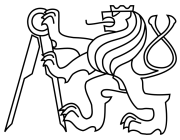
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology
Student:	Petr Nymsa
Supervisor:	Ing. Marek Skotnica
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2018/19

Instructions

Design and Engineering Methodology for Organizations (DEMO) is an enterprise modelling methodology for human interaction and co-production modelling, analysing, and representing of business processes. When operation of an enterprise runs according to the DEMO models, a large amount of precise operational data is collected. The goal of this thesis is to provide managers with a comprehensive graphical overview of this data in order to make important decisions about an operation of an enterprise.
Steps to take:
- Review state-of-the-art how BPM systems visualize process data based on BPMN.
- Propose a way of visualizing DEMO-based process data.
- Design and implement a proof-of-concept interactive mobile application that implements proposed concepts.
- Use .NET technologies for implementation - .NET Core, Xamarin, SignalR

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague November 3, 2017

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology

Petr Nymsa

Department of software engineering science
Supervisor: Ing. Marek Skotnica

May 8, 2018

Acknowledgements

I would like to thanks to anyone who helped me write this thesis. Especially the most gratitude belongs to my supervisor Ing. Marek Skotnica for his guidance. Also many thanks goes to my friends and family for proofreading and valuable ideas and help.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 8, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Nymsa, Petr. *Mobile Enterprise Architecture Process Analytic Tool Based on the DEMO Methodology.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Práce se zabývá problematikou vizualizace podnikových procesů v reálném čase a následné zpracování nasbíraných dat do grafických přehledů. Hlavním témeatem je zaměření se na metodiku DEMO, která přináší jiný způsob a také pohled na modelování podnikových procesů. Cílem práce je navrhnout přístup vizualizace procesů modelovaných metodikou DEMO. Kromě vizualizace je cílem vytvořit mobilní aplikaci formou proof-of-concept, která obsahuje navrhované přístupy a také vizualizaci nasbíraných dat do různých grafických přehledů, které mohou pomoci lépe řídit podnik.

Práce je zaměřena více teoreticky, kde autor řeší různé přístupy vizualizací. K ověření navrhovaných přístupů je vytvořena mobilní aplikace, která demonstruje vizualizaci na modelových příkladech.

Klíčová slova mobilní aplikace, vizualizace procesů, DEMO metodika, business intelligence, procesní řízení, Xamarin

Abstract

Thesis is focused on business process visualisation in real-time and processing collected data and displaying them within graphical overviews. Main focus is on DEMO methodology in the thesis, which brings another approach how business process modelling can be done. Goal of thesis is to propose a way how process visualisation through DEMO can be done. Besides visualisation, second goal is to create mobile application in form of proof-of-concept, which contains proposed approaches and visualisation of collected data within various graphical overviews. This can help people to lead better their business.

Thesis is more theoretical, author is focused on propose various visualisation approaches. The simple mobile application is implemented to verify proposed approach, where visualisation is demonstrated on model examples.

Keywords mobile application, process visualisation, business intelligence, DEMO methodology, business process management, Xamarin

Contents

List of Figures

List of Tables

Introduction

Nowadays almost every company has more independent systems which can help employees to do their job. For example one system for accountancy, another for warehouses and another one for managing orders or something similar. However how business grows, some kind of business analysis is required. Usually some person is instructed to analyse the business, how to make it better. The person collects all required data from every system, typically in some form of "Excel table", and makes analysis over it. This approach at the beginning can be sufficient, however sooner or later, it becomes heavy, uncomfortable and mainly not effective way how to do this kind of analysis. All required data are spread over all systems without any order. Some systems can have functions to export data in some suitable form, but the data from another system could be exported only in some form of plain-text. At some point people find out that in their business "reigns chaos". Fortunately this "chaos" can be controlled with some well known approaches. Most well known term of how "chaos' can be controlled is Business Process Management (BPM).

Analysis within BPM

Every task or set of tasks within the business can be formulated as *business process*. For example task "Order of new components for machine" (which consists of many additional steps) can be expressed as business process "New machine components order". The well-defined business processes are the first step to having better control over own business and also for further analysis.

Next step is that these defined processes connect under one system with internal business applications. The "one system" is typically marked as BPM System (BPMS). Within BPMS the business processes are defined, modelled and executed. Modelling is typically done through Business Process Model and Notation (BPMN) which is a well-known method to model business processes. Execution means, that dependent internal systems within business communi-

cates with BPMS and output from these systems is reflected in BPMS. This is the second important step to have a more precise analysis of own business.

If the company has some kind of BPMS, they have also well-defined data and aggregated them in one place. Usually, every BPMS offers functionalities to do analysis and monitoring over collected data. These functionalities can help people managing their business, more easily finding issues and potentially solving them. Monitoring is typically done through graphical overviews, which offer various views on the business metrics. These metrics are for example business efficiency, employee performance (how much the desired goals are fulfilled), financial efficiency and so on. Thanks to these systems people have the better understanding of their business and they can more easily focus on the critical places that should be improved.

A New Approach

While BPMN is well known and widely used, the new methodology was introduced by *J.Dietz* [1]. This new methodology called DEMO is considered as successor of BPMN. DEMO is build on high-quality scientific foundations, which brings against BPMN (and similar methods) better modelling of business processes. Although DEMO is well defined methodology, BPMS based on DEMO does not (nearly) exists. The problem of BPMS based on DEMO is subject of many researches. There are first attempts to achieve a solution based on DEMO, which can design processes and execute them. However the monitoring and analysis are missing.

This thesis focuses on visualisation of business processes based on DEMO methodology. Within defined process the goal of thesis is to find a way:

- The data processing from internal systems and their connection within DEMO model.
- Offer data to users with appropriate graphical overviews.
- Real-time visualisation of running processes.

Motivation

The main reason why author focused on this topic comes from a school project. With his team created chatbot application for pizza delivering system which was based on DEMO methodology. In fact we used *DEMO Engine*, the BPMS based on DEMO developed by *ForMetis* company. Moreover, on this *DEMO Engine* worked also author's supervisor (and also the leader of the team) *M.Skotnica* [2]. Within *engine* the team defined model for "imaginary" pizzeria. Then the team used this defined processes and connected them with the chatbot. However, this system had only ability to design, model and execute

processes. The monitoring of collected data or visualisation of processes were missed. That's why I have chosen this topic.

Structure

The thesis is divided to four chapters:

- Chapter 1 deals with underlying theories and terms are explained.
- Chapter 2 deals with research of existing BPMS solutions follows. Then some specific solutions are investigated for analysis of own solution. Role of DEMO within BPMS is described.
- Chapter 3 deals with introducing of the proposed approach. The theory and research from previous chapters are used. The concept of a system is described. Principle of real-time visualisation is more precisely explained.
- Chapter 4 deals with implementation of proof-of-concept. Architecture and used technologies are described. An example model of business processes is demonstrated.

In conclusion the results are compared with the goals of this thesis.

Chapter 1

The Theoretical Foundations

In the first chapter, theoretical foundations are described. Firstly, terms such as BPM, Business Intelligence are explained. Then follows, the theory of Enterprise Ontology and DEMO methodology. In the end, Gantt chart is described, because it is used as part of visualisation approach.

1.1 Business Process Management

Business Process Management (BPM) is an approach that focuses on modelling, analysing, improving and monitoring business processes. The BPM life-cycle can be divided to five stages as shown in fig. 1.1:

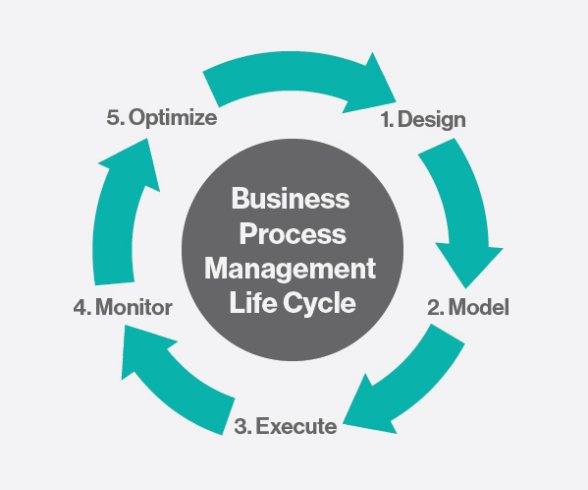


Figure 1.1: BPM Life-cycle [3]

Design – The requirements of the business are collected, analysed and specified. The specification can be "as is" (state of how processes currently

are) or "to be" (how processes would be). Accompanying graphical materials (such as flow-charts) are included.

Model – Specification is expressed (modelled) with graphical notation Business Process Model and Notation (BPMN), which will be investigated later.

Execute – Changes are implemented and deployed.

Monitor – After deployed changes, monitoring system comes to the scene. Processes are monitored and Business Process Model Systems (BPMS) tools are used to collect data and analyse performance through metrics - called Key Performance Indicator (KPI). KPI are defined, optimized metrics which can help to understand the performance of the current business. As an example of KPI can be "an average number of requests for new order per day" and many more, individual metrics to concrete business.

Optimize – At some point, when an appropriate number of data are collected and analysed with monitoring, optimization is done. The goal of optimization is to find issues or some improvements. After that, new specifications and improvements are created and *design* stage is executed again.

1.1.1 Business Process Model and Notation

BPMN is notation created and standardised by *Object Management Group* (see `https://www.omg.org`). According to [4] BPMN can be described as:

> BPMN is a graphical notation that depicts the steps in a business process. BPMN depicts the end to end flow of a business process. The notation has been specifically designed to coordinate the sequence of processes and the messages that flow between different process participants in a related set of activities.

The notation itself, elements which are used and how modelling is done, are not described. For this information, see [4].

1.2 Business Intelligence

Business Intelligence (BI) is a way how business can collect an enormous amount of data, structure them and analyse (see fig. 1.2. From BPM lifecycle view, BI is a *Monitor* stage. Systems which are tagged as BI tools typically offers some kind of reporting, dashboards and scorecards.

Dashboard offers graphical overviews of collected data. Dashboards are typically highly customizable and can offer different types of views for each

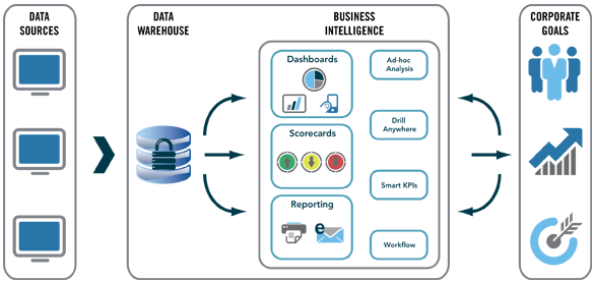


Figure 1.2: Business intelligence diagram [5]

employee to help achieve their job. Dashboards can help understand and find issues within business and potentially solve them more quickly. Figure 1.3 shows an example.

Scorecards offer monitoring of KPI metrics and user can easy compare goals and current results. Scorecards also easily show performance of business, for example if business plan is fulfilled or financial flow (e.g if business is profiting or not,...).

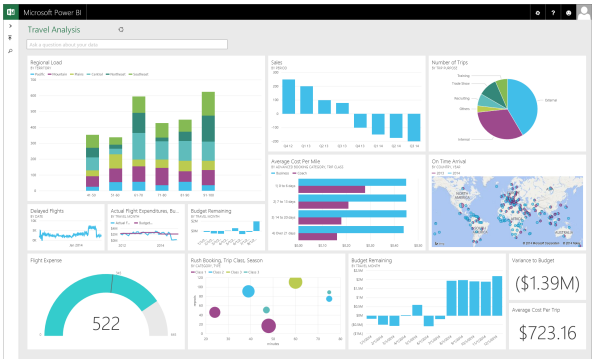


Figure 1.3: Business Intelligence dashboard example [6]

1.3 Enterprise ontology

According to [7], definition of ontology states as "An ontology is a specification of a conceptualization". In other words ontology is description, a formal specification, of concepts and relationships.

Enterprise Ontology [1] is the understanding of the essence of organisation, completely independent of the way in which this essence is realised and implemented.

The DEMO Methodology [8] is an engineering methodology, based on theory of enterprise ontology.

The following text was taken from article describing Enterprise Ontology (EO) theory [9]:

> Enterprise ontology is focused on the essence of the operation of an organization, meaning that it is fully independent of the (current) realization and implementation of the organization. The theory that underlies the notion of enterprise ontology as presented by Dietz is called the PSI-theory. Dietz uses this theory to construct a methodology providing an ontological model of an organization, i.e. a model that is coherent, comprehensive, consistent, and concise, and that only shows the essence of the operation of an organization model. This methodology is called Design and Engineering Methodology for Organizations (DEMO).
>
> Compared to its implementation model, the ontological model of an enterprise offers a reduction of complexity of over 90%. This reduction of complexity makes an organization for a manager intellectual manageable and transparent. It also shows the coherence between all fields within the enterprise, like business processes, workflow, organization structure, etc.
>
> The overall goal of the PSI-theory (the theory behind the notion of Enterprise Ontology) is to extract the essence of an organization from its actual appearance. It presents four axioms that help to achieve this goal.
>
> The **operation axiom** (fig. 1.4) tells us that the implementation independent essence of an organization is that it consists of subjects fulfilling actor roles. A subject fulfilling a certain actor role is called an actor. Actors constitute the operation of an organization by performing two kinds of acts: production acts and coordination acts.

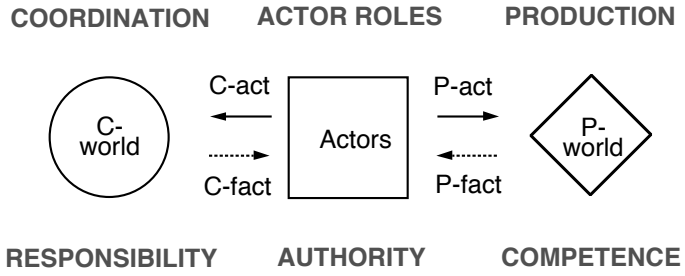


Figure 1.4: The operation axiom [9]

By performing production acts (P-acts for short) the subjects contribute to bringing about the goods and/or services that are

delivered to the environment of the organization. The results of P-acts are production facts (P-facts for short), which can be divided into material (something is manufactured, stored or transported) and immaterial (decisions or judgements) facts.

By performing coordination acts (C-acts for short) subjects enter into and comply with commitments towards each other regarding the performance of production acts. A C-act is performed by one actor, the performer, and directed to another actor, the addressee. C-acts consist of an intention (e.g. request, promise, question, assertion) and a proposition (the performer proclaims the fact and the associated time the intention is about) and result in coordination facts (C-facts for short).

The **transaction axiom** tells us that C-acts are performed as steps in universal patterns, called transactions. This axiom reveals universal socionomic patterns of coordination that hold for all organizations. The standard transaction pattern is shown in fig. 1.5. A white box represents a C-act type and a white disk represents a C-fact type. A gray box represents a P-act type and a gray diamond a P-fact type.

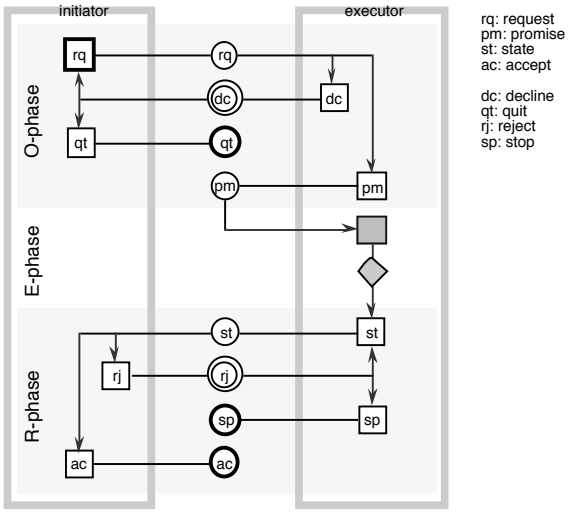


Figure 1.5: The standard transaction pattern [9]

Two actors are involved in a transaction, the initiator and the executor. A transaction evolves in three phases: the order phase (O-phase for short), the execution phase (E-phase for short), and the result phase (R-phase for short).

The **composition axiom** tells us how P-facts are interrelated. It states that every transaction is enclosed in some other transac-

tion, or is a customer transaction of the organization under consideration, or is a self-activation transaction. According to Dietz this axiom provides the basis for a well-founded definition of the notion of business process.

The **distinction axiom** (fig. 1.6) tells us that actors exert three basic human abilities: performa, informa, and forma. Through the distinction axiom a substantial reduction of complexity and diversity is achieved, regarding both the coordination and the production in an organization.

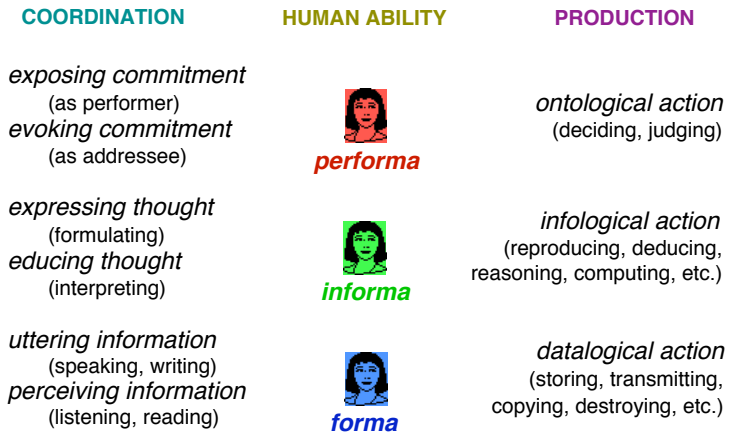


Figure 1.6: The distinction axiom [9]

1.3.1 The Organization Theorem

The organization theorem combines the benefits of these axioms into one concise, comprehensive, coherent, and consistent notion of enterprise. This theorem states that the organization of an enterprise is a heterogeneous system that is constituted as the layered integration of three homogeneous systems: the B-organization (from Business), the I-organization (from Intellect), and the D-organization (from Document). Figure 1.7 visualizes the organization theorem and shows us that the D-organization supports the I-organization, which supports the B-organization. The coordination parts of these three systems are similar, they only differ in the kind of production: the production of the B-organization is ontological, the production in the I-organization is infological, and the production in the D-organization is datalogical.

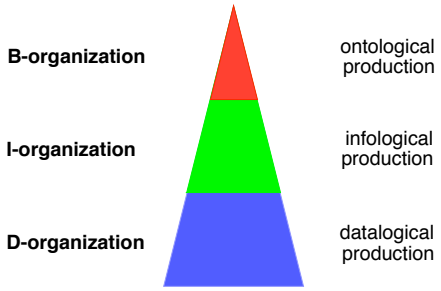


Figure 1.7: The organization theorem [9]

1.4 Modelling with DEMO

In this section, firstly short description of modelling aspects is provided and then a illustrative example how modelling with DEMO can be more precise and shorter than creating model with BPMN.

All used elements in DEMO modelling are described in aspects models arranged as pyramid. However the two core elements are Ontological transaction and Actor roles. Good summarization of these two core elements are from master thesis by Zuzana Vejrazkova [10]:

Ontological transaction – Involves actions that happen on the ontological level, as described by the Distinction axiom. Those involve bringing about the facts that did not exist before, making decisions, or transporting physical elements. Completion of a transaction, in a way that is described by the transaction axiom, results in a new original fact, called the P-fact.

Actor role – There are two actor roles, the initiator and the executor. They play an important role in DEMO modelling, as each transaction needs to have exactly one initiator and one executor. On the implementation level, 1 person can (and often does) posses more actor roles.

Figure 1.8 shows that ontological model can be divided to four sub-models. The good and brief explanation is in the book *The essence of organisation: an introduction to enterprise engineering* [1]:

Construction model (CM) is the most concise sub-model. Therefore it is put at the top of the triangle. An additional meaning of this position is that there is nothing "above" the CM. The Construction Model shows the identified transaction kinds, the corresponding actor roles, and the border of the Scope of Interest.

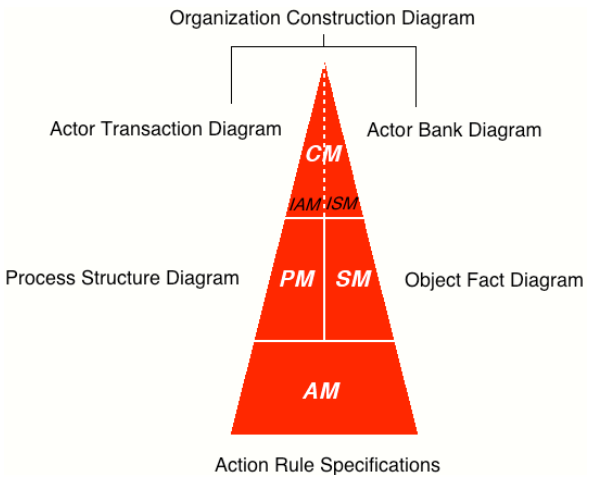


Figure 1.8: The DEMO Aspect Models pyramid [11]

Action model (AM) is the most comprehensive one, in the sense that the other three may be derived from it. The AM of an organisation consists of the action rule specifications for every internal actor role. Action rules are guidelines for dealing with the events that actors have to respond to.

Process Model (PM) shows precisely how the identified transactions are interrelated in tree structures. These tree structures are what people commonly refer to as business process models.

Fact Model (FM) show the fact kinds in the production world of the organisation and their interrelationships.

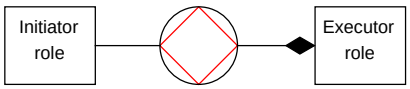


Figure 1.9: The core OCD element - transaction symbol with assigned actor roles

The PM is in between CM and the AM, that means it is more detailed than CM but less than AM. The PM and the FM are on the same layer of the pyramid and it refers to the fact, that PM takes the process view of coordination world, and the FM is for the production world in the same meaning as PM.

From the CM is used Organization Construction Diagram (OCD) which takes standard process pattern and squeeze it to one symbol with two connected boxes which represent initiator and executor actor roles (see fig. 1.9).

From the PM is taken Process Structure Diagram (PSD) sub-model which describes business processes and the exact way how each transaction is connected with another. Following explanation how PSD is modelled is from [1]:

> The disk of the transaction is stretched horizontally, such that it looks like a sausage. One must imagine that there is an invisible and non-proportional time line from left to right (promise is performed after its request,...). Coordination acts and facts are represented by small boxes and disks on the border of the transaction symbol (the sausage). The production act, execute, is represented by a small grey box on the edge of the production symbol. To the left of it is the order phase, and to the right the result phase. Between two transactions (the sausages) can be added arrows, either solid or dashed. Solid arrows represent *response links*. Response link means that the act at the arrow point is performed in response to the event at the shaft. Dashed arrow represent *waiting links*. Their meaning is that performing the act at the arrow point must wait for the event at the shaft having occurred. Through "swim lanes" the responsibility areas of the actor roles are indicated.

An example of PSD is shown at fig. 1.10 which means exactly "When the transaction T1 (Supply order) is *requested*, as a response, T2 (Tax payment) is also *requested*. After that, T1 can be promised only and only, if T2 is *accepted*. After that T1 can be completed".

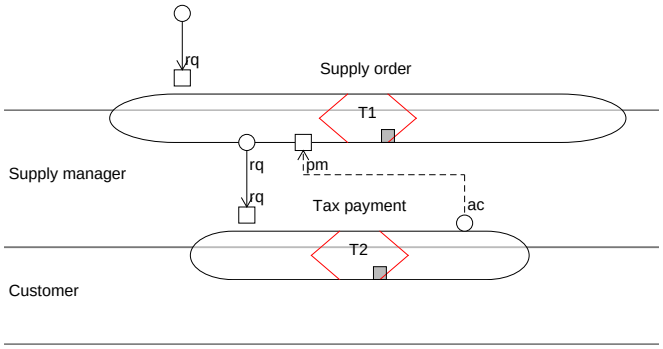


Figure 1.10: PSD example

1.4.1 Advantage of DEMO Over BPMN

BPM notation is well defined, established and widely used. Many BPM solutions exist around and of course, BPMN is used "under the hood". These systems will be described later in chapter 2.

One thing is absolutely clear, with enlarging business requirements, the business processes "grow up" too and so BPMN diagrams. This can lead to very large and complex diagrams, which can be harder to read and understand. One of the advantages of DEMO is the fact, that same BPMN diagram can be represented within DEMO notation. The result is reduced complexity with the same understanding as diagram within BPMN. However, advantages of DEMO notation can be used not only within BPMN. For example, students at subject MI-MEP (Modelling economic processes, translated) at Czech Technical University (CTU) [12] transferred large Trump's flow chart[13] (about highway building process) to DEMO OCD diagram. The flow chart that fills several pages, DEMO's OCD fills only one page of size A4. To summarize, DEMO can really help to understand better business processes without too much complexity of diagrams itself.

1.5 Gantt chart

According to [14], Gantt chart was proposed in 1890 by Karol Adamiecki, but he published his work only in the Polish language. About the year 1910 American engineer, Henry Gantt, introduced his own version of this chart and this work became widely used and is still used by project managers. Gantt chart serves as an overview of activities according to the time schedule. By simplicity, Gantt chart shows, what has to be done (an activity) and when. At fig. 1.11 is an example of Gantt chart. The core element is some kind of bar chart (progress bar) which starts at some time and is scheduled to end in another time. Nowadays, Gantt charts, has more advanced elements, such as connected activities (one activity must finish before another one can start), groups of activities, different colours of activities that can indicate different phases of completion and so on. Many systems for time-management support these types of graphs. Gantt chart and its usage for this thesis are discussed later, in chapter 3.

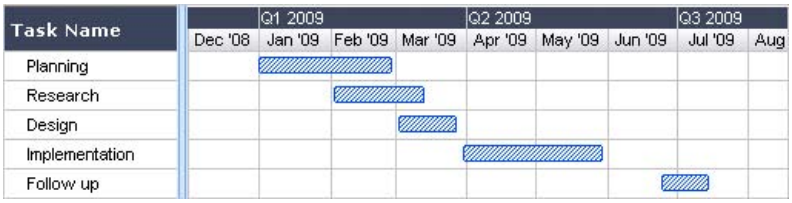


Figure 1.11: Gantt chart example [14]

Chapter 2

Process Visualisation

The state of BPM solutions is described in this chapter. Two concrete solutions are examined closer. Role of DEMO within BPM systems is explained.

2.1 State of BPM Solutions

Figure 2.1: Gartner Magic Quadrant [15]

Nowadays BPM software supports all stages of BPM life-cycle (design, modelling, execution, monitoring and optimization). These systems also offer real-time collaboration, integration with cloud and mobile devices. Many systems also integrate artificial intelligence – for predictive analysis or automatic decisions. BPM software allows to create highly productive applications,

where is no need to manual implementation of business rules. BPMS includes defining processes, data models and also user interfaces. Also highly customized monitoring stage is included, which means users can customize KPI metrics, appearance of dashboard and so on. On top of that functions, almost every solution has ability to display (visualise) current running processes.

The result is typically web based application which employees can use to do their jobs without knowing that there are some defined processes and layer of some BPM software.

According to *Gartner Magic Quadrant* [15], the most used systems are *Appian*, *Pegasystems*, *IBM* and many more as shown at fig. 2.1.

2.1.1 Closer Look at Process Maker

ProcessMaker [16] is web-based BPM solution which allows to build, run, monitor and optimize business processes. Building of processes is done via BPMN. Inside designer users can define data sources (variables) which will be used later within running processes.

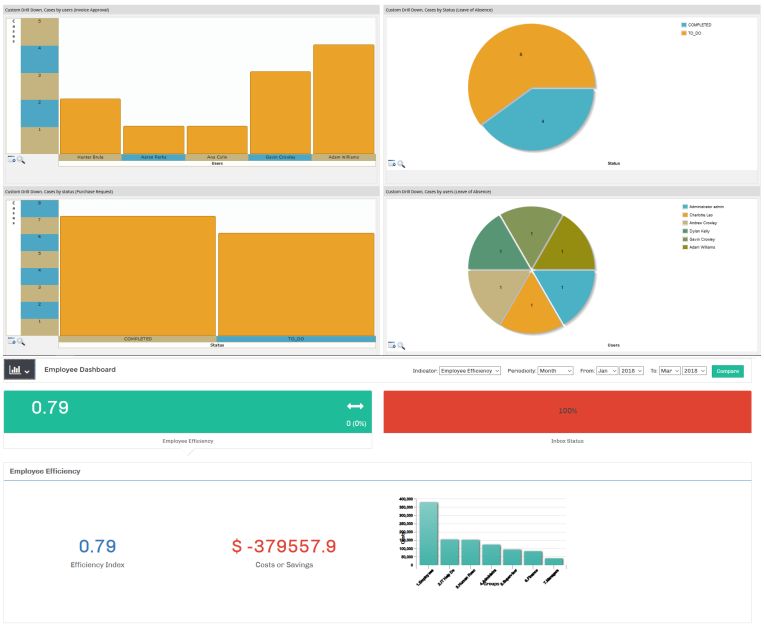


Figure 2.2: ProcessMaker dashboard and KPI examples [16]

When process is designed and variables defined, next step is to define user interface, called *DynaForms*. End users interact mainly with *DynaForms*, where they fill appropriate pieces of data. Inside designer of *DynaForms*, user connects data fields with variables from defined processes. The user can also define input or output documents for processes or define events (called triggers) what will happen when some kind of event occurred.

When processes are defined and *DynaForms* created, processes can be deployed. After deploy, KPIs and dashboards can be managed. *ProcessMaker* offers creating custom metrics, which will be precise to business requirements and customizing of dashboards. Many graphs are interactive, which can display more detailed information. An example is shown at fig. 2.2.

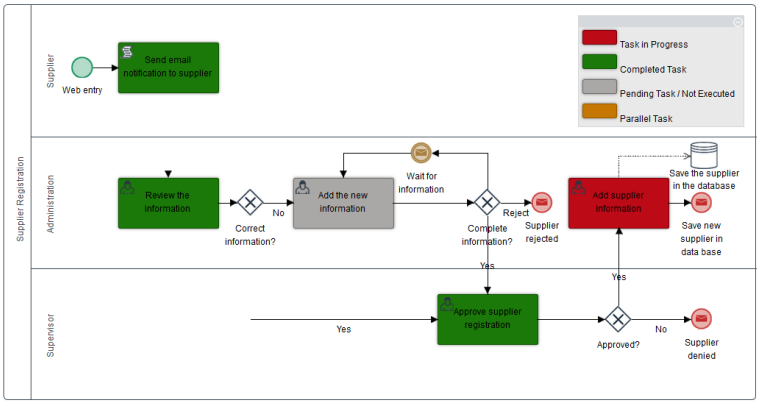


Figure 2.3: ProcessMaker process map [16]

From end-user view, *ProcessMaker* (as many others BPMS) behaves as web-based application, which allows to do his job via predefined *DynaForms*. ProcessMaker moreover offers process map, which is graphical visualisation of running process. Process map is in BPMN and each activity element has different colour to indicate its state (fig. 2.3).

2.1.2 Process Simulation and Analysis

Another tool, *Bizagi Modeler* [17], offers modelling processes with BPMN and run simulations on top of them.

For simulations *Bizagi Modeler* defines scenarios. Each scenario has many attributes, but mainly:

Resources – Resource is some entity (e.g. customer, employee role, equipment) which is used to define how each task of the process use these resources (e.g. task verification uses resource “Inspection agent”).

Time consumption – For each task users can define how much time is consumed to complete. Consumption can be defined as simple (constant) number or with the probability distribution.

Cost – Users can define for tasks how much execution cost (in a financial way).

For gateways (element from BPMN) users can define probability of the next flow (e.g. probability for "yes" than "no"). *Bizagi Modeler* has different levels of simulations.

Process validation – User defines a duration of simulation and amount of generated instances (number of requests). Gateways are also configurable. After simulation ends, the result with summarization is displayed and the user can check if process behaves as expected. For example, if the number of requests is equal to the number of completed instances. This could occur if there are issues with synchronization at parallel gateways.

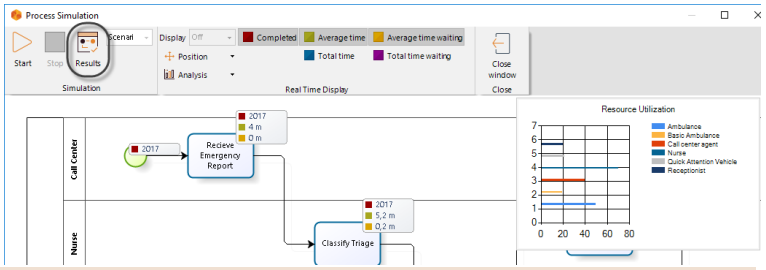
Throughput time analysis – Focuses only on the time measurement and the resources are not included. The user defines consumption for requests (interval time between two instances can be generated) and also for every task (how much time task gets to complete). Analysis counts total completed tasks and also average time and total time spent on each task. For example, it is useful for predictions, e.g. "how long customer will wait until his request is completed".

Resource analysis – Users can define resources and how much resource costs (fixed or cost per hour). Each task has defined which resources needs and how much to execute. Also, the cost of each task and time consumption are included. This analysis is more complex and serves as the real-world simulation to achieve better results, which can help to optimize processes. A simulation results example is shown at fig. 2.4.

2.1.3 The BPM Solutions Use Cases

Based on described analysis of this two concrete solutions, the basic use cases were identified. The use cases can be divided into two categories. The first category provides overviews over collected data and works only with existing data. This is helpful for analysis of the current state. Users have a great summarization of data which can really help them to make the decision in their business.

The second category offers functionalities to analyse and simulate processes. Analysts can define models and create the simulation based on them. From these simulations and results, analysts have precious data which they can use for optimizing processes before they are deployed. These analyses can have many purposes. One of them can provide analysis based on "What if..." where analysts define available resources, costs and so on. This analysis is helpful for pre-deploy analysis – the point where analysts starting to define processes and want to quickly verify efficiency. Another type of analysis can be "As is..." which is basically the first category. The third type can be



Resource	Utilization	Total fixed cost	Total unit cost
Call center agent	39,91 %	6.051	0
Nurse	69,75 %	15.115	0
Ambulance	49,76 %	30.180	134,13
Quick Attention Vehicle	21,40 %	11.124	15,86
Basic Ambulance	19,44 %	9.825	19,65
Receptionist	19,91 %	3.018	0

Figure 2.4: *Bizagi Modeler* Resource analysis example [18]

based on predictive analysis – this analysis use existing collected data and analysts define additional attributes. The analysis tries to predict behaviour of processes based on collected data and added attributes. This can help to optimize processes.

2.2 DEMO Role within Business Intelligence

The thesis focuses on "Business Intelligence", in other words on the fourth stage of BPM life cycle. The goal is to provide an approach how connections between business processes and collected data can be described. In chapter 1 the DEMO methodology was introduced and explained. Describing business processes from collected data within enterprise solution with DEMO brings mainly these advantages:

1. Clear assignment of responsibility – within DEMO, each transaction has clearly defined the responsibility of actor roles.

2. Reduced complexity — there is only and only one way how to describe an enterprise within DEMO.

The first stage of BPM life-cycle (design) within DEMO is well described by Dietz [1] [8]. However next stages of BPM life-cycle are only theories "on paper" and they are not stated to practice. For example, *M. Skotnica* [2] with

his thesis focused on first three stages and introduced an approach of BPMS based on DEMO. But the next stages are not examined yet.

This thesis focuses on fourth stage (the monitoring) and the premise is that underlying BPM solutions are not necessary based on DEMO. The goal is to aggregate data from any enterprise solution, connect them with modelled enterprise within DEMO methodology and display them to the user. The main difference is with the feature of process visualisation itself. Systems based on BPMN visualise processes within BPMN which, as was told, can be more complex, inconsistent and not so clear as approach with DEMO.

In this chapter, firstly more details about BPMS were provided. On the two concrete solutions available were taken the core concepts of process modelling and their visualisations. The role of DEMO within BPMS and Business Intelligence was explained. In the next chapter, the proposed approach itself is introduced and described.

Chapter 3

Proposed Approach

The main goal of this thesis is to propose an approach which will be able to create the graphical overviews of business data and make decisions based on them. This chapter describes each part of the proposed approach. The chapter is divided into these parts:

1. Which kind of data the application collects and how works with them.
2. How real-time process visualization is done.
3. What kind of overviews the application can display.

3.1 Business Data Aggregation

Each enterprise system collects precious data within the business. These data are important in many ways. One is for business analysts. The analyst can analyse processes and optimize them with these collected data. Although every enterprise solution has differently defined structure of these data and every BPMS has a different approach how it works with them, the core concept of how data can be described remains the same. Within BPMS there are elements which can be transformed always to the DEMO concepts.

- There are always some kind of *Actor roles* and concrete *Actors*.
- The processes itself can be transformed to *Transactions*. In chapter 1 was explained that each process can be described on ontological level and has precise definition what kind of process it is and the responsibilities of *Actor Roles*.
- Each BPMS provides some kind of events, which can be used to determine what exactly happened and how it is connected to the model defined within DEMO.

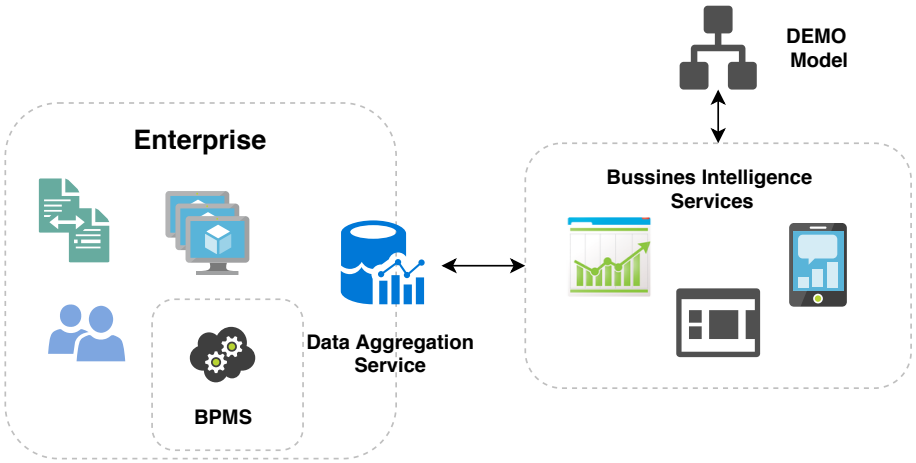


Figure 3.1: Business Intelligence based on DEMO

In this way, the system of visualisation based on DEMO can be nearly independent from other systems within the enterprise. The architecture of Business Intelligence based on DEMO is shown at fig. 3.1. The enterprise does not need necessary a BPMS. Within an enterprise, many systems and applications exist. Even without BPM solutions within the enterprise, there are still the same kind of processes and actor roles which can be modelled through DEMO.

Many systems can be inside and enterperise and (not necessary) in front of them can be a BPM solution. On the "borders" of the *Enterprise* is placed *Data Aggregation Service*. This service has the responsibility of getting data from an *Enterprise* and transforming them into the structure that *Business Intelligence Services* can use. Collected data from *Data Aggregation Service* are mapped to defined *DEMO Model*. With this approach, *Business Intelligence Services* can take advantages of DEMO without the requirement to be dependent directly on enterprise systems.

3.1.1 Business Data Domain Model

The one of the most important thing what needs to be defined is how collected data can be expressed for visualisation. The fig. 3.2 shows domain-data model which describes how DEMO model can be described.

Figure 3.2 is divided to three parts:

Model – Defines how DEMO model can be described. It corresponds with OCD and PSD models.

Instance – The one concrete running process. It indicates state of transactions.

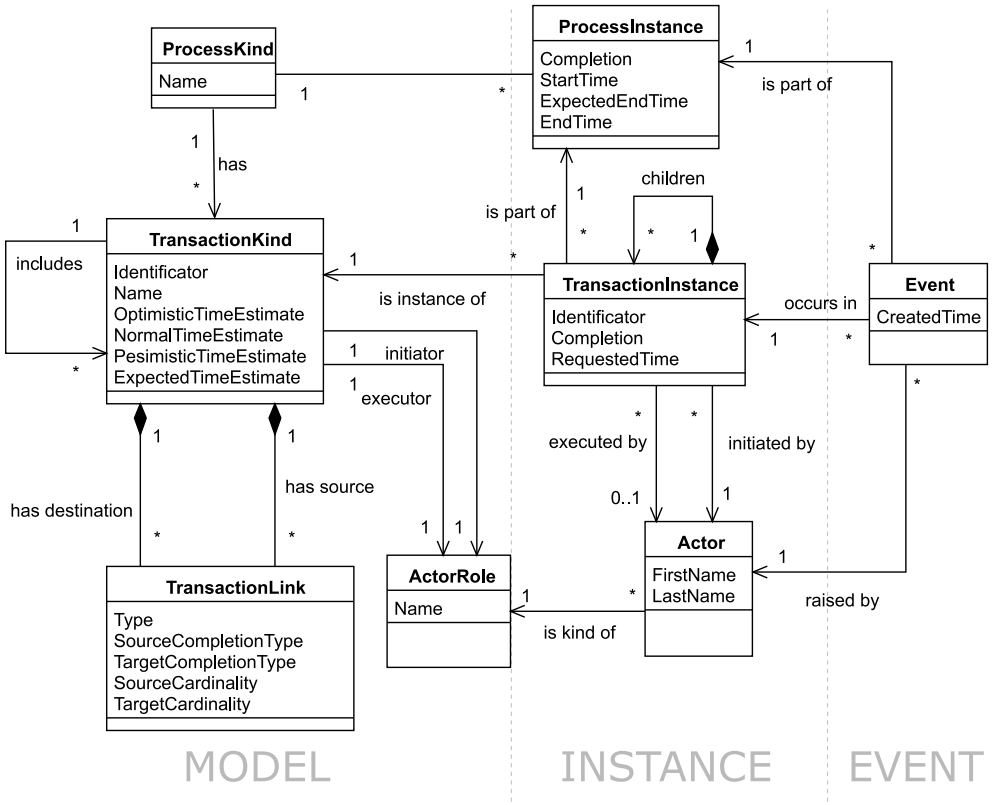


Figure 3.2: Domain model of collected data

Event – Last part defines the concrete occurred events from internal systems which can be collected and transformed to data, that can be used.

The brief explanation of each entity from domain model:

Actor Role – Defines role within enterprise.

Actor – Defines concrete user within enterprise, which has assigned *Actor Role*.

Process Kind – Defines exactly one concrete business process inside the enterprise. Each *Process Kind* has linked many *Transaction Kinds*.

Process Instance – The instance of concrete *Process Kind*. Defines process *Completion*, date when process instance was created (*StartTime*) and expected time when whole process will be completed (*ExpectedEndTime*).

Transaction Kind – Defines the transaction kind. It has linked information, which *Actor Role* is initiator or executor respectively. Also defines the

time estimates for transaction completion. These three time estimates – *Optimistic*, *Normal*, *Pessimistic* are used to compute real expected time.

Transaction Link – Defines the *response* or *waiting* links. Each link has defined target *Transaction Kind* and source *Transaction Kind* to determine which two *Transaction Kinds* link connects. Link also defines *Source Cardinality* and *Target Cardinality* to define cardinalities. Moreover link defines which *C-Act* is assigned to source and target transactions.

Transaction Instance – Each *Transaction Kind* could have more than one instance. Transaction Instance includes information about *Completion* and *Requested Time*. The instance itself can have more than one chilf.

Event –Defines event which can occur during the running process. Each event has time creation (*Created*) and duration how long it takes between.

3.1.2 A Time Estimate Computation

Each *Transaction Kind* has three attributes – *Optimistic*, *Normal*, *Pessimistic* time estimate. Each one indicates how quickly given task could be completed. These attributes are set manually and it is an important decision which values are given. From this three attributes, the resulting one is computed.

Computation is done through technique *Three Point Estimation* [19]. The manager can take a simple average of this three attributes, however, the weighted average is more precise. The following formula comes from *Beta distribution* also known as *PERT*:

$$ExpectedTimeEstimate = \frac{O + 4 * N + P}{6}$$

where (O) states for optimistic, (N) for normal and (P) for pessimistic estimates. It indicates that normal estimate "can happen" most likely.

3.2 Case Study Rent-A-Car

Although all proposed approaches are defined generally, the case study is used to demonstrate usage it is used as model example. The case study is taken from the book by J.Dietz [1]. A Rent-A-Car is a company which offers cars for rent. The simplified description how the company works:

Rent-A-Car (RAC) is a company that rents cars to persons or representatives of legal bodies (e.g. companies). RAC operates from over fifty branches in Europe. Many cities have more than one branch and typically branches are located near all airports. Customer orders are placed through several channels: walk-in, telephone, fax or email. Walk-in customers are usually people

who want to rent a car immediately. In all cases, an electronic rental form is filled out by one of the desk officers. After the renter has signed the contract, the rental is concluded by the desk officer. On the starting day, the driver can pick up a card at the distribution department. When driver shows up, RAC employee checks whether there is a car available. If there is one, he prepares car and sign contract as being picked up. If there is no car, he upgrades the contract and select a different car. After the car of rental has been dropped off, there is chance to pay some fines. There are many types of penalties.

The OCD and PSD of RAC are shown at fig. 3.3 and fig. 3.4 respectively.

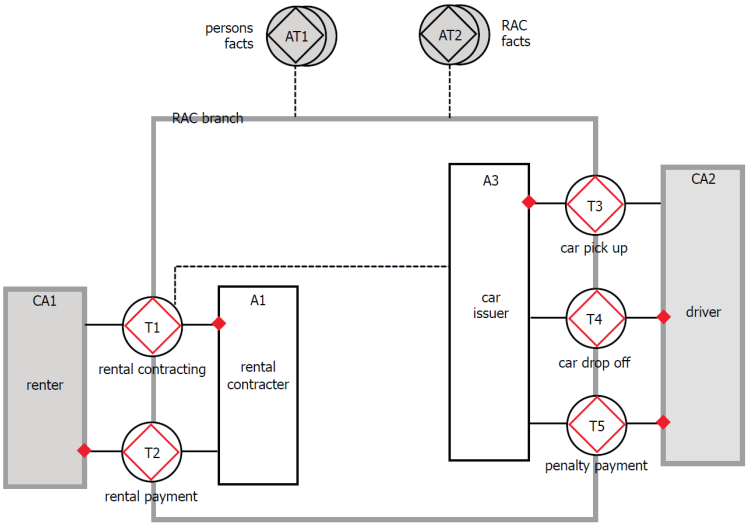


Figure 3.3: Rent-A-Car OCD diagram [1]

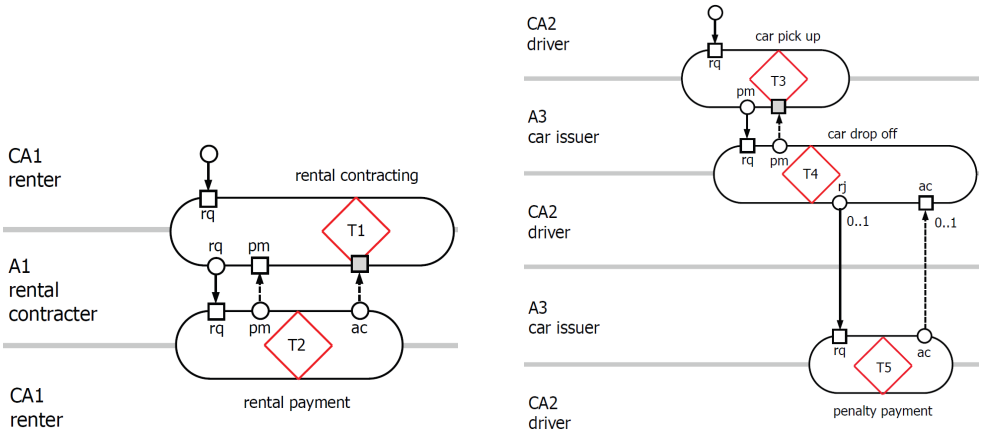


Figure 3.4: Rent-A-Car PSD diagrams [1]

3.3 Data Labelling

As has been said, every system within enterprise provides some collection of data. Problem is, if enterprise does note use centralized solution (a some form of BPMS), format of collected data can be very different. However collected data can be structured and then processed via procedure, named "labelling". Every system can provide collected data with some structured form. Then sooner defined *Data Aggregation Service* must have mechanism how this structure can be mapped to defined DEMO model. This mechanism will be unique for every enterprise and theirs systems, however fundamental idea and work-flow is the same.

Firstly used terms are defined:

log – The structured collected data from enterprise systems

case id – The defined process.

creation timestamp – Timestamp when created.

end timestamp – Indicates end of given task.

activity – An activity (task) within case.

resource – A resource associated with activity. Typically an user.

role – An user role which invoked an activity.

Table 3.1 shows a shortened log from *Rent-A-Car* company. The "case id" is equal to 1 (which is generally some internal identifier) and it is omitted from the table. Also, timestamps are omitted because they are not, for now, important.

There are several activities, resources and roles. "Labelling" mechanism must distinguish each activity and assign appropriate *Transaction* and *Coordination-act* from DEMO model. Also "roles" must be assigned to *Actor roles* and "resource" is assigned as *Actor*. The shortened example of "labelling" is shown at table 3.2. The "role" and *Actor roles* are absolutely the same.

Activity	**Resource**	**Role**
Create Rental Requisition	Peter Freeman	Renter
Create Request for Payment	Alice Gree	Rental Contracter
Payment Promise	Peter Freeman	Renter
Analyse Rental Requisition	Alice Gree	Rental Contracter
Make a Payment	Peter Freeman	Renter
Accept Payment	Alice Gree	Rental Contracter
Create a New Contract	Alice Gree	Rental Contracter
Hand Over Contract to Customer	Alice Gree	Rental Contracter
Customer Accepted Contract	Peter Freeman	Renter
Create Car Pick Up Requisition	Peter Freeman	Driver
...		

Table 3.1: The example log of Rent-A-Car company

Activity	**Transaction**	**State**
Create Rental Requisition	T1	Request
Create Request for Payment	T2	Request
Payment Promise	T2	Promise
Analyse Rental Requisition	T1	Promise
Make a Payment	T2	State
Accept Payment	T2	Accept
Create a New Contract	T1	Execute
Hand Over Contract to Customer	T1	State
Customer Accepted Contract	T1	Accept
Create Car Pick Up Requisition	T3	Request
...		

Table 3.2: Activities labelling from RAC log

3.4 Process Instance Visualisation

System must have ability to visualise running process instances in real-time. User can open running process and can be informed how it continues.

The biggest challenge was how to propose a way to visualise models modelled with DEMO. Moreover, the intent was mainly to provide visualisation on mobile devices (smart-phones, tablets).

There are four conditions how visualisation must be proposed:

i The proposed way must preserve the intention of the modelled model through DEMO

ii Visualisation must be consistent accordingly to defined DEMO model.

iii The proposed way must be easy to use and understand.

iv It must be adaptive to different sizes of mobile devices.

The first idea of visualisation was highly inspired from investigated existing solutions (*Process Maker*, *Bizagi Modeler*). Visualisation could be done with DEMO OCD model itself. With this approach user could have:

1. The exactly same understanding of modelled enterprise.

2. The straightforward visualisation of instances through defined models. If user understands DEMO models, this approach of visualisation has the same meaning.

With this approach first two conditions are accomplished:

1. Defined enterprise is directly exposed with OCD model. However, some information is hidden. As is known, the PSD brings more information how processes are defined and how they work (i).

2. Consistency is tied with OCD itself (ii).

However, the (iii) and (iv) are not so straightforward. The third condition (iii) is not accomplished because DEMO is used and it does not have to be easy to understand. In every enterprise, there are specialists (business analysts) that have the responsibility to analyse given enterprise and specify models (for example within DEMO). Other people do not need to have knowledge about this methodology, overall they do not need to have knowledge about this models either. Although OCD itself is very concise, it is not so suitable for smaller screens of mobile devices. This implies, that (iv) is also violated.

3.5 A Better Option

Gantt chart is widely used in time-management projects and systems. It is well known, established and used. It was found that Gantt chart offers the best approach, how tasks can be visualised. Chosen approach is highly inspired by it and visualisation is based on it.

The visualisation is divided into two parts:

1. The "Gantt-chart canvas" where each transaction has own form of progress bars which displays transaction's current state.

2. Time-line that offers chronological overview which events already occurred and when.

3.5.1 Transaction Box Element

Transaction box is associated rectangle with the transaction which indicates the current state of the transaction. The state corresponds to actual C-Act within *Transaction Pattern*. Current state also indicates progress value which is displayed as inner rectangle inside transaction box filled with colour. Progress value is computed as a percentage value from the current state. The C-Act *Request* corresponds to 10 %, *Promise* to 25 %, *Execute* to 50 %, *State* to 75 % and so on.

The transaction box has four colour stages. **White** for not requested transaction yet (the box has also dashed border), a **green** which indicates "happy path" through *Transaction Pattern*, a **yellow** if transaction is *rejected* or *declined* and **red** for transaction which is *stopped* or *quitted*.

The transaction boxes are placed accordingly to OCD model. That means:

- The transaction which "starts" whole process is placed as first.

- If the transaction is dependent on another, it is placed "under" it and shifted to the right. Also, that means, that "parent" transaction is wider than their descendants.

3.5.2 Transaction Link Element

As has been said, PSD model defines associations between transactions. Each transaction can have many *response* links and *waiting* links. For better understanding, how the process is defined, these links are included.

The links appearance is taken from PSD itself.

- Response link is displayed as a straight line with an arrow at the end. On one side it has a circle with *source* C-Act, on the other side with an arrow it has square with *target* C-Act.

- Waiting links are displayed as response link, but the line between is dashed.

Links are placed between appropriate transactions accordingly to *source* and *target* C-Acts, just like at PSD. However, if *response* link has *target* C-Act as "Requested", the *target* C-Act is omitted and link itself is displayed as angled arrow which points to "start" of *target* transaction. It is absolutely clear, that this type of response means exactly "If given C-Act at source transaction occurred then target transaction is also requested".

Example shown at fig. 3.5 means exactly: "When transaction T1 is *promised*, T2 is *requested*. Then T1 can be *stated* only if T2 was already *promised*. Also T1 can be *accepted* only if T2 was *stated*."

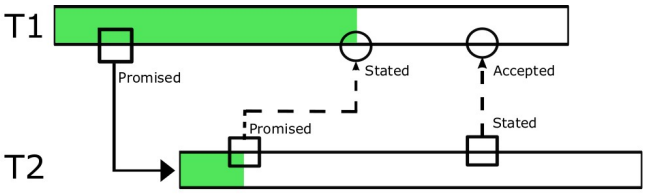


Figure 3.5: The transaction links example

On the canvas's left side are placed transaction labels. Label corresponds to transaction's name and identifier. The labels are structured to tree-form like in folder explorer. Each transaction box has same height position as the corresponding label.

With this approach, the user has a quick and easy way how can check the state of the concrete process. The user can easily determine which transactions are not yet completed and how they are connected together.

However the time-based information still missing and that is the point, where time-line comes in.

3.5.3 Time-line

The main purpose of time-line is to provide the chronological overview of incoming events. The time-line is scalable. This means, that user can zoom in or out to the level of details. Typically more events can occur within one minute, so it is important to have the ability to zoom to more detailed view.

Each event is marked with a timestamp (when the event occurred) and what kind of event it is. If the event is kind of "C-Act occurred", the event has also information about the associated transaction with C-Act's abbreviation. To help the user more easily distinguish which transaction is associated with given event, each transaction has own colour, that is reflected into the colour of text within time-line.

Time-line is independently controllable and provides another point of view to the state of the currently running process.

One note aside, the first idea was to provide connected associations between time-line and transaction boxes. The intent was to provide visuals (anchors) within transaction box that could indicate that "at this point something happened". If the user 'click" on this anchor then a visual connection between the anchor and associated event within time-line appears.

With this approach, the canvas must be capable of responding to time-line "stretching" – transaction box must resize accordingly to positions of events within time-line. Although it makes perfect sense from user "point-of-view", the usability within mobile-devices is reduced. Each event within time-line must be readable and with adding of events, the width of transaction box grows. The result is, that each transaction box is really wide and it becomes very unclear and nearly unusable.

The result is that the visualisation is divided into two views. First one, the time-line offers chronological overviews of occurred events associated with transactions. The second one offers the overview current transactions states and the associated links between them.

The earlier four defined conditions for visualisation are fully accomplished.

i "The proposed way must preserve the intention of modelled model through DEMO"

The defined model through DEMO is displayed within *Gantt-chart canvas*. OCD is transformed to *transaction boxes* and PSD is transformed to *transaction links*.

ii "Visualisation must be consistent accordingly to defined DEMO model."

The OCD model is transformed to another form of view, but the name of transactions and associations between them remains. The links from PSD are taken as they are, without any significant changes.

iii "The proposed way must be easy to use and understand."

As has been said, the Gantt charts are well known and many people on management positions or analysts naturally understand this concept of views. The *transaction boxes* easily show current state of given transaction and *transaction links* show associations between them.

iv "It must be adaptive to different sizes of mobile devices."

The time-line part is perfectly adaptive to various screen sizes. The "Gantt-chart canvas" is more concise than larger OCD and it can adapt to smaller screens more easily.

3.6 The Dashboard

Although the "process visualisation" brings usable kinds of overviews, there are many types of overviews which can't be done through this. As has been

said in chapter 2, typical BPM solutions offer some kind of "dashboard" where these various types of overviews are placed together.

The dashboard's main goal is to provide:

1. The comprehensive and effective way how to get precious information over collected data.

2. Ability to provide highly customized and individual types of overviews. A company will want to propose a different overview for sales manager than for accountant and so on.

3. The overviews must be easy to understand and must be very clear, what they display.

3.6.1 Data Query

To provide a specific view of collected data, it is required to process collected data specifically according to defined overviews. For this purpose, the *data query* is defined.

To put it simply, *Data Query* is named query over collected data. This can be also compared to "stored-procedure" within DBMS (Database Management System). From the end-user point of view, the user only select defined query and data are returned. Within *Data Aggregation Service* query is defined and transformed to "real" query over collected data. In most of the cases, it will be transformed to SQL-like query.

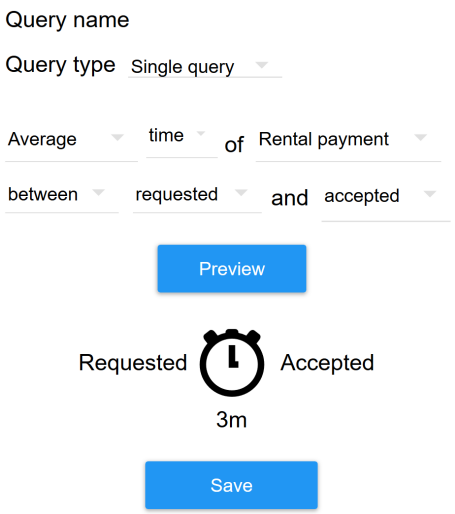


Figure 3.6: The Query editor prototype example

In BPM solution (based on DEMO) is defined *query editor*. It is simple, form-based editor, where employees within an enterprise can define *data*

queries. The user defines *name* of the query and builds the query with interactive guide. For simplicity, the guide can be imagined as interactive way how to build a SQL query with clauses as "join, where, group by, order by, ..." Result is, that editor allows to non-technical users still define basic queries over collected data without requirement to have knowledge, how concretely "grab" the data. The example of query editor (as a prototype) is shown at fig. 3.6.

3.6.2 Widgets

Each overview within dashboard is called *widget*. Widgets provide one concrete type of view. Widgets have defined *name* and associated *data query*. Each widget is highly customisable, but they are sorted to several base types.

3.6.3 Summary Widget

The summary widget provides, by its name, summarization of collected data. It is commonly displayed as "Pie chart". The typical usage, for example, can be: "Summary of the ratio between Accepted and Declined contracts" or "Actual state of all contracts, number of requested, stated, declined, accepted and so on", ...

3.6.4 Time Based Widget

This widget displays data within defined time interval. It uses "Column chart" (each *time x-value* is displayed as column with height of *y-value*) or "Line series chart". An example of typical usage can be: "Monthly income-expense", "Daily number of new contracts", ...

3.6.5 Single Value Widget

A single value computed from collected data. This can be for example "Actual number of unresolved contracts", "Actual financial balance", ...

3.6.6 Widgets As Placeable Elements

The dashboard can be divided into "grid layout" – an unspecified number of rows and from one up to three columns. Each widget has defined the *width* and the *height* – how many columns and rows it takes. Widgets can be arranged as needed.

3.7 The End-User Mobile Application

As has been noted, the visualisation is proposed as "mobile friendly". The concept of the dashboard, widgets are also considered mainly for mobile devices. The mobile application provides these basic functionalities:

- The dashboard with widgets.
- The list of process instances.
- The real-time process instance visualisation.

In the enterprise, the dashboard and widgets will be most likely predefined by eligible employees (e.g. business analysts). The end-users (e.g. managers) will only have access to the dashboard at "read-only" mode – they will not be allowed to edit widgets. This implies that within the *Business Intelligence Service* and *Data Aggregation Service* the *data queries* and the *widgets* are defined. The mobile application takes these definitions and display them. Within these services, the default arrangement of widgets is defined. The end-users can rearrange them as they want.

The users can also list all process instances and view visualisation of them. Also, the application offers the currently running instances, where the user will see visualisation in real-time.

3.8 Another Visualisation Approaches

The main approach how visualise processes were explained. However, there are many other ways how to provide "look" at the data and processes.

3.8.1 OCD with Summary Information

At the beginning of this chapter has been said, that OCD is not suitable for real-time visualisation, but it still has a valuable information and it can be suitable for another approach.

On top of the diagram, the collected summary of information can be shown. For example on top of each transaction can be the summary of success rate – the ratio between accepted and failed transactions. The example is shown at fig. 3.7.

3.8.2 Heat map

An interesting visualisation is heat map. The OCD can be used again. The heat map can display various information. For example, for each transaction is collected average "executing time" – the time how long *execution phase* takes to finish. The more waiting time it is, the more heat around transaction is.

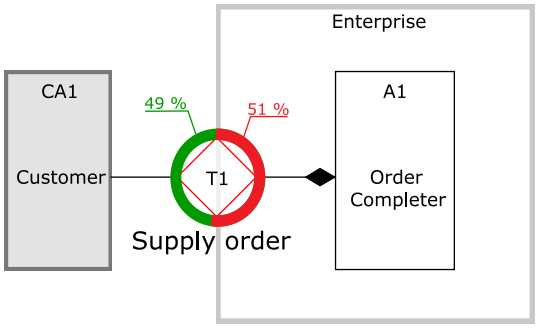


Figure 3.7: OCD with summary information

In other words, more heat means more red colour around. This can be easily used to indicate "bottle-necks" within defined processes – to find problematic transactions where the improvements steps are required. An example is show at fig. 3.8

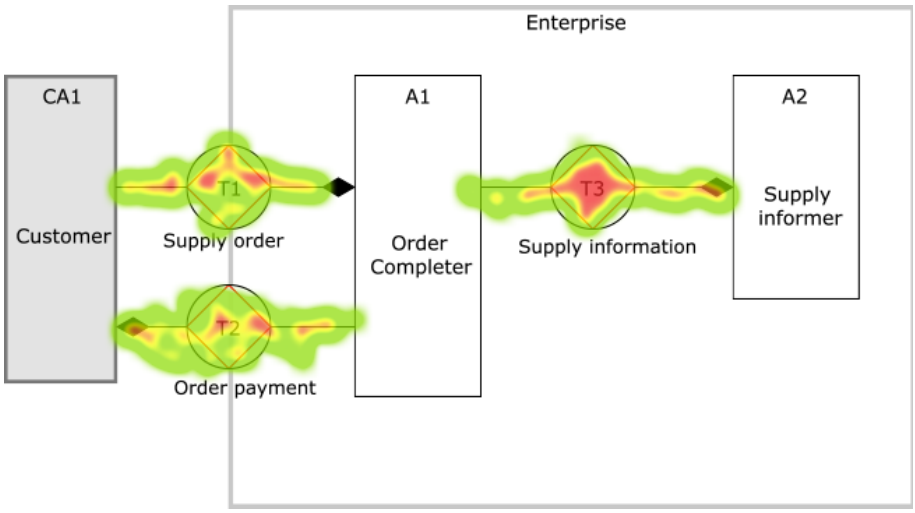


Figure 3.8: OCD heat map example – average executing time

Chapter 4

Proof of Concept

To verify proposed approaches, the mobile application as proof-of-concept was implemented. For verification was chosen especially process instance visualisation.

The given application offers:

1. Process instance visualisation with simulated data
2. Concept of dashboard

Frstly the used technologies are introduced. Then the brief architecture of mobile application and implementation itself are explained. In the end, the issues with implementation are described.

4.1 .NET Platform

For implementation is used .NET platform and programming language C#. The .NET is free, open-source and cross-platform framework for building many different types of applications [20]. There is a lot of frameworks within platform, such as WPF for building native desktop applications for Windows or ASP .NET framework for building websites and server-side applications. Also there is cross-platform Xamarin framework for building mobile applications. These all frameworks were used to implement proof-of-concept.

4.1.1 .NET Standard

Earlier years within .NET platform was problem with many different implementations of .NET itself. There was the "original" (full) .NET. Then a newer, open-source and cross-platform version .NET Core and third was Xamarin. Each platform have different base libraries and if developers wanted to share code between these platforms, there was a lot of work to do.

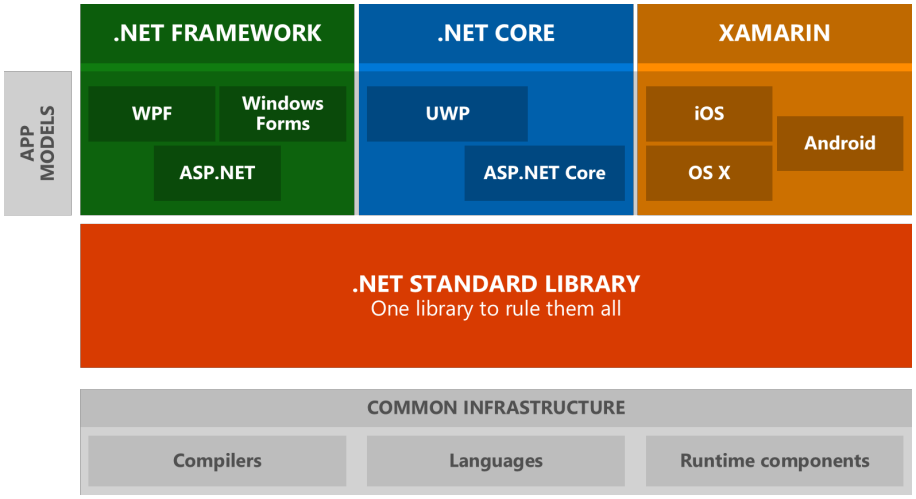


Figure 4.1: The .NET platform overview [21]

The new approach was introduced [21]. The .NET Standard is the unified platform which defines set of APIs that each .NET subset platform must implement. This allows to developers easily share code and write libraries that can be used anywhere. The great graphical summarization is shown at fig. 4.1. Actual version is .NET Standard 2.0. The newer version 2.1 will be introduced within Q2 2018.

4.1.2 .NET Core

The .NET Core is open-source, cross-platform version of .NET Framework. It is new and highly frequently updated framework which becomes very popular. It has many "ports" of frameworks from full .NET such as ASP .NET or Entity Framework – an Object-Relationship-Mapping tool for databases.

4.2 Xamarin

Xamarin is platform for developing native mobile applications in cross-platform way. Developers use C# language and .NET Standard to implement mobile application. When the application is being build, the code is compiled to native platform code.

Following summarization is taken from [22]:

- On iOS, C# is ahead-of-time (AOT) compiled to ARM assembly language. The .NET framework is included, with unused classes being stripped out during linking to reduce the application size. Apple does not allow runtime code generation on iOS, so some language features are not available.

- On Android, C# is compiled to IL and packaged with MonoVM. Unused classes in the framework are stripped out during linking. The application runs side-by-side with Android runtime.
- Windows and UWP use directly C# and included .NET.

Within Xamarin there are two main approaches how to build applications (see fig. 4.2). First one is commonly called "Xamarin Native" or "Xamarin.*" where * is targeted platform. With this approach, the code for business logic is written independently from target platform, but user interface for each platform is implemented separately with native approaches (e.g. for Android developer uses XML based declarations).

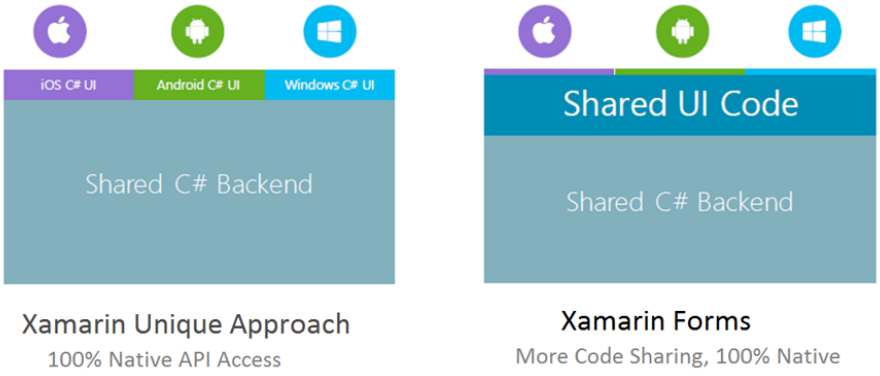


Figure 4.2: The Xamarin.* vs. Xamarin.Forms [23]

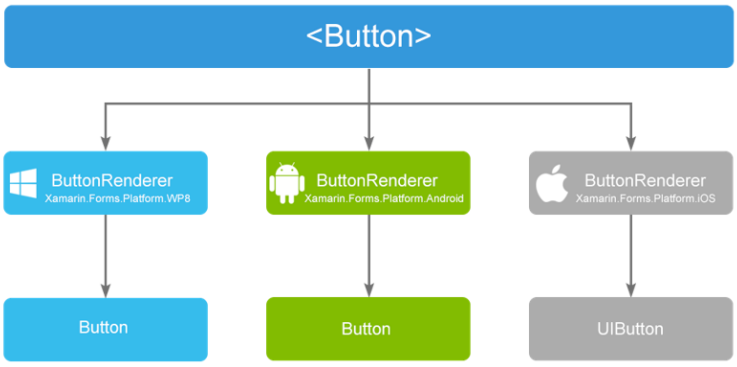


Figure 4.3: The Xamarin.Forms control renderer [23]

Second one is called Xamarin.Forms, which is build on top of first one and allows to define user interfaces in cross-platform way. Developers use XAML, the declaration language similar to XML to define user interface. When application is being build, defined controls are transformed to native controls. This

transformation is done through concept called "Control renderer". Within shared library, an interface for control is defined. On each platform is implemented renderer which transforms given control to native one. An example is shown at fig. 4.3.

4.3 Another Used Technologies

The implementation is divided to two parts. First one is mobile application and second one is server-side, the concept of *Data Aggregation Service*.

For server-side implementation was chosen ASP .NET Core – a newer version of ASP .NET, framework for creating server applications (services, REST API, ...) and websites.

Within ASP .NET Core a web application was created and library SignalR was chosen for real-time communication between clients and server.

4.3.1 SignalR

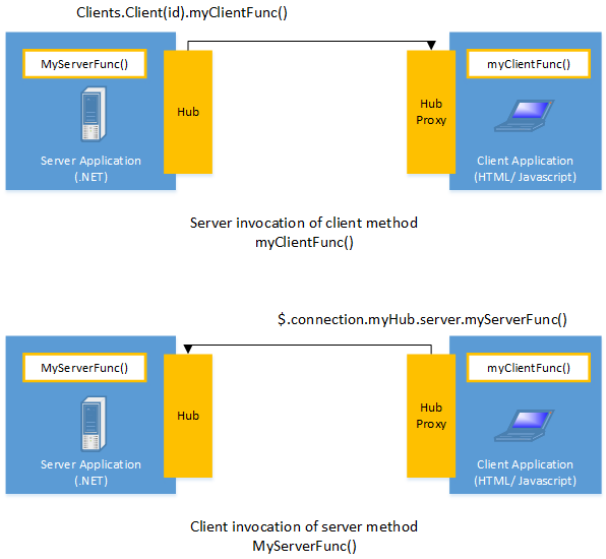


Figure 4.4: How SignalR works [24]

According to [24] SignalR is: "a library for ASP .NET which offers adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data". SignalR handles "many things" for developers automatically, such as connection management or choosing transporting types. SignalR firstly try to use WebSocket and if it fails, automatically fall back to older transports.

On the server, developer defines a *Hub*. *Hub* defines contract which clients will use and communicate through. On the client-side, developers creates *Hub Proxy* – connection to specific *Hub*. The example how it works is shown at fig. 4.4.

4.4 Implementation Decisions

For implementation were chosen a quite new and interesting technologies and combinations of them. The .NET Standard is for now the only one right way how to create shared C# back-end. Older approaches with Portable Class Library (PCL) are deprecated and Xamarin recommends to use .NET Standard. Although it is recommended and PCL are deprecated, many problems occurred. More discussion about problems with implementation is in the end of this chapter.

For mobile-application was chosen Xamarin.Forms for this reasons:

- Author has many experiences with XAML from similar frameworks such as WPF or Universal Windows Platform.
- Xamarin.Forms are more faster for developing with cross-platform way – the user interface is written once.

Xamarin.Forms offers many prepared controls for use, but for creation of process instance visualisation a custom controls were created.

A many different approaches were tried. First one was edit already existing controls, but it was found, that it is not suitable. Another approach was to find existing controls, which resulted to, that the requirements are too specific and required controls do not exists. Third option was to use some basic 2D engine for custom drawing and rendering. The third option seemed as best option to do, so it was chosen.

4.4.1 SkiaSharp

SkiaSharp library [25] provides a way how to quickly and easily do custom 2D drawing. It is used for process visualisation. Each *transaction box* is custom control created with SkiaSharp's canvas – a place-holder for custom drawings. Each *transaction link* is also custom controls made with this library.

4.4.2 Syncfusion Controls Library

Syncfusion Controls Library [26] adds many new controls to Xamarin.Forms. The Dashboard uses this library for its various types of charts.

4.4.3 Simulation Cases

Main goal of proof-of-concept is to verify that proposed visualisation approach is usable. In real-world usage, many other systems and services must be developed. Also there is not any "real-world" data source. For this reason, the simulation approach was chosen. As base model is chosen Rent-A-Car company. From RAC model several simulation cases are preapred.

4.5 Implementation Details

Only fundamental implementation decisions and circumstances are described. At the fig. 4.5 is shown "high-level" architecture of application:

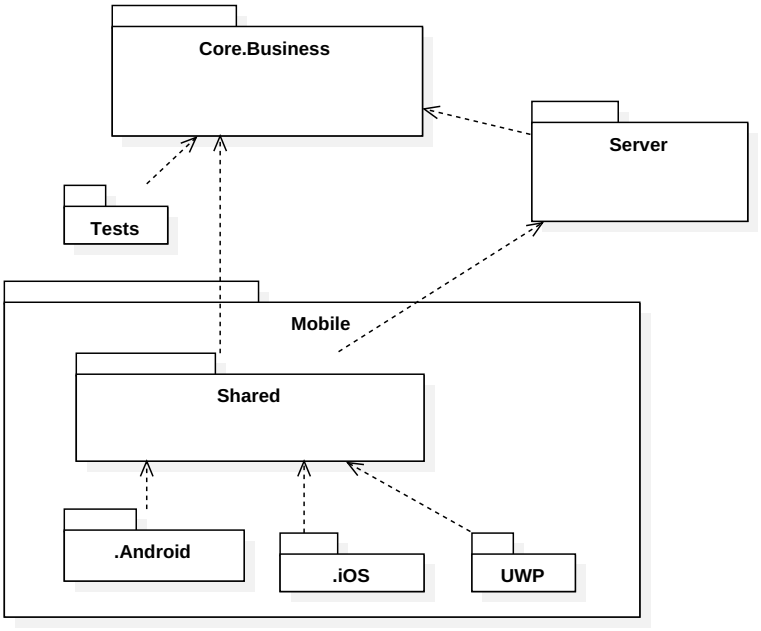


Figure 4.5: The implementation packages

Core.Business – The *Data Domain Model* is placed here, because mobile and server applications use it. Also classes for simulations and XML parsers are there.

Server – The simple ASP .NET Core with SignalR application for serving a simulation data.

Mobile.Shared – A shared part of Xamarin application. All business logic and user interface are implemented here.

Mobile.Shared.Android – A Xamarin.Android project. There are Android specific implementations and initialization for Xamarin.Forms. Also all the graphic assets (logo, icons, ...) are placed in this project.

Mobile.Shared.iOS – A Xamarin.iOS project. Same purpose as Mobile.Shared.Android.

Mobile.Shared.UWP – A Xamarin.UWP (Universal Windows Platform) project. Same purpose as Mobile.Shared.Android.

Tests – Package contains several unit tests for Core.Business.

4.5.1 The Dashboard

As has been said, Dashboard uses several charts from Syncfusion library. Each chart is predefined and customized for better look.

4.5.2 Simulation Cases

Each simulation cases is defined as XML file. Because creating and writing testing data within XML is not so comfortable, the simple utility was implemented. The utility allows interactively define new simulation case.

Each simulation case has three parts:

- First part defines *Actors* – the instances of *Actor Roles*.
- Second part defines *Process Instance* with *Transaction Instances*.
- Third part defines simulation *chunks*. Each chunks store one or more *Events*.

When simulation case runs, every "simulation step" new chunk is loaded and all events from chunk are returned to client. With this approach, it can simulate incoming data from real data source.

The utility allows to define chunks and events with interactive editor, where user can simply fill transaction id, event type, time creation and type of C-Act. An example of simulation case:

```
<?xml version="1.0" encoding="utf-8"?>
<Simulation Name="Case02">
  <Actors>
    <Actor Id="1" ActorRoleId="1" FirstName="George"
    ↪  LastName="Lucas" />
    <Actor Id="2" ActorRoleId="2" FirstName="George"
    ↪  LastName="Lucas" />
    <Actor Id="3" ActorRoleId="3" FirstName="Bob"
    ↪  LastName="Freeman" />
```

```
    <Actor Id="4" ActorRoleId="4" FirstName="Alice"
    ↪ LastName="Freeman" />
  </Actors>
  <ProcessInstance Id="1" KindId="1" StartTime="01-02-2018
  ↪ 15:34:23" ExpectedEndTime="01-02-2018 15:34:23">
    <TransactionInstance Id="1" KindId="1" Identificator="T1"
    ↪ CompletionType="0" ProcessInstanceId="1" InitiatorId="0"
    ↪ ExecutorId="0" ParentId="0">
      <TransactionInstance Id="2" KindId="2" Identificator="T2"
      ↪ CompletionType="0" ProcessInstanceId="1"
      ↪ InitiatorId="0" ExecutorId="0" ParentId="1" />
    </TransactionInstance>
    <TransactionInstance Id="3" KindId="3" Identificator="T3"
    ↪ CompletionType="0" ProcessInstanceId="1" InitiatorId="0"
    ↪ ExecutorId="0" ParentId="0">
      <TransactionInstance Id="4" KindId="4" Identificator="T4"
      ↪ CompletionType="0" ProcessInstanceId="1"
      ↪ InitiatorId="0" ExecutorId="0" ParentId="3" />
      <TransactionInstance Id="5" KindId="5" Identificator="T5"
      ↪ CompletionType="0" ProcessInstanceId="1"
      ↪ InitiatorId="0" ExecutorId="0" ParentId="3" />
    </TransactionInstance>
  </ProcessInstance>
  <Chunks>
    <Chunk>
      <Event Type="0" TransactionId="1" TransactionKindId="1"
      ↪ RaisedById="1" Created="01-02-2018 09:00:00">
        <CompletionChanged Completion="Requested" />
      </Event>
    </Chunk>
    <Chunk>
      <Event Type="0" TransactionId="1" TransactionKindId="1"
      ↪ RaisedById="4" Created="01-02-2018 09:01:00">
        <CompletionChanged Completion="Declined" />
      </Event>
    </Chunk>
    <Chunk>
      <Event Type="0" TransactionId="1" TransactionKindId="1"
      ↪ RaisedById="1" Created="01-02-2018 09:01:10">
        <CompletionChanged Completion="Quitted" />
      </Event>
    </Chunk>
  </Chunks>
</Simulation>
```

4.5.3 Process Visualisation

The process instance visualisation is done as follows:

1. Simulation case is loaded through `SimulationProvider`.
2. Process instance from simulation case is passed to `GanttChartBuilder`. This class is responsible for creating *Transaction Boxes* and *Transaction links* controls and wiring them together.
3. Created controls are inserted to `RelativeLayout` control. This control allows to place controls relatively to another.

A note about the Transaction Box Control. It derives from `SKCanvasView`, which comes from SkiaSharp. The class provides method `OnPaintSurface` which is called every time when the graphics must be refreshed. An argument `SKPaintSurfaceEventArgs` provides `SKCanvas` which contains methods for basic 2D drawing.

4.6 Application Testing

The mobile application, as a whole product, was tested manually. Several people got acquainted with the application. The main part of the manual testing was done on process visualisation, where the main issues were with usability.

Based on this manual testing and response from testers, the whole concept of visualisation was modified several times.

- Application force landscape orientation for process visualisation. The portrait orientation is not useful.
- The time-line size is bigger due to the fact, that earlier version had the too small font and it was barely readable.
- As has been said in chapter 3, the first intent was to wire time-line and transaction boxes together. First implemented concept showed that due to the smaller screen sizes, the user must often scroll to get information – it was unclear and unusable. The result is nearly two independent parts where each serves another purpose.

4.7 Implementation Issues and Consequences

The implementation uses .NET Standard 2.0. This version is recommended for new Xamarin applications. But many used libraries, such as SkiaSharp (which is originally written as PCL) are ported to work within .NET Standard.

During implementation many issues occurred, especially with Xamarin and SignalR library.

4.7.1 Xamarin.Forms Issues

As has been said there are two main approaches how to develop with Xamarin platform. For implementation was chosen Xamarin.Forms. During implementation many issues with compatibility, run-time errors and bugs occurred.

Although Xamarin platform is not new and it is well used, there are still many issues, which make development very hard, sometimes nearly impossible. There are few common issues. For example:

- Sometimes shared code behaves differently on each platforms. For example, simple selecting item within ListView control (a list of items which is scrollable). On UWP platform, the ListView selected different item than the real one.

- Debugging application hangs with unknown errors.

- Many issues with compiling XAML and "fake" error messages.

Fortunately, this issues are addressed and they are in repairing process. These issues led to, that only Android platform is tested. UWP platform was during development omitted due to incompatibility with current version. The iOS platform was omitted at the beginning, because development for this platform requires device with macOS.

In the end, result is that Xamarin.Native could be a better approach. Xamarin.Forms are really great for form-based applications, but if developers want something really specific, it is better to define user interfaces within each platform separately.

4.7.2 SignalR Core Issues

The SignalR for now is in the development process. Although it is supported with .NET Standard 2.0 during development was find out, that there are incompatibility issues between SignalR Core, .NET Standard 2.0 and Xamarin platform. Again this issue is reported and it will be repaired with version 2.1 which will come out in Q2 2018.

This issue led to two things:

1. The server-side and mobile clients are separated. On the mobile client, the fake embedded server is included – simulation are loaded within mobile application itself and it behaves like server communication.

2. The server side is implemented and tested through simple JavaScript client, which works as expected.

4.8 The Result

The goal of implementation was successfully completed, despite the described issues. The process visualisation is implemented and proofed with prepared simulation cases. The mobile application is implemented as "Mobile Service for Rent-A-Car company". Following figures show the result.

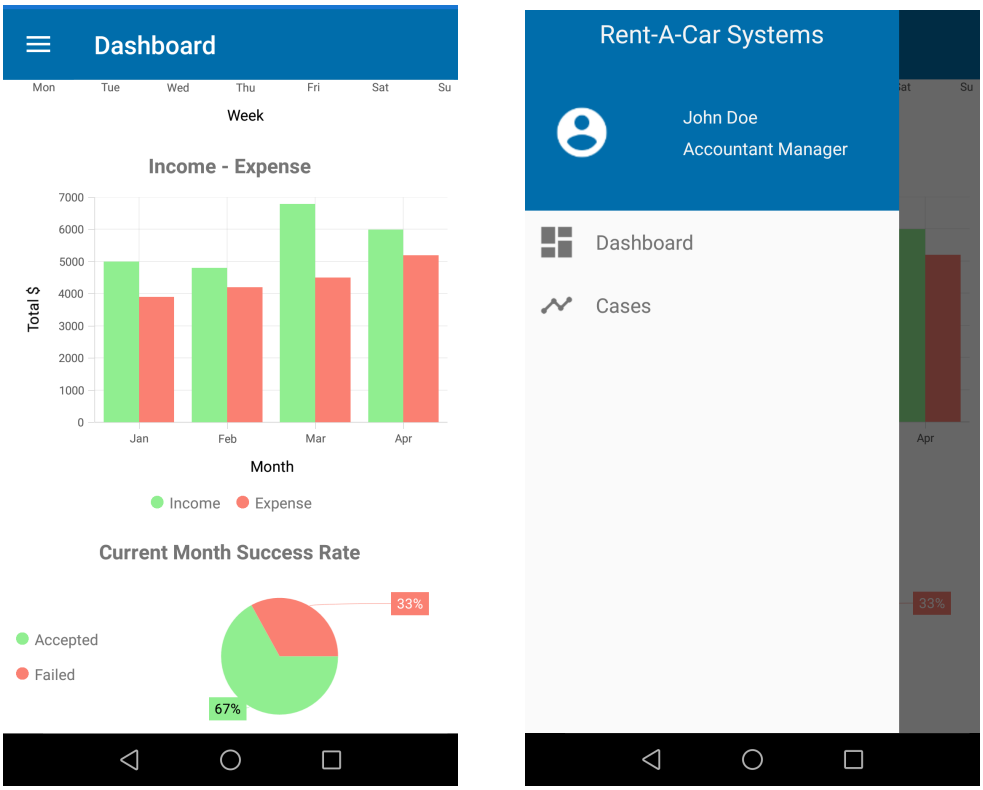


Figure 4.6: The Dashboard (on the left) and the menu (on the right)

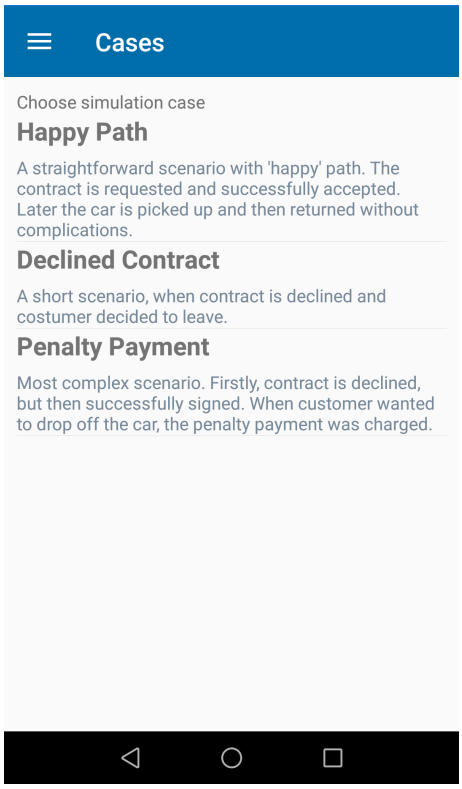


Figure 4.7: The predefined simulation cases

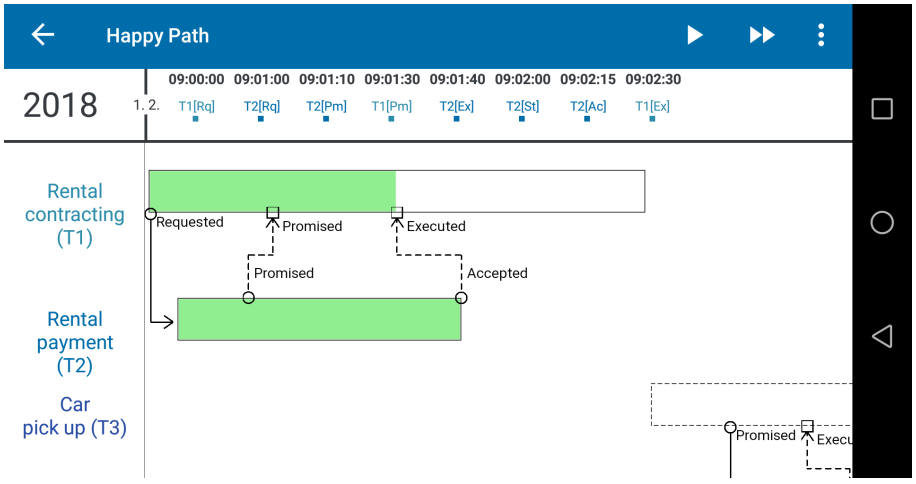


Figure 4.8: The process visualisation

Conclusion

Theoretical foundations were introduced in the introduction and first chapter. The existing BPM solutions which are commercially used were described in second chapter. The goal of thesis was mainly propose a way how visualisation of processes defined within DEMO can be done. A few approaches were introduced. Then followed explanation why chosen approach is the best for the visualisation.

The possibility of real usage was proofed through mobile application. The goal was to provide an approach, a simple to use and to understand, which seems to be accomplished. The visualisation is highly inspired from widely used Gantt charts, which are popular mainly in task-management. Also the concept of widgets within dashboard was described. These widgets provides user high valuable overview of collected data which can really help to improve their business.

Future steps

A visualisation itself could be defined more universally and could provide a detailed approach how any enterprise system can be mapped to defined DEMO models. As has been noted, a few existing solutions based on DEMO exist. The really next step could be to include this visualisation approach into existing BPM solutions based on DEMO. Thanks to this step a complete BPM solution could exists.

Bibliography

[1] DIETZ, J. L. G. *The essence of organisation*. Netherlands: Sapio Enterprise Engineering, second edition, 2015, ISBN 978-90-815449-4-8.

[2] SKOTNICA, M. *Towards the Foundations of Fact and Rules Ontology for Discrete Systems*. Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2016.

[3] HARVEY R. Koeppel. The digital effect on the BPM lifecycle. Mar. 2015, [online] [accessed: 19. 3. 2018]. Available from: `http://searchcio.techtarget.com/opinion/The-digital-effect-on-the-BPM-life-cycle`

[4] OMG. Object Management Group Business Process Model and Notation. [online] [accessed: 19. 3. 2018]. Available from: `http://www.bpmn.org/`

[5] MOTIVITY. Business Intelligence diagram. [online] [accessed: 20. 3. 2018]. Available from: `https://motivitysolutions.com/business-intelligence/`

[6] EM360TECH. Microsoft Business Intelligence example. [online] [accessed: 20. 3. 2018]. Available from: `https://www.em360tech.com/tech-news/microsoft-broadens-appeal-power-bi-business-intelligence-platform/`

[7] GRUBER, T. R. A Translation Approach to Portable Ontology Specifications. *Knowl. Acquis.*, volume 5, no. 2, June 1993: pp. 199–220, ISSN 1042-8143, doi:10.1006/knac.1993.1008. Available from: `http://dx.doi.org/10.1006/knac.1993.1008`

[8] DIETZ, J. L. G. *Enterprise ontology: theory and methodology*. Berlin New York: Springer, 2006, ISBN 978-3-540-33149-0.

[9] HAAN, J. d. Modeling An Organization Using Enterprise Ontology. 2009, [online] [accessed: 17. 3. 2018]. Available from: http://www.theenterprisearchitect.eu/blog/2009/10/10/modeling-an-organization-using-enterprise-ontology/

[10] VEJRAZKOVA, Z. *DEMO Design and Engineering Methodology for Organizations*. Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, Nov. 2013.

[11] DIETZ, J. L. G.; Hoogervorst, J. A. P.; et al. The discipline of enterprise engineering. *International Journal of Organisational Design and Engineering*, volume 3, no. 1, 2013: pp. 86–114, 00042.

[12] CCMI. Studenti FIT krotí zdivočelé diagramy. Feb 2018, [online] [accessed: 23. 3. 2018]. Available from: https://ccmi.fit.cvut.cz/studenti-fit-kroti-zdivocele-diagramy/

[13] QUARTZ. Behold the incredibly long flowchart of federal building permissions Trump showed today. Aug 2017, [online] [accessed: 23. 3. 2018]. Available from: https://qz.com/1235964

[14] GANTT. What is Gantt Chart. [online] [accessed: 19. 3. 2018]. Available from: http://www.gantt.com/

[15] GARTNER. Magic Quadrant for Intelligent Business Process Management Suites. Oct 2017, [online] [accessed: 23. 3. 2018]. Available from: https://www.gartner.com/doc/reprints?id=1-4J4JD9Z&ct=171024&st=sb

[16] PROCESSMAKER. ProcessMaker - BPM & Workflow Software Solution. [online] [accessed: 26. 3. 2018]. Available from: https://www.processmaker.com/

[17] BIZAGI. Bizagi Modeler - BPM Software. [online] [accessed: 26. 3. 2018]. Available from: https://www.bizagi.com/en/products/bpm-suite/modeler

[18] BIZAGI. Bizagi Modeler User Guide. 2018, [online] [accessed: 23. 3. 2018]. Navigation: Simulation - Simulation in Bizagi - Simulation levels. Available from: http://help.bizagi.com/process-modeler/en/

[19] PRAKASH, V. 3 Point Estimate: Triangular Distribution vs Beta Distribution (PERT). [online] [accessed: 28. 4. 2018]. Available from: https://www.pmchamp.com/3-point-estimate-triangular-distribution-vs-beta-distribution-pert/

[20] MICROSOFT. What is .NET. [online] [accessed: 2. 5. 2018]. Available from: https://www.microsoft.com/net/learn/what-is-dotnet

[21] LANDWERTH, I. Introducing .NET Standard. Sep 2016, [online] [accessed: 2. 5. 2018]. Available from: https://blogs.msdn.microsoft.com/dotnet/2016/09/26/introducing-net-standard/

[22] BURNS, e. a., Ammy. Understanding the Xamarin Mobile Platform. Mar 2017, [online] [accessed: 2. 5. 2018]. Available from: https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform

[23] CHAUHAN, S. Understanding Xamarin Forms. Mar 2017, [online] [accessed: 2. 5. 2018]. Available from: https://www.dotnettricks.com/learn/xamarin/xamarin-forms-build-cross-platform-mobile-apps

[24] FLETCHER, e. a., Patrick. Understanding the Xamarin Mobile Platform. Oct 2017, [online] [accessed: 2. 5. 2018]. Available from: https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

[25] LEIBOWITZ, M. SkiaSharp - The Skia 2D Graphics library. [online] [accessed: 2. 5. 2018]. Available from: https://github.com/mono/SkiaSharp

[26] SYNCFUSION. Syncfusion - .NET Components & Controls. [online] [accessed: 3. 5. 2018]. Available from: https://www.syncfusion.com/

Appendix A

Acronyms

BI Business Intelligence.

BPM Business Process Management.

BPMN Business Process Model and Notation.

BPMS Business Process Model Systems.

CTU Czech Technical University.

DEMO Design and Engineering Methodology for Organizations.

EO Enterprise Ontology.

KPI Key Performance Indicator.

OCD Organization Construction Diagram.

PCL Portable Class Library.

PSD Process Structure Diagram.

RAC Rent-A-Car.

Appendix B

Content of Enclosed CD

- `readme.txt` ..... Brief CD content description
- `src`
  - `impl` ..... Implementation source code
  - `thesis` ..... LaTeX source code of text
- `text`
  - `thesis.pdf` ..... Thesis text in PDF form