heuristics Miner algorithm, thus eliminating a number of webpages connections that the application may allow to establish. These webpages may not seem to be dependent at all, but they may be connected due to less used links or shortcuts that the user takes to reach the webpage of interest. So they can be as relevant as the webpage connections that have a strong dependency, and may have interest to analyze to discover which links should be removed or added to each webpage. Therefore both analysis have their own value, relying on the objective to detect frequent visit patterns or vice-versa.

5.2.3 Genetic algorithm analysis

Figure 5.7 depicts the output of the genetic algorithm when the log from the previous sections is given as the input to the ProM plug-in.

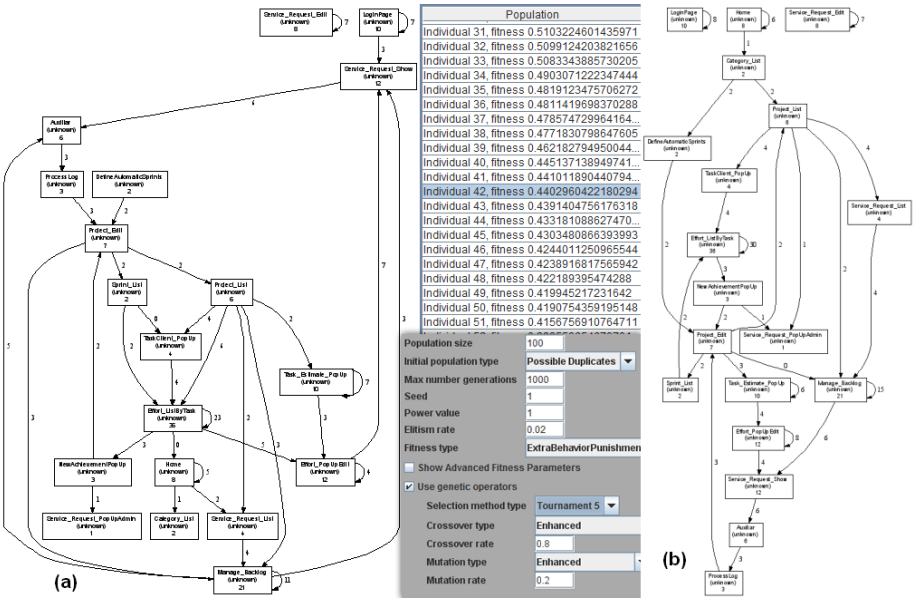


Figure 5.7: Genetic algorithm output based on a Service Manager's log. (a) relates to individual 42 and (b) relates to individual 0 (highest fitness)

The fitness of a process model depends on its completeness and precision from a behavioral perspective. A process model is complete when it can parse (or reproduce) all the event traces in the log and it is precise when it cannot parse behavior other than the traces in the log. The fitness metric punishes this extra behavior. Thus, if a set of individuals can parse all the traces in the log, the one that allows for less extra behavior will have a higher fitness value (Medeiros et al., 2007).

Figure 5.7 (a) depicts one behavior at the top that constitutes one entire process instance of the event log - the transition from the LoginPage to the Service_Request_Show which was not detected by the Heuristics Miner algorithm (Figure 5.5), thus revealing that even at lower fitness individuals, the algorithm tried to build a complete model.

Therefore the genetic algorithm generates interesting models reflecting user behaviors which are significant for a comparative analysis with other algorithms (e.g.: Heuristics Miner) that may not be able to capture all the hidden behavior in the event log. This is also one of the reasons why a specific tool was developed (LogRetriever) to directly extract models from event logs of OutSystems' applications

5.2.4 Sequence Clustering

The ProM sequence clustering plug-in extracts behavioral models for the different behavioral patterns found in the event log. There are thresholds available that correspond to the minimum and maximum probability of both edges and nodes that can be adjusted. Since the event log being analyzed is relatively simple, these were set to allow the largest number of states and transitions in the graph despite being frequent or rare.

When dealing with spaghetti models, these thresholds become useful to avoid to re-run the algorithm and there is also the possibility to generate simpler event logs based on the existing clusters to analyse them separately by other algorithms (Veiga and Ferreira, 2009).

In this case, if there is only one cluster to generate, the Markov Chain created, as can be seen in Figure 5.8, is very similar to the graph generated by the Heuristics miner plug-in (Figure 5.5).

With a different clustering option, each user's session (process instance) may be associated to a cluster, creating one Markov Chain to each one. Therefore the Markov Chain is partitioned to new ones with transition probabilities higher or lower depending on the number of transitions and states each session contains, as Figure 5.9 depicts. Conclusions can be drawn from this, for example, the LoginPage self-loop is more likely to happen considering all the sessions of different users than the immediate transition of this webpage to the Service_Request_Show webpage (only happening in one small session).

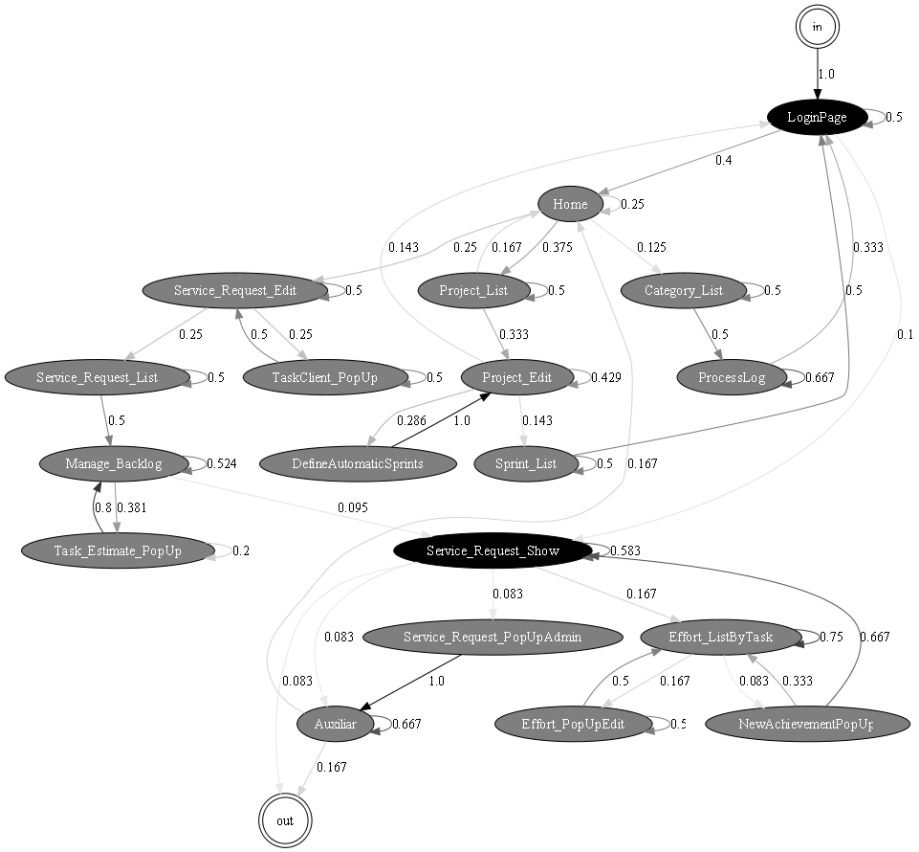


Figure 5.8: Markov Chain generated by the Sequence Clustering plug-in of all sessions in the Service Manager's event log

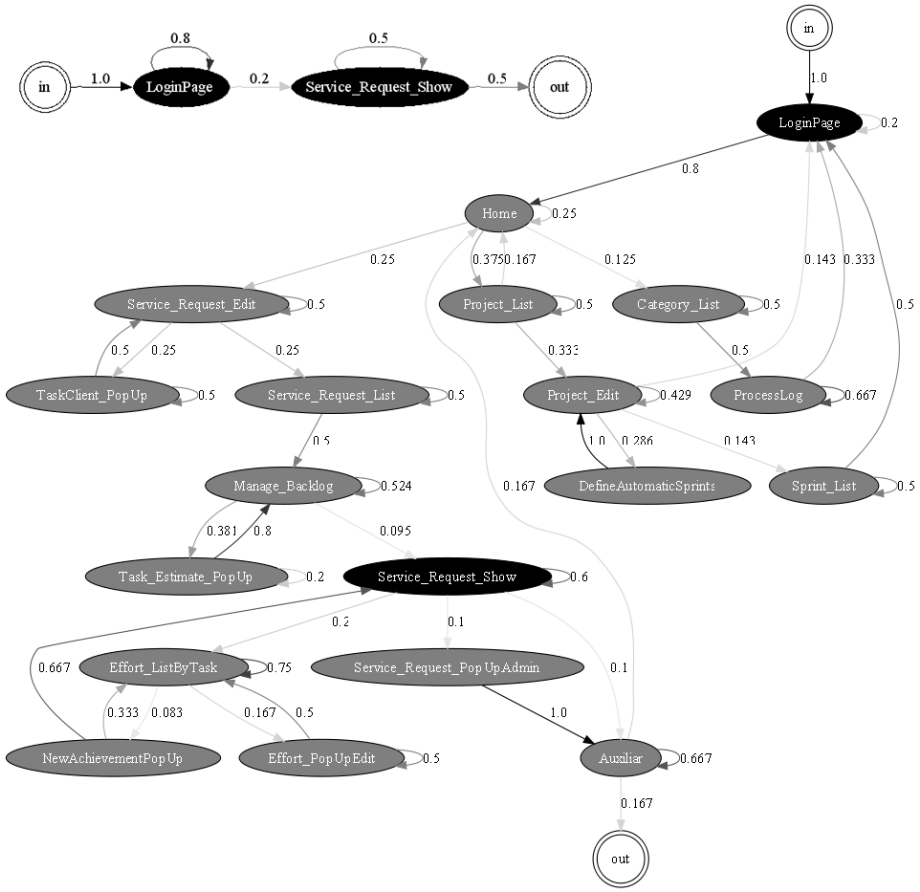


Figure 5.9: Markov Chains generated by the Sequence Clustering plug-in of the two clusters associated to two sessions of the Service Manager

5.2.5 Social Network Miner

The ProM Social Network Miner plug-in allows a set of options to explore the social network embedded in the event log. The social network is built based on the number of users that interact with each other in the same process instances. In this situation these instances or cases refer to sessions.

This poses a problem to the identification of causality between activities (where each webpage visit represents an activity) of different users in one session. The common method to discover a user's workflow by causality dependencies cannot be applied to multiple users in the same session, since one webpage visit from one user may not be related at all to the next webpage visit made by any other user in that same session. One user may be visiting several webpages that have no correlation with other user's visits, albeit both being in the same session.

With only the timestamp information to establish relations between the session's visits of different users to its webpages, causality cannot be taken into account, as well multiple transfers of handover of work or subcontracting within one session. So, by choosing an option of the ProM plug-in to visually present the social network, the

simultaneous appearance ratio (whose values are based on comparisons between the amount of events that each user has performed together with another user in a case), a ProM working together graph following metrics based on joint cases (frequency of performed activities by different individuals in the same case (Aalst et al., 2005)) can be depicted as presented in Figure 5.10. Since causality and an event's dependencies to multiple users are not being considered in this context, the graph's interpretation should rather be restricted to the number of connections between the existing users than their ratios of transfers.

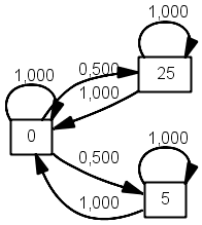
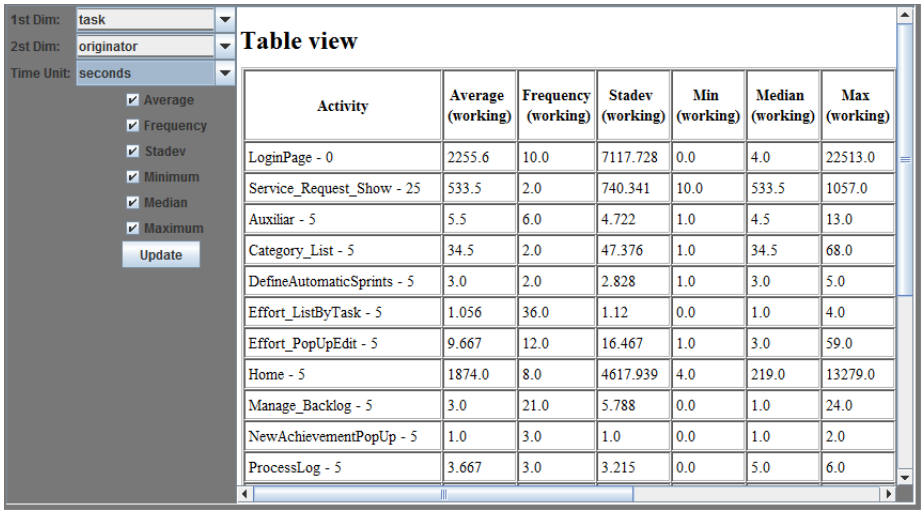


Figure 5.10: Example of working together graph produced by ProM Social Network Miner plug-in

5.2.6 Performance

The basic performance analysis of ProM allows some insight into the details of the visited webpages as depicted in Figure 5.11.



Activity	Average (working)	Frequency (working)	Stadev (working)	Min (working)	Median (working)	Max (working)
LoginPage - 0	2255.6	10.0	7117.728	0.0	4.0	22513.0
Service_Request_Show - 25	533.5	2.0	740.341	10.0	533.5	1057.0
Auxiliar - 5	5.5	6.0	4.722	1.0	4.5	13.0
Category_List - 5	34.5	2.0	47.376	1.0	34.5	68.0
DefineAutomaticSprints - 5	3.0	2.0	2.828	1.0	3.0	5.0
Effort_ListByTask - 5	1.056	36.0	1.12	0.0	1.0	4.0
Effort_PopUpEdit - 5	9.667	12.0	16.467	1.0	3.0	59.0
Home - 5	1874.0	8.0	4617.939	4.0	219.0	13279.0
Manage_Backlog - 5	3.0	21.0	5.788	0.0	1.0	24.0
NewAchievementPopUp - 5	1.0	3.0	1.0	0.0	1.0	2.0
ProcessLog - 5	3.667	3.0	3.215	0.0	5.0	6.0

Figure 5.11: Table view of a basic performance analysis in ProM

This analysis may provide information about which users visited particular webpages, according to the selected dimensions (originators or tasks, referring to users or webpages, respectively), and it may also relate to specific time units to analyse the working time (illustrated in the columns of the table). However each timestamp refers

to one visit of one single webpage (opened with a single click), therefore this analysis is not appropriate to reach conclusions of neither the working time nor the waiting time. So an adapted analysis for this case was taken to achieve relevant values of performance, as was explained in Section 4.3.3, which will be detailed in the next session.

5.3 Analysis with LogRetriever

LogRetriever is the developed application described in Section 4.3 and 4.4 that generated the MXML document from the event log of the LogScreen application. This document was used in the previous section (Section 5.2) in order to apply the process mining techniques available in ProM.

This section details how the LogRetriever application handles with the same document to produce models and relevant information in this context.

5.3.1 Control flow

As Figure 5.6 presented earlier, Figure 5.12 and Figure 5.13 also depict the control flows that the LogRetriever produces with the same MXML document based on an event log of the Service Manager application. As mentioned in the Section 3.1.2, the process, cases and organizational perspectives presented in the technique's description can be applied in this context. The general control flow graph accumulates all the process instances that the document contains in order to strengthen the connections of the same webpages. A new flow can be built with the separation of process instances (Figure 5.12). By distinguishing the users that interacted with the application, a third flow is created (Figure 5.13) and by comparing the sub-graphs of this flow with the previous flows it is observable that before any user authentication in the application, the system (identifier 0) presents the LoginPage to that same user. It is also observable that each user is related to an instance in this specific case, therefore it is possible to conclude that the users did not interact with each other - they only required one initial communication with the system (id 0).

This kind of analysis is useful to know which users share the same sessions and to determine when a user's activity stops in that instance to give place to other user's activities. The general control flow also provides a global understanding of how all the users of the application are interacting with it.

5.3.2 Social Network

The social network of this log is fairly simple as can be seen in Figure 5.14, since only two users interacted with the system.

Further interactions with the application can reveal connections between users if they share the same session, but it is also expected that each new user communicates with the system, at least at the LoginPage.

5.3.3 Performance

The performance analysis described in Section 4.3.3 is here applied to the Service Manager application. Since the graph generates a long table, a sample of that same table is depicted in Figure 5.15.

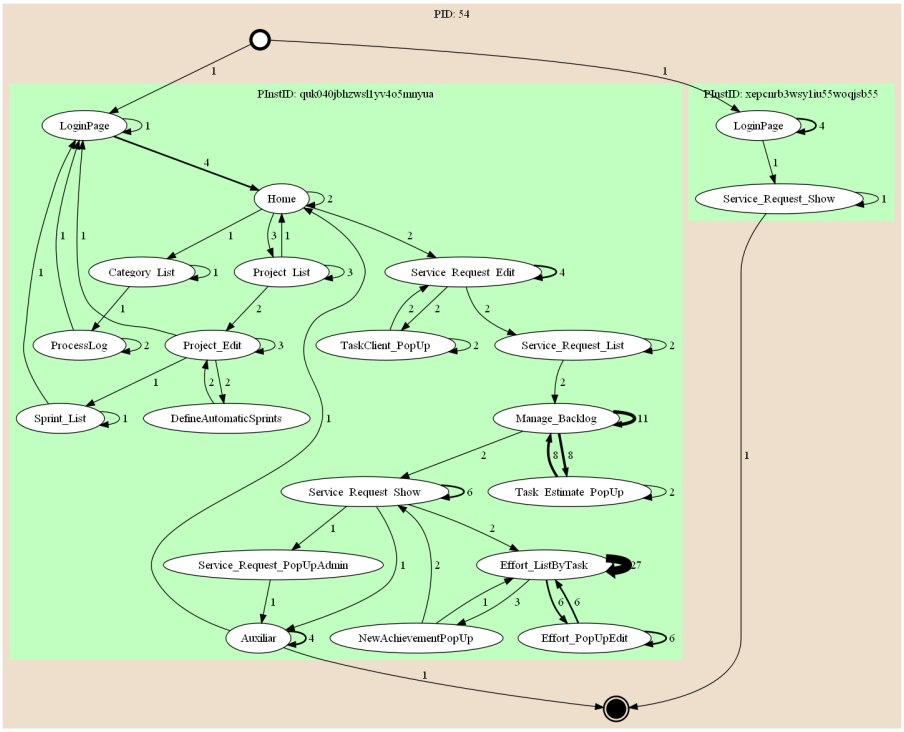


Figure 5.12: LogRetriever's Case (session) flow

Even with this figure only, one can observe that the LoginPage does not have a single exit time, which means the users may often refresh this initial webpage (or the browser itself can save the URL and redirect to that webpage each time a user disconnects). Each time a user connects, the user identifier in the system is swapped to an identified user inside the application. It is also observable that the user with the less visits in the application (identifier 25) was the last one to reach the LoginPage and the one with the most recent interaction recorded in the event log.

5.4 Discussion

The analysis made via ProM plug-ins and via LogRetriever were both useful. The activities of the event log are related to a webpage visit, when they are often related to concrete tasks. Therefore the approach presented in Chapter 4 was needed to complement the results and to achieve appropriate results. By using these analysis, discrepancies from the expected behavior can be detected and corrected by the developer. For example, it is possible to conclude at the control flow level that if a user navigates from a webpage to another which is not depicted in the Service Studio, the user may have used a link from the navigation history to that specific webpage. This can tell the developer whether to restrict the access to the latter or to place a link in the former, according to the business model containing the desired behavior from the user.

In general, there is a potential for more discrepancies when there is an increase in the number of users interacting with the application. Considering this, the analysis

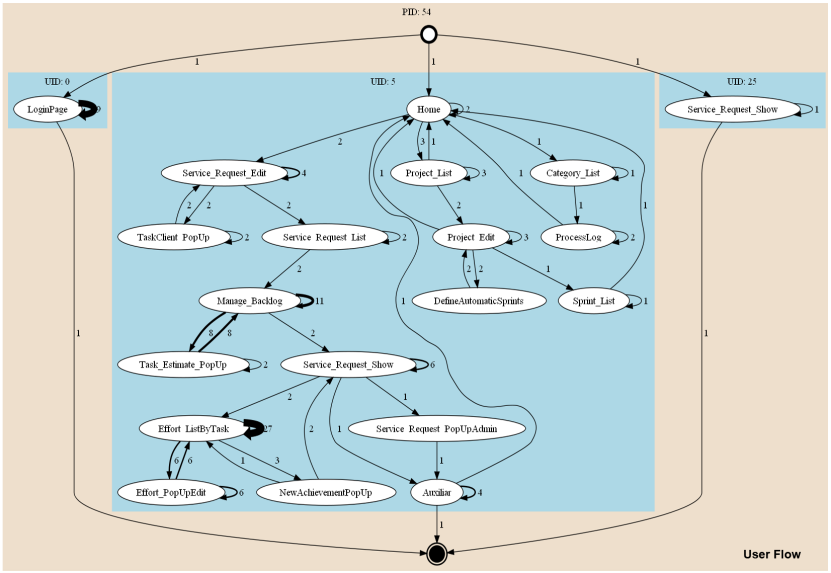


Figure 5.13: LogRetriever's User flow

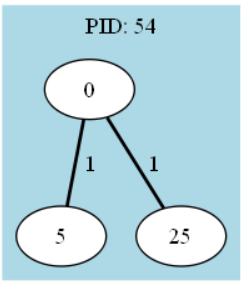


Figure 5.14: Service Manager Social Network

proposed by this work and the developed application become even more significant in order to discover the mentioned aspects, as also which webpages the users are spending most of their time or if the average time spent on them relates to the expected time for the completion of their tasks.

5.5 Summary

In this chapter a case study was presented where a specific application, the Service Manager, was chosen to illustrate several analysis that can be made to event logs from the Agile Platform's applications. These analysis were provided by using the ProM plug-ins and the developed application (LogRetriever), allowing to reach conclusions through the comparison of both analysis.

Name	Process	Visits	Loops	Users	NumberOfUserVisit	UsersVisits
LoginPage1	54	10	5	0 0 0 0 0 0 0 0 0 0	1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th	10
Home1	54	8	2	5 5 5 5 5 5 5 5	1st 2nd 3rd 4th 5th 6th 7th 8th	8
Project_List1	54	6	3	5 5 5 5 5 5	1st 2nd 3rd 4th 5th 6th	6
Service_Request_Show1	54	12	7	5 5 5 5 5 5 5 5 5 5 25 25	1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th 1st 2nd	10 2

Instance	EntranceTimes	ExitTimes	MinDuration	MaxDuration	AvgDuration
quk040jbhzwsl1yv4o5mnyua	24-05-2010 09:55:49	-Loop-	00:00:00	00:00:00	00:00:00
quk040jbhzwsl1yv4o5mnyua	24-05-2010 09:56:09	-User Swap to 5-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:25:32	-User Swap to 5-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:28:57	-User Swap to 5-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:29:29	-User Swap to 5-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:18	-Loop-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:28	-Loop-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:28	-Loop-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:33	-Loop-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:33	-User Swap to 25-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 09:56:13	24-05-2010 09:56:50	00:00:08	00:01:08	00:00:28
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:10:12	24-05-2010 10:10:20			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:20:00	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:27:40	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 14:08:59	24-05-2010 14:10:07			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:25:37	24-05-2010 20:26:01			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:29:04	24-05-2010 20:29:12			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:29:33	24-05-2010 20:29:57			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 09:56:50	-Loop-	00:00:02	00:13:22	00:04:34
quk040jbhzwsl1yv4o5mnyua	24-05-2010 09:56:50	24-05-2010 10:10:12			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:26:01	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:26:03	24-05-2010 20:26:21			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:29:12	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:29:14	24-05-2010 20:29:16			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:11:28	24-05-2010 10:11:31	00:00:03	00:00:13	00:00:06
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:12:28	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 10:12:31	24-05-2010 10:12:44			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:31:48	24-05-2010 20:31:51			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:33:44	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:33:45	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:33:55	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:34:07	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:34:08	-Loop-			
quk040jbhzwsl1yv4o5mnyua	24-05-2010 20:34:11	24-05-2010 20:34:17			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:23:43	-Loop-			
xepcnrb3wsy1iu55woqjsb55	24-05-2010 17:41:20	-End Of File-			

Figure 5.15: Performance analysis of the Service Manager application

Chapter 6

Conclusion

Applications developed with the OutSystems Platform produce relevant run-time data that can be analyzed in order to obtain approximate process models to the ones that were used to develop those same applications. Process Mining has an important role in the discovery of this aspect by applying algorithms over event logs after they have been correctly preprocessed. Those models provide additional knowledge about the development process and actual use of applications.

Finding the right source for gathering event data is an important issue, when there are several options to choose from. After the source of the event logs was chosen, the event logs were automatically converted to a known format in the Process Mining field (MXML) where both existing and custom techniques were applied to the Agile Platform's applications.

6.1 Main Contributions

The examples described in this work show that it is possible to retrieve relevant information of users behaviors and workflows, how they interact with applications developed in the OutSystems Platform and how their behaviors are a basis for improving the original model that defined those same applications. Comparisons in this sense facilitate the comprehension and structuring of the real flow that takes place when users interact with the applications. They also allow developers that are not entirely familiarized with the application to understand where they can start to work on, based on the users interactions with that application.

Therefore this work provided methods to improve applications developed in the Agile Platform of OutSystems via the analysis of the results, stored in event logs, of user interactions and behaviors with those applications.

6.2 Future Work

Further analysis and investigation may lead to different approaches on how to achieve improvements on applications developed in the Service Studio. There are also several ways to apply the process mining techniques to discover further results or to associate information provided by different techniques to the same graph. For example, different visualization options may be created based on the performance approach described in Section 4.3.3, such as graphs to:

- depict the minimum or maximum flow of webpages of one session or of the set of sessions presented in the event log;
- determine which user in the social network takes more time to reach certain webpages;
- find the probability of a user to complete a certain sequence of webpages.

Another actual challenge is to easily allow the comparison of the models produced via process mining techniques with those designed in the Service Studio. Conformance is an important notion in the context of business alignment, auditing and business process improvement (Rozinat, 2006). In the latest version of the OutSystems Platform (OutSystems, June 29, 2010), it is possible to export images from the studio containing the application's Screen Flows, Processes or Actions. Although this becomes visually useful to check the conformance between these models, it is, in fact, a complex task to do so, since a single user workflow can be spread through all the Screen Flows of the application.

Therefore a breakthrough would be to develop a mechanism to export the entire set of Screen Flows, Processes and Actions of a specific application into a single known text file format (such as the ones used in Graphviz, for instance) in order to allow the usage of simple algorithms to produce conformance reports. This mechanism could be either integrated in the Service Studio, or be part of a Service Center option, such as the application's download option (oml file).

The amount of possibilities for a conformance analysis are vast. The greatest benefit would be to provide precise knowledge about the possibilities to improve the application and where to put the development effort.

Bibliography

W. M. P. van der Aalst and B. F. van Dongen. Discovering workflow performance models from timed logs. In Yanbo Han, Stefan Tai, and Dietmar Wikarski, editors, *Engineering and Deployment of Cooperative Information Systems*, volume 2480 of *Lecture Notes in Computer Science*, pages 107–110. Springer Berlin / Heidelberg, 2002.

W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workflow mining: A survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267, November 2003a.

W. M. P. van der Aalst, A.J.M.M. Weijter, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:2004, 2003b.

W. M. P. van der Aalst, Hajo A. Reijers, and Minseok Song. Discovering social networks from event logs. *Comput. Supported Coop. Work*, 14(6):549–593, 2005.

W. M. P. van der Aalst, B. F. van Dongen, Christian W. Günther, R. S. Mans, A. K. A. de Medeiros, A. Rozinat, Vladimir Rubin, Minseok Song, H. M. W. (Eric) Verbeek, and A. J. M. M. Weijters. Prom 4.0: Comprehensive support for real process analysis. In Jetty Kleijn and Alex Yakovlev, editors, *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer, 2007a.

W. M. P. van der Aalst, H. Reijers, A. Weijters, B. F. van Dongen, A. K. A. de Medeiros, M. Song, and H. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, July 2007b.

W. M. P. van der Aalst, B. F. van Dongen, C. Gunther, A. Rozinat, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM: The Process Mining Toolkit. *BPM 2009 Demonstration Track*, 2009.

P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen. New directions on agile methods: a comparative analysis. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 244–254, May 2003.

Rakesh Agrawal, Dimitrios Gunopulos, and Frank Leymann. Mining process models from workflow logs. In Hans-Jörg Schek, Gustavo Alonso, Felix Saltor, and Isidro Ramos, editors, *Advances in Database Technology - EDBT'98*, volume 1377 of *Lecture Notes in Computer Science*, pages 467–483. Springer Berlin / Heidelberg, 1998.

H. Russell Bernard, Eugene C. Johnsen, Peter D. Killworth, Christopher McCarty, Gene A. Shelley, and Scott Robinson. Comparing four different methods for measuring personal social networks. *Social Networks*, 12(3):179 – 215, 1990.

Judith Bishop. Multi-platform user interface construction: a challenge for software engineering-in-the-small. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 751–760, New York, NY, USA, 2006. ACM.

Scott Boag, Donald D. Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. XQuery 1.0: An XML Query Language. World Wide Web Consortium, Recommendation REC-xquery-20070123.

Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Min. Knowl. Discov.*, 7(4):399–424, 2003.

Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7:215–249, 1998.

B. F. van Dongen and W. M. P. van der Aalst. A Meta Model for Process Mining Data. In *Conference on Advanced Information Systems Engineering*, volume 161, Porto, Portugal, 2005.

B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The ProM Framework: A New Era in Process Mining Tool Support. In *Lecture Notes in Computer Science: Applications and Theory of Petri Nets 2005: 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005. / Gianfranco Ciardo, Philippe Darondeau (Eds.)*, volume 3536, pages 444–454. Springer Verlag, June 2005.

Diogo R. Ferreira. Applied sequence clustering techniques for process mining. In Jorge Cardoso and Wil van der Aalst, editors, *Handbook of Research on Business Process Modeling*, Information Science Reference, pages 492–513. IGI Global, 2009.

Diogo R. Ferreira and Miguel Mira da Silva. Using process mining for ITIL assessment: a case study with incident management. In *Proceedings of the 13th Annual UKAIS Conference*. Bournemouth University, April 10-11 2008.

Diogo R. Ferreira, Marielba Zacarias, Miguel Malheiros, and Pedro Ferreira. Approaching process mining with sequence clustering: Experiments and findings. In Gustavo Alonso, Peter Dadam, and Michael Rosemann, editors, *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, volume 4714 of *Lecture Notes in Computer Science*, pages 360–374. Springer, 2007.

Mark Girolami and Ata Kabán. Sequential activity profiling: Latent dirichlet allocation of markov chains. *Data Min. Knowl. Discov.*, 10(3):175–196, 2005.

A. K. A. de Medeiros and A. J. M. M. Weijters. Genetic process mining. In *Applications and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science*, pages 48–69. Springer-Verlag, 2005.

A. K. A. de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters. Process mining: Extending the alpha-algorithm to mine short loops. In *Eindhoven University of Technology, Eindhoven*, 2004a.

A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst. Genetic process mining: A basic approach and its challenges. In *Business Process Management Workshops*, pages 203–215, 2005.

A. K. A. de Medeiros, A. J. Weijters, and W. M. P. Aalst. Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, 2007.

A. K. A. de de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters. Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. In *UMICS*, pages 151–165, 2004b.

Carlos Manuel Martins Mendes. Catálogo de serviços de tecnologias de informação. Technical report, Instituto Superior Técnico, 2010.

Michael zur Muehlen and Michael Rosemann. Workflow-based process monitoring and controlling - technical and organizational issues. *Hawaii International Conference on System Sciences*, 6:6032, 2000.

OutSystems. Agile Platform, Business Overview Datasheet. Published in the OutSystems' website.

OutSystems. Agile Platform, Technical Overview Datasheet. Published in the OutSystems' website.

OutSystems. Transitioning to Agile in an Agile Way: Amplifying Traditional Approaches with Agile Technology. Published in the OutSystems' website.

OutSystems. Agile Platform, What's New in Version 5.0. Published in the OutSystems' website by Jaime Vasconcelos.

OutSystems. Agile Platform, Business Process Technology Overview Datasheet. Published in the OutSystems' website by Manuel Dias.

OutSystems. SCRUM vs OutSystems roles Technical Note. Published in the OutSystems' website by Nuno Teles.

OutSystems. Agile Platform, What's New in Version 5.1. Published in the OutSystems' website by Jaime Vasconcelos.

OutSystems. OutSystems Agile Methodology Overview Datasheet. Published in the OutSystems' website by Rodrigo Castelo.

D. J. Power and S. Kaparthi. Building web-based decision support systems. *Studies in Informatics and Control*, 11:291–302, 2002.

Linda Rising and Norman S. Janoff. The scrum software development process for small teams. *IEEE Software*, 17:26–32, 2000.

A. Rozinat. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *BPM 2005 Workshops (Workshop on Business Process Intelligence), volume 3812 of Lecture Notes in Computer Science*, pages 163–176. Springer-Verlag, 2006.

A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.

Ken Schwaber. SCRUM development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 117–134, 1995.

Gil Aires da Silva and Diogo R. Ferreira. Applying hidden markov models to process mining. In A. Rocha, F. Restivo, L. P. Reis, and S. Torr ao, editors, *Sistemas e Tecnologias de Informação: Actas da 4a. Conferência Ibérica de Sistemas e Tecnologias de Informação*, pages 207–210. AISTI/FEUP/UPF, 2009.

Miguel Mira da Silva and José Sequeira Martins. *IT Governance - A Gestão da Informática*. FCA, 2008.

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

W. H. Morkel Theunissen, Derrick G. Kourie, and Bruce W. Watson. Standards and agile software development. In *SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pages 178–188, , Republic of South Africa, 2003. South African Institute for Computer Scientists and Information Technologists.

Gabriel M. Veiga and Diogo R. Ferreira. Understanding spaghetti models with sequence clustering for ProM. In *Business Process Intelligence (BPI 2009): Workshop Proceedings*, Ulm, Germany, September 2009.

A. J. M. M. Weijters and A. K. A. de Medeiros. Process mining with the HeuristicsMiner algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven.

T. Weijters and W. M. P. van der Aalst. Process mining: Discovering workflow models from event-based data. In Kröse, B., Rijke, M., Schreiber, G., and Someren, M., editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 283–290, 2001.

Lijie Wen, Jianmin Wang, and Jiaguang Sun. Detecting implicit dependencies between tasks from event logs. pages 591–603. 2006.

Appendix: LogRetriever Manual

LogRetriever Manual

Alexandre Gouveia

Contents

List of Figures

1 Introduction

This manual is integrated in the context of the thesis of the same author: *Process Mining of Enterprise Applications based on OutSystems Agile Platform*, supervised by Prof. Diogo R. Ferreira and co-supervised by Prof. Miguel Mira da Silva. For further information about the underlying concepts used by the LogRetriever software, please refer to this thesis [1].

1.1 Software Prerequisites

LogRetriever requires a Windows operating system compatible with:

- A GraphViz [1] installation is required in order to generate the images of the graphs that LogRetriever produces.
- Java Runtime Environment for converting the HTML logs to the MXML format.
- LogScreen OutSystems application[2] published in the OutSystems server containing the applications to be studied in order to retrieve the log file.

1.2 Installation

By running the "LogRetriever Setup" installer the setup searches for the dot.exe and the java.exe executables in the Program Files directory. If one of these files is missing you will be prompted to be redirected to the respective website for the installation. After this, follow the wizard's steps to proceed to the installation of the LogRetriever program in the local disk. The browse button, depicted in the step of Figure 1, allows to select a different directory location for the program.

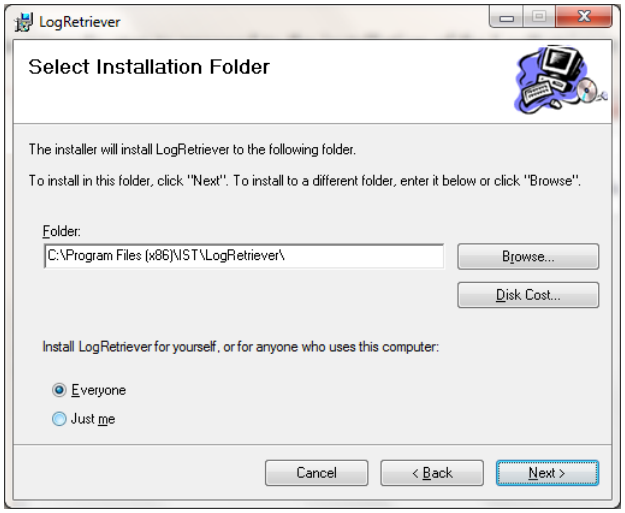


Figure 1: LogRetriever Installation Steps

[1] GraphViz – www.graphviz.org
[2] Developed in the context of the thesis

2 Using LogRetriever

This section presents the features available in the LogRetriever software. To launch the program double-click the executable file (LogRetriever.exe) or the shortcut placed in the Programs Menu.

2.1 Source Selection

After launching the program, the window depicted in Figure 2 allows to select the OutSystems log file's source.

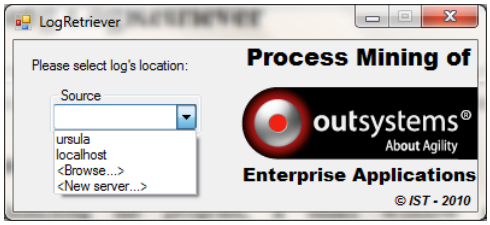


Figure 2: Selecting the log file source

Four options are presented in the ComboBox:

- <*New Server...*>: after publishing the LogScreen application on the OutSystems server, this option uses the log generated by this application as the source. There is the possibility to select only one application to access the respective log by indicating its identifier, as depicted in Figure
- <*Browse...*>: allows the browsing of specific files on the local machine, such as PNG, DOT, MXML and HTML in the OutSystems log file format.
- *localhost*: accesses to the local server used by the community edition similarly to the <*New Server...*> option.
- *ursula*: accesses to the server used in the thesis context similarly to the <*New Server...*> option.

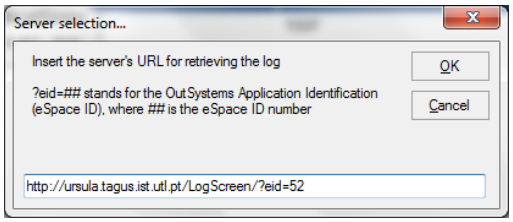


Figure 3: <*New Server...*> option allows to directly access the LogScreen application published in an OutSystems server

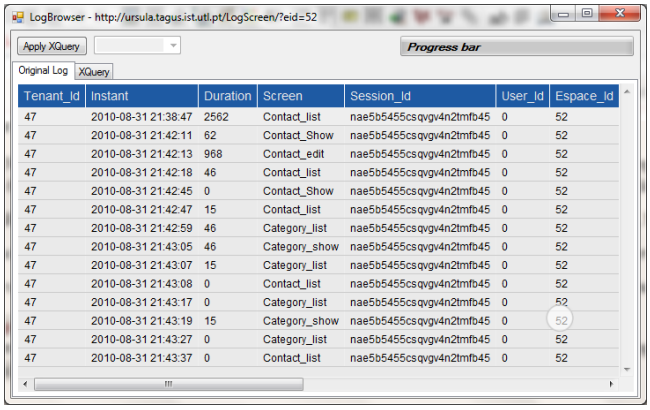
After selecting the source, pressing the retrieve button or the OK button (<*New Server...*> option) will lead to the log analysis window presented in section 2.2.

2.2 Log Analysis

Once the source is selected a window with multiple tabs and buttons is presented, allowing further analysis on the log file depending on the type of the source selected. The next sub-sections detail this analysis.

2.2.1 Using the XQuery

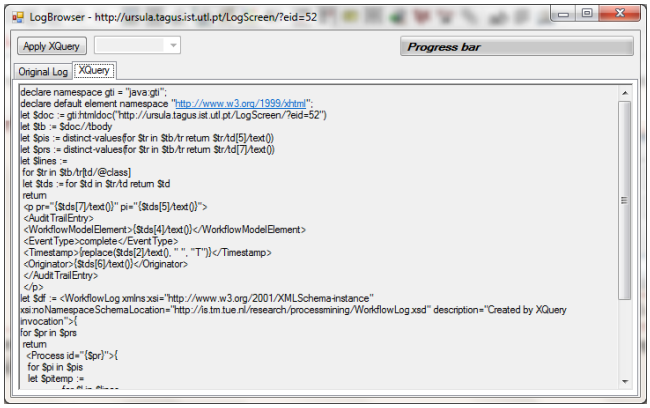
Following the <*New Server...*> source selection or any of the options that lead to a direct access to the LogScreen application HTML log, the window presented in Figure 4 is displayed.



Tenant_Id	Instant	Duration	Screen	Session_Id	User_Id	Espace_Id
47	2010-08-31 21:38:47	2562	Contact_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:11	62	Contact_Show	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:13	968	Contact_edit	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:18	46	Contact_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:45	0	Contact_Show	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:47	15	Contact_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:42:59	46	Category_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:05	46	Category_show	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:07	15	Category_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:08	0	Contact_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:17	0	Category_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:19	15	Category_show	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:27	0	Category_list	nae5b5455csqvgv4n2tmfb45	0	52
47	2010-08-31 21:43:37	0	Contact_list	nae5b5455csqvgv4n2tmfb45	0	52

Figure 4: LogScreen application log

This log must be converted to the MXML format before further analysis by the program, therefore the generated XQuery depicted in Figure 5 can be used by clicking the *Apply XQuery* button.




```
declare namespace gti = "java:gti";
declare default element namespace "http://www.w3.org/1999/xhtml";
let $doc := gti:htmldoc("http://ursula.tagus.ist.utl.pt/LogScreen/?eid=52")
let $tb := $doc//tbody
let $pis := distinct-values(for $tr in $tb/tr return $tr/td[5]/text())
let $prs := distinct-values(for $tr in $tb/tr return $tr/td[7]/text())
let $lines :=
for $tr in $tb/tr[td/@class]
let $tds := for $td in $tr/td return $td
return
 <p pr="{$tds[7]/text()}" pi="{$tds[5]/text()}">
 <AuditTrailEntry>
 <WorkflowModelElement>{$tds[4]/text()}</WorkflowModelElement>
 <EventType>complete</EventType>
 <Timestamp>{replace($tds[2]/text(), " ", "T")}</Timestamp>
 <Originator>{$tds[6]/text()}</Originator>
 </AuditTrailEntry>
 </p>
let $df := <WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd" description="Created by XQuery
invocation">{
for $pr in $prs
return
 <Process id="{$pr}">{
  for $pi in $pis
  let $pitemp :=
```

Figure 5: XQuery generated by the program

On the other hand, if the <*Browse...*> source option was selected, this step may be skipped according to the type of file selected to be opened.

2.2.2 Saving the Log and Analysis Options ComboBox

After applying the XQuery, the generated MXML is presented and can be freely edited directly in the text box and saved to the local disk by either clicking the *Save and Update MXML* button or the *Edit MXML in editor* button.

There is also a ComboBox at the top of the window to select the type of analysis to be applied to the log. Once one of the presented options in Figure 6 is selected the *View DOT* and *View Image* buttons become active.

After pressing one of these buttons, in order to focus in the selected analysis the Options ComboBox is locked. If a different analysis of the same log is required at the same time, the source selection window (Section 2.1) can be reused to pop-up another Log Analysis window with the same source. Although, if the *Save and Update MXML* button (MXML tab) is clicked the tabs at the right of the MXML tab are discarded and the Options ComboBox unlocked again.

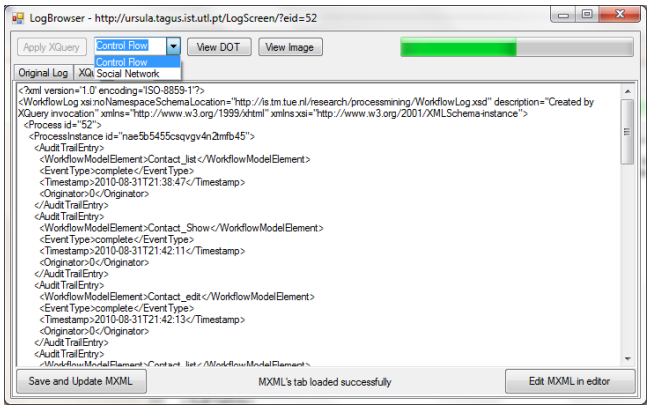


Figure 6: MXML tab and options for analysing the file

2.2.3 Control Flow Analysis

The control flow analysis in this program allows to access a flow graph generated according to the visits of users to the pages presented in the log file and performance analysis based on this same graph.

If the *Control Flow* option is selected from the ComboBox, then the DOT file (containing the explicit pages' transitions from the log) will be generated after pressing the *View DOT* button or, since to create the image the DOT file is required as input, the *View Image* button.

In order to generate this file, a new window is presented (Figure 7) allowing to set several options referring to the DOT file, such as:

- CheckBoxes:
  - *Distinguish Users*: all the visited pages by a certain user are grouped into a specific subgraph.
  - *Separate Process Instances*: all the pages belonging to a specific subgraph are grouped into a specific subgraph.

– *Remove Self-Loops*: if one user visits one page immediately after one visit to that same page, this visit is not taken into account in the graph.

- *Bound Number of Transitions*: allows to restrain the acceptance to the DOT file of particular transitions that exceeded the *Upper bound* limit or that do not reach the *Lower bound* one. E.g.: If one transition from a page to another was made 31 times and other was made only once, by considering the bounds of Figure 7, the former will not be included in the DOT file but the latter will.

- *Output Rename Mask*: serves to include a different name on the generated DOT or image files.

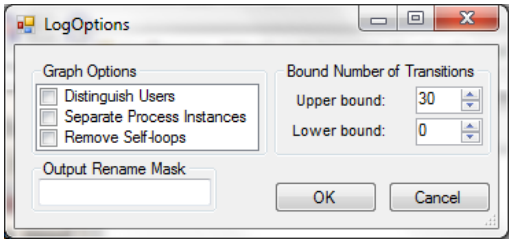


Figure 7: Control Flow Options

If neither the *Distinguish Users* or the *Separate Process Instances* options is selected, the general control flow is displayed containing only the application(s) subgraph(s) contained in the log.

By pressing the *View DOT* button a new tab is presented with the DOT file allowing to alter it in a similar manner to the MXML file presented in the tab of Figure 6.

By pressing the *View Image* button the image is generated using the DOT file displayed and saved in the DOT tab. The image can be zoomed in or out with a zoom bar as Figure 8 depicts.

Based on the control flow graph, the program allows to access a performance analysis by clicking the *Performance* button. This analysis is displayed on a new tab in a similar table to the one depicted in Table 1.

Name	Process	Visits	Loops	Taken	NumberOfUserVisit	TakenVisits	Instance	EntranceTimes	ExitTimes	MinDuration	MaxDuration	AvgDuration
				0	1st		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:38:47	31-08-2010 21:42:11			
				0	2nd		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:42:18	31-08-2010 21:42:45			
				0	3rd		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:42:47	31-08-2010 21:42:59			
				0	4th		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:08	31-08-2010 21:43:17			
Contact_list1	52	5	0	0	5th	5	nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:37	-End Of File-	00:00:09	00:03:24	00:01:03
				0	1st		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:42:11	31-08-2010 21:42:13			
Contact_Show1	52	2	0	0	2nd	2	nae5b5455csqvgv4n2tmfb45	31-08-2010 21:42:45	31-08-2010 21:42:47	00:00:02	00:00:02	00:00:02
Contact_edit1	52	1	0	0	1st	1	nae5b5455csqvgv4n2tmfb45	31-08-10 21:42	31-08-10 21:42	00:00:05	00:00:05	00:00:05
				0	1st		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:42:59	31-08-2010 21:43:05			
				0	2nd		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:07	31-08-2010 21:43:08			
				0	3rd		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:17	31-08-2010 21:43:19			
Category_list1	52	4	0	0	4th	4	nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:27	31-08-2010 21:43:37	00:00:01	00:00:10	00:00:04
				0	1st		nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:05	31-08-2010 21:43:07			
Category_show1	52	2	0	0	2nd	2	nae5b5455csqvgv4n2tmfb45	31-08-2010 21:43:19	31-08-2010 21:43:27	00:00:02	00:00:08	00:00:05

Table 1: Performance analysis table

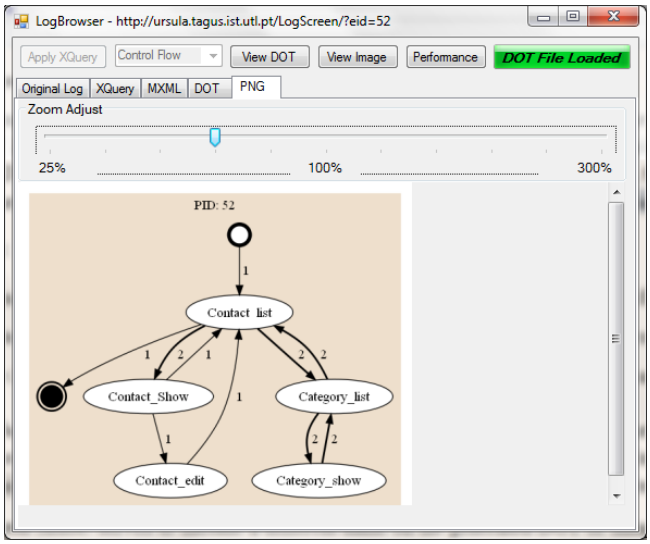


Figure 8: Image visualizer with zoom and pan controls

2.2.4 Social Network Analysis

The social network analysis in this program allows to access a network graph generated according to the users that shared the same sessions on particular applications contained in the log file, as depicted in Figure 9.

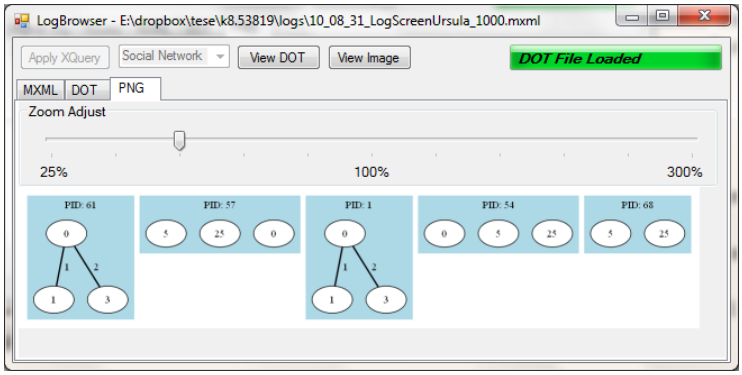


Figure 9: Social Network of a browsed file

If the *Social Network* option is selected from the ComboBox, then the DOT file (containing the connections between the users) will be generated after pressing the *View DOT* button or, since to create the image the DOT file is required as input, the *View Image* button.

3 Glossary

DOT: Graphviz[3] program to produce "hierarchical" or layered drawings of directed graphs from a specific file format (DOT file format).

LogScreen: An OutSystems application that accesses an OutSystems server's database to list the visits made to the pages of the published applications on that server.

MXML: Extensible XML-based format for storing process event logs supported by process mining analysis tools, such as ProM[4].

XQuery: Markup language capable of labeling the information content of diverse data sources including structured and semi-structured documents, relational databases, and object repositories [5]. In the context of the LogRetriever program it allows the conversion of the HTML log from the LogScreen OutSystems application to the MXML file format.

References

[1] Alexandre L. Gouveia, Diogo R. Ferreira, and Miguel Mira da Silva. Process Mining of Enterprise Applications based on OutSystems Agile Platform. Master's thesis, Instituto Superior Técnico, 2010.

[3] GraphViz – www.graphviz.org
[4] ProM – http://prom.sourceforge.net
[5] XQuery – www.w3.org/TR/xquery