Mining Data in the Context of
Semi-Structured Business Processes

Applying Process Mining in IBM WebSphere

By
Gabriela Alina Chipaila

Supervisors:
Prof. Dr. Ir. W.M.P. (Wil) van der Aalst (TU/e)
Dr. Ir. H.M.W. (Eric) Verbeek (TU/e)

Eindhoven, December 2007
ACKNOWLEDGEMENTS

This master thesis encompasses the work I accomplished during my master project for the Business Information Systems study program at the Technical University of Eindhoven (TU/e). The research was carried out at the Architecture of Information Systems (AIS) group of the Computer Science department, and it was part of a collaboration project between TU/e and IBM.

During my study, there have been many people that inspired and motivated me in my work. First of all, I would like to express my gratitude to my supervisors Wil van der Aalst and Eric Verbeek, for giving me the opportunity to work on this challenging project and for their excellent support in carrying out my research assignment. Wil’s passion and commitment for his work have always been an inspiration for my education, both as a student and as a person in general. I also wish to thank Eric for his technical support and stimulating discussions that guided me into the right direction. Furthermore, I would like to thank Gerhard Pfau for his assistance and collaboration with IBM, and Toon Calders for taking part in my master project’s evaluation.

I am also grateful for making many nice friends among the students working in this research environment. I cannot mention all their names, but I want to name the Romanian girls, Carmen Bratosin and Olivia Oanea, for all their support, Minseok Song, for his assistance in implementing my ProM plug-in, Anne Rozinat, Christian Günther and Boudewijn van Dongen for their contributions.

Furthermore, I wish to thank my family and all my dear friends for their love and support throughout this project. A special thanks is given to my boyfriend, Thomas, for being always there for me to overcome the difficult periods of my graduation project.

Finally, I am glad that I had the opportunity to have met such inspiring and friendly people. I can say that, definitely, this master project is one of the most valuable experiences I had so far.

Gabriela Alina Chipaila
Eindhoven, December 7, 2007
CONTENTS

1 Introduction .........................................................................................................................1
  1.1 Context of the master project...................................................................................1
  1.2 Problem statement and objectives ...........................................................................2
  1.3 Thesis outline ...........................................................................................................3

2 Process Mining Overview ....................................................................................................5
  2.1 Business Processes .................................................................................................5
  2.2 Mining Business Processes .....................................................................................6
  2.3 The ProM Framework ..............................................................................................7

3 Data as 1st Class Citizen ................................................................................................10
  3.1 Flexibility Taxonomy ...............................................................................................10
    3.1.1 Flexibility by configuration ...........................................................................10
    3.1.2 Flexibility by adaptation ..............................................................................11
    3.1.3 Conclusions with respect to flexibility .........................................................12
  3.2 Data-driven Approaches ........................................................................................13
    3.2.1 Case Handling ...............................................................................................14
    3.2.2 Product-Based Workflow Design (PBWD) ...................................................17
    3.2.3 Conclusions ...................................................................................................20
  3.3 Mining the Data perspective ..................................................................................22
    3.3.1 Importance of monitoring and mining data ....................................................22
    3.3.2 Classical data mining and data visualization ...................................................22
    3.3.3 Mining data in the context of business processes ..........................................23
  3.4 Conclusion ............................................................................................................28

4 Data Visualization Method ................................................................................................30
  4.1 Motivation and Objective ......................................................................................30
  4.2 Method Description ...............................................................................................32
    4.2.1 Task execution states ...................................................................................33
    4.2.2 Originator states ...........................................................................................36
    4.2.3 Data objects states .......................................................................................38
    4.2.4 Process instance states ..................................................................................40
    4.2.5 Method requirements summary .......................................................................41
  4.3 Implementation .....................................................................................................42
    4.3.1 General plug-in functionality .........................................................................42
    4.3.2 Task perspective ............................................................................................46
    4.3.3 Originator perspective ...................................................................................47
    4.3.4 Data perspective ............................................................................................48
    4.3.5 Process instance perspective ..........................................................................50
  4.4 Evaluation .............................................................................................................51
    4.4.1 Simulation of a case handling model in CPN Tools .......................................52
    4.4.2 Performance analysis ......................................................................................53
1 Introduction

This master thesis reports on a research study conducted in the field of process mining, addressing the needs of flexible, data-intensive business processes. Throughout the thesis we highlight the role of data in designing, executing and analyzing business processes, and we plead for more support for data-focused process mining techniques. Furthermore, we investigate the applicability and the benefits of process mining in a real-life environment. For this purpose, we use IBM WebSphere as an example of an information system used in industry to perform a case study on.

In the first section of this chapter we present the master project setup and introduce the IBM technology that we used for our practical case study. The second section describes the problem statement and the objectives set out for this research. The last section outlines the structure of the thesis.

1.1 Context of the master project

The research assignment for this master project was carried out at Technical University of Eindhoven (TU/e), within the Architecture of Information Systems group of the Computer Science department. The research work is part of a joint project between TU/e and IBM, which aims at investigating appropriate means to provide first class support for both flexible processes and process mining in IBM middleware. IBM WebSphere is the real-life technology used to perform the research assignment.

This master project was intended to follow up the outcome of the IBM Extreme Blue program 2006: Case handling for Knowledge Workers, where TU/e was involved in. During this program, a prototype of a case handling application was built: the Case Handling Execution Framework (CHEF). CHEF was developed to provide more flexibility in handling administrative processes (e.g. processing of insurance claims, credit requests, or traffic offences). Initially, the goal of the master project was to extend CHEF with appropriate monitoring capabilities to enable process mining on such flexible processes. Although we were involved in the development of CHEF, due to technical setbacks, we were not able to both run and monitor the case handling application. Therefore, in this thesis we included only a conceptual discussion about how to monitor and analyze CHEF cases. Moreover, we planned to perform process mining on real-life WebSphere logs, from IBM customers. Unfortunately, this kind of data could not be made available.

Due to the challenges mentioned above, we had to change the direction of our research. Hence, we decided to focus on the role of data in the context of flexible business processes, and extend the support for mining this type of processes. The acquired knowledge can also be used on processes deployed by applications like CHEF. Furthermore, we looked at how to apply the process mining techniques on processes executed in a real-life environment. In this way, we provided the opportunity for IBM middleware to benefit from process mining practices.

In the next section we present the problem statement and the objectives of this thesis.
1.2 Problem statement and objectives

Knowledge intensive domains such as government administration, medical, financial and legal services rely heavily on data being processed. In such domains, the outcome of a process is often an information object (e.g. a decision or a document). Generally, these data intensive business processes cannot be supported by an enforced (rigid) process definition as they may require more flexibility to deal with unforeseen situations. As a result, data intensive domains need more control to assure a correct execution of their processes.

Moreover, the requirements imposed by dynamic environments on today’s information systems result in an increase in systems’ complexity and in the amount of data produced. The Business Intelligence tools market faces a greater demand for supporting more heterogeneous data (e.g. structured, unstructured, semi-structured, rich media information) [27]. Also, the growing volume of data in business environments triggers the acute need for intelligent data analysis to keep up with this trend. This further calls for a more comprehensive analysis approach that takes data into account as an intrinsic part of a business process.

Process Mining has become a popular practice to support process analysis and design, and various tools and techniques have been proposed to discover knowledge about process execution [9, 16, 19, 20, 24-26, 29]. These techniques seek to address the needs of various types of processes. Structured processes are highly standardized and usually exist in the production and manufacturing industry where the process definition must be followed almost by the letter. Opposite to production processes, there are many processes that focus on sharing of information. These are more data-driven and are completely unpredictable, disallowing their structure to be predefined. They are called unstructured processes. The third type of processes, semi-structured, falls in between the range of the two extreme cases mentioned, and is known to exist in knowledge intensive domains. These semi-structured processes require more flexibility than the structured ones, as the execution behavior is more uncertain and they have to deal with exceptions and changes. At the same time, they do have a structure that guides their execution, whilst the unstructured processes do not have the notion of a process definition. All of these process types are further explained in Chapter 2 of this thesis.

Over the last years, process mining has continued to evolve and include specialized techniques to deal with different aspects (i.e. perspectives) of a business process. It distinguishes between three main perspectives: the control-flow perspective (i.e. routing of activities in a process), the organizational perspective (i.e. how the people involved are performing the work), and the case perspective (i.e. properties of process instances, in particular data) [18, 20, 23, 32]. Initially, many of these methods and tools addressed the control-flow, while little support was provided for analysis of data in the context of business processes. Mining patterns and their frequencies in the log data, clustering the log based on defined metrics, correlation of data types and data values, are few examples that address the data perspective. Furthermore, the need for more interactive mining tools has been recognized by [28, 79, 81]. They point out that visual data exploration is intuitive and by exploiting the human practitioner’s knowledge and visual skills it can help reveal tacit knowledge that automated techniques are not able to discover. However, many things are left open in this area of knowledge discovery.

In order to improve their businesses, organizations need more support for analysis of business processes. However, commercial process management tools offer limited support in this area. Since process mining contributes significantly to process improvement and can be used for process control, we take this opportunity to demonstrate the applicability and the benefits of process mining in a real-life environment.

This leads to the objectives of this master thesis, which are:

1. Enhance the support for semi-structured business processes, by providing an innovative data visualization method to improve the data-centric analysis of business processes.
2. Analyze and study the applicability of process mining in a real-life information system.
Our primary goal, objective (1), is to demonstrate that data plays a key role in achieving more flexibility and to show how semi-structured business processes can benefit from data-driven approaches for design, execution and analysis. We bring our contribution to the process mining methods and techniques that focus on data and provide powerful means for interactive data exploration. This research is realized using the process mining framework ProM, developed at TU/e.

The supporting objective, objective (2), is to show how process mining knowledge can be applied on a real information system that is used in industry. One of the most successful modern technologies that offer rich functionality for business process management is IBM WebSphere. Generally, WebSphere is used to administrate structured processes. Because of WebSphere’s focus on structured processes, IBM is interested in adding case handling capabilities on top of the existing tools. This is why in a joint effort with TU/e, IBM started the development of CHEF. IBM CHEF is a prototype developed on top of WebSphere to provide more flexibility and to address the needs of semi-structured business processes. For these reasons, IBM WebSphere and CHEF are used as a case study. This will emphasize the added value of process mining in a real life setting.

Figure 1 illustrates the scope of our research, giving the spectrum of process mining techniques which are positioned with respect to the structure of the business process that is analyzed and the mining perspective that are focused on. The size of the blocks indicates the existing level of support for mining techniques. The picture also highlights the research main target, namely, data-focused process mining techniques that address the semi-structured business processes:

The next section presents the structure of the thesis.

1.3 Thesis outline

In Chapter 2 we present an overview of the process mining concepts and work that has been done so far. We also introduce the reader to the process mining tool that we have used in our study, the ProM Framework. After providing this preliminary knowledge, in Chapter 3 we present a literature review of existing data-driven approaches that address the design, execution and analysis of semi-structured business processes. The purpose of this chapter is to emphasize the importance of data in the context of flexible business processes and, as a consequence, promote data as a 1st class citizen in the process mining practice. To show how we can concretely support this, Chapter 4 proceeds with the description of a data visualization method that we have implemented in the context of the mining tool ProM. This method constitutes our major contribution to the research work in the area of process mining, and
extends the available collection of data-centric mining techniques. In this way, we answer the objective that we have set out at the beginning of this thesis.

In order to apply this knowledge in a real information system, in Chapter 5 we give an overview of the IBM WebSphere technology and describe the existing logging capabilities. Based on this information, Chapter 6 explains how the conversion of the WebSphere logs to the mining format used in ProM was implemented. The conversion is necessary in order to perform a case study on mining WebSphere event logs. The case study shows how structured BPEL processes can be mined using the ProM framework, and proposes some guidelines for monitoring more flexible processes that may be deployed using the IBM Case Handling Execution Framework (CHEF).

Finally, Chapter 7 presents the conclusions of the undertaken research, some of the challenges encountered, and necessary future work that should be carried out.
2 Process Mining Overview

In this chapter we give an introduction to the topic of process mining in order to familiarize the reader with some notions and terms that are used throughout the thesis. In Section 2.1 we classify business processes into three categories: structured, semi-structured, and unstructured. The semi-structured type of processes will be mostly addressed in the context of our research. Section 2.2 gives an overview of process mining, presenting the types of process analysis and various process perspectives that are addressed. The last section introduces the ProM framework. Currently the leading process mining tool, ProM has been used for incorporating the implementation of the data visualization method we have developed and also for our case study of IBM WebSphere.

2.1 Business Processes

Business Process Management (BPM) became an integral part of IT infrastructures of modern businesses, to manage the design, configuration, enactment and diagnosis of their processes [18, 19, 32]. It has been used in the past decade to achieve continuous improvement of business processes. Moreover, Process-Aware Information Systems (PAISs) have been widely adopted in organizations to automate their businesses. PAISs such as workflow management systems (WFMSs), Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), or Software Configuration Management (SCM) control the distribution of work, based on a definition of the process in terms of a process model (i.e. workflow model). Typically, the process definition contains tasks (i.e. self-contained pieces of work) between which causal relationships are defined to indicate the order in which they should be executed. Once a process is started, a PAIS creates a process instance (i.e. a case), assigns tasks to appropriate users, and controls the activation of subsequent tasks based on the process definition.

In order to sustain many different businesses, information systems handle the execution of work in various ways. Some systems enforce a given process description, while others enable a more flexible handling of cases, without enforcing a particular process execution. They are configured to support different kinds of business processes. According to their degree of variability, business processes can be classified as:

- **structured**, 
- **semi-structured**, or 
- **unstructured**.

WFMSs proved to be successful merely for structured processes (e.g. service processes similar to the processes encountered in the production and manufacturing industry), while failing to provide the necessary support for more variable situations. The **structured** processes are completely predefined and do not allow for any deviation from the prescribed behavior. Opposite to production-like processes, are processes that focus on the sharing of information and that are data-driven and completely unpredictable. Hence, their structure cannot be predefined before execution. Therefore, they are not suitable for automatic procedures but they give complete freedom to the users. These **unstructured** processes are usually supported by groupware products, known also as Computer-Supported Cooperative Work (CSCW) products [18, 21]. Generally, groupware products are not process aware, and thus they are often integrated with workflow technology to support cooperative
processes. *Semi-structured* processes are processes that fall in between the range of the two extreme cases presented. They require more flexibility than the structured processes, as the execution behavior is more uncertain and they have to deal with exceptions and changes. The process structure modeled at design time acts like a guide for users and it is not a definitive constraint that has to be enforced. Case handling systems like FLOWer [69] have been successfully used for deploying such processes.

As many studies pointed out, designing process models is a complex and time-consuming endeavor [15, 18-20]. Also, most of the times, there is a big difference between the prescribed process model and the real process which is executed. An eloquent example is the execution of semi-structured business processes, where the workers involved can deviate from the specified workflow model. Clearly, these deviations need to be monitored and used as a basis for improvements in process definitions and in the assignment of resources and services that execute activities.

The next section explains how run-time data can be used to extract more knowledge about the real behavior of the prescribed workflow model.

### 2.2 Mining Business Processes

Process Mining has become a known practice to support process analysis and design, and various tools and techniques have been proposed to discover different types of knowledge about their execution [9, 16, 19, 20, 24-26, 29]. It can be seen as a topic strongly related to Business Intelligence (BI), Business Process Analysis (BPA), and Knowledge Management (KM).

The majority of information systems have the ability of monitoring and logging information about the business processes they support. They usually store event data in some structured form. For example, an event can refer to start or completion of a task (also named an activity), a transaction message, some manipulation of a document or it can even refer to some modification of a data object. The collection of event data into a so-called *event log* (or *audit trail*) serves as input for the process mining methods. For example, an event log can look like:

<table>
<thead>
<tr>
<th>case id</th>
<th>activity</th>
<th>originator</th>
</tr>
</thead>
<tbody>
<tr>
<td>case 1</td>
<td>activity A</td>
<td>John</td>
</tr>
<tr>
<td>case 1</td>
<td>activity B</td>
<td>Carol</td>
</tr>
<tr>
<td>case 2</td>
<td>activity A</td>
<td>Sue</td>
</tr>
<tr>
<td>case 3</td>
<td>activity A</td>
<td>Mike</td>
</tr>
<tr>
<td>case 4</td>
<td>activity A</td>
<td>Sue</td>
</tr>
<tr>
<td>case 5</td>
<td>activity C</td>
<td>Sue</td>
</tr>
</tbody>
</table>

**Figure 2: Example of an event log**

The aim of process mining is to extract knowledge from the execution log [19]. However, there are certain requirements that a log needs to fulfill:

- each event refers to a task,
- each event refers to a case (i.e. process instance), and
- events are totally ordered.

An important usage of process mining is *process discovery* (i.e. the automatic discovery of a process model from the behavior observed in the log). The discovered process shows how people and procedures really work. Also, a process model can be used for *conformance checking*. This means comparing the actual (discovered) process with some predefined process that specifies how people and organizations are expected to behave. The discrepancies found can indicate bottlenecks and errors in the process. Additionally, the learned knowledge can be used for process *extension*. Using information about the real events that took place enables for the addition of performance related data to allow for a
more objective and informed way of (re)designing business processes. All three types of process mining are supported by the ProM framework, which is described in the next section. Figure 3 gives the overview of process mining [34]:

![Figure 3: Process mining overview [34]](image)

Process mining strives to gain insight into various process perspectives, not only into the control-flow (i.e. causal relations between tasks). The organizational, data and performance perspectives are also addressed by various mining methods [22, 24-26]. It is important to understand which performers are involved in a case and how they relate to one another. This social analysis enables a good evaluation of employees’ performance. Performance analysis is providing information about throughput time, failure rates, or dependencies among properties of the analyzed case. Moreover, the data analysis is providing useful insights into how data elements are manipulated, or how data influences other perspectives. Data access patterns may reveal hidden behavior which may not be visible in the control-flow perspective.

The data perspective is still not as well developed as the control-flow perspective, but we do consider that it is of high relevance when analyzing semi-structured business processes. In the next chapter we present some modern practices in designing and analyzing processes where data is playing a central role. We intend to stress the significance of data in mining flexible processes and recommend data as a first class citizen in analysis of business processes.

However, the applicability of these distinct methods depends on the amount of information available in the log. The ProM mining framework developed at TU/e is probably the most advanced tool providing process mining functionality. It has been used to carry out the research investigations for this thesis. We introduce it in the next section.

### 2.3 The ProM Framework

ProM is a generic open-source framework for implementing process mining tools in a standard environment. It allows for the application of various mining methods and algorithms implemented as so-called plug-ins. Currently, this framework has more than 190 plug-ins for process mining, analysis, conversion, import, export and filtering.

The mining plug-ins do the actual mining of the event log (i.e. finding causal dependencies between activities). They are used to discover a process model (e.g. Heuristic Miner, Alpha Miner) or a social network (Social Network Miner). Analysis plug-ins are used to take a mining result and start analyzing it. For example, Performance Analysis can be used to indicate bottlenecks in a process; the Dotted Plot can be used to visualize events in the log, and the Decision Point Analysis plug-in to discover how
data can influence the decision points in a process. The conversion tools allow the translation of mining results in different formats (e.g. convert a Petri net into an Event-Driven Process Chain-EPC model). The import and export plug-ins allow various objects to be opened in ProM and “saved as” in different formats. The log filer plug-ins are used to clean the logs (e.g. removing incomplete process instances) or filter the events in the log based on some criteria. A more complete reference of the ProM functionality can be found in [34, 37].

The ProM framework requires event logs in the Mining XML (MXML) format as input [32, 37]. The ProM import Framework has been designed for converting log data originating from various information systems, into this generic XML format. It is a valuable tool that enables many commercial information systems to have their logs analyzed using ProM mining framework. The mining format MXML is described next.

**MXML format**

A mining XML (MXML) file has a very simple and generic structure that can accommodate the description of different kinds of events. The following diagram presents the MXML format with the main attributes and components characterizing the workflow log:

![MXML schema](image)

Every event corresponds in MXML to an Audit Trail Entry (ATE) in Figure 4. For storing event logs in the MXML format, it is essential that the process instance (i.e. the case) where the event was produced and the process definition (i.e. process template) of the process instance are known. They are used for structuring the events. Moreover, there are two mandatory data attributes for describing the event:

- the **WorkflowModelElement** that describes the name of the activity that generated the event
- the **EventType**, which describes the lifecycle transition of the activity (i.e. what kind of event was generated; for example: start activity, complete activity).

Optionally, the originator describing the entity (person or system component) that triggered the event, as well as the timestamp storing the date and time when the event occurred can be set. However, other data considered relevant in the context of a specific system can be also stored using the generic possibility to add an unlimited number of attributes in MXML. This is a very useful feature, allowing MXML to be extended with various attributes, to support the relevant event data generated by different systems.

The ProM framework is a powerful tool designed to assess the quality and performance of the business processes based on logs translated in the MXML format. It has been used in several industry cases [20,
We use ProM in our research for developing a data visualization method to analyze the lifecycle of different types of process elements (described in Chapter 4) and also for analyzing IBM WebSphere logs in the second part of the thesis (Chapter 6). The description of the ProMimport plug-in we have developed to be able to analyze WebSphere logs shows concretely how the conversion of WebSphere event logs to MXML format has been realized. This is presented in Section 6.2.

As we have already indicated in Section 2.2 of this chapter, we consider that the support for data analysis, in the context of process mining, can be further enhanced. The next chapter presents a literature overview of data-driven approaches and motivates our argument that data should be treated as a first class citizen.


3 Data as 1st Class Citizen

This chapter focuses on the role of data in the design, execution and analysis of semi-structured business processes. We intend to emphasize the importance of data in the context of flexible business processes and, as a consequence, promote data as a 1st class citizen in the process mining practice.

In order to understand the characteristics of semi-structured processes, we first give an overview of flexibility mechanisms that address changes in business processes. Thus, Section 3.1 provides a taxonomy of flexibility based on a review of literature. Next, Section 3.2 gives an overview of two distinguished approaches to support data-driven processes: Case Handling and Product Based Workflow Design. Here we explain how data, as a central concept, is used for process design and execution, and how this increases flexibility and adaptability of administrative business processes. However, due to the variability of execution, process analysis becomes essential to ensure correctness and consistency of the process behavior. A thorough analysis of the real (observed) behavior contributes to achieve better monitoring and control. Therefore, Section 3.3 discusses the challenges and opportunities in mining the data perspective as part of process analysis. Finally, in Section 3.4 we conclude by emphasizing the role of data in mining flexible business processes and elevate data to the same level of importance as the control-flow perspective.

3.1 Flexibility Taxonomy

Over the past years, workflow management systems have been successfully applied to support well-structured business processes [9, 15, 16, 18-20]. An increasing number of organizations use workflow technology to automate their business processes in distributed and heterogeneous environments, facing continuous and unanticipated changes. However, most of the traditional workflow systems are too rigid to cope with change. This is caused by the fact that often the real runtime process is much more variable than the process specified at design time. This is usually the situation of semi-structured processes, which we introduced in Section 2.1. Among the many concerns identified by [21, 39, 44, 49], we would like to mention the following two problems:

- it is not always possible or desired to prescribe at design time all parts of a workflow (either because they are not known or because they are too complex)
- unexpected changes may occur during the execution of a workflow

Therefore, the need for enhanced flexibility was noted by many researchers [38-53]. Flexibility is not related only to the business processes themselves, but also to the information systems that support them. The notion of flexibility has been largely debated and many researchers proposed various definitions. According to [53, 54], flexibility is the ability to yield to change without losing identity: a business process is flexible if it is possible to change it without replacing it completely, and thus changing only those parts that need to be changed and keeping other parts stable. Based on [39, 49], two main approaches to achieve flexibility have been identified: flexibility by configuration (selection) and flexibility by adaptation.

3.1.1 Flexibility by configuration

Flexibility by configuration provides a certain degree of freedom in executing a workflow by offering multiple alternative paths. After all, flexibility refers to the ability of making choices. Many changes
can be avoided if a large number of execution paths are either known or anticipated. These paths can be included in the process model by providing powerful and expressive modeling constructs.

**Mechanisms to implement flexibility by configuration**

According to [49] there are three mechanisms to implement this type of flexibility:

- *Advanced modeling* (before runtime)
- *Descriptive modeling* (before runtime)
- *Late modeling* (during runtime)

The *advanced modeling* method is achieved at design time (i.e. when making the process model). The degree of freedom is explicitly or implicitly defined. The explicit definition is made by using advanced modeling constructs that enable more execution paths. Also, implicit flexibility is achieved by extending the user operations at the process instance level. For example, new actions support him to deviate from the normal flow in exceptional situations (e.g. skip, redo, stop or delete) [40, 49].

Implicit freedom can be provided also by means of *descriptive modeling* [49]. This technique assumes that modeling constructs (e.g. explicit choice) can be omitted from the workflow definition. Thus, rather than enforcing control flow through a rigid model that tries to capture every step and every option in the process, the process model is defined in a more relaxed manner. The user has the freedom to execute individual instances according to the situation at hand. This approach is also used by declarative languages such as DecSerFlow: a Declarative Service Flow Language [89]. The aim of this declarative style to avoid the over-specification of a process model, by modeling only what is strictly needed.

The *late modeling* method assumes a loosely or partially specified process model, where the full specification is made at runtime. This way it compensates for the inability to completely define the process model at design time due to situations that cannot be anticipated. This type of flexibility mechanism is implemented by [41, 42, 46], among others. The idea of this method is to defer the modeling of certain parts of the workflow to the execution time. These parts are usually named “black boxes”. The black box constructs need to be part of the workflow execution language. They represent a collection of workflow fragments, where each fragment can consist of a predefined activity or sub-process. A critical requirement of this method is the verification of the validness of the fragments selection and composition. This is necessary to assure the semantic correctness of the composition in relation with the process under consideration [42].

Flexibility by configuration is useful when the requirements for flexibility in the business process can be identified in advance. The disadvantage of this approach is that incorporating more constructs within the modeling language can increase the complexity of the process model, making it difficult to manage and verify. Additionally, it cannot handle unforeseen circumstances that require change at the process instance or even at the process type level. For these situations flexibility by adaptation is needed.

### 3.1.2 Flexibility by adaptation

Flexibility by adaptation is addressing situations where unforeseen changes are affecting the execution of a workflow. According to [39, 56], the nature of these changes can be:

- *ad-hoc* (triggered by meeting short term goals)
- *evolutionary* (triggered by meeting long term goals such as strategy, laws, stakeholders needs)

Ad-hoc changes are handled on a case-by-case basis, while the evolutionary changes are affecting the structure of the process model itself.

In the case of an evolutionary change, the migration of the running instances to the new workflow definition becomes a problematic issue, because for the process model it is necessary to consider:
• **version** (a new definition of the workflow, that enforces all cases to be executed according to it)
• **variant** (another valid form of the same workflow definition, that allows cases to be executed according to different variants at the same point in time)

More clearly, a change of the process model may result in a new version or a new variant. Either way, the running instances have to be adapted to any of the two.

**Mechanisms to implement flexibility by adaptation**

Related to the two types of change, two mechanisms to implement flexibility by adaptation are considered:

- **instance adaptation** (corresponding to ad-hoc change)
- **type adaptation** (corresponding to evolutionary change)

In general, changes with a momentary effect determine the **instance adaptation**. The process is adapted for one or a group of cases. Exceptions (i.e. undesirable unexpected events) often generate changes in the process instance execution by means of exception handling techniques. Mostly they reflect errors at the system or application level, while human errors or deliberate changes are omitted. Generally, ad-hoc changes result in many variants of a given workflow definition running in parallel.

The **type adaptation** results in a new version, as a result of an evolutionary change. New instances are executed according to the most recent version of a workflow. The biggest problem is the migration of running instances (work-in-progress) of the old to the new specification. Consider, for example, the situation when a workflow definition must be changed due to new laws. When the change has to be implemented, hundreds or even thousands of instances may still be running. Aborting these instances or letting them run under the old workflow specification often is inconvenient or just impossible. Many times the active instances must comply with the new constraints. This is referred in literature by “dynamic change” [39, 45, 50, 56]. Achieving compliance of the affected instances with the new workflow version may lead to loss of work, errors and inconsistencies, so it has to be carefully managed. A few approaches described in literature attempt to deal with problems related to dynamic change. [45] provides a set of policies and strategies to be followed when managing evolution of running instances. An approach based on inheritance of workflows is described in [38, 50]. Frameworks implementing flexibility by adaptation are, for example, the ADEPT WF Model [40] and OPENflow system [47].

Besides the dynamic change, [50, 56] point out that other related issues such as correctness and management information (i.e. aggregated information about the state of the business processes) should be also carefully assessed when dealing with change. Furthermore, [60, 61] sustain that other exceptions, resulting in either ad-hoc or evolutionary change, are almost impossible to be resolved using exclusively technological measures, without external (e.g. organizational) compensative actions. Generally, these changes are generated by organizational or political factors.

### 3.1.3 Conclusions with respect to flexibility

Several approaches and mechanisms are available that deal with the problems related to flexibility. While flexibility by configuration is easier to achieve, flexibility by adaptation still requires more effort to implement, especially in the case of process model modification. The flexibility issues presented address mostly the design and the execution perspectives of workflows. However, many flexibility drivers and requirements can also be found in the context of the business processes, which may include among others time, location, legislation, culture, performance or organizational requirements [44, 51, 53, 60, 61]. Up to now, these aspects and their implications on the information systems have not been investigated in thorough manner.

The data or informational perspective is poorly addressed in the discussions about flexibility mentioned so far. The informational perspective is closely linked to other process perspectives (e.g.
process control-flow, organization, operations). For example, case handling systems are both data and control-flow driven. Sections 3.2 and 3.3 describe some of the most representative data-driven approaches that strive to achieve more flexibility. Data in a process is related to control (i.e. routing variables), or to the production of information objects (e.g. forms, documents). When one workflow perspective is affected by change, then the change possibly propagates to the linked perspectives as well. For instance, the implementation of the late modeling technique should assure that the composition of workflow fragments maintains the consistency of the data and doesn’t loose important information necessary at a later point in the execution of the instance. Data also plays an important role when selecting the fragments to be composed into a process. The evaluation of the business rules involved in these selections depends heavily on availability and accuracy of data. With respect to the flexibility by adaptation, things get even more complicated. Every unexpected change poses big risks on the data associated to the instance or the process model affected. Missing variables or new data introduced that does not comply with the old requirements may trigger complex problems or situations that are very time consuming and difficult to resolve. The close interrelation between data and control is often neglected or superficially treated and, consequently, the role of data in achieving flexibility is undermined.

In our research, we propose to strengthen the position of data in the context of business process flexibility. The next sections will show the implications of data both as a flexibility driver and as means to analyze and improve the business processes. Thus, we advocate the treatment of data as a 1st class citizen in supporting business processes through their entire life cycle, from design to diagnosis phase.

### 3.2 Data-driven Approaches

This section discusses situations and approaches where the design or execution of a business process is data-driven. The purpose is to show how data can be used to achieve more flexibility for business processes that are information intensive.

In semi-structured business processes, availability of information is often the main constraint on whether or not activities can be started and executed. Knowledge intensive domains such as government administration, financial and legal services rely heavily on data that is being processed, because in such an environment, the outcome of a workflow is often an information object (e.g. a decision or a document). Knowledge intensive business processes (e.g. administrative processes) are only partially supported by the process model and require more flexibility due to unpredictable decisions or tasks guided by creativity [63, 65, 68]. This uncertainty depends on the users executing activities and the context of the process execution. Typically, knowledge flows and knowledge transfers between media and persons are necessary to achieve a successful process completion [68].

[69] pinpoints several differences between the traditional production processes and the administrative (informational) processes, such as different understandings of product quality and frequency of loops or rework. As a consequence, manufacturing concepts should not be directly transferred to the dynamic environment of informational processes. However, certain principles of the production processes could be applied to administrative processes. As an example, a product-centric approach is proposed by [14, 69, 73, 74] in designing workflows.

Since the traditional workflow systems are ill-suited for the semi-structured nature of these processes, many studies investigated ways to better cater for this kind of processes. [60, 61, 64] investigate the different perspectives of exceptions in office operations to derive better exception handling routines and decision support. Other approaches [14, 21, 58, 69, 70, 73, 74] focus on the data flow inside a process in order to provide means to increase the process flexibility. They argue that the control flow should not be the only driver of a process execution, but also data can be used as an enactment mechanism. Because of the special attention given to data, we present two representative approaches:
Case Handling and Product-Based Workflow Design. This way we highlight the role that data plays in achieving more operational flexibility.

3.2.1 Case Handling

[21] introduces Case Handling as a new paradigm for supporting flexible and knowledge intensive processes. The main goal of this approach is to overcome the inherent lack of flexibility of tightly prescribed process models and better support the informational processes by focusing on what can be done rather than what should be done to achieve a business goal.

Considering the characteristics of the knowledge intensive business processes presented above, it is obvious that highly standardized procedures are not efficient and that processes must be more fluid and adaptable. [21, 69] identify four main problems induced by the exclusively control-flow based routing of the tasks:

a) Distribution of work among resources requires it to be partitioned (straight-jacked) into activities. Activities are assumed to be atomic for the WFMS. However, workers perform the actual work at a more fine-grained level than prescribed by the process model.

b) Authorization and distribution of work is not distinguished by the WFMSs, such that workers are offered to execute any activity that they are authorized to do. As a consequence, the work lists of employees with higher positions get overcrowded as their roles may include many others.

c) The control-flow focus of workflow execution hides the context of tasks to be executed as case related data created at earlier points in the process is not accessible to the worker. This results in the so-called context tunneling effect, which impedes the efficiency and quality of work.

d) The rigid, push-oriented characteristic of routing tasks hinders the level of control and freedom of the knowledge worker in taking decisions, forcing him to often bypass the system and thus increasing the risk of more errors.

In order to overcome these problems, the case handling approach puts more emphasis on data, considering both data and processes as first class citizens. This change of focus settles the premises for better support of knowledge intensive business process. With this new perspective, [21] positions case handling in the spectrum of approaches for process support:

![Figure 5: Case handling in workflow spectrum [21]](image)

We briefly present the main principles of the case handling concept. The main purpose is to stress how the paradigm shift from control flow to data solved the problems mentioned above. For a detailed definition of the approach we refer the reader to [21, 69, 71]. The main principles of case handling, which differentiate it from the traditional workflows, are as follows:
i. **Focus on the case**

The key idea of this approach is to focus on the whole *case* as the product that is manufactured, and not on the routing of the activities. *Activities* represent chunks of work (e.g. filling out an electronic form), that do not necessarily need to be treated as atomic units, as imposed by workflow management systems [18]. An activity can contain more than one task and, therefore, points of work transfer to another worker can be non-atomic. *Forms* are used to present the case data to the user in different views. To avoid context tunneling, the view should not be restricted to data relevant for executing the next task in line, but it should provide as much information about the case as possible. Activities can be linked to a form to present the relevant data objects, while overview forms can be linked to the case itself. This way a worker can view information provided by his co-workers or identify what tasks still need to be processed.

ii. **Data as primary driver**

Opposite to workflow management, case data is considered to be the primary driver. The produced data is used not only for routing but also for determining which parts of the process have been already accomplished. Data objects can be associated to activities, but the relationship is not a strict dependency. They can be mandatory (the respective activity cannot complete if the data object is not defined), restrictive (only the respective activity can provide access to this data) or free (no dependency between data and activity). This mechanism decouples case data from activities.

iii. **Types of roles associated with tasks**

The *execute* role specifies the authorization for executing a task. However, real case work might involve skipping and rework. To support these situations, *skip* (bypass) and *redo* (re-execute a previously completed or skipped activity) roles are added for tasks. The skip role allows the user to postpone the handling of certain activities or bypass them if they seem to be unnecessary for the case at hand. If a task needs to be undone, the redo action requires that all subsequent tasks that have been completed or skipped to be undone as well. The redo role replaces to some extent the explicit modeling of loops. The new roles enable a more natural and less strict navigation of the case activities.

iv. **No separation of case data and control**

Similar to tasks, data objects have also a life-cycle with states and state transitions (Figure 6). The state of data elements fully determines the state of the case. The meaning of the first two states *undefined* and *defined* is straightforward. An important data object state is *unconfirmed*. This is related to the redo action of a user when he tries to re-execute a previous activity. It allows the values of these data objects to be preserved and used as templates when re-executing the task, instead of discarding them. The new values would have to be confirmed in order to complete the task. As a consequence, the data life-cycle transition model establishes a direct link between data and user roles.

![Figure 6: Data object states [21]](image)

The case progresses by adding and modifying case data. By filling out a form zero or more work items may be executed. A form linked to an activity may contain data entries that are not
mandatory for the respective activity, but it allows the user to see or modify data from earlier points or define data necessary for an activity that may appear later in the process execution. In this way, activities can auto-complete before their actual execution if the required (mandatory) data is provided earlier in the process.

v. Separation of authorization and distribution
The distinct roles execute, skip and redo provide different levels of authorization. The distribution of work is based on a query mechanism of the cases to get a work-list. A user can select the cases that interest him and start performing his work, or he can view or modify cases without explicitly selecting a work item.

Discussion
The major characteristic of this approach is that the case is controlled mainly on the basis of the states of data objects. The case evolves based on the availability of data. Furthermore, decoupling data from activities increases the flexibility. Even if it is not mandatory, data can be provided early in the process if the information becomes available, and thus speed up the execution of the case. Also, having access to case data (assuming proper authorization) that goes beyond the narrow scope of a single task reduces the amount of rework as the knowledge worker is expected to take more informed decisions. Moreover, authorization and distribution are typically directly linked to the process model. By defining authorization and distribution independently and separate from the process definition, reuse and flexibility are increased.

The same benefits are gained by the separation of the execute, skip and redo roles. These constructs not only give more power to the worker but also make the process more adaptable to changes. This means that the (authorized) user can easily deviate from the prescribed process model without affecting it or going behind its back. Having the means to skip or redo activities supports the user in reacting efficiently to exceptions and in accommodating the necessary changes. This type of flexibility was introduced in the previous section as flexibility by configuration, with advanced modeling technique.

If every person allowed to work on the case is mapped on the execute, redo and skip roles for each task, then there are few constraints for the processing of the case. Implicitly, almost all execution paths are possible. On the other hand, if all data objects are only mandatory and restricted for activities and no person has the skip or redo role, then the resulting flexibility is almost comparable with that of the common workflow management systems. More clearly, due to the advanced concepts of user roles and detachment of data from activities, the model doesn’t need to be precisely modeled at design time, incorporating all possible execution paths. A more relaxed and less complex model might prove to be even more comprehensible and easier to handle. This implies the flexibility by descriptive modeling.

In a dynamic, knowledge intensive environment, adaptability, support for work and support for information are essential. Knowledge workers need to have more power to handle the cases and more knowledge about the context surrounding the flow of work. As empowerment increases, so does the organization’s potential for process change and requirements for flexibility. Rather than prescribing what must be done, the system should leave it to the user’s judgment what activity is necessary, whether it should be bypassed or redone. Case handling is designed to provide a high degree of flexibility to employees and implicitly to the business processes themselves.

Recognizing these benefits, organizations such as Pallas Athena and IBM consider the opportunity to incorporate case handling functionality into their process management suites. The IBM prototype CHEF mentioned in the introduction of this thesis aims at providing case handling support by using available off-the-shelf IBM technology. A description of CHEF is provided later, in Chapter 6 of the thesis.
3.2.2 Product-Based Workflow Design (PBWD)

A similar trend of taking data as a 1st class citizen can be seen in Product-Based Workflow Design (PBWD). PBWD is a method to design and redesign administrative business processes – also known as workflows - inspired by manufacturing principles [14]. A typical characteristic in manufacturing is that the structure of the product is used to determine the steps needed to manufacture it (i.e. the process) [73, 76]. Administrative processes produce products as well. The manufactured product in a workflow is usually denoted as the case that is processed. A case corresponds to a service to the environment (e.g. a decision) [74]. For example, in government, banking and insurance companies workflows produce tax assessments, purchase orders, credits and health insurances.

However, current workflow management systems maintain a loose coupling between the product notion and the process structure. This means that the business process is usually designed without much consideration for the product. It has been argued in [14, 73, 74] that administrative business processes can also be derived based on the product specification and that the product should be the focus of the (re)design efforts rather than an existing process.

In order to show how data is used as the key driver in the process design efforts, first, we present the mechanisms that lay out the basis for the workflow design. By adopting the logistical principles in PBWD, the product becomes the central point in deriving the structure of a process. Therefore, the specification of the workflow end product needs to be analyzed to identify the data elements, their dependencies and the logic involved. The workflow product is represented by a Product Data Model (PDM), which is a network structure describing the construction of the product [14]. This is similar to the static representation of a product in manufacturing, referred to as the Bill-Of-Materials (BOM), which is used for material requirements planning and inventory control [76]. This is explained in more detail in the next section.

Product Data Model (PDM)

As aforementioned, the structure of a product in production environments is defined using a so-called Bill-Of-Material (BOM). This specifies what materials and components are necessary to manufacture the product. We will illustrate this using an example given by [14]. Figure 7.i) shows the BOM of a car and the structure of the assembly line. The car is made of a chassis, 4 wheels and an engine. We can see that first the chassis is assembled with the 4 wheels, resulting in a subassembly. After this, the final product is produced by putting together the subassembly and the engine.

In a similar manner, a product in an administrative process is defined by a Product Data Model (PDM). In this situation, the product is an information object, whose internal structure is made up of information elements (e.g. client history, balance, salary). The structure of the PDM is graphically represented by a graph where circles represent the data elements, while the arcs represent operations on these data elements. For simplicity, we present a simple example given by [14], where the process data model shows how the decision whether a candidate is suitable for becoming a helicopter pilot is structured. This is shown in Figure 7.ii).

![Figure 7: i) The BOM of a car; ii) The PDM of suitability to become a helicopter pilot [14]](image-url)
The meaning of the data elements (labeled with $a$, $b$, $c$, $d$, $e$, $f$) is the following:

- $a$ – suitability to become a helicopter pilot
- $b$ – psychological fitness
- $c$ – physical fitness
- $d$ – latest result of suitability test taken in previous two years
- $e$ – quality of reflexes
- $f$ – quality of eye-sight

Figure 7.ii) indicates that a person can become a helicopter pilot if certain requirements are met: they should pass a psychological test, be in a good physical condition, have a good eye-sight, and they should not have been rejected in the past two years. We can see already that there are different conditions that can determine the result of the evaluation procedure. If the candidate’s eye-sight is bad ($f$), then it will automatically determine a negative result, independent of the values of other data elements. Also, the result of a recent evaluation ($d$) will directly determine the result of this evaluation as well. Otherwise, the results of both a psychological test ($b$) and a physical test ($c$) are needed. The physical test result ($c$) is based on two other tests: eye-sight ($f$) and reflexes ($d$).

The product data model shows that the outcome of the helicopter pilot test (data element $a$) is a decision which depends on either both data elements $b$ and $c$, or on $f$, or on $d$. The availability of data elements determines which operations can be executed and hence what new data elements can be defined. For instance, once elements $e$ and $f$ are known, then $c$ can be produced, as a result of an operation upon the two initial elements. Moreover, constraints to the operations can be added. For example, the quality of eye-sight ($f$) can influence the final decision only if the value of $f$ is “bad”. Otherwise, the operation should not be executed. Therefore, operations can be executed only when the required input data is available and when the values of these input elements satisfy the conditions associated to the respective operations.

Although the BOM idea is taken as a source of inspiration, there are several differences between a PDM and a BOM model. First of all, a PDM does not contain cardinalities for the data elements, as it is not considered appropriate to represent in the model that the same data element is produced multiple times. Note that a data element can be used multiple times. However, this limits the applicability of multiple instance patterns in the process. Second, constraints, cost, flow time and probabilities are associated with production rules [14]. Production rules determine how operations can be executed to produce a new informational element. Constraints specify the circumstances when a production rule can be applied. Seen as part of the product specification, they are based on the data values used to produce a new information element. The other attributes related to cost, flow time and probabilities are representing design criteria. They can be used as optimization parameters in performing the operations, aiming at obtaining an efficient design model.

We want to stress here the strong similarity with the case handling approach. In Section 3.2.1 we explained how data is used to drive the case execution. The availability of data elements determines the enabling and completing of the tasks of the business process. The same importance is given to data in the PBWD approach. The availability of data triggers the execution of the operations towards the production of the end product.

Due to the strong link between the two approaches, [70] made a study of two case handling systems (FLOWer and BPI’s Activity Manager) to assess the level of support they can provide to PBWD. They concluded that FLOWer is the closest to support the idea of the product-centric design of a workflow. Even so, case handling systems are still not suitable to fully support the new design approach, as they do not offer the means to display and edit a PDM structure in the tool and activities are still directly related to data elements.
The PBWD method has been applied in some industry cases [14, 75]. Nonetheless, the lack of tool support made its adoption rather limited. Nevertheless, it seems obvious that PBWD has several advantages that sustain the benefits of applying logistical principles to designing administrative processes [73, 75]:

i. **Radicalism**

PBWD is a revolutionary approach to workflow process design; namely, a clean sheet of paper is taken to (re)design the process from scratch. The opposite approach of using an existing process as basis for redesign, allows for errors and inefficient constructs to be taken over in the new design. Hence, the radical method PBWD counteracts the inheritance of undesired constructs that hinders the performance improvement objective of the redesign efforts.

ii. **Objectivity**

The method takes the product specification as the starting point to derive the minimal number of required tasks to produce the product. Since every information element and production rule can be verified and justified with the product specification, no irrelevant information or unnecessary tasks are included in the new workflow structure. Furthermore, the ordering of the tasks is completely driven by the performance targets of the design.

iii. **Analytical approach**

PBWD is based on explicit recognition of design parameters and degrees of freedom for a new process definition, contrary to participative design methods that involve group of experts to yield a new business process. The deliverables of the PBWD approach reflect information about the business needs that could be useful for systems development. In this way, the workflow model and the systems can be better integrated.

iv. **Focus on the product**

Taking the product as the starting point in the design of a business process, can help stakeholders to reach consensus and, at the same time, reduces the vagueness of a high-level design. The clear and objective representation of the product gives the necessary information for the design, implementation and maintenance of the new workflow model.

**Discussion**

PBWD is a prescriptive methodology that fits well with the Business Process Redesign (BPR) paradigm. Assuming a clean sheet approach, it derives a workflow structure that consists of tasks that retrieve or process data elements. Following the manufacturing principles, PBWD works backwards: the end product is taken as starting point and unraveled into necessary processing steps to produce the end product.

No existing process model or reference model is taken into account. The workflow product is specified in terms of informational elements needed for its production. First, the relevant data is determined, after which the processing steps are defined on the basis of data manipulations. The PBWD was also studied in the context of case handling support by [69, 70]. Since both approaches are data-driven and the focus is on the end product of the workflow (i.e. the case), it seems a good idea to investigate how workflow management and business process design can strengthen each other.

We have already showed that the data-centric view of case handling systems allows them to be more flexible in dealing with exceptions or changes that occur in the business process. Since the product is the central point of PBWD and is tightly connected to the workflow, if changes in the product specification arise, then it is clear what parts of the workflow are affected by these changes. Minor changes in production rules could be easily incorporated in the workflow design. Major changes affecting the dependencies of the product data model may result in new workflow specifications. Handling these evolutionary changes can still benefit from the PBWD approach because it provides more flexibility and adaptability of the business process. [74] even proposes the use of product families concept in workflows to increase the adaptability. Product families are based on derivations
of the BOM, allowing a range of product types to be modeled based on a generic structure. Hence, using generic workflows that can be extended with process families, improves the ability of the processes to adapt to changes. This idea was extrapolated in [38, 50] where an approach based on inheritance of workflows allows the use of generic process models.

Our analysis of the PBWD revealed also several weak points that need to be further addressed. [73] stresses that a clear product specification is critical, as this is the basis of the approach. Evidently, this requires an intensive effort in the analysis of the product specification, in building a formal design method, and evaluation of the workflow model. Another drawback is that usually, people involved in the workflow design still have the existing process in mind, which may affect the radical, clean-sheet approach.

3.2.3 Conclusions

The rapid expansion of automation of business processes using workflow technology replaced the initial focus on data with process-centric approaches like Business Process Re-engineering (BPR) and Continuous Process Improvement (CPI) [18]. However, the applicability of workflow management systems in dynamic environments is still limited. While the repetitive nature of manufacturing-like processes fits well with the classical workflow support, the semi-structured or unstructured processes are not properly addressed by the classical workflow paradigm. Adaptability and flexibility are stringent requirements for processes that are highly dependent on information.

Recognizing the limitations of traditional workflow technology, several approaches emerged to better address the needs of administrative business processes [14, 21, 69, 71, 73, 74]. These studies realized that knowledge intensive business processes can not be driven by routing work along activities only, but data should be also considered as a key-driver. We have presented two different types of approaches that elevate data to the central point of the business process design and execution. The paradigm shift places the case (i.e. the product of the workflow) in focus, at the same level with the routing of activities.

The Case Handling approach demonstrates how data can be used to determine the evolution of the case. By focusing on data, systems are able to deliver an evolving way of doing business, instead of restricting the execution to a set of predefined rules. Moreover, the unique constructs offering more power to the case workers provide implicit flexibility. The business process is thus able to adapt over time in line with the user’s needs. The underlying flexibility allows the worker to step outside the boundaries of the prescribed workflow model and take the necessary actions to better handle the case.

The second method we presented addresses the design of business processes. The Product-Based Workflow Design takes a similar position as case handling concerning the role of data. The method advocates that the product structure should drive the design of the business process. In administrative domains, the product is an informational object composed of different data elements. Analogously to case handling, the availability of data objects drives the execution of the operations according to the production rules, with the goal of producing the ultimate product. Furthermore, the close relationship between the process and the product specification enhances the flexibility and adaptability of the business process. This is because changes in the product specification are better tracked, understood and transferred to the business process model.

Nevertheless, the methods we described are not the perfect solutions. Improving flexibility and adaptability always has a side effect on the control of the business process. The situation of conflicting constraints in achieving flexibility is known as the flexibility paradox [72]:
The more flexible processes are the more support and control they need. With much more behavior allowed, control is necessary to ensure the process correctness and compliance with organization regulations.

The biggest disadvantage of the data-driven approaches is that they increase the risk of inconsistency and redundancy in maintaining and handling large amounts of data. This may negatively affect the efficiency and quality of the process execution. In order to reduce the context tunneling effect in handling administrative processes more data is necessary to provide a broader perspective on the entire case. However, increasing the amount of data is generating the problems we just mentioned. Concurrency is also an important issue that has to be carefully considered when accessing data. The case handling system FLOWer introduced a system of locking: case data is completely locked by the current user and released only at log-off [72]. The downside of this method is that it prohibits simultaneous handling of a single case by different users, and therefore affects the performance. However, in FLOWer, multiple users are allowed to view case data. For PBWD, the ability to collect data for the product data model and the existence of a clear product specification are vital for a successful workflow design. An incomplete or inaccurate set of data leads to another inefficient design. Finding the relevant information is definitely not a trivial task.

Moreover, the author of [74] argues that data alone cannot run a business process. Instead, both the product-centric view and the process-centric view are needed. For instance, distribution of work over the resources involved in a case cannot be expressed by a product data model, but is very important for the logistical control of the business process. Therefore, a trade-off between flexibility and control is necessary.

However, we consider that placing data in a central position in designing and executing administrative workflows enables more operational flexibility and support for the knowledge workers, as well as enhances the adaptability of the business process. The two data-driven approaches presented above prove that this paradigm shift can overcome the limitations of the traditional workflow technology and better address the needs of less structured business processes. Additionally, the successful application of the case handling approach in industry triggered a new wave of data-aware solutions. The market overview produced by [66, 67] shows an increasing demand for Business Process Management Systems able to support business processes that have a high level of information and human involvement.

In conclusion, we advocate that the role of data in semi-structured business processes should be recognized as a primary one, and therefore addressed properly by workflow technology.
3.3 **Mining the Data perspective**

In the previous two sections (Sections 3.1 and 3.2) we presented how data plays an important role in providing more flexibility in modeling and executing business processes. At the same time, we made the observation that the more flexible the business processes are, the more control is needed to ensure a correct behavior. Process mining contributes significantly to process improvement and can be used for process control. By discovering the *actual* behavior of processes, process mining enables a better alignment of the real-world business processes with the information systems that support them. Therefore, we propose to investigate how data can be used in process mining to gather more knowledge and control of semi-structured business processes.

### 3.3.1 Importance of monitoring and mining data

We argue that data plays a major role in process mining and that more mining techniques should address the specific needs of the informational perspective, because of the following reasons:

- Process mining methods have to deal with *richer and more complex data*
- Emerging approaches consider data as the *central point for modeling and execution* of administrative business processes (e.g. Case Handling, PBWD)
- Data is *one of the key-drivers of flexibility* in knowledge intensive business processes (e.g. enables change and adaptation to exceptions)

Further on, we give more explanations for these reasons.

The requirements imposed by dynamic environments upon today’s information systems determine an increase in systems complexity as well as in the amount of data produced. Moreover, [27] mentions that the Business Intelligence tools market faces a demand for supporting more heterogeneous data (e.g. structured, unstructured, semi-structured, rich media information). [28, 77, 79, 83] stress the increasing volume of data in business environments and the acute need for intelligent data analysis to keep up with this trend. According to [83], many businesses are drowning in data. The problem is that they can not make sense of it all and can not turn it into useful information. Therefore, to address this challenge, *process mining methods should focus more on the data perspective.*

Additionally, data-driven approaches (e.g. case handling, PBWD) put data on the central place for modeling and executing administrative processes. This increases both the quality of the workflow product (i.e. information object) and the flexibility of the process, due to a better alignment of the process model and the product manufactured. The close link between design and execution should be carefully monitored in order to derive better redesign solutions.

Moreover, flexible processes driven by the availability of data require more control to assure a correct execution. Only a thorough analysis of data could provide the necessary knowledge to comprehend when and why changes or exceptions occur, and thus help learn and reuse this information for better support of flexible processes.

We continue to present some existing approaches of using data for extracting more information about the real behavior of business processes.

### 3.3.2 Classical data mining and data visualization

*Data mining*, also known as *knowledge discovery*, has been defined by [79] as the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. It involves searching through large amounts of data in order to obtain relevant information. Over the past decades, it has been successfully applied in areas such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, fraud detection, medical diagnostics and computational biology [77, 78, 80, 81].
In organizations, data mining evolves as a major component of business operations to support business processes [77, 79]. Tools combining various data mining concepts such as decision trees, association rules or clustering are, for example, Weka [82] and Microsoft SQL Server Data Mining Platform [81]. Using integrated tools that address several data mining techniques can bring more benefits to an organization rather than a standalone data mining application. The knowledge extracted using the individual data mining methods can be used not only for data analysis, but also for data integration and reporting. As an example, discovered associations in data attributes can help identify unacceptable data (e.g. inconsistent, wrong format, outside acceptable boundaries) when integrating data from distributed sources [81]. Eliminating this noise (i.e. erroneous or missing information) can help realize a more accurate and meaningful analysis of the remaining data.

Moreover, the need for more interactive mining tools has been recognized by [28, 79, 81]. According to [81] one limitation of data mining techniques is related to the interpretation of the mining results. Although data mining can help discover patterns and associations, it can not establish the significance and relevance of these results. The user has to use his domain knowledge to understand the meaning of the results, and therefore he needs to participate in the data mining process. Supporting this idea, [79] mentions the usefulness of interactive clustering algorithms that combine the computer’s computational powers with the human user’s knowledge and visual skills. The human ability to identify useful clusters (i.e. groups of similar data objects) appears to outperform the computer’s ability, when visualization of data is possible and data dimension is low.

Another drawback of data mining tools is that often they are too complex to use and accessible only to expert analysts. As a consequence, most business decisions are based not on a data-driven analysis, but simple rules of thumb [77]. To compensate for this limitation, data visualization tools are used to accompany data mining techniques. As [28] points out, the main advantages of visual data exploration are:

- visual data exploration is intuitive and requires no understanding of complex mathematical and statistical algorithms or parameters
- visual data exploration can easily deal with heterogeneous and noisy data

Naturally, exploiting the human perceptual capabilities can help reveal tacit knowledge that automated techniques are not able to discover. Therefore, data mining can be more efficient if the user is also involved in the data exploration process.

Furthermore, to support business processes, organizations use data mining tools integrated with other IT tools such as databases, version control and workflow management systems. This enables both an improved data analysis and reuse of information. Special attention is paid in mining structured data like graphs, trees or relational databases. Powerful data mining techniques such as graph mining and sequence mining are used to find a good description of existing data [90]. Still, data mining techniques do not provide sufficient support for analysis of business processes. A different approach of mining data from the process perspective is necessary to understand the influences of data upon the process behavior.

### 3.3.2 Mining data in the context of business processes

We have seen that the traditional mining of data has been successfully used to determine relationships, predict future events, spot bad data or describe data structures (e.g. graphs, trees, molecules). However, the majority of data mining techniques mainly focus on data and do not consider the process perspective in their analysis. Therefore, it is difficult to gain insight into the behavior of a process and locate problems such as bottlenecks or deviations from the prescribed process definition. Mining data in the context of a process driven analysis can help extracting more knowledge about the process execution. This is important especially in knowledge intensive processes, where data is a key driver in process execution and in achieving operational flexibility.
Most of the process mining efforts concentrate on the control-flow of a business process, whilst only few address the informational (i.e. data) perspective [15-16, 18-20]. Moreover, the majority of the process mining algorithms can deal only with logs that contain information about activities and process instances events. By discovering the order and dependencies of activities execution, a process model can be reconstructed from the log. However, many information systems execute processes without having a clear notion of an activity. These are usually unstructured processes. For instance, document management systems are aware of documents and the changes affecting them, but are not aware of activities of the underlying process [22, 59, 85]. There are many examples of information systems supporting unstructured processes, such as Product Data Management (PDM), Software Configuration Management (SCM), or Enterprise Resource Planning (ERP). The logs of these systems contain information about organizational and informational perspectives (e.g. documents, files, data modifications), but they typically do not contain structured activity logs. Obviously, the applicability of the mining algorithms depends on the quality and characteristics of the log. Therefore, new process mining techniques are needed to deal with these situations.

We present some interesting mining approaches that have a strong focus on data perspective and also try to address the challenges we have just mentioned.

### 3.3.2.1 Decision mining

A powerful application of data mining in the context of business processes is decision point analysis [24]. Decision point analysis investigates which properties of a case (i.e. process instance) might lead to taking certain paths in the process. The method was implemented in the process mining framework ProM [34] as the Decision Point Analysis plug-in, to directly support the data analysis.

The main idea of this method is to identify all decision points (i.e. points where a process is split in alternative branches) in a process model and investigate how data attributes actually influence every decision made. The data mining concept of decision trees [84] is used for classifying the decisions in the process. Moreover, the decision mining algorithm uses not only the process specific data, but also information about resources and timestamp of activities, and even additional information available from other mining activities. For example, performance analysis results may provide information such as service time and waiting time. The new information can be used for decision point analysis.

The main requirements for applying decision point analysis are a process model and an execution log containing activity events and data objects. There are no specific requirements about the data attributes. The ultimate goal is to identify if cases with certain data follow a specific path in the process.

For structured processes this technique may help discover the business rules defined for automatically selecting paths in a process. It may also discover how these rules are actually applied, and thus provide a good support for the process redesign. In data-centric semi-structured processes however, decision mining can reveal more interesting insights of the way work is performed. Behavior that is not explicitly modeled in the process or user actions that are triggered by certain information can be more easily explained and understood by analyzing the data flow. For example, the amount of money involved in an insurance claim might constitute an important factor for the decision to approve the claim. Observing how process’ behavior is influenced by this data object may uncover inefficiencies (e.g. long waiting time or execution time) or even prohibited practices (e.g. fraud, inside employee attacks, breaches of authorization rules). Thus, the benefit of this data analysis becomes obvious in the context of conformance verification and process improvement.

### 3.3.2.2 Activity mining

Another interesting mining approach that focuses on data is activity mining [22]. This approach addresses the challenge of execution logs that do not contain information about activities of a process, but information about low-level events (e.g. data modifications). This type of logs is very common in unstructured business processes. The idea of the method in [22] is to discover activities in a process by
mining data events. An activity represents an instance of a task. [22] uses the principle that a task groups a set of data-modifying operations that are closely semantically related, and which typically occur together.

The difference between activity mining and other mining approaches is that it focuses on low-level events (e.g. data manipulations). No activity or process model is assumed. The goal is to discover activities by clustering data events and reconstruct the process model. In order to obtain this high level structure of the process, several assumptions are made with respect to the data events. Data events refer to transitions in data object lifecycle, such as changes in the value or the validity of the data object. [22] uses the following state-transition system for describing the lifecycle of data objects, which is an extension of the data model presented in the case handling approach (Section 3.2.1, Figure 6):

![State-transition system for data objects](image)

Each transition in Figure 9 represents an event type for a data object that is monitored: define, delete, update, rollback and confirm. Apart from the event type, the data event should contain information about the resource that initiated the data modification, corresponding to the originator attribute. The timestamp of the event is also important to determine the exact time of the event occurrence. Additionally, the data type indicating the potential values of the data object is considered. These are requirements that the execution log should satisfy in order to apply activity mining.

To discover the hidden activities in the log, [22] uses clustering techniques, based on a set of hypotheses:

- each activity and its low-level events occur in the same process instance
- all low-level events resulted from the same higher-level activity have the same originator
- execution of an activity has a rather short time span, thus the resulting low-level events occur in each other’s proximity
- for hierarchical transactional systems only, all low-level events of a higher-level activity should have the same event type

The proximity concept defines the semantical relationship between low-level events. More clearly, for data objects of the same activity, data modification events take place between the start and the end of the higher-level activity execution. Therefore, a metric is defined for proximity, in order to decide if data events belong to the same activity. However, in order to get accurate results, the system that generates the log should satisfy a condition. It is important that there is a considerable gap between the duration of activity executions and waiting time between distinct activities of the same process instance.

In activity mining data has the central focus. The benefits of this mining approach are more than clear for systems that monitor low-level events. When the execution log contains such fine-grained information, abstraction at activity level enables further analysis of the log using the more advanced process mining tools and techniques. Even if certain information systems do not have the explicit
notion of a process in the log, resources performing work do have some procedures or models that they follow to manipulate and exchange information. These practices can be discovered by applying activity mining. Furthermore, the method can be applied as well on logs that do have the notion of process instance and activity, provided that data events are available. Case handling systems are, for example, a good application domain for activity mining.

### 3.3.2.3 Discovering PBWD

Another area of interest in process mining is the automatic discovery of Product Data Models (PDM) for administrative processes that could benefit from a Product Based Workflow Design (PBWD) approach. The PBWD method was presented in Section 3.2.2. So far, no formal approaches exist for the automatic discovery of a PDM from process execution logs. However, the topic is highly relevant for data-intensive processes and it is part of ongoing research taking place at TU/e. This section does not intend to give a solution to this problem, but discusses its challenges and how other mining approaches can relate to it.

Discovering the PDM from the data event log can provide a strong support for the improvement of the administrative business processes. This challenge addresses logs that contain data events. We remind the reader that a PDM model (e.g. Figure 7.ii) represents the structure of the workflow product (i.e. information object) in terms of data objects, operations on data elements and constraints on executing operations. Discovering a PDM model from the execution log requires detection of the way data elements are manipulated (i.e. operations) and conditions when operations are enabled.

At a first glance, it appears that the activity mining technique described in previous section is the best candidate for discovering PDM from data logs. The activity mining clustering algorithm can be used to group data objects that are necessary to produce the information for the next step in the process. This would determine the data elements that are operated together. Since the PDM has a hierarchical structure, the clustering algorithm can also use the data event types as an additional criterion to group the data elements. This criterion would be a strong aspect in grouping the data elements.

However, there are several drawbacks that make this mining approach not suitable for discovering PDM. First of all, the proximity function used for grouping data is based on the assumption that waiting time between activities is typically bigger than the time spent on executing activities. However, this might not be always the case. Activities may be executed immediately one after another, or even in parallel, if the system allows it. This situation can generate different results of the clustering algorithm. Moreover, multiple data access is a common practice in execution of processes. Data can be manipulated in several parts (activities) of the process, at different moments in time. The system may have also free case data (i.e. general case data, not linked to a specific activity), that may be available for manipulation during the entire execution. This makes it very difficult to decide to what activities these data elements belong to.

Still, the most complicated task in the derivation of the PDM is the discovery of operations and constraints. To build the semantics of the model, dependencies among activity clusters must be determined. The challenge is how to arrange two discovered activity clusters in the PDM model. Do they belong on the same level in the graph or on different levels? If they are on the same level, are they representing alternative paths, or parallel execution? Is there a direct hierarchical dependency between them (e.g. parent-child type of relationship)? What information should be appropriate to use to derive such relationships? This type of questions needs to be addressed. Clearly time, originator, and event type are just few aspects that should be considered for defining the classification criteria.

Mining the data log as if it were a normal activity log, may return some useful information about the data access patterns. Yet, as we mentioned already in the previous section, this data access model is not sufficient to derive the dependency relations at the higher-level of activity clusters. Applying clustering methods on the data access model may help distinguish between highly correlated data
objects and less significant connections. The Fuzzy Miner plug-in [86] of ProM mining framework supports this functionality.

However, these are only simple observations. Further research is required to derive a formal method for automatic discovery of a PDM model. Nevertheless, a deeper analysis of data is essential for creating such a technique.

3.3.2.4 Discussion

The mining techniques that we presented take data as a main ingredient in the process of knowledge discovery. The decision mining approach is a strong tool that shows how data values influence the decision points in a process, determining a certain execution path. In this case, data is used together with the discovered process model to perform the analysis. Also, for each decision point, all known data can be considered for investigation of the respective decision.

Based on discovered correlations among data objects, the activity mining technique derives a process model, abstracted at the activity-level. The method considers a transactional model for the lifecycle of data objects, assuming a set of events reflecting data manipulation at runtime. The direct link between data and originator is also necessary for clustering data that potentially belong to the same activity. The event types are not strictly required. They are used by the clustering method only for logs from hierarchical transactional systems.

Discovering PBWD does not have a formal mining method developed yet, as it is still subject of further research. Mining methods such as activity mining or clustering techniques can provide only partial support in derivation of a PDM. A more profound analysis of data is necessary to develop this method. However, the strong emphasis on data and the potential to support the PBWD approach make it an important topic in mining the data perspective of a business process.

Still, the possible benefits of using data event types in process mining are poorly explored. We believe that more knowledge could be extracted by using data specific events. Consider, for example, case handling systems. The strong emphasis on data and the availability of data events provide even more information that is currently not exploited by the process mining algorithms and methods. Even though the data object states are actually driving the evolution of the case (i.e. completing, redoing activities), the lifecycle of data objects is not regarded as a pivotal element in mining administrative processes. Efforts in applying process mining to case handling systems such as [13] pointed out the necessity of placing more emphasis on the information associated with the data elements during the mining process. [13] states how mining a data log considering data objects as activities can result in a process model with data access events as tasks. The results, however, are not very intuitive. Especially in information intensive processes where data objects are highly reused, the discovered model can be very complex and difficult to understand. Not to mention that often data objects are reused in groups (i.e. a set of data objects often occur together, in several parts of the process), which may lead to the generation of even more confusion.

Thus, there seem to be opportunities to extract more knowledge from the data events. The close intertwining of data and control flow triggers questions that could be addressed by the process mining techniques:

- How does the lifecycle of data objects reflect/impact the execution of activities and of processes, respectively?
- What is the relationship between specific data events and the originators?
- How to check the authorization role of originators using data events?
- How to use data events as performance indicators?

We give here an example of information derived from one data event type. Rollback events setting data objects to earlier values indicate loops in the process execution, at the same time inferring a
higher authorization role of the workers performing the rollback. Frequency of this type of events may give information about the performance of the worker and of the process, implicitly. By observing the moment and the frequency of data objects manipulation during execution, a better structure of the process can be derived. The enhanced or new model should be better tailored towards the real way people execute their work.

Obviously, data usage patterns can uncover many hidden practices and bring forward more knowledge that could not be visible by mining activity events alone. Therefore, a proper monitoring of data access in a process execution can help manage and improve the efficiency and reliability of the business process.

### 3.4 Conclusion

In this chapter we presented different methods and approaches that use data as a primary focus in design, execution and analysis of business processes.

We showed that administrative processes need more operational flexibility and that data-oriented approaches are able to improve the capabilities of processes to deal with unforeseen exceptions and change. The case handling paradigm is the most representative approach that uses data as a key-driver to provide better support for semi-structured business processes. However, the induced flexibility also makes it more important to closely monitor processes such that undesired behavior can be detected and corrected efficiently. The more unstructured the process is, the more important is to know how the process actually behaves.

All of this suggests putting more emphasis on the informational (data) perspective in the business process analysis. Many business intelligence tools use data mining techniques and visualization applications for extracting information from large collections of data. However, process-driven analysis of data is not sufficiently exploited. Only a few data focused approaches exist in the context of mining business processes. We described the decision point analysis and the activity mining as two representative data-driven methods already available in ProM today. The applicability of such methods opens the perspectives for more information systems to benefit from the process mining practices. At the same time, we stressed the idea that more opportunities lie ahead for data-centric analysis of business processes. Due to the increasing amount of data as well as data complexity, more powerful tools are needed, to better address these requirements.

The diagram in Figure 10 sums up the discussion that we followed throughout this chapter:

**Figure 10: Outline of Chapter 3**

Furthermore, we presented the importance of the human involvement in the process of analyzing data. Interactive data exploration can exploit the human knowledge and visual skills, which automated discovery methods lack. For this reason, we decided to extend the collection of data-centric analysis methods in the process mining domain. An innovative method for visualizing the lifecycle of data elements as well as of other process entities such as activities or originators is presented in the following chapter. The new data visualization method is implemented in the context of process mining framework ProM. Once again, we want to highlight the opportunities for knowledge discovery and the benefits this method can bring to the analysis of business processes.
In conclusion, we believe that the arguments put forward in this chapter support the promotion of data as a 1st class citizen in the area of process mining. In the remainder of the thesis we focus more on the case handling approach that is also used by IBM CHEF, described in Chapter 6. Next, in Chapter 4, we continue by presenting our contribution to mining of data in the context of business processes.
4 Data Visualization Method

In Chapter 3 we have identified the role that data plays in achieving more flexibility and adaptability in modeling and executing business processes, and we have highlighted the importance of mining the data perspective in the context of process mining. This chapter presents a new approach for visualizing data elements, named Element State Analysis (ESA). ESA displays the lifecycle of data during process execution and has been implemented in the process mining framework ProM. Although ESA was initially developed to show states of data elements, it turns out that it is also useful to show states of other workflow elements such as tasks, process instances, and originators.

This chapter is organized as follows. First, we introduce the motivation and the objective of the visualization method. The second section of the chapter describes the method, and includes design and architecture specifications. The third section presents the implementation details. The fourth part describes the simulation realized for evaluation of the method’s functionality and the results obtained. The last section presents the conclusions.

4.1 Motivation and Objective

The necessity of auditing capabilities and the augmenting complexity of modern information systems have direct consequences upon the way information is logged. Not only the amount of event logs produced at monitoring time is continuously increasing, but also more detailed and elaborated definitions of event structures are adopted for logging process execution information. Additionally, less structured processes demand for more control, as they need to be verified to assure a correct execution. The event logs of processes with such a dynamic behavior are more interesting and challenging to mine due to the interrelation between data and tasks. Because of this, the behavior of workflow elements is much more complex and therefore the informational (data) and the case perspectives require more attention than in traditional structured processes. These perspectives can also pinpoint performance issues such as delays or malfunctions. For instance, by correlating information about the working practices of task performers and how frequent certain data is accessed we can derive relations between the inefficiencies in some parts of process execution and the nature of the data that was involved. Discovering these kinds of relations can improve the analysis of flexible processes.

The need for a more appropriate support in analyzing less structured processes where data is playing a central role is the main reason for the development of the current data visualization method. In order to position the work properly, we also briefly present some tools addressing this area of process mining. While most of the process mining practices are focusing on structured processes [15, 16, 18, 19, 20], new techniques start addressing the special needs of highly variable situations. Since there is not a single method that addresses all aspects of process execution, several approaches are necessary to unfold its distinct facets. Techniques such as Decision Point Analysis [24] or Social Network Analysis [26] bring a valuable contribution to the data and organizational perspective. The former discovers how data dependencies are influencing the routing of the tasks in a process, while the latter discovers the organizational structure and the way in which the workers (i.e. people or systems) collaborate in
performing their tasks. The Decision Point Analysis is however bounded to the mined process model, which in the case of a flexible process is rather difficult to discover from the systems logs. Another technique, the LTL Checker, is aiming at verifying properties of cases and checking the compliance of the log with a set of business rules [25], without assuming a process model. Other mining methods have been focusing on the performance aspects of the business processes, gathering information about how cases were executed and highlighting long execution times or bottlenecks. Many approaches for performance analysis are based on graphical representations such as Message Sequence Charts or dashboards as found in most of the industry tools [3, 8, 30, and 35]. Apart from statistical information and performance measurements, they also display interactions among different entities and discover patterns in their way of communication. Moreover, visualization of log data has recently gained more attention in the process mining domain. New techniques such as the Dotted Plot or the Cloud Chamber Miner [25] are targeting at easy identification of patterns in the log.

However, the process mining support for data and case perspective still can be enhanced. What the methods mentioned above lack to provide is a clear display of an element’s state during its lifecycle. In a performance sequence diagram we can see the interactions among similar entities and intervals between interactions. These intervals, however, do not have an explicit meaning. It may represent a working/running state, or just a period of time when the workflow element is not working or running, or even a period when an originator has no work to do (i.e. he was not assigned any task). The states are currently not displayed in a distinctive manner, to be easily identified. The same observation holds for the other visualization methods as well. They focus mostly on the interactions or moments when events happen, but not on the meaning of the time intervals in between. It is rather difficult to identify the state of an originator or of a task when many instances of the process are considered and hundreds or thousands of events are displayed in the same view. Even by selecting few event types to display, still it is hard to recognize the distinct states when the corresponding time intervals are all identically represented.

For these reasons we propose a method that brings to the surface the state of the workflow elements. We try to answer questions like “what happens in between events occurrences?”, “is the worker busy executing tasks or is he idle?”, “is the data defined or unconfirmed?”, or “is there any relation between the data state and the authorization role of the worker?”.

Additionally, visualization of event data can be a valuable asset in the context of process performance analysis. Visualization is the graphical presentation of information, with the goal of providing the viewer with a qualitative understanding of the information contents [31]. Innovative ways for visualization facilitate the data to be interpreted holistically by exposing rudimentary attributes of the data such as patterns, trends, structure and exceptions. In this way, implicit knowledge hidden in large amounts of data is effectively conveyed to the practitioner, at the same time leveraging his domain knowledge. This is the motivation for developing a graphical implementation for displaying the states of workflow elements.

To give an indication about the kind of information displayed, we present in Figure 11 a short example. We assume that a task is an workflow element for which the lifecycle is most easy to understand, thus we base our example on task states rather than data states. Consider two tasks (e.g. TaskA and TaskB) being executed and each generating 3 event types in the log: assign, start and complete. Based on these event types, we can define the states assigned, running and completed, that we can visualize along the timeline of the process execution. For TaskA we represent only the events along the timeline, while for TaskB we show the states of the task. By comparing the two rows, we can see a significant difference in the information conveyed for each of the two tasks. For TaskB the time intervals capture more significance and enable an immediate identification of distinct states, as well as an effortless comparison of states duration. Even the empty intervals have more meaning now, as they exclude the existence of the other states.
Naturally, the Element State Analysis (ESA) method complements the views of other visualization approaches. Exposing the lifecycle of workflow elements reveals new information about the actual execution of the cases. Not only more information about the elements themselves is made available at one glance, but correlating the states of different elements may also provide new insights into performance characteristics, execution patterns or even user working practices and authorization roles.

Thus, the objective of this visualization method is to:

Implement a novel approach to visualize the lifecycle of workflow elements (e.g. data, tasks, process instances, originators) during process execution.

Next, we explain the architecture and design of the visualization method.

### 4.2 Method Description

Initially, the visualization method was intended for analysis of data elements. However, in a business process there are other entities that evolve during execution and that may be also interesting to visualize. Therefore, we decided that, besides data, a process instance, a task, a person or machine responsible for executing some task, may be appropriate for state analysis.

In general, a state is an attribute of a workflow element, which is determined in the context of a process instance, as a result of an emitted event. There is one important condition in defining elements states, to assure the consistency of the method: one event type can generate only one state. On the other hand, a state can be generated by more than one event type.

The fact that the states of workflow elements change over time, due to the emitted events, can help discovering hidden behavior. If the state running is evidently a state that is most wanted to occur, with as few interruptions as possible, other states might appear such as suspended or aborted which may be useful to see at a glance. If we think about more flexible processes, where data is one of the key drivers of the execution, then it is important to see the data flow, how data is being manipulated within process execution and what implications this have on other types of workflow elements. More specific use cases and benefits for the visualization method are mentioned for each type of workflow element.

The workflow element types to be explored for visualization are the following:

- Task (activity)
- Originator
• Data object
• Process instance

Although data was the trigger for developing this method, we decided to describe first the task and the originator states. There are two main reasons for this. First of all, there already exists a transactional model for task events, used by the process mining format MXML. This model (see Figure 12) is used as a reference in our task states definition, and makes it easier to understand our methodology of defining states. For data there is no formal model currently defined in MXML, so we had to propose a set of data specific events that could generate the data elements states. Second, we want to present the data lifecycle in the context of a process execution, by showing the relationships with the task and the originator states. Therefore, we consider appropriate to explain these elements first, in order to provide the required preliminary knowledge to understand these relationships.

For each of the mentioned elements, a set of states has been defined, considering event types that are specific for them. Most of the considered event types are part of the MXML format presented in Chapter 2, only few additions have been made if no such state or event type had been already defined. The reason for selecting this standard set of event types as a basis for the state definition step of our method is that they have been established after analyzing various real-life systems, and selecting the relevant event types that reflect how most of the systems log information [32]. The MXML transactional model illustrates the events specific for an activity and a process instance (case):

![MXML transactional model](image)

Figure 12: MXML transactional model [32]

When an activity is created the system can schedule it by taking the control over it or automatically skip it (autoskip), if certain logical conditions are not met. After it has been scheduled, the activity can be assigned (assign) to a person (or group of persons) that are authorized to execute it. It is also possible to reassign the task to another person (or group). At this point, a user can start working on the task, or some user can decide to skip it (manualskip) or withdraw it. The last two events can happen also before the task is assigned. Withdrawing means that the activity execution was not correct, contrary to manualskip. After starting working on a task, the user can suspend and resume it, or complete it. If an error occurs while the task is running or suspended, then the task gets aborted (ate_abort event), but the activity is also aborted if its process instance is aborted (pi_abort event).

We continue by presenting the states that we consider for each type of workflow element. Finally, a short summary of the requirements and assumptions of the method closes this section.

### 4.2.1 Task execution states

The task execution states are derived from the predefined activity events of the transactional model in MXML format displayed in Figure 12. Since the events in the transactional model refer also to the process instance (i.e. a case), the states that we define for activities are also applicable for process
instances. However, a few deviations from the original model were considered, as explained in the remainder.

We consider it useful in the context of an element lifecycle to define a new state *ManuallySkipped*, separate from the *Completed* state. The rationale for making this distinction is the specific nature of the *manualskip* event. If the *complete* and *autoskip* are events automatically triggered by the system, the *manualskip* event is emitted as a consequence of a deliberate action of a person working on the case. This person actively skipped the execution of an activity, even though the process model assumes that it should have been executed at that point.

As an example of the usefulness of this state, consider the situation of less structured business processes, where the knowledge worker involved in a case has more freedom in the way he is executing activities. In such cases, the process model is designed to guide the worker in performing his tasks, but not to enforce the exact order they were defined. Therefore, being able to identify the states where the activities have been skipped due to a user decision, gives a better insight in the way people are performing their work. It may enable discovery of user working practices, as well as the real importance of the defined activities in the process execution. As a consequence, if many deviations from the prescribed model are recorded, then a closer inspection of the working practices should be engaged and if necessary, a redesign of the process should be considered. For this reason we consider that distinguishing the *ManualSkipped* state from the *Completed* will provide a better view on the case perspective.

For the *autoskip* event, we did not include a separate state as in the case of the *manualskip* event. The reason for this is that an automatic skip reflects a choice made by the system based on the state of the process instance at a certain moment. Basically, the execution reaches a point where several distinct paths diverge, and one or more of the outgoing branches can be followed, depending on some logical conditions associated with them. This moment represents a decision point in the process. The branches where the control flow is not passed to are skipped, meaning that all the activities on those paths are automatically skipped. Therefore, considering the *Completed* state as the result state for the *autoskip* event would suffice for the purpose of the method and keep the list with activity states to the minimum required.

Every rectangle in Figure 13 corresponds to a state of a task:
- **Initial**: the first state, before schedule / autoskip or any other event.
- **Ready**: the result state of the schedule event
- **Assigned**: the result state of the assign / reassign events
- **Running**: the result state of the start / resume events
- **Suspended**: the result state of the suspend event
- **Aborted**: the result state of the withdraw / ate_abort / pi_abort events
- **Manually Skipped**: the result state of the manualskip event
- **Completed**: the result state of the complete / autoskip events

So far we defined activity states having as a basis the event types defined in the MXML transactional model.

We want to stress here that the MXML transactional model has been used merely as a reference for the definition of the states, due to the generality and applicability of the event types it contains. Also, we think that by using this model as a reference, the reader may understand better the rationale for definition of these states. However, for the visualization method we do not restrict the usage of these states to the existence of these specific event types.

We are aware that the ability to visualize states of elements depends on the event types available in the log. Less event types may probably result also in less states being displayed (e.g. the absence of the suspend event may imply missing the Suspended state as well). On the other hand, some information systems use a list with event types more detailed than what is considered by MXML transactional model in Figure 12. In order to cope with these differences in real life logs, we assume the definition of states to be independent of one another and the definition that we presented above to be considered as a guide rather than an enforced description. The user of visualization tool will be given the freedom to interactively adjust and define the states that he is interested in using the event types that exist in the execution log. More details about the configuration settings will be given in the implementation section (Section 4.3).

In any case, we assume that the list of states that we defined in Figure 13 provides a reliable basis for the objective of this research. Even from this list we do not consider all states having the same importance for the person performing the process analysis. The states Assigned, Running and Suspended are relevant as they can indicate when and for how long some user waits to perform some task, how much time he spends on its execution, or when he suspends it. Therefore, they are depicted continuous, as the Assigned state of Task B in Figure 11. The states Aborted, Manually Skipped and Completed show how a task reached the final state in its lifecycle. Thus, they are depicted as a small rectangle of fixed size (e.g. Completed state of Task B in Figure 11). For simplicity of the graphics, the other two states Initial and Ready are not considered for visualization, as we consider they are not that important. They would only clutter the picture. The following figure presents the selected task states for the graphical representation, with no dependencies assumed between them:

![Figure 14: Task/process instance independent states definition](image-url)
Our intention is to keep things as simple as possible, to provide a visualization method that is easy to grasp and appeals to the human eye, focusing on the most important aspects. This issue will be readdressed in the implementation description, to strengthen our motivation.

We point out that we do not assume any dependencies among specific states. Since we offer the freedom to the user to define the states according to available events in his execution log, if certain event types are missing, then, consequently, some states may not be defined. Therefore, the unavailability of some states does not affect the existence of others.

Another question of interest is the duration of a state. If the triggering event marks the starting point of a state, how is the ending point of the state determined? Since we use events to determine states, naturally, we use events also to define the end of states. The lifecycle of a task is a sequence of states, determined by a timely ordered sequence of events. Thus, the end of a task state is determined by the event which triggers the activation of a new state for the same task and same process instance. The example given in Figure 11 is clearly showing how the first state of TaskB, *Assigned*, is ended by the presence of state *Running*, which is ended by state *Completed*. In case a state doesn’t have an end event (e.g. due to missing events, errors), then the end of the corresponding process instance finalizes the state duration.

Furthermore, there is another aspect that requires attention. In general, several instances of the same process are run in parallel. It may be interesting to see the states of a task when more than one process instance is selected for visualization. In this case it may be useful to have an aggregated view of the task states over the selected instances. We assume that a task can have only one state at a time in a process instance. This assumption is derived from the transactional model used as a reference. Thus, if several instances are considered, it is probable that some states overlap over the same period of time. This issue will be discussed also in the implementation section.

### 4.2.2 Originator states

The originator states are defined using the task events to which they are associated. We assume this because usually, the relationship between an activity and the person that is working on it or the persons that have been assigned to it is available in the information system. This is either explicitly outlined, as a task attribute, or if not, it may be derived from, for example, user sessions. What is important from the performance point of view is to monitor not only the activities in a process, but also how the workers are carrying out their duties. The efficiency of a process depends not only on the structure of the process but is definitely related also to the management of the resources, the working regulations and standards, and especially on how people are actually doing their job. For these reasons we consider that monitoring the states of the workers will provide valuable information on the case perspective.

A state of an originator is defined based not only on the process instance but also on the task to which he is associated with. We assume this definition because of the following reasons. It is possible that in the same time interval, in the same process instance, the originator has several states. During the execution of one case a worker may be assigned to several tasks while he is working on another task. The originator is the only workflow element type that can have multiple simultaneous states for the same process instance. Also, the worker may be involved in several cases simultaneously. However, for every process instance, the performance of the originator is shown considering all originator states belonging to that instance. This means that all existing originator states corresponding to the tasks in that instance will be aggregated in one view.

The states that we considered for the originator element, depending on a process instance and a task, are (Figure 15):

- *Assigned* (waiting)
- *Working*
- *Unassigned* (idle)
The *Assigned* state reflects the situation when a person is assigned to a task, but he is not yet working on it. It may be triggered by a task *assign* event (Figure 15). The state might have a reasonable explanation as for instance, the worker is waiting for some forms or documents to proceed with his work. On the other hand, it can also be the case that the worker is working outside the process or task being logged. Either way, it becomes important if the state persists for an unjustifiable period of time.

Probably the most desirable state of a worker, from the company perspective, would be the *Working* state. This state shows that the respective person is involved in the execution of some activity. As an example, it may be triggered by a task *start* or *resume* event (Figure 14). For each process instance, displaying the time intervals in which the employee is busy with work can uncover many hidden facets of the working behavior. One can see how much time a person is investing in his work on a time period (i.e. week, month, year), which can serve as an indicator for either efficiency or ineffectiveness in the worker’s evaluation. Moreover, the working behavior of the worker becomes visible: discovering in what part of the day or week, or a project timeline, a person is more productive. Many studies of human working behavior show that people have the tendency to postpone work towards the deadline of the project and that they are more relaxed at the beginning of the project.

The *Unassigned* state suggests that, for a period of time, the employee is neither assigned nor working on a specific task. Clearly, it is important to identify those time intervals where the originator has an empty work list. This may give some insights about how resources are managed and how work is distributed in the process.

Due to the fact that the task events are used to determine the states of the originator, we have a special case in the definition of the *Unassigned* state. This state can represent the initial state, before the originator gets assigned or starts working on the task. It can also be triggered by some task event (e.g. *complete, assign, abort*) that finishes the originator’s previous state. This requires a discussion about the possible end event of the *Assigned* and *Working* states, which also determines their duration.

For these two states we consider the same constraint for the end event of an originator state as in the task case. That is, the end event of an originator state must have the same task name and the same process instance. We make the observation that no constraint is related to the originator. A worker may have multiple states at the same time, either from different cases executed in parallel or even in the same process instance. Still, the end of his *Assigned* or *Working* state may not only be determined by his own actions, but also by “external” events. By “external” we refer to events that do not relate directly to the current originator. For example, one assign task event may generate 3 *Assigned* states for three different workers. Once the first originator starts executing the task, the other originators will be automatically unassigned for that specific task. In this way, they lose the opportunity to execute the task. This is a very common example, where the task is being blocked during it’s execution from other users access, to prevent from concurrency problems. If no proper end event can be found, then the originator state remains active until the end of the process instance. As a consequence, the duration of these two states is defined in the same manner as for tasks. On the other hand, the ending of the *Unassigned* state, and implicitly its duration, is determined by the activation of any of the other two states for the same originator. Figure 15 shows the originator states and possible triggering events:
However, the analysis of the originator in the context of one process execution will consider the aggregation of all originator states relevant for that specific instance (i.e. considering all tasks involved in that instance). To clarify the situation of multiple states, we give here an example. Figure 16 illustrates the situation of Originator A involved in one case and having 3 Assigned states in one of the intervals (for tasks T1, T2 and T3), and another interval where he is in state Working (for T1) and in state Assigned (for T2 and T3). In the latter case, the states are depicted by combining the colors representing specific states:

![Diagram showing originator states](image)

**Figure 16: Visualizing originator states**

Obviously, visualizing how the worker’s states interleave would pinpoint essential information necessary for worker evaluation, improvement of resources management and even distribution of work. The visualization method will enable project or department managers to have a clear view on the amount of work needed for executing process activities, and enable them to take better decisions in order to strike a good balance between the necessary workload and employee satisfaction. We can see that the originator perspective of the Element State Analysis method is similar to the so-called Gantt Charts, only that it shows real facts and not planned behavior.

### 4.2.3 Data objects states

After defining the states for task and originator, we can introduce the states of data objects. The objective of the data perspective is to display the data flow of a process execution. As we mentioned in Chapter 3, allowing for more flexibility in a process means putting more emphasis on data. In data-driven systems such as Case Handling systems, data determines the state of the process by enabling and/or completing activities. How is the data flow influencing the process execution? How frequent are variables updated? Where in the process are the most frequent changes? What is the link between the variable updates and activities and users executing them? These are just a few questions that can be answered by using this perspective.

The transactional model considered as a basis for our state definition in previous cases is especially addressing workflow elements like activities and process instances. Event types specific for data are not included. Nevertheless, in order to better serve the needs of less structured business processes, we claim that data is as important as the tasks, and treat it as such. In Chapter 3 we have already argued why data is important in mining semi-structured processes. In order to extract more information about the data flow in a case, states cannot be derived from the task events. Our experience so far reveals that in commercial systems not many specific event types are usually defined for variables used in a process, especially considering systems where data is not a major concern but rather the process control flow. By analyzing several workflow systems like WebSphere, YAWL and also more data-oriented systems like FLOWer, we concluded that a transactional model for data objects as the one used in [22] would be appropriate:
This model extends the data model proposed in the case handling paper [21] by adding the *update* and *unset* events. We consider the states in the model above are sufficient, as these appear to give most relevant information about the data flow in a process instance. The model also assures that one data element can have only one state at a time during one process instance execution.

The *Unconfirmed* state of the data object may indicate a rollback/redo action of the originator, as in the case handling system FLOWer. This state can unveil more information than a common *Update* state. It infers not only the re-execution of some task, but also the authorization role of the originator responsible for the task execution (see Section 3.2.1). Moreover, specific access patterns may indicate redundant parts in the process or deficiencies in the process design. For example, if the data appears to be frequently *Unconfirmed*, due to many rollback actions, this immediately suggests that the corresponding activities are redone not in exceptional situations but rather regularly, and thus it might be appropriate to consider the redesign of the process model. All the displayed characteristics of the data usage can be traced back to user interactions and working practices. Correlating information from the data view with that implied by task and originator perspectives can provide a more comprehensive understanding of the actual case execution.

In some information systems, however, the relationship between data events and task events is not explicitly logged. In this case it would be rather difficult to establish the missing link, due to concurrency issues. This situation exists also in IBM WebSphere, which will be discussed later, in Chapter 5. However, when the dependency between data and task is known, then the data analysis may provide additional valuable information about the case. Figure 18 shows an example of how data states can be visualized, given the existence of data specific events:

![Data object lifecycle](image)

*Figure 17: Data object lifecycle*

The reason why we did not consider the task events for deriving data object states is that there is not a strong dependency between data and task events. If some activity is completed, for instance, it does not imply that every data object belonging to that activity has been updated or defined. Even more, if there is a *redo/rollback* event for a task which is associated with some data object, still we can not
decide how the original information system relates this event to that data. It is possible that the data object state becomes Unconfirmed, or maybe the data state remains Defined if there is no other state specifically defined. In this situation it is rather unrealistic to assume a certain interpretation of the task events with respect to data object states and so, we don’t consider the task events for defining data specifically. In this situation it is rather unrealistic to assume a certain interpretation of the object state becomes 

Figure 19: Data independent states

The reason for this restriction is that we needed to tell whether some event corresponds to some data object or to some task. Simply put, we assume that the events named define, update, rollback, confirm and unset correspond to some data object, and that other events correspond to tasks. This seems to be the most reasonable choice for the moment. The duration of the data states is handled in the same manner as for the task elements.

4.2.4 Process instance states

The process instance execution states are based on the tasks events available in the log, belonging to the same instance. The reason for considering tasks events to determine the state of the instance itself is twofold. First, not all information systems have specific event types considered for process instances. In highly flexible environments, such as hospitals, some systems don’t even have the notion of a process instance or a case. Therefore, we don’t want to limit the applicability of our approach only to systems with more advanced monitoring capabilities. Second, for the systems that do have specific process event types, the possibility to see the instance states derived only from these events still exists. By treating the process instance as an activity and making this information available in the log, it is rather straightforward to display the state of the case based only on these events. Therefore, we provide the means for the user of the tool to choose the appropriate approach to investigate the process instance perspective.

For process instances the same transactional model can be regarded as a reference, as presented in the case of activities, in Figure 14. A process can be seen as a bigger or complex activity, after all. Therefore, we assume that the states of a process instance are aggregations of all task states corresponding to the considered instance. To give an idea about how to visualize the states of a process instance, we show an example in Figure 20. The states of the process instance PI1 are determined by combining the states of the tasks T1 and T2 involved in the execution of PI1 (i.e. the colors of overlapping states are also combined to render the aggregated state):
We can see that by aggregating the task states, we obtain a more refined view over the process instance, as information about parts of the process instance (i.e. tasks) is visible. We can identify concurrently executed tasks, or tasks that can be executed in parallel but instead they are sequentially carried out, as in the example above. Consider, for example, the situation where several Assigned states are overlapping with a Running state. This means that even though many tasks can be executed in parallel, only one is actually executed in a certain time interval. Evidently, this kind of information is extremely valuable for performance analysis of the process execution. The combination of task states enables these situations to be easily identified.

Moreover, an interesting use case of this perspective would be to display the process instances as aggregations of all data states corresponding to each instance, instead of combinations of task states. ProM has advanced functionality to filter event logs and swap tasks with data elements or rename event types, such that our visualization method could treat data elements as if they were tasks. For applications with case handling functionality, like IBM CHEF, this may provide useful information. The lifecycle of a process instance will be visualized based on data states.

In this perspective it is also interesting to see the global view of the process, where all process instances lifecycles are exhibited. Information about concurrent, failed or suspended instances may give a quick impression about the performance and transactions load of that process. Additionally, the number of process executions might give other qualitative insights into the business the company is operating. Processes that have a small number of executions and possibly high operating costs might be considered for outsourcing. Or, it may be as well the case that very frequent processes incur high costs that can be reduced by delegating their execution to other parties. In conclusion, this perspective could help the organization to realign its business to its core competencies.

### 4.2.5 Method requirements summary

In previous sections we described the specific characteristics of the visualization method for each type of workflow element. We also discussed some use cases and gave a short preview of how the lifecycle of elements can be visualized. Here we want to pinpoint the most important requirements of the method and assumptions we made in defining the elements states. They are as follows:

- an execution log is required as input
- events in the log should contain timestamp information
- data specific events (Figure 17) are required for the data perspective
- all states excepting Unassigned are defined considering only the triggering event type(s)
- no dependencies are assumed between existence of states themselves
- a state can be triggered by one or more event types (Figure 21, a))
- one event type can trigger at most one state (Figure 21, b))
- the duration of an element’s state is determined based on available states related to the same workflow element and the same process instance
- one Originator may have several simultaneous states in the same process instance, depending on the tasks that he is associated with; a task or a data element can have only one state at a time (corresponding to a certain process instance); a process instance state is the aggregation of the task states on a certain time interval

![Figure 21: Consistency rules in defining states](image)

a) b)
The next section of this chapter will present the implementation of the method as a plug-in of the process mining tool ProM.

### 4.3 Implementation

The data visualization method presented in the previous section has been implemented in the process mining framework ProM as the **Element State Analysis** plug-in. The plug-in intends to display the workflow element’s lifecycle, graphically, as a band following the timeline, with alternating colors depending on the states the element is going through. This provides a new way of looking at the log data and hopefully will open new doors for process knowledge exploration and for more advanced business analysis.

As mentioned in the method description above, four types of workflow elements are considered for the state visualization. Since some specifications are different for each type of element, we will present them for each case, separately, besides the general functionality discussed in Section 4.3.2.

#### 4.3.1 General plug-in functionality

First, the objective of this implementation is to provide an effective, easy to interpret and adaptable visualization tool that can serve multiple needs of the end user. Moreover, the visualization tool should enhance the understanding and the knowledge learning via interactive techniques and provide information from multiple views. These constitute the quality requirements for the Element State Analysis plug-in.

The main functional purpose of the Element State Analysis plug-in is to graphically display the lifecycle of workflow elements, based on the process event log, enhanced with data specific events. The lifecycle of elements is defined as a set of states that are valid on a certain time interval, during a process execution. The elements states are defined based on the event types available in the log, and the type of workflow element (i.e. task, process instance, originator, and data). The states considered for analysis were already introduced in the previous section, where we described the visualization method (Section 4.2). Those selected for visualization are listed below, for every workflow element type:

- **Task / process instance states:** Assigned, Running, Suspended, Completed, ManuallySkipped, Aborted
- **Originator states:** Working, Assigned, and Unassigned
- **Data states:** Defined, Unconfirmed, and Undefined

In order to manipulate these states, we define two types of shapes for their graphical representation:

- **Separator:** a small rectangle of fixed width (e.g. Completed)
- **Lane:** a band having the width equal with the duration of the state (e.g. Running).

The reason for using a Separator shape that does not reflect the actual duration of a state, is to mark the presence of a state, but not to overload the image with a colored band. We consider that the duration of such states is less important to visualize. Apart from the shape and size, the state object has also a color attribute, which should indicate at a glance the meaning of the state.

The Element State Analysis plug-in is similar to the Dotted Chart technique available in ProM, considering the arrangement of the elements in the chart [91]. However, instead of plotting dots on a chart that indicate events, we provide a complementary view by displaying the states in between events occurrences. One may think about these two approaches as being two sides of the same coin. Figure 22 gives a screenshot with the layout and main functionality of the plug-in.
As depicted in the previous picture, the arrangement of the graphical elements on the page is based on two dimensions:

- Time (horizontal axis)
- Element (vertical axis)

In this way, every row will show the different states of one element, corresponding to one type of workflow element (i.e. task, instance, originator or data). For example, Figure 22 displays the states of the tasks. We can easily distinguish between the two types of shapes that we defined previously: the separator (small rectangles) and the lane (rectangles with variable length). In this picture, the separators represent Completed states, while the lanes show the Assigned (lighter color) and Working (darker color) states. We can already identify some tasks that take longer time to be executed by just looking at the lengths of the Working states, while others are waiting for a worker to start their execution (Assigned state). When moving the mouse over these colored lanes and separators, information about the state and the event that generated it is provided (e.g. process instance, element name, originator, etc).

More detailed information about the kind of information that can be derived from these views, depending on the type of element, will be given shortly. We want to stress here the importance of having a separation of the views based on the type of workflow elements. This will enable the viewer to focus on a specific aspect of the process execution, while changing the element type will allow him also to make comparisons, measurements and to detect dependencies between the distinct views.

The left panel in the picture contains the options for the tool user to interactively modify the perspective’s properties and the workflow element type. For each type of element it is possible to select the time unit for the time axis using the Time Sort option, or to zoom in and out or to select areas in the picture that need a closer inspection. Another important functionality is the time option that allows the user to use different notions of time (e.g. actual, relative, logical, etc). The time options will be described separately. The most important advantage of these options is that they provide the
means to involve the user in the data exploration process and make use of his flexibility, creativity and general knowledge.

**Multiple time options**

Apart from seeing the states based on the actual timestamp of the events, we can see the states based on other definitions of time. Based on these different time notions, the states will be positioned in different locations on the time axis. Apart from the actual time, there are additional four time options:

- relative
- relative (ratio)
- logical (based not on time, but on order of events)
- logical relative (based not on time, but on order of events)

The *relative* time option shifts time on an instance-by-instance basis such that every instance starts at time 0, instead of the actual timestamp. In this case it is easier to see what instance takes longer time to be executed, or after what time since a case started some task is actually performed. The *relative ratio* assumes also that cases start at time 0, but stretches the duration of cases such that they end at the same point. In this way we can see better the relative distribution of states during the case duration. For example, we can see what tasks are executed in the first 30% of the process instance duration, and how their states span over this interval. Discovering a task that remains in the assigned state for over 20% of the process instance duration can give a good insight into the performance of these tasks. The *logical* time setting assumes that events are ordered based on the position number in the sequence of all events relevant for the chosen workflow element type. The *logical relative* assumes also a discrete ordering of events, but relative to the set of events belonging to the specific element that is being analyzed. A more detailed explanation of these time settings is provided in [91].

Just to give a flavor of what these options can do, we show in Figure 23 a screenshot with the *logical relative* time for the task element. In this case the duration of a state is not the actual one, but is determined based on the position of the event that generates it, and the position of the event that triggers a change in the state of the element. However, in this “firefly view”, for each task it is easy to identify patterns related to the order in which states appear or to spot any exceptional situations.

![Figure 23: Task view with Logical Relative time option](image)

Next, we present other means for adjusting the properties of the plug-in to the user needs.
Another important aspect of the plug-in implementation is that it is configurable to the specific user needs. The configuration options for each element type are provided in the Settings Panel. The picture in Figure 24, for example, presents the configuration options for the task elements. For other workflow elements the settings are the same, except that the set of possible states varies per element type. The Settings Panel brings a major contribution to the usability and adaptability of the tool. It provides the following properties that can be adjusted to fit the user needs:

- state generating events
- state color
- state priority
- selection of process instances
- two methods for painting the overlapping states

First, the user is able to define what event types from the execution log can generate every state of the selected element type. Defining states according to the available events in the log enables more flexibility by not constraining the plug-in user to a rigid definition of states. By changing the mapping of the event types onto the various states, the user can experience different views and visualize different behavior of the workflow elements. This option leaves the functionality open to any kind of log, independent of the event types contained. However, the user needs to be aware that the interpretation of the results also changes when the event type mappings are changed. For supporting the user in the definition of the states, we also provide a default mapping, provided that the events we considered are present in the log (e.g. start, assign, reassign, complete, suspend, abort, update).

Moreover, the user is free to choose the color that defines each state. Again, we must warn the user about the risk of using inappropriate colors. Most likely there are states that overlap in a given time interval when considering several process instances. This situation is possible even for one instance selected, for the Originator perspective. We explained already in Section 4.2.2 that the originator state is determined based on a process instance and a task. Therefore, if the originator is involved in several tasks, he may have multiple simultaneous states. For these situations we propose two methods for coloring the lanes: combining the colors of the overlapping states or display the color of the state with highest priority. In the latter case, the priority settings of all states will be taken into account. For the combination of the colors we apply the arithmetic mean on the array of colors of the overlapping states to derive the combined color.

The user can also select any set of process instances from the list enumerating the instances existing in the execution log. When several instances are selected, the result is an aggregated view on the workflow element type chosen to be visualized. This means that for every element (e.g. task Id), all the states corresponding to the selected instances are displayed in the same timeline (i.e. lane in Figure 23). Overlapping states will be represented in a combined way, as explained in the previous paragraph. This enables a more thorough analysis at different abstraction levels, from the case level to utmost level, considering all instances in the log.

We make the observation that the data specific events and the states defined in the previous section for each process element type are hard coded. Apart from this, the user has the freedom to use the configuration settings to adjust the visualization to his needs.
4.3.2 Task perspective

In the method description (Section 4.2.1) we defined six different states for tasks: Assigned, Running, Suspended, Aborted, Manually Skipped and Completed. Nevertheless, as we stated already, we want to keep the image simple and clear, that doesn’t require much effort to identify the meaning of the colors, and at the same time, we want to render the most important information that is relevant for the analysis. Overloading the screen with rainbow-like colored graphics will jeopardize the most important benefit and the objective of this visualization method: legibility.

For this reason we decided not to give the same importance to all activity states defined. We distinguish between:

- **Main states:** Running, Assigned, and Suspended
- **Final states:** Completed, Manually Skipped and Aborted

A main state is displayed as a colored lane that can have variable lengths. A final state indicates the end state for a task, thus it will be marked by separator shape instead of a lane. It usually signals the ending of one of primary states. This way it is placed on a secondary level in the field of vision, not obstructing the view of the important states.

The length of a lane reflects the duration of the represented main state. We explained in Section 4.2.1 the criteria for determining the duration of a state (i.e. the time between the triggering event and the end event of a state). As we have already pointed out, the task final states Completed, Manually

---

**Figure 24: Settings options for task elements**

Further we present the specific requirements and assumptions we consider important for implementation for every type of workflow element.
Skipped and Aborted are displayed with fixed size and thus no indication about duration. The reason is that a final state remains active until the end of the process instance, and thus no other state is enabled for the same task in the given instance. For the sake of image simplicity and clearness, we assume the separator shape is sufficient to convey the meaning of it.

By making this selection and prioritizing the states we hope to generate a cleaner image. This is important for combining the distinct, instance based task lifecycles into one view (i.e. one colored lane), showing the state evolution of the task at different abstraction levels. It is possible to identify execution patterns of the task just by seeing through the colored bands. If several process instances execute concurrently, states may overlap. If many instances are selected, then it is essential that only few states, and thus few colors, are explicitly drawn as lanes, to allow a meaningful result when combining them. For these reasons we assume that three main states and implicitly three colors are the maximum that should be combined, without loosing valuable information.

There is an important observation regarding the duration of main states. Our visualization method assumes that if no proper end event for a state is found in the log, then the end timestamp of the respective process instance is considered as the end of that state. Obviously, this creates an ambiguous situation, in case of “hanging” events. If the log is not complete, or some events are missing, then these situations may occur and may not necessarily describe the real situation. This risk needs to be assumed in case of logs with this kind of noise.

4.3.3 Originator perspective

For the Originator perspective we consider displaying all three states defined in the description part of the method: Working, Assigned and Unassigned. It results in a view similar to a Gantt chart model, only this time there are no predictions, nor estimations, but just concrete facts. Moreover, the view can be displayed based on one or several selected process instances. Depending on the situation, the originator (being a person or a system) may be working on several cases simultaneously or even on several activities of the same case in parallel, if the system allows it. If multiple states are overlapping, the user of the ESA can also select the priority based coloring option for displaying the states. By giving priorities to states, the user can visualize the most relevant behavior if some originator states are overlapping.

If, for an originator, the state is Unassigned on some interval, for all selected instances, then the general view will also display the Unassigned state. If some originator is not involved in a case, then his Unassigned state is visible for the duration of the entire process instance. We implemented this particular state, such that it is displayed only when neither Working nor Assigned state is active on a certain time interval. We assume that Working and Assigned are more important to visualize.

In order to display the Assigned state for some originator, our implementation assumes that the name of the considered originator is provided by the task event that triggers this state. More specific, any event of the assign-like type (i.e. mapped on Assigned state) should contain the names of the workers that have been assigned to that specific task. Every name of the assigned workers should be provided in the ATE of the MXML log as an attribute. The attribute’s name should be prefixed with “assignedUser” string. Figure 25 gives an example of an ATE to show how the assignedUser attribute should be used:

```
<AuditTrailEntry>
  <Data>
    <Attribute name="assignedUser_1">Mike</Attribute>
    <Attribute name="assignedUser_2">Dan</Attribute>
  </Data>
  <WorkflowModelElement>TaskA</WorkflowModelElement>
  <EventType>assign</EventType>
  <Timestamp>2007-06-12T07:42:45.543+01:00</Timestamp>
  <Originator>Eric</Originator>
</AuditTrailEntry>
```

Figure 25: ATE example with assignedUser attribute
Computing the duration of originator states has been implemented according to the description given in Section 4.3.3. However, handling the states of the originator is more difficult than of the other types of workflow elements, as it requires a more careful manipulation of the necessary events, due to the so-called “external” event. As in the case of task states, if some events are missing, then the duration of the states may not be accurate. The risk of providing misleading information should be considered and therefore a proper inspection of the log is recommended before starting the analysis.

Figure 26 presents the originator view. The most dominant state in the chart is Unassigned (dark color), followed by the Assigned (light color). The least dominant state is the Working (middle color) state. We can see now that the Unassigned state provides clear information about when the originator is not actively involved in a case, by distinguishing from intervals where actually no process instance is running (light gray):

![Figure 26: Originator view](image)

4.3.4 Data perspective

Visualizing data objects states is the principal trigger for developing the ESA plug-in. We highlighted already the key role played by data in semi-structured processes but also the weak support received in the area of process mining. We also argued that lifting data to the same audit level as the task enables for more thorough analysis of the data perspective. As a consequence, certain implementation efforts are necessary. The most important requirement for the data perspective is the presence of data elements as ATEs with specific event types in the MXML log, as discussed in the context of Figure 17. ProM treats the ATEs as the principal elements for mining business processes (Figure 4, page 8). Up to now data elements have been handled more like attributes of the task ATEs (and thus at an inferior level). One of the reasons for this is that common execution logs do not contain elaborate data events. However, most of data-centric information systems record more detailed information about data manipulation. Our visualization method exploits this opportunity and offers a more adequate support for the analysis of such systems.

If the original execution log contains information about data elements used in the process, then an initial processing of the log can map the data events to Audit Trail Entries (ATEs) in the MXML log. Such pre-processing of the log can be realized in the ProM Import plug-in (see Section 2.3) that converts the original event log into MXML format. This way data is treated in the same manner as
activities. However, having both data and activities on the same level, as ATEs in the MXML log, requires means to distinguish between them. Thus, it is important that the data event types described in the transactional model in Figure 17 are used in the pre-processing phase. This model assures the correct distinction and selection of data ATEs from the activity ATEs, to display the data perspective. The data model is restricting the set of data event types used for rendering the data perspective. In situations when the original log has different data event types, the developer of the conversion plug-in should be aware of this limitation and find the most appropriate use of the event types in the data model.

Another way to distinguish data from activity ATEs is to consider a special attribute of the Audit Trail Entry (ATE) in the MXML format to indicate the workflow element type (e.g. task, data, process instance). This attribute is not defined at the moment in the MXML format, and consequently, the import plug-ins and their resulting MXML logs are not aware of it. If we had implemented this approach, the available logs could not have benefited from the ESA. Thus, we decided to implement the previous approach, using the data specific events. However, we believe this alternative would be more beneficial in the future. A clear advantage of this alternative is the complete freedom of the data events, and no strict dependency on the data model. Additionally, this method could be easily used when other workflow elements such as originator or process instance have specific events in the log and require the same treatment as tasks.

Another question to consider is how to include in the data ATE information about the task and originator to which it is related. Apart from the optional attributes, the ATE main attributes are: WorkflowModelElement, EventType and Originator (see MXML schema in Figure 4, page 8). We could accommodate the task name using the Originator attribute, as it seems natural that data should be linked to the activity where it is accessed. This can be done if the original execution log contains the clear link between data and task. Then, the relationship with the actual originator of the task could be traced back through the task events, provided that information about the task originator exists. Obviously, having specific properties in the MXML ATE structure that could store all this information would be a more consistent and accurate solution, and simplify the implementation.

Regarding the duration of the data states, we applied as in the case of the task element. The states Defined and Unconfirmed have variable durations. The Undefined state is considered to be a final state and, thus, is depicted as a Separator shape, with a fixed length. Figure 27 shows the data perspective of a process log with 64 cases. The picture displays all data states we defined. The alternation of the colors and the duration of some states can provide valuable insights on how data flows in the execution of the process. This novel approach of visualizing data elements brings to the surface new knowledge about data access patterns in the process. Naturally, this perspective increases the support for understanding how semi-structured processes are actually executed and how data is influencing the operational flexibility.
Testing the data perspective in ESA is based on a simulation log. Since we did not have a real-life log containing data specific events, we generated synthetic logs from the simulation of a case handling model realized in CPN Tools. We also used the simulation logs to analyze the performance of the plug-in. The simulation model and the performance analysis are described in Section 4.4 of this chapter.

Similar to the other workflow elements views, the data perspective can be aggregated using several process instances selected by the user. This option might make the image very complicated, especially if there are many concurrent instances. Therefore, choosing an appropriate time option and a reasonable set of instances may help provide a clearer picture. However, in many situations it is sensible to display the data perspective based on only one process instance.

4.3.5 Process instance perspective

The process instance perspective will display information about the states of the individual process executions. The same set of states defined for task elements are considered to characterize process instances as well. We mentioned already in the method description (see Section 4.2.2) that a process instance lifecycle is based on the evolution of all tasks contained in the instance. The process instance states are the aggregation of all tasks states belonging to that specific instance. Therefore, the information transmitted by this perspective depends on existing task events. The duration of states is determined as in the other cases, when aggregation of several states is involved.

In case the log contains process instance specific events that are desired to be used for defining the states, a similar situation to the data perspective needs to be discussed. For using specific process instance events, we need to have the case (i.e. process instance) treated as a task and have the specific event as an Audit Trail Entry in the MXML log. Additionally, as mentioned in the case of data, we need to have the means to separate the different types of ATEs based on the workflow element type. Therefore, the ATE in MXML format should accommodate such an attribute that can enable a systematic approach to distinguish between them. However, the decision to use the task events related to the process instance permits existing logs to be visualized in ESA as it does not impose additional requirements on the MXML log. Until further modifications of the MXML structure, we consider that our choice is well justified. If the MXML schema is extended to include this attribute, then it would require only a small modification of the ESA code.
4.3.6 Guidelines for ProM Import developers

The implementation of ESA was restricted in some aspects by the process mining framework ProM. We had to adapt the implementation of the method to the available functionality in the framework (e.g., event structure in MXML format). As we have already indicated in the implementation section of the visualization method, we had to consider certain requirements for being able to manipulate events to render workflow elements states. These requirements should be considered by everyone that is developing import plug-ins to MXML logging format in ProM Import tool, in order to provide the necessary information required by the Element State Analysis plug-in.

The implementation of ESA assumes the existence of:
- data specific events
- special attributes for assign-like events

The first assumption indicates that data is treated in the MXML log at the ATE level, considering the specific data events of the model presented in Figure 17. The second assumption mentions that all task assign events (or any task event type with similar meaning) contain information about the assigned workers to the task. Every assigned worker to the task should have his name available in the ATE as the value of an attribute. The implementation of ESA requires that this information is available in the log, as explained in Figure 25. This is the only situation where specific attributes for ATEs are required.

In case data specific events (as defined in Figure 17) are not available in the log, the data perspective is not able to show any data states. These events are necessary to be able to display data states (i.e., they are hard coded). In case some data event is missing, then the states can be defined only using the available ones. If no data event exists in the log, then the perspective would be empty. Additionally, if the assignedUser attribute is missing from the assign-like event, then the originator state Assigned cannot be visualized.

4.4 Evaluation

In this section, we present the performance evaluation of ESA. The functionality offered by ESA looks very promising especially for the analysis of semi-structured processes in industry. Therefore, we performed several tests with various logs to stress the execution of the plug-in. The purpose is to see how the plug-in can cope with log files of various sizes and how certain characteristics of the log can affect its performance. The results of our measurements provide an indication on how the tool may behave in a real working environment.

Apart from quantitative factors such as throughput time and memory consumption, we also look at qualitative aspects. It is relevant to see how the number and distribution of the events in a log may affect the quality of the visualization. Moreover, analyzing the data perspective is of great importance in the context of data-driven systems such as case handling. Since we did not have an appropriate log containing all data specific events, we had to create a synthetic log to be able to test the data perspective in ESA. The data perspective screenshot shown earlier in Figure 27 is based on this simulation log. To generate such logs, we considered a Colored Petri Net (CPN) model representing a case handling system to simulate using the CPN Tools [87]. The CPN model proposed by [71] incorporates the basic principles of case handling; therefore, we adopted it as a basis for our simulation. However, we had to extend it in order to obtain the necessary data events in the log. Details about the simulation and the CPN model are given in the next subsection. The resulting simulation log was further used for the performance measurements of the ESA plug-in. The performance results are presented in Section 4.4.2.
4.4.1 Simulation of a case handling model in CPN Tools

The CPN model in [71] describes a reference model that contains the basic aspects of a case handling system, which we previously explained in Section 3.2.1. The model is made of three parts:

- task lifecycle
- user interaction
- data lifecycle

The business process considered for simulation is not displayed in the model layout, but it is encoded in the net’s marking. However, the role of data in the execution of a case is not thoroughly modeled in [71]. Thus, we extended the data lifecycle model such that we could log the necessary data events: define, update, rollback, confirm, unset. Also, we added time to the Petri net, to simulate the duration necessary for certain user actions (e.g. define, update, redo) to be executed. Time is important to obtain various durations for the states of process elements that could be visualized in the ESA plug-in. Moreover, we added some additional constraints to the task actions (e.g. redo) to ensure data integrity and prevent concurrent access to data elements. A description of the initial CPN model is presented in [71], whilst the changes we made are given in Appendix A. Figure 28 shows the enhanced data lifecycle view of the simulated model:

Figure 28: Data lifecycle model

The four tasks in the Petri net of Figure 28 are used to log the data specific events. Also, the rollback data event is registered when some task is being redone. This event is not modeled in this net, but it is taken care of in the task lifecycle part (Figure A2, Appendix A). With these changes made, the CPN model becomes a better representation of a case handling application, and allows us to generate logs that could be used for analysis of the ESA plug-in. Moreover, by inspecting these logs, we can also make some assumptions about what information we can find in logs generated by applications such as CHEF. Due to the fact that the simulated process is encoded in the net’s marking, the behavior is more limited than that of a real case handling process. However, this does not diminish the importance of the simulation. The biggest advantage of the simulation is that it provides us with rich information...
about the data perspective and the close interplay between data and tasks. The resulted log gives us the opportunity to investigate these aspects and build up knowledge on how to better analyze semi-structured processes. The screenshot in the Figure 27 shows the data perspective in the ESA, based on simulation logs from the case handling model.

Next, the simulation logs resulting from this simulation are used for performance analysis of the ESA plug-in. The measurements and the results are presented in the following section.

### 4.4.2 Performance analysis

In this section we present the measurements we conducted in order to assess some performance characteristics of the ESA plug-in. The focus of these measurements is on *computation time* and *memory consumption*. We consider that these aspects are of most interest for the users of ESA. The logs obtained from the simulation of the case handling model mentioned previously are used for testing the plug-in. Additionally, we also used an industry log for testing. The measurements were performed on a Microsoft Windows XP system, with an Intel Pentium processor of 1.86GHz and 1GB RAM.

It is important to mention that the performance of the ESA depends not only on the number of events existing in the log, but also on the number of process instances (i.e. cases) and the way events are distributed over time and ordered in the log. It is expected that the more condensed the view is (i.e. more states to be computed), the more time it takes to be displayed in ESA. Also, the visualization method requires the events in the log to be sorted by timestamps, therefore the bigger the disorder in the log, the bigger the computation time. Due to these issues, we decided to measure the computation time and the amount of consumed memory in different settings. In this way we can better understand how different factors affect the performance.

The settings we investigated are the following:

- a) One case, variable number of events, highly disordered log
- b) 2000 events, variable number of cases
- c) Variable number of cases, each with 20 events
- d) Industry log with 154966 events and 24 cases

The computation time we measured represents the time necessary to display one perspective in ESA (i.e. the time necessary to paint all states in the view). The calculation of the execution time was included in the plug-in code. The heap memory we monitored reflects three operations in the ProM framework: opening ProM, loading the log and opening the ESA plug-in (opening ESA includes displaying of the task perspective\(^1\)). Since ESA is part of ProM framework, these three operations are closely related, thus it seemed sensible to monitor them all together. It is rather difficult to perform an objective measurement of the opening ESA operation alone, when other actions triggered in the background by the framework consume also space resources. Therefore, the results we obtained indicate the memory consumption within the scope of the ProM framework. For memory monitoring we used the Java profiler application JProfiler 5.0.1 [88].

Further on, we present the results of our measurements, for each of the above settings. The tables with all the measurements taken are given in the Appendix B.

**a) One case, variable number of events, highly disordered log**

In this situation we assume the log contains only one case, where the events are not sorted, but highly disordered. To stress the plug-in even further, all the files contain the events in the same time span. This means that the log file with 16000 events will have the most condensed view in ESA (i.e. more states need to be displayed in the same size region). We tested 7 log files containing from 250 to 16000 events, each file bigger than the previous one by a factor of 2. For each file we made 5 tests,

---

\(^1\) When a user selects to apply ESA on some log, the plug-in opens by displaying the task perspective as default.
and we calculated the mean and the confidence intervals (not visible here). The measurements and confidence intervals are given in Appendix B, case a). Finally, we obtained the following results.

Figure 29 indicates that the computation time for displaying a perspective in ESA is linear in the size of the log. We need to stress here that the values are exponential, on both axes (i.e. a logarithmic scale is used). This is a good result, considering the fact that the visualization method scans the entire set of events corresponding to the workflow element type selected to be analyzed. The second graph displayed in Figure 30 indicates the memory usage when performing the three operations: open ProM, load the log file, and open ESA. The memory used hints also to a linear trend; the graph seems to have an exponential growth, but the values on X-axis are also exponential. Even though our measurements are based on a rather small sample (i.e. 7 log files), we consider the results to be a good indication for the trend of bigger log files.

Due to the high disorder in the log that severely affects the time for loading and processing it, we restricted our tests to a maximum of 16000 events. We tested also a log with 64000 events log, just to get an idea about how slow it is. Loading and processing the log took almost 180 minutes, while opening the ESA plug-in (including the display of the task perspective) took almost 7 minutes. We also registered that after loading the log, opening the plug-in doubles the memory consumption, reaching approximately 26MB. We want to emphasize that this long execution time is not a problem of the plug-in, but of the loading the log. Although the log can be loaded, it is desired that the user does not have to wait for almost 3 hours to be able to use it in ProM. However, we remind here that we used an extreme log (i.e. highly disordered), to stress the execution. In general, event logs are ordered by timestamp. The difference in processing time of a sorted log will become clear in the last test setting, where we present the monitoring results of an industry log with almost 155000 events.

b) 2000 events, variable number of cases
In this setting we assume a constant number of events in the log file, but distributed over a variable number of cases. We used log files with 1 up to 16 cases. This sampling was convenient to keep in each log file the cases balanced, having the same number of events. The number of cases for each file is chosen by a multiplication factor of 2.
We can see that the display time of the task perspective tends to increase marginally with an increase in the number of cases, while the memory slightly decreases. Due to the small sample monitored we can not guarantee that this trend for memory consumption will persist when the number of cases increases even higher. It is probable that the memory increases when there are hundreds of cases, or remain stable in this range. However, the processing of the log seems to consume less memory if the events are spread over several process instances rather than one, due to the way the log is being stored by the framework.

c) Variable number of cases, each with 20 events
In this situation we monitored the time and memory for several log files containing variable number of cases, and each case having 20 events.

Analogously to setting a), the first figure indicates the same linear increase of execution time when the number of cases and, implicitly, the number of events in the log are increasing. The memory consumed is expected to grow with increasing the number of cases and events, following a trend similar to the one in Figure 30. Even though for a log with few cases it is possible to use less memory than a log with only one case, once the number of cases rises, it will consume more memory due to increasing number of events.

d) Industry log with 154966 events and 24 cases
For the previous measurements we used the simulation logs of the case handling model. In this setting we want to test how ESA performs on a real industry log. The log contains information about the testing process of a high-tech systems manufacturer. It contains 154966 events distributed over 24 cases, with an average of 6450 events per case. The log contained the events ordered by their timestamp, therefore the loading and pre-processing did not consume much time, in comparison with
the 64000 events log tested in setting a). The time necessary to change the perspective in ESA is in on average, approximately 2 minutes. Assuming a normal distribution for the samples we took, then in 95% of the cases, time is in between 131021 and 136401 ms. The 12 samples taken to measure the display time are given in Appendix B, Table B13. This shows that working with such a big log may not be very convenient when using a computer with such modest specifications. However, the perspective is fully loaded with painted states, as it can be seen in Figure 35, which explains the high processing time.

For the memory consumption we monitored again the same three operations. For open ProM operation we obtained an average of 9.69MB. To load the log file takes 12.91MB, and for opening the ESA plug-in uses 97.79MB, on average. All the measurements are given in Appendix B, Table B14. We can see that the memory usage grows significantly for this log, when opening the ESA plug-in. Since opening the plug-in also displays the default perspective, with all corresponding events, it definitely affects the performance. Still, for an industry log, 100MB of memory usage is not a big issue. Thus, it is expected that for bigger logs the memory consumption would stay at a reasonable level.

In conclusion, we can say that ESA is performing rather well for a visualization application. We have seen that log characteristics such as the ordering of events can put additional stress on the application. While the memory necessary to process a real life log is acceptable for an industry setting, the processing time for using the plug-in may be too long for a truly interactive experience. However, this problem can be overcome by enhancing the processing power of the system. Moreover, Figure 35 indicates that too many events can affect the quality of the visualization, by generating an overloaded picture. Even though the zooming functionality usually can help clarify the image, the user of ESA should consider the tradeoffs in using such large logs.

4.5 Benefits and Limitations

By providing the user with means for the interactive exploration of the log data, without assuming a process model, the Element State Analysis plug-in enhances the set of discovery techniques available in the process mining framework ProM. The data visualization method provides an innovative way of presenting event data in an effective, accurate and adaptable manner. Knowledge about the states and lifecycles of different workflow elements becomes clear and intuitive.

The distinct views presented for different types of workflow elements and using different time options or other adjustable settings allow for a better comprehension of the process execution. By comparing the various perspectives new dependencies or relationships between them can be established. Patterns in the data view may reveal the cause of some less efficient tasks or give more contextual information about changes or deviations in the process, especially when flexible systems allow workers more
freedom in performing their work. The data perspective can bring even more information by correlating it with the originator view. New insights about the way people execute tasks or the authorization role they have in a process can be derived. Being able to visualize the states of the various elements enables an effortless perception of their performance and easy detection of problematic parts in the process. As a consequence, a better support for mining the case perspective is provided.

Also, special attention is paid to the data analysis. The data perspective of the plug-in brings data forward as a first class citizen. Whilst the importance of data has been downplayed in the process mining area, researches recognize the need for a combined analysis of data and tasks [21, 23]. In order to discover the interrelations between the two perspectives we need to treat data at the same level of importance. Only this way can we provide appropriate support for flexible, less structured processes, where data is tightly connected to the control flow. In order to show the potential of this perspective, we simulated a case handling model to generate logs with specific data events. Figure 24 displays the results based on the simulation log.

The separation of the four distinct perspectives is allowing a thorough analysis of each workflow element type. However, we admit that finding correlations in these separate views is not always straightforward. In analyzing large logs might be difficult to find the subtle interrelations among them. An improvement of the visualization tool could be realized by implementing an additional synchronous view on the data, originator and task. More clearly, the purpose of this view is to display in one window, all three perspectives, in parallel, such that dependencies between data and task events, as well as the responsible resources for the events would be easily identified. Synchronizing the scrolling on the time axis for all three perspectives would enable an efficient scanning of the elements’ lifecycle and provide a comprehensive view on the case evolution in time. Due to time constraints, we were not able to implement this view, but we consider that it can be a valuable addition to ESA.

Moreover, due to the incipient stage of the approach of considering data at the same level as the activities in a business process, we had to deal with some limitations. We hope that by presenting them we can create awareness and lay out the requirements for further improvements in this direction. First, since there are few information systems that actually log detailed information about data modifications in the context of business process monitoring, we had to assume a data transactional model for data specific events (Figure 17). This model, as it is, might be too restrictive for certain application domains, but allowing an extension mechanism may enable the necessary flexibility. Extension of the ATE structure in MXML accommodating a special attribute to identify the type of the workflow element described by an event would be a good alternative. However, this proposal needs to be evaluated in the context of the entire ProM framework before taking a decision whether to adopt it or not.

Another issue that we had to deal with is the noise in the log. In situations when main states (e.g. Running, Defined, Assigned) do not have a proper end event, then the end of the corresponding process instance was assumed to finish the state duration. This situation can generate confusion or misleading information, therefore it should be carefully addressed. Events that generate states without having a corresponding end event may be discarded from the log to avoid such situations. On the other hand, identification of these “hanging” states may give some information about missing operations in the process, provided that the logging of the events is assumed correct (i.e. errors in the monitoring procedure are excluded). However, we recognize the danger of these situations that can jeopardize the interpretation of the information displayed.

Concerning the performance of the ESA in the context of the mining framework ProM, Section 4.4 shows the evaluation results obtained when testing the plug-in. We consider that ESA exhibits a good behavior in dealing with logs of various sizes and complexities. Both memory consumption and processing time appear to have a linear dependency with the size of the log. The memory consumption is reasonable for an industry log size (see Figure 35), considering that testing was conducted on a
modest machine, and not on a real-life computer configuration. Therefore, we expect this not to be an issue in using ESA in a real industry setting. The major drawback is, however, the processing time. Performing various operations in ESA, such as zooming or scrolling, can affect significantly the execution time. This aspect can be addressed by increasing the processing power of the system. Moreover, we showed that not only the size of the log is important, but also the distribution of the events in the log. In situations with many process instances and high concurrency, the perspectives may become incomprehensible, due to high density of states to be rendered. The screenshot given in Figure 34 is a good example of this situation. The user of the plug-in should adjust the settings appropriately such that the quality of the visualization is not hindered.

In conclusion, we envision a wide application of the new data visualization method in the area of process knowledge discovery. If analyzing structured processes may not reveal much unexpected behavior, due to the repetitive patterns in their execution, the flexible cases and processes with high human involvement are definitely very interesting to analyze using ESA. The benefits of this new visualization approach become obvious in these situations. We strongly believe that the Element State Analysis plug-in will add value to the collection of available mining techniques, and we place it as a visualization technique in the spectrum of process mining:

![Figure 36: ESA in the process mining spectrum](image_url)
5 IBM WebSphere

In the previous chapter we presented our contribution to mining techniques addressing the data perspective of business processes. Visualizing the lifecycle of various types of process elements may reveal interesting behavior especially in flexible, semi-structured processes. To apply this knowledge on a real system, we carried out a case study of mining business processes deployed in IBM WebSphere environment.

This chapter presents an overview of IBM WebSphere technology and standards available for business process support. The first section briefly presents the IBM WebSphere suite for business process management and related technologies that offer interesting process analysis functionality. The second section of this chapter presents the logging infrastructure implemented in WebSphere. Here, we describe the logging format that is used to store the events, as well as the event types that may be logged during process execution. This knowledge is indispensable for realizing the conversion of the WebSphere logs into the mining format MXML, necessary for process mining in ProM. The results of the case study are going to be presented in Chapter 6.

5.1 IBM technology for business process support

In a highly dynamic business environment, where business process changes are happening much more frequently nowadays, flexibility and responsiveness are imperative. Streamlining business processes can be very demanding and costly, often triggering changes in the IT infrastructure, which may also lead to incompatible solutions. Therefore, the challenge is to integrate and share processes as well as IT assets inside and outside of an organization. Service Oriented Architecture (SOA) is about to structure business processes and IT assets into interchangeable components (services) that can be reused or changed [3, 11]. Based on Web Services technology, SOA provides a framework for systems and enterprise integration that allows information systems to be more flexible in adapting to dynamic business requirements.

IBM WebSphere products are built on top of a SOA foundation that provides extensive support for business and IT standards, achieving greater interoperability and portability between applications. This makes business process management more challenging and necessary. However, this flexibility is often undermined by rigid modeling structures provided in the WS-BPEL [10] (i.e. the modeling language for business processes used in WebSphere), especially when people are involved in these processes. Efforts to relax the modeling constraints and provide more flexibility emerged into new extensions of WS-BPEL such as BPEL4People (BPEL for human activities) [5], BPEL-SPE (BPEL for sub-processes) [6], and BPELJ (BPEL for Java) [7], that are proposed as new standards in this.

IBM business process management suite

The extensive WebSphere family has many products that support process management at various levels. The standard IBM business process management suite solution includes development tools and business performance management tools used to monitor and manage the runtime implementations at both the IT and business process level. Figure 37 displays the components that are used to provide support during the lifecycle of business processes:
Figure 37: IBM support for business process management lifecycle

Built on top of the SOA foundation, these tools allow the integration of business processes as business services. A process can invoke a service, and at the same time it can expose its own interface to make its functionality available for other applications. In WebSphere Process Server, a business service is called a component. A business process can be composed of several components. The Service Component Architecture (SCA) is the service component oriented model of the WebSphere Process Server that separates the business logic from the implementation. The benefit of SCA is that it allows components to be reused and combined in larger applications components, statically or dynamically, allowing for a greater degree of flexibility.

However, the functionality of the analysis tools is rather limited, as most the commercial products in the Business Process Management market. The WebSphere Business Monitor only measures the performance of a process based on the key performance indicators (KPI) and the business metrics defined in WebSphere Business Modeler. Performance related results are displayed in dashboards and used as reference for redesign. The monitoring and analysis tools are not able to discover causal relations between tasks or employees involved in the process, and, thus, they can not extract a process model from the event log. Moreover, an audit of the process to see if it conforms to the organizational procedures and regulations is hardly objective or efficient without having a good understanding of the real process. Detailed information about the monitoring capabilities of this suite can be found in [3].

Web Services Navigator
Another interesting application is IBM Web Services Navigator, an Eclipse/RAD (Rational® Application Developer) plug-in for interactive visualization of web service transactions based on SOAP messages [8]. The Web Services Navigator visualizes logs generated by web service activities in the IBM WebSphere Process Server, and displays the discovered behavior of the web service applications in different abstract views.

According to [9], the Web Services Navigator can be used for verification of workflow choreography or verification of the correctness of the business rules implementation. Still, the process analysis is bounded to the service components level, as the investigation addresses only the way these components interact, and what messages are sent between them. Thus, it is highly possible that errors or deviations inside the components implementation may go undetected. Also, there is no automated discovery of the errors or highlighting of bottlenecks.
Nevertheless, the message sequence charts and the graph views of the systems topology make a step forward into the web services mining arena. For more information about the Web Services Navigator the reader is referred to [8, 9].

**Eclipse Test & Performance Tools Platform plug-in (TPTP)**

Eclipse TPTP is an open platform and not a proprietary IBM product. However, the long term contributions of IBM to development of Eclipse are well known, and therefore we have decided to mention this tool as well. Eclipse TPTP provides interesting functionality for monitoring applications executions. This is an Eclipse plug-in for evaluating the overall performance and quality of Java source code.

One project of the TPTP framework is the TPTP Monitoring Tools that focuses on the monitoring and logging phases of the application lifecycle. It includes tools for monitoring application servers (e.g. WebSphere) and system performance, and also tools for collecting, aggregating, analyzing and visualizing data captured in the log and statistical models. Application events are captured and correlated, allowing for a more structured analysis of distributed application problems. The framework allows the import, filtering and correlation of various log files and includes symptom databases against which the log files can be analyzed. A symptom database provides solutions or indications about how to solve the problems identified. A more detailed description of Eclipse TPTP can be found in [35].

In this section we have described a few IBM technologies that aim at providing process management support. In general, their functionality is rather limited in comparison to process mining tools such as ProM. However, interesting tools such as Web Services Navigator and Eclipse TPTP are opening new perspectives for mining processes deployed in a SOA environment. The next section describes the logging functionality included in WebSphere.

### 5.2 WebSphere logging infrastructure

WebSphere is a complex environment, where various components are connected to offer the desired services. It provides a service-oriented process management based on SOA. In order to track the interactions of web services in choreography and monitor the execution of business processes it is essential to have a logging infrastructure. This subsection describes the monitoring infrastructure for logging events, the event format used and the predefined events specific for business processes deployed in WebSphere. This way we hope to facilitate the understanding of how the monitoring of business processes in WebSphere is operated and what information can be tracked.

#### 5.2.1 Common Event Infrastructure

The Common Event Infrastructure (CEI) is an IBM embeddable technology intended to provide basic event management services. It provides facilities for the generation, propagation, persistence and consumption of events. The major benefit of using a common infrastructure is that diverse products that are not tightly coupled with one another can integrate their management of events, providing an end-to-end view of enterprise resources and correlating events across domain boundaries.

Events are represented using the Common Base Event (CBE) model, a standard XML-based format defining the structure of the event. The CBE format is described in Section 5.2.2. CEI and CBE support are core parts of WebSphere Process Server, thus being available to any application that is built on top of this infrastructure (Figure 38). CEI enables event generation at different levels for all Service Component Architecture (SCA) components. The architecture shown in Figure 38 lists the main SCA components of the WebSphere Process Server: Business Processes, Human Tasks, Business State Machines and Business Rules. The complete list with monitored components in WebSphere Process Server can be found in [2]. Each type of component has a predefined set of business events that can be stored. Moreover, the log detail can be appropriately set up for each component, individually, from a low degree such as “info” up to the “finest” degree of detail.
For our case study, we focus on the Business Processes and Human Tasks. These two types of service components are managed by WebSphere Business Process Choreographer, which is part of WebSphere Process Server. This will be introduced in Section 5.2.3, where we describe how it monitors the two components.

We have presented CEI as an integrated component of the WebSphere Process Server, but it is also part of WebSphere Enterprise Service Bus, another IBM deployment environment. Thus, CEI is available to all applications using one of the two as a runtime environment. A few examples of WebSphere products using the management services for events offered by CEI are: WebSphere Partner Gateway, WebSphere Adapters (Email, FTP, JDBC, SAP software, Siebel Business Applications, PeopleSoft Enterprise, etc), WebSphere Business Monitor, and WebSphere Business Modeler [3]. This makes it interesting to further investigate the applicability of CEI and find out how to avail of its capabilities. The architecture of CEI is presented in Appendix C.

Further on, we introduce the logging standard used by CEI: the Common Base Event format.

### 5.2.2 Common Base Event

The Common Base Event model is a standard defining a common representation of events that was developed by IBM [4]. This standard supports encoding of logging, tracing, management and business events using a common XML-based format which makes it possible to correlate different types of events that originate from different applications.

An event represented in the CBE format reports information related to a *situation*. A situation can be anything that happens in the computing infrastructure, e.g. server shut-down or a failure user login. The information reported describes the situation itself, the identity of the affected component and the identity of the component reporting the situation (often being the same as the affected component). The structure of a CBE contains mandatory properties such as `globalInstanceId`, `version`, `creationTime`, `sourceComponentId`, `situation`, and optional properties such as `priority`, `severity`, `msg`, `extensionName`, `contextDataElement` and `extendedDataElement`. These attributes are listed in the XML schema of the CBE format, depicted in Figure 39. For a complete specification of the standard, we refer the reader to the specification document [4].

The most important attributes for capturing business related information are the `contextDataElement` and the `extendedDataElement` (see Figure 39). The first one allows saving the process instance id and parent process instance id, thus enabling the correlation of process instances. The second one captures information about activities, control-flow links and variables in the process. It is of foremost importance that the event format allows the identification of the process template and the process...
instance. This information is essential when mining the execution logs for discovery of the business processes. Additional information stored in the event such as the creation time or the person executing some activity enables more knowledge to be extracted during process analysis. Not only information about the control-flow structure can be discovered, but also information about the staff assigned for each case, what decisions they’ve made during the case, or how data was manipulated in the execution of the process.

With respect to the IT-related events, the most relevant property of the CBE object is the *sourceComponentId*. For example, when monitoring a Java application, the attributes of this element retain valuable information about the component thread Id, execution environment and location of the component that generated the event.

Furthermore, the CBE format is extensible. This means that in addition to the standard event properties, an event can also contain *extendedDataElements*. These are application-specific elements that may contain relevant information for the case. Therefore, using this extension mechanism, the event format can be easily adapted to a variety of event types generated by different IT or business applications.

The event structure facilitates storing data at both the IT and the business level. Also, the event can be extended with additional, application (business) specific information. Due to this elaborate structure of the CBE event, different types of processes and applications can be easily monitored. Moreover, other tools independent of WebSphere and CEI, such as the Eclipse TPTP plug-in, adopted the CBE format for logging events, due to this versatility. Naturally, extending the usage of the CBE format will allow applications to have their executions more easily and efficiently monitored.

The next section presents the events that are emitted during the process execution by the WebSphere Process Server runtime environment.
5.2.3 WebSphere Business Process Choreographer and CEI

This section explains how business processes and tasks performed by people are actually managed and monitored in WebSphere and what kind of events are sent and stored in the CEI database. This information is valuable for understanding what kind of information is available in the log, and how it can be retrieved at mining phase.

As part of the WebSphere Process Server V6.0, the Business Process Choreographer (BPC) provides the support for two types of service components: business processes and human tasks. Human tasks components represent activities that are performed by people involved in the process (also named staff activities). Business Process Choreographer manages the life cycle of business processes, navigates through the associated process model, invokes the appropriate web services, and involves persons to handle staff activities. Therefore, we intend to investigate what aspects of a process BPC can monitor and what information can be retrieved later for process analysis.
Events in BPC can be captured either as Audit Log and stored in the BPC database, or as CBE events, and saved in the CEI data store. Since the CEI is the monitoring infrastructure that can monitor all WebSphere Process Server components, we focused on how this can be used to capture and consume process events. The Business Process Choreographer provides a predefined set of events that can be enabled for process relevant BPC objects (i.e. types of process elements) [2]:

- process instances (e.g. process started, completed, failed)
- activities (e.g. activity ready, failed)
- variables
- control-flow links
- human tasks (e.g. task claimed)

Additionally, we have considered as relevant for our case study the events at the SCA level, which store information about interactions of service components. A process can be assembled in a choreography of sub-processes, represented by service components. Hence, the SCA events enable a process analysis at a different level of abstraction. These events are managed by WebSphere Process Sever itself, and not by BPC. Thus, they will be described separately. More components are monitored by the WebSphere Process Server (e.g. business rules, business state machines, adaptors, mediation components), but we have not included them into our case study, due to complexity and timing constraints. However, the selected components provide all the necessary information for process mining in WebSphere.

First of all, there are some CBE properties that are common to all BPC emitted events, providing information about the execution environment. The importance of this data becomes obvious for the monitoring of distributed applications, in a SOA environment. The most relevant ones are presented in the following table, which is based on the specification given by [4]:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>creationTime</td>
<td>The time at which the event is created.</td>
</tr>
<tr>
<td>globalInstanceId</td>
<td>The identifier of the common base event instance. This ID is automatically generated.</td>
</tr>
<tr>
<td>extensionName</td>
<td>The value depends on the process choreographer object that creates the event. E.g. ‘BPC.BFM.ACTIVITY.STATUS’</td>
</tr>
<tr>
<td>sourceComponentId</td>
<td></td>
</tr>
<tr>
<td>component</td>
<td>For business processes and human tasks: identification of the current platform and the version identification of the underlying software stack.</td>
</tr>
<tr>
<td>executionEnvironment</td>
<td>A string that identifies the operating system.</td>
</tr>
<tr>
<td>instanceId</td>
<td>The identifier of the server.</td>
</tr>
<tr>
<td>location</td>
<td>Set to the host name of the executing server.</td>
</tr>
<tr>
<td>processId</td>
<td>The process identifier of the operating system.</td>
</tr>
<tr>
<td>threadId</td>
<td>The thread identifier of the Java Virtual Machine (JVM).</td>
</tr>
<tr>
<td>componentType</td>
<td>To distinguish between business process and human task</td>
</tr>
<tr>
<td>extendedDataElement</td>
<td></td>
</tr>
<tr>
<td>attributes</td>
<td></td>
</tr>
<tr>
<td>BPCEventCode</td>
<td>The BPC event code that identifies the type of the event</td>
</tr>
<tr>
<td>processTemplateName</td>
<td>The name of the process template.</td>
</tr>
<tr>
<td>processTemplateValidFrom</td>
<td>The date of the process template deployment</td>
</tr>
</tbody>
</table>

Table 1: CBE properties common to BPC events
Clearly, the properties of the sourceComponentId store the necessary information for retrieving at the mining stage the topology of web services choreography. Along these properties for the IT level, BPC sets also some common extendedDataElement attributes that store business information like the process template name. The extensionName identifies the type of the process element that generated the event: process instance, activity, link or variable. Other event data relevant for each type of element is described separately in the remainder. A complete description of the content of these events is provided in [2].

**Process instance events**

Business process instances have different events emitted during their lifecycle. For every process instance event, the following CBE properties are set automatically by BPC, as contextDataElements (see Figure 39):

- The ECSCurrentID contains a string representation of the process instance ID (PIID) of the process that emitted the event.
- The ECSParentID is set to the value of the ECSCurrentID before the process instance start event of the current process.

The extendedDataElement contains additional properties:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>processTemplateName</td>
<td>The name of the process template from which the instance was derived.</td>
</tr>
<tr>
<td>processTemplateValidFrom</td>
<td>The date from which the template is valid.</td>
</tr>
<tr>
<td>processTemplateID</td>
<td>The identifier of the process template.</td>
</tr>
<tr>
<td>processInstanceDescription</td>
<td>If this attribute is set, it is the description of the process instance.</td>
</tr>
<tr>
<td>Username</td>
<td>The name of the user who created the process instance.</td>
</tr>
<tr>
<td>processInstanceExecutionState</td>
<td>A string representation of the state of the process.</td>
</tr>
<tr>
<td>processFailedException</td>
<td>If a failure occurred, a string description of the exception that caused the failure</td>
</tr>
</tbody>
</table>

Table 2: CBE ExtendedDataElements properties for process instances events

The first two attributes were given also in Table 1. However, we have repeated them here to give all important properties specific for process instances. We also presented them separately, as they are contained in a different type of attribute: contextDataElement rather than in an extendedDataElement. For the next types of process elements, we make the same distinction.

**Activity events**

Like for the process instances, the activities of a business process have their own lifecycle managed by Business Process Choreographer. For every activity event the following properties are set automatically as contextDataElements:

- The ECSCurrentID contains a string representation of the activity ID (AIID) of the activity that emitted the event.
- The ECSParentID contains a string representation of the process instance ID (PIID) of the containing process instance.

Additional information is stored in the extendedDataElement:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activityTemplateName</td>
<td>If this attribute is set, it is the name of the activity template from which the instance was derived.</td>
</tr>
<tr>
<td>activityInstanceID</td>
<td>The identifier of the activity instance.</td>
</tr>
<tr>
<td>activityKind</td>
<td>A string value that identifies the activity kind, for example KIND-INVOKED.</td>
</tr>
<tr>
<td>state</td>
<td>A string value that represents the state of the activity, e.g. STATE-RUNNING.</td>
</tr>
<tr>
<td>bpelId</td>
<td>The identifier of the activity in the BPEL file. It is unique for activities inside a process model.</td>
</tr>
<tr>
<td>username</td>
<td>The user who claimed the activity.</td>
</tr>
</tbody>
</table>

Table 3: CBE ExtendedDataElements properties for activities events

**Link events**

Links refer to the transitions between activities, which are evaluated to determine which path in the process can be followed. For link events, the following information is written as `contextDataElements`:
- The ECSCurrentID provides the ID of the source activity of the link.
- The ECSParentID provides the ID of the containing process.

Control-flow links have the following additional information sent through events as `extendedDataElements`:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elementName</td>
<td>The name of the link that was evaluated.</td>
</tr>
<tr>
<td>Description</td>
<td>If this attribute is set, it contains a description of the link.</td>
</tr>
</tbody>
</table>

Table 4: CBE ExtendedDataElements properties for link events

**Variable events**

Variable events refer to updates of the data elements used in the process. For the variable events, attributes as `contextDataElements` are:
- The ECSCurrentID provides the ID of the containing scope or process.
- The ECSParentID is the ECSCurrentID before the process instance start event of the current process.

Variables have also the following information sent through events, as `extendedDataElements`:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elementName</td>
<td>The name of the variable that was changed.</td>
</tr>
<tr>
<td>variableData</td>
<td>A string representation of the content of the variable. The string represents the name-value pairs.</td>
</tr>
<tr>
<td>variableData_BO</td>
<td>The variable's data content in a Business Object (BO) representation.</td>
</tr>
<tr>
<td>bpelId</td>
<td>The ID for the variable.</td>
</tr>
</tbody>
</table>

Table 5: CBE ExtendedDataElements properties for variables events

**Human task events**

BPC distinguishes between common BPEL activities (e.g. assign, flow, receive [10]) and human tasks [5]. While the common activities are used to orchestrate interactions with Web services, the human
tasks objects are specially built for managing human interactions in a business process. For task events there are the following properties as contextDataElements:

- The ECSCurrentID provides the ID of the task instance.
- The ECSParentID is the ECSCurrentID before the task instance event.

Additional information that may be stored by a task event:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskTemplateName</td>
<td>The name of the task template from which the instance was derived.</td>
</tr>
<tr>
<td>taskTemplateValidFrom</td>
<td>The date from which the template is valid</td>
</tr>
<tr>
<td>taskTemplateId</td>
<td>The identifier of the task template from which the instance was derived.</td>
</tr>
<tr>
<td>username</td>
<td>The name of the user who is associated with the event</td>
</tr>
<tr>
<td>current</td>
<td>The name of the current owner of the work item</td>
</tr>
<tr>
<td>target</td>
<td>The name of the new owner of the work item</td>
</tr>
<tr>
<td>followTaskId</td>
<td>The ID of the task that was started as a follow-on-task</td>
</tr>
</tbody>
</table>

Table 6: CBE ExtendedDataElements properties for human task events

Service Component Architecture events
WebSphere Process Server provides events for SCA components as well, with additional data in the extendedDataElement:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE COMPONENT</td>
<td>The component sending the message</td>
</tr>
<tr>
<td>SOURCE INTERFACE</td>
<td>The interface sending the message</td>
</tr>
<tr>
<td>SOURCE METHOD</td>
<td>The method sending the message</td>
</tr>
<tr>
<td>SOURCE MODULE</td>
<td>The module sending the message</td>
</tr>
<tr>
<td>TARGET COMPONENT</td>
<td>The component that should receive the message</td>
</tr>
<tr>
<td>TARGET INTERFACE</td>
<td>The interface that should receive the message</td>
</tr>
<tr>
<td>TARGET METHOD</td>
<td>The method that should receive the message</td>
</tr>
<tr>
<td>TARGET MODULE</td>
<td>The module that should receive the message</td>
</tr>
<tr>
<td>Input</td>
<td>The input data in the message</td>
</tr>
</tbody>
</table>

Table 7: CBE ExtendedDataElements properties for SCA events

The event attributes that we presented in this section are important for two main reasons. First of all, they allow us to understand what kind of information may be available in the log and to what extent we can apply the process mining techniques. Second, we need them to identify a proper translation from the CBE to the MXML format used in ProM, to be able to perform process mining on WebSphere logs. The conversion from CBE to MXML is presented in the next chapter, Section 6.2.

5.2.4 Conclusion
As we can see, BPC is sending large amounts of information with the emitted events. For every type of BPC object there are specific attributes that store information about the name, the content, and the context of that object (e.g. process instance, username, scope). The richness of the log makes it possible to answer various questions and derive meaningful knowledge about the processes executed,
both at the IT and the business level. On the other hand, we can expect that managing events containing so much information might affect the system’s performance. The monitoring configuration settings at both design and run-time allow the individual selection of components and/or objects of a process as well as the level of detail. An appropriate selection of elements to be monitored can avoid overloading the CEI event bus with events that contain extensive information which may result in a system performance penalty.

At this point we can observe a limitation in the definition of the business related events. The variable event (see Table 5) gives information about the name and the content of the variable, as well as about the moment of update. However, no information is given with respect to the activity where the variable is actually modified. Also, in general, the activity event does not contain any information about the data that is associated with it. Usually in a process, variables are reused in several parts (i.e. different activities). In this way the users working on the process instance are given several opportunities to modify it. Obviously, it is desirable to know in what place the variable had been changed and who was responsible for that change. Variables that are involved in taking a decision are even more interesting to be closely tracked to see how their value is affecting the decision outcome. Inspecting the WebSphere log to derive this kind of information may become a cumbersome task, when concurrency is involved, if deriving this information is possible at all, under these circumstances. Furthermore, BPC only registers information about the data updates; no other data events are being recorded. It seems that so far, monitoring data has not been considered of primary interest in WebSphere. Nevertheless, we can still use the Element State Analysis tool developed in ProM (see Chapter 4) to investigate the data perspective.

An important observation is that every event contains information about a process template and a process instance through the `processTemplateName` and `ECSCurrentID` or `ECSParentID` attributes. This allows us to organize the events based on each process instance, and therefore, it meets the requirement for process mining in ProM (see Chapter 2, Section 2.3). Moreover, the extensible feature of the CBE format allows for specific information not available in the common structure to be added to the events, customizing the event according to the needs.

Observing the event types and their data attributes that are stored by CEI we conclude that WebSphere can provide a good quality event log for process mining. Good quality refers here to the richness and availability of event data, as well as facility in collecting and correlating information about processes, which most of the current SOA based systems fail to properly support. As the study in [33] indicates, the majority of the SOA-based systems offer logging functionality only at the level of Web services interactions, and lack information about the workflow identifier and case identifier. In these situations, it is hardly possible to mine the business process. According to [33], there are five levels of logging in a SOA environment:

- Level 1: standard HTTP-server events
- Level 2: complete HTTP request and response events
- Level 3: web services container level
- Level 4: client activities events
- Level 5: process information

WebSphere is providing a powerful logging infrastructure, based on a detailed logging format, which addresses all monitoring levels in SOA. Unfortunately, existing monitoring and analysis IBM tools available for process management are offering only limited support. From the tools we presented in Section 5.1, the Web Services Navigator seems to deliver useful information about services choreography, but detailed insights into processes are still missing.

Thus, we can exploit this opportunity by using ProM and provide a more in-depth analysis of the business processes deployed in WebSphere. We can discover knowledge at a finer level, by applying various mining algorithms and methods (e.g. heuristics algorithms for process model reconstruction, social network analysis, data analysis). In principle, we can use ProM to investigate all the process
perspectives on such a log, and show the benefits of ProM’s functionality. A case study on mining WebSphere processes is presented in the next chapter.
6  Case study: Mining WebSphere logs

In the previous chapter we presented how and what kind of information can be logged when business processes are deployed in WebSphere. We used this knowledge to build a conversion tool based on ProMimport framework that translates the WebSphere logs into the mining format MXML. This enables us to perform process mining on WebSphere logs, using the ProM framework.

In the first section of this chapter we give the overview of the case study, by placing a few of the most popular plug-ins in ProM in the process mining spectrum. Some of them are used on WebSphere logs to perform process analysis. Section 6.2 in this chapter describes the development of the conversion plug-in for ProMimport. Section 6.3 discusses the process mining case study we performed on a regular BPEL process. Here we introduce the interesting and challenging aspects of analyzing BPEL processes. In Section 6.4 we describe the IBM Case Handling Execution Framework (CHEF) application and the challenges in mining CHEF logs. CHEF supports the execution of semi-structured processes, by offering case handling functionality. Hence, it is valuable to understand how the enhanced flexibility is reflected in the event logs.

6.1 Case study overview

Figure 40 presents the overview of this case study, displaying a few of the process mining algorithms and methods available in ProM, including the visualization plug-in, ESA. The figure refines the generic image with the scope of the thesis research, discussed in the introduction chapter, in Figure1. Some of these methods are used to mine the WebSphere logs:
Mining the WebSphere log is performed using the mining framework ProM, because it is probably the most advanced tool for process analysis at the moment, offering an extensive palette of algorithms and techniques for process mining, analysis, and verification.

### 6.2 Log conversion

In order to be able to analyze WebSphere logs in ProM, first we need to transform the event logs from the Common Base Event (CBE) format into the MXML format. To realize this conversion we implemented a plug-in for the ProMimport framework.

The CBE structure has been already explained in the previous chapter, in Section 5.2.2. This event representation is not addressing only the business events, but is used also for logging, tracing, and management types of events. Therefore, many attributes and components have been defined to store specific information about the execution environment and the situation in which the event occurred. The MXML structure described in Chapter 2, Section 2.3, is extremely simple, and relies mainly on two elements: the WorkflowModelElement and the EventType. However, an Attribute Trail Entry (ATE), the main cell of the MXML structure, can be easily extended with optional information such as the event timestamp or the originator (i.e. the person/system executing the task).

#### 6.2.1 Mapping CBE to MXML

A XML file with WebSphere logs contains a succession of common base events, in general ordered chronologically. To map this log onto the MXML structure, we actually transform each CBE event into an Audit Trail Entry (ATE). The final MXML log contains the ATE’s arranged according to the process templates and process instances existing in the log.

Due to the simplicity of the MXML structure, the mapping between the two formats is done by selecting for each of the main properties in the MXML structure, the CBE attribute that best represents its meaning. The following tables provide the corresponding CBE components of the main MXML elements, thus illustrating the mapping between the two formats:

<table>
<thead>
<tr>
<th>MXML element</th>
<th>Corresponding CBE element (type: property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcessTemplateID</td>
<td><code>cbe/extendedDataElement: processTemplateID</code></td>
</tr>
<tr>
<td>ProcessInstanceID</td>
<td><code>cbe/contextDataElement: ECSCurrentID (for process instances, variables, links, ) or ECSParentID (for activities and human tasks only)</code></td>
</tr>
<tr>
<td>WorkflowModelElement</td>
<td><code>cbe/extendedDataElement: processTemplateID (for process instance), activityTemplateName (for activity), taskTemplateName (for task), elementName (for variable and link), SOURCE MODULE/TARGET MODULE (for SCA), or BPCEventCode description (for other elements)</code></td>
</tr>
<tr>
<td>EventType</td>
<td><code>cbe/extendedDataElement: BPCEventCode (for process, activity, variable or link), HTMEventCode (for human task), or EventNature (for SCA)</code></td>
</tr>
<tr>
<td>Timestamp</td>
<td><code>cbe: creationTime</code></td>
</tr>
<tr>
<td>Originator</td>
<td><code>cbe/extendedDataElement: Username (for activities and human tasks) or &quot;System&quot; (when no &quot;Username&quot; property exists)</code></td>
</tr>
</tbody>
</table>

Table 8: CBE corresponding properties for MXML attributes
For the WorkflowModelElement we use the name attribute of the process, activity, link, variable, human task, or SCA module described by the CBE event. In situations where the event is not related to one of these categories because it may belong to a different type of component (e.g. Business Rules), we consider the BPCEventCode description attribute value for mapping to WorkflowModelElement. In this way, even if we do not specifically address certain types of components (e.g. business rules), they are not omitted in the conversion, so they will be available for mining.

There are two special mappings that need further explanations: EventType and Originator.

First of all, the information provided by BPCEventCode description is examined in the plug-in code so that it can be properly mapped to one of the predefined values of the EventType attribute in MXML. In Chapter 4 we presented the MXML transactional model for activities (Figure 12, page 32) and the transactional model for data (Figure 17, page 38). However, in comparison to these two models, BPC uses more than 80 event types (currently defined in [2]) for the specified process objects. Clearly, the definition of these events is more detailed than what is considered in MXML. For part of the event types there is a straightforward correspondence to MXML event types. In situations where no appropriate mapping is possible, we preserve the original CBE event description for the EventType attribute.

Nevertheless, there is a special case where we have to apply a different approach. The human task events emitted by the BPC cover a wide range of situations. Some events are very important for mining but they do not have the same semantics as the ones in MXML, so we have to establish an appropriate mapping to the EventType attribute. We describe here the meaning of some human task events in BPC:

- **Start**: indicates the moment when the execution of the process reaches the human task
- **Work items created**: assigns the task to a list of users authorized to execute it
- **Task claimed**: indicates that a certain user from the assigned list is selecting the task to work on it
- **Complete**: indicates the moment when the execution of the process leaves the human task

In business processes that include human interactions, we are interested to see when an assigned person starts and completes working on an activity, and not how the task is scheduled by the system. In this case, the MXML START event represents the moment when an Originator starts the execution of the task. Therefore, the BPC start event is mapped on the SCHEDULE event type in MXML. The mapping used for human task events is shown in Table 9:

<table>
<thead>
<tr>
<th>BPC Human Task event</th>
<th>Corresponding MXML EventType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task started</td>
<td>SCHEDULE</td>
</tr>
<tr>
<td>Task claimed</td>
<td>START</td>
</tr>
<tr>
<td>Work items created</td>
<td>ASSIGN</td>
</tr>
<tr>
<td>Work items transferred</td>
<td>REASSIGN</td>
</tr>
<tr>
<td>Task suspended</td>
<td>SUSPEND</td>
</tr>
<tr>
<td>Task resumed</td>
<td>RESUME</td>
</tr>
<tr>
<td>Task completed</td>
<td>COMPLETE</td>
</tr>
<tr>
<td>Task terminated</td>
<td>ATE_ABORT</td>
</tr>
<tr>
<td>Task failed</td>
<td>ATE_ABORT</td>
</tr>
<tr>
<td>Task expired</td>
<td>ATE_ABORT</td>
</tr>
</tbody>
</table>

Table 9: BPC human task events mapping to MXML

The task claimed event in BPC represents the moment when a user selects a task for which he is assigned to execute it. Still, this does not imply that the worker starts working immediately. He can claim a task and postpone working on it at a later time. However, there is no other event in BPC that is closer to the START definition in MXML. The rest of the human task events maintain their original description.
With respect to the **Originator** attribute of the ATE, we map the value of the extended data element *Username* available in the *activity claim* and *process start* CBE events. For the rest of the events we put into the corresponding ATE the default value *System* defined in MXML, as the Originator. The reason for keeping a default Originator (i.e. System) for the other audit trail entries is to be able to track the system actions for each process instance. This may help us understand how the system interprets and executes the definition of the process model. Another reason is that some types of BPEL activities such as JavaSnippet (Script) or Assign may be important for the business logic, representing automatic activities (e.g. returning the result of a web service invocation in a JavaSnippet). In this case it is important to know that the System was performing the task, and not some person.

Apart from the main components of the MXML format that have been set, other attributes from the CBE event are added to ATE to store additional information. For example, variableData, processTemplateValidFrom, or processFailedException, may provide useful information about the context of a process and content of data elements. A list with all additional attributes is given in Appendix D, Table D1.

An example of a CBE event mapped to MXML ATE element is given in the Appendix D. The next section describes the implementation in ProMimport of the conversion we have just presented above.

### 6.2.2 ProM Import conversion plug-in

The mapping procedure described above was implemented as a conversion plug-in for our ProMimport framework that provides log transformation functionality. As we mentioned already, BPC can send a large amount of data and events even for a small process that is monitored. However, we need to filter the logged information and retain what is in fact useful for the mining goal. For example, somebody interested in mining IT related events can discard many of the business specific events that do not influence the knowledge extraction for that perspective. For this reason it is important to refine the log and preserve only the information that best serves the mining purposes. The utility of the filtering properties is further stressed in the phase of using ProM to mine and analyze the log.

In order to reduce the amount of data that has to be processed and this way helping the mining techniques have a better performance, we provided some filters in the conversion plug-in that can be seen in the ProM Import screenshot (Figure 41):
Apart from two properties `logFile` and `defaultOriginator` that are used to initialize the log file that needs to be extracted and the default originator (person or system component responsible for execution of an activity), the rest of the properties address the types of events and data stored in the CEI log. To facilitate the mining of specific types of events, we provided filtering options for different types of process entities and components:

- **Process**: `useEventsProcessInstance`
- **Activity**: `useEventsActivity business relevant`, `useEventsActivity structural relevant`
- **Link**: `useEventsControlFlow`
- **Variable**: `useEventsDataModification`
- **Human task**: `useEventsHumanTask`, `useEventsEscalation`
- **Service Component Architecture**: `useEventsSCA`

Additionally, the filtering option `useEventsSystem` allows the user to decide if it is necessary to keep the events or data related to the execution environment, referring to the IT level rather than the business level. This kind of information is available in the common CBE properties, described in previous chapter, Section 5.2.3, Table 1. For the events related to the Human Task component of BPC, we provided also the opportunity to separate the events concerning the escalation of human tasks.

Furthermore, we also decided to separate the activities in two groups, and provide filtering options for each of them: `useEventsActivity business relevant` and `useEventsActivity structural relevant`. The reason for splitting the activities in two groups comes from the way activities are defined and the role they play in WS-BPEL, the process modeling language on which WebSphere is based. WS-BPEL defines two types of activities [10]:

- Basic activities: Receive, Reply, Invoke, Human Task or Staff, Assign, Java Snippet, Throw, Compensate, Wait, Terminate, and Empty.
Structured activities: Sequence, Flow, Scope, While, Switch, Pick.

Many from the activities listed above are used to provide the functionality for routing the activities in a process, to realize the internal control-flow and structure, but they do not represent a business task, an atomic piece of work that needs to be enacted by a person or a machine. Therefore, they are not so valuable from the business perspective. The ones that we considered to be relevant for mining the business perspective are: Invoke, Receive, Reply, Java Snippet (also named Script), Assign and Staff (also named Human Task). These activities will be filtered from the log by selecting the option useEventsActivity business relevant. The rest of the activities are falling under the other option, useEventsActivity structural relevant, which filters their corresponding events, if enabled. For more information about the BPEL activities we refer the reader to the WS-BPEL specification document [10].

Another important utility of this conversion plug-in is that provides a solution to the setback of the event definition discussed in Section 5.2.4, with respect to the variable events. In order to allow the mining techniques to make use of the data handled in a process instance, we need to have data associated with activities. Since the event definition of BPC fails to provide this link, we included data elements as additional attributes for the ATEs corresponding to the other types of process elements and components. Basically, for each ATE except for the variable related ones, we have attached all data elements available in the process instance log up to the moment when that specific event was produced. Obviously, it is not an accurate solution, but it is good enough to allow us to investigate the data and the case perspective. Since data objects can be reused in different activities in a case, it is impossible to determine to which activity a data object belongs to, just considering the time property of the variable event. Therefore, the solution included in the plug-in might not be an accurate one, but it serves its purpose.

Now that we explained how the conversion of the WebSphere logs is realized, we can proceed with the investigation of a BPEL process model that has been designed using WebSphere Integration Developer and then deployed and monitored on the WebSphere Process Server. Section 6.4 will present the investigation of the Case Handling Execution Framework (CHEF) application.

6.3 Analysis of a BPEL process

In this section we present the description of a BPEL process model, the event log and the analysis results. For process mining we considered the three dimensions of process mining: discovery, conformance and extension.

6.3.1 Process model description

In this section we present the description of the process model that was designed using WebSphere Integration Developer (WID) in order to be executed and monitored. We modeled a simple process that encompasses some of the most important functionality offered in WebSphere: combining automated flow with human tasks.

The model represents an order handling process of a fictive company producing and assembling two types of bikes - standard and customized, according to the customer specifications. The company has a special collaboration with the “Extreme Bikers” club, for which it is a partner at international cycling contests and demonstrations. Therefore, members of this club receive a different treatment than the regular customers, their orders having a higher priority than the rest. The process starts with receiving an order and follows with an initial check of the customer profile, to establish the priority of the order. If the customer is a member of the “Extreme Bikers” club, the priority receives a high mark. After this, the verification of the customer credit is performed in parallel with the order processing being passed on to the logistics and manufacturing departments. Here the information from the customer is processed and according to the requirements, the order is released to one of the two manufacturing
groups responsible for producing the two types of bikes. The standard bikes contain the same components and are assembled continuously based on the customers’ orders. The customized bikes are designed to incorporate specific desired features requested by the customer. Once the production of the bikes is finished, a confirmation is sent back to the logistics department. At this point, depending on the verification result of the credit and the priority of the customer order, a decision is made. If the credit verification has a positive result or the order priority is high enough, then the products are packaged and delivered to the customer. Otherwise, the order is cancelled. Finally, the case ends with the archiving of the customer order.

Involving human tasks was a primary objective of the case study; therefore for the execution of the process we have configured security settings. Like in a real working environment, case workers have to authenticate themselves in order to connect to the information system and perform the tasks in their work list. Workers should execute activities according to the predefined authorization roles they have. In this case, several groups of users have been defined for the execution of the process human tasks:

- **creditApprovers**: personnel authorized to perform financial related tasks
- **designers**: responsible for the design of the customized bikes
- **workers**: take care of the assembly procedure
- **operators**: carry out administrative work
- **admins**: manage system administration

Figure 42 depicts the model realized in WID. The process model is build using three types of components, as defined in WebSphere:

- a BPEL business process
- Human tasks
- Business rules

For the human tasks, we considered two ways of modeling them: as a **Staff** activity, inline with other process activities, and using **Invoke** activities, in this way each human task representing a different service component in the assembly of the process. Figure 43 of the process illustrates the assembly view.
Figure 42: BikeShop process model
In Figure 43 we see how the main process is connected with several Service Component Architecture (SCA) components. These components represent human tasks that are invoked asynchronously from the main process, and the business rules group – PriorityRules, also invoked to perform an automated activity. The main advantage of modeling certain activities or parts of the process as distinct service components is the possibility of reuse and sharing of such pieces of functionality in other processes as well, by exposing them as business services.

In order to test this, we actually consider a second implementation of this process model (Figure 44). Here, the human task component CheckCredit is implemented as a stand-alone process in a separate module, which is imported by the main module at run-time as a business service (Import_CreditService). This scenario gives some insights into how processes can interact as services and about the behavior that can be observed if they are monitored.

It is important to note that normal BPEL processes are, generally, not semi-structured. However, WebSphere provides some degree of freedom. For example, dynamic selection and/or combination of business services (i.e. SCA modules), as well as allocation of work to groups of authorized workers instead of a specific user, provide some sort of flexibility. The working behavior of people assigned to a process is also interesting to analyze. Moreover, certain paths in the process may be enabled only if triggered by some external events. These types of flexibility can be investigated by mining the execution logs.

6.3.2 Process mining results

After monitoring the execution of the process described in Figure 42, we obtained a log containing 20 process instances (cases). In the first step we used the ProMimport converter plug-in (introduced in Section 6.2.2) to transform the logs into the MXML format and filter the necessary information for mining business events.

General log observation

Before applying any mining algorithm, we first had a preview of the log, to get an idea of the basic log characteristics. Using the Log Summary (Figure 45) and the Basic Log Statistics (Figure 46) plug-ins of ProM, we got some initial information about the process behavior.

We can see that there 20 process instances together contain 1589 ATEs. In the list of log events we observe the event type and occurrence rate of each workflow model element. For example, in the screenshot Figure 45, we see that the most frequent workflow element in the log is the process variable...
The variable `var_response` had its value changed 124 times in all 20 process instances. Also, we can see the activity `ConfirmOrderProcessing` being scheduled for 61 times and assigned only 21 times. This already raises questions about how tasks are actually managed and what is the relationship between the various events of the same element. Additionally, information about the originators and the frequency of their involvement in executing work items is also presented.

We can also see basic statistics about the duration of activities based on the general log, or based on a specific instance. Without having started mining the process we can already identify some performance indicators by seeing activities that take too much time to be executed, or the biggest variance in the activities duration.

![Figure 45: Log Summary](image)

For the BikeShop process we observe that activity `IncludeSpareParts` stands out from the rest, considering the average duration of execution. This view is already giving a hint for the places in the process that need to be further investigated for performance analysis.

![Figure 46: Basic Log Statistics](image)
Process Discovery
After gathering some knowledge about the elements in the process, the event types and their frequencies, we continue with mining the log to discover the actual process model. It is important to note here that ProM offers the possibility to filter the log at a much finer grain, before applying some mining algorithm, in order to keep only the most relevant information. Undesired event types can be ignored, process instances containing specific event types or originators or that do not end or start with a specific event type can be discarded from the log.

In this case, our goal is to discover the control-flow of the process (i.e. the order in which activities are executed) by applying process mining algorithms. However, the event log, as it is, contains too much information that would hamper the correctness and clarity of the mined model.

We have already mentioned in Section 6.2.1 that several events are logged when some activity or human task is being executed. For example, when the execution of the process reaches a point where some human task needs to be performed (e.g. CheckCredit in Figure 42), then several events are triggered for the specific task (see also Table 9 in Section 6.2.1): assign, schedule, start, and complete. The first two events are triggered automatically by the system when it schedules the human task. The last two events are triggered by a user action, when the user claims the task in order to work on it, and later on, when the user completes performing the task, respectively. However, for system activities (e.g. Assign or Snippet kind of BPEL activities), we observed that only the complete event is being logged, because they are executed automatically. Moreover, the BPEL process (Figure 42 and Figure 43) uses Invoke BPEL activities to implement the human tasks as service components. This Invoke activity adds even more events to the log (e.g. start, complete), related to the same logical business activity (e.g. CheckCredit in Figure 42). As a result, we may have six events for the same business activity (i.e. self-contained partitions of work) in the same process instance.

Obviously, when several persons are working simultaneously on a case, these events may interleave for different tasks. The mining algorithms determine the dependencies between activities based on their ordering in the log. Such a mixture of events can generate a confusing and even an incorrect process model, because only a fraction of the possible interleavings is observed. Therefore, we need to select only a minimum number of events that are able to identify the logical business activity. We have to keep only the events that are necessary to discover the routing of business activities in the process.

In our case study, we have decided to keep only the assign events that relate to human tasks, and complete events for the system activities, except for the Invoke activity type that is used to wrap a human task. After filtering the log to keep only this information, we applied different mining algorithms to discover a process model. By example, using the Multi-phase Mining algorithm in ProM, we obtained an Event-driven Process Chain (EPC) model, displayed in Figure 47 (left side). The discovered model using Heuristic Miner is shown also in Figure 47, on the right, and the model in the Petri net format is displayed in Figure 48. These three models have the same structure. However, they are not the same with the original BPEL model in Figure 42. The difference is that the discovered model shows only 2 activities in the first parallel construct, which is followed by a decision point. The original BPEL model includes the decision point inside the parallel construct, following only the activity ProcessOrder and not CheckCredit (Figure 42). The discovered behavior reflects how the system actually assigns the activities in the process.

Further, we mined the social network, showing the “handover of work” metric in Figure 49. Thus, using distinct mining algorithms on the event log, we were able to discover the actual (scheduled) process and the social network model, in different modeling formalisms. We make the observation that the screenshots are presented for showing the different formats and structures in general, and not for a detailed discussion of the resulted models.

Using a different filtering approach, we have selected only the complete events of system activities, and discarded the specific human task events. In fact, the complete event of a human task corresponds to the complete event of the Invoke activity that wraps it as a service. In this way, we avoid having
duplicate events for the same logical business activity. Figure 50 shows the model generated by the Alpha++ Miner in a Petri net format.

Figure 47: Discovered process using Multi-Phase Miner (left) and Heuristics miner (right)
Figure 48: Discovered model using Alpha++ Miner on filtered log

Figure 49: Social network discovered using the Social Network Miner

Figure 50: Discovered model for filtered log with complete events

Figure 51: Conformance check of discovered models
We mention that both discovered Petri net models in Figure 48 and Figure 50 returned a positive result for the correctness verification in ProM, using the Woflan analysis plug-in. Furthermore, we can observe that different filters may determine distinct models, as they reflect different situations. The first model discovered (Figure 48) shows the order in which the system schedules the activities and human tasks. The second model (Figure 50) indicates the order in which activities and human tasks were in reality completed. We can see that the second model displays much more behavior than the first one, as it reflects the workers’ way of executing tasks. This kind of dynamic behavior or working patterns can be detected in the discovered model and used for process improvement. Additionally, mining can reveal not only the process structure but also the social network of the people involved (see Figure 49).

In Section 6.2.2 we mentioned that we have implemented in our conversion plug-in a filter for the SCA events. Mining the filtered log provides the following model in Heuristic Miner plug-in:

![Figure 52: Discovered SCA model](image)

As we can see, this model gives a different perspective on the execution of the process. It displays how business service components interact with each other. The model indicates on arcs the value of dependency metric between any two connected services, and also the frequency of the interaction. We can see that the SCA_ManualProcessOrder interacts with the main process (SCA_BikeShopProcess) in all 20 cases existing in the log, while SCA_CancelOrder only in 4. Hence, a different level of abstraction can provide useful insights, especially in a service oriented environment. The analysis can focus on different aspects, such as correctness of services choreography.

Moreover, it is important to see what we can learn about the execution of BPEL processes in WebSphere. By filtering the log based on schedule events of system activities, we realized that activities that are modeled as parallel (e.g. invoke activities CheckCredit and ProcessOrder) in the BPEL process (Figure 42) are actually always started (i.e. scheduled) sequentially by the BPEL engine. Figure 53 shows the discovered model. Because of this, patterns considered at modeling time (e.g. BPEL flow) are not present in the discovered model. This may generate a confusing or misleading picture of the process execution, when system activities and human tasks are analyzed together. Therefore, the log needs to be carefully prepared (i.e. filtered) before applying mining algorithms.

![Figure 53: Discovered process from the filtered log with schedule events](image)

Understanding how a system is executing the business processes is very important for making a more appropriate selection of the events that we want to mine.
Next, we introduce the results of the conformance checking analysis.

**Process Conformance**

As we explained at the beginning of the thesis, in Chapter 2, checking process conformance means to compare an a-priori model with the “real” behavior stored in some MXML log. The goal is to identify any discrepancies between the predefined model and the actual executed process. Understanding the context of these deviations and their severity may be used to improve the process.

In our case study we checked the conformance of the models discovered in Figure 48 and Figure 50, based on the filtering approaches that we described. The results are presented in Figure 51. We can see that the first model has a fitness of 1, while the second one has a fitness of 0.9. The fitness metric evaluates whether the observed behavior in the log complies with the control flow specified by the process. The fitness value ranges from 0 to 1. The closer the value is to 1, the higher the compliance is. The first model is able to “parse” all executions, while the second one displays some deviations. These deviations need to be investigated and resolved. In a real situation, however, a prescribed model would be used for conformance checking, and not a discovered one.

For checking the conformance, the LTL Checker plug-in in ProM is also a powerful technique that can verify process properties using the event log. For verification it uses an approach based on Linear Temporal Logic (LTL) [25]. The LTL language allows for more declarative style of modeling, and it is used as a basis for the declarative language DecSerFlow [89], mentioned in Section 3.1.1. Figure 54 shows the answer to the question if activity ProcessOrder is always followed by activity PackageBike (i.e. if all processed orders are actually fulfilled). The result shows that there are 4 cases where the property is not valid. By inspecting one of these cases, we can see that actually the order is canceled, which excludes the fulfillment of the order.

**Process Extension**

Another type of process mining is the performance analysis of a business process. In this situation we can use ProM performance analysis plug-ins to discover any bottlenecks or patterns in the process execution. Figure 55 presents the results of the Performance analysis with Petri nets plug-in. The plug-in evaluates the performance of the process execution, using metrics like throughput time or waiting time, and projects this information on the model. In this way, the inefficient places in the process (e.g. where the waiting time is higher) are highlighted such that they can easily detected.
In our model, we used the Petri net discovered in Figure 50, based on complete events. We see that the waiting time is higher before the completion of CheckCredit activity, which may be explained by a long execution time, for example.

Another important plug-in that takes also data into account, is the Decision Point Analysis. We have already described the decision mining technique in Section 3.3.2.1 of this thesis. Applying this method on the WebSphere log, we obtained the following result:

Figure 56 indicates that for the decision point highlighted in the Petri net (section of the discovered model in Figure 50), there are two variables that influence the outcome: var_creditValid and var_priority. The discovered rule in this case assumes that cancelSnippet occurs only if...
var_creditsValid is false and the variable var_priority has the value 10. Otherwise, packageSnippet is executed. This rule complies with the description we gave for the BPEL model in Section 6.3.1.

In conclusion, the insights obtained by using these plug-ins may help improve the original model, by making it more efficient or correct possible errors in the process business rules. The next section shows how the ESA plug-in can be applied on WebSphere logs and what knowledge can be derived from it.

### 6.3.3 Knowledge discovery with ESA

In this section we discuss how to discover more knowledge from the execution log, by using the ESA method we have developed in ProM. In particular, we want to highlight the benefits of using the ESA method to visualize the states of various process elements.

Using the WebSphere log containing events for activities, human tasks and data elements, we have applied distinct settings in the ESA, for different perspectives. We have used the following events to define process elements states:

- **start** – to define *Running* (for task, instance) and *Working* (for originator) states
- **complete** – to define *Completed* (for task, instance) state
- **assign** – to define *Assigned* (for task, originator) state
- **update** – to define *Defined* (for data) state

Figure 57 shows the task perspective with the actual time option:

The task perspective shows how much time the human tasks remain in the *Assigned* state (light color), waiting for some employee to start executing them. Also, the *Running* state explicitly displays the time intervals where the task is actually performed (i.e. from the moment the employee selects the task from his working list, until he completes it). For system activities (e.g. setPrioritySnippet) that have only *complete* events registered in the log, only the *Completed* state (i.e. marked by small rectangles) is visible in the picture.

If we look at the instance perspective, using the logical relative time option, we can discover even more information about the process. This view allows us to compare different cases, and detect exceptional behavior or frequent patterns in their execution. Figure 58 clearly shows that certain instances have a long execution time (i.e. indicated by *Running* state), distinguishing from the rest. Also, it is easy to identify the time intervals where no work is carried out, even though tasks are assigned to case workers, just by looking at the *Assigned* state.
Since the WebSphere log contains the variable event *update*, we can also use the data perspective in ESA, to investigate how data is manipulated in the process. Although the data model defined in Section 4.2.3 for our visualization method includes more data specific events, we might find interesting behavior just by looking at data *Defined* states. Figure 59 shows only the *Defined* state of the data elements in the BPEL process:

![Instance perspective with relative time option](image1)

**Figure 58: Instance perspective with relative time option**

We have already discussed in Section 5.2.4 of the previous chapter that WebSphere BPC logs only variable *update* events. However, the image above conveys clear information about the moments of the updates and the duration of the *Defined* state. It enables an easy recognition of data access patterns and their frequencies. Yet, due to insufficient information available in the event log, no direct connection can be made with the task perspective.

Our BPEL process contains also information about originators (i.e. workers involved in the process execution). The ESA plug-in provides useful information about the performance of case workers, as Figure 60 illustrates:

![Data perspective](image2)

**Figure 59: Data perspective**
The Running states show how much time the originators are actually spending on performing their work. The Unassigned state (in dark color), which is very prominent in Figure 60, gives important information about the management of process human resources. In this case, it clearly indicates that resources are scarcely used (i.e. assigned or working on some task). It is important to remind here that visualization of the originator states depends on the availability of task events. For example, the system appears not to be assigned nor working, which in reality is not the case. In fact, the task events to which the system is related to, are: assigned, complete and update. The start events (which determine the Running state) correspond, in this case, to human tasks only. Therefore, the Running state cannot be displayed. Moreover, the system does not appear as an assigned resource for some task, hence the Assigned state can not be displayed either.

ESA provides more information than other mining techniques, by making use of the plug-in user’s knowledge. The interactive data exploration is a major benefit of the visualization method. Furthermore, the states of various process elements convey more detailed information than a common sequence diagram. The time intervals capture concrete meanings in ESA, enabling a better analysis of the process and the resources used for its execution. Although WebSphere supports only one data specific event, we have seen that the data perspective can bring valuable insights, even for structured processes like BPEL. Moreover, information revealed by the other perspectives can help provide more suitable support for processes with human involvement. Obviously, the more dynamic the behavior of a business process is, the more it can benefit from the ESA’s functionality. In conclusion, we demonstrated that ESA can be applied to structured processes like BPEL to reveal important information, although it was initially intended for data analysis. However, the usefulness of the data analysis becomes stronger in the context of data-intensive semi-structured processes.

This case study brings forward the challenges in mining WebSphere logs, and shows the results of some mining algorithms and methods that we have applied. We have seen that the original log contains rich information about process, activities, data and people involved in the execution of a process. This allows most of the mining plug-ins in ProM to be used for process analysis.
After applying process mining techniques on WebSphere logs, generated by a common BPEL process, we continue in the next section with a discussion about more flexible processes that can be deployed in WebSphere.

### 6.4 Case Handling Execution Framework (CHEF)

The previous section presented how we can use the mining tool, ProM, to analyse BPEL processes, and what measures we need to take in order to obtain meaningful results. In this section we introduce the Case Handling Execution Framework (CHEF), implemented in the context of IBM WebSphere. The goal of this discussion is to point out what flexibility CHEF brings to the execution of normal BPEL processes, and how this flexibility can be better controlled by using appropriate monitoring and analysis methods.

CHEF is a prototype of a case handling application developed during the IBM Extreme Blue program 2006: *Case handling for Knowledge Workers*. The author of this thesis was one of the 4 students working in this project, which resulted in a fully functional prototype. It follows the case handling paradigm presented in Chapter 3, Section 3.2.1 (page 12). The prototype serves as a proof of concept, and shows how more flexibility can be added to execution of the BPEL processes, using the IBM WebSphere technology at hand. Figure 56 shows an overview of the IBM products that have been used for implementing CHEF:

![Figure 56: IBM technology used in CHEF implementation](image)

By using proven technology, CHEF inherits the benefits of a unified user interface combined with collaborative features and a reliable and secure deployment environment for business processes.

However, CHEF enhances the operational flexibility in handling processes where information is critical. CHEF extends the classical workflow features with new concepts such as redoing and skipping activities in a case, and runtime modification of the process instances (e.g. adding activities on-the-fly). This functionality provides implicit flexibility, by offering more freedom to the case workers that handle the case. The process behavior becomes more variable, as the execution of the activities in the case is less enforced by the prescribed model. Furthermore, the process becomes more data-driven rather than control-driven. Data plays a central role in the execution of cases, as it allows the enabling or auto-completion of activities. The process evolves as a consequence of information becoming available in the case.

This poses interesting questions with respect to the monitoring and control of such processes:

- What kind of information should be monitored?
- Are the CEI logging facilities offering enough support for monitoring such a behavior?
- What actions should be properly monitored in CHEF to be able to retrieve the business logic when mining the event logs?
- What are the problematic issues that can hinder the quality of the information available in the log?
In Section 6.2 of this chapter we presented the logging format and the information that is stored by the deployment server for business processes. Since CHEF is built on top of WebSphere and manages BPEL processes, information about cases (i.e. process instances), activities and process data elements may be logged as for normal BPEL processes. However, important information that addresses directly the case handling principles is not stored in the event log by the default monitoring setting. As we explained in the case handling approach (page 12), data is the key-driver in the execution of a case. Namely, data elements states determine the enabling or auto-completion of case activities. Therefore, actions related to data manipulations should be monitored and appropriately logged. Currently, WebSphere Business Process Choreographer (BPC) supports only one data event: *variable update* (Chapter 5, Section 5.2.3, page 65). The events that we proposed in the data model for the data visualization plug-in ESA (Chapter 4, Section 4.2.3, page 37) are necessary to be able to track such low-level behavior.

CHEF provides also security features in handling of cases, inherited from WebSphere. Only authenticated and authorized workers are allowed to open cases and work, read or administrate activities. Information about the people involved in the execution of some case is available in activity events. Hence, process mining techniques such as Organizational Miner or Social Network Analysis can provide valuable insights into how people are performing their work. However, only knowledge about the execution roles is provided. Following the case handling principles, CHEF provides also *skip* and *redo* roles for case workers. By logging data specific events, we would be able to detect also the *redo* authorization role, due to the close interrelation between data *rollback* event and the *redo* action. In order to have this information in the event log, CHEF needs to be enhanced with customized monitoring features. Currently, BPC is not providing this kind of data in the event log. Furthermore, the customized logging should store in the data event information about the activity where the data element was accessed by the user. This is necessary to overcome the limitation of BPC logging that lacks the link between data and activities.

The user interface of CHEF is built using the WebSphere Portal and Lotus Workplace Forms, and integrates also collaborative services. However, the logging facilities of these products are independent of process monitoring offered by BPC. Consider, for example, that workers are exchanging documents or forms using the collaborative services. These actions are not logged by the BPC, even though they might contain relevant information for explaining the outcome of a certain decision or the context of a change or modification in the case. Moreover, forms for handling cases usually provide general data about the case. This data might not be mandatory for completing the case, but it can provide useful insights. It is important to note that these data elements should be explicitly declared in the process model as well, to be logged by BPC. Otherwise, the content and the modifications on them will not be available in the log. As in the previous case, it may be useful to consider a customized logging of all these actions to assure that all relevant data exists in the event log. This customized logging may be a good temporary solution until BPC is extended to include all necessary data events.

As we can see, a lot of importance is placed on data elements in the process. Therefore, they should be monitored in a thorough manner in order to provide sufficient information in the event log that can be used further on for process mining. The processes executed in CHEF are semi-structured in nature, allowing more freedom and flexibility to the case workers than the normal BPEL processes. Hence, they display a much more dynamic behavior. However, this behavior may go undetected if the relevant actions are not properly monitored. In analyzing CHEF cases, the ESA plug-in would be one of the most useful process mining techniques. The dynamic behavior of the cases, as well as the interrelations between data, activity and originator perspective would be easier to discover and understand. Subtle dependencies among these perspectives may become clear to the persons performing the process analysis.
6.5 Conclusions

Workflows enabled by web-services technology pose many challenges in managing and understanding the services composition, when logs from a variety of servers must be collated and interpreted. Identifying problems or exceptions in both the business layer (e.g. workflow choreography, incorrect implementation of business rules, etc) and the IT layer (e.g. semantic and syntax errors, transactional bottlenecks, etc) has a tremendous impact on process improvement. Moreover, operational flexibility currently represents one of the most required features of workflow technology. As a consequence, many approaches have been considered to meet these flexibility requirements.

Nowadays, IBM WebSphere is one of the most well known commercial technologies for business process management. Built on SOA, WebSphere strives to provide support for both highly automated processes and processes that involve human activities. Therefore, development efforts concentrate on adding more flexibility to address the specific needs of administrative processes. Moreover, in order to assist for better control, powerful monitoring and logging infrastructure has been provided in the process engine. In the previous chapter we described the CEI infrastructure that allows an extensive logging of the business processes. However, the IBM monitoring and analysis tools offer limited functionality for process analysis. Therefore, we stressed the idea that the quality of the WebSphere log can be further exploited by applying process mining techniques.

In this chapter we studied the CEI logging format and we described the implementation of the conversion plug-in in ProMimport, to be able to mine the WebSphere log. Hence, our implementation opens new opportunities for the analysis of BPEL processes. The case study presented in Section 6.3 explains what challenges in mining the WebSphere logs we found, and how we can apply various mining methods in ProM. We also discussed about the case handling functionality added to WebSphere by CHEF, and what measures should be taken to be able to provide sufficient information in the event log. The increased degree of flexibility can be properly analyzed only if detailed information and proper attention to data elements is given.

The case study demonstrates that ProM can provide a more in-depth analysis of the business processes deployed in WebSphere, taking into account various perspectives (e.g. control-flow, data, and organization). Moreover, we have seen that the data visualization plug-in ESA can be very useful for analyzing semi-structured business processes, such as those supported by CHEF. Therefore, IBM WebSphere can benefit from considering ProM’s functionality as a valuable extension to its analysis tools.

Furthermore, we acquired useful knowledge about how a successful commercial information system is designed to support business processes. Our research revealed that this kind of powerful technologies should be better exploited and used to discover information at a finer level. Only a comprehensive analysis of business processes can enable appropriate support and cater for the needs of more flexible processes.
7 Conclusion

The research presented in this master thesis primarily addressed the needs of data-centric, semi-structured business processes. In the introduction, we stated two objectives for this thesis, which are:

1. Enhance the support for semi-structured business processes, by providing an innovative data visualization method to improve the data-centric analysis of business processes.
2. Analyze and study the applicability of process mining in a real-life information system.

Before presenting the main results, we provided the reader with an overview of process mining. We outlined the types of business processes and types of analysis that process mining can perform on execution logs. We also introduced the mining framework ProM developed at the TU/e which is used to implement process mining methods in a standard environment.

To stress our motivation with respect to the stated objectives, Chapter 3 showed that administrative processes need more operational flexibility and that data-oriented approaches are able to improve the capabilities of processes to deal with unforeseen exceptions and change. We emphasized the case handling paradigm as the most representative approach that uses data as a key-driver to provide better support for semi-structured business processes. However, the induced flexibility makes it more important to closely monitor processes such that undesired behavior can be detected and corrected efficiently. The more unstructured the process is, the more important it is to know how the process actually behaves. Although process mining is, currently, the most advanced practice for business process analysis, few process mining techniques address the data perspective. Therefore, in this chapter, we claimed that data should be treated as a 1st class citizen in the process mining practice.

The data visualization method described in Chapter 4 provides an innovative way of analyzing the states of workflow elements. Implemented as the Element State Analysis (ESA) plug-in in the process mining framework ProM, it assumes that data elements are as important as activities in a process, with their own lifecycle. The method displays the data flow of a process execution at a very refined level, where individual data elements states can be identified. Clearly, data access patterns and their frequencies can be easily detected, and their implications on the process execution can be better understood. Initially developed to support the data perspective, ESA proved to be useful for other perspectives as well. By comparing the various perspectives, new dependencies or relationships between them can be established. Moreover, patterns in the data view may reveal the cause of less efficient tasks or give more contextual information about changes or deviations in the process, especially when flexible systems allow workers more freedom in performing their work. With this generic functionality, a more comprehensive analysis of business processes should be enabled.

Additionally, we assessed the performance of the ESA plug-in considering different test settings. We considered the performance in terms of time and memory consumption. For this, we simulated a case handling model and generated synthetic logs that allowed us to test the implementation and also to demonstrate the importance of the data perspective. The results of our measurements provide an indication on how the plug-in performs in a real working environment. We believe that ESA exhibits a good behavior in dealing with logs of various sizes and complexities, and the test case of the industry log shows it. Therefore, we strongly believe that the Element State Analysis plug-in will add value to the collection of mining techniques available in ProM framework. Consequently, the Element State
Analysis plug-in increases the support for semi-structured processes where data has a central role. This accomplishes the primary objective of this thesis.

In the second part of this thesis we applied some of the process mining techniques on a real-life information system. We learned that modern information systems such as IBM WebSphere are very complex, and, thus, it is essential to have domain (system) knowledge in order to apply process mining to analyze the supported processes. This knowledge is necessary for preprocessing the event log and for carefully selecting the relevant data for mining. Obviously, the quality of the analysis depends on the quality of the input information. In order to analyze processes deployed in WebSphere, we developed a conversion tool to translate the original event logs into the mining format used in ProM. We have seen that the IBM logging infrastructure (CEI) provides strong functionality to store extensive information about process execution. Therefore, it has the capabilities to sustain the logging of both structured (e.g. BPEL) and semi-structured processes (e.g. CHEF cases), and provides rich information for applying most of the process mining techniques. However, the logging support for data elements could be extended to benefit even more from analysis tools that focus also on data, such as ESA. The case study described in Chapter 6 demonstrates that process mining can be successfully applied on WebSphere logs and extract more knowledge than common monitoring and analysis tools. Furthermore, we have shown that ESA reveals interesting insights into the process behavior when used to analyze a BPEL process. As a consequence, we argue that ProM’s functionality would be a great added value to the process analysis tools of real-life information systems. With this, our secondary objective is also achieved.

Overall, we have seen that ESA is a promising analysis tool that increases the support for data centric semi-structured processes. As further work, we intend to apply it on real-life logs and improve its performance such that it can be efficiently used in industry settings. Moreover, IBM WebSphere can extend the capabilities of its analysis tools with functionality available in ProM, and thus strengthen the link from analysis to design in process management lifecycle support.
REFERENCES


http://www.redbooks.ibm.com/abstracts/SG247148.html

http://dev.eclipse.org/viewcvs/indextools.cgi/hyades-home/docs/components/common_base_event/cbe101spec/CommonBaseEvent_SituationData_V1.0.1.pdf

[5] BPEL4People : WS-BPEL 2.0 Extension for People  


[7] BPELJ: BPEL for Java technology  

[8] IBM Web Services Navigator  


http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpe1-v2.0-CS01.html


http://fp.tm.tue.nl/beta/publications/working%20papers/Beta_wp189.pdf


[70] I. Vanderfeesten, H. A. Reijers, W.M.P. van der Aalst: *Product Based Workflow Design with Case Handling Systems*


[77] F. F. Soulié: *Data Mining in the real world. What do we need and what do we have?* Workshop on Data Mining for Business Applications, held in conjunction with the KDD conference, August 20, 2006.


[86] ProM framework, Fuzzy Miner plug-in:  
http://ga1717.tm.tue.nl/user/christian/research/fuzzyminer


APPENDIX A

Simulation of a case handling model realized in CPN Tools

Figure 1 shows the top view of the case handling model extended from [71], illustrating the interaction of the case workers with the system (e.g. open form, close form). The “Clock” place is used to add time to the Petri net model, such that every user action can be logged with a different timestamp. Also, a delay is added for the resources used, before they become available, to model a more realistic situation where resources don’t have a 100% busy rate. This will reflect the “Unassigned” state of the originator in the ESA plug-in.

Figure A1: Top View - user interaction

The second screenshot depicts the task lifecycle view, with all task transitions being logged. The redo transition logs also the data event rollback, for all data elements that relate to the specific task which is being redone. An extra condition has been added to this transition, such that a task may be redone, only if all related data elements are not used in some other task, at the same time. This prevents concurrent access to the same data.
Figure A2: Task View
APPENDIX B

The measurements we realized for assessing the performance of the ESA plug-in are presented in the following tables, for each test setting, separately. We have used the t-Statistics to calculate the confidence intervals.

a) One case, variable number of events, highly disordered log

<table>
<thead>
<tr>
<th>#events</th>
<th>#events</th>
<th>display time (ms)</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>500</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>2000</td>
<td>60</td>
<td>60</td>
<td>50</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>4000</td>
<td>130</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>140</td>
</tr>
<tr>
<td>8000</td>
<td>220</td>
<td>200</td>
<td>221</td>
<td>270</td>
<td>210</td>
</tr>
<tr>
<td>16000</td>
<td>501</td>
<td>511</td>
<td>501</td>
<td>491</td>
<td>511</td>
</tr>
</tbody>
</table>

Table B1: Display time (ms) for task perspective in ESA

<table>
<thead>
<tr>
<th>#events</th>
<th>open ProM</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>9.81</td>
<td>9.77</td>
<td>9.24</td>
<td>9.22</td>
</tr>
<tr>
<td>500</td>
<td>9.29</td>
<td>9.53</td>
<td>9.26</td>
<td>9.35</td>
</tr>
<tr>
<td>1000</td>
<td>9.76</td>
<td>9.22</td>
<td>9.64</td>
<td>9.68</td>
</tr>
<tr>
<td>2000</td>
<td>9.57</td>
<td>9.22</td>
<td>9.64</td>
<td>9.76</td>
</tr>
<tr>
<td>4000</td>
<td>9.8</td>
<td>9.29</td>
<td>9.22</td>
<td>9.69</td>
</tr>
<tr>
<td>8000</td>
<td>9.76</td>
<td>9.68</td>
<td>9.76</td>
<td>9.37</td>
</tr>
<tr>
<td>16000</td>
<td>9.68</td>
<td>9.45</td>
<td>9.53</td>
<td>9.62</td>
</tr>
</tbody>
</table>

Table B2: Memory consumption (MB) for opening ProM

<table>
<thead>
<tr>
<th>#events</th>
<th>load log file</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>11.2</td>
<td>11</td>
<td>11.09</td>
<td>11.67</td>
</tr>
<tr>
<td>500</td>
<td>11.12</td>
<td>11.05</td>
<td>11.08</td>
<td>11.66</td>
</tr>
<tr>
<td>1000</td>
<td>11.59</td>
<td>11.47</td>
<td>11.51</td>
<td>11.47</td>
</tr>
<tr>
<td>2000</td>
<td>12.02</td>
<td>12.21</td>
<td>12.02</td>
<td>12.1</td>
</tr>
<tr>
<td>4000</td>
<td>12.37</td>
<td>12.39</td>
<td>12.68</td>
<td>12.9</td>
</tr>
<tr>
<td>8000</td>
<td>16.14</td>
<td>15.15</td>
<td>15.39</td>
<td>15.46</td>
</tr>
<tr>
<td>16000</td>
<td>15.93</td>
<td>15.7</td>
<td>15.62</td>
<td>15.83</td>
</tr>
</tbody>
</table>

Table B3: Memory consumption (MB) for loading the log

<table>
<thead>
<tr>
<th>#events</th>
<th>open ESA</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>12.33</td>
<td>12</td>
<td>11.96</td>
<td>11.9</td>
</tr>
<tr>
<td>500</td>
<td>12.68</td>
<td>12.02</td>
<td>11.82</td>
<td>12.52</td>
</tr>
<tr>
<td>1000</td>
<td>12.91</td>
<td>12.91</td>
<td>12.87</td>
<td>12.87</td>
</tr>
<tr>
<td>2000</td>
<td>14.47</td>
<td>13.42</td>
<td>14.31</td>
<td>14.31</td>
</tr>
<tr>
<td>4000</td>
<td>15.91</td>
<td>15.71</td>
<td>15.48</td>
<td>15.54</td>
</tr>
<tr>
<td>8000</td>
<td>19</td>
<td>18.2</td>
<td>19.6</td>
<td>17.73</td>
</tr>
<tr>
<td>16000</td>
<td>24.68</td>
<td>23.75</td>
<td>24.21</td>
<td>24.31</td>
</tr>
</tbody>
</table>

Table B4: Memory consumption (MB) for opening ESA

b) 2000 events, variable number of cases
c) Variable number of cases, each with 20 events

<table>
<thead>
<tr>
<th>#cases</th>
<th>display time (ms)</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60 60 50 50 70</td>
<td>58</td>
<td>8.3666</td>
<td>7.333513721</td>
</tr>
<tr>
<td>2</td>
<td>90 100 110 100 100</td>
<td>100</td>
<td>7.071068</td>
<td>6.197950323</td>
</tr>
<tr>
<td>4</td>
<td>110 110 100 110 100</td>
<td>106</td>
<td>5.477226</td>
<td>4.800911676</td>
</tr>
<tr>
<td>8</td>
<td>130 120 130 110 110</td>
<td>120</td>
<td>10</td>
<td>8.765225406</td>
</tr>
<tr>
<td>16</td>
<td>120 150 130 130 140</td>
<td>134</td>
<td>11.40175</td>
<td>9.993894603</td>
</tr>
</tbody>
</table>

Table B9: Display time for task perspective in ESA
 Industry log with 154966 events and 24 cases

Table B13 shows the test results for displaying the task perspective in ESA, with respect to time.

<table>
<thead>
<tr>
<th>#cases</th>
<th>open ProM</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.39</td>
<td>9.35</td>
<td>9.32</td>
<td>9.69</td>
</tr>
<tr>
<td>2</td>
<td>9.28</td>
<td>9.66</td>
<td>9.69</td>
<td>9.32</td>
</tr>
<tr>
<td>4</td>
<td>9.32</td>
<td>9.62</td>
<td>9.62</td>
<td>9.69</td>
</tr>
<tr>
<td>8</td>
<td>9.32</td>
<td>9.66</td>
<td>9.69</td>
<td>9.62</td>
</tr>
<tr>
<td>16</td>
<td>9.66</td>
<td>9.69</td>
<td>9.77</td>
<td>9.73</td>
</tr>
<tr>
<td>32</td>
<td>9.73</td>
<td>9.66</td>
<td>9.66</td>
<td>9.58</td>
</tr>
<tr>
<td>64</td>
<td>9.69</td>
<td>9.28</td>
<td>9.24</td>
<td>9.39</td>
</tr>
</tbody>
</table>

Table B10: Memory consumption (MB) for opening ProM

<table>
<thead>
<tr>
<th>#cases</th>
<th>load log file</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.13</td>
<td>11.69</td>
<td>11.81</td>
<td>10.98</td>
</tr>
<tr>
<td>2</td>
<td>12.11</td>
<td>11.77</td>
<td>11.84</td>
<td>11.73</td>
</tr>
<tr>
<td>4</td>
<td>11.2</td>
<td>11.54</td>
<td>11.84</td>
<td>10.98</td>
</tr>
<tr>
<td>8</td>
<td>11.39</td>
<td>11.13</td>
<td>11.09</td>
<td>11.13</td>
</tr>
<tr>
<td>16</td>
<td>11.62</td>
<td>11.69</td>
<td>11.35</td>
<td>11.77</td>
</tr>
<tr>
<td>32</td>
<td>11.35</td>
<td>12</td>
<td>11.96</td>
<td>11.88</td>
</tr>
<tr>
<td>64</td>
<td>12.18</td>
<td>11.62</td>
<td>12.03</td>
<td>12.11</td>
</tr>
</tbody>
</table>

Table B11: Memory consumption (MB) for loading the log

<table>
<thead>
<tr>
<th>#cases</th>
<th>open ESA</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.53</td>
<td>12.22</td>
<td>12.18</td>
<td>12.22</td>
</tr>
<tr>
<td>2</td>
<td>12.3</td>
<td>12.15</td>
<td>12.37</td>
<td>12.15</td>
</tr>
<tr>
<td>4</td>
<td>11.66</td>
<td>11.81</td>
<td>12.33</td>
<td>12.26</td>
</tr>
<tr>
<td>8</td>
<td>12.22</td>
<td>11.84</td>
<td>11.84</td>
<td>11.66</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>12.83</td>
<td>12.41</td>
<td>11.81</td>
</tr>
<tr>
<td>32</td>
<td>12.79</td>
<td>12.94</td>
<td>12.94</td>
<td>12.83</td>
</tr>
<tr>
<td>64</td>
<td>13.13</td>
<td>13.05</td>
<td>12.98</td>
<td>12.94</td>
</tr>
</tbody>
</table>

Table B12: Memory consumption (MB) for opening ESA

e) Industry log with 154966 events and 24 cases

Table B13 shows the test results for displaying the task perspective in ESA, with respect to time.

<table>
<thead>
<tr>
<th>display time (ms)</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>128697</td>
<td>133711.3</td>
<td>3361.938</td>
<td>2690.061151</td>
</tr>
<tr>
<td>128835</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130186</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130560</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>133023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>134994</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135045</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135261</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136015</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136252</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>137129</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>138539</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table B13: Display time for task perspective in ESA
Since the display time was easier to measure than the memory consumption, we took more samples (i.e. 12 instead of 5) to calculate the mean and the confidence interval. The second table displays the measurements taken for the memory consumption, with respect to each of the three operations: open ProM, load log, and open ESA:

<table>
<thead>
<tr>
<th>operation</th>
<th>memory consumption (MB)</th>
<th>mean</th>
<th>std dev</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>open ProM</td>
<td>9.32 9.81 9.73 9.43</td>
<td>10.18</td>
<td>9.69</td>
<td>0.34 0.30</td>
</tr>
<tr>
<td>load log</td>
<td>13.33 12.83 12.45 13.20</td>
<td>12.73</td>
<td>12.91</td>
<td>0.36 0.31</td>
</tr>
<tr>
<td>open ESA</td>
<td>99.52 95.47 99.62 95.84</td>
<td>98.49</td>
<td>97.79</td>
<td>2.00 1.75</td>
</tr>
</tbody>
</table>

Table B14: Memory consumption (MB)
APPENDIX C

CEI architecture
In order to understand how events become ready for consumption, it is necessary to describe how CEI infrastructure works. The figure below depicts the main building blocks of CEI and the interactions with external applications [1, 2].

![CEI Architecture](image)

**Figure C1: CEI architecture overview**

The CEI components in this architecture are briefly described in the following table:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Source</td>
<td>Event Sources are applications or components that submit event creation requests through the Event Emitter component. Examples of such event sources are log file adapters, monitors for workflow containers, applications explicitly designed to emit CEI events.</td>
</tr>
<tr>
<td>Event Emitter</td>
<td>The Event Emitter component is a library that allows event sources to submit event creation requests to CEI.</td>
</tr>
<tr>
<td>Event Bus</td>
<td>The Event Bus processes event creation requests from Event Emitters. It ensures that events are routed to the Event Data Store component for persistence (when configured to do so), and to the event distribution component for subsequent publication to consumers. The Event Bus is implemented as a set of services on top of WebSphere’s embedded platform messaging capabilities.</td>
</tr>
<tr>
<td>Event Distribution</td>
<td>The Event Distribution component delivers events to event consumers distributed throughout the network. The distribution component also</td>
</tr>
</tbody>
</table>
informs event consumers about changes to events for which they are interested, including event purges, updates and deletions.

| Event Access | Event Consumers interact with the Event Access component whenever they need to query event data from the Event Database or effect changes on the events already persisted into the Event Database. The component is also responsible to coordinate the interaction between the Event Data Store and the Event Distribution components. The event access services provide an architected, pluggable interface between the bus and event data store persistence mechanisms. |
| Event Data Store | The Event Data Store component implements the event data store plug-in model supported by the Event Access component. Its sole responsibility in the system is to adapt requests from the Event Access component to an actual persistent event repository. |
| Event Consumer | Event Consumers are applications that subscribe to the Event Bus in order to receive event notifications or that use the CEI APIs to query or update event data persisted in the event database. |
| Event Catalog | The Event Catalog is a repository of event metadata in the system. The rationale behind the Event Catalog component is that the CBE model specifies the syntax and semantics for all event properties in the base event, but the base model cannot specify the specific syntax or semantics for application-specific data in extended data elements or context data elements (both event fields in the CBE specification). The Event Catalog allows the definition of these semantics. |

Table C1: CEI components

To emit events in the context of a business process, CEI methods can be also used in a Java snippet activity or implemented as a service which is then called by an invoke activity in the process. The advantage of using the CEI API directly is that it provides full control over the data to be emitted. On the other hand, the drawback is that the business process needs to be extended with these Java snippets to emit the relevant events.

It is important to note that the CEI is allowing for definition of specific event types in the Event Catalog, extending in this way the CBE model. This will provide a practical way of saving the necessary business information in the process, and customizing the events structure according to the needs. Also, the possibility to consume the events as soon as they are produced by subscribing to some queue or topic is provided along with the possibility to query historical data stored in the event data store using the Event Distribution and the Event Access component, respectively.
APPENDIX D

The process relevant events (e.g. activity, human task and variable events) sent by BPC contain more information than it is necessary to map on the MXML format. Thus, additional CBE attributes are stored in the MXML log to provide more information:

<table>
<thead>
<tr>
<th>MXML additional attributes</th>
<th>Corresponding CBE element (type: property)</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id</td>
<td><code>cbe: globalInstanceId</code></td>
</tr>
<tr>
<td>extension_name</td>
<td><code>cbe: extensionName</code></td>
</tr>
<tr>
<td>parent_process_instance_id</td>
<td><code>cbe/contextDataElement: ECSParentID</code></td>
</tr>
<tr>
<td></td>
<td>(only for process instance events)</td>
</tr>
<tr>
<td>process_template_valid_from</td>
<td><code>cbe/extendedDataElement: processTemplateValidFrom</code></td>
</tr>
<tr>
<td></td>
<td>(only for process instance events)</td>
</tr>
<tr>
<td>process_template_name</td>
<td><code>cbe/contextDataElement: processTemplateName</code></td>
</tr>
<tr>
<td></td>
<td>(only for process instance events)</td>
</tr>
<tr>
<td>process_instance_execution_state</td>
<td><code>cbe/extendedDataElement: processInstanceExecutionState</code></td>
</tr>
<tr>
<td></td>
<td>(only for process instance events)</td>
</tr>
<tr>
<td>process_failed_exception</td>
<td><code>cbe/extendedDataElement: processFailedException</code></td>
</tr>
<tr>
<td>activity_template_name</td>
<td><code>cbe/extendedDataElement: activityTemplateName</code></td>
</tr>
<tr>
<td></td>
<td>(only for activity events)</td>
</tr>
<tr>
<td>activity_instance_id</td>
<td><code>cbe/extendedDataElement: activityInstanceID</code></td>
</tr>
<tr>
<td></td>
<td>(only for activity events)</td>
</tr>
<tr>
<td>claim_initiated_by_principal</td>
<td><code>cbe/extendedDataElement: Principal</code></td>
</tr>
<tr>
<td></td>
<td>(only for activity claimed events)</td>
</tr>
<tr>
<td>activity_failed_exception</td>
<td><code>cbe/extendedDataElement: activityFailedException</code></td>
</tr>
<tr>
<td>element_name</td>
<td><code>cbe/extendedDataElement: elementName</code></td>
</tr>
<tr>
<td></td>
<td>(only for link and variable events)</td>
</tr>
<tr>
<td>variable_data</td>
<td><code>cbe/extendedDataElement: variableData</code></td>
</tr>
<tr>
<td></td>
<td>(only for variable events)</td>
</tr>
</tbody>
</table>

Table D1: Additional MXML attributes corresponding to CBE properties

Mapping CBE to MXML

The following two XML fragments show how an event is transformed from the initial CBE format into the MXML format:
Resulted ATE in MXML format:

```xml
<CommonBaseEvent creationTime="2007-06-12T07:46:20.041Z" extensionName="BPC.HTM.TASK.INTERACT" globalInstanceId="CE11DC18B90245AF90DE79801E9F49DFE1" sequenceNumber="57" severity="10" version="1.0.1">
  <contextDataElements name="ECSCurrentID" type="ECSID">
    <contextValue>_TKI:a01b0113.1ee58920.e0b8647c.e57f01d8</contextValue>
  </contextDataElements>
  <contextDataElements name="ECSParentID" type="ECSID">
    <contextValue>_PI:90030113.1ee4de6e.e0b8647c.e57f0166</contextValue>
  </contextDataElements>
  <extendedDataElements name="EventNature" type="string">
    <values>ASSIGNED</values>
  </extendedDataElements>
  <extendedDataElements name="EventCode" type="string">
    <values>51006</values>
  </extendedDataElements>
  <extendedDataElements name="TaskTemplateName" type="string">
    <values>http://BikeShopModule/IncludeSpareParts</values>
  </extendedDataElements>
  <extendedDataElements name="Username" type="string">
    <values>assembler1</values>
  </extendedDataElements>
</CommonBaseEvent>

<AuditTrailEntry>
  <Data>
    <Attribute name="CBE_version">1.0.1</Attribute>
    <Attribute name="Component” type>http://www.ibm.com/xmlns/prod/websphere/scdl/human-task</Attribute>
    <Attribute name="CurrentTaskInstanceId">_TKI:a01b0113.1ee58920.e0b8647c.e57f01d8</Attribute>
    <Attribute name="EventCode">51006</Attribute>
    <Attribute name="EventCodeHumanReadable">task claimed</Attribute>
    <Attribute name="EventId">CE11DC18B90245AF90DE79801E9F49DFE1</Attribute>
    <Attribute name="ExecutionEnvironment">Windows 2003[x86]#5.2</Attribute>
    <Attribute name="ExtensionName">BPC.HTM.TASK.INTERACT</Attribute>
    <Attribute name="JavaThreadId">WebContainer : 2</Attribute>
    <Attribute name="Location">winsphere.campus.tue.nl</Attribute>
    <Attribute name="ParentProcessInstanceId">com.ibm.websphere.CorrelationSphere.implicit</Attribute>
    <Attribute name="ServerIdentifier">widCell\widNode\server1</Attribute>
    <Attribute name="Severity">10</Attribute>
    <Attribute name="SituationCategory">ReportSituation</Attribute>
    <Attribute name="SourceComponent">WPS#Platform 6.0 ..........</Attribute>
    <Attribute name="SourceComponentId">WPS#Platform 6.0 ..........</Attribute>
    <Attribute name="TaskClaimedByUsername">assembler1</Attribute>
    <Attribute name="TaskTemplateName">http://BikeShopModule/IncludeSpareParts</Attribute>
  </Data>
</AuditTrailEntry>
```