camunda
The Business Process Company

Stuttgart University of applied science
in cooperation with camunda GmbH

Master Thesis

Building an open source Business Process Simulation tool with JBoss jBPM

submitted by:	Bernd Rücker
	bernd.ruecker@camunda.com
1st Supervisor:	Prof. Dr. Gerhard Wanner
2nd Supervisor:	Prof. Dr. Jörg Homberger
date of submission:	1st of February 2008

Abstract

Business Process Management (BPM) is a very famous buzzword today. Applied correctly, BPM can lead to better architectures and more flexible software systems. But when new processes are designed or existing processes shall be improved, it is always hard to predict the resulting performance.

Business Process Simulation (BPS) is of great help in such estimations by using statistical methods to gain a better understanding of runtime behavior. But good BPS tools are rare and costly, making simulation uninteresting for a lot of companies. Because of closed source and closed documentation the tools are seldom useful in research, too.

The main objective of this master thesis is to create an open source BPS tool on top of JBoss jBPM, an open source business process engine. The simulation can answer what-if questions concerning resource numbers or input parameters as well as compare different versions of processes regarding special performance indicators, for example costs or performance. Statistical input data can be taken from audit log data whenever possible. Altogether the tool can simulate completely new processes as well as support continuous improvement.

A tutorial with a simple showcase and a real life prototype demonstrates that the simulation works and how it can be configured and used.

Declaration of Academic Honesty

I hereby declare to have written this Master Thesis on my own, having used only the listed resources and tools.

Bernd Rücker
Stuttgart, 1st of February 2008

Contents

Chapter 1

Introduction

1.1 Motivation

Simulation is not a new topic in computer science. Especially in the field of logistics or production planning it has played an important role for some years now. An example could be the simulation of a harbor loading dock's utilization[1]. Because it is quite expensive to correct planning errors after constructing the dock, a good modeling is a crucial success factor. And due to the random nature of events it is hard to develop an analytical formula for the result under different circumstances, that's why simulation is used.

In the last years, Business Process Management (BPM) became very popular. This maybe surprising at first, because focusing on business processes is not new, actually it got very famous in 1993, when Hammer and Champy introduced the Business Process Reengineering[2]. But because of the evolution of the affected tools, the situation has changed. In the beginning, process descriptions were normally "paper ware", which means documentation only. In most cases today, process descriptions are real models, which have some graphical representation, but can also be interpreted automatically. This increases the value of the process descriptions and makes it interesting for a wide range of companies. Additionally, the current hype around Service-Oriented Architectures (SOA) pushes the development further towards BPM.

When simulation of business processes emerged, these projects included a high effort for capturing statistical data from your running business to enable valuable simulations. This situation is changing, because of the automation of business processes with so called Business Process Engines. These engines do not only execute the process model, but also collect a lot of audit data during execution, which is a good starting point for simulation.

But most of the currently available simulation tools concentrate only on pure simulation, without putting any focus on business processes. Thus you need a special model for simulation on the one hand and have to provide figures from external sources on the other hand, for example via spreadsheet tables. The simulation can answer questions concerning the resulting behavior, compare versions or show impact on changing fig-

[1] for example in [PK05], p. 32

[2] [HC96]

ures. Even if this approach has the advantage, that nearly everything can be modeled and simulated, the downside is the increased effort on the dedicated model you need to build.

A better approach is to integrate the simulation tool in the BPM environment, so it can share the same model with the process engine. This safes effort when modeling a business process and enables accessing historical data of the engine for later optimizations. But there are only few tools available in this area without big shortcomings. And as these tools are very expensive they are normally overkill for small or midrange companies. What is missing is a simulation environment, which is lightweight in the terms of operation and costs. This leverages business process simulation to a much wider range of people, for example smaller companies or universities.

A good opportunity to make it easily available is the more and more accepted open source software. One existing Open Source BPM tool, meeting the requirement to be easy to use and cost-effective, is JBoss jBPM, which is already used in some bigger real life projects. The company JBoss has achieved a good reputation, even in the field of mission critical business software. For this reasons, I consider JBoss jBPM as a good foundation for the simulation environment developed during this thesis.

Summarized the main motivation behind this thesis is to develop a complete BPS environment based on open source tools. With available source code, a good documentation and tutorials on hand, this environment enables a wider range of companies to leverage, a wider range of technicals to learn and a wider range of scientists to study or research BPS concepts.

1.2 Objective and structure of the thesis

The objective of this thesis is:

- To examine the simulation of business processes from the business as well as from the technical perspective.
- To conceptualize the combination of an existing simulation tool with an existing business process engine.
- To implement a business process simulation tool based on open source components.
- To develop a tutorial to enable other people to use or improve my work.
- To use the tool in a real life case study.
- To provide a foundation of freely available components to do further research on business process simulation and optimization.

The thesis starts with an introduction of Business Process Management (BPM) in *chapter 2*. Main concepts are explained and Business Process Simulation (BPS) is motivated from the business point of view. *Chapter 3* introduces simulation itself and provides the reader with the required knowledge to understand the basic concepts. It contains a short outlook on BPS tools on the market.

Chapter 4 describes how I combined the simulation framework DESMO-J and the process engine JBoss jBPM to form an open source BPS tool. To demonstrate the functionality a show case is included covering a small business problem, solved with simulation. The integration of the tool in a real life project is described in *chapter 5*.

A short summary can be found in *chapter 6*. Resources of interest, like for example source codes, are contained in the appendix.

Chapter 2

Business Process Management

2.1 Basic Introduction

2.1.1 BPM and SOA

Everybody talks about "Business Process Management" (BPM), some even expect it to change the IT-Industry[1]. But is that true? Or is BPM just a new buzzword to sell expensive products to customers?

To answer this question, I want to look at the problems and challenges companies have to face. In a global market there is much more competition. More and more products or services can be purchased from a bunch of different companies and through the internet all of them are just "one click away". In this world it is important to be very flexible and to adapt your business model to changes of the market fast. Therefor speed and the ability to change get much more important than size for example.

The situation also changes the demands on the IT. Almost any business model runs without software support. But to allow changes in the business model you have to provide a flexible software system. This flexibility is hard to achieve and very expensive if your business model is hard coded into some programming language.

One observation is that basic functions, like customer or order management, are relatively stable. Only the business processes, using this functionality, are changed frequently. So much more flexibility is needed to change the business process than the basic functionality beyond. Together this brought up the idea of BPM and so called "service oriented architectures" (SOA). The vision is to provide independent software services and "orchestrate" (use) them in business processes.

From an architectural point of view, BPM and SOA make absolutely sense in a lot of typical business software. And to develop software as a collection of independent services is not a new idea, maybe call it *component* instead of *service*. New is the capability to create a business process model, using existing services, which is not only for documentation but also serves as a runtime model for executing the process. A closer look to this concept is shown in chapter 2.1.3. Because the modeled process is as a real runtime model, it always reflects the running processes and must be kept up to date. Changes in the processes may be done by simple changes in the process model.

[1] see for example [HR07]

Hence Business Process Management has a lot of advantages[2]: It forces you to model your business processes, that means you have to understand them. The processes are also used for the software implementation, which saves effort and guarantees that documentation and software are in sync. The process model can be changed much more easily than other parts of the software, which enhances flexibility in turn. Improved processes and automation save time and money. And BPM applied correctly leads toward a SOA, which can improve your overall architecture.

The downside of BPM is the introduction of a new paradigm and new tools which have to be adopted by the people working with it. But this only raises the barrier for starting BPM projects, it is not a disadvantage of BPM itself. The only true disadvantage of process management could be[3], that defined processes constrain the personal freedom of people working within the processes, a problem which is not investigated further in this thesis.

2.1.2 Business processes

In BPM projects the very first task is to identify the business processes. Very often, the information about the process is somehow hidden in the mind of the employees. They know what to do, but an explicit description of the overall business process is missing. This is not only a problem if a employee leaves the company, it also favors process errors, especially on interfaces between different people, departments or companies.

The missing process knowledge is very surprising, because process errors are a main reason for unnecessary high costs, delays, slow fulfillment or little flexibility[4]. Many companies have tried to improve their process knowledge in quality management projects, which produced process landscapes, process descriptions and operating instructions. The problem in the past was very often, that results were "paper ware", which means the process documentations were not really useful and very fast out of date. The people still had to execute the process on their own, there was no real IT support for working with it. If different software systems were involved, data belonging to the process sometimes had to be maintained twice or some complex IT integration was done, for example with proprietary and expensive EAI tools. This situation led to the need of BPM and SOA.

But what exactly is a business process? In the available literature there are different definitions, for example in [HC96] p. 20, [vdAtHW03], [Ses04] p. 40 or [Fis06] p. 12. The main content is almost similar:

> A business process consists of ordered activities, which create from a given input a defined output, which has a value for the customer or market.

In this short definition something is missing, which is obvious for business departments, but has a strong influence on the later implementation in software: Business processes normally include wait states, in which the process waits for an external event or human tasks. In a process-oriented application the human tasks should be triggered by the business process, not the other way round.

[2] compare to [Hav05], p. 7
[3] from a interview with bpm-guide founder Jakob Freund
[4] [Ses04]

The activities of the process can easily be expressed in a graphical format, which is normally some kind of flowchart, for example an UML activity diagram[5]. An important difference between the business process model and more technical models, like UML class diagrams, is the target audience: The process model should be understandable by business *and* technical users. In fact it should serve as a common language between them, which eliminates the gap between business and IT world as much as possible. This vision is promising for the future, even if it is not reached completely yet. One big problem is to enrich the process with all necessary technical details, but to keep an easy graphical representation for the business people. For the future, there could be help on this by defining a real metamodel for business processes (like BPDM, see 2.1.5).

Most of the existing process languages (see 2.1.5) are defined as XML languages. Crucial for the success of a language are editors, which can show the graphical representation of the process definition and allow direct manipulations on them.

2.1.3 Process engines and the BPM architecture

Because of the algorithmic structure of the business process model it is possible in most of the cases to execute it with special software, called *business process engine*. The process model must meet some additional requirements, like being syntactically and semantically unambiguous, but then it can be interpreted directly from the engine. To use the right terms, this means that the engine reads the *process definition* as well as it creates and runs *process instances*. The process instances can contain *manual or automated activities and tasks*. A detailed look at this concept on how it is supported by JBoss jBPM can be found in chapter 4.

Beside the advantage to use the graphically modeled process for the process engine too, a BPM solution targets some additional requirements[6]:

Versioning: Business processes are long running, sometimes up to a year or more. If a new process definition is deployed, the engine must provide some functionality to create new versions of the process. It must be possible to finish already running process instances with the old definition.

Logging and Auditing: During execution of process instances a lot of data can be logged, like for example the amount of time passed, time in between events or duration of tasks. This data can be used later for a deeper analysis of processes or to find process bottlenecks. It can typically be used as simulation input as well.

Monitoring and Administration: At runtime it should be possible to get a detailed overview about all running process instances. Also the administrator should be able to analyze or change certain instances, for example in the case of some exception.

[5] see for example [PP05]
[6] compare to [Hav05] and [BR07]

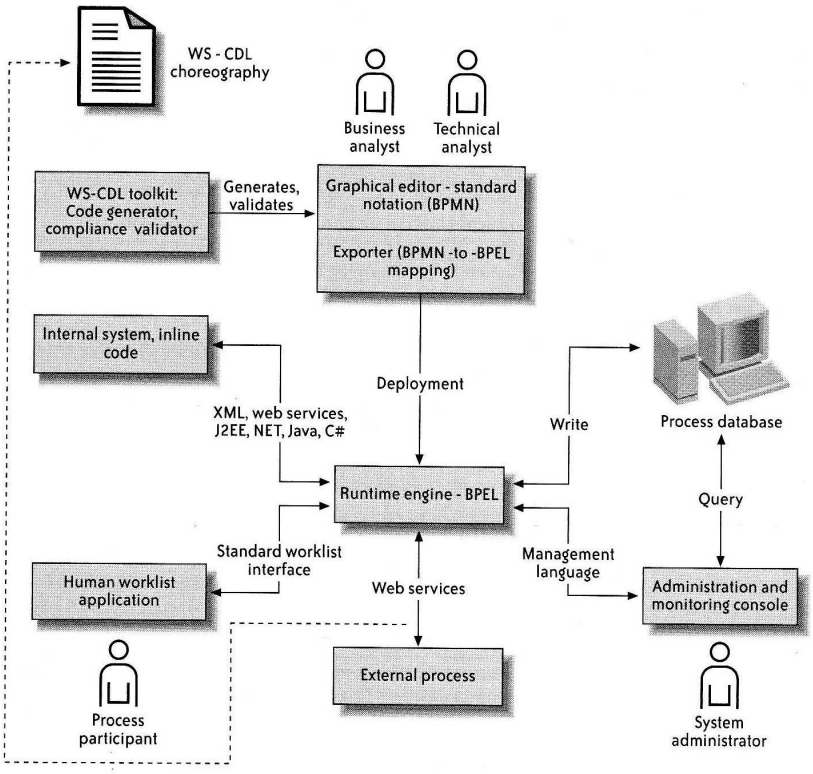


Figure 2.1: A good BPM architecture (from [Hav05]).

Human Interaction: To support human tasks a BPM solution must include some basic role model, which maps special tasks to user groups and some worklist[7] functions or even a graphical worklist user interface.

System Interaction: The process engine must be able to call external functions when they are needed in the process. The architecture should support different technologies, for example web services or Java EE integration technologies.

Figure 2.1 shows the overview of a BPM architecture with similarities to the WfMC's reference model[8]. This architecture is the basis of the most modern BPM products, even if the technolgies mentioned can differ. JBoss jBPM, which is used in this thesis, follows basically the same architecture, but neither BPMN nor Web Services are used for example.

2.1.4 Definition summary: business process and engine

A good summary for the definition of a business process and a process engine can be found in [Hav05], p. 18:

> A business process is the step-by-step algorithm to achieve a business objective. The best visualization of a business process is a flowchart. A process

[7] also called todolist or inbox
[8] [WfM07]

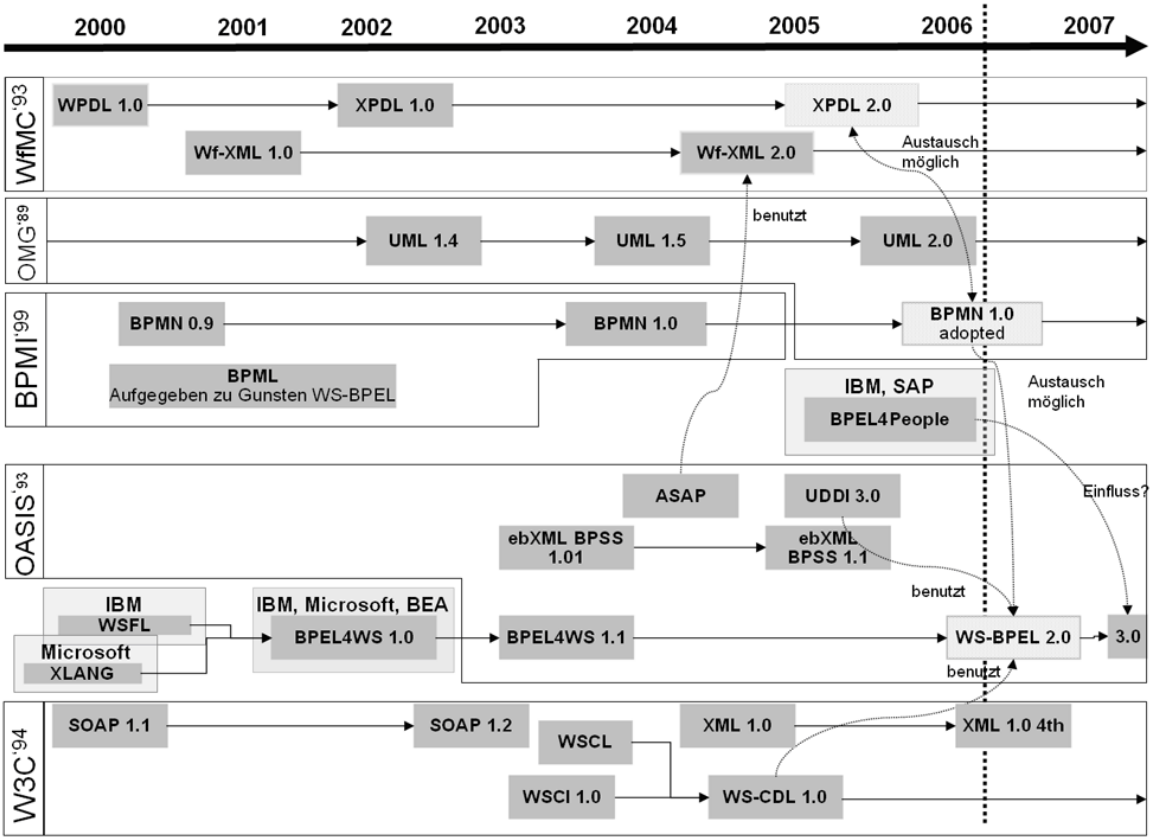


Figure 2.2: History of BPM standards.

> can actually be executed by a process engine, provided its logic is defined precisely and unambiguously. When a process definition is input to an engine, the engine can run instances of the process. The steps of the process are called activities. Business process modeling (BPM) is the study of the design and execution of processes.

Please note, that Havey speaks of BPM as business process modeling. In this thesis the abbreviation refers to the whole business process management, which widens the scope by including monitoring and continuous improvement of the process. Simulation plays an important role in these advanced topics.

2.1.5 Standards, notations and products

This chapter should give a short overview about the existing standards and notations in the field of BPM. It is based on [BR07] and [Bar06]. Knowledge of the standards is interesting to judge the future direction of the BPM products or to evaluate tools. Figure 2.2[9] shows the different standards and the historical development.

Business Process Modeling Notation (BPMN): The goal of BPMN is to have a process notation for the business analyst, which is later transformed into a

[9] from [Bar06]

technical process language (like BPEL). Some tools are already capable of this functionality, which should be improved with the introduction of a complete metamodel for BPMN, called BPDM. With that functionality BPMN bridges the gap between the business and technical world. The main part of BPMN is the business process diagram (BPD), which is supported by a wide range of tools (visit http://www.bpmn.org for a list). After the merge of the BPMI with the OMG, it is now definitely the most important notation for business processes.

Business Process Definition Metamodel (BPDM): The metamodel defines model elements for business processes. Similar to the UML metamodel this allows to generate code out of the models or to look at the same model from different views, for example a graphical and textual view. A concrete application will be the generation of BPEL processes from BPMN process models.

Unified Modeling Language (UML): The very famous modeling language UML from the OMG can also be used for business process modeling. Especially the activity diagram is a good choice. But BPMN provides more features for processes and is easier to understand by business people, so it should be preferred if possible.

Business Process Modeling Language (BPML): BPML from the BPMI allows you to define business processes in XML. Important topics like transactions, process data, exception handling and so on are completely covered. The BPML process should be completely independent from a special business process engine or tool. But BPML lost against BPEL and is not developed further. So it does not play an important role any more.

Business Process Execution Language (BPEL): BPEL (also called BPEL4WS in the past) is a XML based language, which is able to orchestrate web services to a business process. To be correct, you don't need web services but a WSDL based interface. BPEL evolved from WSFL from IBM and XLANG from Microsoft and is now developed further from the OASIS. It is backed by a group of big vendors like for example IBM, Microsoft, Oracle and Bea. So there are already a lot of tools and business process engines around, which support BPEL and it looks like it wins the standard war. You are abtle to express complete business processes with BPEL, but on a very technical level. So it should be used together with more business oriented notations, like BPMN.

XML Process Definition Language (XPDL): XPDL is again a XML based language, standardized by the WfMC. Compared to BPEL, XPDL is not limited to web services. The business process engine has to provide a mapping from the calls to external activities to the concrete implementation. Even if XPDL is not so famous like BPEL today, there are a lot of tools and graphical designer available.

Business Process Specification Schema (BPSS): This specification from OASIS defines the exchange format for negotiation between collaborating companies, for instance automated procurement. The focus is on choreographie, not on company internal business processes.

WF-XML: WF-XML does not define a business process language, but administration interfaces, for example how process definitions can be deployed to a business process engine.

To complete the picture, the involved bodys are also mentioned:

BPMI: The "Business Process Management Initiative" deals with standards around BPM. The BPMI merged in June 2005 with the OMG after unsuccessful negotiations with the WfMC.

OMG: The "Object Management Group" is a well known international consortium, developing vendor independent standards in the field of enterprise applications.

OASIS: The "'Organization for the Advancement of Structured Information Standards"' is an international non profit organization, which deals with the development of standards in the area of e-business and web services.

WfMC: Software vendors, customers and research institutions have joined together in the "Workflow Management Coalition". The objective of this non profit organization is to define standards for workflow management systems and environment. The WfMC got famous with their widely accepted reference model for workflow systems.

These standards and the acceptance of at least some of them in the market is an important success factor for BPM. But even today there is no standard available which takes simulation into account. However, the aim of this thesis is not to define simulation features for some of the mentioned standards above, but to develop a prototype based on the proprietary process language used inside of JBoss jBPM , introduced in chapter 4. Some reasons why this tool and the proprietary language was chosen were already given in chapter chapter 1.1.

2.2 Business demands

2.2.1 Process Management life-cycle

This chapter examines business processes from a management perspective, including the application of BPM in own projects. The BPM life-cycle in figure 2.3 on the following page should serve as a basis. Currently there is no uniform view to the life-cycle in the literature or across BPM vendors. You can find a lot of versions differing in granularity or maybe expressing different aspects, depending on the features of the vendor's software. Typically all the life-cycles are related to the famous Deming cycle[10], which contains the four phases plan, do, check and act. It is often called PDCA cycle and can be applied for problem solving in gerneral. My life-cycle is based on [NRvdA06] and [BR07], only the names of the phases have changed into more intuitive ones. Related life-cycles can be found in [RvH04], [vdAtHW03] or [WMF+07]. The phases of the life-cycle are:

[10] see for example http://en.wikipedia.org/wiki/PDCA

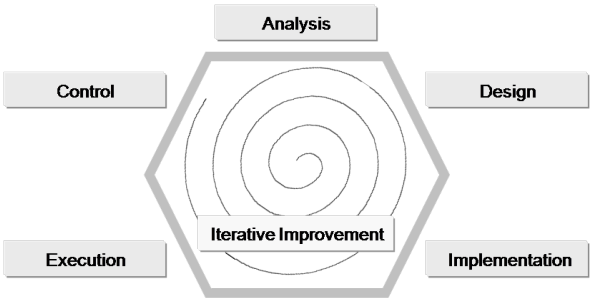


Figure 2.3: Business Process Management life-cycle.

Analysis: If a process is modeled for the first time, the analysis phase is typically the first one in BPM projects. It's aim is to study existing workflows and to identify bottlenecks or weaknesses. For already running processes, the focus of this phase is to examine aggregated performance data to identify weaknesses of the process design, for example resource bottlenecks or delays. The results from the analysis provide direct feedback for design changes and deliver data for simulation of process alternatives.

Design: If there doesn't exist a process model yet, it will be developed in this phase. If there is already an existing process model, the task will be to create an improved alternative process, which should remedy diagnosed weaknesses (e.g. discovered bottlenecks). The design phase should not only concentrate on static modeling but also support dynamic experiments with the process to evaluate design decisions and to allow a better understanding of them as soon as possible. To support this, simulation capabilities of the BPM tool are necessary. To make simulation easier while improving existing processes, the use of historical data of the already running process as input for the simulation is desired.

Implementation: The focus of the implementation phase is on putting the modeled process into execution. Therefor it must be enriched by IT engineers with technical details as shown in figure 2.4. Also the process should be tested somehow in this phase, like any other piece of software.

Execution: The prior preparation enables the deployment of the process on a business process engine. This engine can now execute process instances (see 2.1.3 for details).

Control: In the control phase, running processes are inspected and monitored. This can be done either by watching specific instances or an aggregated view. Process controlling may also include the comparison of plan figures or goals with current performance figures. This allows the user to identify temporary bottlenecks, deal with exceptional cases or recognize deviations from the plan as soon as possible.

Currently not all BPM tools support all phases. And if they do, very often the interoperability among the phases is not very sophisticated, which decreases the usability of the tools. The biggest problem is the analysis phase and the feedback to the

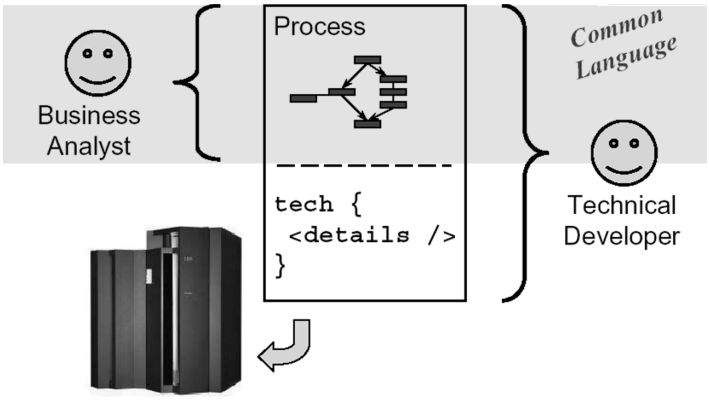


Figure 2.4: The process, a common collaboration language.

design. Simulation tools are often some kind of add on to the BPM tool and poorly integrated, especially in using historical data. Additionally, you can also think of simulation integrated with the control phase. An example is the simulation of alternative process designs while the real process is executed. By doing so, the tool always has up-to-date information about potential improvements. This will be an important task for the future.

At least since management concepts like KAIZEN, Six Sigma or Total Quality Management (TQM), companies care about continuous improvement of their business processes. This is supported by the presented BPM life-cycle pretty well, but is still complex and as mentioned not all tools are of great help in this field. Basically continuous improvement means defining and measuring key performance indicators to identify weaknesses as soon as possible and to take actions to stop it. In research there exist some ideas to leverage continuous improvement. One basic idea is, to optimize processes or at least to find possible improvements automatically by defined heuristics or genetic algorithms. These ideas are covered in section 3.5.

2.2.2 Business Process Reengineering (BPR)

In contrast to continuous improvement Business Process Reengineering (BPR) means

> the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed.[11]

The big problem of BPR projects is the high risk of failure[12] and the need for big investments. Nevertheless, historically BPR has done a very good job in raising peoples awareness of business processes. Because of the radical change, BPR applies primarily to companies which never dealt with business processes before. In this case, it can be a good choice, even if I agree with [Ses04], that you shouldn't skip an as-is analysis as proposed by [HC96].

[11] [HC96]
[12] [Ses04]

One challenge of the radical changes in the processes is that nobody can really predict the outcoming performance and costs. Unsubstantiated estimations may often be misleading and because of the high dynamics of complex processes bottlenecks are frequently not intuitive. Hence, simulation can be an important tool to support BPR. Work durations and frequency of events are easier to estimate and the simulation cares about the dynamic characteristic of reality. The difference between simulating for continuous improvement and simulating for BPR is the missing historical data in BPR projects, so input must be provided manually. To keep it practicable, it is vital to design easy process models for simulation and concentrate on key tasks first.

2.2.3 Key performance indicators for business processes

To achieve easy process controlling in the BPM life-cycle you need to define performance indicators, which give information about your running processes. These figures can be used to define goals, monitor the current performance and compare different process versions. The most important performance indicators are called Key Performance Indicators (KPI). KPIs refer not only to business processes but to strategic company planning at all. Slightly different definitions for KPI exist, for example [Web07] or the very short one from [Reh07]:

> Key Performance Indicators are quantifiable measurements, agreed to beforehand, that reflect the critical success factors of an organization.

Normally some KPIs of the company are related to the processes or can be defined at least. The challenge in this field is to identify as few as possible, but powerful performance indicators[13]. During the improvement of processes you should concentrate on these performance indicators. According to [Ses04] the 5 main categories of KPIs related to processes are: customer satisfaction, meeting deadlines, process quality, process costs and process duration.

This chapter will only look at some of the indicators, where the measurement can be automated. These indicators are possible candidates for results of simulation runs later. Indicators, which have no direct connection to the process runs and can only be measured by surveys or things like that, are skipped. Average process duration is very easy to record for example, so it is included. It may have a strong influence on customer satisfaction, a KPI which is skipped, because it is not measurably automatically.

I will split up the indicators into two groups. The first group consists of measures, which are the same for all processes and can be provided by the business process engine itself. I call them *technical indicators*. For the second group, which I call *business indicators*, more business information is needed, for example the value of an order to calculate the loss in case of order cancellation. The business indicators cannot be defined globally, they are special for every single business process.

First I will mention the *technical indicators* covered in this thesis:

Process duration: Process duration is the time from the start of the process till the end of the process. From a customer perspective it is the time from the request till the desired result is reached.

[13] [Fis06], p. 25

Cycle time: The cycle time is the sum of the duration of all process paths, even if they run in parallel. This gives some evidence about the needed overall processing time.

Transfer and Waiting time: Transferring goods (or paper) from one machine (or desk) to another can take some time. During this time the process is idle and waits. These times are often not included in business process models.

And if all resources (people or machines) are busy, the process has to queue. During this time it is idle again. Waiting time is often not modeled. From historical data about process runs, the sum of waiting and transfer time can be extracted easily, it is just the difference of the finish date of the last activity and the start date of the next task. The problem is to distinguish between them, which is normally not possible without additional informations.

Processing time: This is the total time spent in different activities. If transfer and waiting times are added, the result should be the cycle time.

Now I will present some *business indicators*, explanations how these can be captured automatically will follow in chapter 4:

Error rate: The error rate defines how much processes were unsuccessful. Normally an error is indicated by some alternative process flow or special process data.

First Pass Yield (FPY): The FPY gives the rate of processes, which were successful in the first run. This doesn't mean that they only didn't fail, but corrective actions are not allowed. Normally, there are special process flows, disqualifying a process instance to count as passed for this indicator.

Adherence to delivery dates: There are different figures to express the adherence to delivery dates. First of all, the rate of deliveries in time is interesting. But also the average delay can be an important measure. To get these information, it must be defined in the process model how to get the delivery date or delays must be marked by some special process data or changed process flow.

Loss of orders: It is theoretical possible to calculate a companys' loss because of canceled processes, for example orders. In practice, this is a very special topic because there can be a lot of reasons for cancellation, so a further examination is required. Also the value of an order must be provided somehow in the process data.

Process costs: The sum of all costs caused by the run of a single process instance. Basically I see four types of costs: People performing a human activity, resources needed to fulfill a human or automated activity, objects needed for the process respectively external fees (for example shipping fee) and service calls, which cause some fee (for example the German SCHUFA information service). [Fis06] combined the last two as the group "object costs" and named the other "employee costs" and "material costs".

Resource utilization: If the maximum number of resources is given, it is possible to calculate the utilization from historical data. The problem with this figure is that it may behave very strange when resources are added or removed. Because this results of the high dynamics of a system, the figure is a perfect candidate to be explored through simulation.

Beside these standard indicators you can find much more in different project environments. One final word about staff requirements: With the cycle time (per user group) you can get some idea about the required staff. Together with the process duration you may even calculate an average. But this average is not a very good one, because you don't know if there are any peaks for resources on special times or if the process duration is already longer than needed because of bottlenecks. So you should apply some further thoughts about process dynamics in this case.

Everybody agrees with the need to include support for KPIs in the BPM tools today. Basically, there is one important acronyms for this: *Business Activity Monitoring (BAM)*[14]. BAM deals with calculating indicators from historical data, diagnosis of current performance or even more sophisticated data mining in audit data. BAM tools can also contain rules to take actions on their own if some conditions are met. BAM is expected to be the next big hype[15], but includes a lot of challenges. BAM tools can deliver information helpful in the design phase of the life-cycle, which also includes input for possible simulations.

2.3 Business Process Simulation (BPS) from the BPM perspective

2.3.1 Vision and position in the life-cycle

Business Process Simulation (BPS) and some applications of it were already mentioned a few times before in this thesis. This section now focuses on the vision, typical goals and requirements in simulation projects. For a general introduction into simulation, please refer to chapter 3. In short, BPS allows you to achieve a detailed understanding about the behavior of business processes during runtime without putting them into production. The simulation can evaluate an *experiment*, which includes certain process definitions and special parameter settings [16]. For evaluation the defined performance indicators can be used. The discovery of the average process duration for a given process definition with a predefined number of available resources could be an example. Section 4.4 demonstrates based on a showcase two typical use cases and the business value achieved.

If you look at the BPM life-cycle in figure 2.3 again BPS is located in the design phase. But the overall vision of BPS for process improvement is to use real life data from historical process runs, normally retrieved from BAM. So it also has connections to the Analysis phase. The most recent visions[17] include a connection to the execution

[14] see for example [vdAtHW03]
[15] [DWM02]
[16] [BKR05]
[17] like expressed in a newsletter from BPTrends: [PH03]

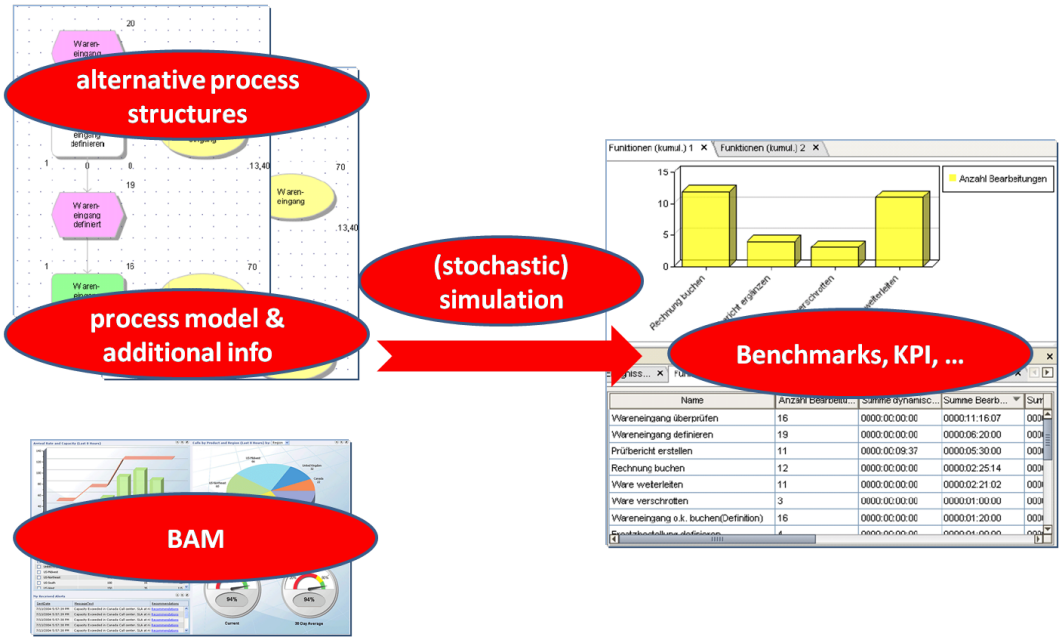


Figure 2.5: The business process simulation vision.

and control phase as well, for example you could image to simulate alternative process structure with real life data in parallel to the execution and have an always up-to-date benchmark of possible process improvements.

Simulation *cannot* find optimal solutions for parameters or process alternatives, the aim is only the evaluation of defined experiments. Which improvements of the process are worth to simulate is a decision made by the user, normally based on experience and examination of historical data[18]. One alternative to this approach is to combine simulation with evolutionary algorithms, which "search" for optimal solutions and use the simulation as a fitness function. Section 3.5 looks on this in more detail.

Another aim of simulation is visualization. Sometimes this refers to a walk through of one single process instance, but often it means that a simulation run like described above can be watched or debugged while running. Or at least some playback of the simulation can be shown. This capability is helpful especially when studying or explaining the process. In the rest of the thesis, I will skip this goal because it is out of scope. Maybe it is an interesting field for future work.

2.3.2 Typical goals

Simulation causes significant effort to provide all necessary input, so it should not be applied without any reason. Typical goals, according to [BKR05] are listed below, examples can be found in chapter 4.4:

- Identify cycle times and process durations for new or changed processes.
- Identify process costs.

[18] [BKR05]

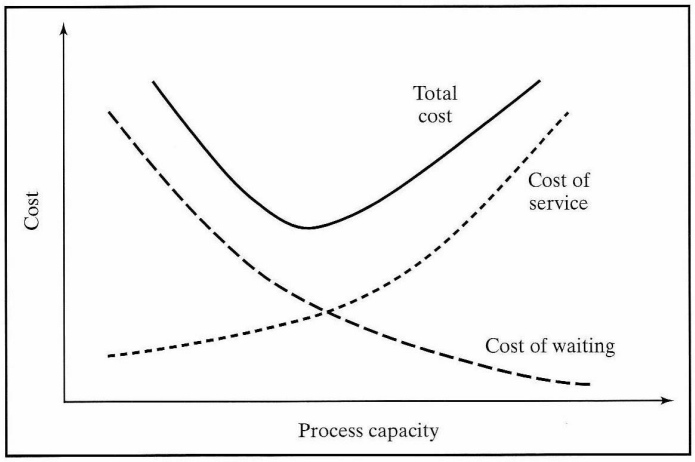


Figure 2.6: Economic Tradeoff between service capacity and waiting times.

- Support capacity or staff planning.
- Forecast effects of changing amount of input events (for example the double amount of orders).
- Benchmark alternative process structures.
- Benchmark different parameter configurations.

The most famous goal in literature is to simulate new designed or improved processes[19], for example in BPR or so called Business Process Change (BPC) projects. This is followed by capacity or staff planning[20], where a typical goal is to find a good tradeoff between costs for resources and costs of waiting times like illustrated in figure 2.6[21].

Taking into account that only a small percentage of the BPM clients use simulation[22], I expect the focus in reality on goals, which are easy to achieve. If the simulation environment is capable of using historical data as input for simulation, easy achievable goals are basically capacity planning and benchmarks of different parameter configurations. But the poor tool support of historical data usage is a big show stopper for simulation.

2.3.3 Missing attributes in process models

As already mentioned, extra information about probabilities or quantities is needed to simulate a process model. Independently how this data is specified in the chosen tool or process language, or if it can be retrieved directly from historical data, the main attributes needed are always more or less the same. Chapter 4.3 looks on a concrete realization in JBoss jBPM later. But first the process attributes according to [BKR05]:

[19] for example [BVCH07] or [JVN06]
[20] for example [PH03]
[21] from [LML04], p. 173
[22] [PH03]

Probability of decision paths: Decisions in processes can be made either automatically, based on process data, or manually by humans. Maybe the automatic decision can be kept for simulation, if the content of the data is simulated correctly. Otherwise or in case of human decisions, probabilities for every outgoing path must be provided, which enables the simulation engine to make the decision on itself.

Frequency or probabilities of start and external events: External events, most prominent is the process start event, must be generated by the simulation engine. Most of the time, these events should contain some sample data, too.

Resources needed for an activity: For every activity, manual or automated, there has to be a definition which resources (e.g. user groups or machines) are needed to perform the activity.

Processing time of activities: The time needed to fulfill an activity is also an essential part of information for simulation. If the final results should include costs, also costs have to be defined for every resource.

Transport times: As mentioned in section 2.2.3, there may be additional transport time between activities, which in most of the cases is not modeled, but can have significant influence on the simulation result.

Probability for waste: In a production process there is often some percentage of waste. If these figures can influence the simulation result, some probabilities should be modeled.

Additional costs for an activity: If there are additional costs for activities (compare to section 2.2.3), these must be added to the process model for simulation.

Resource availability or shift schedules: To gain a realistic view of process duration and resource utilization, the availability of resources must be specified. The more detailed figures you have, the better conclusions you can make about bottlenecks, even on special times of the day or dates.

Changes of process data: Some calls to external systems, events or human activities can change the process data. Sometimes this can be ignored for simulation, but often you have to deal with it. So these process data changes must be somehow included in the simulation model, which is not an easy task to generalize. The realization obviously will depend very much on the chosen process language and tool.

Which attributes are needed and what level of detail should be provided depends very much on the simulation goal. Frequency or probabilities can be expressed by simple or very sophisticated statistical distributions. The same goes for for processing or transport times, they can be defined statically by some average or by a much more realistic distribution. The art is to provide as much details as necessary but as little as possible to get reliable results with little effort. A simulation tool should support very sophisticated statistic distributions as well as easy default values. To use this tool correctly it needs some experience of the user.

2.3.4 Requirements on business process

Not every business process is suitable for simulation. According to [BKR05] there are some basic requirements, which should be met:

Reasonable stable process: In simulation runs you often look at longer durations, like one year. If the process changes much more often, the simulation results are maybe not reliable. Also the effort of simulation grows if you change the process very often. But this is not a strict requirement. If you can use historical data as simulation input and make continuous improvements on process designs, simulation could become a very helpful tool, even if you change process designs more often[23].

High frequency of occurrence: Obviously it doesn't make too much sense to put a lot of effort in processes, which are not executed frequently enough.

Input data can be collected in reasonable quality: The quality of simulation results depends strongly on the provided input data. If appropriate parameters cannot be collected with reasonable effort, simulation can't help.

At least one goal is applicable: This requirement is also obvious. Without any goal in mind, a simulation project can not be started.

As a remark I want to add, that some of the requirements will get less important if there is a good simulation tool available. Especially the possibility to use historical data as simulation input lowers the limit dramatically.

2.3.5 Process model for simulation projects

As explained in section 2.2.1 simulation is basically applied in the design phase of the BPM life-cycle. This section takes a closer look on how to leverage simulation for this purpose. Typically every simulation project contains the following steps[24]:

1. collecting data in real word,
2. depict randomness, e.g. find statistical distributions,
3. create a simulation model,
4. create scenarios, for example for different parameter values,
5. execute and evaluate, which normally gives feedback to create new and maybe better scenarios.

This is basically the same in business process simulation[25]. Nevertheless, a more detailed process model special for BPS projects is possible. One recommendation is visualized in figure 2.7 on the following page[26]. It includes the following steps or phases:

[23] compare to [PH03]
[24] [Pro97], p. 25ff.
[25] see for example [LML04], p. 71
[26] after [BKR05], p. 440

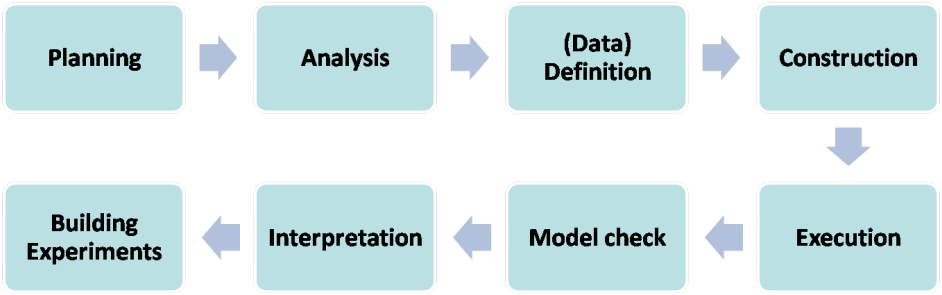


Figure 2.7: Steps of a business process simulation project.

Planning: In the planning phase the main goals of the simulation are defined. This includes a selection of performance indicators, which can be used to evaluate simulation results and different alternatives. This phase should also clarify if a simulation project can delivery any value at all.

Analysis: In this phase, processes and its variants to simulate have to be chosen. This means as well to agree on the scope of the simulation. Neither all processes have to be included in a simulation project, nor have all resources to be modeled. This is a decision made by the business analyst.

Definition: After defining which processes should be simulated, input parameters have to be found. This may include the identification of statistical distributions for stochastic input. In the best case, this information is available from historical data. Otherwise, for example in BPR projects, it has to be captured somehow, which can be complex and costly.

Construction: In the construction phase a simulation model is designed. When improving a business process the model should already be available, in the case of BPR projects it has to be build. Additionally it has to be enriched with information or configurations for the simulation run. It can also be desirable to reduce the process elements to make the simulation run easier, faster or maybe the data collection process cheaper or even feasible at all. The changes on the model should be made with respect to the defined simulation goal.

Execution: Executing means running the simulation based on the constructed process model.

Model check: After the first simulation run the results should be checked for plausibility. It is proposed to start with the simulation of the status quo, with means supplying input parameters corresponding to the current active process[27]. This facilitates correctness testing of the simulation model, because it should reflect the performance indicators experienced in real life. This step can be split into verification and validation. The verification checks if the model is correctly simulated, which should always be the case when using a dedicated BPS tool. The

[27] see for example [Pro97], p. 38

interesting part is to validate, that the process model and the input parameters reflect the reality as close as possible.

Interpretation: The simulation results and especially basic performance indicators have to be analyzed to find weaknesses of the process or parameters like resource count. This can lead to ideas how to improve the process or how to deal with resource bottlenecks.

Building experiments: Every simulation run leads to new ideas for scenarios. These scenarios can be bundled into experiments, which allows easy comparison between them. Scenarios can differ in process structure as well as in parameter configurations.

Chapter 3

Simulation

3.1 Basic Introduction

Simulation is an experimental technique for exploring the behavior of modeled systems[1]. A model is always an abstraction of the reality, a simplification, applied in order to manage complexity[2]. But even a simplified view on the world can be arbitrarily complex, especially when a lot of independent components and events play together.

If you think again of business processes, the complexity emerges from many process instances, running in parallel, each one in a different state, consuming different resources. Maybe one process has to wait for resources to become available, so there are interferences between the instances which are not always intuitive.

Even if mathematical optimization is sometimes possible for such problems, it is simply too complex to apply in most of the cases. "Analytical models differ from simulations in that they provide a set of equations for which closed-form solutions can be obtained[3]." But this puts much more constraints on the model and makes it normally less realistic. On the negative side, simulations cannot find optimal solutions, they "just" compute results for special situations and gives hints for the behavior of the model. Optimization must be done separately by "playing around" with different scenarios to find good models, this is called *sensitivity analysis*. Especially in the domain of BPM this approach is much more intuitive, because domain experts or BPM consultants normally don't have very sophisticated mathematical skills. Even if simulation is not easy to use, good tools can minimize the required mathematical and statistical knowledge and make this kind of optimization available to a much wider range of people.

A lot of slightly different definitions for the term simulation can be found, in this thesis I will quote an older but still fitting one from Shannon:

> Simulation is the process of describing a real system and using this model for experimentation, with the goal of understanding the system's behavior

[1] compare to [PK05], p.3
[2] compare to [PK05], p.6
[3] compare to [PK05], p.10

> or to explore alternative strategies for its operation.[4]

There is evidence that this definition is still valid because of newer definitions, which are based on it. For example in [Smi98]:

> Simulation is the process of designing a model of a real or imagined system and conducting experiments with that model. The purpose of simulation experiments is to understand the behavior of the system or evaluate strategies for the operation of the system.

There are basically two different kinds of simulation[5]:

Discrete Simulations: The assumption is made that all state changes of the simulation model happen in a discrete event in time. Hence, nothing happens between two neighboring points in time. So the simulation run is a finite sequence of model states.

Continuous Simulations: By contrast, state changes continuously in this kind of simulation, for example chemical processes. Describing models for this type of simulation normally ends up in finding fitting equations.

In this thesis the so called *Discrete Event Simulation (DES)* is used. Obviously it is the natural choice for simulating business processes, because everything in the context of processes can be expressed as a discrete event in time. For example the order of a customer in a webshop, the start and end of packing the goods and the moment it is handed over to the delivery service. Information about other types of simulation are skipped in this thesis.

It is very important to be aware that the quality of the result of a simulation depends very much of the correctness of the model and provided input data. Especially with complex models, small errors here and there combined with small numerical errors can accumulate and change the whole simulation result dramatically. Therefor results should never be treated as facts and models or input data should be validated carefully.

3.2 Discrete Event Simulation (DES)

3.2.1 Overview

This section looks more into the basic concepts behind Discrete Event Simulation (DES). Basically a DES model consists of entities, whose state can change over time, following some rules. This can be compared to objects in the object oriented paradigm. They also have an internal state, which can be changed by defined methods. To trigger the state changes some special model time is needed and state changes must be mapped to a certain point in the model time.

The model time is completely independent from real time, which keeps the simulation unaffected from real computing times. This is necessary if you want to simulate

[4] [Sha75]

[5] compare to [PK05], p.11

very complex systems which are much slower to compute than runtime in reality as well as if you want to simulate a whole year in some short running simulation.

State changes are triggered through "discrete events, which occur at discrete points along the simulated model's time axis"[6]. The simulation time jumps from event to event, in time between events the state of the entities cannot be changed. To implement this, you need some kind of central controller, which provides a central model clock and keeps a list of candidate events. Then an easy algorithm can always select the next event in time, set the model time to the appropriate time and execute the event. Events at the same time can either be prioritized, executed in any order or handled by some sophisticated algorithm. This depends on the simulation requirements.

This concept allows to express a whole simulation run as a sequence of events[7]. Events can be generated either externally, for example process start events, or internally, which means they are generated by entity state changes. Event times can be deterministic or stochastic[8].

For time synchronization in DES, events are not the only existing concept. Beside it, you can also use activities, which express two events, start and end, and the time in between. The third valid option is to use processes, which describe a set of activities to completely model the life-cycle of an entity. These concepts are not explained any further in this thesis because they are not necessary, you can find information about them in [PK05], p. 28.

A typical DES system, like the framework I use in this thesis, consists of the following components[9]:

Model state: At each point in time, the entities must have some defined state, which must be somehow implemented in the system.

Simulation clock: The simulation clock can always be asked about the current model time, which doesn't has any connection to the real time.

Event lists: As described above, an ordered list of events scheduled for some model time in the future is needed.

Central controller / Scheduler: Some central controller must execute the described main algorithm (selecting events, setting the model time and execute the events) and terminate the simulation at some defined point.

Random number generator: So-called pseudo-random number generators can create streams of data, needed to simulate random events or behavior. The generator can create reproducible streams of numbers following some stochastic distribution, thats why they are called pseudo-random. It is realized by an initial value, called seed. With the same seed, the generator produces the same stream of numbers. Chapter 3.3.1 goes into more detail.

Statistical counters / Data collectors: They can collect statistical data of experiment runs and maybe interpret them somehow.

[6] [PK05], p. 25
[7] [PK05], p. 26
[8] [PK05], p. 27
[9] compare to [PK05], p. 265

3.2.2 Modeling styles

This section takes a short look into the main modeling styles of simulation systems in order to provide a basis to evaluate which is best suited for Business Process Simulation. There are other modeling styles as well, but they are not so common and not needed for BPS, so I will skip them in this thesis. Modeling types in general deal with the problem how to model the passage of time, because simulation time is totally decoupled from real processing time. Discrete events in time have to be scheduled and executed at the right model time. The most dominant solutions[10] are the *event-oriented* and *process-oriented* view, which are used in a lot of out-of-the-box simulation tools. Additionally the *transaction-oriented* style is interesting for simulation. The main difference is how the entities and their interactions are modeled.

Process-oriented approach

In the process-oriented approach, the whole life-cycle of all entity types is modeled. No more information is needed.

> During a processes' active phases model state changes will occur. While the modeled activities or actions may ask for some model time to pass, the state transformation they cause are instantaneous at the model time level.[11]

The system is "controlled" by the sum of all entity processes, which sometimes ask for model time to pass. This is the time when the control flow passes to the simulation framework, which can pass it on to other processes. Figure 3.1 on the next page visualizes this idea. Please note that the shown sequence diagram is only for visualization and not correct. Normally the inner implementation is more efficient than the one shown in this figure.

Event-oriented approach

In event-oriented simulations the focus is on the events in time, looking at all events of all entities. The events are not clustered to processes for each entity like in the process-oriented approach, but grouped by their time of occurrence. This "allows a clear separation between the specifications of system structure and behavior"[12]. The Controlling is realized via some central scheduler with a event list, which processes queued events sequentially. The control flow always returns after the event execution, like shown in figure 3.2 on page 30.

If you image the reactivation of processes in the process-oriented modeling style as events, both approaches become very similar. "The main difference to an event method is that a process does not need to start execution at its beginning, but will rather continue from the point at which is was last deactivated."[13]

[10] according to [PK05], p. 97 and [BFS87], p. 13
[11] [PK05], p. 98
[12] [PK05], p. 108
[13] [PK05], p. 100

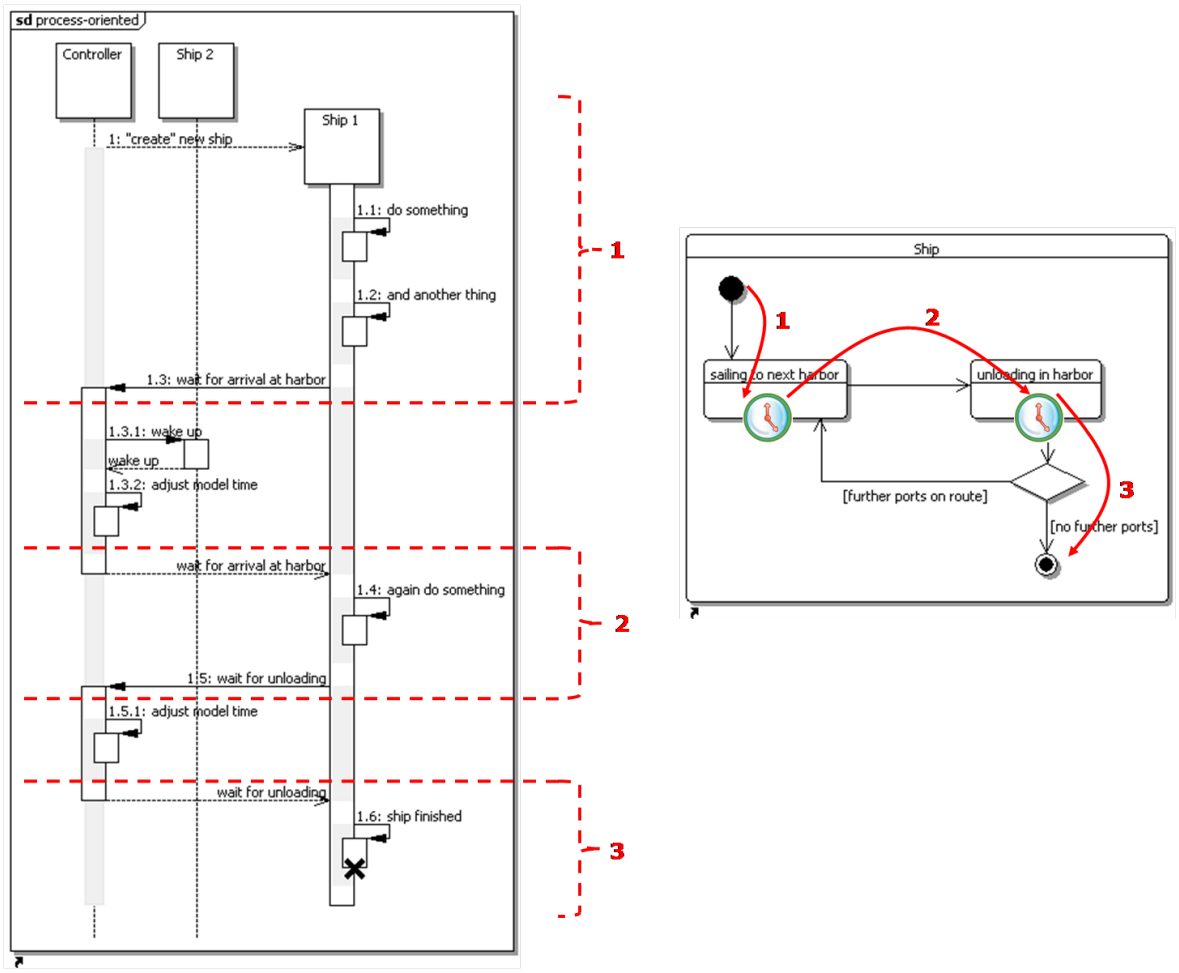


Figure 3.1: Process-oriented modeling style.

Transaction-oriented approach

In transaction-oriented models the entity life-cycle is expressed as transactions, which flow through blocks. These blocks (of execution) can change the entity state. It is very similar to the process-oriented view. The difference is that required resources are not modeled as a processes itself, but as resources constrained by number. A transaction can acquire one or more resources and release them after some time. If no resource is available, the transaction has to wait until one becomes available. The resulting model is much easier[14], because most of the entities of the process-oriented model are realized as simple resources in the transaction-oriented one. On the negative side, this modeling style can cause deadlocks if transactions need more than one resource. This must be considered while building transaction-oriented models. As already mentioned, process-oriented models can be expressed event-oriented, the same goes for transaction-oriented models.

[14] compare to [PK05], p. 129

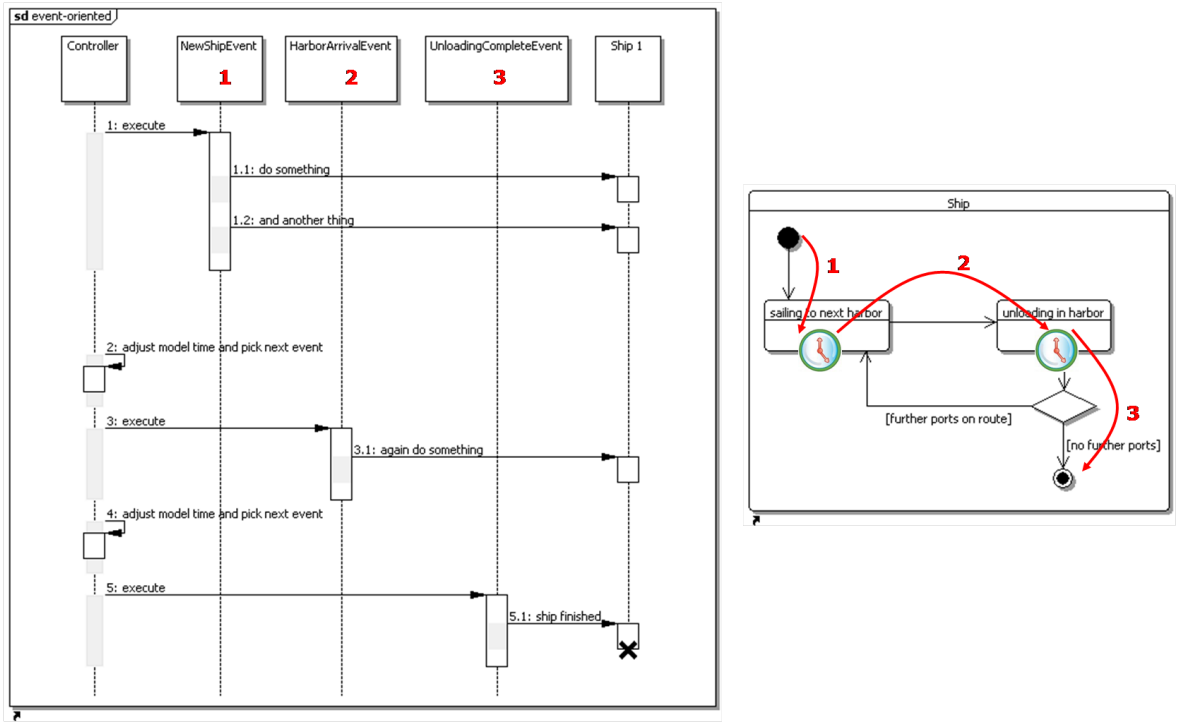


Figure 3.2: Event-oriented modeling style.

Choose modeling style for BPS

The choice for one of the styles depends very much on the type of simulation. Often it is handy to mix modeling styles to express different aspects of the system in the modeling style fitting best. For example unexpected external events are well suited for event-orientated modeling, whereas some complex entities may be modeled transaction-oriented more naturally. This mix of modeling styles is supported by the simulation tool used in this thesis (see chapter 4.1.2), even if that is not the case for most of the available simulation tools today[15].

The decision, which modeling style should be used for BPS is a very crucial one. Actually I think it depends on the concrete implementation approach, there are two main possibilities:

- Create a dedicated simulation model for the business process in the language of the simulation tool.
- Use a business process engine and express the business process in the appropriate language (like described in section 2.1.3). The simulation tool only steers the engine.

This thesis puts the main focus on bringing together BPM and Simulation, hence the choice to use the existing business process engine is obvious. With using a business process engine there are some main aspects, which should be taken into account while choosing the modeling style:

[15] [PK05], p. 140

Business Process Model: If an existing business process model can be reused a lot of work for building an own simulation model can be saved. And the own simulation model wouldn't be integrated in the BPM life-cycle, so it can become outdated quickly. What needs to be mapped from the business process model to use it in simulation is the passage of model time in wait states.

Resources: Resources are an important concept in BPS, they can be humans, equipment like machines or vehicles, or maybe money. It is enough to model them as constrained resources, which are consumed when needed and released afterwords. Processes which need unavailable resources have to queue up. To avoid confusion, resources in this meaning of constrained resources with a queue are called *resource pools* from now on to. Depending on the approach of implementation, the resource pools can be real resource pools in the transaction-oriented modeling style or entities, in other modeling styles.

Events: Business Process Engines are very event-based, because they react on external events with special process flows and actions.

Entities: At the first look, it seems like the business processes itself and maybe resources are the entities. This is not wrong, but we will see later, that a better fitting approach is to make tasks or wait states to the main entities of the simulation.

So the whole system seems to be process-oriented or maybe transaction-oriented. But if we ignore the features, handled already by the business process engine, and just look at the simulation tool steering the engine, it gets more event-oriented. This is because DES just "provides" the events and hand them over to the business process engine. What happens there is out of scope of the DES. And if a business process waits in a state and model time needs to be consumed, it is easier and less confusing to create an event inside the business process engine to reactivate it later, than to hand over the control flow from inside the engine to the DES framework. This matches with the statement in section 3.2.2 on page 28, that the process-oriented approach gets event-oriented, if you "image the reactivation of processes in as events". This is exactly what should happen in BPS steered by DES in my opinion.

3.3 Statistics

3.3.1 Random numbers

Discrete Event Simulation runs should include some randomness, actually this is one of the main reasons to use simulation instead of mathematical optimization. This randomness is implemented in simulation models by stochastic components, which often rely on statistical distributions in the background.

Randomness is part of reality. How many orders are placed at which times by customers is random, same when throwing a dice. In computer systems randomness is a problem, because computers can only calculate. True randomness can only be achieved by special hardware. But this is not necessary in most of the cases, because

there are so called *pseudo random numbers*, which are calculated by special algorithms. The main requirement for the produced random number streams is, that they "do not repeat themselves in a cycle (or at least only in very long ones)"[16].

Most random number generators begin with a starting value, called *seed*. This seed enables the user to reproduce the stream of numbers later, which is a very interesting feature for simulation, because it allows to repeat simulation runs or to compare them with different parameter settings. Please note, that it is important in simulation runs to have statistically independent number streams for different parameters[17].

There are different algorithms for implementing pseudo random number generators, which are discussed extensively in literature[18]. For this thesis, I rely on the choice done by the used simulation framework: The Java Random Number Generator[19]. So it is enough to know the basic concept of pseudo random numbers and the meaning of seeds in this context.

3.3.2 Theoretical Distributions

Let's assume you don't have any historical data available, for example from application log files, it is obvious you have to calculate the randomness as described. But if data is available, why not use it directly as an input to the simulation? The problem with these so called *empirical distributions* is, that they only reflect the systems' behavior within some period of time[20], thus not providing all possible values. Translating these values into a *theoretical distribution* (see section 3.3.2) has some advantages:

- The handling of the distribution gets simpler because theoretical distributions often only have two or three parameters.
- Coverage of the whole range of possible values, not only some observations.
- The translation itself requires some thoughts about which type of distribution to use. This makes the distribution more general and more valuable.

Internally, the value streams for a theoretical statistical distribution function are calculated by a given formula from the generated random numbers, which generates numbers between 0.0 and 1.0. This section now gives a rough introduction to some important distributions and advice which of their usage in certain situations. All figures are taken from `http://en.wikipedia.org/`.

The *constant distribution* is a pseudo distribution which always returns the same value. This is only useful for tests where you may want to predict the outcome. For example in automated unit tests randomness is not desired. Another simple distribution is the *uniform distribution*, where all possible values are equally probable. Figure 3.3 on the following page shows the so called probability density function (pdf) for the uniform distribution graphically, which is quite intuitive to understand. Simplified you

[16] [PK05], p. 161
[17] [PK05], p. 161
[18] for example [PK05], p.161 ff.
[19] see `http://java.sun.com/javase/6/docs/api/java/util/Random.html`
[20] [PK05], p. 170

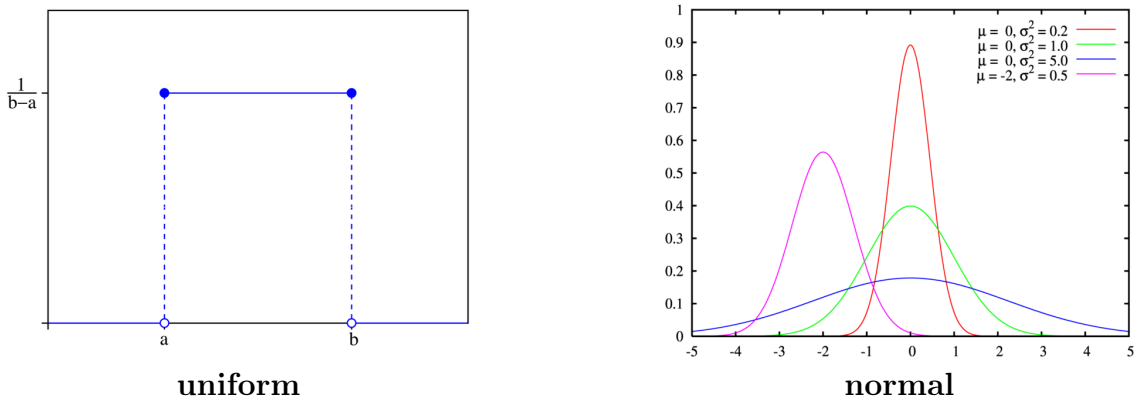


Figure 3.3: Uniform and normal distribution.

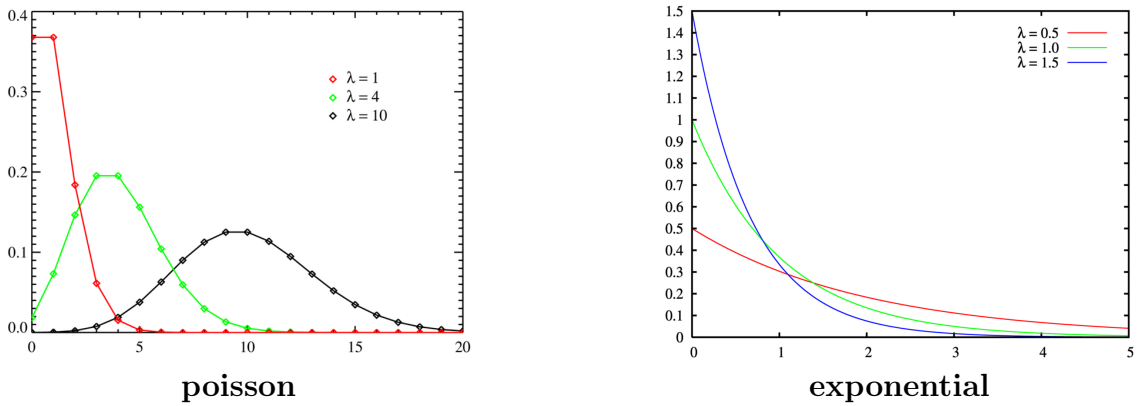


Figure 3.4: Poisson and exponential distribution.

can say that this graph shows the probability of the possible values, which can be calculated with integral calculus.

The *normal distribution*, also called Gaussian distribution after the German mathematician or bell curve from the shape of the pdf, is one of the most important statistical distributions. Very important in that context is the central limit theorem, which states that large numbers of indicators (for instance sum or mean) from random variables are approximately normally distributed. This also means that if a large number of independent effects acting additively, the observations should be normally distributed. A lot of biological or chemical phenomenas follow this distribution.

"The *Poisson distribution* is a discrete probability distribution that expresses the probability of a number of events occurring in a fixed period of time if these events occur with a known average rate, and are independent of the time since the last event"[21]. The Poisson distribution is used for a lot of different applications and fits well into most of the problems we have in business process simulation, for instance working time or the occurrence of start events. But note, that it only gives you the number of events per time interval.

[21] http://en.wikipedia.org/wiki/Poisson_distribution

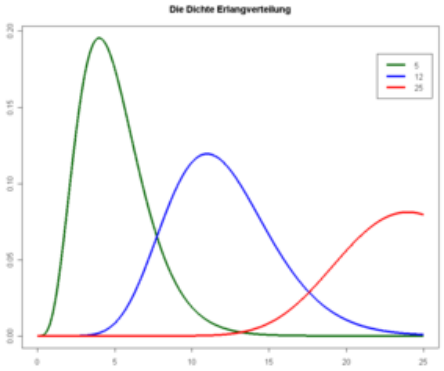


Figure 3.5: The Erlang distribution.

The *exponential distribution* is used to model the time between two events, not the number of events in some interval like the Poisson distribution. As a precondition, the events have to be independent and happen at a constant average rate. The exponential distribution is often used in queuing theory for inter-arrival times[22], which means it is well suited for simulating the time between start or work finish events in business process simulation. The *Erlang distribution* is related to the exponential one and can model waiting times between independent events too. The usage for business process simulation is the same.

Which distribution to use must be decided on a per-case basis, mostly by using a statistical software. As a rule of thumb the Erlang or exponential distribution are most important for business process simulation.

Estimate input distributions

The estimation of a theoretical distribution out of historical values is not a simple task, which normally involves some special statistical software like SPSS[23] or the freeware tool GSTAT2[24]. The theoretical background is described for example in [Bou02]. More information on distributions, finding the right one and validating it, are available in literature[25].

Unfortunately, there is no open source Java component available which implements this task out of the box. Because it is not the main focus of this thesis, I didn't develop it myself either, so this feature was skipped in the developed simulation tool. If real historical values should be used without a manual translation into a theoretical distribution, they can be used only as empirical distribution.

[22] compare to http://en.wikipedia.org/wiki/Exponential_distribution
[23] http://www.spss.com/
[24] http://www.statoek.wiso.uni-goettingen.de/user/fred/
[25] for example [GW04], [BFS87], [LK00] or [Kre79]

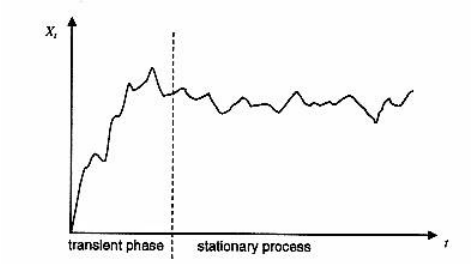


Figure 3.6: The warm-up period of a simulation.

3.3.3 Warm-up phases and steady states

In the beginning of a simulation experiment model most variables have biased values. For example all queues may be empty at the beginning, resulting in unrealistic good process cycle times for the very first process instances. The whole systems need some time for warming up before meaningful observations can be made. This is shown exemplary in figure 3.6[26]. This phase is called *warm-up period* or *initial transient phase*, the time after it is called *stationary phase* or *steady state*. [PK05][27] defined the stationary phase like this:

> We will refer to a process $\{X_t|t \in \tau\}$ as stationary if the probability of observation X_t does not depend on the value of t (time). Many stochastic processes converge to a stationary process over time.

The big problem now is to identify the end of the warm-up period. At this point of time we want to reset all statistical recorders, so no observations from the initial transient period are included in any statistical analysis[28]. For detecting the warm-up period many more or less complicated methods are discussed in literature, they can be grouped into five categories[29]:

- Graphical methods: Involve visual inspection of the time-series output and human judgment.
- Heuristic approaches: Provide rules for determining when initialisation bias has been removed.
- Statistical methods: Based upon statistical principles.
- Initialisation bias tests: Test if the warm-up period has been successfully deleted in the statistical measures. Therefore a null hypothesis (H0) is created and tested stating that "no initialisation bias is present in a series of data".
- Hybrid methods: Combine initialisation bias tests with graphical or heuristic methods to determine warm-up period.

[26] from [PK05], p. 187
[27] p.174
[28] [PK05], p.174
[29] from [Hoa07], compare to [PK05], p. 175

For example, one possible relatively easy heuristic approach for detecting the end of the initial transient is *Conway's Rule*:

> When using a single run, Conway (1963) suggests the following rule for dropping observations: discard initial observations until the first one left is neither the maximum nor the minimum or the remaining observations[30].

The procedure is very easy and starts with examining observations, which can be single observations or the mean of multiple observations. If they are a minimum or maximum for the rest of the observations they belong to the initial transient, otherwise you test the next observation. The approach is discussed controversial in literature, but it is still better than nothing. "Though there is little theoretical support for this, no obviously better rule has been suggested since Conway's paper"[31]. [Hoa07] quotes a couple of authors with critics on this approach.

For the simulation tool developed in this thesis it would be nice to have an automatic warm-up detection. Unfortunately this is neither included in the used simulation tool nor available as out of the box. And some simulation processes may never reach a steady state. Currently there is a lot of up-to-date research still going on on this topic but the most of the published articles were not available to me. All together, this made it impossible to deal with the warm-up problem in more depth in this thesis.

The automatic warm-up period detection is skipped. What can be done is to examine the line plot of the simulations results to get an idea at which point in simulation time the steady-state begins. This is also a well known approach[32]. After figuring out that point in time, it can be configured into the simulation tool, which will reset all statistical counters at this model time. This approach doesn't need complex tool support and is the only supported one in the developed simulation by now.

3.3.4 Analyzing simulation results

Because of the stochastic input to simulation models, the output of simulation must be treated as random samples too[33], because different simulation runs can yield to different results and none of the runs can really predict future. Hence measures like mean or standard derivation are of particular interest and statistical analysis of simulation output is necessary to draw valid conclusions about the real behavior of the process under simulation.

An important issue in this area is the sample size, because valid conclusions cannot be drawn from a single simulation run with arbitrary length[34]. To obtain information about the right sample size you have to distinguish two different types of simulation[35]:

Nonterminating: The most business processes are nonterminating, that means the processes don't end within some defined practical time horizon. As long as a web

[30] [BFS87], p. 94
[31] [BFS87], p. 94
[32] [LML04], p.352
[33] compare to [PK05], p.179 or [LML04], p.349
[34] [LML04], p.349
[35] [LML04], p.351 and [PK05], p. 188

sale company exists for example, it fulfills orders. The simulation is nonterminating, because the end condition of one day, shift or something like that, is the start condition for the next one. If breaks would be skipped, the result is one large never terminating simulation run. Most nonterminating systems reach a steady state like described in section 3.3.3, so the initial start condition is unimportant from this time on. To improve significance of statistical measures, simulation run length can be extended. This means, sample size can correspond to simulation run length in this case.

Terminating: A terminating simulation starts in some empty state and ends in an empty state. The termination happens either after some amount of time or is controlled by an event. These processes often don't reach a steady state, but even if they do, it is less interesting, because the initial transient phase is part of the models real behavior and has to be included in the statistical analysis. The length of the simulation run is predefined by the termination event, hence it cannot be extended to improve statistical results. The only possibility to get more reliable statistical figures is to execute multiple simulation runs. So the sample sizes corresponds to number of runs in this case.

Because simulation can only use limited sample size, calculated statistics like the mean are only estimates to the true but unknown statistics in reality. So it is not enough to state these values as results. Statistic knows the concept of *confidence intervals* for this problem, so the results are intervals rather than single values. It is never sure if the true value is really within the interval, but it can be expressed with a defined confidence level, which gives information about the probability, that the real value is within the confidence interval. There are two main factors influencing the width of the confidence interval[36]: sample size and standard derivation of the measure. The larger the sample size or the smaller the standard derivation is, the narrower the confidence interval. Normally the business analyst specifies the required confidence level and confidence interval width first and calculates the required sample size for that afterwords.

For computing confidence intervals, the *central limit theorem* is very important. It states that the mean of different simulation runs is normal distributed if the sample size is large enough[37]. A rule of thumb for the sample size is typically 30[38]. This theorem allows to calculate the confidence interval from known sample values. Details of the derivation can be found in many statistic textbooks or for example in [LML04], p. 357ff. For the calculation the "Inverse Cumulative Standard Normal Distribution Function (NORMSINV)"[39] is needed which I will not explain any further here.

$$\text{population mean} = \text{sample mean} \pm \frac{Z * \text{sample standard derivation}}{\sqrt{\text{sample size}}} \tag{3.1}$$

$$Z = NORMSINV(1 - \frac{1 - \text{confidence level}}{2}) \tag{3.2}$$

[36] [LML04], p.357
[37] for example [GW04], p.205
[38] see [PK05], p. 183 and [LML04], p.357
[39] [Ack04]

Equations 3.1 and 3.2 show the calculation of the confidence interval, which is symmetric around the sample mean. The random variable Z is calculated with the NOMSINV function and relates to the required confidence level. The provided formulas are only valid if the sample size is bigger than 30, because otherwise the central limit theorem is not valid. In that case the "student t distribution" should be used instead of the normal distribution to calculate Z. This produces wider confidence intervals which can balance the statistical shortcomings of the low number of samples. The student t calculation is skipped in this thesis but described in more detail in [LML04], p. 358.

The confidence level has to be specified by the business analyst and should reflect the risk a wrong decision can bear. Specifying the level is not a mathematical but a business problem.

As mentioned above, confidence intervals can be used to calculate required sample sizes. A business analyst can specify the necessary confidence level and provide an exemplary mean and standard derivation from a former simulation run or historical data and the needed sample size can be calculated. The last part is interesting: You cannot calculate the sample size without running the simulation model once to have statistical data available[40]. The formula is shown in equation 3.3.

$$\text{required sample size} = \left(\frac{Z * \text{sample standard derivation}}{\text{half confidence interval width}} \right)^2 \tag{3.3}$$

The last question is how to get independent samples for simulation results like means. They can be retrieved by two different techniques: Independent replications or batch means.

The method of *independent replications* requires multiple simulation runs for each model parameter constellation[41]. The difference between the runs may only be the seed used by the random number generators. The method of *batch means* requires one long simulation run. It is split into different parts considered to be independent. The longer the simulation run the more realistic is this assumption[42]. The initial transient should be truncated in both cases for nonterminating processes, the first method can be applied to terminating processes too.

The formulas presented in this section are included in the simulation tool developed later in this thesis, so sample size calculation could be automated in later simulation projects if the required input parameters are specified.

3.4 Business Process Simulation (BPS) in practice

3.4.1 Status Quo and challenges

As already seen in this thesis, BPS has a big business value: Simulation results allow to optimize processes before they go into production. This leads to better performance, less resources required for execution and a significant risk reduction. But despite this

[40] [LML04], p. 361
[41] [PK05], p.185
[42] [PK05], p. 186

fact it is not very popular today. In a survey from the university of Munic and Erlangen-Nürnberg from 1996[43] it is stated, that business process simulation is the less important application for simulation, only three percent of the simulation projects where about BPS. If you take into account, that only 65 % of the interviewed users use simulation somehow, this is a quite small figure. Even, if that survey is more than ten years old, things seem to change slowly. Unfortunately there is no up-to-date survey about the topic, but some evidence also comes from [PH03]:

> Most business process modeling tools have simulation capabilities [...]. Talks with vendors, however, suggest that simulation is used by, at most, five percent of their clients.

This has several reasons. Simulation is not easy to apply, you have to specify very detailed what can happen with which probability to get useful results. Another reason is the lack of tools, not simulation tools at all but practical useful tools for BPS. And the simulation capability of most of the business process modeling tools are poor in terms of usability, required features and primarily integration throughout the BPM life-cycle. Section 3.4.2 describes different types of tools and evaluation criteria for this problem.

Looking at good BPS tools, I see two problems. First, they are very expensive. Not especially the simulation engine, but the whole product suite. Sometimes expensive middleware is additionally required. This disqualifies whole customer segments for applying BPS. Another problem is missing transparency and documentation. You can get the impression, that several vendors are used to closed source and somehow closed knowledge. For example there is no step-by-step tutorial for BPS, showing the business value, the concrete implementation and the result on a real life showcase. At least, this is not freely accessible, for instance via Internet. Accessible are only high level presentations, whose value is at least questionable.

By requiring expensive tools and a specialized consultant to understand all basics, simulation projects gets even more expensive. Without a good motivation, customers don't ask for BPS. This is somehow very sad, because it can really boost the value of BPM, especially if integrated in the BPM life-cycle. The important challenge today is in my opinion to bring BPS to a broad range of companies. In the best case it should come with the introduction of BPM, with no mentionable additional effort and integrated smoothly in the BPM life-cycle. As a vision, BPS automatically finds possible improvements of processes during normal execution and recommends them pro active to the business analyst.

3.4.2 Categories of tools and evaluation criteria

Applicable tools for BPS can be grouped into three categories[44]:

Business Process Modeling tools: Pure modeling tools are "developed to describe and analyze business processes"[45] and therefor can capture a lot of information

[43] [iuF97]
[44] [JVN06], p. 4
[45] [JVN06], p. 4

from control flow to involved resources. If the resulting models can be enriched with statistical inputs for simulation, this is a valid BPS solution. The problem is, that these tools often are isolated and not integrated in the BPM life-cycle. The process model is often for documentation only and too simple, the statistical inputs are provided manually and no historical data can be used. This lowers the value of the simulation results. In [BVCH07] this group of tools is not even mentioned, which can be some indicator of the practical value of it. But it should be noticed that some of the tools, for example ARIS[46], integrate with third-party tools to take part in the BPM life-cycle. So there is no strict border line between this group and the next, which are

Business Process Management tools: These tools care about the whole BPM life-cycle, Gartner introduced the name *Business Process Management Suites (BPMS)* for them in 2005[47]. The main features are the execution of process models with some business process engine, which also makes it possible to automate parts of the process. During execution, the engine can write sophisticated audit logs, which gives good quality data about the processes. This data is a good candidate to use as input for simulation. Also the modeled processes for the engine must be formal enough to be executed, so they are a solid basis for simulation too.

General purpose simulation tools: These tools can support each and every kind of simulation, even if a lot of these tools are tied to some specific domain. Some of them can be used to simulate business process as well. The problem is that you have to learn the specific tool or language and provide a special model only for simulation, which has nothing to do with the modeled business process from BPM. This makes it more complicated and almost impossible to be integrated in the BPM life-cycle. But nevertheless it is sometimes a good idea to use such a tool, especially when designing completely new processes and if you want to get some high level understanding of special aspects.

This thesis focuses only on the second group of tools because the integration of BPS in the BPM life-cycle is one main goal, which is only possible with true BPMS tools. Evaluation criteria for BPS tools can be split into three groups[48]: modeling, simulation and output analysis. Modeling requirements are essential to every BPM tool and should not be examined further here, but the other two groups are worth to look at in more details.

The simulation evaluation criteria of BPS tools are[49]:

- The support of different *performance dimensions*, like time or cost.
- The possibility to provide input *distributions*, so not only averages but also on extremities are included.
- *Animation*, like a replay of simulation runs, may provide a better understanding of the process or reveal bottlenecks.

[46] http://www.ids-scheer.de/aris
[47] [Gar06b]
[48] [JVN06], p. 8
[49] [JVN06], p. 9

- The support of *scenarios* comparing different parameter sets, like arrival patterns of events, to prepare processes for new situations in reality.

Output analysis capabilities are important to understand the simulation results. Evaluation criteria are[50]:

- *Statistics*: Simulation results should be interpreted with proven statistical methods and should not only show means, but also standard derivation and confidence intervals for example. Simulation settings, like simulation length, number of repetitions or ramp up period must be indicated by the tool, or even better adjustable by the user.
- *Reports* are essential to get a nice view on the results.
- *What-if analysis* enables the analyst to compare provided scenarios and find the best solution. To do this seriously, the scenarios must be simulated within the same simulation settings.
- *Conclusion-making support* helps to interpret simulation results by showing trends or provide slicing and dicing features to examine the results[51].

3.4.3 Market overview

After the groups and evaluation criteria I want to provide some information about available BPS tools. In this area, the market is changing almost every day. Additionally it is very intransparent, especially because vendors don't classify their products after the groups mentioned above and just mark their existing BPM or infrastructure tools with a "simulation enabled" label. Some tools provide for example only an animation of *one* process run and call it simulation, even if almost none of the above presented capabilities is present.

There is no list of tools included into the thesis because of the limited value resulting from too fast changes in market. The Magic Quadrant for Business Process Management Suites[52] shown in figure 3.7 is a good starting point to find BPS tools because simulation is one of the requirements from Gartner to be a BPMS. But there are a lot of tools available, not (yet) mentioned in this Magic Quadrant, because they don't meet all Gartner requirements. Unfortunately I found no real up-to-date study on BPS tools, so the main sources for tool candidates could be the Magic Qudrant for Business Process Analysis Tools[53], the tool list on www.bpm-netzwerk.de[54], the BARC study on process modeling tools[55] and [Mac05].

[50] [JVN06], p. 10

[51] Slicing and dicing are concepts from data warehousing, meaning zooming in our out details of data.

[52] see [Gar06b] available online for example here: http://www.5acts.com.br/br/download/pdf/GartnerMQ_BPMS_2006.pdf

[53] see [Gar06a] available online for example here: http://www.ncsportugal.com/docs/gartner_Magic%20Quadrant_2006.pdf

[54] http://www.bpm-netzwerk.de/content/software/listSoftware.do, accessed on 26.09.2007

[55] [BEFF06]

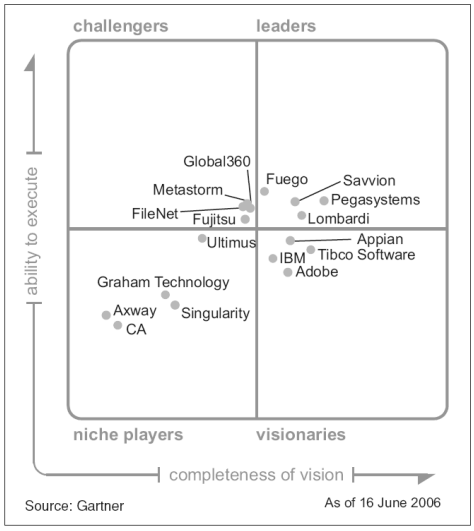


Figure 3.7: Magic Quadrant for BPMSs, 2006, from [Gar06b].

Tool candidates should be rated according to the following four evaluation criteria: Animation, Stochastic Simulation, Statistical Reporting and log data as input. This help to judge the existence and power of simulation capabilities in different tools. For a finer grained evaluation you should apply the evaluation criteria presented in the last section. According to an evaluation in the run-up to this thesis, no single open source BPS tool is available today.

The vision of Business Process Simulation and its role in the BPM life-cycle is convincing. But a market overview shows that the tool support is still far away from being optimal, even if some vendors already have good tools and the most important ones share the right vision. Open Source solutions are completely missing in this area, making BPS dependent on expensive tool stacks.

3.4.4 The human factor and BPS

The BPM life-cycle describes a vision which uses simulation to try out new processes or changes before going into production. One limitation of this concept should be mentioned here, the "human factor". Especially if you are dealing with new developed processes this factor can have a huge influence on your real cycle times or resource requirements.

The human factor can be for example:

Motivation or willingness: If you introduce an IT system which automates a process which was done manually before, you can face fear or refusal of the people who have to work with the system. This decreases the motivation of people, in the worst case they may even sabotage it.

Incomplete process: The automated business process never captures 100 per cent of all cases, which is indeed not the goal. But the less automation, the less the

quality of the results. And if big parts of the process still exists in the head of the people, the simulation results are not really usable.

Incorrect process: Even worse than missing parts are modeled and executed processes, which don't reflect reality correctly. The people have to create workarounds or apply known tricks to work with the system.

Some of the effects of the human factors described above are taken into account when querying statistical distribution of historical data, because these effects were present in history. They may play a smaller role if you only want to simulate different parameters for a known process, but bigger when simulating new processes. Obviously the described problems are present especially in human centric process. The more automated the process is, the smaller are the human factor problems. Because these phenomenas can never be eliminated, it is important that the business analyst is aware of them.

3.5 Simulation-based optimization of business process performance

As already described, simulation can only check provided scenarios as demonstrated in the showcase in section 4.4. The goal of a simulation project is to find the best parameter settings for a given business process, an example is the number of resources. The task is to find optimal values for these variables, which is part of the decision process a business analyst has to make, e.g. to define shift plans and staffing strategies. But simulation can only answer "what if" questions, and not "how to"[56].

As you can see in the showcase, finding well suited parameters can be done by a very simple procedure[57]: Manual guessing or brute force. The latter is implemented by testing all possible input parameter combinations within some easyly definable boundaries. Obviously, the number of scenarios which have to be simulated increases very fast with a growing number of input variables. Since simulation runs are computing intensive and time consuming, this approach doesn't work for more complicated simulation projects. [LML04] expresses it this way:

> When processes are complex and the configuration depends on a number of strategic choices, the trial-and-error approach can be applied with only limited success.[58]

Hence a more sophisticated approach has to be applied. For this problem *optimization technologies* can be used. They are able to generate scenarios in such a way, that only a fractional amount of the possible scenarios has to be tested, but the probability

[56] [PK05], p. 387
[57] see also section 4.4.4
[58] [LML04], p. 375

to find an optimal or nearly optimal solution is high[59]. This is done by heuristic approaches, since a mathematical optimization is not possible or too complex in most of the cases[60].

From the optimization point of view, scenarios are "candidate solutions which are searched for the most favorable behavior for reaching a given goal, i.e. optimizing an appropriate objective function"[61]. An example is finding the best staffing strategy for a known process. The evaluation of the scenarios, means computing the objective function result, is done by the simulation when combining both technologies, even if it still needs to be defined which resulting figure should be included into the objective function. The *objective function* should produce only an absolute rating for the whole scenario, so it is ensured that the comparison of two alternatives always leads to a clear decision which one is better. An example can be the costs caused by a scenario. Not all candidates are possible in reality, so unfeasible solutions have to be recognized. The focus of optimization algorithms is on producing good candidates. This is done by making good decisions about candidates to include and candidates to ignore and strategies how to create new candidates. The simulation run can be seen as a black box for the optimization.

Because the optimization algorithm cannot use information about the inside of the objective function, only *direct optimizations* can be used[62]. One very prominent example are *genetic algorithms* or *evolutionary algorithms*, which are used in that area.

> An evolutionary algorithm is an iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the genome, encoding a possible solution in a given problem space.[63]

The procedure of such an algorithm combined with simulation is visualized by pseudo-code in listing 3.1 on the next page[64]. It involves genetic operations for selection, recombination and mutation. Population size has to remain constant over time to filter out bad performing individuals. Genetic operations include random strategies but clearly favor better individuals over worse ones, so the probability that bad solutions slightly disappear from the populations is high. This approach "offers a high probability of reaching promising regions of the solutions space, thus being random-based rather than random"[65]. By the way, unfeasible solutions can be included while searching the solution space, because they can lead to new regions which wouldn't be discovered otherwise. This decision depends on the concrete implementation of the genetic algorithm and is not discussed further here.

The optimization algorithm is used for generating different scenarios for defined input parameters. This approach is not too complicated and already discussed in literature, for example [PK05] or [LML04]. A prototypical implementation is done for

[59] see [LML04], p. 376 or [PK05], p. 388
[60] see [PK05], p. 388
[61] [PK05], p. 388
[62] [PK05], p.394
[63] [LML04], p. 376
[64] after [PK05], p. 396
[65] [PK05], p. 396

Listing 3.1: Pseudo-code for a genetic algorithm.

```
Population p = createRandomInitialPopulation();

// run simulation and compute resulting performance indicator
// for every individual
runSimulation( selected );
evaluate( selected );

while (!terminationConditionFulfilled) {
  Population selected = p.getBest();

  // form 2 new individuals from 2 already existing ones
  selected.recombine();
  // mutate selected single individuals
  selected.mutate();

  runSimulation( selected );
  evaluate( selected );

  // remove worst of old and add best of new population
  p.removeWorst();
  p.add( selected.getBest() );

  checkTerminationCondition();
}
```

instance by the university of hamburg in the DISMO framework[66]. It is very promising and should be included in the developed simulation tool in future. Actually it will be needed for the improvement of the prototype explained in section 5.

Genetic permutation of business processes

Another idea to use optimization is not to optimize input parameters but the processes itself. This idea is therefore very interesting, because it would lead toward self optimizing business process engines, or at least to BPM tools which can automatically recommend process improvements, for example in their BAM components. Unfortunately there is not much up-to-date research material for this topic, except for [Bau99]. Because of the limited time and the need to build a basic open source simulation tool first, it wasn't possible to address this issue further in this thesis. So I just want to give some of my basic ideas here.

The problem when using genetic algorithms to improve business processes is the encoding and especially building feasible candidates by genetic operations. To give a better impression what mutating business processes itself can mean, here are some examples: Changing the order of nodes, parallelizing the process flow, eliminating duplicate work or maybe combining to nodes into one. Obviously the most generated solutions will be unfeasible in reality, because there are a lot of constraints not modeled in the process. A good example is the flow of objects in reality, for example the

[66] [GP01]

transport of goods to another workstation. Also the genetic algorithm can hardly know which steps of a process are how important. What is missing are additional constraints for the solution space of possible candidates.

One idea to get some of them is to analyze running processes or audit log data. By doing so it can be figured out which process variables are changed in which nodes. This information can give some hints on a required order of the nodes. If for example two nodes work only on different data it can be guessed that the nodes can be parallelized. Obviously, this doesn't have to be true. There can be still constraints on the order, because data is changed offline on papers or in a foreign system for instance. But it can serve as starting point.

So the big problem is to discover and manage the mentioned constraints. One possible approach could be to use a rule engine to manage them[67]. The idea is to discover possible mutations of business processes and present them to the user. The business analyst can mark some of them as unfeasible by adding constraints. By doing so, the process model can be enriched with additional requirements, not modeled directly in the process model. Imagining a good graphical user interface, these constraints might even be managed during model time or additional attributes can be introduced.

All in all, this idea is far from being usable in production environments, but I expect it to be a feature of BPM tools in future. If included seamlessly in the BPM lifecycle presented earlier, these optimizations may help to gain a better understanding of processes, discover weaknesses or hot spots of the currently deployed processes and lead the business analyst to create a more realistic model, because additional constraints have to be included.

[67] Rules engines are not explained here, see for example [Ros03]

Chapter 4

Implementation of the open source BPS tool

4.1 Used software components

4.1.1 The open source business process engine JBoss jBPM

JBoss jBPM[1] is a very flexible business process engine which is available under the open source LGPL license[2]. In autumn 2004 the project around Tom Baeyens joined the open source company JBoss, which meanwhile got acquired by Red Hat. Because of this there are commercial services available, like support and training. Core developers of jBPM are employed by Red Hat and a future roadmap for the product exists. Currently it gets integrated with other products of the JBoss SOA stack, like the ESB or Rule Engine. Many customers, even big companies, start developing business process projects on JBoss jBPM. Additionally jBPM has a very vital and alive community, solving a lot of problems within the web forums. All in all, this qualifies JBoss jBPM even for mission critical software projects. This is the main reason I choose jBPM as a basis to develop an open source business process simulation tool on top.

The latest version of the engine at the time of writing this thesis was 3.2.2. Even if the development of jBPM 4 is already quite far, the simulation environment was build on the current version, because it is stable and a case study with some real life project was possible.

Internally, jBPM can be seen as simple state machine, but with persistence and additional services. You can imagine it as "Virtual machine for processes", in fact the core of the upcoming version is called Process Virtual Machine (PVM)[3]. jBPM can read process definitions written in the proprietary jPDL or standardized languages like BPEL or XPDL[4], start process instances and controls their state for their whole run time. The engine persists this state on wait states. Another important feature is the process context. There you can add variables to process instances, which are automatically persisted together with the process itself. Variables can be everything

[1] http://labs.jboss.com/jbossjbpm/
[2] http://www.gnu.org/licenses/lgpl.html
[3] [BF07]
[4] XPDL will be introduced with jBPM 4

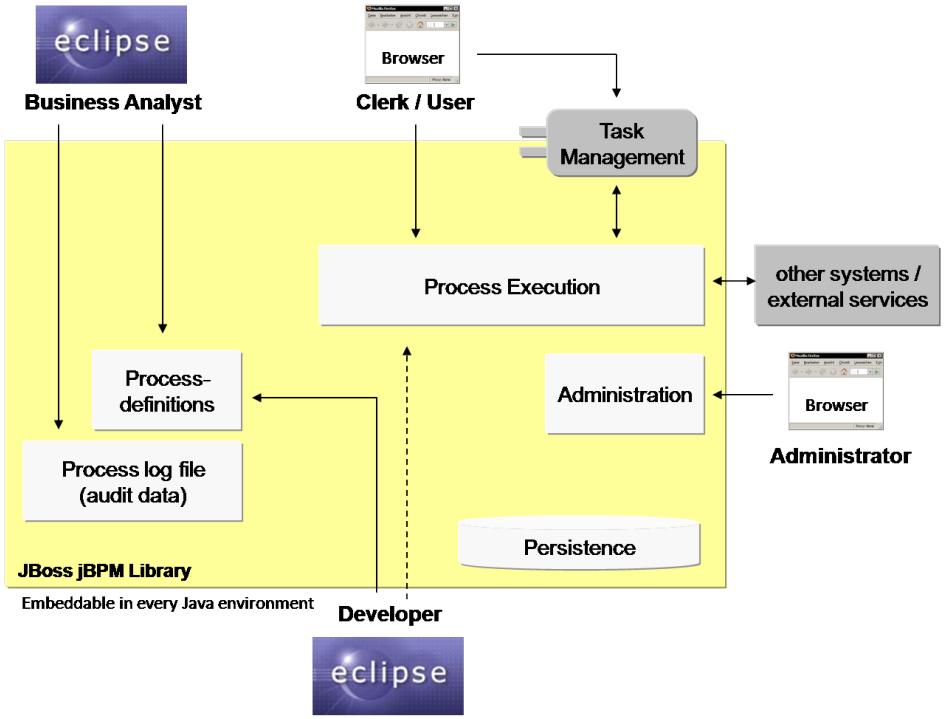


Figure 4.1: Overview of JBoss jBPM.

supported by Java, like plain objects or even EJB entity beans.

In this thesis I will concentrate on jPDL. There business logic or calling services can be included in the processes by Java code. In fact, you have to write simple Java classes which implement a special interface. These so called actions can be added at the desired places in the process definition. This approach supports a separation of concerns: The business process definition is defined in XML and separated from technical issues written in Java code.

The process engine itself is implemented with Plain Old Java Objects (POJO), so it is "just" a simple Java library. This enables it to be included in any type of environment, for example standalone applications based on Java Standard Edition, web applications or complete Java Enterprise solutions. Persistence is realized via the well known open source object-relational mapper Hibernate[5], so any database supported by Hibernate can be used. This architecture is completely different from heavyweight business process engines, which have to run as a server. A schematical overview is illustrated in figure 4.1.

jBPM Process Definition Language (jPDL)

The jBPM Process Definition Language (jPDL) is the proprietary process definition language used in this thesis. The so called *nodes* are the most important part of this language. There are different types of them which behave differently during runtime:

Node: A node itself doesn't have any behavior, the execution just passes through

[5]http://www.hibernate.org

it. But it is the base class of all other node types and can be a basis for own implementations or a point to hook in usercode.

Task-Node: Task-Nodes represent human tasks. Internally the engine creates a task in the task management component of jBPM, which can be queried from the client to build a todo list for an user. A Task-Node is also a wait state, because the process instance has to wait for task completion before it continues.

State: The State represent a typical wait state. The engine persists and waits for an external event to continue.

Start- and End-State: Obviously these nodes mark the start and the end of the process. Only one Start-State is valid, but many End-States can exist.

Decision: In this node the process engine makes an automatic decision on its own via specified logic.

Fork and Join: These constructs allow to parallelize process execution paths, used for example if different tasks are processed by different people independently in parallel. Each fork must be finished by an corresponding join.

There are other interesting elements which I only want to name here: sub processes, with the expected behavior of calling sub processes. Super states, which create some blocks inside a process. But these are not really relevant in this thesis, so the keen reader is pointed to the jBPM documentation[6] for more details.

User groups or roles are represented as so-called *swimlanes* in jPDL. A swimlane typically refers to a special role, but can also implement more advanced concepts, like figuring out the responsible role by evaluating the process context. Therefor even business rules can be used.

An important concept in jBPM 3 is the token, existing in every running process instance. It always points to the current node, the process instance is in. The concept and name is borrowed from an important theory for business process engines: Petri nets[7]. Every process instance has exactly one root token, which can have child tokens if forks are involved.

Finally there are transitions, which connect the nodes to a process graph. So the token can "travel" from a node to another via the specified transitions. This is how the process flow is implemented internally. An example for a jPDL process is skipped here, but included in appendix A.3.1.

4.1.2 The open source simulation framework DESMO-J

For the business process simulation tool developed in this thesis, I use an existing framework for the simulation aspects. It is never a good idea to reinvent the wheel and luckily there is a simulation framework available, which meets the two main criteria: It is open source *and* "alive", which means that there is good documentation available and

[6] [JBo07]
[7] for example [Hav05], p. 55 ff.

it is maintained and improved further on. The framework is called DESMO-J (Discrete Event Simulation and Modeling in Java)[8] and is developed from the University of Hamburg, which is involved in development of simulation software in various languages for many years[9]. The Java based framework was first finished in 1999 and is going to be maintained and improved further[10]. Most recently there is some effort in developing a new Eclipse[11] based graphical user interface[12], so the maintenance plans seem to be serious.

The whole framework is available under the Apache License, Version 2.0[13]. This allows you to use the framework in any product you want, open or closed source, free or commercial, as long as the copyright and disclaimer is preserved. So the license permits to use DESMO-J as part of the planned simulation environment.

DESMO-J was chosen because it is the only existing Java simulation framework I found, which meets all the following requirements:

- Available under a liberal open source license (like LGPL): This is important to include the developed simulation environment later in the JBoss jBPM project, which has a LGPL license itself.
- Maintenance: The tool has to be maintained actively. A big number of users help to achieve the maintenance as well as continuously improvement.
- Solid scientific basis: Discrete-Event-Simulation tools should reflect the current status of research.
- Good documentation: A good documentation is vital to use the framework correctly and bring out the best of it.
- Java: The framework must be implemented in Java to allow a good integration with JBoss jBPM.
- Support required features: The simulation framework must support the required features for BPS of course.

Most of the available tools are commercial. The few open source ones normally have a poor scientific background or an inactive user community[14]. So DESMO-J was identified to be the only tool matching all requirements. But since it is a very satisfying choice, this is not a problem at all.

[8] see http://www.desmoj.de
[9] [PK05], p. I
[10] [PK05], p. II
[11] http://www.eclipse.org
[12] http://asi-www.informatik.uni-hamburg.de/desmoj/eclipse/index.html
[13] see http://www.apache.org/licenses/LICENSE-2.0
[14] for example the Feynman Java Simulation Framework: http://feynman.sourceforge.net/

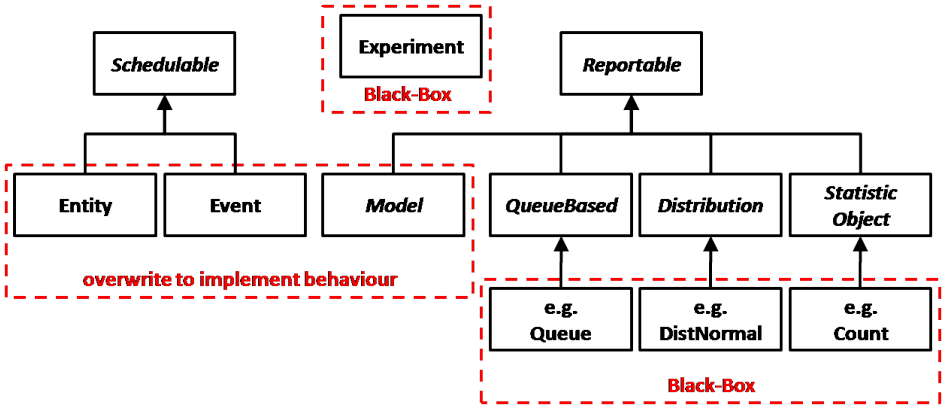


Figure 4.2: DESMO-J types overview.

Concepts and main classes

I want to take a short look at the inside and most important concepts of DESMO-J. It tries to be a black box wherever possible, but is a white box where it is needed to allow enough flexibility for simulation tasks.

The required components of a DES framework are described in section 3.2.1 on page 26. These components are all present in DESMO-J, the DES concepts are mapped directly on Java classes. The important classes can be divided into black and white-box. The white-box ones are classes the user has to extend in order to implement the desired simulation behavior, basically "Model", "Entity" and "Event". The important configurations are wired together in the model class: Properties for queues, distributions and data collectors. The black-box part of DESMO-J, like scheduler, event list and model clock, is encapsulated by the "Experiment" class. Figure 4.2 shows the basic DESMO-J class library.

As mentioned in section 3.2.2, we use the event-oriented modeling style for the Business Process Simulaion tool. The basic idea how to use DESMO-J in the event-oriented world view is now:

- Implement subclasses of "Entity", these can have own attributes,
- Implement subclasses of "Event" with the desired behavior,
- Entities and Events can schedule themselves (or other Events) easily within an "Experiment".

Own code has to be implemented subclasses by overwriting methods.

I will describe how this works in more detail and in combination with JBoss jBPM-later in this chapter. More details on DESMO-J in general can be found in [PK05], p. 268 ff..

4.1.3 Dependencies and versions

Beside JBoss jBPM, DESMO-J and their own dependencies, the developed BPS tool doesn't need much more. The used version of jBPM is 3.2.x. There is no concrete

version number mentioned, because the development took place on the HEAD revision of the version control, that means it was always the latest, not even released version of jBPM 3. Unfortunately there is a policy that jBPM 3 has to run under Java 1.4, so the simulation tool could not use a newer version of Java, too. The used version of DESMO-J was 2.1.1, the latest official released version at the time of writing this thesis.

All code was committed directly in the JBoss version control system, so it is available to everybody. It is included in all nightly builds and normal release cycles. One important requirement from JBoss was, to keep the simulation independent of the core libraries, so it can not affect stability or cause side effects.

For the graphical simulation reports I use JasperReports[15] in version 2.0.2, which was now added to the general jBPM dependencies. JasperReports is an open source reporting engine, which also provides the graphical editor iReport[16] for maintaining report templates.

4.2 Overview

4.2.1 Components and architecture

The solution built in this thesis is based on the already mentioned software components. My personal contribution was now to plug them together in the right way to form a Business Process Simulation tool. The components and their relationships are shown in figure 4.3 on the next page:

jBPM: It should be obvious that the jBPM library is the heart of the simulation tool. It is used as transient process engine during simulations runs. The productive jBPM instance shown in the figure is out of scope of the simulation environment and refers to a normal productive application running on jBPM. The simulation tool can connect to this jBPM instance to get information about process statistics or other historical data.

The simulation instance of jBPM runs without any persistence and doesn't need any specific environment, like a web or ejb container. However, due to the implementation of DESMO-J it is not possible to run different simulations at the same time in one virtual machine.

jBPM designer: The graphical process editor which ships with jBPM will be improved in the near future to support visual editing of the simulation configuration and it will provide tools to run simulations directly within the common Eclipse environment.

DESMO-J: As mentioned in section 4.1.2 this library realizes the simulation parts of the tool and therefor steers the process engine. It provides the model clock for the simulation and synchronizes the jBPM clock with it.

[15] http://www.jasperforge.org/jaspersoft/opensource/business_intelligence/jasperreports/

[16] http://www.jasperforge.org/jaspersoft/opensource/business_intelligence/ireport/

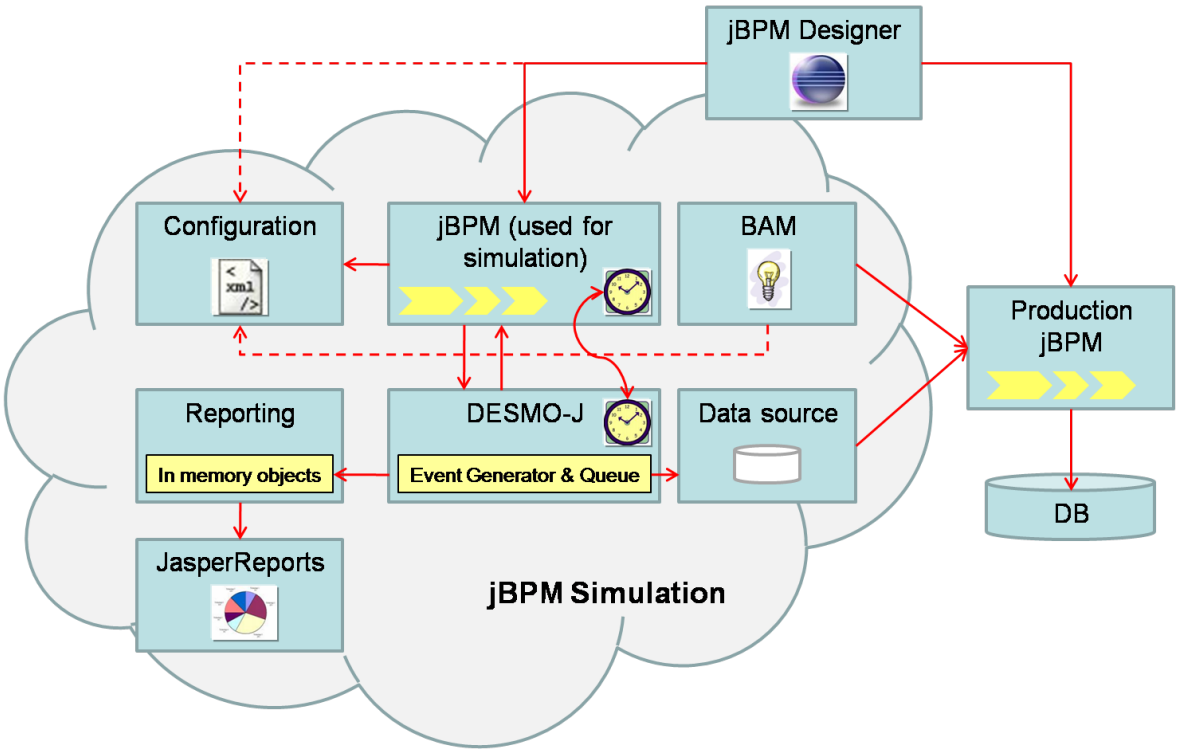


Figure 4.3: Components of simulation tool.

Configuration: The simulation configuration is provided as XML file. This is read during simulation runs and specifies all necessary parameters and scenario details.

Reporting: DESMO-J provides a lot of statistics during simulation runs. They are recorded in memory by the the simulation tool and provided afterwords as Java objects. These results can be evaluated even automatically.

JasperReports: Some graphical reports are provided out-of-the-box with the simulation tool. They can be filled with the data collected from the reporting component and show how to use the JasperReports report engine.

BAM: A real business activity monitoring tool is currently under development for jBPM[17]. Unfortunately, it was not yet ready to use for this thesis. So I provide some mini BAM component, which can read process statistics from existing jbpm log data and propose simulation configurations based on it, corresponding to the stats quo.

Data source: Depending on the simulation requirements you may need to provide data for your business processes. This means that you need process variables at some events in time, for example when starting new process instances. This data must be generated somehow, thats what I call data source in that context. One standard way could be to provide historical data from a productive jBPM.

[17] http://wiki.jboss.org/wiki/Wiki.jsp?page=JbpmBAM

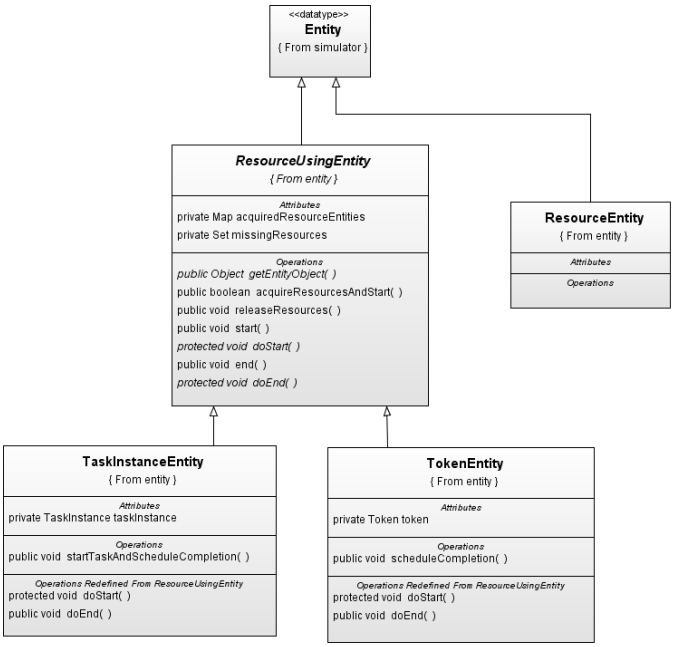


Figure 4.4: Entities of simulation tool extended from DESMO-J.

This was a brief first introduction, section 4.3 will go into more detail.

4.2.2 Using DESMO-J to control jBPM

This section explains how DESMO-J steers the process engine. For background information why it is done this way you may want to look again in section 3.2.2 or 4.1.2.

The first important concept of Discrete Events Simulation is the concept of *entities*, which are represented via the Entity class in DESMO-J. The simulation tool uses three entities which extend from the DESMO-J Entity class like shown in figure 4.4.

TaskInstanceEntity: If a human task inside a business process gets activated, a "TaskInstanceEntity" is created and handed over to the simulation framework. This tries to consume the required resources for the task, typically some human actor, and schedules the entity to be waken up later, when the task processing will be simulated to be finished. The point in future model time when the task will be finished is queried from the configured distribution for the task. This realizes the statistical simulation of the varying processing time.

If there is no resource available at the moment the entity is created, the simulation tool puts the entity in the queue for the missing resource. As soon as a resource gets available, the next entity from its queue is taken and started. In the current implementation there is the possibility to get a deadlock if the task needs more than one resource, this is a known open issue and will be handled in later versions.

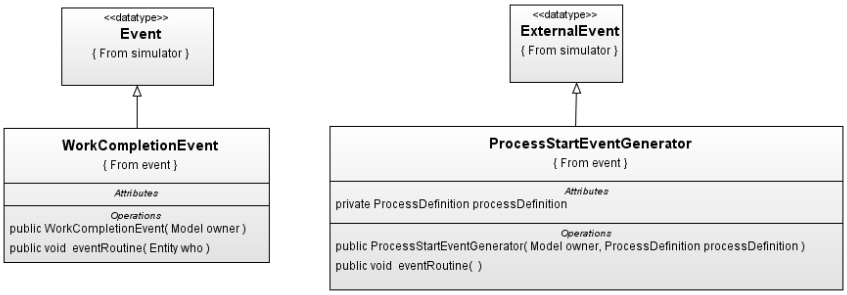


Figure 4.5: Events of simulation tool extended from DESMO-J.

TokenEntity: This serves the same purpose as the "TaskInstanceEntity" but in the case of states. Normally resources are not people but machines or the like.

ResourceEntity: This entity realizes the same idea described above with the "TaskInstanceEntitiy", but looks at the problem from the other perspective: It is used to create the pools for resources. Every available resource results in such an entity object, so it can be easily queued by DESMO-J build in features.

The second important concept is *events* which can be scheduled from DESMO-J. We only need the two events shown in figure 4.5 to implement the whole simulation tool at the current stage.

WorkCompletionEvent: This event can take a "TaskInstanceEntity" or "TokenEntity" from above and schedule the completion of the work like described above. This is how the mentioned queuing of entities really works.

ProcessStartEventGenerator: The Event generator starts process instances. It is implemented as event and as soon as this event is due and executed from DESMO-J , it starts the process instance and reschedules itself after the time configured via the process start distribution. So you only need to create and schedule the first event when starting up your simulation run, then this event generator works on itself.

The last concept I want to mention now is the *model* as you can see in figure 4.6 on the following page. This is the third and last white-box concept of DESMO-J[18]. The responsibility of the model is to set up everything needed during the simulation run, this includes distributions, resource pools, queues and components to report statistical data. It is the central class for controlling the simulation run. In the current implementation there is a concrete DefaultJbpmSimulationModel, which takes the process definitions of all processes which will take part in the simulation. These process definitions should be instrumented with additional simulation configuration parameters. An instance of this simulation model can be connected to a simulation experiment and executed. This is done by the standard DESMO-J Experiment class.

[18]see figure 4.6

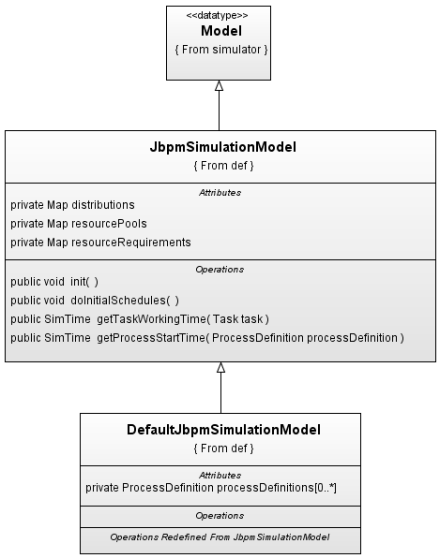


Figure 4.6: Simulation Model extended from DESMO-J.

The described concepts are sufficient to enable DESMO-J to steer jBPM processes by the event-oriented modeling style of Discrete Event Simulation. The next section will take a look on necessary changes and extensions of jBPM to support the combination.

4.2.3 Weaving simulation into jBPM

There are some differences between process runs in the productive application and simulation runs of the same process. Even if the simulation should be as realistic as possible, some changes are always necessary. The important changes are:

Task Management: Human tasks will not be processed by humans. The tool will simulate some processing time and finish the task automatically instead. Maybe the simulation tool must also simulate data changes which would have been done by the user in reality.

Wait States: The same concept is true for wait states. If you don't want to trigger a asynchronous system in simulation for example, you have to simulate the return after some time.

Called Services / Actions: You have to decide if services should be called during simulation. In jBPM services can be action classes (Java code) or scripts, embedded in the process definition. The user of the simulation has to decide for all of these services, if they should or should not be executed during simulation. If not, you may need to simulate some data changes or return values.

Decisions: Decisions made by humans, which means there are different outgoing transitions modeled for a task in the process definition, have to be made differently in simulation. Thus probabilities of different transition possibilities can be provided,

so the simulation tool can choose on itself. The same concept may be applied even for some automated decisions, based on process data. There it is useful, if for example some service call and the required return value was skipped.

Emails: Simulation should suppress Emails to avoid confused or annoyed recipients.

To deal with the described problems, I extended jBPM at two points: The parsing of the process definitions from a XML file to Java objects and the configuration which Java classes implement certain concepts.

The parsing is done by the so called "JpdlXmlReader" in a "readProcessDefinition" method. The simulation tool provides an extended version of it, the "SimulationJpdlXmlReader". This reads in addition to the normal process definition the simulation configuration and instruments the resulting process, which includes adding jBPM events. These events are for example instantiated when a task is created, so the code to connect to DESMO-J can easily be hooked in a jBPM action which will be executed within the event. This event mechanism is used to capture process end events and measure the process cycle time. The additional simulation configuration is encapsulated in a so called "SimulationDefinition" which can be connected to the process by a standard jBPM mechanism.

The other extension mechanism is the implementation class configuration. There are two XML files[19] used inside of jBPM to discover the right implementation class for a process construct, like for example node, task-node or action. I changed the configuration for the simulation version of jBPMto use special classes. These classes have slightly different behavior, for example a state adds an event like described above to itself, the decision takes an outgoing transition on itself if probabilities are configured, the MailNode does not send a mail but just does nothing. And action classes which contain own user code, can be configured to be executed or skipped during simulation.

There is one concept which is completely skipped and left to the open issues at the moment: timers. Timers can be configured to be due some time in future and are used for timeouts for example. Including the timers in simulation runs would require a correct connect to the simulation model time and maybe require some more sophisticated handling. If the timer models a timeout for example, the simulation environment can decide already at the start of the job if the timer will get due, because processing times are generated at that time. These topics were considered as not so important and out of scope for the moment. The focus was to get a working simulation tool first from which a better understanding of the real use cases and further requirements can be gained.

4.2.4 Simulation Execution: Experiments and scenarios

The tool knows two concepts to execute simulations and to support definition, simulation and especially comparison of different scenarios: *experiment* and *scenario*. One experiments is the frame for a whole simulation project, which consists of multiple different scenarios. The experiment is a collection of scenarios. The scenario defines one special simulation run, so a scenario corresponds to one DESMO-J Experiment. So

[19] org/jbpm/graph/node/node.types.xml and org/jbpm/graph/action/action.types.xml

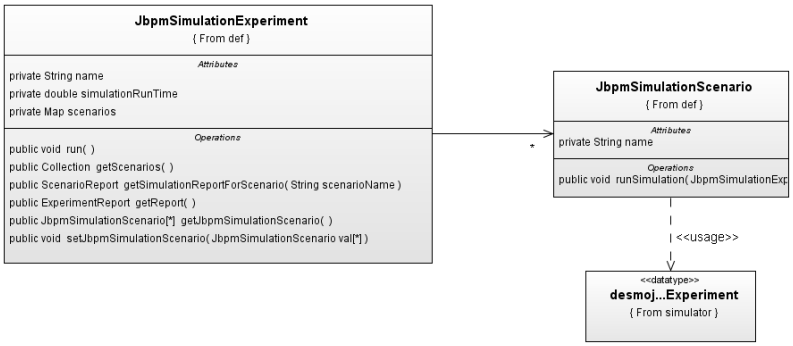


Figure 4.7: Simulation experiments and scenarios.

the term experiment is used differently in the developed tool as it is in DESMO-J. At run time, the experiment iterates over all scenarios and executes them. The Scenario therefor creates a simulation model as described in section 4.2.2 and a DESMO-J experiment, connects these two and starts the simulation.

When DESMO-J has finished the simulation a report for the scenario is created. Report just means that results and statistical measures are captured in some POJOs (Plain Old Java Objects) to be evaluated or shown graphically later.

Experiments and Scenarios are defined in one XML file, examples are contained in the appendix. The simulation tool has a parser to read these files and generate the necessary objects out of it. Afterwords the experiment can be run easily. This task of parsing and running can be done programatically, but is also available as Eclipse Runtime configuration. This makes it possible to right-click on a simulation configuration and just select "run"[20].

4.3 Details on jBPM simulation tool

4.3.1 Features and configuration possibilities

All supported features of the simulation tool have to be configured somehow. The simulation tool supports two different configuration possibilities:

1. Including the complete configuration into the normal jBPM process definition. This makes is it easy to use if you want to take an existing process and put it into simulation.

2. Defining everything in an own configuration file for the experiment. This makes it possible to separate concerns and leave the process-definition untouched. It allows to define different scenarios for one process definition.

The second option is the recommended one. This chapter explains roughly the basic options available. Examples for the configuration syntax are available in appendix A.1.

[20]This functionality is not yet available in public.

Distribution	Sample type	Required Parameter
Erlang	real	mean
Constant	real, integer, boolean	constant value
Poisson	integer	mean
Uniform	real, integer	min value, max value
Exponential	real	mean
Normal	real	mean, standard derivation

Table 4.1: Supported statistical distributions

Distributions

The main difference of process runs in simulation is, that randomness doesn't occur automatically but has to be expressed via statistical distributions like described in section 3.3. These distributions have to be configured somewhere.

The simulation tool supports all distributions supported by DESMO-J, actually it uses their implementation behind the scenes. An overview gives table 4.1 and you can find a configuration examples in appendix A.1.1. The choice of the distribution and correct parameters is up to the user of the tool, section 3.3.2 gave a short introduction on some distributions. If you have historical data available it is the best choice to use an own statistic software to discover the best matching distribution.

Once the distributions are defined they can be used at different places in the process:

- as start event distribution,
- as processing time distribution on tasks or
- as processing time distribution on states.

Another case, where distributions are used internally, is the specification of outgoing transitions. The user configures probabilities for the outgoing transitions and the simulation tool generates an uniform distribution to realize that internally. If these probabilities are configured on process elements, the simulation tool always chooses the outgoing transition itself. The probabilities can be configured for the following elements:

States: When states are signaled automatically after some time distribution, the tool has to decide the transition. If no probabilities are set, the default transition is used.

Task-Nodes: The framework ends the task after the configured time distribution and choses the transition. If no probabilities are set, the default transition is used.

Decisions: The simulation framework decides about the leaving transition if probabilities are configured. In this case, the default behavior (execution of expression or a handler class) is skipped. If no probabilities are specified, a decision still behaves "normal" and makes the decision depending on the result of the expression or handler.

Resource pools

Resource pools are a very important concept to allow realistic simulation or to support staff planning. In the simulation tool you can define multiple resource pools with a number of resources and costs per time unit. What is still missing is the possibility to define more sophisticated resource pools, which may support shift calendars or maybe even connect to some LDAP server to query these figures. Resources can not only be humans: every limited resource is a resource pool candidate, for example test equipment, machines or even money in the terms of liquidity. A state or task-node can consume any number of the same or different resources.

There is one remaining problem in the current implementation: if a task needs more than one resource pool, there is the risk of causing a deadlock, because the simulation locks resources as early as possible but wait for all required resources before a task can be processes. More details on how to configure resource pools are included in appendix A.1.2.

Called services / Actions

A process has to interact with other systems or software to fulfill its job. This is sometimes problematic in simulation runs, because you don't want the simulation processes to change data in your productive ERP system. And you don't want to provide a fully equipped integration environment either, because it is too expensive and accessing the ERP system does not add any new information to the simulation.

For this situation the simulation tool provides the possibility to skip the calls to other parts of the system. This feature is available the other way round, too, which means that special usercode can be added only for simulation. This can be used if a call to an external system is suppressed to simulate result data or to change process variables accordingly for example. More information on this behavior can be found in appendix A.1.4.

Data sources and data filters

Sometimes you need data in the process context during process runs. The context is implemented as a "Map" with variables in jBPM. Every process instance can have an unlimited number of variables. These data is normally changed by the application steering the process, for example a business order object may be provided when starting the process or entered by the user during a human task.

In simulation a lot of these data changes cannot be done the normal way, at least data at process start events or data provided by humans must be supplied in different ways. To support this, the simulation tool implements two concepts:

Data source: A data source creates a stream of new objects for a process variable, for example order objects to start order processes. The data source must be implemented by the user of the simulation, a default implementation which can take historical data from a productive jBPM instance is provided out-of-the-box.

Data Filter: The data filter was named after the "Pipes and Filter" architectural style[21], it is a filter which can be applied to modify data. This can be used for example to simulate data changes done by humans in reality. The filter has to be implemented by the user of the simulation and because there is no reasonable default behavior, no default implementation is provided.

Exemplary configurations of data sources and data filters are contained in appendix A.1.3.

Warm-up phases

No automatic recognition of the beginning of the steady state, as explained in section 3.3.3, is implemented in the simulation tool. The user of the simulation has to evaluate the first simulation run results to decide at which point in model time the steady state begins. If that point is figured out, the simulation experiment can be easily configured to reset all statistical counters at this point in time, which is realized by an event internally. An example is contained in the experiment configuration in appendix A.1.6.

Open issues

There are still some open issues in the current implementation of the simulation tool. Most of them occurred because of the motivation, to implement a basic working simulation tool fast and distribute it to the community as open source software. If more and more projects start to use it, new requirements and the really missing features will show up and be requested. And the more users play around with the simulation, the more ideas can be collected in the jBPM forum. For the case study in this thesis, the provided functionality was absolutely sufficient. Open issues and ideas for the simulation tool are collected in the issue tracker of the jBPM project[22]. Some ideas about more features and ideas can be found in section 6.2 as well.

4.3.2 Configuration proposal from historical data

Section 2.2.1 explained the BPM life-cycle and motivated why it makes sense for a Business Process Simulation tool to support simulation as part of the life-cycle. This includes the possibility to use existing historical data to simulate process change proposals for example. The data should be retrieved from business activity monitoring. Unfortunately the BAM component of jBPM is just planned and not implemented yet. To support an easy use of historical data as simulation input anyway, the simulation tool implements queries for this use case itself.

jBPM uses a command pattern internally to support user specific extensions or to execute code in different architectures[23]. The simulation tool contains a "GetSimulationInputCommand", which collects statistical information. The standard use case is to use the result for generating a scenario configuration for the simulation, which automatically corresponds to the status quo. Hence, running it should result in nearly

[21] see for example http://www.enterpriseintegrationpatterns.com/PipesAndFilters.html
[22] http://jira.jboss.com/jira/browse/JBPM
[23] http://wiki.jboss.org/wiki/Wiki.jsp?page=JbpmCommands

the same performance indicators than reality, which makes it also a good validation if your model works right. The information queried from the jBPM audit log data is:

- time between start events,
- average waiting times in states,
- processing time for task nodes and
- probabilities of outgoing transitions for states, tasks and decisions.

4.3.3 Measures and performance indicators

The DESMO-J library defines a "Reportable" interface, which is implemented by all statistical objects. This means that these objects can report their measures easily to the experiment via the "Reporter" class. Reporters are added automatically by DESMO-J when creating statistical objects. The only thing the simulation tool has to do, is to register a receiver for reports at the experiment. This so called "InMemoryOutput", which implements the "MessageReceiver" interface of DESMO-J, collects all the statistical data and generates a "ScenarioReport" object out of it. Figure 4.8 on the following page shows the involved classes and the relationships between them, figure 4.9 on page 64 shows the value objects which store the statistical measures inside the ScenarioReport. The usage of the types are shown in table 4.2.

Resource pool utilization

One special remark on resource pool utilization, because this is the only measure which is not really obvious and not copied one to one from DESMO-J. This is because the simulation library don't know about resource pool or resource utilization. It only supports the concept of queues. Hence, the simulation tool used an additional queue for every resource pool just to measure utilization. This queue is not really used as a queue, but it always contains the not utilized resources. As soon as resources are used, they are removed from the queue. This makes it possible to get the utilization data from the statistic of this queue, because the length (min, max and average) of the queue corresponds to the number of not utilized resources and the waiting time corresponds

Value object type	**records / used for**
ValueStatisticsResult	statistics about a special value process cycle times, wait times before states
QueueStatisticsResult	statistics about queues resource queues (e.g. avg. waiting time or queue length)
UtilizationStatisticsResult	statistics about utilization utilization of resource pools
TimeSeriesStatisticsResult	values for different points in time resource utilization over time

Table 4.2: Value object types and usage.

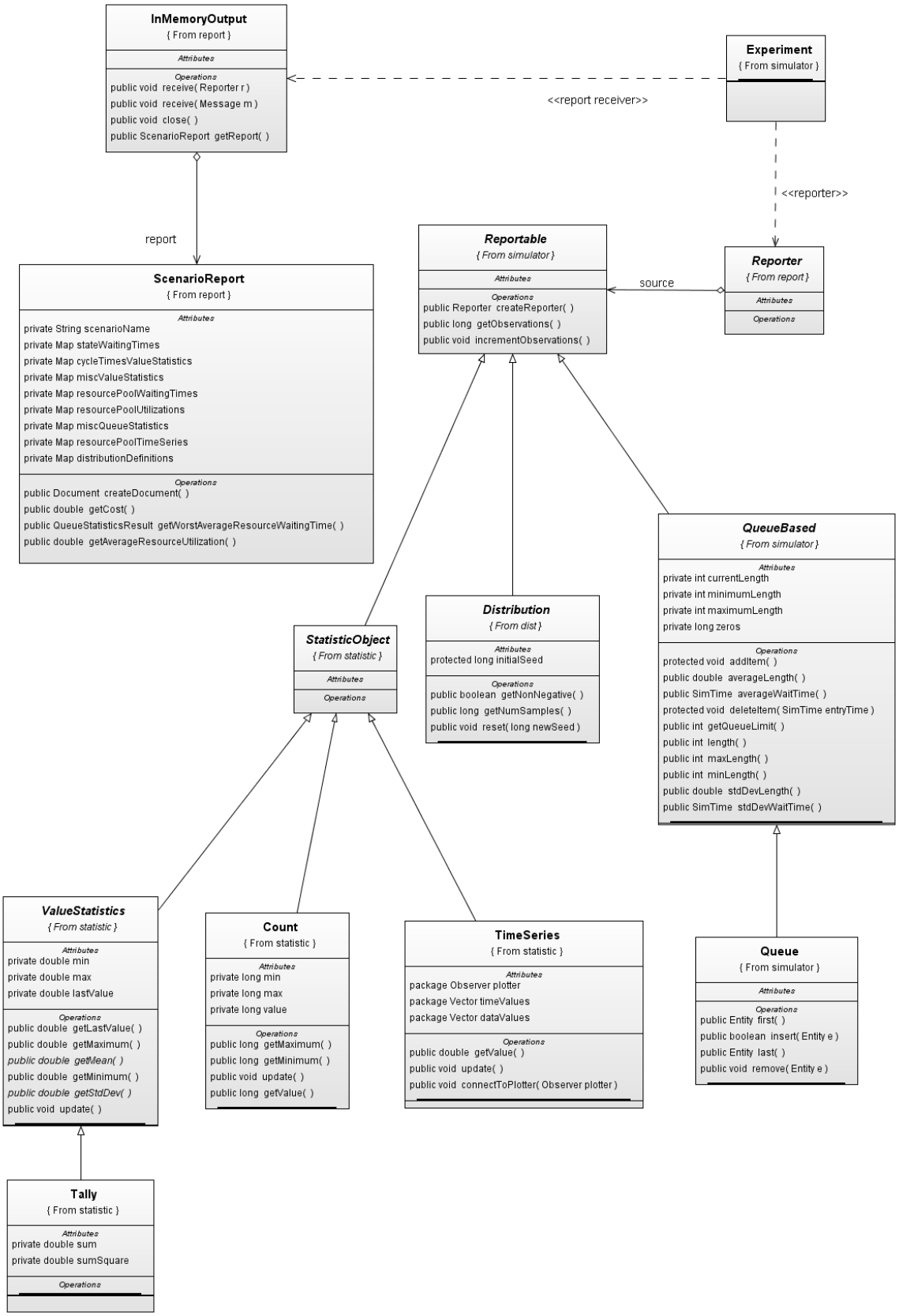


Figure 4.8: DESMO-J Reportables and in-memory report in simulation tool.

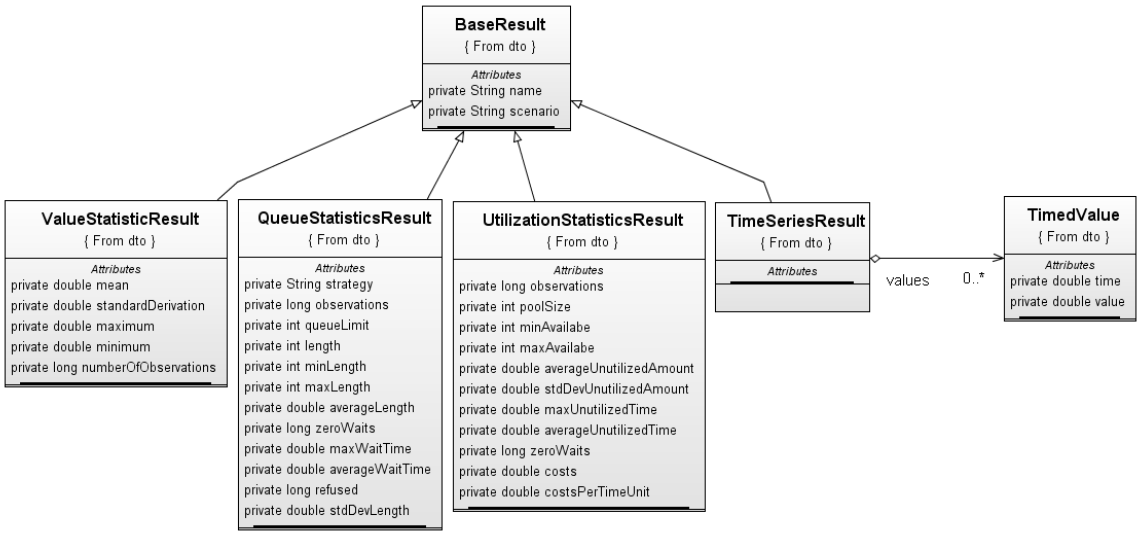


Figure 4.9: DESMO-J Value objects representing statistical measures.

to the time, resources wait between usages. So every resource pool is realized internally with two queues, the one described in this passage and the real queue for processes waiting for a resource to get available.

Costs

Also very interesting is to look at costs of process runs, because quite often the user wants to compare costs of different process scenarios. There are more than one possible causes of costs inside a business process. I will distinguish two groups of costs here: The first group are costs which are measurable with money and the second contains weaker factors which are not easily measurable, like damage of the image because of delayed orders. Examples for the first group of costs, on which I will concentrate, are

- resource costs,
- (fix) costs per process run,
- (fix) costs per service call,

but even weaker factors may be also expressed in money, for example

- missed profit because of missed opportunities,
- missed profit from a lost order,
- penalties for slow processing.

The costs for resources are configured as costs per simulation time unit via the resource pools. If your time unit for simulation is second, you have to specify the costs

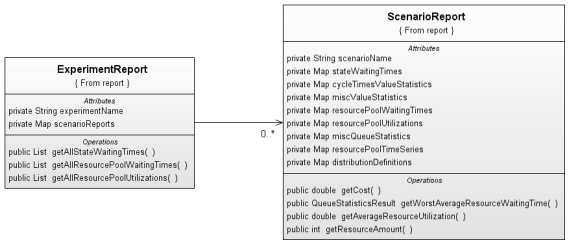


Figure 4.10: DESMO-J Experiment and scenario report.

for the resources per second. One important question with resource costs is how to handle the not utilized time. In reality there are different possibilities, either the people really don't have any work to do and simply wait. Or they have a big pile of todos from outside the processes under simulation, so the people can easily do other jobs. In the first case, the not utilized time of people should be calculated with 100 % into the process costs, in the other case you maybe don't want to calculate any costs for not utilized times. But even in this case, you might calculate some percentage of this time as costs, because of set up times occuring when people have to change the type of job.

Other costs than resource costs are slightly different in every simulation project. The goal while developing the simulation tool was to provide flexibility and allow the user to plug in exactly what he needs. Maybe after gaining some experience with the use of the simulation tools, some meaningful default implementations can be provided. Currently the user can flexibly specify business figures.

The first decision when defining a business figure is to determine the type. For this thesis I limited the type to costs, but it is easily extensible to support any type of performance indicator and provide statistical results of them after simulation runs. If the type of the business figures is costs, the sum of the costs collected over all process instances will be summed up and added to the resource costs. This makes it possible to provide only one figure as costs and to make simulation scenarios easily comparable.

The configuration of business figures is shown in appendix A.1.5, A.1.2 shows the attributes available for resource costs definition.

4.3.4 Result visualization

As described in the last section the simulation tool captures all results in own Java objects and provides an "ExperimentReport" as a collection of "ScenarioReports" like shown in figure 4.10. Sometimes the simulation projects want to evaluate the results automatically, for example to optimize process parameters during runtime. Then this representation is fine. But very often the results should be presented to the user of the simulation or to some business analyst. To support this, some visual representation of the results must be available.

DESMO-J itself provides a HTML report with statistical measures for all objects which report them. These are for example queues, tallies reporting waiting and process cycle times or distributions. An example of this report can be found in appendix A.2.1. This report shows almost all important figures for simulation results, except for

costs, but only for one special scenario and in a format which requires some deeper understanding of the simulation tool insides.

To integrate the simulation in the BPM life-cycle better, reports which can be evaluated by business analysts must be provided. So I see intuitive graphical reports as a vital part of a good BPS tool. Unfortunately, the required reports differ from simulation use case to use case, so it wasn't possible to provide detailed reports for every question which may be asked. The requirement was to provide reports, which can be customized easily or serve as a starting point for own reports. This solution has the advantage that the users can build own reports fitting to their needs. He has working examples available as templates and the possibility to contribute own reports back to the community.

To support this flexibility a reporting engine was integrated in the simulation tool. Like already mentioned it is JaserReports and iReport as graphical report designer. There are three different types of reports:

Scenario report: The report shows details of one special scenario. Therefor the "ScenarioReport" objects is added as parameter to the report. Details, like a pie chart showing waiting times, are realized as subreports.

Experiment report: This reports shows a comparison of scenarios contained in the "ExperimentReport", which is a parameter of this report. Like scenario reports the main details are realized via subreports.

Subreports: The subreports show one specific detail, like a pie chart of waiting times. Because JasperReport is row based, like most report engines, the subreports need a data source providing rows. These rows can be used as values of the pie chart too. The whole simulation results are available as Java objects, for this case JasperReports provides the "JRBeanCollectionDataSource" class, which is a simple facade allowing to use a Java collection as data source for reports.

More details on the implementation and information how to extend the reports are contained in appendix A.2. Table 4.3 on the following page shows the provided default reports and the contained information. Examples of these reports created for the showcase can be found in appendix A.2.3 and A.2.4.

4.4 Simulation showcase

4.4.1 Introduction

The simulation showcase is an easy business case leveraging simulation and showing possible benefits. It was developed to provide a freely available tutorial for the use of the jBPM simulation tool. This tutorial is available online on the camunda website[24] and may be improved when the simulation tool itself is improving.

Because it serves as an introduction, the business case is not really complicated. This is a basic requirement of an introduction, but also limits the demonstrated value

[24] http://www.camunda.com/jbpm_simulation/jbpm_simulation_tutorial.html

report name	**contained information** **Subreport name**	**contained information**
scenario report	scenario details	
	WaitingTimeBeforeTask	waiting time before states
	WaitingTimeForResource	waiting time for resources
	ResourcePools	resource pool performance indicators
experiment report	scenario comparison	
	ScenarioComparisonTable	overview table comparing performance indicators of scenarios
	WaitingTimeBeforeTaskComparison	comparison of waiting time before states
	WaitingTimeForResourceComparison	comparison of waiting time for resources
	ResourceUtilizationComparison	comparison of resource utilization

Table 4.3: Provided default reports and contained information.

of simulation, which gets more powerful the more complex the processes are or the more different processes are involved.

4.4.2 Business process and business story

The showcase looks at an easy business process: The handling of returned goods from customers. This applies for example for a webshop selling technical devices, where the customers can send back any goods within two weeks without any reasons (at least in Germany) or within the warranty, if there is a defect. The goods are checked after arrival from some clerk, if they are really broken. In this example the check is divided in two parts, a first quick check after the goods arrival which can approve "easy" defects and an extended test with additional test equipment. If the defect cannot even be approved with that test, the goods will be sent back to the customer, because they are assumed not to be broken. If the defect is approved, the money will be refunded to the customer. The defect goods are just thrown away in this easy tutorial.

In Germany there is one additional requirement: If the customer returns his shopping within two weeks, the shipping costs must be paid for him. So the shop has to transfer the money to the customer's bank account in this case. This process was modeled with JBoss jBPM with the already mentioned jPDL language as shown in figure 4.11 on the next page. The XML source code for this process is included in the appendix A.3.1.

4.4.3 Generating and reading historical data

Section 2.2.1 stresses the importance of using historical data as simulation input. The process in this showcase never run on a productive system, so there is no historical data available. To demonstrate the usage of historical data as input I created a small

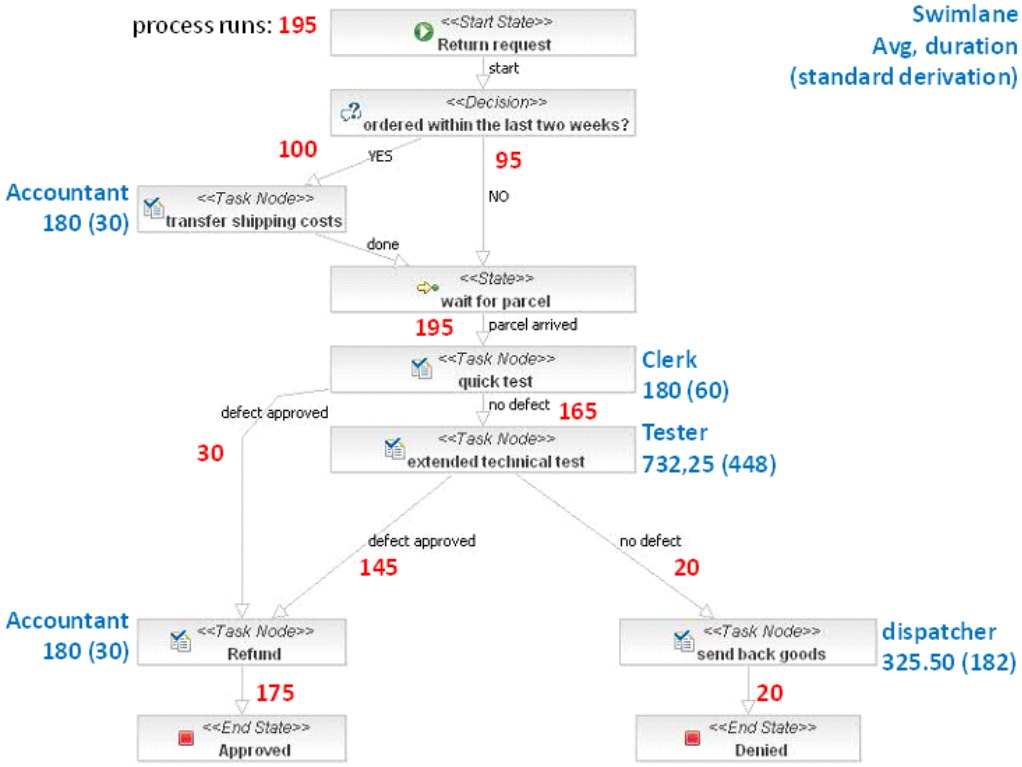


Figure 4.11: jBPM process diagram with actors and simulation statistics.

Java program which executes different process instances taking different paths through the process. This can be done easily via the jBPM API, the source code is contained in appendix A.3.2.

This creates a faked history, so we can use the simulation tool to extract a scenario configuration from it. This configuration provides exactly the same statistics as the jBPM history, so a simulation run should result in more or less the same performance indicators than the real life process would do. The scenario configuration is included in appendix A.3.3, which shows a complete scenario configuration example and the statistics of historical process runs, because they are used as input parameters there.

The sample data here was only generated to demonstrate the features to retrieve statistics from jBPM and use it as simulation input. Because of the poor algorithm of generating the data, its value is limited. Transferring costs by the accountant for example should always take duration close to the average and the big standard derivation in the tutorial is not realistic. It would make more sense for tasks, which processing times differ a lot from case to case. An example could be the extended technical test.

For the showcase the numbers above were slightly changed, primarily to use shorter numbers. Figure 4.11 shows the used numbers included into the process diagram for a better overview.

4.4.4 Use Case 1: Discover best staffing strategy

Assuming that this process run fine during the year, there might be the question about staffing strategies in Christmas time, because you expect a lot more return of goods than during the year. A forecast of the returned goods number is available and you want to calculate the required number of people.

Using spreadsheet

The first idea is to apply some spreadsheet calculation and calculate required staff figures depending on scenarios, because nobody knows the exact number of returned goods in advance. So most likely a "worst case" is calculated additionally to the "normal case". This calculation is not too hard. Figure 4.12 shows the resulting table for the data of the showcase. From this table a staffing strategy can be derived. Depending on the importance that your process cycle times don't blow up, this strategy will orientate closer to the normal or worst case.

Scenario Status Quo	Percentage	Amount	Duration (seconds) average	Duration worst case	costs average	costs worst case	people needed average	people needed worst case
Total	100%	**195**			**4.601 €**	**6.813 €**		
transfer shipping costs	51%	100	180,00	210,00	400 €	467 €	0,67	0,78
quick test	100%	195	180,00	240,00	390 €	520 €	1,22	1,63
extended technical test	85%	165	732,25	1180,25	3.021 €	4.869 €	4,20	6,76
Refund	90%	175	180,00	210,00	700 €	817 €	1,17	1,36
send back goods	10%	20	325,50	507,50	90 €	141 €	0,23	0,35

Scenario Christmans	Percentage	Amount	Duration (seconds) average	Duration worst case	costs average	costs worst case	people needed average	people needed worst case
Total	100%	**500**			**11.797 €**	**17.469 €**		
transfer shipping costs	51%	256	180,00	210,00	1.026 €	1.197 €	1,71	1,99
quick test	100%	500	180,00	240,00	1.000 €	1.333 €	3,13	4,17
extended technical test	85%	423	732,25	1180,25	7.745 €	12.483 €	10,76	17,34
Refund	90%	449	180,00	210,00	1.795 €	2.094 €	2,99	3,49
send back goods	10%	51	325,50	507,50	232 €	361 €	0,58	0,90

Resources	cost per hour	hours per day	cost / second	**Status Quo avg**	**Bad**	**Christmas**	**Bad**
Accountant	80 €	7,5	0,022222222	**1,83**	**2,14**	**4,70**	**5,48**
Clerk	40 €	8	0,011111111	**1,22**	**1,63**	**3,13**	**4,17**
Tester	90 €	8	0,025	**4,20**	**6,76**	**10,76**	**17,34**
Dispatcher	50 €	8	0,013888889	**0,23**	**0,35**	**0,58**	**0,90**

Figure 4.12: Calculation of required staff with spreadsheet.

This approach looks straight forward and is obviously better than no planning at all. However it has some disadvantages:

Ignorance of statistics: Always calculating the worst case adds safety but also unnecessary costs. Because the spreadsheet calculations ignore randomness it also ignores statistical knowledge, so it is more a rough guess. The user don't get a feeling about how many people are really necessary to continue running with average process cycle times. Simulation can use statistical distributions and simulate

long periods of real time, making a more qualitative conclusion about required resource numbers, backed by statistical theory.

Complex tables when looking at big processes: The table shown in figure 4.12 was not too complicated, but we handle a very simple and even simplified process in this showcase. The table can get huge, complex and unclear. And if the user knows about statistics and includes sophisticated formulas to get a more realistic result, it gets even more complex.

Difficult to apply to different processes: The showcase in this tutorial consists of one process definition only. In reality you deal with different processes consuming the same resources. This adds a level of complexity which is really hard to handle by spreadsheet calculation, because possible dynamic phenomena will increase. If for example every process has a bottleneck at the same time, the average utilization and the per process calculation looks fine, but all processes running together mess up the cycle times.

Another argument for applying simulation technologies over spreadsheets is, that if the simulation tooling is provided out-of-the-box with the process engine, the required effort is small, even smaller than setting up a spreadsheet. This really leverages simulation to broad range of people and allows them to participate in the BPM life-cycle.

Using simulation

For using the simulation a configuration was developed including scenarios which reflect exactly the spreadsheet calculation to allow an easy comparison. The final result, showing the average resource utilization, process cycle times and costs for each scenario, is given in table 4.4 on the following page. In this showcase, the "unutilized-time-cost-factor" is set to zero to keep comparability to the spreadsheet approach, so not utilized resources are not included in the cost calculation, thats why the costs for the normal and worst case scenario are not very different among each other. Please note, that a required warm up period was ignored in that experiment. This means, the results are not completely realistic because the system is underutilized at the beginning.

The table only shows a very compact summary of the simulation results. There are much more figures available and some of them are also shown in the graphical report. Parts of it are contained in appendix A.2.3 and A.2.4. One question may be, why the process cycle times are better, that means shorter, in the worst case than in the normal case. This is true, because more resources are available in the worst case scenarios yielding to better cycle times. Because the underutilized time of resources is not included in the cost calculation, it has no influences on the costs.

Optimization

The simulation came up with results for a given staffing strategy. But the user wants to gain the best staffing strategy for the given situation. Unfortunately simulation cannot calculate the right staffing strategy out-of-the-box, it can only test assumptions. So there are two approaches to get the desired staffing strategy:

scenario name	avg.resource utilization	costs	process cycle time
christmas normal case (Longest waiting time: tester with 1.266s)	85,26 %	11.206 EUR	2.226 s
christmas worst case (Longest waiting time: dispatcher with 442s)	62,70 %	11.797 EUR	1.199 s
status quo worst case (Longest waiting time: tester with 2.266s)	79,35 %	5.401 EUR	2.972 s
status quo normal case (Longest waiting time: accountant with 1.572s)	85,29 %	5.295 EUR	4.150 s

Table 4.4: Simulation results overview of use case 1.

Manual guessing: The user can come up with resource numbers which make sense to him and the simulation tool tests this hypotheses. If the resulting cycle times or utilization figures are not good enough, the user can change the scenarios and simulate again until the result is satisfying. By playing around with the simulation the user can gain a better understanding of specific parameters or possible bottlenecks. Even if "playing" sounds not very sophisticated it can really help to understand your processes and to find a good staffing strategy.

Automated guessing: A more scientific approach is to apply optimizing technologies to find good staffing strategies. As mentioned in section 3.1 simulation is introduced because mathematical optimization is too hard to apply. So optimization comes normally down to automatically generate scenarios and chose the best one. This can even be done by a simple brute force approach but works normally better with more sophisticated algorithms, for example genetic ones. More details on this idea were discussed already in section 3.5.

The showcase includes more scenarios and thereby demonstrates the guessing approach. Additionally a simple brute force implementation example of the automated approach is provided, see appendix A.1.7 for more details.

4.4.5 Use Case 2: Evaluate process alternatives

A second use case for simulation could be the question if costs could be saved when the expensive technical tests were skipped. Therefor an alternative business process is developed, which just refunds money for returned goods. To avoid that customers recognize that they can send back everything, even working equipment after one year for example, random checks of incoming goods are still made in 5 % of all cases. All returned goods are handed to a third partys. From the figures presented above it can be calculated that 89,2 % of all returned goods are really defect[25]. The partner company charges 90 % of the value of goods based on this percentage, the difference is calculated for their handling and safety reasons. The changed business process is shown in figure 4.13 on the next page.

[25] 15 % of defect goods are recognized in the quicktest, plus 78 % of the raiming 85 % in the extended test

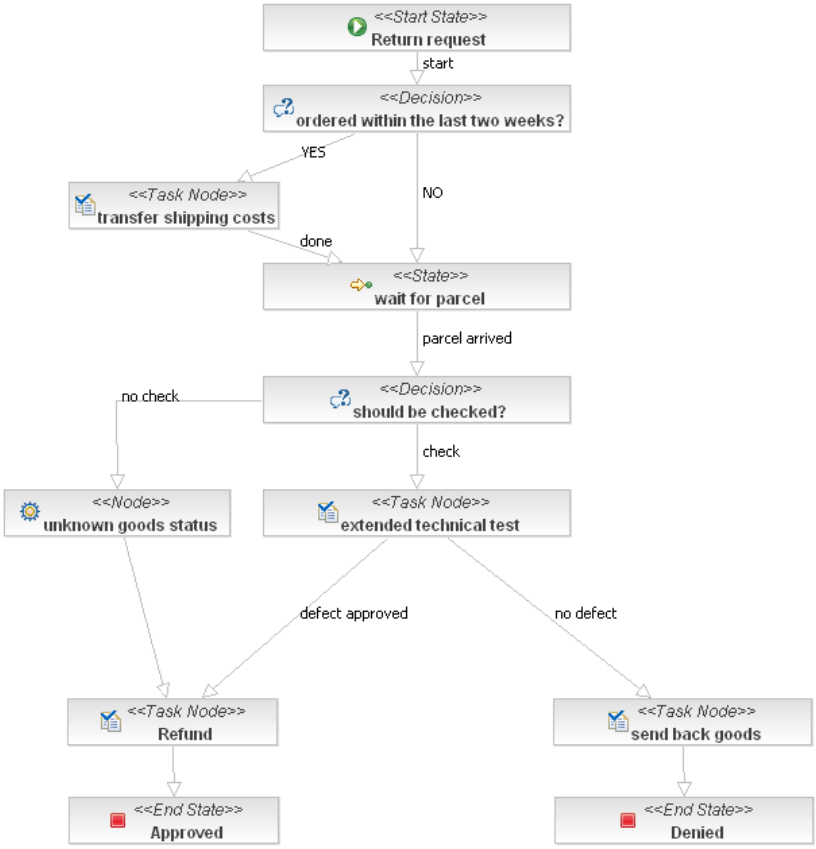


Figure 4.13: Alternative process model.

The simulation scenario developed for this use case includes a data source to generate return orders as process variables. These are needed to know which value of goods are handled by each process instance. Additionally a business figure is added, which calculates 90 % of the goods value as costs to the overall process costs, which consist of resource costs besides that. To make the scenario really comparable, costs of the full value are added to the status quo process in the case of an approved defect. Details on the configuration and source code can be found in appendix A.3.4.

After running the simulation the result confirms, that the alternative process saves costs. Overall costs of 16.401 euro are calculated instead of 19.516 euro for the old process. This motivates to investigate further into that idea, even if I want to stress, that this simple example ignores some parameters. For example the company may get a refund for defect goods from the manufacturer, if it is broken within the warranty. These additional pieces of information have to be added to the simulation model in reality to make it more meaningful.

Chapter 5

Case study: dataphone GmbH

5.1 Introduction and simulation goals

During this thesis a real life prototype was developed in cooperation with the Austrian company dataphone GmbH[1]. dataphone focuses on mobile systems which can improve efficiency of user interfaces in business processes significantly. A lot of applications are in the field of logistics and stock control, for example stock control centers or consignment with mobile scanner or even voice control.

The problem of providing stock control software is, that there is a lot of competition in that area[2]. And all software vendors are able to deliver the basic functionality required by the customer. But then this software is different for every customer in a lot of aspects, hence a comprehensive customizing of the software is necessary. So the goal of the so called *Logis2* system of dataphone was to provide a basic infrastructure and ready to use components for stock control software. This enables dataphone to implement a new customer specific version by plugging together already available parts and implement new parts by using provided extension points. Unstable parts of the system, like business processes, should allow flexible changes which enables an agile and fast adoption of new business requirements.

Logis2

Based on the described situation, dataphone developed an architecture, which supports changes and customizations. It is shown in figure 5.1 on the following page. The required flexibility is achieved by the use of the business process engine JBoss jBPM for all processes and a second framework called JBoss Seam[3]. By integrating different technologies, like Enterprise Java Beans 3, JBoss jBPM, Java Server Faces (JSF) or Java Portlets, Seam hides a lot of complexity of these techniques from the developer[4]. Web user interfaces can be created by providing pageflows with JBoss jBPM. This allows to customize pageflows as flexible as business processes. Basic functionality is provided as Enterprise Java Beans and can be reused in different customer applications.

[1] http://www.dataphone.at
[2] This passage is based on an internal dataphone paper
[3] http://labs.jboss.com/jbossseam/
[4] seehttp://labs.jboss.com/jbossseam/

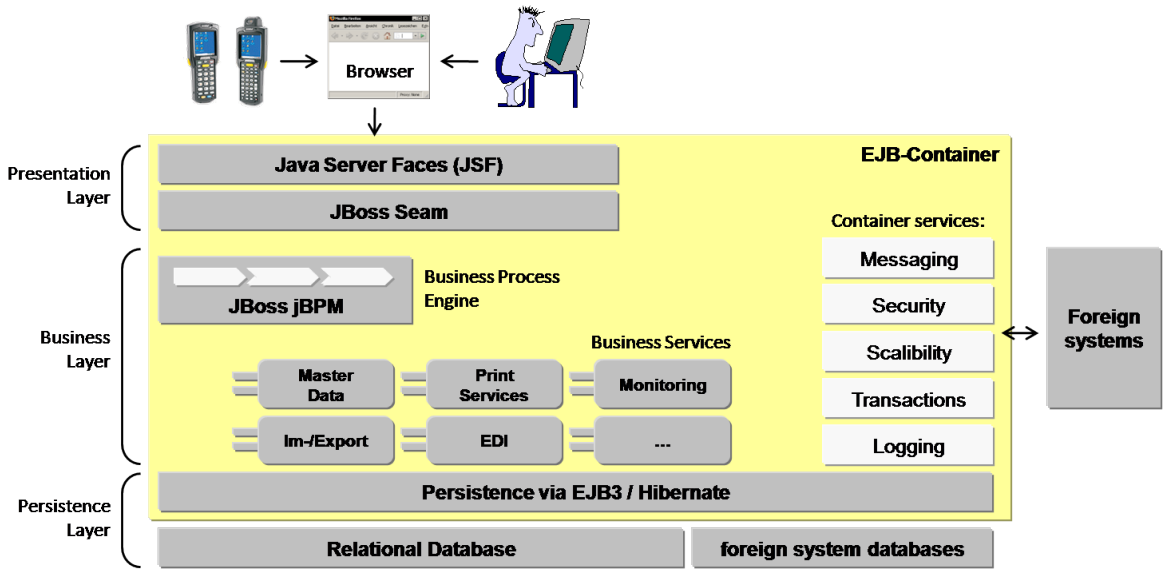


Figure 5.1: Basic architecture of Logis2.

This is an implementation of the idea of reusable software components or services.

One central process of Logis2 is the consignment process. This process implements the whole consignment of orders until they are collected in the collection area. One main component of the solution is a special tour planning algorithm from the university of vienna, which creates pick items out of the order items, whereas an order item can be split in multiple picks. The picks are combined into tours, which represent one picker walking through the stock to pick all items on the list. The algorithm tries to minimize the required walking time for each of them. It takes into account that all articles required for a special order should arrive nearly at the same time in the collection area, because this has limited space. The consignment process and basic business objects are shown in figure 5.2 on the next page.

During execution, every tour becomes one consignment process instance in jBPM and pick items are realized as task nodes. This use of the process engine is not really characteristic, because the consignment is not a typical business process. A better term than business process may be workflow in this case. But this workflow is one of the most important functions of the stock control system and flexibility was a requirement for Logis2, so it makes sense to use a lightweight process engine like jBPM anyway.

Simulation in Logis2

One problem in companies using a stock control system is staff planning. The amount of orders is different every day and may vary from season to season. Another phenomena is that all people in the stock are always utilized, not really depending on the amount of work. So the manager should get a tool to plan the required resource amount. This planning includes some assumptions about the future, because not all orders for a day are known in advance. Normally a big part of them are placed during the day on which they are delivered. This adds randomness to the input, so an easy spreadsheet

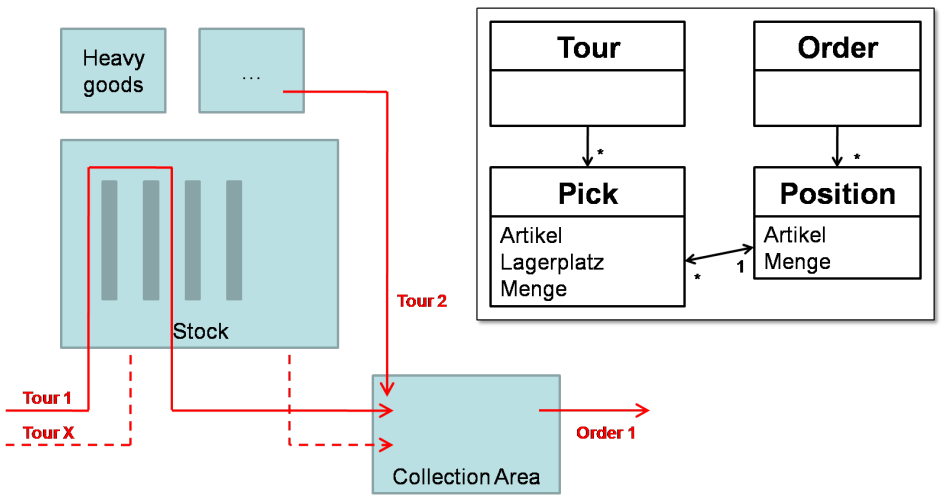


Figure 5.2: Visualization of consignment process.

calculation of the resource amount is problematic as described in the last chapters.

The randomness of the input, occurring independently at different locations in the process, makes the staff planning with figuring out how many resources are needed, an ideal candidate for a simulation experiment. Hence this was the basic use case which motivated the introduction of the simulation tool in Logis2. To be more specific, the goal was to provide information to the manager which looks like:

- If 10 people are available, the orders of today can be finished until the deadline with 99 % probability,
- if 9 people are available the probability is 96 %,
- with 8 people 90 %,
- and so on.

These informations can be calculated using statistics. To get reliable results a sufficient number of repetitions of simulation run per scenario with different seeds is required. As mentioned in section 3.3.4 a rule of thumb is 30 repetitions. Because this may cause performance problems an alternative solution was developed, which provides the following information to the manager:

- If 10 people are available, 98 % of the due orders can be delivered today,
- if 9 people are available 95 % of the orders,
- with 8 people 80 %,
- and so on.

The two results look very similar. But the quality of the first conclusion is higher, because it includes already statistical uncertainty, whereas the second result suggests that the result is fixed. For the prototype both solutions are implemented because the choice which one to use is shifted to a time in future, when experiences about runtime and performance from the productive system are present

For the realization an iterative approach was chosen. In the first version, developed during this thesis, it is sufficient to assume there will be the same amount of people all over the time. For next versions a more sophisticated shift calendar should be taken into account. This would make it possible to plan resources more exact and provide better information to the manager.

Another use case identified is the optimization of parameters of the tour planning algorithm. The tour planning itself is a special algorithm, hard coded and optimized for performance. But it has some configuration parameters which can be changed, for example the maximum of picks per tour or processing time in the collection area. Because of limited space, the later influences the amount of orders which can be consigned in parallel. The simulation can evaluate different parameter configurations and provide a recommendation to the stock manager. This use case will not be addressed any further by this thesis but can easily be added by using a similar approach to the resource parameter optimization.

Unfortunately, the Logis2 software was not deployed at the pilot customer as planned. Even worse, the consignment process wasn't fully implemented at the time finishing this thesis. So the integration of the simulation tool into Logis2 was done based on another process, the incoming goods process. This is sad, because no real life data and experiences can be included in this thesis, even if the case study proves that the integration works and shows how it can be done.

5.2 Implementation

Figure 5.3 on the following page shows an overview of the planned simulation. Referring to the process model defined in section 2.3.5, the first phases after defining the goal are analysis, data definition and construction. The result of the analysis was that the whole consignment process, or the whole incoming goods process for the prototype, can be used for simulation as it is deployed for production, only some calls to external systems have to be mocked to avoid real bookings in the backend systems. The statistical data required for processing time of tasks, which corresponds to walking times basically, is taken from historical data automatically when running the simulation. Necessary input data, like orders and order forecasts, can be queried from an already available service in Logis2.

So the construction phase is reduced to wire the existing components together and to develop a simulation experiment configuration. From the technical point of view all this can be done in one Java class representing a service, which is a stateless SessionBean[5] in the Logis2 architecture. The simulation experiment configuration is constructed on the fly in memory and executed afterwords. The simulation start and end date are handed in as parameter, as a default this is from "now" till 5 p.m., where

[5]for information about Enterprise Java Beans see for example [BR07]

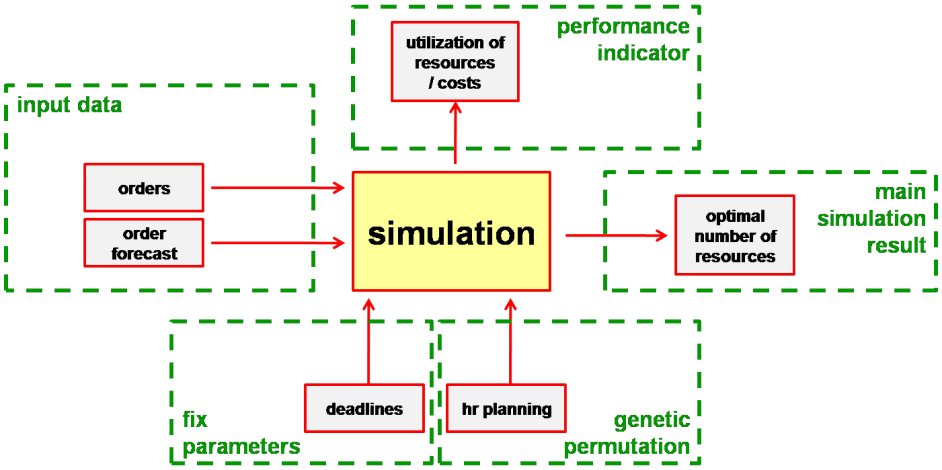


Figure 5.3: Overview over planned simulation with inputs and outputs.

the logistic partner collects all shipments. The percentage of finished orders is the important result of the simulation run and is the only performance indicator taken into account at the moment. The simulation run is repeated several times for the same parameters with different seeds to get a more reliable result.

Since the goal is to find a best suited staffing strategy, different number of resources have to be tested. As already stressed in this thesis, the simulation can only test a special set of input parameters called scenario. So the mentioned service includes a small algorithm to generate scenarios as well. In the prototype, developed in this thesis, this algorithm is quite simple and just add resources one by one until no significant change is recognizable in the results for some rounds. The sequence of operations is illustrated in figure 5.4 on the next page.

The implemented stateless SessionBean offers three different services which vary only in the provided result. One service returns a single recommended resource number, the second one returns the probabilities to finish all orders and the last one the percentage of finished orders per resource amount. These results correspond to the goals defined in the last section. The usage of the results or the presentation to the business analyst is not part of this thesis and will be done by dataphone itself.

5.3 Results and experiences

Due to the lack of productive data and the delay while implementing the consignment process, no real life experience could be gained. This is obviously very sad but also everyday life in real projects. Especially performance experiences and feedback about gained business value would have been very interesting.

What can be said is that the integration of the developed simulation tool into the Logis2 architecture was pretty straightforward. Most of the work is encapsulated inside the tool, so not much code was needed and everything could be hidden in one stateless SessionBean plus one data source. The generation of the simulation configuration via historical data was easy and proved very useful, because no manual interaction is

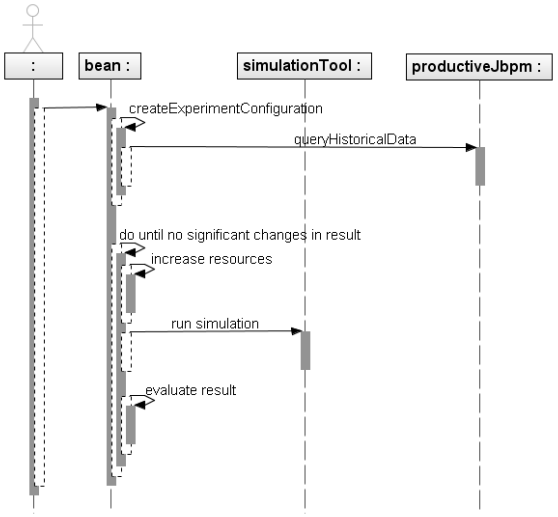


Figure 5.4: Sequence diagram of simulation StatelessSessionBean.

necessary. It reduces later maintenance work also.

An important feature missing in the tool is the ability to optimize input parameters, so this had to be implemented prototypically in the SessionBean itself. Other open issues are also cumbersome, like the missing shift calender, but can be introduced on the basis laid with this thesis easily afterwords.

One problem should not be concealed: The Logis2 architecture had one flaw when integration JBoss Seam and JBoss jBPM. Seam can wire together everything, means the JSF GUI, the business logic as SessionBeans and the processes in jBPM. Unfortunately this can lead to an architecture, where logic is located in the presentation layer. This was partly true for some of the dataphone processes where business logic was contained in SessionBeans which relied on the Seam environment. This made it hard to access these business logic without Seam, as it is done in simulation or unit tests. This had to be corrected first by moving the appropriate logic to usercode in the business process or changing the SessionBeans to work without Seam. I mention this problem because it is an important hint for simulation projects working with the jBPM based simulation tool: The architecture should be clear and correct, otherwise problems are preassigned. But this is true not only for simulation projects.

Chapter 6

Conclusion

6.1 Summary

During this thesis I have developed a working open source business process simulation tool. The code was contributed to JBoss jBPM and helps this project one step towards being a complete BPMS as defined by Gartner[1]. A showcase, which is available as online tutorial as well and a real life prototype proved the applicability of the tool and demonstrated the business value achievable.

Additionally the reader should be familiar with the basic concepts of Business Process Simulation, from the business and technical perspective. Links to important literature are provided. Hence I think this thesis can serve as a starting point for further research on BPS topics. And since the code is open, it is possible for everybody to improve the tool.

The goals defined for this thesis in section 1.2 were reached, even if a lot of ideas for improvements couldn't be taken into account. And looking back into section 3.4.2, the developed tool satisfies the defined requirements for a BPS tool, except for animation and a sophisticated conclusion-making support. These and other open issues or unaddressed ideas are collected in the next section.

The biggest weakness of the solution is the missing graphical user interface. This is the main difference to existing closed sourced products. But it should be mentioned that some of these tools have indeed a fancy user interface but a worse scientific background. I think it's more important that core abstractions and their implementation are elaborated, a GUI can be added later. Despite these weaknesses jBPM simulation is already easier to apply and more powerful than simple spreadsheet calculations for use cases like staff planning or evaluation of process alternatives.

Since the market for BPM tools is still growing rapidly and the demand for tools supporting the whole BPM life-cycle is increasing, I expect BPS tools to be more important in future. jBPM simulation is the first one available open source and I hope it will be used in a lot of simulation projects in future.

[1] see section 3.4.3

6.2 Future Work

As mentioned, the simulation tool is not yet complete. The following list contains known shortcomings as well as improvements ideas I can think of for future work:

Optimization of Parameters: As described in section 3.5, the tool can not optimize parameter configuration, so this task is left to the user of the tool. Because most simulation projects will need such a functionality, an algorithm should be included which is sufficient for "normal" cases, like the one developed for the dataphone[2].

More powerful resource configurations: Currently the resource pools can only contain a fixed number of resources. A more sophisticated configuration is required, covering for example shift calenders. Additionally the problem of deadlocks[3] if a node needs more than one resource has to be addressed.

Complete support of jBPM Features: Not all concepts of jBPM are supported by the current version of the tool. Timers, sub processes and super states are still missing.

Support for jBPM 4 and different languages: At some point in the near future, jBPM 4 will be officially released. It will support XPDL as well as BPEL beside jPDL. The simulation has to be adopted to the concepts of the Process Virtual Machine (PVM)[4] and it has to examined, which parts of the scenario configurations can be made language independent and what has to be added to support XPDL and BPEL.

Better reporting: Reporting is an important issue in BPS. The reports provided in the current version are very basic and can be seen as examples. Better reports have to be developed and a more open interface should be included, for example an export of the results to a known spreadsheet format like Excel tables. This would enable power users to evaluate the results with the tool of their choice.

Graphical User Interface: A GUI is very important to enable people to use the simulation tool. A lot of configurations use node names, so support is necessary to avoid typos and reduce the work to maintain the XML files. A sophisticated interface can leverage the simulation to a wider range of people by hiding a lot of complexity.

Include more statistics: The tool doesn't include all statistical concepts described in this thesis. It is desirable that it supports for example automatic warm-up detection, automatic sample-size and confidence-interval calculation, statistics of multiple runs and so on. Some of the formulas or algorithms required are already implemented in the tool or the prototype, but not yet wired together to be ready to use out-of-the-box.

[2] see section 5.2
[3] see section 4.3.1
[4] see section 4.1.1

More performance indicators: Section 2.3.3 describes possible performance indicators. Not all of them are supported by the tool yet, transport times or the probability of waste are skipped for example.

Process animation and debugging: One interesting feature of some existing BPM tools is to animate single process runs or to allow the debugging of them. Even if this topic was skipped in this thesis, it would be a valuable feature of the jBPM process designer.

Beside these concrete improvement proposals it would be very interesting to do serious research on the topic of automatic business process optimization like introduced in section 3.5. This could be the content of a successive master thesis using my work as basis.

Appendix A

Details of developed BPS tool and source code

A.1 Simulation configuration

A.1.1 Distributions

The distribution configuration consists of two parts:

- The distribution itself identified by name, shown exemplary in listing A.1, and
- the usage of the distribution, an example can be found in listing A.2.

By doing so, distributions can be used at different places or reused in different scenarios. Distributions are also used behind the scenes if probabilities are configured per outgoing transitions as shown in listing A.3.

Listing A.1: Configuration of distributions.

```
<distribution name="dist1" sample-type="real" type="erlang"
              mean="94.3247"/>
<distribution name="dist2" sample-type="real" type="normal"
              mean="28.636717" standardDeviation="17.576881"/>
```

Listing A.2: Configuration of distribution usage.

```
<scenario ...>
  ...
  <sim-process name="processname">
    <process-overwrite start-distribution="dist1"/>
    <state-overwrite state-name="wait for parcel"
       time-distribution="ReturnDefectiveGoods.wait for parcel" />
    ...
```

Listing A.3: Configuration of leaving transition probabilities.

```
<decision name="ordered within the last two weeks?"
           expression="#{decisionOne}">
```

```
    <transition to="transfer shipping costs"
                name="YES" probability="100" />
    <transition to="wait for parcel"
                name="NO" probability="95" />
</decision>
```

A.1.2 Resource pools

Listing A.4 shows an example of a resource pool configuration inside of the business process and how the pool can be assigned to a special task. A task-node should have an assignment, which can be a swimlane. If the swimlane is a valid resource pool name, the resource with that name is automatically added to the required ones with a default amount of one.

Listing A.4: Configuration and usage of resource pools.

```
<resource-pool name="worker"
               pool-size="5" costs-per-time-unit="0.025"/>
<resource-pool name="clerk"
               pool-size="2" costs-per-time-unit="0.011"/>
<resource-pool name="machine"
               pool-size="7" costs-per-time-unit="0.005"/>
...
<task-node name='processing with machine'>
  <!-- using resource pool 'worker' with amount '1' -->
  <task swimlane='worker' />

  <resource-needed pool='big machine' amount='2' />
  <transition to='next' />
</task-node>
```

Listing A.5 shows the resource configuration in the experiment configuration file and available attributes to control the resource cost calculation.

Listing A.5: Configuration of resource pools and costs in experiment XML.

```
<experiment name='ReturnDefectiveGoods'
          time-unit='second'
          ...
          currency='EUR'
          unutilized-time-cost-factor='0.0'>
...
<scenario name="status_quo_normal_case" base-scenario="status_quo">
  <resource-pool name="tester" pool-size="5"
                 costs-per-time-unit="0.025"/>
  ...
  <task-overwrite task-name='processing with machine'
                  time-distribution='...'>
    <resource-needed pool='big machine' amount='2' />
  </task-overwrite>
  ...
</scenario>
```

A.1.3 Data source and data filter

Listing A.6 shows an exemplary configuration of data sources and filters, like introduced in section 4.3.1, in the scenario configuration.

Listing A.6: Configuration of data source and filter.

```
<scenario name="test" execute="true">
  ...
  <variable-source name="orders"
    handler="org.jbpm.sim.tutorial.TutorialProcessVariableSource" />
  <variable-filter name="orders"
    handler="org.jbpm.sim.tutorial.TutorialProcessVariableFilter" />

  <sim-process path="...datasource/processdefinition.xml">
    <process-overwrite start-distribution="start">
      <use-variable-source name="orders" />
    </process-overwrite>
    <task-overwrite task-name="change return order"
                    time-distribution="change return order">
      <use-variable-filter name="orders" />
    </task-overwrite>
    ...
  </sim-process>
</scenario>
```

A.1.4 Called Services / Actions

The simulation tool provides different possibilities to handle situations, in which the process controls foreign systems. In jBPM these parts are called "usercode", which is implemented either as Java classes called "ActionHandler" or scripts[1]. To control execution of usercode in simulation runs the tool provides an additional attribute called "simulation" with the values "execute" or "skip" with "skip" as default. If the user wants usercode to be executed, he has to provide that attribute like shown in listing A.7.

Listing A.7: Configuration of usercode execution during simulation.

```
<node name="normal actions">
  <transition to="special simulation actions">
    <!-- This one is not executed because of default: -->
    <action name="action 1" class="...TestAction" />
    <action name="action 2" class="...TestAction"
            simulation='skip' />
    <action name="action 3" class="...TestAction"
            simulation='execute' />

    <script name ='not executed in sim 1'>
       <expression>...</expression>
    </script>
    <script name ='not executed in sim 2' simulation='skip'>
       <expression>...</expression>
    </script>
```

[1]The scripts must be written in Beanshell

```
    <script name ='executed in sim 1' simulation='execute'>
       <expression>...</expression>
    </script>
  </transition>
</node>
```

The tool also provides an option to add special usercode only for simulation. This is useful to simulate result data or to change process variables in the case that a call to an external system is suppressed for example. This can be configured in two ways, either special "simulation-action" and "simulation-script" elements are added to the process definition or a "simulation-class" attribute is added to the normal action element. In the later case, this class is executed instead of the original implementation. This is possible with the "simulation-expression" inside of the "script" accordingly. Listing A.7 show the configuration options.

Listing A.8: Configuration of special usercode for simulation.

```
<node name="special simulation actions">
  <transition to="task">
    <action name="sim-action 1" class="...TestAction"
            simulation-class='...SimTestAction' />
    <simulation-action name='sim-action 2' class='...SimTestAction' />

    <script name ='simulation script 1'>
       <expression>...</expression>
       <simulation-expression>...</simulation-expression>
    </script>
    <simulation-script name ='simulation script 2'>
       <expression>...</expression>
    </simulation-script>
  </transition>
</node>
```

jBPM also knows "mail-nodes" which send emails. These nodes are always skipped in simulation, because it doesn't make sense to send mails in simulation experiments in most of the cases. If the mails are used to communicate with foreign systems and important for the simulation to work well, the user has to implement them himself or to activate mail nodes in the node configuration[2].

The last possibility to influence usercode execution are two provided interfaces: "SimulationNoop" and "SimulationHandler". If a jBPM action handler implements the first, the execution is always skipped in simulation, independent on the configuration in the XML file. The second interface defines a method "simExecute", which is executed instead of the normal "execute" method in simulation runs. This makes it possible, to implement real-live and simulation behavior in the same Java class.

The configuration in the XML file or the interface can make sense. The first is more oriented on the process model and the latter on the implementation.

[2] org/jbpm/sim/simulation.node.types.xml

A.1.5 Business figures / Costs

As you can see in the configuration example in listing A.9, business figures can be realized in two different ways. Either the user implements the BusinessFigureCalculator interface and hooks in the resulting calculator class in the configuration or an expression to calculate the figure is provided directly in the configuration via expression language. For details on the expression language, which is very similar to the JSF expression language, see [JBo07], section 16.3.

Listing A.9: Configuration of business figures.

```
...
<scenario name="slow invoice payment">
  <business-figure
     name="missed discount"
     type="costs"
     automatic-calculation="none|process-start|process-end"
     expression="#{order.discountForFastPayment}" />
  <business-figure
     name="missed discounts 2"
     type="costs"
     automatic-calculation="none|process-start|process-end"
     handler="org...tutorial.TutorialBusinessFigureCalculator"/>
  ...
</scenario>
```

The business figures can be plugged in at different places in the business process as shown in listing A.10. They are evaluated at the time the process runs through the corresponding node. Another possibility to apply business figures is to configure the auto calculation property, then the figures are automatically evaluated at the process start or end[3]. If the type of the business figure is "costs", it is automatically added to the overall costs for a scenario at the end. Other types of figures can be queried separately from the "ScenarioReport" class.

Listing A.10: Usage of business figures.

```
<sim-process name='test'>
  <node-overwrite node-name='node1'>
    <calculate-business-figure name='missed discount' />
  </node-overwrite>
</sim-process>
```

If the business figures should be hooked in the process definition directly, an existing action handler of the simulation tool can be used as shown in listing A.11.

Listing A.11: Usage of business figures via action class.

```
<simulation-action class='org.jbpm.sim.kpi.BusinessFigureAction'>
   <name>missed discount</name>
</simulation-action>
```

[3]This feature is not implemented in the version developed in this thesis.

A.1.6 Experiment configuration

As mentioned in section 4.3.1 the configuration option can be included in the process definition, but the more recommended way is an own experiment configuration XML file. Listing A.12 shows a part of an experiment configuration. The possibility to use abstract scenarios[4], which have the attribute execute set to false, is included in this example.

Another configuration parameter shown is the "reset-time", which can be used to configure the point in model time the steady-state begins. At this point all statistical counter are reseted.

Listing A.12: Experiment configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment name='ReturnDefectiveGoods'
          time-unit='second'
          run-time='28800'
          reset-time='50'
          real-start-time='30.03.1980 00:00:00:000'
          currency='EUR'
          unutilized-time-cost-factor='0.0'>
  <!-- 28800 seconds = 8 hours = 1 working day -->

<scenario name="status_quo" execute="false">
  <distribution name="start"
         sample-type="real" type="erlang" mean="95"/>
  ...
  <sim-process path="/ReturnDefectiveGoods/processdefinition.xml">
    <process-overwrite start-distribution="start"/>
   ...
    <task-overwrite task-name="quick test"
                    time-distribution="quick test">
      <transition name="no defect" probability="165"/>
      <transition name="defect approved" probability="30"/>
    </task-overwrite>
    ...
  </sim-process>
</scenario>

<scenario name="status_quo_normal_case" execute="true"
          base-scenario="status_quo">
  <resource-pool name="tester"     pool-size="5"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="clerk"      pool-size="2"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="accountant" pool-size="2"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
</scenario>
```

[4]This is a similar concept as abstract classes in Java.

A.1.7 Optimization example: Brute force guessing

Listing A.13 shows the implementation of an own "ExperimentReader", which creates multiple scenarios on the fly by a simple brute force approach. The technique can be used to build much more sophisticated strategies as well, which may include reading additional configuration parameters from the experiment configuration. The code provided here is only a simple demonstration on how to implement such functionality.

Listing A.13: Extension of the jPDL reader to generate scenario configurations by a brute force approach.

```
public class StaffingExperimentReader extends ExperimentReader {

  private long addCount = 2;

  // flag to avoid endless loop
  private boolean afterScenarioReadActive = false;


  public StaffingExperimentReader(InputSource inputSource) {
    super(inputSource);
  }

  public StaffingExperimentReader(String experimentXml) {
    super(experimentXml);
  }

  protected void afterScenarioRead(JbpmSimulationScenario scenario,
            Element scenarioElement, Element baseScenarioElement) {
    if (afterScenarioReadActive)
      return;

    afterScenarioReadActive = true;
    HashMap pools = new HashMap();
    HashMap costs = new HashMap();

    Iterator poolElementIter =
       scenarioElement.elementIterator("resource-pool");
    while (poolElementIter.hasNext()) {
      Element resourcePoolElement = (Element) poolElementIter.next();
      String poolName = resourcePoolElement.attributeValue("name");
      String poolSizeText =
         resourcePoolElement.attributeValue("pool-size");
      Integer poolSize = new Integer(poolSizeText);
      pools.put(poolName, poolSize);
      costs.put(poolName, readCostPerTimeUnit(resourcePoolElement));
    }


    // add more scenarios with more people
    // (so the provided people count is the lower limit)

    for (int add = 1; add <= addCount; add++) {
      JbpmSimulationScenario generatedScenario =
         readScenario(scenarioElement, baseScenarioElement);
```

```
      generatedScenario.changeName(scenario.getName() + "+" + add);

      for (Iterator iterator = pools.keySet().iterator();
           iterator.hasNext();) {
        String name = (String) iterator.next();
        Integer size = (Integer) pools.get(name);
        size = new Integer(size.intValue() + add);
        generatedScenario.addResourcePool(
           name, size, (Double)costs.get(name));
      }

      addScenario(generatedScenario);
    }
    afterScenarioReadActive = false;
  }

  protected void beforeScenarioRead(JbpmSimulationScenario scenario,
      Element scenarioElement, Element baseScenarioElement) {
  }

  public static void main(String[] args) {
    String experimentConf = "/org/.../simulationExperiment.xml";

    StaffingExperimentReader reader = new StaffingExperimentReader(
        new InputSource(...getResourceAsStream(experimentConf)));

    JbpmSimulationExperiment experiment = reader.readExperiment();
    experiment.setWriteDesmojHtmlOutput(true);
    experiment.setRememberEndedProcessInstances(false);

    experiment.run(); // can take some time

    ExperimentReport report = experiment.getReport();

    ScenarioComparisionReport r =
       new ScenarioComparisionReport(report);
    r.show();
  }
}
```

A.2 Reports

A.2.1 DESMO-J HTML report example

ReturnDefectiveGoods.status_quo_normal_case Report
Model JBoss jBPM simulation model

Description

jBPM-Simulation Report drawn at 28800.0. Last reset at 0.0.

Tallies

Title	(Re)set	Obs	Mean	Std.Dev	Min	Max
Waiting before 'wait for parcel'	299.4063881043369	264	0.0	0.0	0.0	0.0
Waiting before 'quick test'	345.59959235428613	261	162.97514	173.43446	0.0	638.1967
Waiting before 'transfer shipping costs'	364.5959564756849	130	679.8215	407.00192	0.0	1382.16518
Waiting before 'extended technical test'	544.0194939287724	169	2028.64238	1174.86629	0.0	5186.37099
Waiting before 'refund'	564.0743508479242	182	707.54574	402.62746	0.0	1461.30459
Cycle times for 'ReturnDefectiveGoods'	731.3355875284932	207	3652.96618	1797.44948	268.0	8057.0
Waiting before 'send back goods'	3264.3469454646643	26	94.63805	145.51102	0.0	494.83563

Counts

Title	(Re)set	Obs	Current Value	Min	Max
Amount finished in end state 'ReturnDefectiveGoods | Approved'	731.3355875284932	182	182	0	182
Amount finished in end state 'ReturnDefectiveGoods | Denied'	3874.5268034967953	25	25	0	25

Queues

Title	Qorder	(Re)set	Obs	QLimit	Qmax	Qnow	Qavg.	Zeros	avg.Wait	refus.
Resource pool 'tester'	FIFO	0.0	169	5	5	0	0.2252	156	38.37696	0
Resource pool queue for 'tester'	FIFO	0.0	156	unlimit.	41	41	15.50243	0	2197.69591	0
Resource pool 'clerk'	FIFO	0.0	261	2	2	0	0.3811	183	42.05271	0
Resource pool queue for 'clerk'	FIFO	0.0	183	unlimit.	8	2	1.48977	0	232.43996	0
Resource pool 'accountant'	FIFO	0.0	312	2	2	0	0.06281	301	5.79826	0
Resource pool queue for 'accountant'	FIFO	0.0	301	unlimit.	17	11	7.73583	0	721.42897	0
Resource pool 'dispatcher'	FIFO	0.0	26	1	1	0	0.69844	11	773.65542	0
Resource pool queue for 'dispatcher'	FIFO	0.0	11	unlimit.	1	0	0.08544	0	223.68994	0

Distributions

Title	(Re)set	Obs	Type	Parameter 1	Parameter 2	Seed
ReturnDefectiveGoods.start	0.0	273	Erlang	1	95.0	71529105
Selecting outgoing transition of TaskNode(transfer shipping costs)	0.0	128	Real Uniform	0.0	100.0	11401638
Selecting outgoing transition of TaskNode(Refund)	0.0	182	Real Uniform	0.0	175.0	78135182
Selecting outgoing transition of SimState(wait for parcel)	0.0	263	Real Uniform	0.0	195.0	21201764
Selecting outgoing transition of TaskNode(quick test)	0.0	259	Real Uniform	0.0	195.0	66351764
Selecting outgoing transition of TaskNode(extended technical test)	0.0	164	Real Uniform	0.0	165.0	38755291
Selecting outgoing transition of TaskNode(send back goods)	0.0	25	Real Uniform	0.0	20.0	95129352
Selecting outgoing transition of SimDecision(ordered within the last two weeks?)	0.0	272	Real Uniform	0.0	195.0	59119942
ReturnDefectiveGoods. wait for parcel	0.0	264	Normal	28.0	17.0	91080577
ReturnDefectiveGoods. transfer shipping costs	0.0	130	Normal	180.0	30.0	43946618
ReturnDefectiveGoods. quick test	0.0	261	Normal	180.0	60.0	44944377
ReturnDefectiveGoods. extended technical test	0.0	169	Normal	732.2485	448.1038	38440435
ReturnDefectiveGoods. send back goods	0.0	26	Normal	325.5	182.0718	9050595
ReturnDefectiveGoods. refund	0.0	182	Normal	180.0	30.0	62542346

created using DESMO-J Version 2.1.1 at Tue Dec 18 15:15:57 CET 2007 - DESMO-J is licensed under Apache License, Version 2.0

A.2.2 JasperReport sources

Listing A.14 shows how reports work in JasperReports. The result of the simulation is passed as parameter into the main report and collection data sources are created for the subreports. These data sources provide row based data like required by the subreports. The listing also shows how simple attributes can be shown in a report.

All reports work in this manner, so it is easy to add, change or remove subreports from a main report or to change them completely. Other details of the implementation or the integration of reports in the simulation tool are skipped here. A good starting point to get an impression is the source code of the "ScenarioDetailsReport" and "ScenarioComparisionReport" class. These classes load the report definition, provide the necessary data and show it on screen or save it as PDF file.

Listing A.14: Source code of scenario comparison report.

```
<jasperReport name="ScenarioComparison" ...>
  ...
  <parameter name="EXPERIMENT_REPORT" class="java.lang.Object"/>
  ...
  <subreport>
    ...
    <dataSourceExpression>
        <![CDATA[  new JRBeanCollectionDataSource(
         ((org.jbpm.sim.report.ExperimentReport)$P{EXPERIMENT_REPORT})
                                          .getScenarioReports() ) ]]>
    </dataSourceExpression>
    <subreportExpression>
      <![CDATA[$P{SUBREPORT_WaitingTimeBeforeTaskComparison}]]>
    </subreportExpression>
  ...
</jasperReport>
...
<jasperReport name="WaitingTimeBeforeTaskComparison" ...>
  ...
  <field name="simulationRunTime" class="java.lang.Double" />
  <field name="scenarioName" class="java.lang.String" />
  <field name="cost" class="java.lang.Double" />
  <field name="resourceAmount" class="java.lang.Integer" />
  <field name="averageResourceUtilization" class="java.lang.Double" />
  <field name="cycleTimesValueStatistics"
                                        class="java.util.Collection" />
  <field name="worstAverageResourceWaitingTime"
              class="org.jbpm.sim.report.dto.QueueStatisticsResult" />
  ..
</jasperReport>
```

To support fast previews of subreports in the graphical designer, the simulation tool provides sample data sources for the important objects, which return hard coded data. The data sources can be used in iReport, the GUI for designing reports, which allows to use the preview functionality. An example is shown in listing A.15.

Listing A.15: JasperReport data source with sample data.

```
public class SampleJRDataSourceFactory {
  public static ValueStatisticResult[] createTaskDistributionArray() {
```

```
    ValueStatisticResult row1 = new ValueStatisticResult(
       "Task 1", "Scenario 1", 373.0, 0, 0, 0, 0);
    ValueStatisticResult row2 = new ValueStatisticResult(
       "Task 2", "Scenario 1", 100.0, 0, 0, 0, 0);
    ...

    return new ValueStatisticResult[] {row1, row2, ...};
  }
  ...
```

A.2.3 Scenario report example

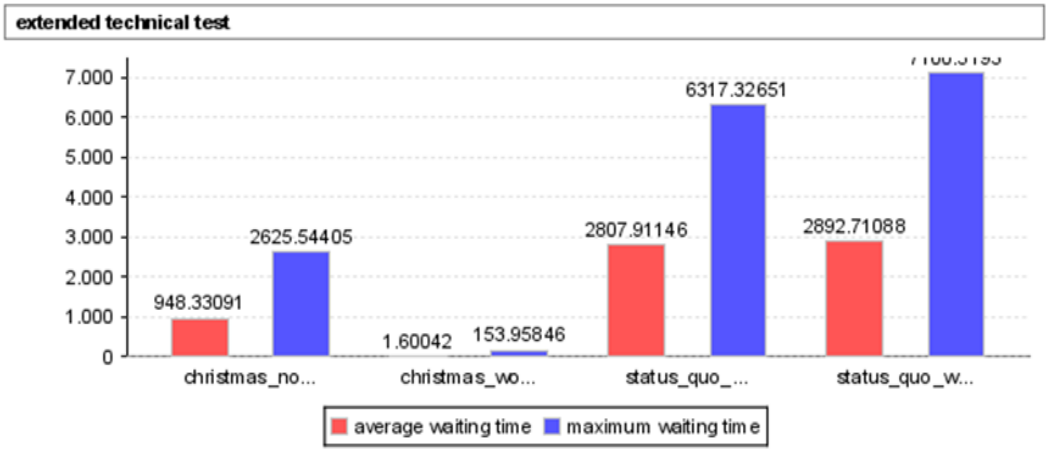
JBoss jBPM Business Process Simulation Report - Scenario comparision

Experiment: ReturnDefectiveGoods
Generated at Tue Jan 22 16:31:07 CET 2008

Overview

Scenario	avg.resource utilization	costs	process cycle times	
status_quo_worst_case	78,62 %	5.351	ReturnDefectiveGoods	3.740
Bottleneck 'tester' with wait time:		3.110		
status_quo_normal_case	86,89 %	5.355	ReturnDefectiveGoods	4.044
Bottleneck 'tester' with wait time:		2.981		
christmas_normal_case	85,78 %	11.256	ReturnDefectiveGoods	1.985
Bottleneck 'tester' with wait time:		1.125		
christmas_worst_case	64,14 %	12.135	ReturnDefectiveGoods	1.228
Bottleneck 'dispatcher' with wait time:		615		

Distribution of average waiting time before states/tasks



Scenario name	Mean	Std Dev	Minimum	Maximum	Observations
christmas_normal_case	948,33	792,33	0,00	2625,54	376
christmas_worst_case	1,80	11,72	0,00	153,96	417
status_quo_normal_case	2807,91	1715,61	0,00	6317,33	172
status_quo_worst_case	2892,71	1834,30	0,00	7100,52	172

Figure A.1: Scenario report example, page 1, overview.

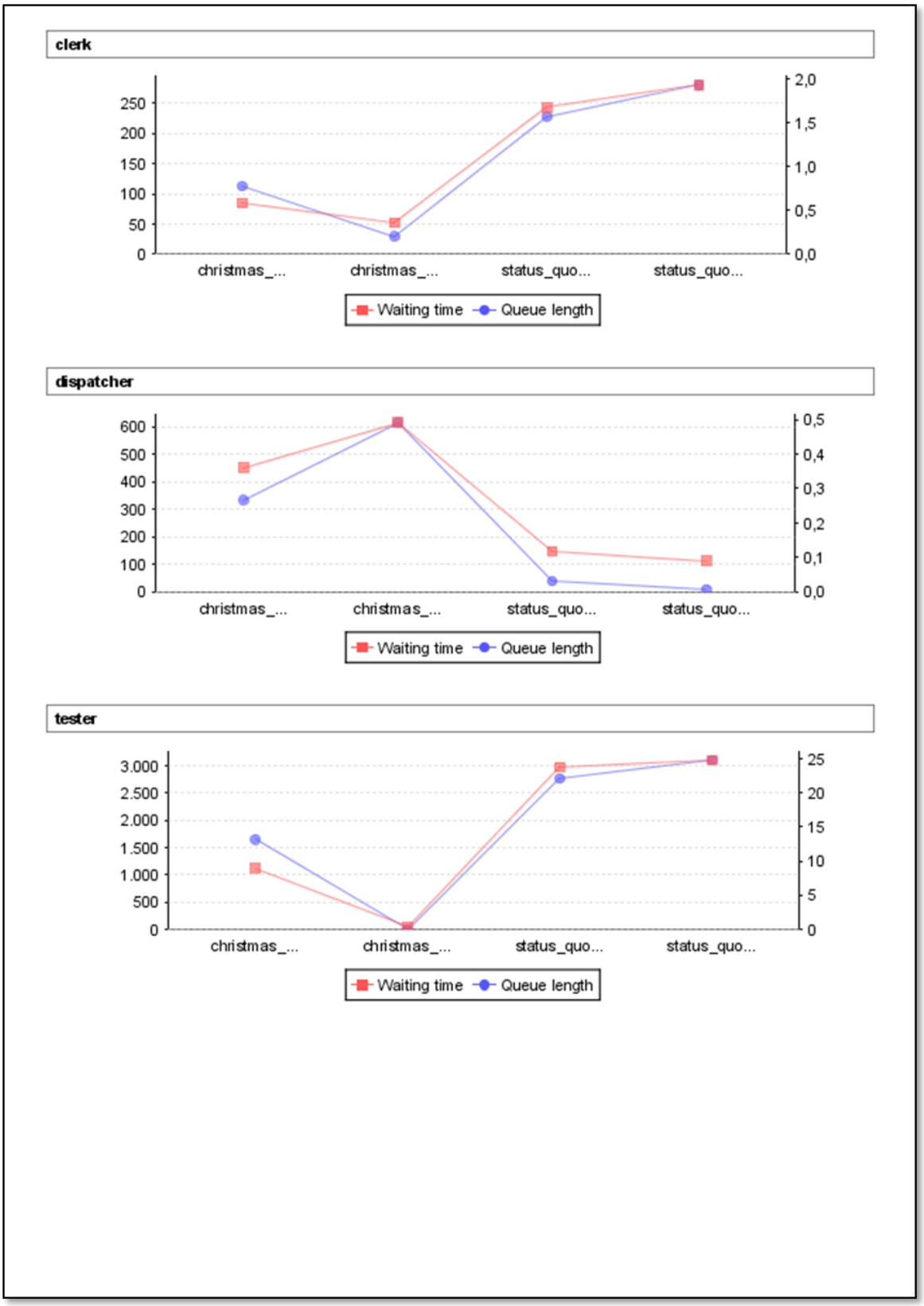


Figure A.2: Scenario report example, page 5, waiting times for resources.

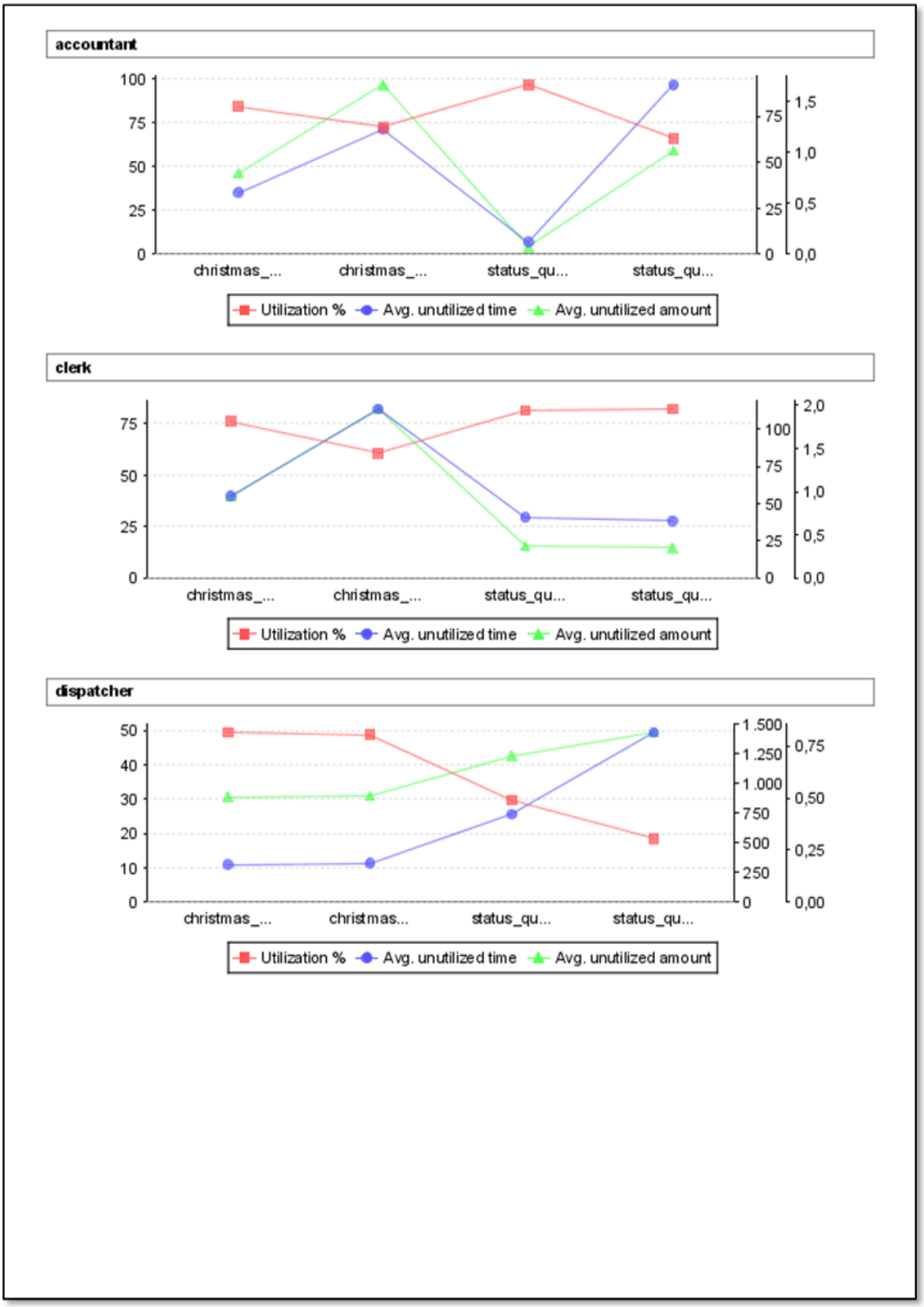


Figure A.3: Scenario report example, page 6, resource utilization.

A.2.4 Experiment report example

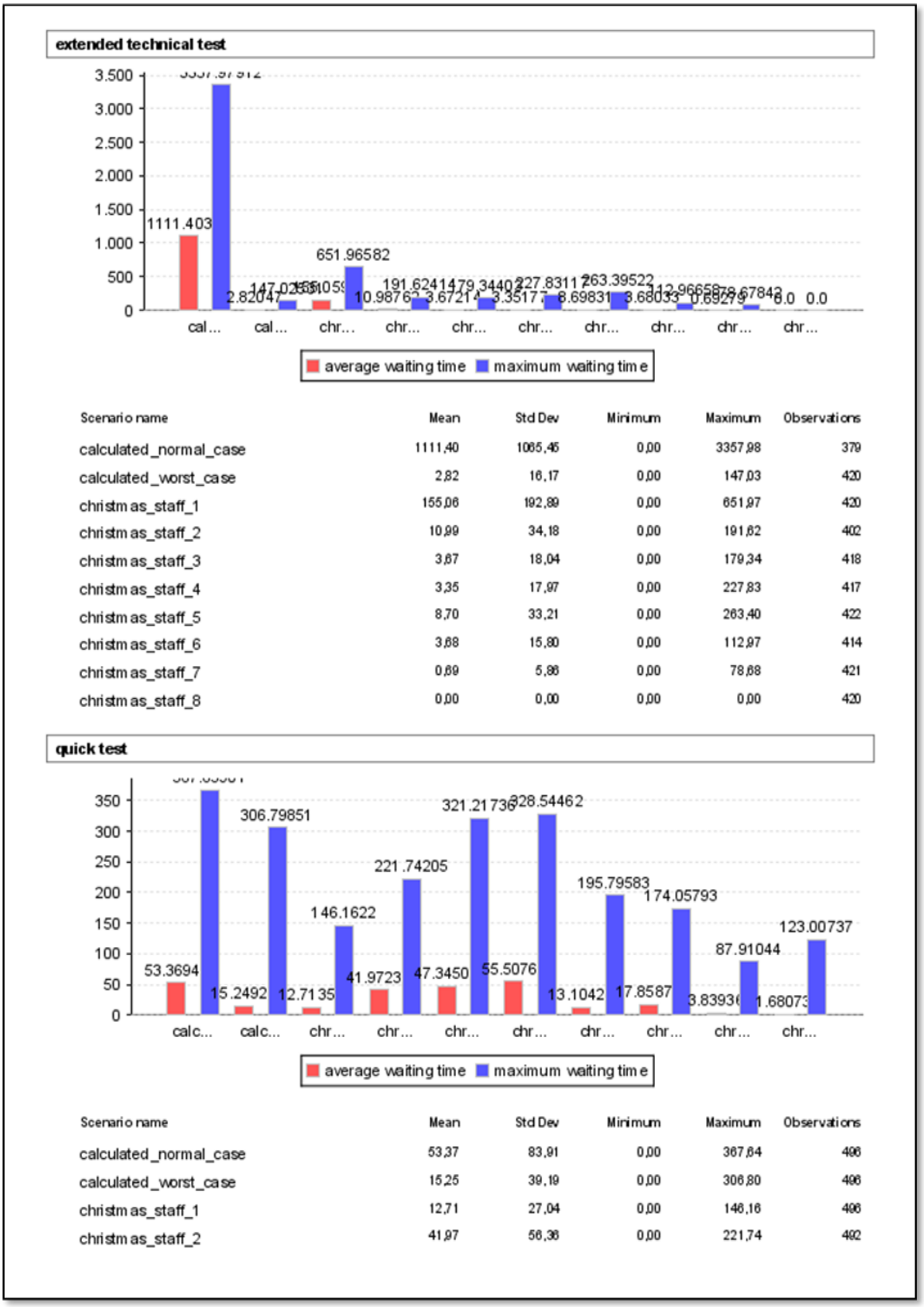
JBoss jBPM Business Process Simulation Report - Scenario comparision

Experiment: ReturnDefectiveGoods
Generated at Tue Jan 22 16:58:14 CET 2008

Overview

Scenario	avg.resource utilization	costs	process cycle times	
calculated_normal_case	85,64 %	12.031	ReturnDefectiveGoods	2.150
Bottleneck 'tester' with wait time:		1.329		
christmas_staff_1	77,57 %	13.573	ReturnDefectiveGoods	1.512
Bottleneck 'tester' with wait time:		270		
christmas_staff_2	72,46 %	13.970	ReturnDefectiveGoods	1.536
Bottleneck 'accountant' with wait time:		316		
christmas_staff_3	68,42 %	14.788	ReturnDefectiveGoods	1.327
Bottleneck 'accountant' with wait time:		130		
christmas_staff_4	68,72 %	14.928	ReturnDefectiveGoods	1.268
Bottleneck 'dispatcher' with wait time:		335		
christmas_staff_5	66,24 %	14.970	ReturnDefectiveGoods	1.291
Bottleneck 'dispatcher' with wait time:		198		
christmas_staff_6	64,23 %	15.288	ReturnDefectiveGoods	1.202
Bottleneck 'dispatcher' with wait time:		141		
calculated_worst_case	64,35 %	15.468	ReturnDefectiveGoods	1.246
Bottleneck 'dispatcher' with wait time:		366		
christmas_staff_7	55,18 %	16.865	ReturnDefectiveGoods	1.183
Bottleneck 'dispatcher' with wait time:		131		
christmas_staff_8	48,19 %	18.282	ReturnDefectiveGoods	1.161
Bottleneck 'dispatcher' with wait time:		195		

Figure A.4: Experiment report example, page 1, overview.



Scenario name	Mean	Std Dev	Minimum	Maximum	Observations
calculated_normal_case	1111,40	1065,46	0,00	3357,98	379
calculated_worst_case	2,82	16,17	0,00	147,03	420
christmas_staff_1	155,06	192,89	0,00	651,97	420
christmas_staff_2	10,99	34,18	0,00	191,62	402
christmas_staff_3	3,67	18,04	0,00	179,34	418
christmas_staff_4	3,35	17,97	0,00	227,83	417
christmas_staff_5	8,70	33,21	0,00	263,40	422
christmas_staff_6	3,68	15,80	0,00	112,97	414
christmas_staff_7	0,89	5,98	0,00	78,68	421
christmas_staff_8	0,00	0,00	0,00	0,00	420

Scenario name	Mean	Std Dev	Minimum	Maximum	Observations
calculated_normal_case	53,37	83,91	0,00	367,64	496
calculated_worst_case	15,25	39,19	0,00	306,80	496
christmas_staff_1	12,71	27,04	0,00	146,16	496
christmas_staff_2	41,97	56,36	0,00	221,74	492

Figure A.5: Experiment report example, page 2, waiting times before states.

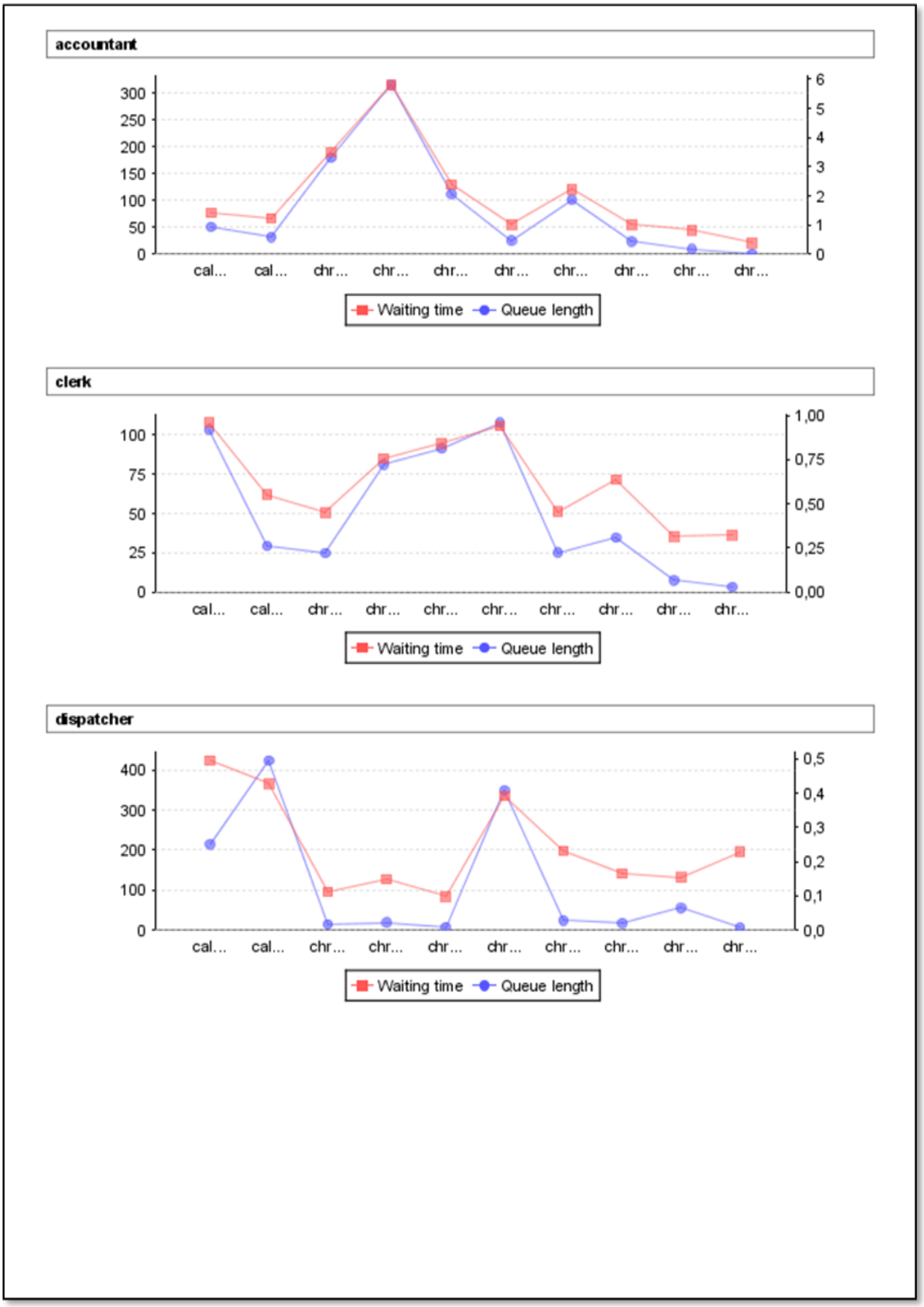


Figure A.6: Experiment report example, page 3, waiting times for resources.

A.3 Showcase

A.3.1 Process source code (jPDL)

Listing A.16 contains the complete source code of the process developed for the show case. It is used in both use cases.

Listing A.16: jPDL source code for process definition.

```
<process-definition name="ReturnDefectiveGoods">
  <swimlane name="clerk">
    <assignment pooled-actors="clerk" />
  </swimlane>
  <swimlane name="accountant">
    <assignment pooled-actors="accountant" />
  </swimlane>
  <swimlane name="tester">
    <assignment pooled-actors="tester" />
  </swimlane>
  <swimlane name="dispatcher">
    <assignment pooled-actors="dispatcher" />
  </swimlane>

  <start-state name="Return request">
    <transition name="start"
                to="ordered within the last two weeks?" />
  </start-state>

  <decision name="ordered within the last two weeks?"
            expression="#{decisionOne}">
    <transition to="transfer shipping costs" name="YES" />
    <transition to="wait for parcel" name="NO" />
  </decision>

  <task-node name="transfer shipping costs">
    <task name="transfer shipping costs"
          swimlane="accountant" />
    <transition name="done" to="wait for parcel" />
  </task-node>

  <state name="wait for parcel">
    <transition name="parcel arrived" to="quick test" />
  </state>

  <task-node name="quick test">
    <task name="quick test" swimlane="clerk" />
    <transition name="no defect"
                to="extended technical test"  />
    <transition name="defect approved"
                to="Refund" />
  </task-node>

  <task-node name="extended technical test">
    <task name="extended technical test"
          swimlane="tester" />
    <transition to="send back goods" name="no defect" />
```

```
    <transition to="Refund" name="defect approved" />
  </task-node>

  <task-node name="send back goods">
    <task name="send back goods" swimlane="dispatcher" />
    <transition to="Denied" />
  </task-node>

  <task-node name="Refund">
    <task name="refund" swimlane="accountant" />
    <transition to="Approved" />
  </task-node>

  <end-state name="Approved" />
  <end-state name="Denied" />
</process-definition>
```

A.3.2 Create faked historical data

As explained in section 4.4.3, the showcase should use historical data, even if no real historical data is available. The code in listing A.17 creates faked historical data by execute process instances programatically.

Listing A.17: Creation of faked historical data by execute process runs programmatically.

```
ProcessInstance pi = pd.createProcessInstance();
pi.getContextInstance().setVariable("decisionOne", "YES");
pi.getRootToken().signal(); // start state
simulateRandomTime();

TaskInstance ti1 = (TaskInstance) pi.getTaskMgmtInstance().
    getTaskInstances().iterator().next();
ti1.start("me"); // task "transfer shipping costs"
simulateRandomTime();
ti1.end(); // task "transfer shipping costs"
simulateRandomTime();

pi.getRootToken().signal(); // state "wait for parcel"
simulateRandomTime();

TaskInstance ti2 = (TaskInstance) pi.getTaskMgmtInstance().
    getUnfinishedTasks(pi.getRootToken()).iterator().next();
ti2.start("me"); // task "quick test"
simulateRandomTime();
ti2.end("defect approved"); // task "quick test"
simulateRandomTime();

TaskInstance ti3 = (TaskInstance) pi.getTaskMgmtInstance().
    getUnfinishedTasks(pi.getRootToken()).iterator().next();
ti3.start("me");  // task "refund"
simulateRandomTime();
ti3.end(); // task "refund"
ctx.save(pi);
```

A.3.3 Experiment configuration for Use Case 1

Listing A.18 contains the complete experiment configuration for use case 1 of the showcase as described in section 4.4.4. Four scenarios, differing in the resource numbers, are created based on two abstract main scenarios, the normal case and christmas. Two abstract scenarios are required, because they use different distribution configurations.

Listing A.18: Experiment configuration for showcase, use case 1.

```
<experiment name='ReturnDefectiveGoods' time-unit='second'
    run-time='28800' real-start-time='30.03.1980 00:00:00:000'
    currency='EUR' unutilized-time-cost-factor='0.0'>
    <!-- 28800 seconds = 8 hours = 1 working day -->
 <scenario name="status_quo" execute="false">
  <distribution name="ReturnDefectiveGoods.start"
    sample-type="real" type="erlang" mean="95"/>
  <distribution name="wait for parcel" sample-type="real"
    type="normal" mean="28" standardDeviation="17"/>
  <distribution name="transfer shipping costs" sample-type="real"
    type="normal" mean="180" standardDeviation="30"/>
  <distribution name="quick test" sample-type="real"
    type="normal" mean="180" standardDeviation="60"/>
  <distribution name="extended technical test" sample-type="real"
    type="normal" mean="732.2485" standardDeviation="448.1038"/>
  <distribution name="send back goods" sample-type="real"
    type="normal" mean="325.5" standardDeviation="182.0718"/>
  <distribution name="refund" sample-type="real"
    type="normal" mean="180" standardDeviation="30"/>
  <sim-process path="/.../ReturnDefectiveGoods/processdefinition.xml">
    <process-overwrite start-distribution="start"/>
    <state-overwrite state-name="wait for parcel"
                     time-distribution="wait for parcel">
      <transition name="parcel arrived" probability="195"/>
    </state-overwrite>
    <decision-overwrite
          decision-name="ordered within the last two weeks?">
      <transition name="YES" probability="100"/>
      <transition name="NO" probability="95"/>
    </decision-overwrite>
    <task-overwrite task-name="transfer shipping costs"
                    time-distribution="transfer shipping costs">
      <transition name="done" probability="100"/>
    </task-overwrite>
    <task-overwrite task-name="quick test"
                    time-distribution="quick test">
      <transition name="no defect" probability="165"/>
      <transition name="defect approved" probability="30"/>
    </task-overwrite>
    <task-overwrite task-name="extended technical test"
                    time-distribution="extended technical test">
      <transition name="no defect" probability="20"/>
      <transition name="defect approved" probability="145"/>
    </task-overwrite>
    <task-overwrite task-name="send back goods"
                    time-distribution="send back goods">
      <transition probability="20"/>
```

```
    </task-overwrite>
    <task-overwrite task-name="refund" time-distribution="refund">
      <transition probability="175"/>
    </task-overwrite>
  </sim-process>
</scenario>

<scenario name="christmas" execute="false">
  <distribution name="start"
    sample-type="real" type="erlang" mean="55"/>
  <distribution name="wait for parcel" sample-type="real"
     type="normal" mean="28" standardDeviation="17"/>
  <distribution name="transfer shipping costs" sample-type="real"
     type="normal" mean="180" standardDeviation="30"/>
  <distribution name="quick test" sample-type="real"
     type="normal" mean="180" standardDeviation="60"/>
  <distribution name="extended technical test" sample-type="real"
     type="normal" mean="732.2485" standardDeviation="448.1038"/>
  <distribution name="send back goods" sample-type="real"
     type="normal" mean="325.5" standardDeviation="182.0718"/>
  <distribution name="refund" sample-type="real"
     type="normal" mean="180" standardDeviation="30"/>
  <sim-process path="/.../ReturnDefectiveGoods/processdefinition.xml">
    <process-overwrite start-distribution="start"/>
    <state-overwrite state-name="wait for parcel"
                     time-distribution="wait for parcel">
      <transition name="parcel arrived" probability="195"/>
    </state-overwrite>
    <decision-overwrite
          decision-name="ordered within the last two weeks?">
      <transition name="YES" probability="100"/>
      <transition name="NO" probability="95"/>
    </decision-overwrite>
    <task-overwrite task-name="transfer shipping costs"
                    time-distribution="transfer shipping costs">
      <transition name="done" probability="100"/>
    </task-overwrite>
    <task-overwrite task-name="quick test"
                    time-distribution="quick test">
      <transition name="no defect" probability="165"/>
      <transition name="defect approved" probability="30"/>
    </task-overwrite>
    <task-overwrite task-name="extended technical test"
                    time-distribution="extended technical test">
      <transition name="no defect" probability="20"/>
      <transition name="defect approved" probability="145"/>
    </task-overwrite>
    <task-overwrite task-name="send back goods"
                    time-distribution="send back goods">
      <transition probability="20"/>
    </task-overwrite>
    <task-overwrite task-name="refund"
                    time-distribution="refund">
      <transition probability="175"/>
    </task-overwrite>
```

```
  </sim-process>
 </scenario>

 <scenario name="status_quo_normal_case" execute="true"
          base-scenario="status_quo">
  <resource-pool name="tester" pool-size="5"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="clerk" pool-size="2"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="accountant" pool-size="2"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
 </scenario>

 <scenario name="status_quo_worst_case" execute="true"
          base-scenario="status_quo">
  <resource-pool name="tester" pool-size="5"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="clerk" pool-size="2"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="accountant" pool-size="3"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
 </scenario>

 <scenario name="christmas_normal_case" execute="true"
          base-scenario="christmas">
  <resource-pool name="accountant" pool-size="5"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="clerk" pool-size="4"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="tester" pool-size="11"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="dispatcher" pool-size="1"
                costs-per-time-unit="0.013888889"/>
 </scenario>

 <scenario name="christmas_worst_case" execute="true"
          base-scenario="christmas">
  <resource-pool name="accountant" pool-size="6"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="clerk" pool-size="5"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="tester" pool-size="18"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
 </scenario>

</experiment>
```

A.3.4 Experiment configuration for Use Case 2

The second use case is the evaluation of an alternative process design. The source code of the alternative process is shown in listing A.19 and the experiment configuration in A.20. The source code of the original process is shown in listing A.16 on page 99.

Listing A.19: Alternative process source code (jPDL).

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition name="ReturnDefectiveGoodsAlternative">

  <swimlane name="clerk">
    <assignment pooled-actors="clerk" />
  </swimlane>
  <swimlane name="accountant">
    <assignment pooled-actors="accountant" />
  </swimlane>
  <swimlane name="tester">
    <assignment pooled-actors="tester" />
  </swimlane>
  <swimlane name="dispatcher">
    <assignment pooled-actors="dispatcher" />
  </swimlane>

  <start-state name="Return request">
    <transition name="start" to="ordered within the last two weeks?"/>
  </start-state>


  <decision name="ordered within the last two weeks?">
    <handler class="" />
    <transition to="transfer shipping costs" name="YES" />
    <transition to="wait for parcel" name="NO" />
  </decision>

  <task-node name="transfer shipping costs">
    <task name="transfer shipping costs" swimlane="accountant" />
    <transition name="done" to="wait for parcel" />
  </task-node>

  <state name="wait for parcel">
    <transition to="should be checked?" name="parcel arrived" />
  </state>

  <task-node name="extended technical test">
    <task name="extended technical test" swimlane="tester" />
    <transition to="send back goods" name="no defect" />
    <transition to="Refund" name="defect approved" />
  </task-node>

  <task-node name="send back goods">
    <task name="send back goods" swimlane="dispatcher" />
    <transition to="Denied" />
  </task-node>

  <task-node name="Refund">
```

```
    <task name="refund" swimlane="accountant" />
    <transition to="Approved" />
  </task-node>

  <decision name="should be checked?">
    <transition to="extended technical test" name="check" />
    <transition to="unknown goods status" name="no check" />
  </decision>

  <node name="unknown goods status">
    <transition to="Refund" />
  </node>

  <end-state name="Approved" />
  <end-state name="Denied" />
</process-definition>
```

Listing A.20: Experiment configuration for showcase, use case 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<experiment name='ReturnDefectiveGoods' time-unit='second'
  run-time='28800' real-start-time='30.03.1980 00:00:00:000'
  currency='EUR'unutilized-time-cost-factor='0.0'>
  <!-- 28800 seconds = 8 hours = 1 working day -->

<scenario name="status_quo" execute="true">
  <business-figure name = "defect goods"
       type = "costs"
       automatic-calculation = "none"
       handler = "org.jbpm.sim.tutorial.DefectGoodsCostsCalculator" />
  <data-source name="return orders"
     handler="org.jbpm.sim.tutorial.TutorialDataSource" />

  <distribution name="start"
    sample-type="real" type="erlang" mean="95"/>

  <distribution name="wait for parcel" sample-type="real"
    type="normal" mean="28" standardDeviation="17"/>
  <distribution name="transfer shipping costs" sample-type="real"
    type="normal" mean="180" standardDeviation="30"/>
  <distribution name="quick test" sample-type="real"
    type="normal" mean="180" standardDeviation="60"/>
  <distribution name="extended technical test" sample-type="real"
    type="normal" mean="732.2485" standardDeviation="448.1038"/>
  <distribution name="send back goods" sample-type="real"
    type="normal" mean="325.5" standardDeviation="182.0718"/>
  <distribution name="refund" sample-type="real"
    type="normal" mean="180" standardDeviation="30"/>

  <sim-process path="/.../ReturnDefectiveGoods/processdefinition.xml">
    <process-overwrite start-distribution="start">
      <use-data-source name="return orders" />
    </process-overwrite>
    <state-overwrite state-name="wait for parcel"
                     time-distribution="wait for parcel">
      <transition name="parcel arrived" probability="195"/>
```

```
    </state-overwrite>
    <decision-overwrite
       decision-name="ordered within the last two weeks?">
      <transition name="YES" probability="100"/>
      <transition name="NO" probability="95"/>
    </decision-overwrite>
    <task-overwrite task-name="transfer shipping costs"
                    time-distribution="transfer shipping costs">
      <transition name="done" probability="100"/>
    </task-overwrite>
    <task-overwrite task-name="quick test"
                    time-distribution="quick test">
      <transition name="no defect" probability="165"/>
      <transition name="defect approved" probability="30"/>
    </task-overwrite>
    <task-overwrite task-name="extended technical test"
                    time-distribution="extended technical test">
      <transition name="no defect" probability="20"/>
      <transition name="defect approved" probability="145"/>
    </task-overwrite>
    <task-overwrite task-name="send back goods"
                    time-distribution="send back goods">
      <transition probability="20"/>
    </task-overwrite>
    <task-overwrite task-name="refund" time-distribution="refund">
      <calculate-business-figure name='defect goods' />
      <transition probability="175"/>
    </task-overwrite>
  </sim-process>

  <resource-pool name="tester" pool-size="5"
                 costs-per-time-unit="0.025"/>
  <resource-pool name="clerk" pool-size="2"
                 costs-per-time-unit="0.011111111"/>
  <resource-pool name="accountant" pool-size="2"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
</scenario>

<scenario name="alternative" execute="true">
  <business-figure name = "value reduction of returned goods"
       type = "costs"
       automatic-calculation = "none"
       handler = "org.jbpm.sim.tutorial.AlternativeCostsCalculator" />
  <data-source name="return orders"
     handler="org.jbpm.sim.tutorial.TutorialDataSource" />

  <distribution name="start"
    sample-type="real" type="erlang" mean="95"/>

  <distribution name="wait for parcel" sample-type="real"
     type="normal" mean="28" standardDeviation="17"/>
  <distribution name="transfer shipping costs" sample-type="real"
     type="normal" mean="180" standardDeviation="30"/>
```

```
<distribution name="extended technical test" sample-type="real"
   type="normal" mean="732.2485" standardDeviation="448.1038"/>
<distribution name="send back goods" sample-type="real"
   type="normal" mean="325.5" standardDeviation="182.0718"/>
<distribution name="refund" sample-type="real"
   type="normal" mean="180" standardDeviation="30"/>

<sim-process
  path="/.../ReturnDefectiveGoodsAlternative/processdefinition.xml">
  <process-overwrite start-distribution="start">
    <use-data-source name="return orders" />
  </process-overwrite>
  <state-overwrite state-name="wait for parcel"
                   time-distribution="wait for parcel">
    <transition name="parcel arrived" probability="195"/>
  </state-overwrite>
  <decision-overwrite
     decision-name="ordered within the last two weeks?">
    <transition name="YES" probability="100"/>
    <transition name="NO" probability="95"/>
  </decision-overwrite>
  <task-overwrite task-name="transfer shipping costs"
                  time-distribution="transfer shipping costs">
    <transition name="done" probability="100"/>
  </task-overwrite>
  <task-overwrite task-name="extended technical test"
                  time-distribution="extended technical test">
    <transition name="no defect" probability="20"/>
    <transition name="defect approved" probability="145"/>
  </task-overwrite>
  <task-overwrite task-name="send back goods"
                  time-distribution="send back goods">
    <transition probability="20"/>
  </task-overwrite>
  <task-overwrite task-name="refund" time-distribution="refund">
    <transition probability="175"/>
  </task-overwrite>

  <!--overwrite decision to simulate random check factor-->
  <decision-overwrite decision-name="should be checked?">
    <transition name="check"    probability="5"/>
    <transition name="no check" probability="95"/>
  </decision-overwrite>
  <!-- and calculate "virtual business costs" -->
  <node-overwrite node-name='unknown goods status'>
    <calculate-business-figure
       name='value reduction of returned goods' />
  </node-overwrite>

</sim-process>

<resource-pool name="tester" pool-size="5"
               costs-per-time-unit="0.025"/>
<resource-pool name="clerk" pool-size="2"
               costs-per-time-unit="0.011111111"/>
```

```
  <resource-pool name="accountant" pool-size="2"
                 costs-per-time-unit="0.022222222"/>
  <resource-pool name="dispatcher" pool-size="1"
                 costs-per-time-unit="0.013888889"/>
</scenario>

</experiment>
```

A.4 dataphone prototype

The SessionBean in listing A.21 was implemented to include the simulation tool as a service into the Logis2 architecture. Evaluation of results and presentation to the user is done in the client of the SessionBean. Listing A.22 shows how statistics can be used to cope with multiple simulation runs. Because of the central limit theorem the presented normal distribution is appropriate to calculate the probability to reach a certain percentage of completed orders. More details can be found in section 3.3.4.

Listing A.21: SessionBean to integrate the simulation tool into Logis2.

```
@Stateless
@Name("SimulationService")
public class SimulationServiceBean implements SimulationServiceLocal {

  @EJB
  private QLagerServiceLocal lagerServiceBean;
  @EJB
  private LieferkalenderServiceLocal lieferkalender;

  private JbpmContext jbpmContext = (JbpmContext) Component.
                      getInstance("jbpmContext");

  private int runTime = 100;
  private int warmUpPeriod = 10;
  private int repetitions = 3;

  private BamSimulationProposal bamSimulationProposal = null;

  private String processName = null;
  private String resourceSwimlaneName = null;

  private class SimulationResultComparator implements
          Comparator<SimulationResult> {
    /**
     * best one first means, the bigger the percentage of completed
     * orders, the "smaller" the comparison result, because a sorted
     * set starts with the smallest values first
     */
    @Override
    public int compare(SimulationResult r1,
                       SimulationResult r2) {
      if (r1.getPercentageOfCompletedOrdersMean() >
          r2.getPercentageOfCompletedOrdersMean())
        return -1;
```

```
    else if (r1.getPercentageOfCompletedOrdersMean() <
             r2.getPercentageOfCompletedOrdersMean())
      return 1;
    else
      return 0;
  }
}

/**
 * leverage simulation
 * @return a sorted set of results with the best one first
 */
public SortedSet<SimulationResult> applySimulation(
    String processName, String swimlane, double requiredPercentage){
  SortedSet<SimulationResult> result =
    new TreeSet<SimulationResult>(new SimulationResultComparator());

  this.processName = processName;
  this.resourceSwimlaneName = swimlane;

  double lastResourceAssumptionResult = 0;
  int resultNotChangedCounter = 0;

  int resourceCountGuess = 1;

  // replace by a more sophisticated algorithm
  while (!isHappy(result.first().
        getPercentageOfCompletedOrdersMean(), requiredPercentage)) {
    SimulationResult simulationResult = checkResourceAssumption(
        resourceCountGuess);
    result.add(simulationResult);
    resourceCountGuess++;

    if (lastResourceAssumptionResult - simulationResult.
        getPercentageOfCompletedOrdersMean() < 0.00001) {
      resultNotChangedCounter++;

      if (resultNotChangedCounter>=5)
        throw new SimulationExecutionException("could not find "
           + "suitable number of resources.");
    }
    else
      resultNotChangedCounter = 0;

    lastResourceAssumptionResult = simulationResult.
       getPercentageOfCompletedOrdersMean();
  }

  return result;
}

private boolean isHappy(double finishedOrderPercentage,
                        double requiredPercentage) {
  return (finishedOrderPercentage > requiredPercentage);
}
```

```
/**
 * Starts the simulation runs (repeated n times like configured)
 * with the given amount of resources
 *
 * @param resourceCount number of resources
 * @return percentage of finished orders
 */
private SimulationResult checkResourceAssumption(int resourceCount){
  SimulationResult result = new SimulationResult();

  String scenarioName = String.valueOf(resourceCount);

  // create basic experiment configuration
  Element experimentConf = org.dom4j.DocumentFactory.getInstance().
     createElement("experiment");
  experimentConf.addAttribute("name", processName + "."
     + resourceCount);
  experimentConf.addAttribute("time-unit", "second");
  experimentConf.addAttribute("run-time",
     String.valueOf(runTime + warmUpPeriod));
  experimentConf.addAttribute("reset-time",
     String.valueOf(warmUpPeriod));

  // create & add scenario
  Element scenario = getScenarioConfigurationFromHistory();
  adjustScenarioElement(scenario, resourceCount);
  experimentConf.add(scenario);

  // now build experiment object
    ExperimentReader reader = new ExperimentReader(
            new InputSource(experimentConf.asXML()));
    JbpmSimulationExperiment experiment = reader.readExperiment();
    experiment.setWriteDesmojHtmlOutput(false);

    adjustScenario(experiment.getScenario(scenarioName),
      resourceCount);

  // use same seed to make simulation runs comparable
  Random rnd = new Random(1000);

    // everything is prepared now, so run the simulation
    for (int i = 0; i < repetitions; i++) {
      // use different seeds for every simulation run
      experiment.setSeed(rnd.nextLong());
      experiment.run(); // can take some time!

      evaluateSimulationResult(
          experiment.getScenario(scenarioName).getScenarioReport(),
          result);
  }

    return result;
}
```

```
private void adjustScenarioElement(Element scenario,
            int resourceCount) {
 scenario.attribute("name").setValue(String.valueOf(resourceCount));

 // TODO: adjust scenario to forecast:
 // lieferkalender.xxx

 // add data source usage
 scenario.element("process-overwrite").addElement("use-data-source")
         .addAttribute("name", "lieferungen");
}

private void adjustScenario(JbpmSimulationScenario scenario,
            int resourceCount) {
 // add data source
 scenario.addDataSource("lieferungen",
   createLieferungenDataSource());
 scenario.addResourcePool(resourceSwimlaneName, resourceCount, 0.0);
}

private ProcessDataSource createLieferungenDataSource() {
return new LieferungenDataSource(lagerServiceBean.getLieferungen());
}

/**
 * evaluate ended process percentage.
 *
 * ATTENTION: The current implementation works only
 * if one process is included in simulation, otherwise
 * the end states should be checked correctly
 *
 * @param report
 * @param result
 * @return
 */
private void evaluateSimulationResult(ScenarioReport report,
            SimulationResult result) {
  long startedProcesses = report.getProcessStartCount(processName).
                                getCount();
  long endedProcesses = 0;
  for (CountResult count : (Collection<CountResult>)report.
          getProcessEndCounts().values()) {
    endedProcesses += count.getCount();
  }

  double percentageOfCompletedOrders =
         endedProcesses / startedProcesses;
  double resourceUtilization = report.getResourcePoolUtilization(
    resourceSwimlaneName).getAverageUtilization();

  result.addSimulationRunResult(percentageOfCompletedOrders,
    resourceUtilization);
}

private Element getScenarioConfigurationFromHistory() {
```

```
    if (bamSimulationProposal==null) {
      GetSimulationInputCommand cmd = new GetSimulationInputCommand(
                                      processName);
      try {
        bamSimulationProposal = (BamSimulationProposal) cmd.execute(
                                jbpmContext);
      }
      catch (Exception ex) {
        throw new SimulationExecutionException(
          "problem while retrieving historical data from jbpm",
          ex);
      }
    }
    return bamSimulationProposal.createScenarioConfigurationXml();
  }
}
```

Listing A.22: SimulationResult class, responsible to calculate statistics about the result of multiple simulation runs.

```
public class SimulationResult {

  // The statistics of the Apache commons math are used here
  private SummaryStatisticsImpl percentageOfCompletedOrders =
    new SummaryStatisticsImpl();

  private SummaryStatisticsImpl resourceUtilization =
    new SummaryStatisticsImpl();

  public void addSimulationRunResult(
    double percentageOfCompletedOrders, double resourceUtilization) {
    this.percentageOfCompletedOrders.addValue(
      percentageOfCompletedOrders);
    this.resourceUtilization.addValue(resourceUtilization);
  }

  public long getSampleSize() {
    return percentageOfCompletedOrders.getN();
  }

  public double getPercentageOfCompletedOrdersMean() {
    return percentageOfCompletedOrders.getMean();
  }

  public double getPercentageOfCompletedOrdersStandardDerivation() {
    return percentageOfCompletedOrders.getStandardDeviation();
  }

  public double getResourceUtilizationMean() {
    return percentageOfCompletedOrders.getMean();
  }

  public double getResourceUtilizationStandardDerivation() {
    return percentageOfCompletedOrders.getStandardDeviation();
  }
```

```
  public NormalDistribution
    getPercentageOfCompletedOrdersNormalDistributions() {
    return new NormalDistributionImpl(
        getPercentageOfCompletedOrdersMean(),
        getPercentageOfCompletedOrdersStandardDerivation());
  }
}
```

Appendix B

Content of CD

The CD attached to this thesis contains:

- The sources of the jBPM simulation tool, exported from CVS. This includes the source code of the tutorial.
- The jBPM simulation library as byte code (jar).
- The Javadoc documentation of the jBPM simulation.
- The reports generated for the show cases.
- The sources of the required Java classes to integrate jBPM simulation into Logis2.
- The master thesis itself as PDF.

Bibliography

[Ack04] Peter J. Acklam. An algorithm for computing the inverse normal cumulative distribution function. http://home.online.no/~pjacklam/notes/invnorm/index.html, 2004.

[Bar06] Dr. Martin Bartonitz. Wachsen die BPM- und Workflow-Lager zusammen? http://www.bpm-netzwerk.de/content/articles/viewArticle.do?id=66, 2006.

[Bau99] Joachim Bauske. *Ein objektorientiertes Verfahren zur Optimierung von Geschäftsprozessen unter Verwendung eines genetischen Algorithmus*. Springer, Berlin, 1999.

[BEFF06] Martin Böhn, Stephan Englisch, Dirk Friedrich, and Jakob Freund. *Prozessmodellierungswerkzeuge: 14 Systeme für Dokumentation, Entwurf, Simulation und Analyse im Vergleich*. Oxygon, 2006.

[BF07] Tom Baeyens and Miguel Valdes Faura. The process virtual machine. http://docs.jboss.com/jbpm/pvm/, 2007.

[BFS87] Paul Bratley, Bennet L. Fox, and Linus E. Schrage. *A Guide to Simulation*. Springer, 1987.

[BKR05] Jörg Becker, Martin Kugeler, and Michael Rosemann. *Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung: Ein Leitfaden Zur Prozessorientierten Organisationsgestaltung*. Springer, Berlin, 2005.

[Bou02] Günther Bourier. *Wahrscheinlichkeitsrechnung und schließende Statistik . Praxisorientierte Einführung. Mit Aufgaben und Lösungen (Gabler Lehrbuch)*. Gabler Verlag, 2002.

[BR07] Martin Backschat and Bernd Rücker. *Enterprise JavaBeans 3.0. Grundlagen - Konzepte - Praxis*. Spektrum Akademischer Verlag, 2007.

[BVCH07] Vesna Bosilj-Vuksic, Vlatko Ceric, and Vlatka Hlupic. Criteria for the evaluation of business process simulation tools. http://www.ijikm.org/Volume2/IJIKMv2p073-088Bosilj396.pdf, 2007.

[DWM02] Gartner David W. McCoy. Business activity monitoring: Calm before the storm. http://www.gartner.com/resources/105500/105562/105562.pdf, 2002.

[Fis06] Guido Fischermanns. *Praxishandbuch ProzessManagement*. Schmidt (Götz), Wettenberg, 2006.

[Gar06a] Gartner. Magic quadrant for business process analysis tools, 2006. http://www.gartner.com/DisplayDocument?id=489780, 2006.

[Gar06b] Gartner. Magic quadrant for business process management suites, 2006. http://www.gartner.com/DisplayDocument?id=493187, 2006.

[GP01] Björn Gehlsen and Bernd Page. A framework for distributed simulation optimizations. http://asi-www.informatik.uni-hamburg.de/personen/gehlsen/pub/2001wsc_paper.pdf, 2001.

[GW04] Uwe W. Gehring and Cornelia Weins. *Grundkurs Statistik für Politologen*. VS Verlag für Sozialwissenschaften, 2004.

[Hav05] Michael Havey. *Essential Business Process Modeling*. O'Reilly Media, Inc., 2005.

[HC96] Michael Hammer and James Champy. *Business Reengineering. Die Radikalkur für das Unternehmen*. Campus Fachbuch, 7. edition, 1996.

[Hoa07] Dr Katy Hoad. Steady state models and the initial transient. http://www2.warwick.ac.uk/fac/soc/wbs/projects/autosimoa/website_warmup_methods_total_doc.pdf, 2007.

[HR07] Katariina Hoffmann-Remy. Prozesse immer wichtiger für Unternehmen. 2007 – ein starkes Jahr für BPM. http://www.it-business.de/themenkanaele/marktforschung/trends/studien/articles/62102/, 2007.

[iuF97] iwb und FAPS. Zusammenfassung der Studie: Einsatz der Simulationstechnik. http://www.costsim-consulting.de/iwb_faps_simulationsstudie.pdf, 1997.

[JBo07] JBoss. jBPM jPDL 3.2 User Guide. http://docs.jboss.com/jBPM/v3.2/userguide/html/, 2007.

[JVN06] Monique Jansen-Vullers and Mariska Netjes. Business process simulation - a tool survey. *Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2006.

[Kre79] Erwin Kreyszig. *Statistische Methoden und ihre Anwendungen*. Vandenhoeck & Ruprecht, 1979.

[LK00] Averill M. Law and W.David Kelton. *Simulation Modelling and Analysis*. McGraw Hill Higher Education, 2000.

[LML04] Manuel Laguna, Johan Marklund, and Laguna. *Business Process Modeling, Simulation, and Design*. Prentice Hall, 2004.

[Mac05] Lori MacVittie. It detours on the road to bpm. http://www.intelligententerprise.com/showArticle.jhtml?articleID=165700251, 2005.

[NRvdA06] Mariska Netjes, Hajo A. Reijers, and Wil M.P. van der Aalst. Filenets bpm life-cycle support. 2006.

[PH03] BPTrends Paul Harmon. Simulation and business process change. http://www.bptrends.com/publicationfiles/bpt%20email%20advisor%2011%%5F25%5F03%2Epdf, 2003.

[PK05] Bernd Page and Wolfgang Kreutzer. *The Java Simulation Handbook. Simulating Discrete Event Systems with UML and Java*. Shaker Verlag GmbH, Germany, 2005.

[PP05] Dan Pilone and Neil Pitman. *UML 2.0 in a Nutshell (In a Nutshell (O'Reilly))*. O'Reilly Media, Inc., 2005.

[Pro97] David Profozich. *Managing Change with Business Process Simulation*. Prentice Hall, 1997.

[Reh07] F. John Reh. Key performance indicators (kpi). http://management.about.com/cs/generalmanagement/a/keyperfindic.htm, 2007.

[Ros03] Ronald G. Ross. *Principles of the Business Rule Approach (Addison-Wesley Information Technology Series)*. Addison-Wesley Professional, 2003.

[RvH04] H.A. Reijers and K.M. van Hee. Business Process Management in een Notendop (in Dutch). *VIP: Vakblad voor Documentmanagement*, pages 30–32, 2004.

[Ses04] Hermann J. Schmelzer; Wolfgang Sesselmann. *Geschäftsprozessmanagement in der Praxis*. Hanser, 2004.

[Sha75] Robert E. Shannon. *Systems Simulation: The Art and Science*. Prentice Hall, 1975.

[Smi98] Roger D. Smith. Simulation article. http://www.modelbenders.com/encyclopedia/encyclopedia.html, 1998.

[vdAtHW03] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske. usiness process management: A survey. *International Conference on Business Process Management (BPM 2003), volume 2678 of Lecture Notes in Computer Science*, pages 1–12, 2003.

[Web07] Webopedia. Kpi, 2007.

[WfM07] WfMC. Workflow reference model. http://www.wfmc.org/standards/referencemodel.htm, 2007.

[WMF+07] Branimir Wetzstein, Zhilei Ma, Agata Filipowska, Monika Kaczmarek, Sami Bhiri, Silvestre Losada, Jose-Manuel Lopez-Cobo, and Laurent Cicurel. Semantic business process management: A lifecycle based requirements analysis. 2007.

List of Figures

List of Tables

Listings