Designing a standard architecture for service management based on the BASE/X framework

Traganos, K.

Award date:
2014

Technische Universiteit Eindhoven University of Technology

Department of Industrial Engineering and Innovation Sciences
Department of Mathematics and Computer Science

Designing a Standard Architecture for Service Management based on the BASE/X framework

Konstantinos Traganos

In partial fulfilment of the requirements for the degree of
Master of Science in Business Information Systems

Supervisors: prof.dr.ir. Paul Grefen (TU/e)
dr. Dirk Fahland (TU/e)
Eric van der Linden MBA (DLL)
Tutor: Robert Diependaal (DLL)

Eindhoven, September 2014

Abstract

Many business domains are currently transitioning towards a service-oriented business setting. Before the transition, the business value used to be in owing assets while in the new setting the business value is in using the services offered by these assets. This creates service-dominant business markets requiring high levels of agility. Providers in these markets find their agility, however, heavily constrained by the business and IT platforms they use to deliver their services. Thus, there is a need for a structured way to business design in a service-dominant context.

De Lage Landen International B.V.[1], operating in the service sector, is heading towards this trend and in order to achieve that transition they helped Eindhoven University of Technology in the development of a framework, called BASE/X[2], which will support the efforts of dealing with the agility of service-dominant business, dealing with the complexity of solution-oriented business (value-in-use) and dealing with multi-sided business models. BASE/X is based on a structure of stable and flexible layers composing a pyramid (from top to bottom the layers are: Strategy, Business Model, Service Compositions and Business Services). During past projects between De Lage Landen and TU/e, most of the concepts of BASE/X have been applied in practice. What is still needed is to put more focus on the Service Composition layer, which is the operationalization of the Business Models layer through composition of Business Services.

The business concepts of each layer of BASE/X require support in the form of business applications which in turn need support by underlying platforms. The current project is focusing on the realization of the support of the Service Composition layer with the help of the Business Process Management (BPM) discipline. With architectural frameworks and through a number of design steps, it proposes an enterprise standard architecture for De Lage Landen which will be a blueprint for the organization of automated systems and processes, helping in the composition of new services and consequently supporting the agility in service-dominant business. Service Oriented Architecture (SOA) principles are applied in the design of the standard architecture, aiming at the "servitization" of rigid legacy systems.

A prototype of service management in a Business Process Management System is also implemented to act as a Proof of Concept (PoC) that the realization of service compositions is feasible.

Keywords: BASE/X framework, service-dominance, agility, service compositions, standard architecture, business process management.

[1] De Lage Landen is a global provider of leasing, business and consumer finance solutions, including vendor finance and factoring. http://www.delagelanden.com/

[2] BASE/X is the acronym for Business Agility through Service Engineering in a Cross-Organizational Setting.

Acknowledgements

This thesis document is the result of my graduation project which completes my Master's degree in Business Information Systems at Eindhoven University of Technology (TU/e). The project was performed at the Information Systems (IS) research group of the Industrial Engineering and Innovation Sciences (IE&IS) Department of TU/e and was carried out at De Lage Landen International B.V. (DLL), a global provider of leasing and finance solutions. Throughout this project I met and collaborated with many professionals from the university and the company, all of them deserving my appreciation for the help they provided me.

First of all, I would like to thank my main thesis supervisor, professor Paul Grefen, for introducing me to this interesting research project and providing guidance throughout the entire course of this project. He taught me how to think in a structured way, an attribute that every excellent IS architect should poses. He also offered me solutions when things seemed to diverge from initial planning and started to worry me. It was really a pleasure for me to have meetings with Paul and I am grateful to his supervision.

Next, I would like to thank Eric van der Linden, my supervisor from DLL, who gave me the opportunity to conduct such a research project, especially in a large, well-established organization. Eric provided the right support so I could realize my ideas and also inspired me on how to make the project more challenging. Special thanks go to Robert Diependaal, my daily supervisor from DLL for all the efforts he put on that project. Robert spent countless hours sitting with me in order to delineate the project, help me understand concepts and put things "on the road" among all stakeholders. Without his continuous help, this project would not have been completed in the way it did.

A few people involved to the implementation of the prototype, both from Athlon Car Lease and Pegasystems, to all of whom I am grateful. More specific, many thanks to Jeroen Buelens for the input and feedback on setting the requirements of the use case, to Jeroen Koek for working on the integration part of Athlon's systems and to Angelo Jitbahadoer for his feedback on the evaluation part. Special thanks go to Emek Ozer-Gulyaz, Blaise Jamroz, Titto Thottodiparampil and the rest people from Pegasytems who worked on the building of the application and who showed me how to use a powerful tool like Pega 7.

Thanks also go to Dirk Fahland who agreed to participate in the assessment committee and provided important feedback, to Irene Vanderfeesten for the interest she showed on my project and to Egon Lüftenegger for his work that proved valuable for my project. I would also like to thank IS architects from DLL, Hans Tonneijk, Wido van Eersel and John van de Voorde for their helpful insights on how things work in industry.

Special thanks go to my friends and fellow graduates Juby Joseph Ninan, Goppy Gita Gustaman, Ekaterina Sabelnikova and Igor Stojanov who have been there the past two years to support me with my studies and be part of my student life. My thanks go also to Tim Claessens for his work that I stepped on to continue the research on the domain of service dominant business and also to Kritika Maurya for the thoughts we shared on the domain of the project.

Last, but not least, I have to thank my parents and my sister who always show me their unconditional love and support. They always believe in me and give me the freedom to pursue my wishes.

Konstantinos Traganos

Eindhoven, September 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The current project is carried out at De Lage Landen International B.V.[3] (DLL) in the context of the graduation phase of the Master's program Business Information Systems at Eindhoven University of Technology (TU/e). The project is performed at the Information Systems (IS) research group of the Industrial Engineering and Innovation Sciences (IE&IS) Department of TU/e. The IS group, with the help of DLL, is conducting research on how organizations should shift towards a service dominant business environment. This project is a contribution to that direction.

This first chapter introduces the project by explaining the context of the thesis in Section 1.1, stating the problem in Section 1.2 and setting the goals in Section 1.3. The scope of the project is defined in Section 1.4 and finally, the approach and the outline of the thesis are presented in Section 1.5.

1.1 Project context

Many business domains are currently characterized by a move from an asset-orientation to a service-orientation: customers recognize that business value is not in owning assets, but in using the services offered by assets (which they do not need to own), referred to as value-in-use. A good example is the music market, where physical means like CDs and DVDs are replaced by services offering music experience, e.g. Spotify[4]. This move creates service-dominant business markets where services are the basic mechanism for interaction. In [1], the service-dominant business paradigm is defined as the style of defining and implementing business models such that the following three characteristics apply:

1) value-in-use is the main entity that is exchanged (traded),
2) this value-in-use is encapsulated in a set of services, and
3) these services are offered to a market through a service delivery mechanism.

As such, the service-dominant business paradigm is a way of thinking about doing business that uses the service-oriented paradigm as an underlying way of thinking about delivering business value.

The service-dominant markets place high demand on the agility of their providers, who must be able to adapt dynamically to changes. In order to do so, they need systems with high level of automation, and engage in business networks in which parts of offered services can be delivered by business partners.

DLL, operating in the service sector, is following the trend of moving to service-dominant markets and in order to achieve that transition they helped Eindhoven University of

[3] De Lage Landen is a global provider of leasing, business and consumer finance solutions, including vendor finance and factoring. http://www.delagelanden.com/

[4] www.spotify.com

Technology in the development of a framework, called BASE/X [1]. The framework is a structured way to support the efforts of dealing with the agility of service-dominant business, dealing with the complexity of solution-oriented business (value-in-use) and dealing with multi-sided business models.

BASE/X is based on a structure of stable and flexible layers composing the pyramid of Figure 1.

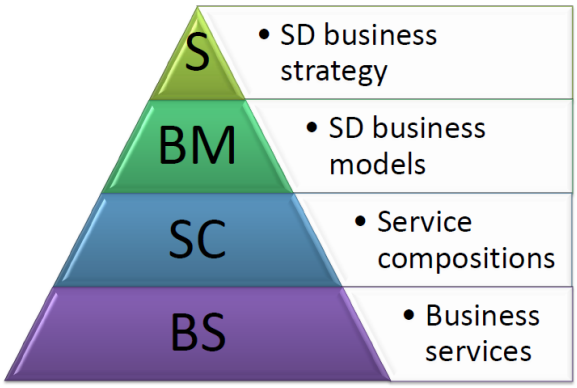


Figure 1 The new pyramid [1]

The business strategy defines the identity of an organization and it is the basis for the definition of its core capabilities in the business services layer. These two are the stable layers that do not change often. The business model layer is the operationalization of the business strategy while the service composition layer is the operationalization of a specific business model by aggregating a set of business services. The two inner layers are the flexible ones since they require adapting to market changes. BASE/X provides a concept model to relate all these major concepts into one model. This can be seen in Figure 2 below.

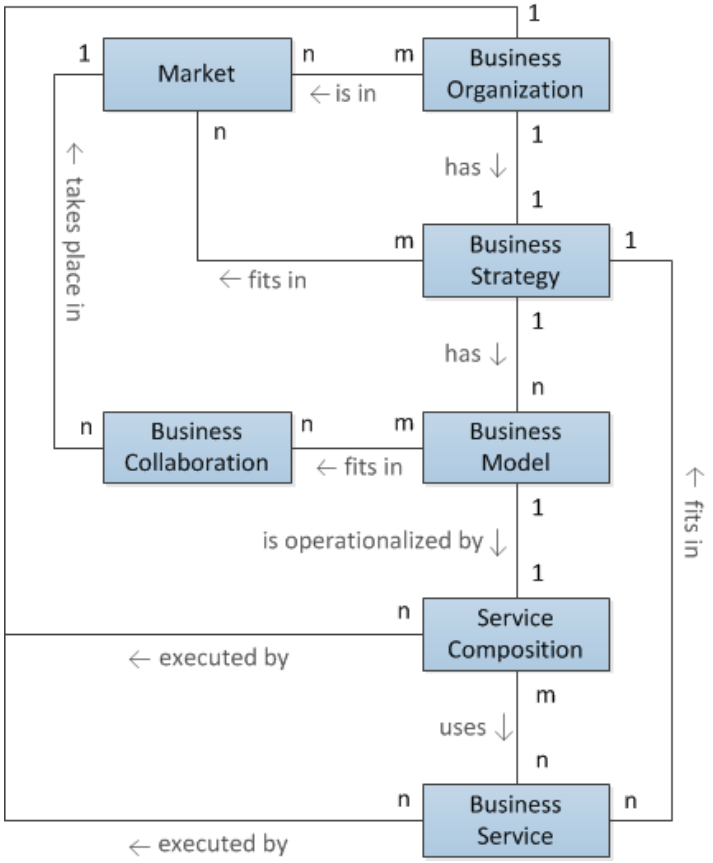


Figure 2 Concept model of the BASE/X framework [1]

During past projects between DLL and TU/e, most of the concepts of BASE/X have been applied in practice. Tools have been designed for defining the business strategy [3], the business services [4] and the business model of the organization [3]. Also, a light prototype has been implemented to deal with the concepts of the service composition layer [5].

However, the prototype of [5] was aiming to create awareness of the flexibility of the service composition layer (and its goal was achieved successfully) but it was not operational since it did not connect any systems of the company. Thus, more focus has to be placed to the realization of the support of the service composition layer.

1.2 Problem Description

As discussed in the previous section, organizations need to be agile to cope with dynamism in service-dominant business markets. However, service providers find their agility heavily constrained by the limitations of the IT platforms they use to deliver their services, i.e., the general-purpose systems that support specific applications. Usually, current legacy systems are characterized as monolithic in which the elements are tightly coupled to each other. All functionality is included in one system and there is a complete absence of explicit structure and modularity. Thus, we cannot exploit the services offered by different modules. On the other hand, the service-dominant paradigm requires a well-defined structure of systems and

interoperability of them. Therefore, there is a need to decouple components used to provide services in order to flexibly communicate to each other and to systems of external partners as well. In that way, organizations will be able to compose services across their systems and services provided by collaborative external parties into new service compositions, introducing new business offerings to the market.

The BASE/X framework does not look only at the business side of the service-dominance paradigm, but provides also support in the form of organizational processes and automated systems. The pyramid of Figure 1 is called Business pyramid which needs support in the form of business applications. That leads to the Information Systems pyramid which in turn needs support by underlying platforms. Consequently, we have a three-pyramid model, as shown in Figure 3, covering both the business and the IT side of the service-dominant business.

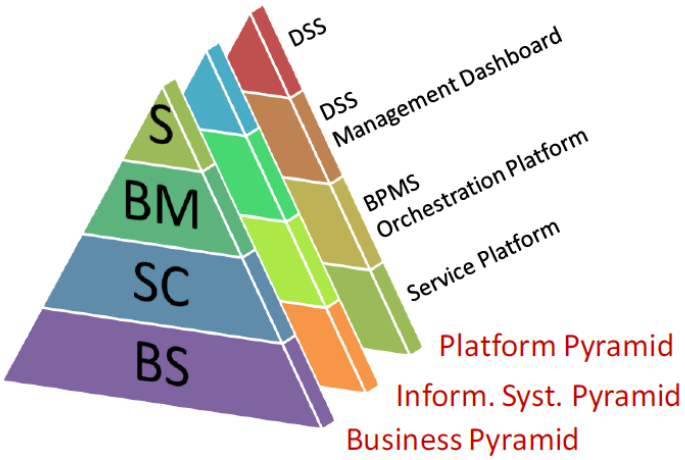


Figure 3 Three-pyramid model [1]

Going back to the problem, we see that the approach of the BASE/X framework of dealing with business agility is to support the service composition with the help of the Business Process Management (BPM) discipline (in Chapter 2 we discuss in more details about BPM). However, this theory needs to be applied in practice. The realization of the service composition layer is the main issue of this thesis and we can summarize it in the following statement:

Problem statement: *How to form the realization of service compositions?*

1.3 Project Goals

From the problem description discussed in the previous section, we realize that current systems are not able to support the service dominant paradigm. There is a high need to re-design the current IT and Application architecture of an organization in a way that will support the creation of service compositions. BASE/X offers a structured way to do this by providing a reference architecture for agile, service-dominant business. Based on that, we will provide more technology and organization-specific details in order to show how re-design can be implemented in an organization (Distinction of reference and standard

architectures is explained clearly in Section 3.1.2). Thus, a main goal of this project is to design an enterprise standard architecture for DLL which will be a blueprint for the organization of automated systems and processes that will support the agility in service-dominant business.

Speaking in terms of the BASE/X framework, taking as input context elements from the Business Model and Service Composition layers from the Business pyramid, we focus on designing the architecture of the second and third pyramid in the Service Composition Layer, as we can see in Figure 4.

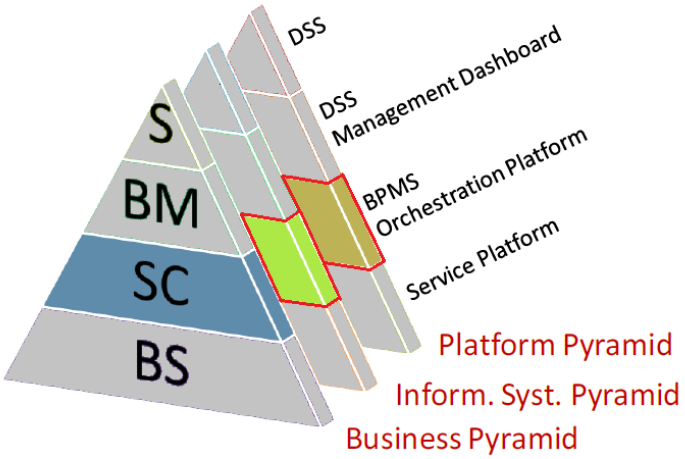


Figure 4 Support of Service Composition layer

In order to design that architecture, we have to consider three different aspects (based on what we have as input elements and what we aim to achieve):

- Current architecture of the company (IT and Application architecture)
- Business Process Management (BPM)
- Service-dominant business paradigm

After the design of the architecture, by developing a prototype in a BPM platform in a service-dominant way, we will map the current architecture onto the Architecture/Platform pyramid of BASE/X. In that way, we will be able to support the Service Composition layer. The main goal can be summarized as follows:

Goal of the project: *Design of an enterprise standard architecture for service management in order to show how the service composition should be realized.*

The working prototype will act as a Proof of Concept (PoC) that the realization of service compositions is feasible.

1.3.1 Sub-goals

Through the process of designing the architecture and creating the prototype several findings can be interesting, thus the project has a number of sub-goals. More specifically, it is important to:

- Discover how difficult it is to make current systems ready for the service-dominance paradigm through exploring IS and the services they can provide.
- Assess whether the choice made about supporting the service composition layer in terms of a BPM platform are suitable or not for our needs and consequently be the choice for future, actual implementations.

1.4 Scope

Since the current application landscape of the organization is large, we focus on a specific business domain, i.e. the car leasing solutions that Athlon Car Lease International B.V.[5] offers. This is a domain which provides many opportunities for business agility and during past projects many concepts of BASE/X have been tested already. By taking as input a part of the application landscape of Athlon, we will provide service management with a BPM tool in order to automate the related processes. In that way we will show how the redesign of other business lines of DLL should be done.

The operational ambition level of the project is a prototype of the realization of the Service Composition layer in a BPM tool. By designing representative business processes composed of a few business services from the business line we mentioned above, we will execute them in BPM tool in order to test the concepts handled in the current project.

Given the fact that it is a 6-months project, the technical scope of the prototype will be limited to the connection of a few existing internal and possible external services of the legacy systems. Services that are required for the prototyping purposes but are not currently used, will be implemented as mock-up services.

1.5 Approach and Outline

In order to achieve our goals, we divide the project in three main phases, namely:
- Preparatory phase (Phase 1),
- Design phase (Phase 2),
- Prototyping phase (Phase 3)

The report starts with the current introductory chapter that provides a description of the problem handled in this project, its formulation and goals, the scope and the approach we follow.

Phase 1 is the analysis of the three aspects of the target architecture as mentioned in Section 1.3. Thus, it can be further split in Phase 1A, where we extract the relevant part of the current architecture, and Phase 1B where we explore the BPM system that we need and the Service Dominance specifications. In that second sub-phase, we also specify the service compositions that will be used as an input to the prototype.

[5] Athlon Car Lease International B.V. is a leading European car lease company and forms part of De Lage Landen International. http://www.athloncarlease.com/

In Chapter 2 we present a market analysis of Business Process Management tools and the final decision of the tool we use.

Chapter 3 presents the results of Phase 1A. We first introduce the theory behind architecture of information systems. Then, given an overview of the Application landscape, we chart the relevant part required for the specific business model and corresponding business services. We specify the elements that compose the architecture and the interfaces needed for their relationships.

In Chapter 4 we discuss how can legacy systems be converted in order to support the service-dominant paradigm.

In Chapter 5 we present the specifications of the business processes that are implemented in the prototype. This is done after analysing the business model and the service repository.

Chapters 2, 4 and 5 are the results of Phase 1B.

In Phase 2, we integrate Phase 1A and 1B. Having analysed the business model and the corresponding services, having specified which building elements are included and how they are connected to each other and having chosen the BPM platform, we design the blueprint architecture for the Service Composition layer. This is presented in Chapter 6.

Phase 3 is split in two sub-phases, where in Phase 3A we conform the design of the architecture by developing a prototype of the information system. In the chosen BPM platform, we prototype a few business processes in order to visualize the service management and we connect the modules of the architecture in an operational way. In Phase 3B, we evaluate the prototype and assess the selected tool. The results of those sub-phases are presented in Chapter 7.

The thesis is concluded by Chapter 8 in which we assess the outcomes of the project and whether the goals are achieved or not. We also discuss about limitations faced and future work that needs to be done for improvements of our findings. Finally, a personal reflection is presented to serve as a guideline for future projects.

The outline of the document is presented in Figure 5 below.

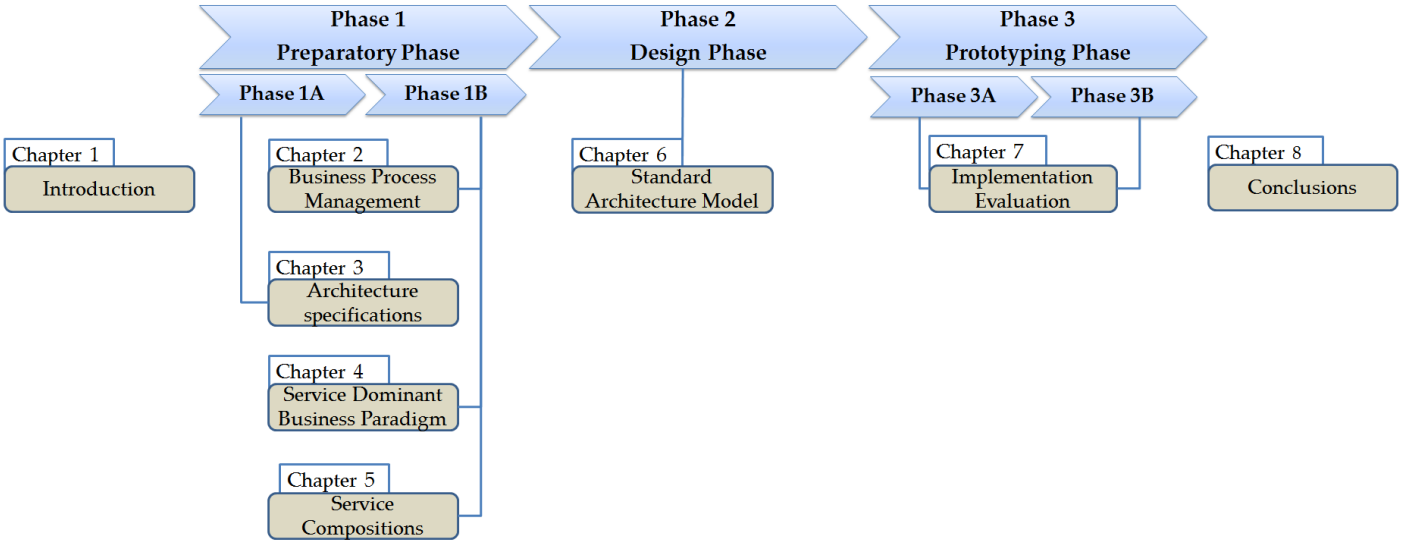


Figure 5 Thesis Outline

A detailed project plan with the deliverables of each phase is presented in Appendix A.

Chapter 2

Business Process Management

Business Process Management (BPM) is a field continuously growing, starting a few decades ago as a result of people's and organizations' efforts to redesign/reengineer boundary-crossing processes with the use of IT, aiming to improve customer services. Over these years, many definitions have been given for BPM. Examples are: "A customer-focused approach to the systematic management, measurement and improvement of all company processes through cross-functional teamwork and employee empowerment" [6], "Supporting business processes using methods, techniques, and software to design, enact, control, and analyse operational processes involving humans, organizations, applications, documents and other sources of information" [7].

From all these definitions, key observations can be made concluding in the following:
BPM can be seen as a discipline that intersects knowledge from management and information technology and applies this to operational business processes. It covers all phases of these processes, from identification to discovery, diagnosis, planning, design, deployment, execution and control. Applying BPM in practice leads to consistency, lower operating costs, faster processes, enhanced flexibility and improved customer satisfaction translating into improved enterprise performance [8].

In our project, BPM plays the role of the realization of the Service Composition layer of the Business Pyramid of the BASE/X framework [1]. This is done by implementing a Business Process Management System (BPMS) which is an information system that coordinates automated business processes in such a way that all work is done at the right time by the right resource [9].

This chapter deals with the process of selecting a BPM tool in which we are implementing our prototype. First, in Section 2.1 we provide the necessary requirements for an appropriate tool. Then in Section 2.2, we present a shortlist of available tools in the market. We evaluate these tools in Section 2.3 and finally, in Section 2.4, we conclude with the final decision.

2.1 BPM Tool Requirements

In order to select a BPM tool that will help us in the orchestration and automation of business processes, we first need to understand what we want to achieve with it and what it can offer to us. In that way we elicit a list of high level requirements and then we check a shortlist of tools that can meet those.

Our aim is to build executable "end-to-end" business processes combining application integration and human workflow. This should be achieved taking into account existing systems that we have to couple the BPM platform with. That transition of the current situation to the new situation after the redesign of the architecture will also affect the human

interaction since people already use various user interfaces and systems to perform their work. Another goal is the linking of services from disparate systems of Athlon regardless location, platforms and communication protocols. Furthermore, we care about the management of processes by viewing business metrics in real time and optimizing when deviations from targets appear.

Based on these goals, we view BPM from a few different aspects:

- Platform/Software aspects
- Integration/Compatibility with legacy systems
- Features and actual development
- Connectivity among services
- Human interaction
- Other requirements/restrictions

For each of these aspects, we provide a list of functional and non-functional requirements (we indicate them with an F/NF in parenthesis. The distinction is based on [10]).

Platform/Software aspect:

- Supported Platforms (F)
- Supported DBMS (F)
- On premises/Cloud servers (F)
- Security/Authorisation solutions (F)
- Robustness (NF)
- Runtime scalability (NF)

Integration/Compatibility with legacy systems:

- Supported middleware layers (F)

Features and actual development:

- Ability to create a prototype/Test environment (F)
- Model-driven composition environment (F)
- Supports administration/roles management (F)
- Supports business rules (F)
- Import process descriptions from other tools (e.g. ARIS, MS Visio) (F)
- Supports Dynamic Case Management (F)
- Well-documented/Support/Training (NF)
- Minimizing need of hard coding (NF)

Connectivity among services:

- Supports web services (F)
- Supports SOA/Interoperability (NF)

Human interaction:

- Comprehensibility and easiness of learning for both developers and end users (NF)
- Possible integration with UIs of other systems (F)
- Well-managed forms (NF)
- Supports external web applications as user interaction (F)

Other requirements/restrictions:

- Cost/Licensing (NF)
- Simulation capabilities (F)
- Business activity monitoring (F)
- Supports mobile development (F)

The next step is to give weights to these requirements based on priorities of our final goals. Having in mind that our goal is the interoperability of services, in a two-point scale, we give weight two to these requirements referred to connectivity. Also, we weight heavier the integration of UIs since there is a need for exploiting current systems that used for user input. The rest requirements have a trivial weight of 1. The weights are used in the evaluation step in Section 2.3.

2.2 Shortlist Tools

Nowadays, there is a plethora of vendors offering BPM tools, most of which promising to provide solutions that will help enterprises to achieve their goals. Since it is impossible to assess a large list of those tools[6], we focus on a shortlist, consisting of Pega BPM[7] (Pegasystems), Bizagi BPM Suite[8] (Bizagi) and IBM BPM[9] (IBM). These are selected based on suggestions of IT staff from DLL, professors from TU/e and knowledge available in the web ([11], [12], [13]). Here we shortly present these three tools.

Pega BPM

In today's BPM technology market Pegasystems is the big independent influencer pushing things forward [14]. Its primary strength is its unified object architecture, which structures all process artifacts, including processes, policies and user interfaces. This architecture enables Pegasystems to deliver a declarative modeling composition environment that improves the ability of the BPMS to model, change and adapt to new business needs [11]. There's strong support at both design time and runtime for automated work scenarios. Pega BPM promises to simplify and automate enterprise's operations so that they can reduce costs and improve business agility.

On the other hand, Pegasystems is growing so fast that it will have to work hard to build enough skills to match and exceed its growth rate. Its declarative composition environment represents a paradigm shift for most prospects, and will require training to take advantage of the closed environment. Also, Pegasystems' prices are high relative to others, reflecting the value it believes it delivers [12].

Bizagi BPM Suite

Originally focused on the Latin American market, Bizagi has made significant progress expanding its client base in North American and European markets. Bizagi delivers a very strong, BPMN-based, model-driven composition environment [11]. A key differentiator for

[6] http://www.businessprocessincubator.com/tools/bpms.html, retrieved September 10th 2014.
[7] http://www.pega.com/products/business-process-management
[8] http://www.bizagi.com/en/products/bizagi-bpm-suite/overview-bpm-suite
[9] http://www-03.ibm.com/software/products/en/category/BPM-SOFTWARE

Bizagi is its unique data modeling approach that allows process designers to create a virtualized data model that provides a consistent way to manage and synchronize between business process models and source data and legacy applications [13].

However, Bizagi is functionally less complete compared to other leading pure-play BPMS tools (e.g. rules and simulation/optimization) [11]. Also, users report various weak areas, including the forms designer, the integration layer and insufficient documentation.

IBM BPM

IBM is also a leader in the BPM market. It consistently extends beyond its integration-centric heritage to produce flexible tools that are easy to use, yet able to cope with very significant design-time and operational scale. With its latest release IBM has focused particularly on enriching the transactional work capabilities of Business Process Manager – providing a stronger and more flexible underpinning for human-centric workflow applications at design time, runtime and also from an optimisation perspective [14]. Business Process Manager mostly uses a direct, model-driven architecture that delivers one of the most integrated design, simulation, analysis and testing experiences available in the market. Authors can collaboratively visualize trouble spots, explore optimization options through prototyping and integrated heat maps and use visual testing to aid continuous process improvement [12].

On the downside, although IBM has made a lot of progress unifying the architecture within IBM BPM as it merged the BPM specialist Lombardi Software acquisition (in 2010) with existing technologies, there is still debugging/troubleshooting that requires administrators to check multiple log files and correlate entries manually [12] .

2.3 Tools Assessment

Through vendors' corporate sites and documentations on their tools and after searching relevant information in the web (assessment reports; blogs; BPM groups/forums), we map in Table 1 below each tool's features with the list of our requirements. Where possible, we use a three-point evaluation scale, indicating complete (+), partial (+/–), or lack (–) of support for the corresponding requirement. The others are textual information related to the requirement.

Table 1 BPM Tools assessment

BPM aspect	Requirement	Weight	Pega BPM	Bizagi	IBM BPM
Platform/Software	Supported Platforms	1	J2EE	JEE, .NET	Java, .NET
	Supported DBMS	1	Oracle/SQL/ DB2	Oracle/ SQL	Oracle/SQL
	On premises/Cloud servers	1	Both[10]	Both	Both
	Security/Authorisation solutions	1	+	+	+
	Robustness	1	+	+/-[11]	+

[10] The Pega Business Cloud is not just for process design, but can also handle process execution.

	Runtime scalability	1	+	+/-	+
Integration/ Compatibility with legacy systems	Supported middleware layers	2	Service and connector types (SOAP, JMS, etc.)	Component Library (API, connectors)	WebSphere Enterprise Service Bus/ WebSphere Message Broker
Features and actual development	Ability to create a prototype/Test environment	2	+	+	+
	Model-driven composition environment	1	+/-[12]	+	+
	Supports administration/roles management	1	+	+	+
	Supports business rules	1	+	+	+
	Import process descriptions from other tools (e.g. ARIS, MS Visio)	2	+/-[13]	+/-	+
	Supports Dynamic Case Management	2	+	-	+/-[14]
	Well-documented/Support/Training	2	+	+/-	+
	Minimizing need of hard coding	2	+/-[15]	+	+/-
Connectivity among services	Supports web services	2	+	+	+/-
	Supports SOA/Interoperability	2	+	+	+/-[16]
Human interaction	Comprehensibility and easiness of learning	1	+/-	+	-
	Possible integration with UIs of other systems	2	+	+/-	+
	Well-managed forms	1	+	+/-	+
	Supports external web applications as user interaction	1	+	+/-	+

[11] Bizagi Xpress Edition does not offer options for fault tolerance and clustering. Only in Enterprise editions they are possible.

[12] Pega BPM offers business-oriented, tactical process optimization solutions where business processes are heavily rules-driven.

[13] XPDL and Visio files can be imported. However, because Pega embeds so much of the decision management and rules capabilities within the process model, these ports are of limited use since they only cover the actual model itself and not all the underlying features and definitions [15].

[14] Based on analysis on Dynamic Case Management made by the Rabobank Group, IBM performed not so well in that area. The final assessment report can be obtained upon request to DLL.

[15] Integration to backend services, databases and applications is cumbersome in PRPC, frequently requiring hand coding in Java and Pegasystems consultants to assist on projects [16].

[16] IBM Websphere ESB is included for BPM but is restricted and not available for general use unless purchased separately (only in Advanced edition).

Other requirements /restrictions	Cost/Licensing	1	100,000€[17]	100€/user[18]	Unknown
	Simulation capabilities	1	+/-[19]	+/-	+
	Business activity monitoring	1	+	+	+
	Supports mobile development	1	+	+	+
		Total	21	13	18

The final rating is extracted by adding/deducting the "+"s/"-"s of each tool (each "+"/"-" is first multiplied with the corresponding weight).

Let us note here that we used only product's criteria in our assessment. In order to evaluate these tools we should also consider other aspects that are not presented in the list of requirements. These are vendor's viability and market share of the product, already successful implementations and lessons learned from possible failed projects in other companies. Also, we could have performed a profitability/ROI analysis and calculated the time-to-value (the time from project start to the start of business benefit accrual). These factors were out of scope and were not analysed in the above assessment.

2.4 Final Verdict

Before proceeding to the BPM platform selection, the final step should be asking all the assessed vendors for a product demonstration. By presenting a demo with the tool's capabilities, we would have a better insight. However, due to time constraints we did not proceed to that step. The final choice could have be done based on the assessment of the previous section, where we see that Pega BPM scores a value of 21, while Bizagi has a value of 13 and IBM BPM gets a value of 18. Taking into account though, the decision made by Rabobank Group[20] to pick PegaSystems for their needs (especially for the Dynamic Case Management capabilities that provide extra flexibility as we explain in Section 7.1), we adopt that choice and Pega tooling is our final choice.

[17] Pegasystems licenses its product on a usage basis with a license fee for each named user, rule execution and connection to an external system. The above price is for a small-medium process scenario [16].

[18] Price for Bizagi Xpress Edition, perpetual license. Maintenance costs are 20 €.

[19] Simulation is available both at the skeleton process level and at whole process level. However this support is fairly basic [15].

[20] DLL is a fully-owned subsidiary of the Rabobank Group.

Chapter 3

Architecture Specifications

In Chapter 2 we discussed about the BPM aspect of the project which plays the role of service management. In this chapter, we discuss the architecture viewpoint of the project by first introducing a few concepts of architecture of information systems in Section 3.1. In Section 3.2, we draw a number of abstract architecture models in order to understand the concepts of orchestration of business processes viewed from different levels. Then, we apply the theoretical concepts in our case by charting the current Application Landscape of Athlon Car Lease and scoping to relevant elements in Section 3.3 and finally making the specifications of the simplified architecture in Section 3.4 in order to act as the basis for the design phase of the standard architecture model.

3.1 Concepts of Architecture of Information Systems

The concept of architecture has emerged in the information systems field during the past decades to deal with their growing complexity. The use of architecture aims at bringing structure into the design of complex business information systems [2].

The IEEE defines architecture in its 1471 standard [17] as follows:

Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.

The relationships between the components and to the environment describe how the functional units interact to achieve the overall behaviour of the system.

As we can see from the above definition, architecture refers on the one hand to the parts that compose an information system and their interactions as well and on the other hand to a set of guidelines, modeling methods and approaches to actually design it. These two viewpoints of the definition help us in our approach to design the desired standard architecture.

3.1.1 Architecture Analysis Frameworks

Before analysing the elements of an architecture, we need to understand that architectures of large information systems are large and complex themselves. It is necessary therefore to look at the architecture in a well-structured framework. In our project, we make use of the framework presented in [2] which relies on a set of four architecture dimensions. These are:

- The **aspect** dimension, which describes a number of aspects from which we can view an architecture, providing a certain 'cross-section' of an entire architecture description.

- The **aggregation** dimension, describing the levels of aggregation which determine the level of detail of an architecture description – ranging from very coarse (few elements) to very detailed (many elements).
- The **abstraction** dimension, which describes the abstraction levels at which we describe an architecture, ranging from very abstract (i.e., few concrete choices have been made) to very concrete (i.e., all concrete choices have been made).
- The **realization** dimension, describing the spectrum from very business-oriented descriptions (no attention for IT elements) to very IT-oriented descriptions (no attention for business elements).

An **aspect** of an information system architecture is a specific way to look at that architecture by focusing on specific characteristics of that architecture only. A single architecture can be looked at from different aspects. Frameworks have been developed to distinguish among important aspects. Two of them are the Truijens 5 aspect framework [18] and the Kruchten 4+1 aspect framework [19]. Here, we use a modernized version of Truijens framework consisting of the following aspects [2]:

- data
- process
- software
- platform
- organization

Some of these aspects are taken into account later when specifying Athlon's architecture.

In the **aggregation** dimension, we determine how detailed an architecture model is with respect to the number of components identified. By using several levels of aggregation, we explode components at a higher level of aggregation into their components at a lower level of aggregation.

In the **abstraction** dimension, we determine how abstract or concrete an architecture description needs to be. Completely abstract means that no concrete choices have been made with respect to components (functional software blocks) in the architecture – these elements are described in very general terms. Completely concrete means that the components are described in very precise terms. We use different values in the analysis of our architecture.

The **realization** dimension defines a number of levels ranging from architecture goals (i.e., what we want to accomplish with the systems described by an architecture) to means (i.e., how we want to accomplish things with an architecture). The dimension can be interpreted as ranging from very business-oriented to very technology-oriented.

Each architecture model has a specific value with respect to the aggregation, abstraction and realization dimensions and covers one or more specific aspects. Thus, we can say that each architecture model can be placed at a specific point in a design space consisting of these four dimensions. When performing an architecture design process, we have to traverse the design space. We start at an abstract, highly aggregated, business-oriented architecture specification. In a number of design steps, we need to arrive at a concrete, detailed, IT-oriented specification. To visualize this process, we make use of a three-dimensional design "cube", as shown in Figure 6 where each cell represents a specific combination of values

along the aggregation, abstraction and realization dimension. Each cell of the cube contains a number of architecture models. These models in one cell deal with all relevant aspect dimension values for that cell. In other words: the aspect dimension is 'hidden' in the cells described by the other three dimensions [2].

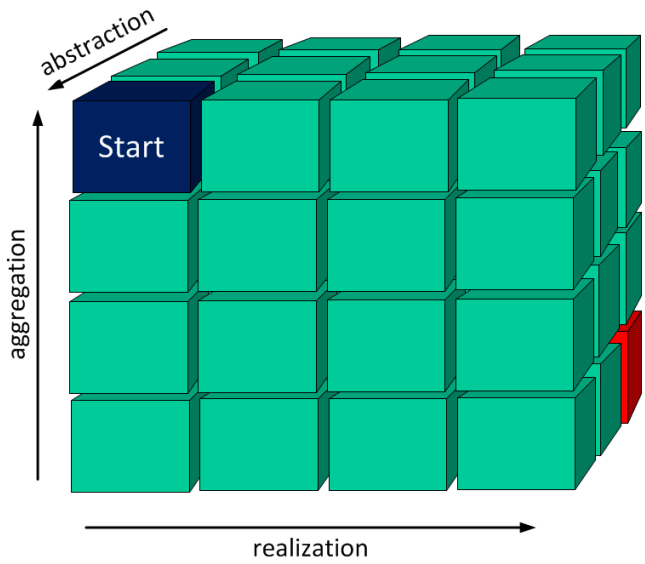


Figure 6 Three-dimensional design "cube", adapted from [2]

Performing a complete architecture design process means going via a number of model transformations from the front top left cell (labelled "Start") to the back bottom right cell.

3.1.2 Reference and Standard Architectures

The aforementioned frameworks allow us to have a "conceptual analysis toolbox" to deal with the complexity of architectural design. Now, we need a "conceptual synthesis toolbox" to actually design a new architecture. This toolbox consists of well-established structures where proven knowledge is reused. An important class of those structures contains Reference Architectures, which are abstract blueprints for information systems architectures.

Grefen in [2] defines Reference Architecture as "a general design (abstract blueprint) of a structure for a specific class of information systems. A Reference Architecture can in general be descriptive or prescriptive". It is used to describe the structure of an information system or a combination of information systems.

An interesting kind of reference architectures are the ones used to connect individual systems, mentioned in [2] as reference architectures for enterprise integration. Their focus in not on the decomposition of specific functionality, but on interoperability issues between systems that provide specific functionality. That is the situation we face in our project, in which we have many systems like ERP, CRM, DMS, etc. that we need to integrate.

While reference architectures provide an "ideal", abstract structure for an information system, we need more concrete details in order to implement it. A standard architecture, as defined in [2], describes a standardized structure for a specific set of architectures (e.g. sites

of a distributed organization) in the context of a specific organization. As such, its design is governed by a single user organization.

Standard architectures are instantiated to concrete architectures in the specific organization context they were designed for. These concrete architectures are typically related to specific IS (re)design projects and can hence be called project architectures.

In the design of standard architectures, reference architectures should ideally be used as a basis. BASE/X framework provides a Reference Architecture for agile, service-dominant business and is the basis of our designed standard architecture. In Figure 7 we present this reference architecture [1].

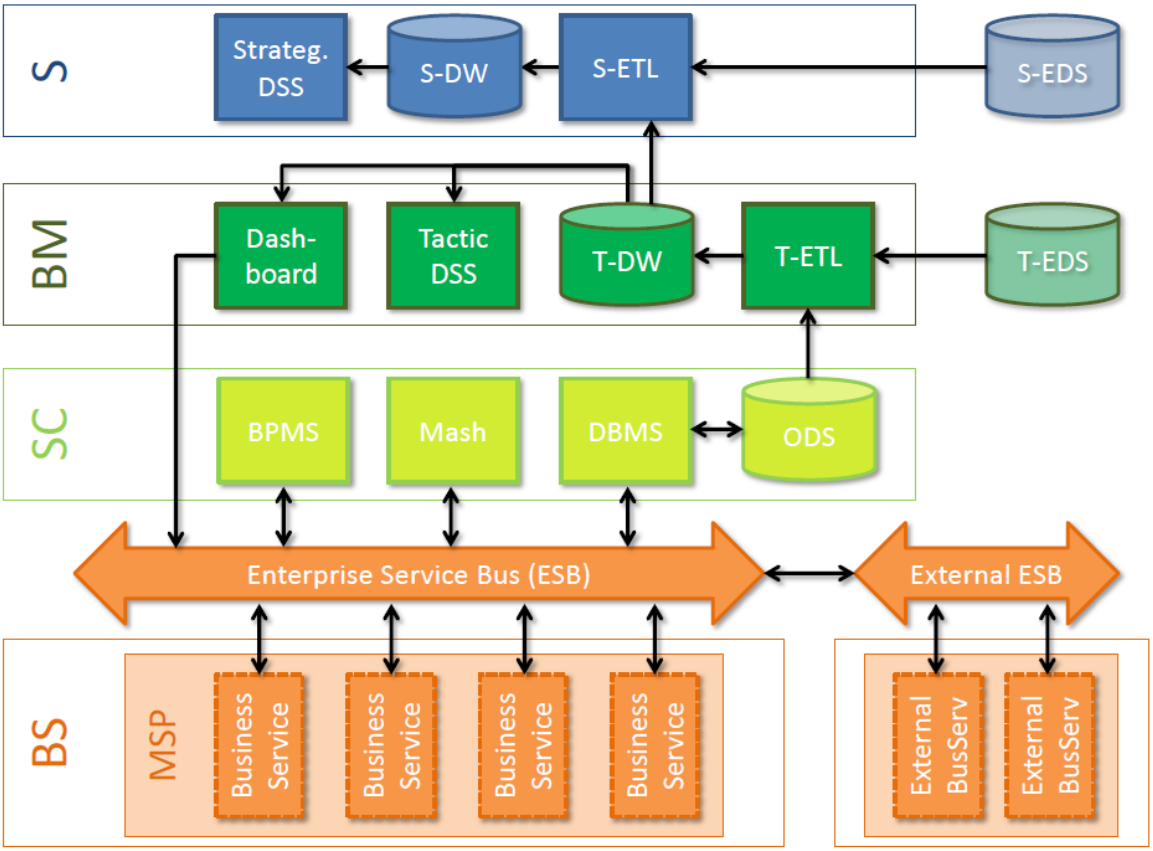


Figure 7 Reference architecture of BASE/X framework for service-dominant business [1]

3.2 Separation of Concerns of Business Process Orchestration

As we have already discussed, the BASE/X framework aims at flexible composition of services in order to deal with the dynamism and complexity of business markets. These compositions, realized as business processes, need the right orchestration. In this section, we illustrate the meaning of orchestration, viewed from different architectural aspects. With the help of the three-dimensional design "cube" we presented in Figure 6, we draw a number of architecture models for realizing the concept of business process orchestration.

We are using two aggregation levels and three realization levels to have a more complete view. We however skip the abstraction dimension of the cube since the purpose of the section is to provide a separation of concerns and there is no need to specify concrete details of the presented models. This is represented by the orange-coloured cells in the "cube" as shown in Figure 8.

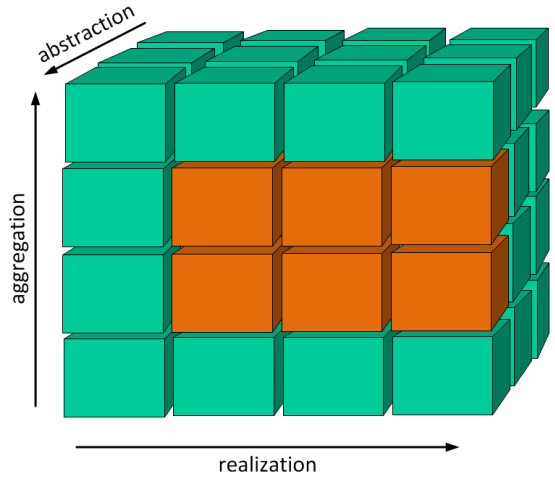


Figure 8 "Cube" for illustration of business process orchestration

In Section 3.2.1, we first discuss the realization dimension. In Section 3.2.2 we explain what is modelled in the aggregation level. In the next subsections, we present the conceptual models for each combination of the aggregation and realization dimensions. Finally, in Section 3.2.8 we combine all models into a single picture by providing an example.

3.2.1 Realization Dimension

The operationalization of the realization dimension is expressed with the use of the BOAT framework as presented in [21]. It distinguishes four levels:

Business (B): the business level describes the business goals of an information system. As such it answers the question why a specific information system exists or should exist or what should be reached. Topics can be leverage of efficiency levels, support new business functions, etcetera. How things are done is not of interest at this level.

Organization (O): the organization level describes how organizations are structured to achieve the goals defined at the B level. Organization structures and business processes are main ingredients here – automated systems are not yet in scope in this level.

Architecture (A): the architecture level covers the conceptual software structure (software architecture) of automated information systems required to make the organizations defined at the O level work. As such, it describes how automated systems support the involved organizations.

Technology (T): the technology level describes the technological realization of the systems of which the architecture is specified at the A level. The T level covers the concrete

ingredients from information and communication technology, possibly including hardware, software, languages and protocols.

Mapping these levels to the design cube, we see in Figure 9 the levels of the realization dimension.

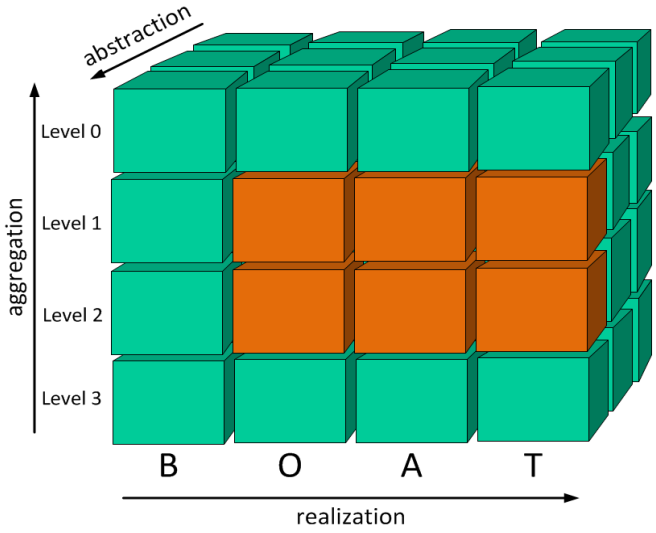


Figure 9 Operationalization of realization dimension

Note that in the current analysis of orchestration we do not cover the B level as this refers mostly to the business strategy and business models which have already been set by the organization and are not under the scope of our project.

3.2.2 Aggregation Dimension

In the aggregation level 1 we speak about business capabilities and business services.

Business capabilities are the core functionality a business organization offer to its (commercial) context. They are derived from the business strategy and are the business competences that allow an organization to differentiate its business and activities.

The term of a business service is used as the encapsulation of an elementary business capability of an organization making it accessible through well-defined, business-level interface [1].

In the aggregation level 2 we decompose business capabilities and business services into corresponding building blocks in order to provide more details (as elements in the upper aggregation level can be seen as black boxes).

3.2.3 O Level – Aggregation Level 1

The first Aggregation level of Organization level relates the Business Capabilities to the Business Propositions. Business capabilities are stable in time as they are based on the

resources of an organization (both human and non-human). In order to provide flexible offerings to its customer, an organization needs to combine these business capabilities into new, agile business propositions. The business propositions demonstrate the business logic of creating value for customers and/or for each party involved through offering products and services that satisfy their needs. On the contrary to business capabilities, these business offerings change over time – they revolve with market dynamics as operationalization of network-centric business models of a provider.

A business proposition combines a number of business capabilities in an abstract business process definition. This is a customer-facing business process meaning that the steps in this process definition have interaction and meaning for the customer. We can say that the process definition is an operational specification of the life cycle of customer experience in terms of interaction of that customer with the service providers.

One business proposition usually involves more than one business capabilities. A business capability may be involved in more than one business proposition, thus we have an n:m relation between these two terms.

We illustrate the aforementioned concepts with the model shown in Figure 10.

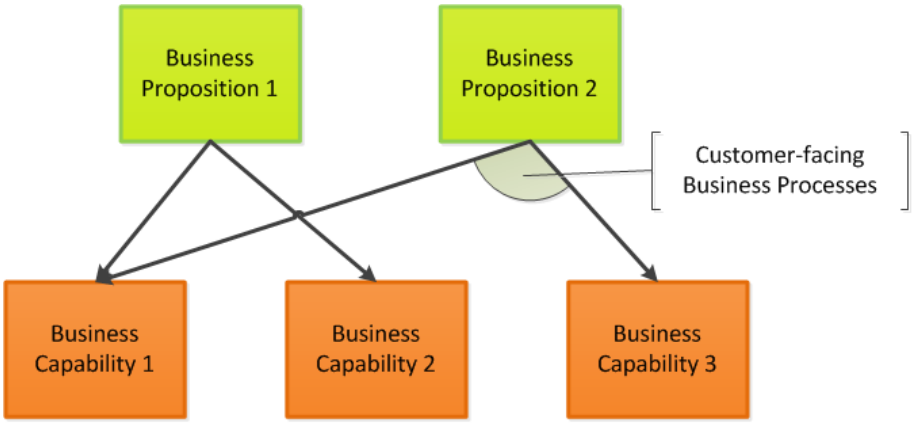


Figure 10 Organization level - Aggregation level 1

One could argue here why we started from the aggregation level 1 and not from level 0, but the reason is because in level 0 we would have the business propositions as black boxes, which are of no interest for the purposes of our project.

3.2.4 O Level – Aggregation Level 2

At the second Aggregation level we decompose the business capabilities into building block capabilities. These modular smaller blocks provide maintainability, standardization and reuse of functionality to the business capabilities.

A business capability combines a number of building block capabilities by means of an internal realization process. This process defines the steps that have to be performed to deliver a (function of a) service, as well as the order in which these steps have to be

performed (the so-called control flow). This internal realization process is completely invisible to the consumer of the capability.

Like in the previous level, there is an n:m relation between the business capabilities and the building blocks capabilities. We represent this in Figure 11 below.

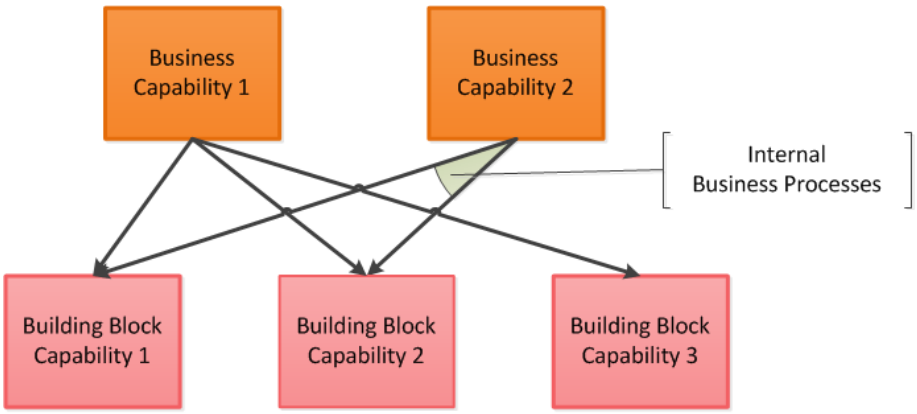


Figure 11 Organization level - Aggregation level 2

3.2.5 A Level – Aggregation Level 1

In the Architecture level we design (conceptually) the software systems that are required to realize the concepts of the Organization level.

Business services need to be composed according to the business process definitions. However, in complex organizations, these business services are provided by a large number of information systems, usually disparate and heterogeneous. Thus, there is a high need of a middleware to allow the interconnection between the services [2]. The management and the automation of the services are done by an Orchestrator element, for example a Business Process Management System as we saw in Chapter 2. A repository completes the picture by storing information about the definitions of the business processes and the states of the process instances.

The above information systems are designed in Figure 12.

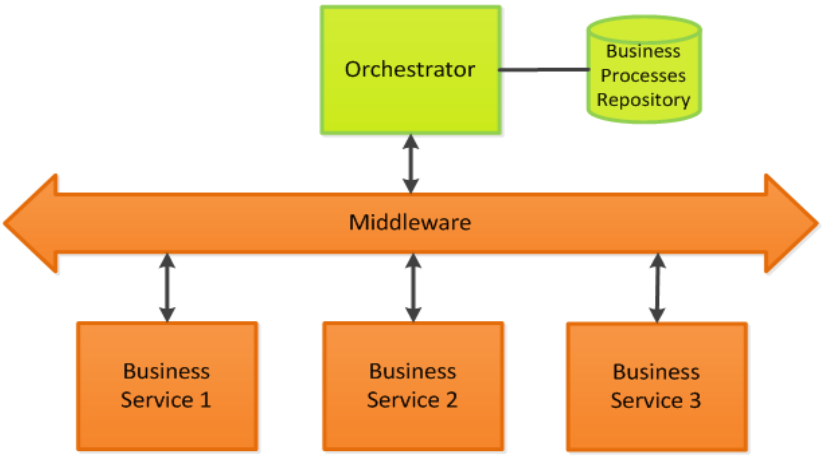


Figure 12 Architecture level - Aggregation level 1

This model is in accordance to the reference architecture of the BASE/X model, as presented in Figure 7.

3.2.6 A Level – Aggregation Level 2

Like we did in the Organization level of BOAT framework with the explosion of business capabilities to building block capabilities, we do the same in the Architecture level by decomposing the business services to building block services. These are internal business services, not visible to a service consumer. Similarly, an Orchestrator information system composes the building blocks with the help of a middleware. A repository with information of the internal business process definitions is needed as well. The model is shown in Figure 13.

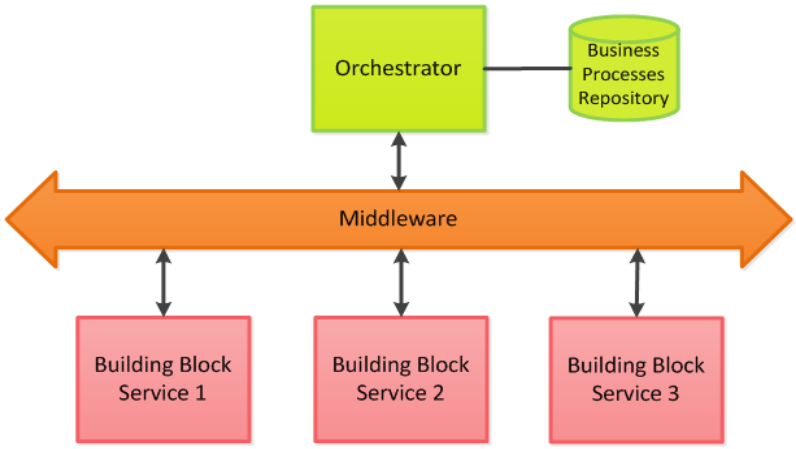


Figure 13 Architecture level - Aggregation level 2

3.2.7 T Level – Aggregation Level 1 and 2

The Technology level of the realization dimension provides the information about the required platform infrastructure for the automation of application presented in the Architecture level.

A Business Process Execution Language (BPEL) Engine, a main component of a Business Process Management System (BPMS), takes the responsibility to create executable process instances and orchestrate the process among participants. Data about the process definitions and executions are stored in a Database Server.

The middleware of the Architecture level model is realized with a service-oriented middleware that supports distributed service invocation for collaboration between services. The most common form of service-oriented middleware is an Enterprise Service Bus (ESB), which is a communications broker that connects services in the context of a corporate information system.

The services are implemented in a managed service platform (MSP) that facilitates development, maintenance and evolution of the services.

In Figure 14 we present the model of the Technology level as described above.

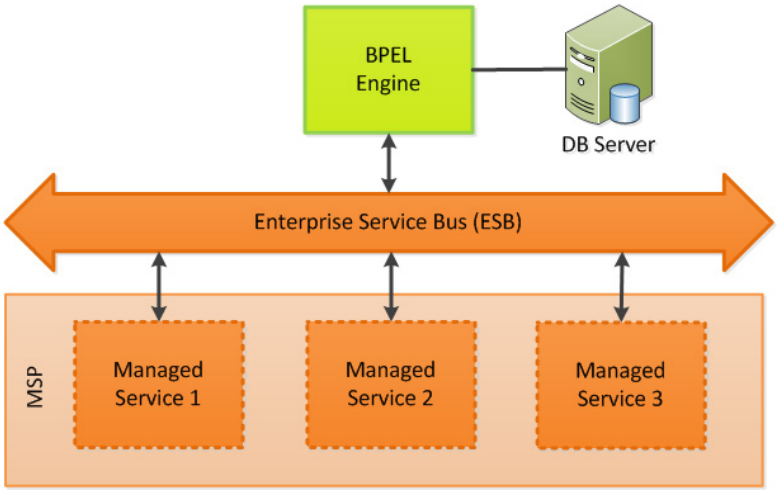


Figure 14 Technology level - Aggregation level 1

From a technology point of view, Aggregation level 2 can be identical to Aggregation level 1. We do not see any differentiation between the way a BPEL engine orchestrates either business services or building block services. Therefore we can say that in the T level of realization, we have a single Aggregation level.

3.2.8 Business Process Orchestration levels

We put all the above five models in Figure 15 by presenting a simple, illustrative though, example based on the content of Athlon Car Lease. For sake of simplicity, we assume that all business capabilities and business services are internal to the organization (offered by the focal organization).

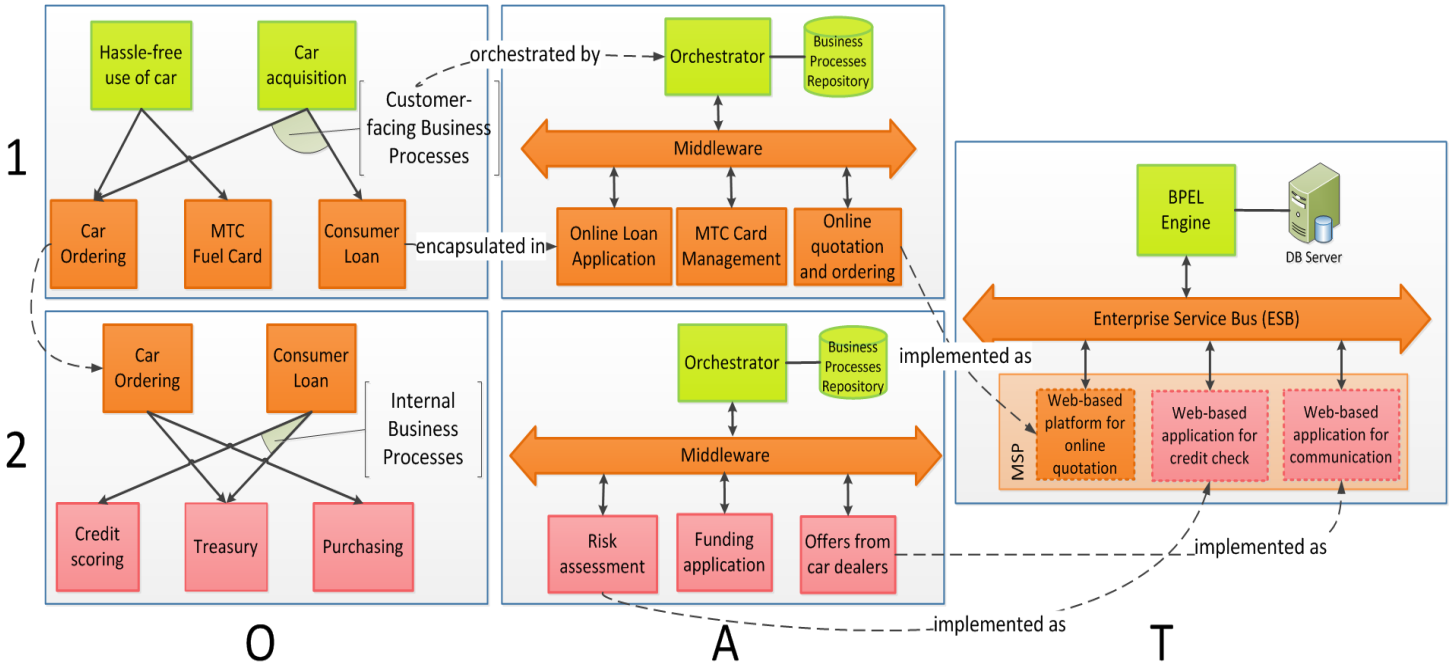


Figure 15 Business Process Orchestration levels

3.3 Application Landscape Charting

Now that we have discussed how to approach the design of an architecture of information systems, we are able to start applying the theory in practice.

In order to design the standard architecture though, we first need to specify which information systems should be included in it and how they are connected to each other. Since architecture models already exist, we first apply a bottom-up approach. That is, given the current application landscape of Athlon Car Lease, we re-design it with essential elements, relevant to the car leasing business model.

The scoping of the application landscape is done based on which applications are useful to us for prototyping compositions of business services. From the entire landscape, where systems are not composed of services and explicit business processes, we keep those core systems that provide functionalities related to main services of the car leasing business model (the business model is analysed in Chapter 5). These processes have already been designed in ARIS tool[21], named as "2BE Processes". From that documentation[22], we elicit the current systems that perform the services. The selection is done having in mind that our aim is the organization of automated systems offered by a BPM platform through coupling existing systems and connecting external ones. Since our scope is a prototype, we include a few elements, enough though to achieve our goals.

[21] http://www.softwareag.com/corporate/products/aris/default.asp, retrieved September 10th 2014.
[22] The link and the credentials to access the ARIS Business Publisher can be obtained upon request to DLL and Athlon Car Lease.

However, the connection with external service partners/providers is limited to two. This is because we do think that this is sufficient for testing our concepts and trying to connect to more external services will not add more value to the project.

Given the current application landscape of Athlon Netherlands, as shown in Figure 16, our focus is on the following systems:

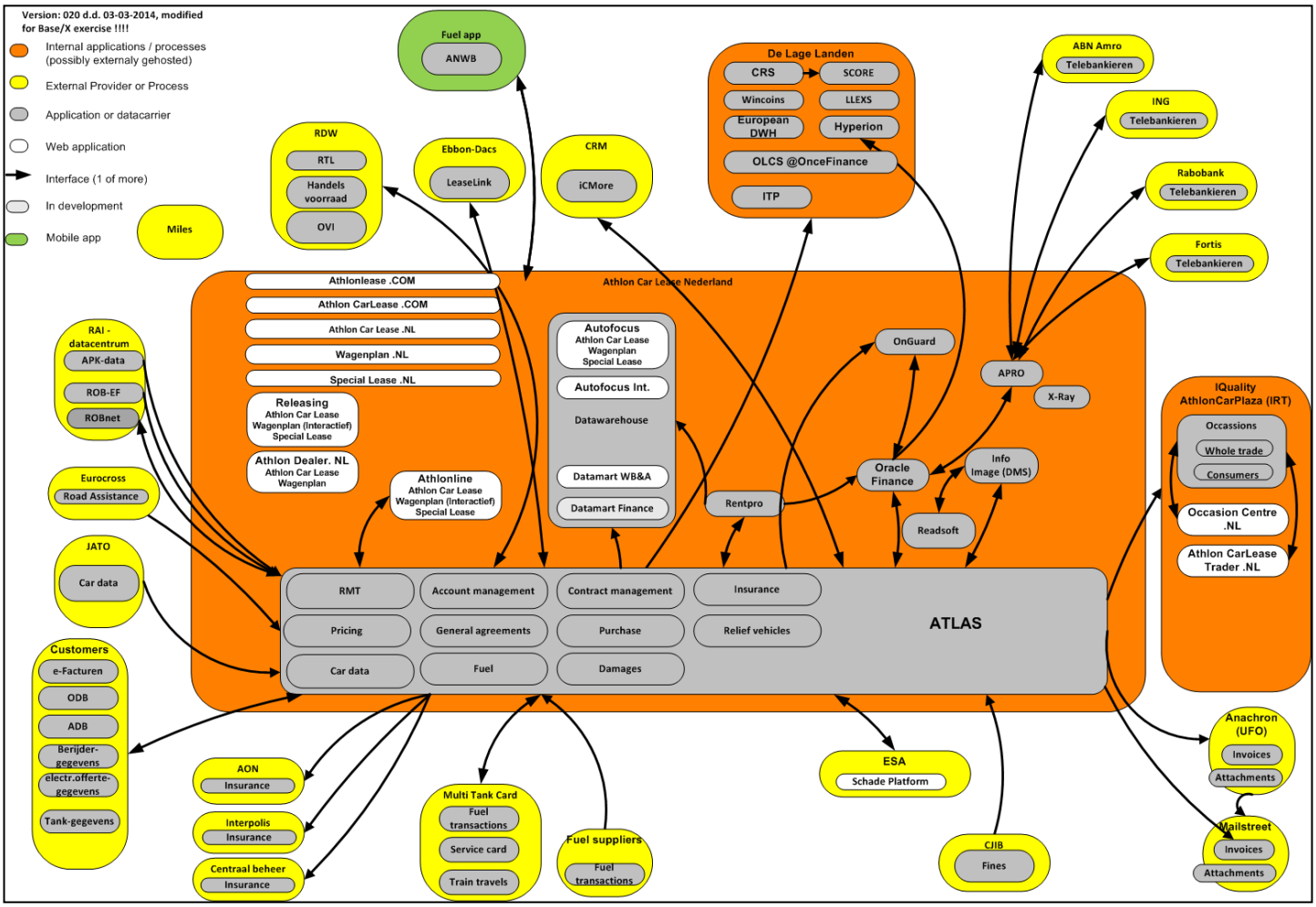


Figure 16 Athlon Car Lease Application Landscape

- ATLAS: This is the overall contract management system. It's the core ERP system of Athlon Car Lease which supports various business capabilities and vehicle related services. It contains the following list of functionality:
    - RMT: Performs services related to Repair, Maintenance and Tires.
    - Pricing: Contains routines calculating the lease rate of a given vehicle.
    - Car data: Provides information about car details.
    - Account management: Supports customers' management.
    - General Agreements: Covers all the aspects of the business relations. Contracts (one for each vehicle) are pinned to the General Agreements. Such agreements exist for each customer.
    - Fuel: Supports the fuel management of the vehicles. It interacts with systems for managing fuel transactions and fuel cards, as we will explain later.

    - Contract management: The main system for providing services related to contracts.
    - Purchase: Services related to the actual purchase of a vehicle are performed in this application.
    - Damages: Supports services for damages occurring on the vehicles.
    - Insurance: Manages the insurance of each vehicle by interacting with external providers.
    - Relief vehicles: Used when a replacement of a vehicle is needed (in case the lease car is under maintenance or repair).
- Datewarehouse: Keeps the data and any information enters ATLAS system.
- Oracle Finance: Supports all finance services of the back office systems.
- X-Ray: Used to support determination of the residual value of vehicles (car data).
- Athlonline: Web-based platform that can be used by customers/drivers to configure and calculate the cost of a new lease vehicle. Generates offers and handles the online quotation process.
- De Lage Landen: Container for a set of applications like @OnceFinance, which is for credit scoring of prospects and existing customers (the rest applications are of unknown functionality or not relevant to the car leasing business model).
- Info Image (DMS): The Document Management System which stores digital copies of documents.
- CRM: Contains the iCMore platform which is the cloud-based CRM system that Athlon has, based on Siebel on Demand by Oracle.
- Ebbon-Dacs: Supplier of the Leaselink application. Leaselink is a web-based application that handles the communication between Athlon and dealers in the Netherlands for supporting ordering and delivering of cars.
- Multi Tank Card: Manages fuel cards and fuel transactions for Athlon vehicles (if a fuel card is part of the lease contract).

Elements that may be considered into the design of the architecture, depending on possible usefulness and if time permits are the following:

- APRO: The out of the box solution for creating payments, direct debits and bank reconciliation. It is the Banking Gateway for connections with banks that Athlon collaborates with. Since in our project, we will not implement any interaction with banks, we may use this system for payments purposes internally.
- ESA (Electronic Damage Management): Platform used to manage the communication between Athlon and suppliers that repair cars (the repair refers to actions after a damage, not to actions taken in RMT). This system contains the "Schade platform" which is the application that offers that communication. This is not a high priority external system to include in our architecture, however we may consider it if time permits.
- CJIB: Company that sends out traffic fines for the Dutch government. Athlon receives fines for all the vehicles that are registered directly to Athlon. We may consider a direct communication to this provider in order to get any fines in our prototype, otherwise we can use the existing fines in the internal systems.
- JATO: Supplier of all car data Athlon uses in its systems. If we can extract such information from the Car data application of ATLAS, this can be excluded.

- RAI Data Centrum: Central Dutch organization that supports all sorts of car data inquiries. It hosts the applications:
    - APK data: This system informs about the next due date of any general periodical car check. This is an obligatory check of vehicles that are older than 3 or 4 year.
    - ROB-EF: System used for electronic invoicing. It is related to invoices for Repair, Maintenance and Tires. Athlon receives electronic invoices from the dealers that worked on the RMT.
    - ROBnet. System that is used for the interaction between Athlon and dealers. Dealers ask approval for RMT from Athlon. Athlon grants approval or rejects the request (sometimes partially). Without approval, no RMT is done on any vehicle.

  These applications provided by RAI are interesting to include in our systems, since interactions with car dealers for RMT are part of the lease contract lifecycle. However, since it is an external provider and we have already included enough external systems, we decided to put it in hold.

From the entire application landscape we decided to not cover some systems due to either their complexity, either their irrelevance to our purposes of prototyping main business processes or time constraints. These are:

- Athlonlease .Com, Athlon CarLease .Com, Athlon Car Lease .NL, Wagenplan .NL, Special Lease .NL: These are general communication platform with customers. All are web-based marketing tools. Such systems are considered irrelevant for our purposes.
- Releasing: Application for releasing cars.
- Athlon Dealer .NL: Web platform for data interchange between Athlon and dealers. We exclude it since it has become partially obsolete because of the introduction of Leaselink.
- Rentpro: Handles services related to short-term car rental.
- OnGuard: Financial system to support money collection. In case of arrears, customers are contacted and persuaded to pay their bills. In our prototype we will make the assumption that such scenarios are not possible and therefore we can omit this application.
- Readsoft: The scanning software for paper documents. It is connected to Info Image (DMS) and we can exclude it since we will handle only digital documents.
- ABN Amro, ING, Rabobank, Fortis: Banks in Netherlands that Athlon collaborates with. Telebankieren is the software used to manage bank accounts online. Actual transactions with banks are out of scope of this project, excluding thus these applications.
- IQuality AthlonCarPlaza (IRT): All applications in this quadrant aim for the remarketing of cars, whether that is to sale dealers or to consumers. Since our focus is on processes related to the leasing of a new car and not on selling and ex-lease one, we will omit that system from our landscape.
- Anachron (UFO): Supplier that handles the outgoing digital invoices for Athlon. We will handle documents internally, so we will not cover this application.

- Mailstreet: Supplier that handles the outgoing paper invoices for Athlon. We will handle documents internally (only digital ones), so we will not cover this application.
- Fuel suppliers: Resembles the Multi Tank Card organization, used for fuel transactions. The functionality is more or less the same as MTC, therefore there is no reason to duplicate work.
- AON, Interpolis, Centraal beheer: Insurance companies that provide insurance on Athlon vehicles. Although insurance is an important part of car leasing, we set the priority of implementing connections to insurance companies as low due to time constraints.
- Customers: Container of several systems that aim to inform customers about data regarding fleet, drivers, fuel, etc. It is mainly used for reporting purposes and it is out of our scope.
- Eurocross: Supplier providing road assistance. By simplifying our business cases, we will exclude scenarios where road assistance is needed and we will not cover this application.
- Miles: ERP leasing system, not in use for the Netherlands but in use for Portugal, Spain and Italy, thus out of scope.
- RDW (Rijksdienst voor het Wegverkeer): Official organization within the Netherlands that manages vehicle registration and other administrative tasks. It offers applications like RTL (Registration Ascription Lease company) which helps to register a lease vehicle to a driver, while the vehicle actually belongs to the lessor. We do not see any interest to include such a system that will be useful for any service compositions though.
- Fuel app/ANWB (Algemene Nederlandse Wielrijders Bond): A consumer organization supportive to the interests of people that own a vehicle. It is irrelevant to our case.

The aforementioned distinction leads to a projection of the entire Application Landscape of Figure 16, shown in Figure 17.

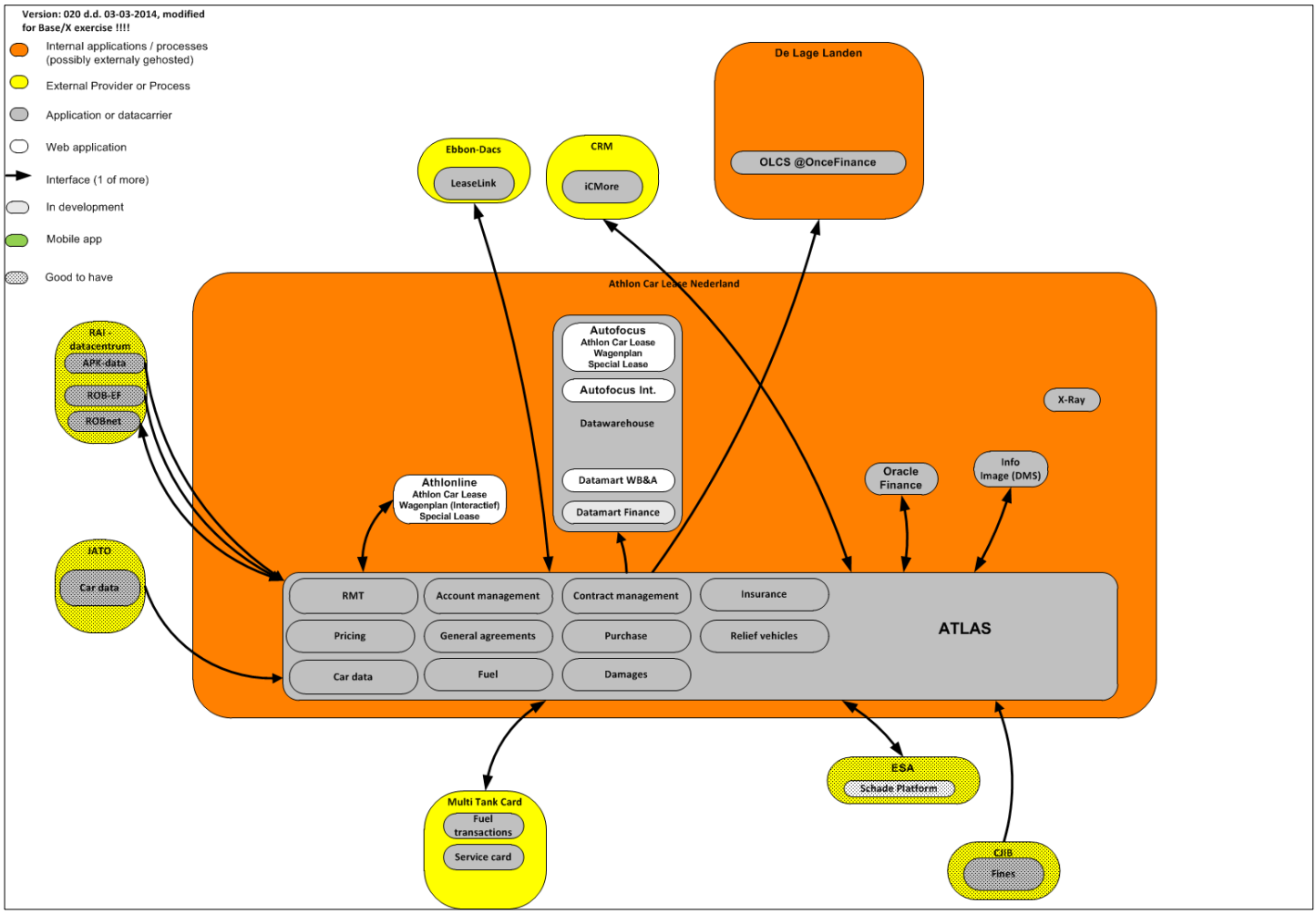


Figure 17 Projection of the entire Application Landscape

3.4 Simplified Application Landscape Specifications

Having charted the current application landscape and selected the main systems and applications that will facilitate our service compositions, we redraw the landscape with these remaining systems. This is done with a UML component diagram showing the interfaces of each system and the linking to each other. We draw a UML 2 component diagram, implementing it with the notation of MS Visual Studio 2013[23].

Figure 18 shows the UML diagram of the simplified application landscape.

[23] http://msdn.microsoft.com/en-us/library/dd409390.aspx, retrieved September 10th 2014.

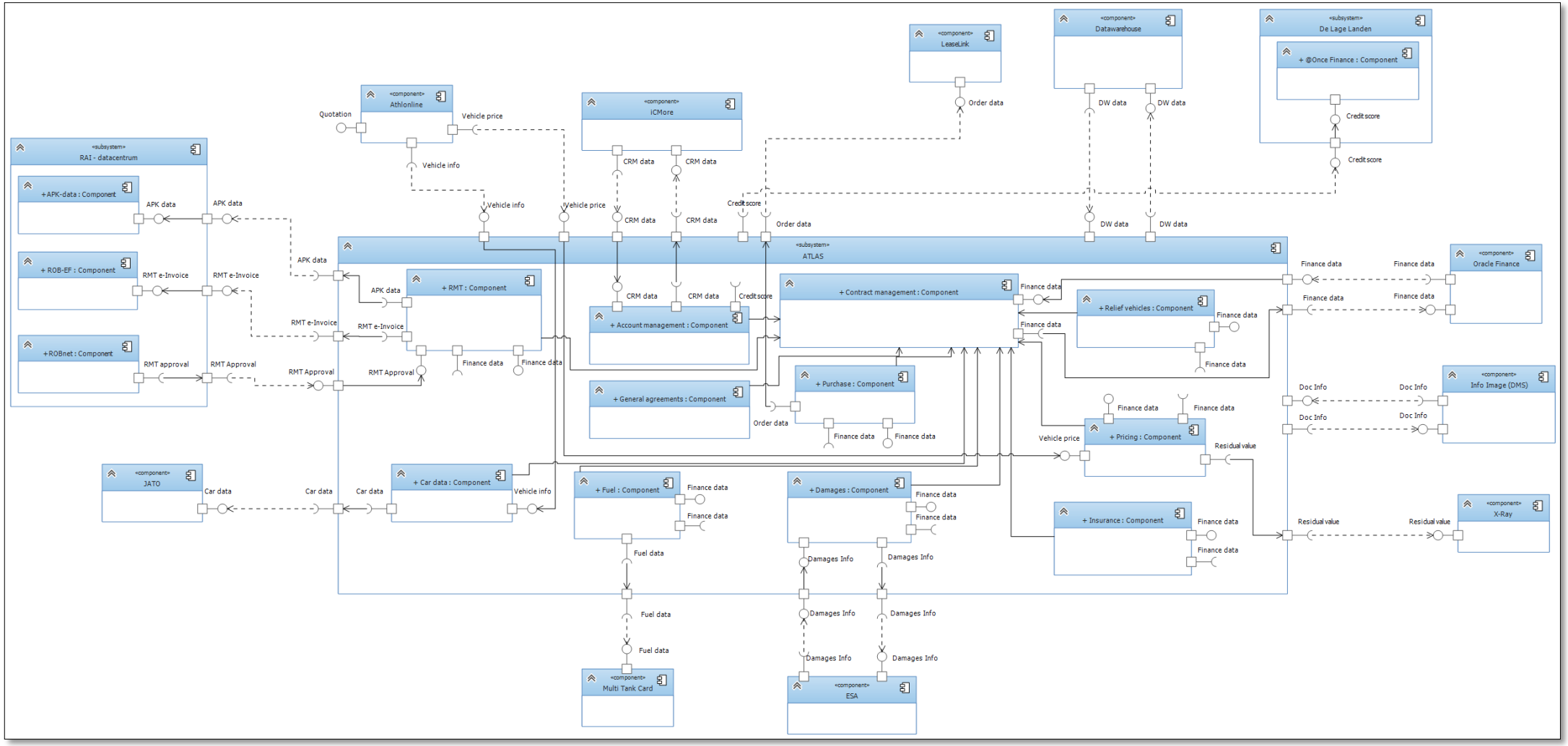


Figure 18 UML Component diagram of simplified application landscape

We have to note the following about the UML component diagram:

- With the help of the design cube of Figure 9, we place the above model in the Aggregation level 1, first Abstraction level (we assume that more abstract levels can be used but we do not see any interest to abstract more the concepts) and with respect to Realization dimension we can say that we use the Architecture level of BOAT framework. The aspect we are looking the model at is the software aspect (see Section 3.1.1). We use this as the leading aspect of our architecture and skip the rest since our aim is to describe the structure of the applications. We visualize this combination in Figure 19 below.

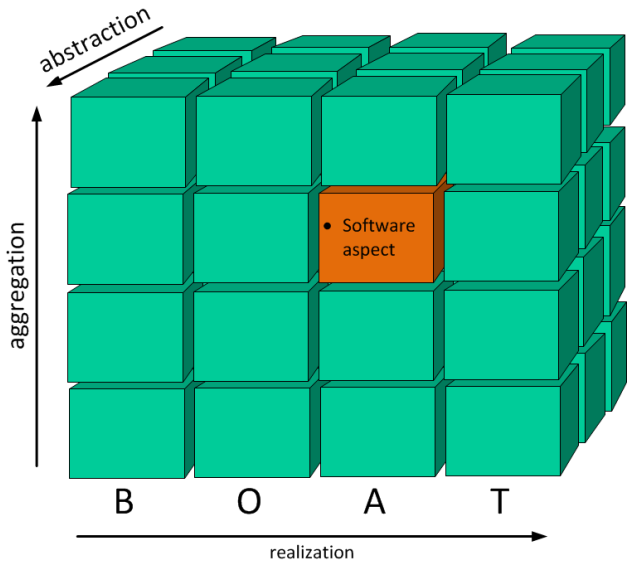


Figure 19 Simplified Application Landscape in design cube

In Aggregation level 0 we would have the ATLAS system as a black box which does not add any value to the design process. Therefore we decided to skip that level.

- Our aim is to model how we perceive the simplified landscape of Figure 17 in order to use that as a guide of the designing of our architecture. Thus, in the above figure, the orange cell is a starting point. Next steps regarding concrete details of the systems, information about platforms and protocols of communication are presented in Chapter 6 where we handle the standard architecture.
- We give the same names to interfaces of components that communicate to each other for better consistency.
- Applications that have a bi-directional communication are modelled with both type of interfaces (provided and required) with the same name.
- Datawarehouse keeps information of every application of ATLAS system. For sake of simplicity, we did not draw a connection of each application to Datawarehouse in order to avoid a spaghetti diagram. We connect though the Datawarehouse component to the ATLAS subsystem, meaning that all the internal components of ATLAS interact with the Datawarehouse. The same design choice is applied to Info Image component. We connect it to the ATLAS system, not to each internal component individually.

One could argue here that connecting Datawarehouse to ATLAS is like having a model of Aggregation level 0 (as we said earlier that ATLAS is seen as a black box). However, we do not try to put two aggregation levels in one picture but just save space and not make the picture unreadable.

- Many internal applications of ATLAS have links to Oracle Finance component. In order to avoid drawing many ports, we decided to design only one interface of each type (provided/required) between ATLAS and Oracle Finance. All related internal applications have interfaces for connection to finance activities that delegate to the "finance" port of ATLAS. In the diagram above, we show the finance interfaces of each application but we did not draw the links to the main ATLAS port so as to make it less messy. The only delegation links are to the Contract management component (same decision choice on how to represent aggregation levels as above).
- The connection between components of the same parent components (internal applications of ATLAS) should be done through Part Assembly links between the corresponding required and provided interfaces. However, we decided to make it simpler and connect internal applications with just Connectors links. For example, most of the applications are linked to the Contract management component. This is also an indication of how tightly-coupled are the systems in the current Application landscape. If each system has its own interfaces, then it will be easier to be decoupled from another system.

Chapter 4

Service Dominant Business Paradigm

Having discussed the necessity of the architecture of Information Systems and having charted the application landscape of the organization in Chapter 3, we focus here on the "servitization" of these systems, the method of making them ready for the service-dominant business paradigm.

In a service-dominant world, companies and service providers are organized into agile networks in which participants provide each other with specialized services. In that way, new offerings are provided to customers through business processes in which several partners collaborate. In order to be able to compose services, there must be the appropriate support by means of applications and platforms. As we discussed in previous chapters, Business Process Management Systems offer the automation and orchestration of processes. In these processes, applications and services are invoked using standard interfaces and protocols. However, in the current situation where the application landscape exists and probably is old, there is a high need for transforming applications and systems so they can be ready to be used by a BPM tool in a service-oriented approach.

By service-enabling its existing legacy systems and using service-oriented development techniques for new application development, the enterprise should be able to create loosely coupled and reusable services that can be composed and orchestrated into complex business processes which integrate human tasks and systems across departmental silos [22].

In Section 4.1 below, we explain the idea of wrapping of legacy systems.

4.1 Wrapping Legacy Systems

Service-oriented architecture (SOA) can be viewed as an architectural construct for flexible connection of separate components in response to changes in business. SOA focuses on the exchange of information among major software components and on the reusability of the components by separating the interface from the internal implementation [23]. There are several features of SOA approaches that allow the modernization of legacy systems including loose coupling, reusability, statelessness, abstraction of underlying logic, standards-based and protocol-independent distributed computing. Information on each principle of SOA can be found in [24]. In this section, we use the concepts of modularization of services and encapsulation to explain how we can achieve the servitization of the current legacy systems.

4.1.1 Modularization

An SOA provides a flexible architecture that unifies business processes by modularizing large applications into services. In a single information system, we distinguish service modules that can offer specific functionality. The purpose of modularization is service composability. Services should be standardized within an organization (or even across its boundaries), such that the offered functionalities can be reused and participate into multiple compositions.

In Figure 20 we can visualize the application modularization of information systems into service modules.

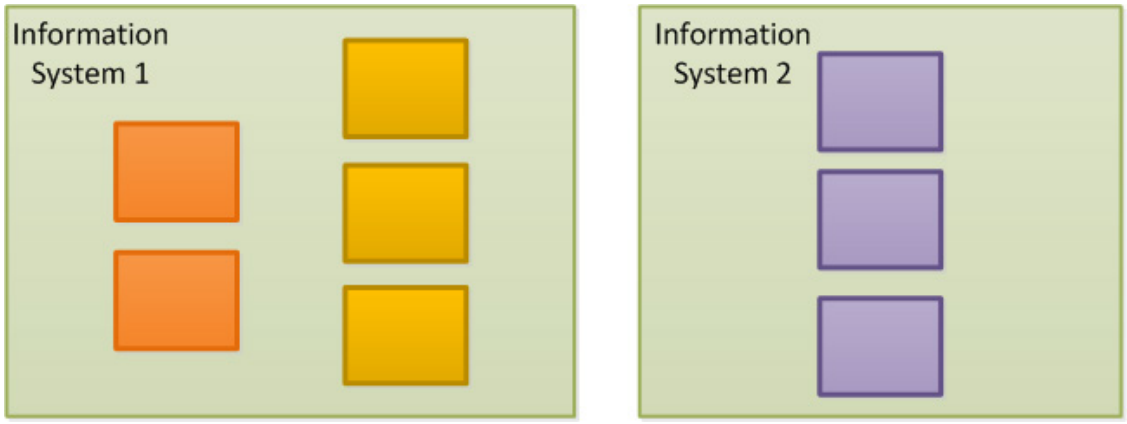


Figure 20 Application Modularization of Information Systems

4.1.2 Encapsulation

The concept of encapsulation is used to provide a higher level of abstraction. By encapsulating the functionality of a module from its environment with the definition of explicit interfaces, we are able to hide any internal complexity. The use of a middleware-compliant software packaging (the "wrapper") provides the ability to call a module that resides in an information system. The wrapper has the duty to convert all communication between the system and its environment from the middleware standard to the internal mechanisms and the other way round [2].

In Figure 21 we see how we can call a module A from an Information System through an Interface A.

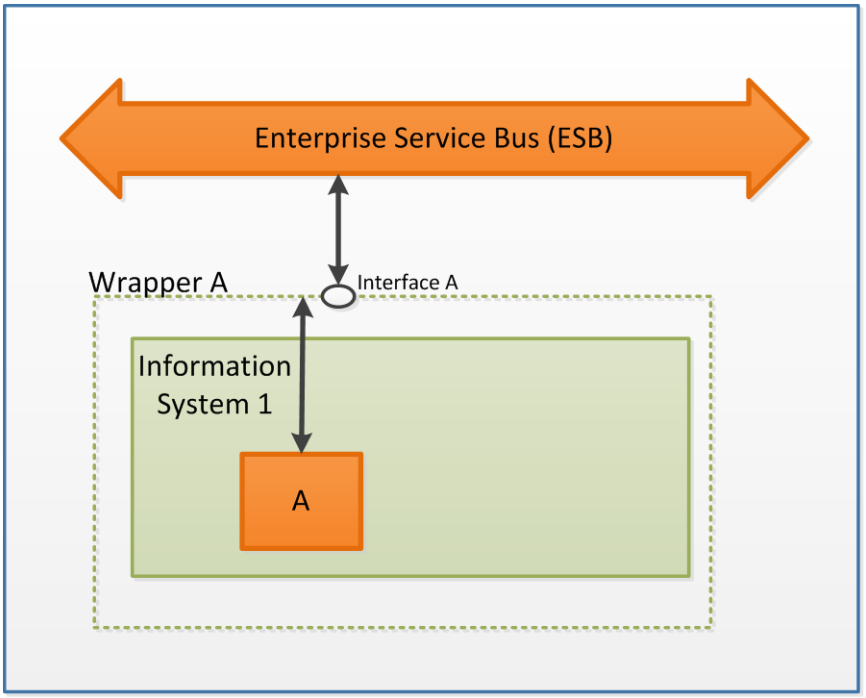


Figure 21 Encapsulation of a service module

Suppose now that in the above Information System there is a service D which we want to call. Again, a wrapper with a well-defined Interface D provides that functionality. We visualize the encapsulation of both services A and D in Figure 22.

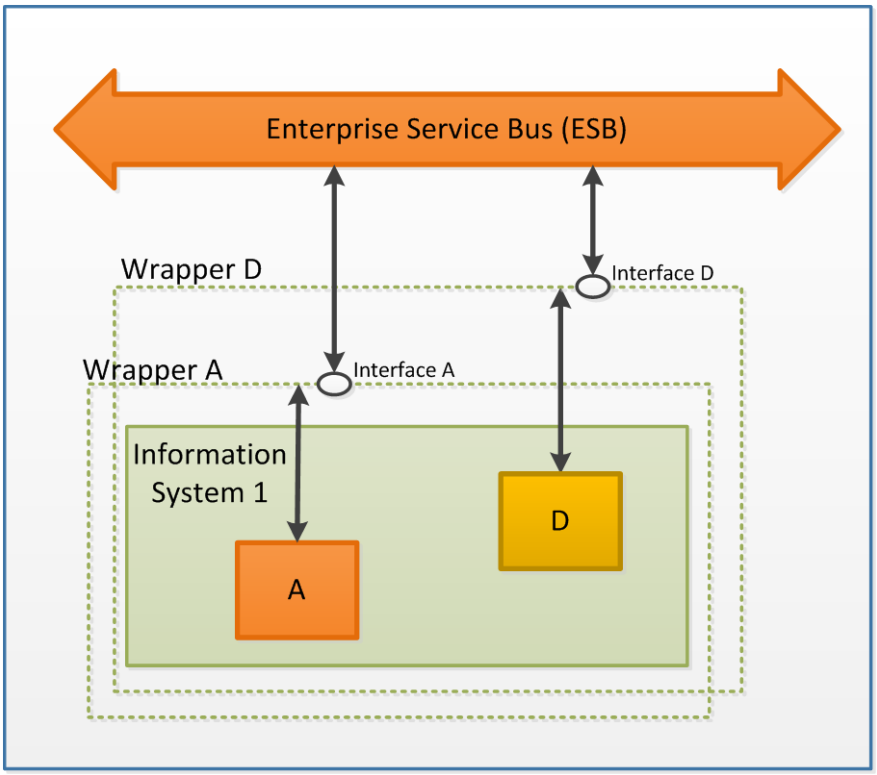


Figure 22 Encapsulation of multiple service modules

Figure 22 is a conceptual model of the encapsulation of services in a single Information System. As we will see later in Chapter 6 in more detailed models, the wrapping is done by a single wrapper with multiple interfaces and not by multiple wrappers, one for each service.

4.1.3 Mapping legacy situation to greenfield situation

The models we draw in Section 3.2 and especially those of Architecture and Technology level of realization dimension in Figure 15 refer to a greenfield situation where the architecture is to be designed from scratch without pre-existing technical constraints. When this situation is not the case for an organization, we have to adapt the current architecture in order to be mapped to the architectural models we discussed in Section 3.2.

Assume now that we have a composition of Services A, B, C, D where services A, B, D are provided by a Managed Service Platform (MSP) and service C is a manual service. All these are orchestrated by a BPEL engine and the invocation is done through an ESB. This situation is presented below in Figure 23.

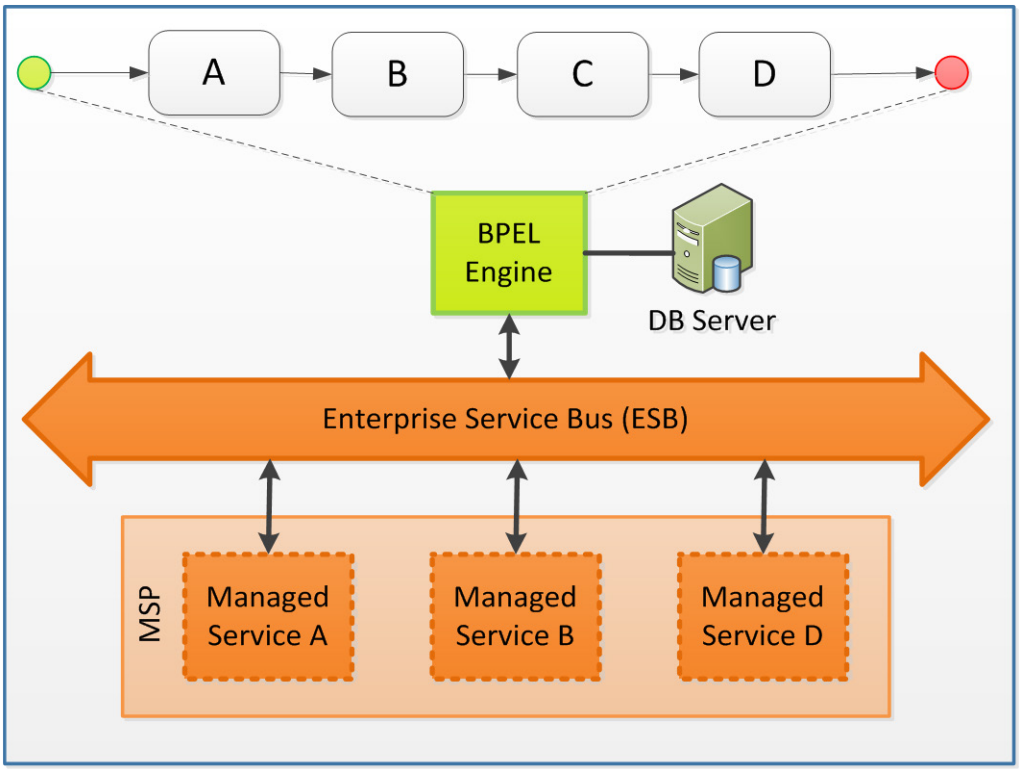


Figure 23 Greenfield situation of service orchestration

In a current legacy architecture of an organization, services A, B, D may be resided into two individual Information Systems. After modularizing these Information Systems in order to use these services explicitly, we encapsulate them so they can be invoked by an ESB.

The same service composition of Figure 23 is now achieved with the model of Figure 24.

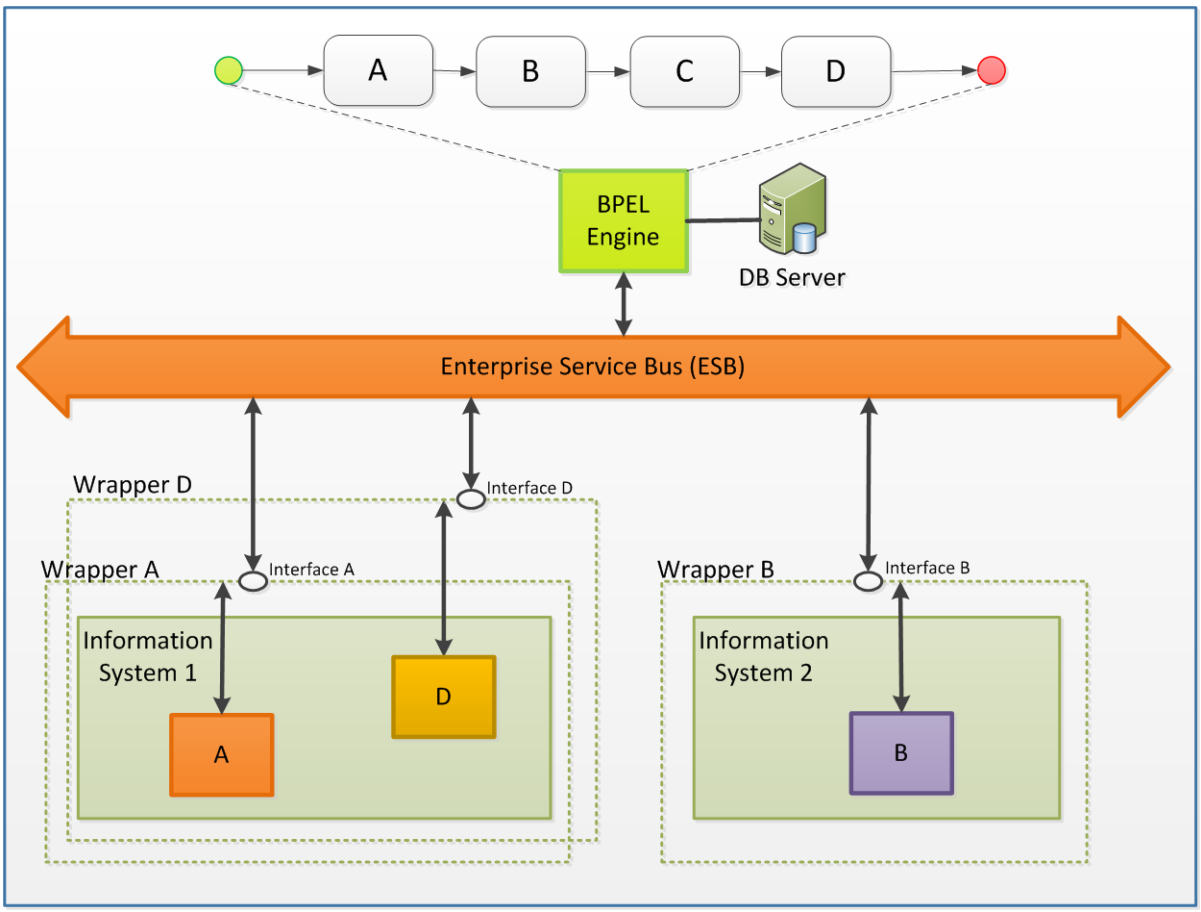


Figure 24 Targeted situation of service orchestration

Placing the current architectural model in the three-dimensional design cube of Figure 9, we can say that it is in the Technology level of the realization dimension. As we explained in Section 3.2.7, the aggregation level does not make any difference in the concept of service orchestration (whether the modules refer to business services or building block services). Thus, we can merge the cells of two aggregation levels into one. This is shown in Figure 25 below.

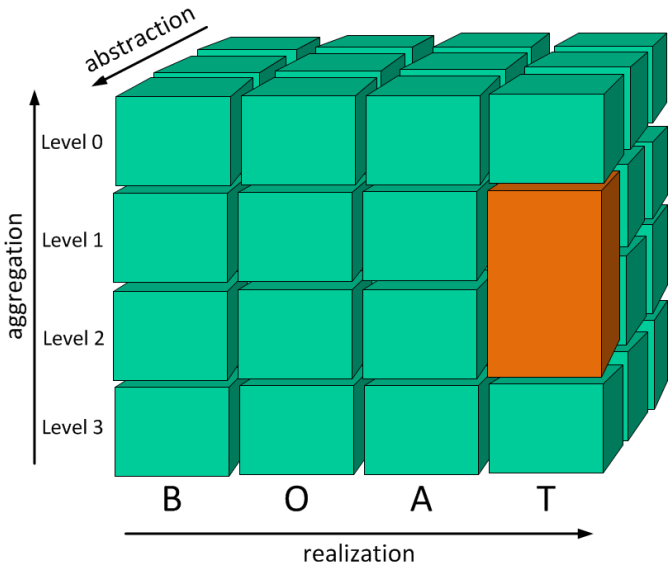


Figure 25 Positioning service orchestration in legacy situation

Chapter 5

Service Compositions

In the document so far we have discussed the BPM aspect of the project for the selection of a tool for the prototype (Chapter 2), the architecture viewpoint for specifying the steps and elements in order to design the standard architecture (Chapter 3) and how to achieve the "servitization" of legacy systems (Chapter 4). In this chapter, we complete the Preparatory phase of our approach by discussing the business viewpoint of the project, the service compositions.

As we have already mentioned, the focus of this project is on the service composition layer of the BASE/X framework. In order to set any compositions, we use as input, elements from the adjacent layers, i.e. the service-dominant business model and the business services. By using a set of services, we operationalize a specific business model in terms of service compositions. The service compositions are specified with the corresponding business processes that are subsequently designed and executed in the BPM platform.

Thus, the subject of this chapter is to set up the input for the design of the prototype. We first discuss about the car leasing business model in Section 5.1. Then, in Section 5.2 we present the use case of Athlon Car Lease we use for our prototype and finally we present the conceptual models of the service compositions in Section 5.3.

5.1 Car Leasing Business Model

A business model is a set of assumptions about how an organization will create value for all its stakeholders [25]. The design of a business model is done by using tools like the Business Model Canvas (BMC) [26]. However, approaches like this are typically not focusing on service-dominant business and are organization-centric, not network-centric. Therefore, we use here a tool proposed in [3] that has a service-dominant starting point, called Service Dominant Business Model Radar (SDBM/R).

In the current project we use the Car Leasing business model of Athlon Car Lease. This model is already drawn in the BMC and is presented in Appendix B. Using this as the base we apply the approach presented in [3] in order to create the SDBM/R from the BMC.

Athlon Car Lease offers a variety of services for car leasing like full operational service leasing, fleet management and vehicle purchasing. The complete business model cannot be within the scope of our project, so we are focusing on a representative simplification of it. More specifically, we handle the case of leasing a vehicle to a customer providing also maintenance and fuel management for a hassle-free use of a vehicle. This is the value co-creation proposition of the simplified business model. The main co-creator actors that contribute to that value are the focal organization Athlon Car Lease, the dealer network

ROBnet responsible for repair, maintenance and tires (RMT), Multi Tank Card (MTC) for fuel management and of course the driver who will use the vehicle.

The focal organization provides long-term mobility through leasing a vehicle. The lease is based on a contract upon agreement with the customer. The financial benefits for Athlon are the lease rates while on the other hand it incurs with the costs of funding the purchase of a vehicle.

Dealers are the actors that provide the vehicle. Athlon communicates with a number of dealers to get some offers and finally proceed to the actual purchase of the vehicle. Their operational costs are compensated by the sales of their assets.

Vereniging ROB is the network that provides RMT and contributes to the business model by ensuring proper functioning of a lease vehicle. The benefits for this actor are the fees gained by providing its offerings which of course require operational costs.

Multi Tank Card is the organization that provides fuel cards that a driver can use in a gas station without having to pay. It aims to customer's convenience and the benefits occur through the transaction fees. The management of the cards and the building of the supporting systems are the main financial costs of MTC.

The final actor of the business model is the driver who experiences the hassle-free use of a vehicle. He plays an active role in selecting the vehicle of his needs and the following services when the vehicle is obtained. The costs for him are the lease rates he pays during the lease contract and any possible service fees. (In the complete car leasing business model of Athlon, fleet managers are the customers of Athlon who manage the drivers of their companies. We simplified however the business model by including only the driver who gets benefited from the value-in-use).

All the above are depicted in the radar tool shown in Figure 26.

Figure 26 SDBM Radar for Car Leasing business model

5.2 Driver Desk Use Case

Athlon offers a single point of contact for all questions regarding lease vehicles, called Driver Desk. Drivers can turn to the Driver Desk, with literally any type of request, from the moment that their vehicle is on the road, up and until it is being returned. Through several communication channels such as telephone, e-mail or internet portal, they communicate with Athlon in order to ask a service or a combination of them. Driver Desk supports drivers on all the following topics:

- Fines
- Fuel cards (MTC)
- Fuel management
- Repair, maintenance, tires (RMT)
- Damages
- Returning vehicles
- Replacement cars
- Ordering progress (occasionally)
- General complaints handling

The offered services are explained in the sequel of this section.

5.2.1 Business Services

We defined in Section 3.2.4 the business service as the encapsulated business functionality that brings value-in-use to the customer. There is an interaction between a business service and a customer which is achieved by a well-defined input and output.

Business services of Athlon Car Lease have already been identified in a previous project leading to a service catalogue [4]. From that catalogue, not all services are relevant to the business model we presented in the previous section. Also, many others need modification or are missing at all. In Table 2 we present the list of business services we use for the purposes of our prototype. This is not an exhaustive list, but a sample of services that Driver Desk can handle. For example, for Repair, Maintenance and Tires (RMT) services, Driver Desk can reply to requests about next scheduled maintenance but also to requests regarding information on fix store where the vehicle can be repaired.

Table 2 Business Services list

Business Service	Input	Functionality	Output
Enabling online contact to Driver Desk	- Driver's information (name, contract No, vehicle license plate) - Driver's Inquiry	To empower the driver to request a service/inquiry online	- Information on delivered service/Reason for not completion
Providing information on registered fines	- Driver's information (name, contract No, vehicle license plate) - Request for fines report	To provide the driver information on the fines he is charged for	- Fines transactions report
Replacement of fuel cards	- Driver's information (name, contract No, vehicle license plate) - Reason for replacement	To replace an existing fuel card in case of loss/theft/defect.	- Delivered new fuel card to driver
Providing information on nearest gas stations	- Driver's information (name, fuel card number, location)	To provide information on nearest gas stations with their addresses and the fuel prices.	- List of gas stations with addresses and fuel prices.
Providing information on fuels transactions	- Driver's information (name, contract No, vehicle license plate)	To provide the driver information on the fuels transactions in a last predefined period	- Fuels transactions report

	- Request for fuel transactions report		
Providing information on Repair, Maintenance, Tires (RMT)	- Driver's information (name, contract No, vehicle license plate) - Request for RMT information	To provide the driver information on RMT services like next date of car's maintenance	- Response with information on RMT services
Providing information on damages	- Driver's information (name, contract No, vehicle license plate) - Request for information on damages	To provide the driver information on damages like costs or expected date of repairing	- Response with information on damage
Providing information on returning vehicles	- Driver's information (name, contract No, vehicle license plate) - Request for information on returning vehicles	To provide the driver information on returning a vehicle like time and location	- Response with information on returning vehicles
Enabling request for a change on a replacement car	- Driver's information (name, contract No, vehicle license plate) - Preferences on replacement vehicle	To enable driver to ask for a replacement car of his needs.	- Delivery of replacement car
Providing information on ordering progress	- Driver's information (name, contract No, vehicle license plate) - Driver's Inquiry	To provide the driver information on the status of an order like expected date of vehicle delivery	- Response with information on ordering progress

Note here that the handling of general complaints is not a specified service but is left up to the staff of Driver Desk to handle those scenarios. This is explained how in Section 7.1.

5.3 Service Composition Specifications

According to [3], a Business Service Composition is the combined application of Business Services for solving a customer's problem. This combination is achieved by defining the interaction of business services as a business process, where each business service represents a partial solution of the complete customer problem.

Service compositions can be either of process type or mash-up type (in practice hybrid types are found). In process type, activities of multiple actors need to be synchronized and information needs to be passed sequentially. The management of the state is responsibility of a service orchestrator. On the other hand, mash-up types are used when a single actor invokes the functionalities of other actors and it is his responsibility to manage the state of service delivery [1].

In our project we are applying a hybrid type. As we discussed in Section 3.2, we distinguish business processes between customer-facing, where steps in the process definition have interaction with the customer, and internal ones where steps that are taken to deliver a service are invisible to the consumer. In the use case of Driver Desk, the interaction with the customer can be implemented as a mash-up service composition where the customer is free to select and combine services from the ones presented in Table 2. On the other hand, the internal business processes can be seen as a process type service composition since the combination of building block of services are predefined, structured and standardized.

In Chapter 7 we explain how we implement the hybrid type of service composition. In the next subsection we design the conceptual business process models for the internal process definition which are subsequently used in the implementation phase.

5.3.1 Internal Business Process Models

The business services in Table 2 are those offered to customer through the Driver Desk. The steps taken to fulfil a driver's response define the internal business processes. In our prototype, we implement four of these services, the inquiry of a fines report, the request of nearest gas stations, a list of fuel transactions and the replacement of a fuel card (as a mock-up service though). The processes are described below.

Fines report: A driver asks for an aggregated report of his fines in a last predefined period. Given the required period, driver's data and contract, the assigned employee gathers all the fines and creates the report. This is the output of the task which is passed to Driver Desk in order to respond to the customer by sending the report.

In Figure 27 we present the process described above, modelled with a tool that uses the Business Process Model and Notation (BPMN), Version 2.0[24].

[24] http://www.omg.org/spec/BPMN/2.0/

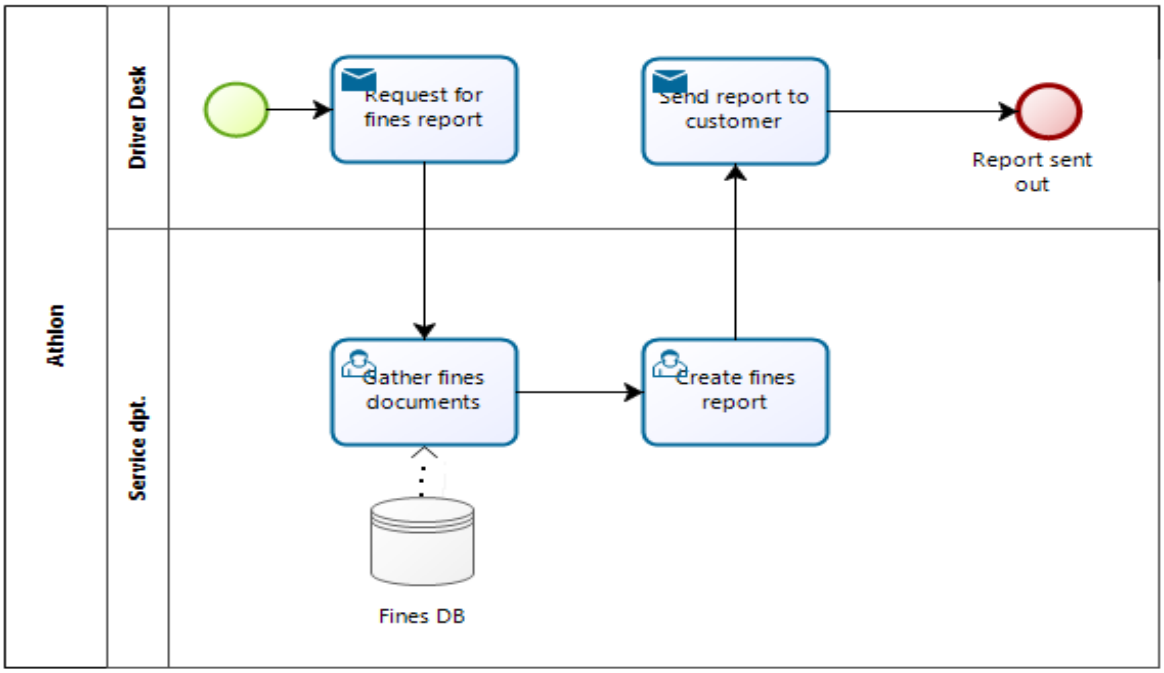


Figure 27 Fines report internal process

Information on nearest gas stations (MTC): A driver calls to ask for a list of the nearest gas stations along with the fuel prices, based on his current location. After registering his data, the assigned employee sends the request to the provider (MTC) and gets the response in his screen. He can then either inform the customer by reading out any records or sending an email to him. The process is shown in Figure 28 below.

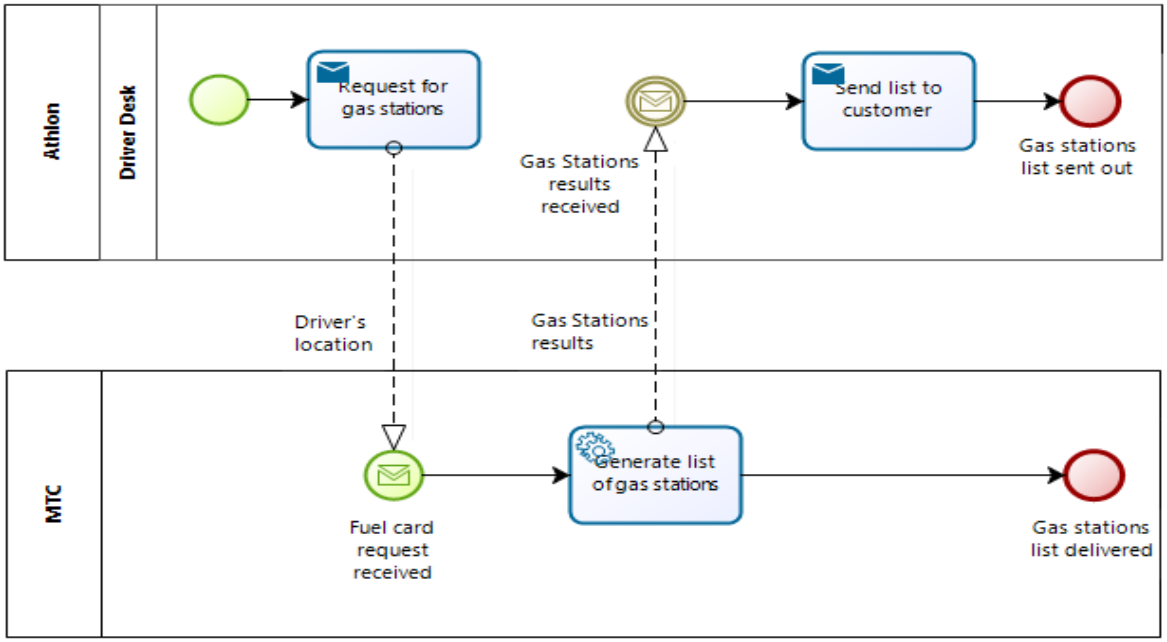


Figure 28 Information on gas stations internal process

Information on fuel transactions (MTC): A driver calls to ask for a list of fuel transactions in a predefined period. After registering his data, the assigned employee sends the request to the provider (MTC) and gets the response in his screen. He can then either inform the customer by reading out any records or sending an email to him. The process is shown in Figure 29 below.

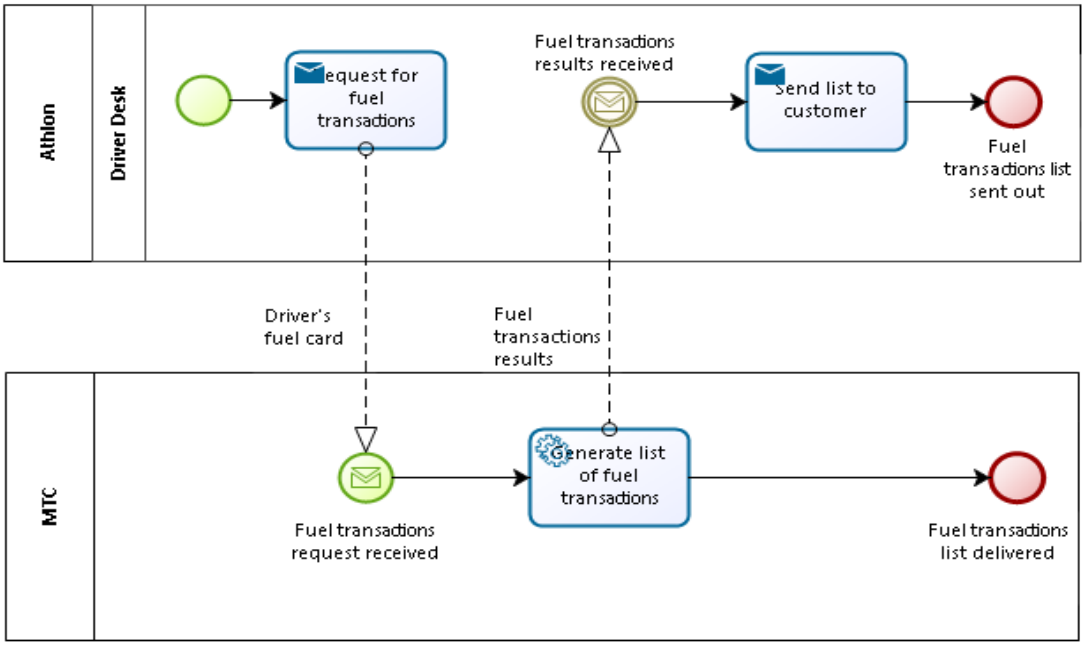


Figure 29 Information on fuel transactions internal process

Fuel card replacement (MTC): A driver calls for a card replacement. After registering his data, the assigned employee select driver's declared card and registers the reason of replacement. In parallel, he creates a request for termination to be sent to the fuel cards service provider (MTC) and starts the process for replacing the card. He first checks if a card exists in the stock. If yes, he registers the new details, informs the Driver Desk about the number and estimated deliver date and sends out the card to customer's address. If there is no card in the stock, the employee creates a new request and sends it to MTC. When MTC issue a card and send it to Athlon, Driver's Desk is informed, card is sent to driver. The internal process is shown in Figure 30.

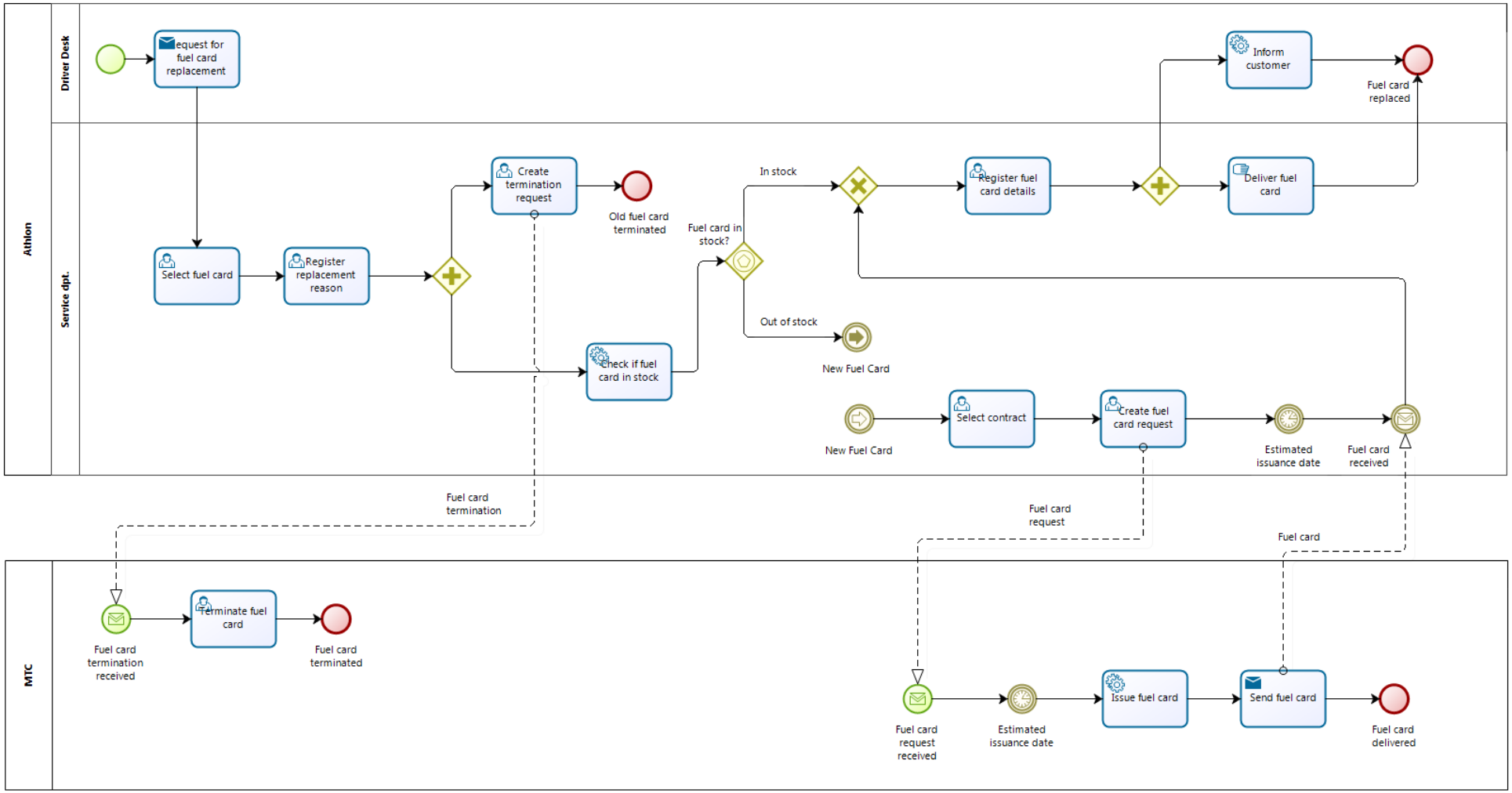


Figure 30 Fuel Card replacement internal process

We still see that some steps require being exploded to finer details. However, since here the goal is to set the input and use it as a basis for the actual implementation of the prototype, these processes are elaborated in more details during the implementation phase and presented in Chapter 7.

Chapter 6

Standard Architecture Model

In Section 3.1.1 we explained how the three-dimensional design cube helps us to design the standard architecture. In the report so far we have drawn a number of models to introduce the concepts of business process orchestration (Section 3.2) and "servitization" of legacy systems (Section 4.1). These models though are of high level of abstraction, having no concrete details.

In this chapter we continue with the design process, focusing on the abstraction dimension. Based on the architectural models we have already drawn and based on the existing IT architecture of the organization, we specify the targeted standard architecture with more specific details.

First, in Section 6.1, we present the standard architecture model with more details in the Technology level of the realization dimension. Then, in Section 6.2 we parameterize the reference architecture models into more detailed, organization-specific context.

6.1 Technology Embodiment

To accommodate business agility, enterprises are supposed to streamline (existing) business processes while exposing the various packaged and home-grown applications found spread throughout the enterprise in a highly standardized manner. A contemporary approach for addressing these critical issues is embodied by Web Services [29] that can be easily assembled to form a collection of autonomous and loosely coupled business processes [30]. Web services use the Internet (and intranet) as communication infrastructure and Internet-based standards including the Simple Object Access Protocol [31] (SOAP) for transmitting data, the Web Services Description Language [32] (WSDL) for defining services, and the Business Process Execution Language for Web Services [33] (BPEL4WS, BPEL for short) for orchestrating services.

With the help of these and a few additional standards, we explain below how is the realization of service compositions achieved with current technology.

6.1.1 Integration Layer

We start the discussion about the technology embodiment with the integration layer since it gathers all the key standards and the required functionality to support a SOA and integrate other layers. As we already mentioned in Section 3.2.7, an Enterprise Service Bus (ESB) plays the role of the communication broker. An ESB decouples the service requester from the service provider by mediating (if necessary) the service interaction between communicating partners which usually possess differing service characteristics. The main tasks of an ESB are [34]:

- *Routing*: The ESB acts as a match-maker between service requester and service provider.
- *Conversion*: The ESB handles protocol or interaction differences.
- *Transformation*: The ESB can be programmed to handle interface mismatches that might occur and compensate accordingly, if possible.

An ESB that supports Web Services standards (WS-*) provide a convenient way to add web service connectivity to existing applications that lack web service interfaces. Below we present the most common standards that are useful for the "servitization" of existing legacy systems and for the further facilitation of service compositions. According to [34], these are subdivided into the following interoperability aspects:

- *Transfer*: An ESB must support the use of HTTP or HTTPS as a transfer protocol to carry messages between the service requester and the service provider. Other transport mechanisms can be supported, for example Java Message Service (JMS).
- *Message Format and Protocols*: Messaging protocols define both the message structure and rules governing the processing along the message path between message participants. An ESB must support the use of Extensible Markup Language (XML) messages that use SOAP 1.2 as the message format and protocol for message exchange. As ESB may also support non-SOAP interaction, specifically REST-style messages encoded in XML (Representational State Transfer) or WebSphere's MQ.
- *Service Definition*: An ESB must support the use of WSDL 2.0 to describe Service interfaces, with XML Schema being used as the way of representing the format of data used in XML-based message exchanges.
- *Identity or Location*: A definition of the addressing and the message header information is required in a message exchange. An ESB must support the use of WS-Addressing "endpoint references" to convey addressing information of the service requesters and service providers. Universal Description, Discovery and Integration (UDDI) is also a standard for platform-independent, XML-based registries (brokers) by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications.
- *Quality of Service*: This enriches a standard message exchange with extended aspects of requester-supplier interaction. Supported standards should be WS-Security, WS-Agreement, WS-ReliableMessaging and WS-Trust.

6.1.2 Services – Resources Layer

As we explained in Section 4.1.1, a SOA modularize large applications into services. These services can be found and exposed for use in a number of information systems, including ERP, CRM, DBMS, etc. The wrapping of these IS can be achieved with the use of (resource) adapters which provide connectivity, semantic disambiguation and translation services between applications and collaborations [35]. They encapsulate legacy code and logic and define services of an IS in a wrapper Application Programming Interface (API). Application-specific adapters, like Oracle DB adapters, provide a basic intermediary between the application and the ESB by translating application's messages to and from a common set of standards [30]. In that way, interoperability of legacy systems is enabled.

Applications including .NET, WebSphere and J2EE applications can be exposed for service compositions through appropriate service specifications and interfaces. Also, new web services can be implemented by the enterprise while others from external partners can be invoked as well.

6.1.3 Business Processes Layer

The specification of the control flow of a business process in which activities (steps) are specified as web services can be done by the widely accepted XML-based Business Process Execution Language (BPEL). BPEL describes the orchestration of web services which may be provided by more than one organizations. It establishes links to partners, defines the link types, declares the parameters to be sent and the results to be received, and invokes the web services. The services are described in WSDL and a SOAP binding is used for the transportation of the messages [36]. Processes and especially long-running ones are managed by a BPEL engine which is tied to the integration layer through WSDL interfaces.

6.1.4 Standard architecture model with technology embodiment

Combining all the above technologies from all three layers, we design the standard architecture model in Figure 31.

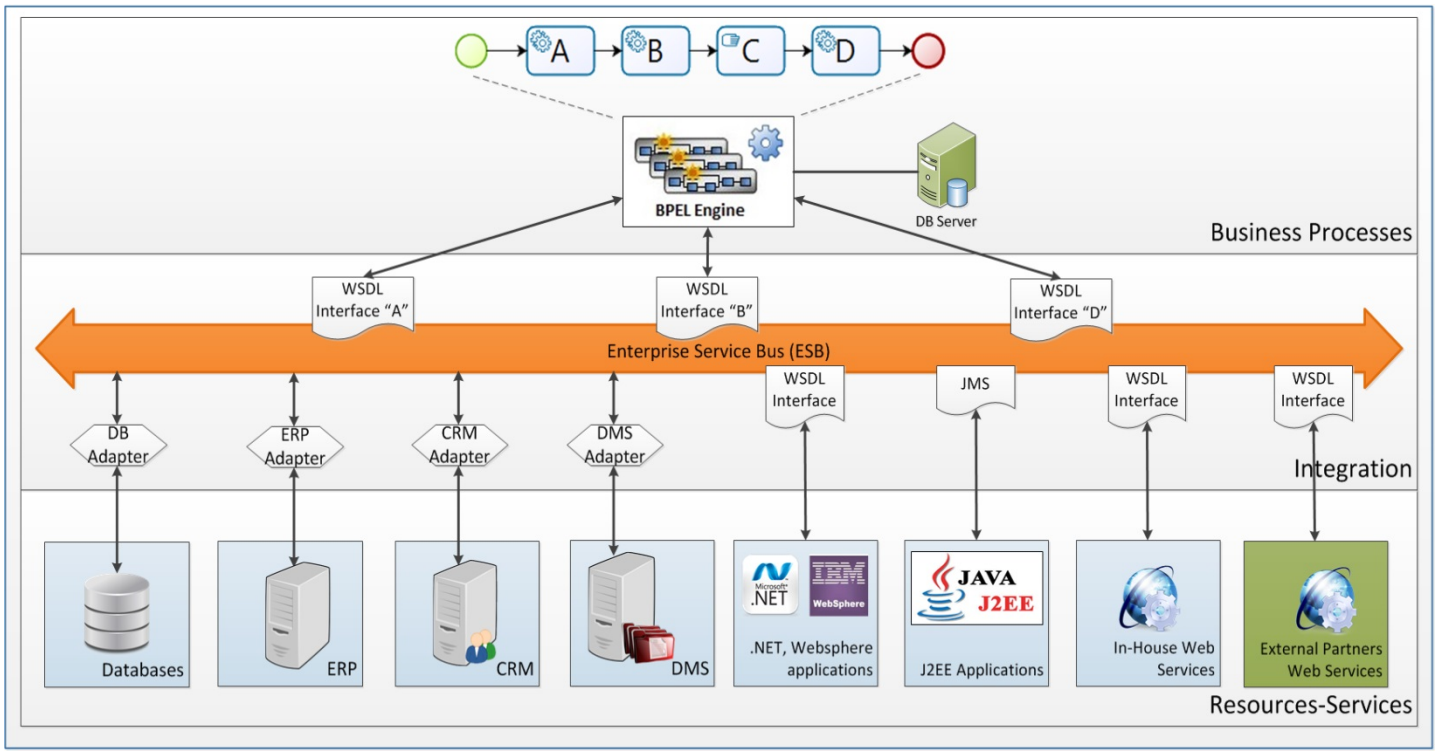


Figure 31 Standard architecture model with technology embodiment

In the above model we see that each service of the business process has its own interface to the ESB. This is how we perceive that each business service is described technically by a

WSDL interface. (In the example above, Service C is a manual service, therefore there is no WSDL interface for it).

The concept model of Figure 2 covers the Business pyramid of the BASE/X framework. We can extend that model by mapping concepts of the Information Systems pyramid to concepts of the Business pyramid. More specifically, we map the 1:1 relation between Business Services and WSDL Interfaces. The same goes for the specification of service compositions which can be done with the BPM Notation. The extended model is shown in Figure 32. Technical specifications of Business Models and Business strategy are out of scope of the current project.

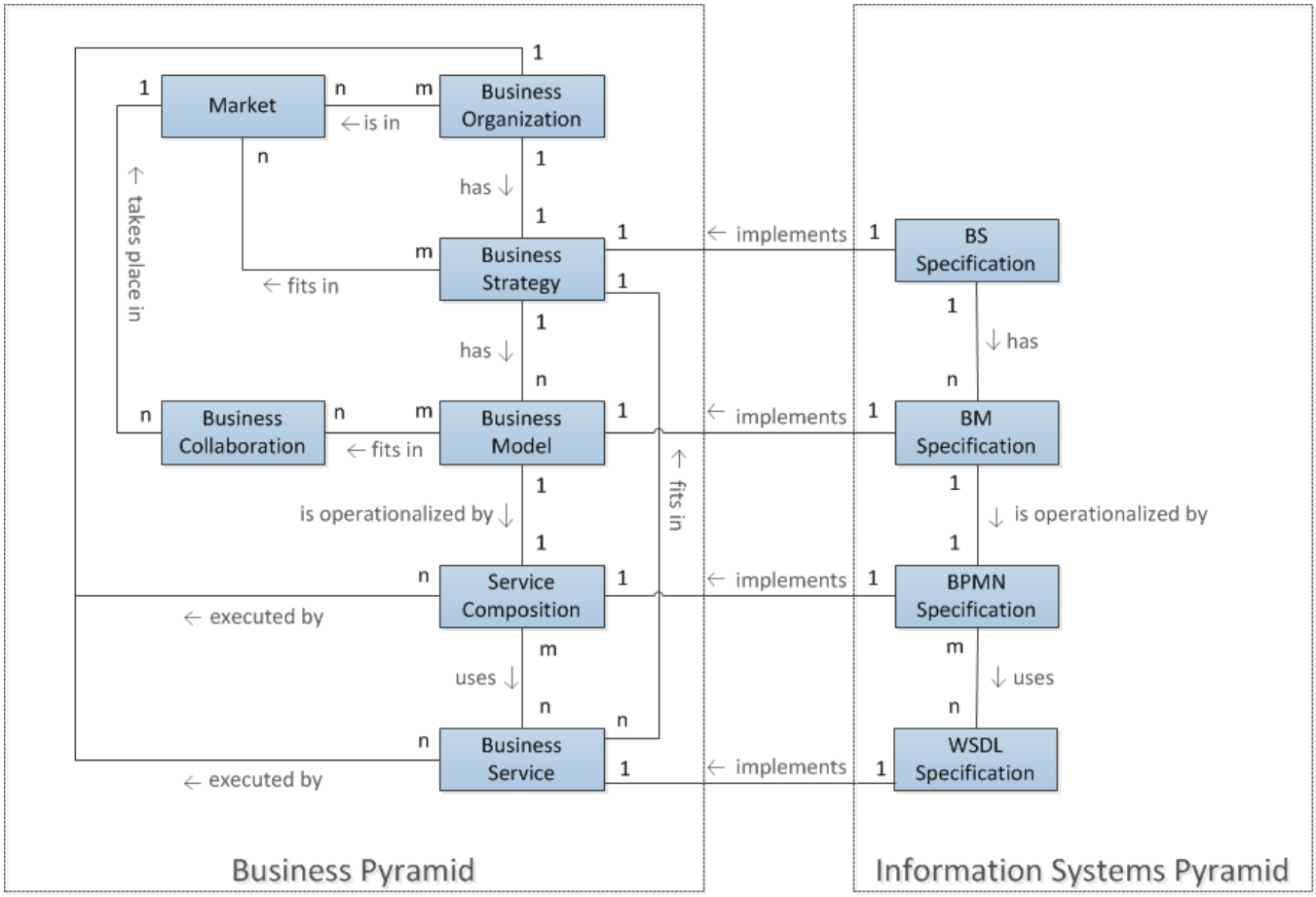


Figure 32 Extended concept model of BASE/X

6.2 Organization-Specific Architecture Model

Based on the previous model with the technology embodiment and taking into account the simplified application landscape of Athlon Car Lease as presented in Figure 17, we design the Standard Architecture model for service management. We present this organization-specific model in Figure 33.

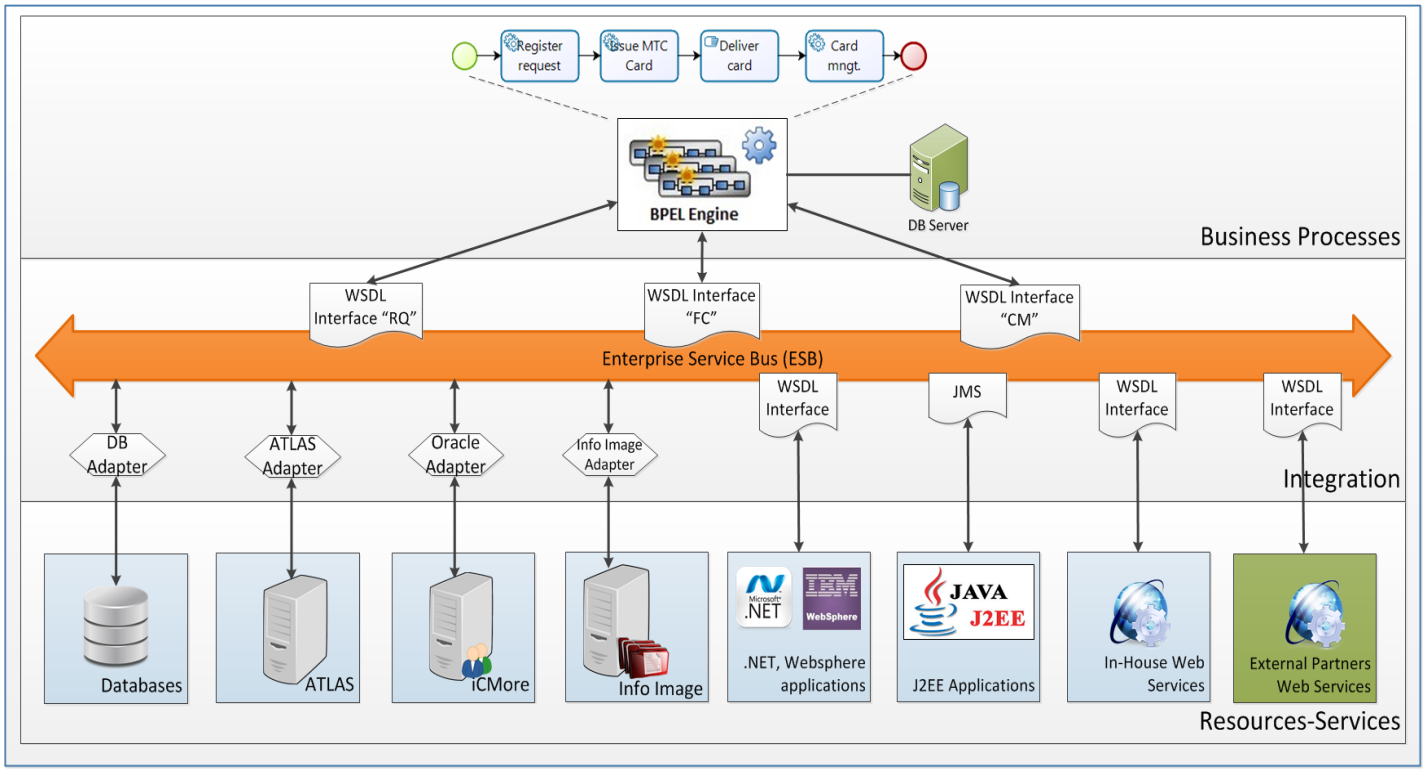


Figure 33 Standard architecture model for service management

Placing this architectural model in the three-dimensional design cube of Figure 9, we can say that we are still on the Technology level of the realization dimension, like in Figure 25. However, we have traversed the abstraction dimension since we present more technical and organization-specific details. Again, we keep the merge of the two aggregation levels we have used so far, since we do not make any technological distinction between business services and building block services. This is shown in Figure 34 below.

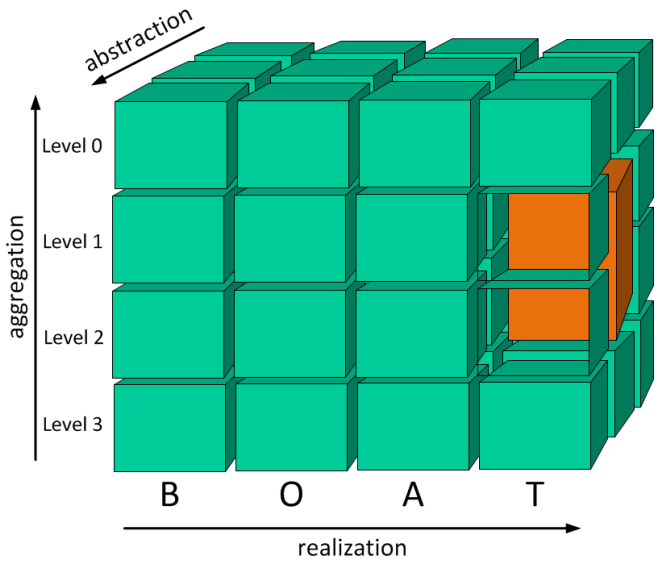


Figure 34 Positioning standard architecture model

With the model of Figure 33 we can say that we have completed the design of the Standard Architecture for service management. Next steps will be to design even more concrete

details including vendor type and version components. Such models will refer to the back bottom right cell of the three-dimensional design “cube”, as depicted in Figure 6.

Chapter 7

Implementation

In Chapter 2 we mentioned that PegaSystems is the BPM vendor that will facilitate our prototype. In Chapters 3, 4 and 6 we presented the main information systems that we use and how they should be "servitized" in order to develop the prototype. Also, in Chapter 5 we analysed the business model and the business services we are using for our prototyping purposes. In this chapter we have all the information to handle the implementation of the prototype of service compositions. We first discuss in Section 7.1 how the Dynamic Case Management (DCM) paradigm helps us to implement the Driver Desk use case. Then, in Section 7.2 we present the functionality, the data model, business process models and the user interface of the application. Finally, in Section 7.3 we evaluate the implemented prototype.

7.1 Dynamic Case Management

Case Management (or Case handling) is a paradigm for supporting flexible and knowledge-intensive business processes. Unlike workflow management, which uses predefined process control structures to determine what should be done during a workflow process, case management focuses on what can be done to achieve a business goal [37]. Case is the central notion, which can be seen as the coordination of multiple tasks (planned and unplanned) and associate content, towards a concrete objective. In case management, the knowledge worker in charge of a particular case is a cognitive worker who actively decides on how the goal of that case is reached, and the role of a case management system is assisting rather than guiding and restricting him in doing so.

The core features of case management are [37], [38]:

- avoid context tunnelling by providing all information available (i.e., present the case as a whole rather than showing just bits and pieces),
- decide which activities are enabled on the basis of the information available rather than the activities already executed,
- separate work distribution from authorization and allow for additional types of roles, not just the execute role,
- allow workers to view and add/modify data before or after the corresponding activities have been executed (e.g., information can be registered the moment it becomes available).

The term Dynamic refers to highly variable, unpredictable, loosely structured and subject to change cases and processes. It is related to flexibility and adaptability and the basic idea is to allow for changes at run-time, i.e. while work is being performed processes may be adapted [39].

In the sequel of this section we compare DCM to BPM and see how these two paradigms serve the role of service management in our prototype. Then, we describe the use case of Driver Desk with the help of DCM.

7.1.1 Comparison to BPM

Characteristics of each of the two approaches indicate their differences. While traditional workflow management focus on the complete definition and control of structured, repeated processes, case management works on an ad hoc basis to manage dynamic, unstructured processes. BPM is a process-driven discipline which routes processes through specific activities. On the other hand, DCM advances through events based on the case data, characterizing it as event-driven and data-driven. Also, DCM is better used in processes where many exceptions and deviations appear, since attempting to capture all of these scenarios with traditional BPM [40], results in complex models that are hard to manage and maintain.

Based on the Mintzberg's Five Organizational Structures [41], we can say that BPM is best applied in Machine bureaucracy structures, where the standardization of work processes is the prime coordinating mechanism, while DCM is most suitable for Professional bureaucracy organizations where standardization of skills is the dominant mechanism.

However, despite their differences, one approach can be found in the other. The lack of flexibility of workflow systems has been addressed by different researchers like with the Flexibility as a Service (FaaS) approach presented in [42]. Also, in case management systems, workflows are part of the associated content of a case.

In the current project, with the use of the tools offered by Pega we exploit the characteristics of both approaches. BPM is used for the automation of the existing standardized and optimized processes of Athlon, while DCM is used for the whole handling of cases that are associated with offered services.

7.1.2 Driver Desk as DCM use case

In Section 5.3 we stated that we implement the use case of Driver Desk as a hybrid type of service compositions. The customer-facing business process can be seen as a mash-up type where services are offered to the consumers who are free to opt, adhering though to any constraints related to these services. Driver Desk handles inquires which represent different cases. A tool with dynamic case management capabilities supports caseworkers to combine required knowledge, information and content in such a way that they either can solve the case or initiate the corresponding service(s), in compliance with rules, constraints and objectives.

The associated content refers to the documents that are required to handle a case and are the lease contract of the driver, fines, fuel transaction reports, damage reports/photos and maintenance reports. The objectives are set by the Driver Desk's manager and refer to the

quality of service. Indicators are percentage of resolved cases directly by caseworker, percentage of open cases per case type, average time to provide a solution.

The overview of Driver Desk is presented in Figure 35. A driver, through a number of available channels, contacts Driver Desk which handles his case with the help of a Dynamic Case Management dashboard. Based on the inquiry, Driver Desk initiates a number of supporting business services. Figure 35 is not an architectural model and should not be treated as such, especially not to be confused with any BASE/X layers, but serves as a picture to describe the high-level functionality of Driver Desk.

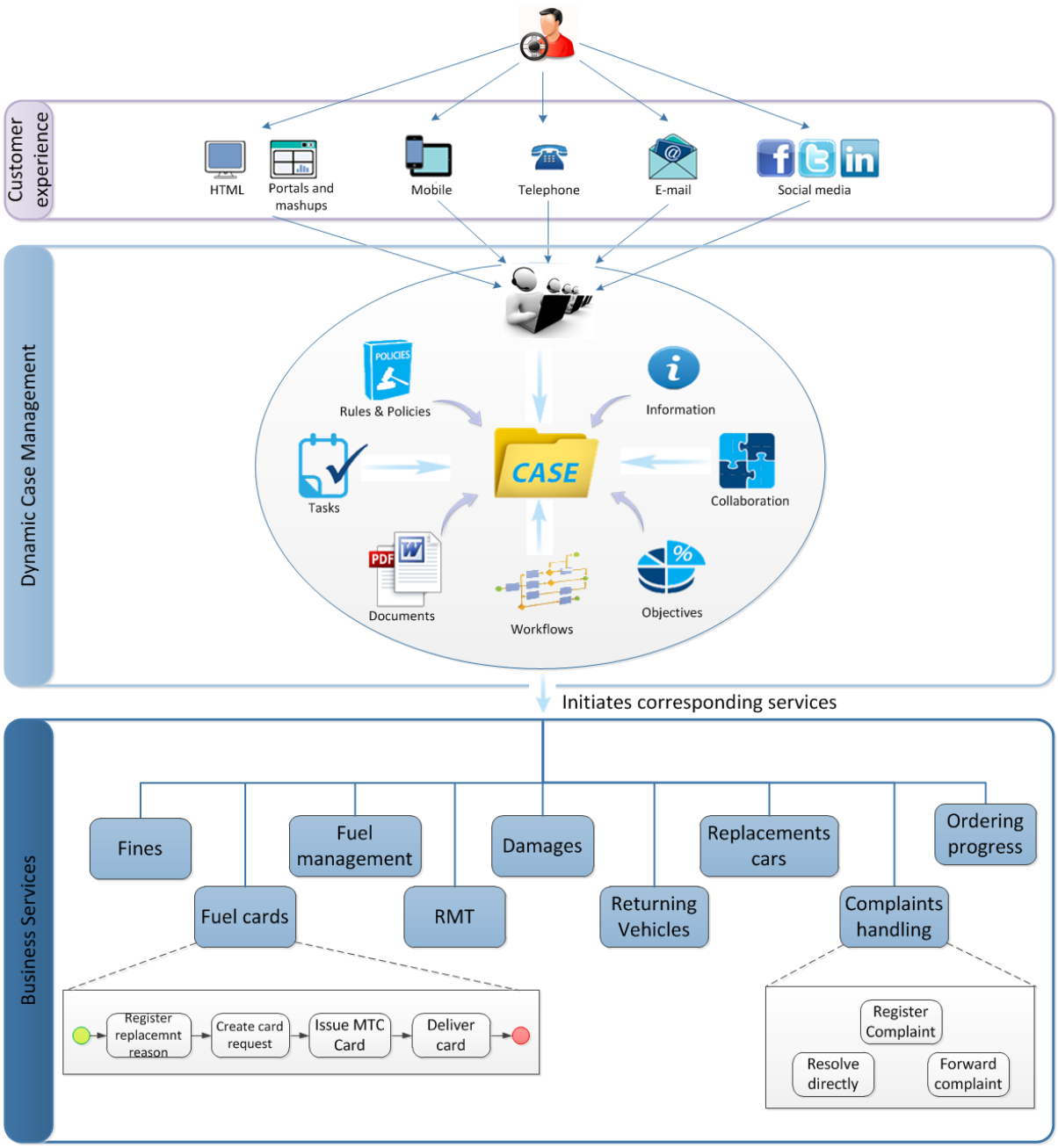


Figure 35 Driver Desk use case overview

7.2 PEGA Tool Prototype

This section discusses the implementation of the prototype, the design of which was presented in Chapter 5 and in the previous Section 7.1. We used the Pega 7[25] tool which is a single platform combining BPM and DCM capabilities. We used a cloud environment that the company offers. We first discuss in Section 7.2.1 the functionality of the prototype and the requirements it should meet. Then, we discuss the data model and the business process models in Section 7.2.2. Finally, in Section 7.2.3, we present the user interface of the prototype.

7.2.1 Functionality

Before describing what the application is able to do, we have to mention the groups of people that will use it:

- Driver Desk, consisting of team leaders and officers. The latter ones are those who interact with the customer.
- Service Department, consisting of managers and users. This department is a simplification for the purposes of the prototype for handling back-end processes.

The simplified organization structure is shown below in Figure 36.

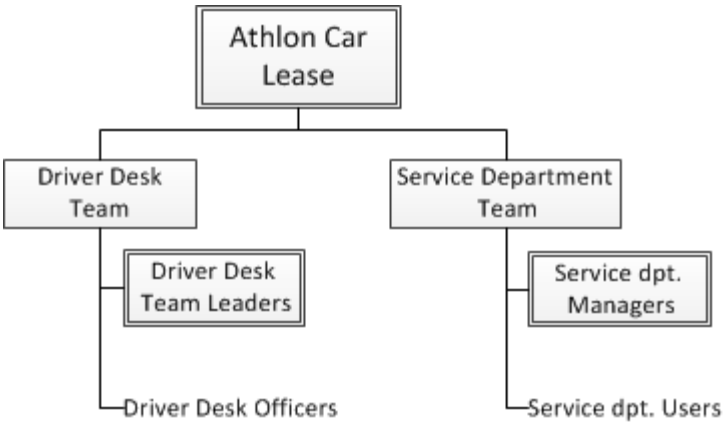


Figure 36 Organization Structure for use of the prototype

Each of these roles has its own functionality. However, since our main purpose is implementing an application for serving customers, we are focusing in this section on the functionality of the Driver Desk officers.

The application is a single dashboard where an officer logs in and handles requests from drivers. In that dashboard he accepts incoming calls (simulated though since an integration to an Interactive Voice Response (IVR) system is not in our scope), captures inquiries, views information about the driver, his contract and his vehicle and finally initiates a requested service. He can also re-open past cases or view open or pending requests. (A team leader, in addition to these tasks, can see statistics of requests (per case type, per status), generate

[25] http://www.pega.com/products/pega7, retrieved September 10th 2014.

reports, check and (re)assign cases to his team members, change business rules and decision tables).

In Table 3 below we present a representative list of the functional requirements that the prototype should meet, along with related specifications. The full list can be seen in Appendix C. This was our basis for the implementation of the application. However, since the goal of this section is to present the functionality that an end user sees, we do not include any technical requirement and specification.

Table 3 Representative list of functional requirements and specifications for the prototype

S/N	Requirement	Specification
1	Driver Desk shall provide the ability to be reached by customers/drivers through the following channels: phone/email/web form (social media are out of scope of the prototype).	In case of a phone call, the following driver's data will be required: - Vehicle's License Plate - Driver's Last Name - Company's Name
2	The dashboard shall provide all the right information to the caseworker related to the current case/inquiry/customer. These are driver's personal details, case details and information on how to handle a case.	The dashboard shall provide the following information for each case/inquiry/customer: - Case ID - Case Type - Enter Date - Case Description - Vehicle's License Plate - Contract No - Driver's Name/Telephone/Email/Address
3	The application shall be able to create a report with a list of fines.	The driver will request a report with a list of his fine tickets indicating the period. The report will contain the fine ticket type, the date and the amount.
4	The application shall provide services for (re-)issuing a fuel card or cancelling and existing one.	The driver will provide the following information: - Fuel Card No (if known) - Fuel Card Type - Vehicle's License Plate - Reason for replacement - Driver's Name
5	The application shall provide information on nearest gas stations (specific radius), given the location of the driver.	Given driver's zip code, the application shall return a list of gas stations in a specific radius with information of gas prices.
6	The application shall generate a report with the fuel transactions of a vehicle.	Give a vehicle's license plate and the fuel card number, the application shall generate a report with a list of transactions containing information on date of transaction, type of fuel, gas station name and address and amount.

The above requirements and their related specifications were entered in Pega 7 Designer tool to be taken into account for the actual implementation of the prototype.

However, before contacting Pega to start collaboration for the implementation of the prototype, we had to present them the aforementioned functionality and requirements. In order to do so, we captured the core functionality of the Driver Desk dashboard into a mock-up interface. This can be seen in Appendix D[26].

7.2.2 Data and Business Process Models

There are different types of data in workflow management systems, and consequently in business process management and case management systems. According to the Workflow Management Coalition (WfMC), we should distinguish three kinds of data in workflows: control data, workflow relevant data and application data [43]. Another classification can be found in [44], where data are categorized in application data, workflow schema data, historical data and internal data. To keep it simple, we can make the distinction of Process Flow Data (or Case Data) and Application Data. The first kind refer to data manipulated by the workflow (or BPM) system to keep information about process instances (and their states) and internal process variables. The latter data are application-specific and are managed by the applications supporting a process instance. Generally, Application Data are not accessible by the workflow management system and are managed by each application.

We focus here on the Application Data since Process Flow Data are not so relevant for our purposes. Data storing information of drivers, customers, vehicles can be found in Athlon database systems and the iCMore cloud system. Due to security reasons, real data should not be stored in the Pega application, so we had to access these systems to retrieve appropriate information. The first step was to query the on premise databases of Athlon to get information about drivers and vehicles. Based on identifiers of these records, we could access the cloud-based CRM system (iCMore) to get the right information. The complete database model is complex, thus we present below a simplified E-R model of the main data tables and their relationships to each other. The purpose of that model is to show information on what kind of data we are handling in the prototype and it is not mapped to any specific database.

The model is drawn in MS Office Visio 2010, using Crow's Foot notation[27] and is shown in Figure 37.

[26] A separate document with a complete documentation of the Driver Desk use case that we used as a basis to implement the prototype with Pega tools can be obtained upon request to DLL.
[27] http://www2.cs.uregina.ca/~bernatja/crowsfoot.html, retrieved September 10th 2014.

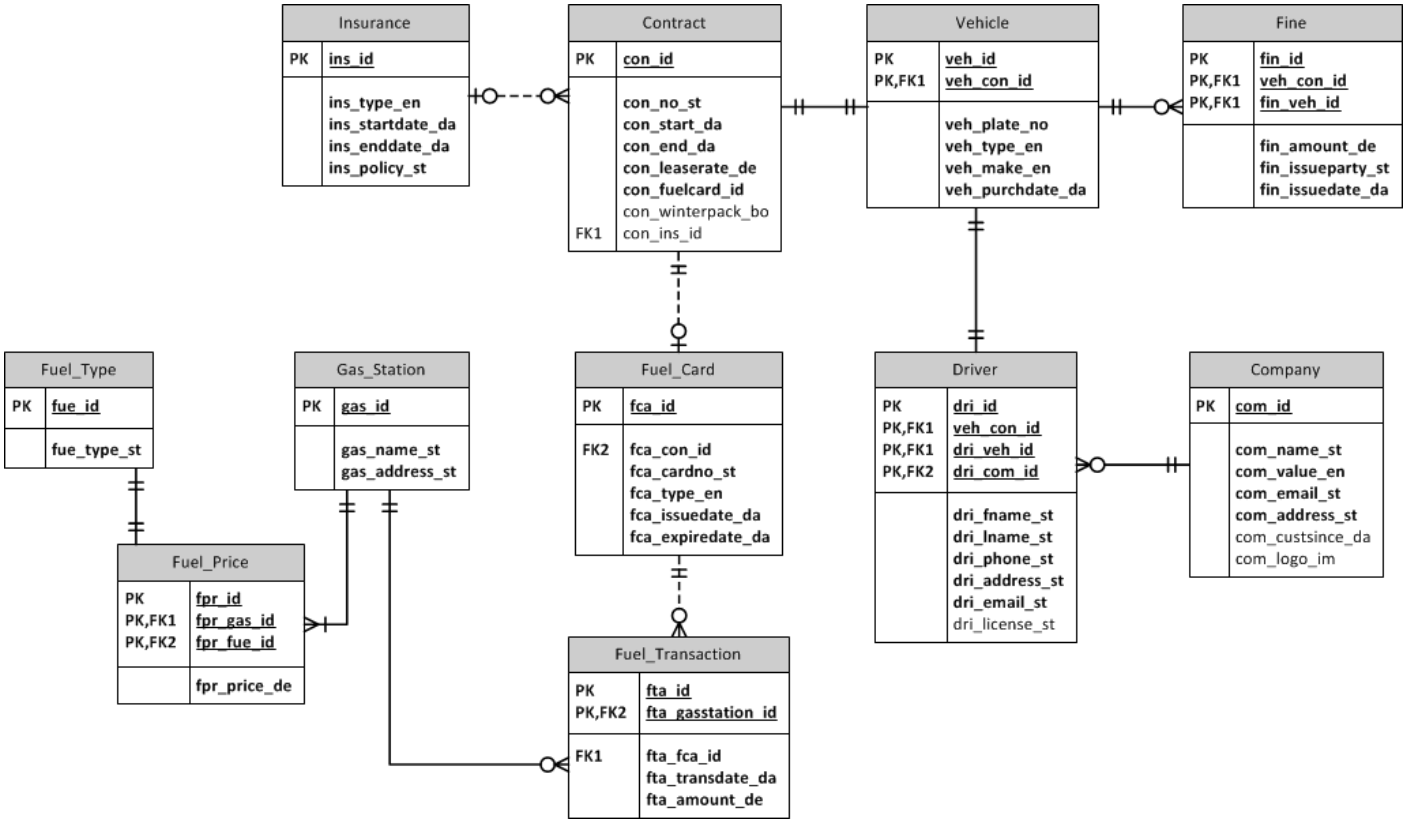


Figure 37 Application Data Model

Contract is a main object where information about lease rate and components like winter package and insurance are stored. Each vehicle is tied to a contract. Fuel cards are also related to a specific contract. Drivers, who belong to a company which is considered as a customer of Athlon, are assigned to a vehicle. Information on gas stations is also available.

Regarding now the business process modeling, Pega 7 uses a stage-based approach to first define a high-level overview of the whole process. For each stage, steps are defined which in turn need to be specified into more detailed business process model. The approach also allows the definition of screens and sub-cases. An example of a stage definition is presented in Figure 38.

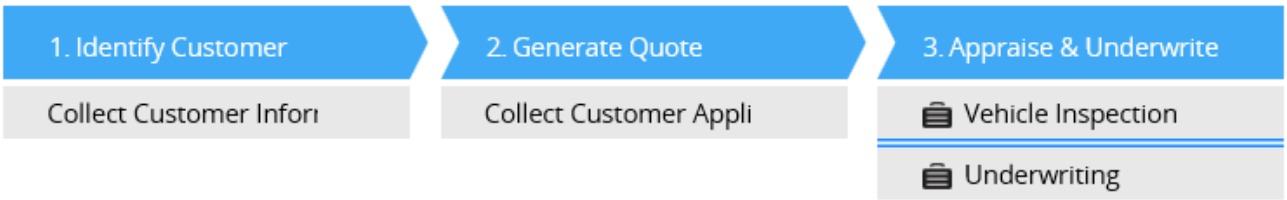


Figure 38 Stage-based process definition

In our prototype however, processes are not that long (after the request details, we execute a service where single screens are sufficient). Therefore, we did not apply the logic of stages but we designed the process models directly.

Below we present three of the internal business processes related to corresponding services. The first one is the process of re-issuing a fuel card. The flow is rather straight-forward. We

first have to cancel the existing fuel card and then issue a new one. Note here the existence of the "Cancel" sub-process which can be used also independently to run a card cancellation case. This flow is presented in Figure 39.

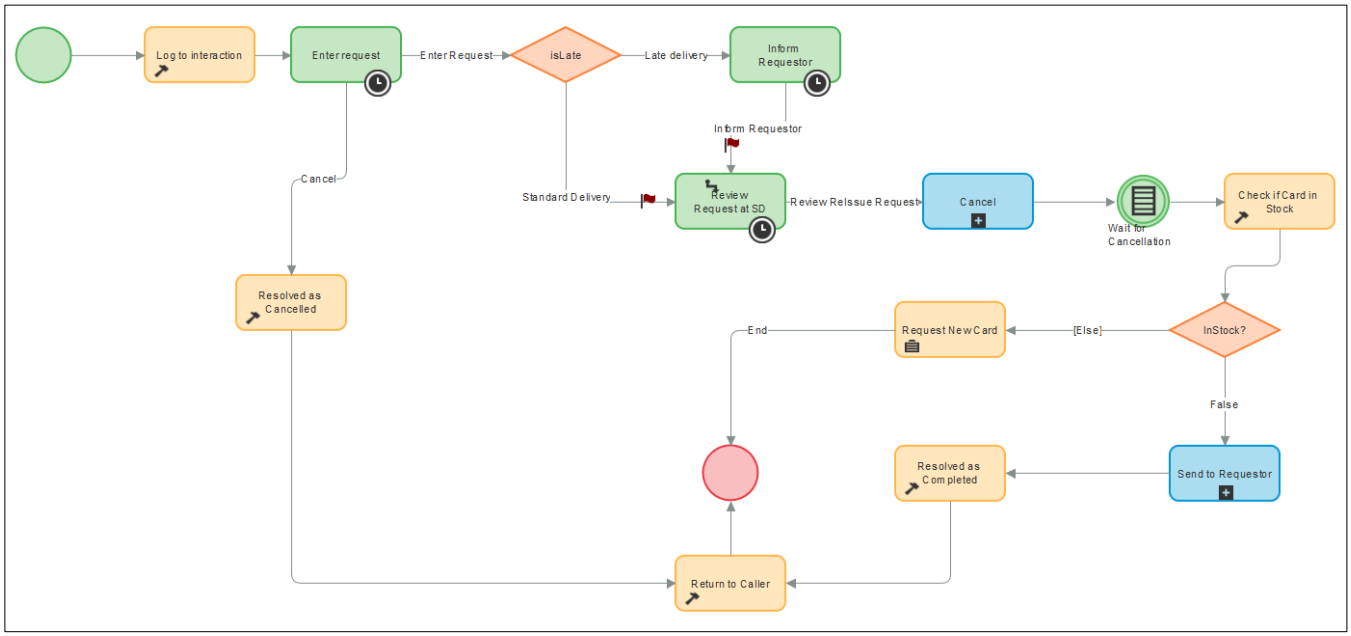


Figure 39 Re-Issue Fuel Card process flow

Some steps, like "Log to Interaction" and "Return to Caller", are pre-built steps necessary for the initiation and execution of a corresponding case.

The next flow is the one referring to the request of information on fuel card transactions. An external service call to MTC organization is implemented to fetch a list of transactions. The flow is shown in Figure 40.

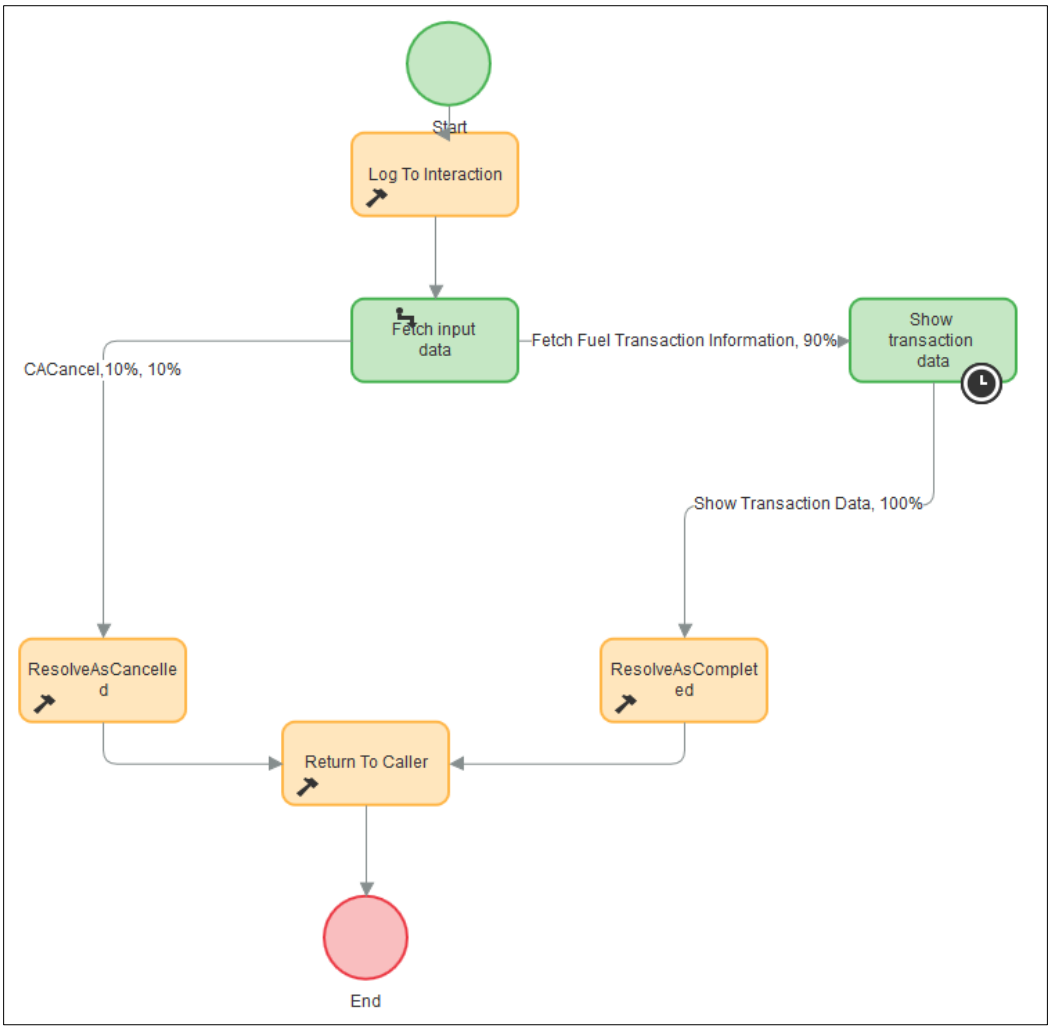


Figure 40 Fuel Transactions Info process flow

One more process that is implemented in the current prototype is the one related to the service of "Information of nearest gas stations". The process requires the zip code of the driver. The first step is to get the coordinates based on that zip code and then fetches a list of gas stations with information on fuel types and prices. An extra step is added to attach that information to an e-mail to be sent to the driver. The flow is shown in Figure 41.

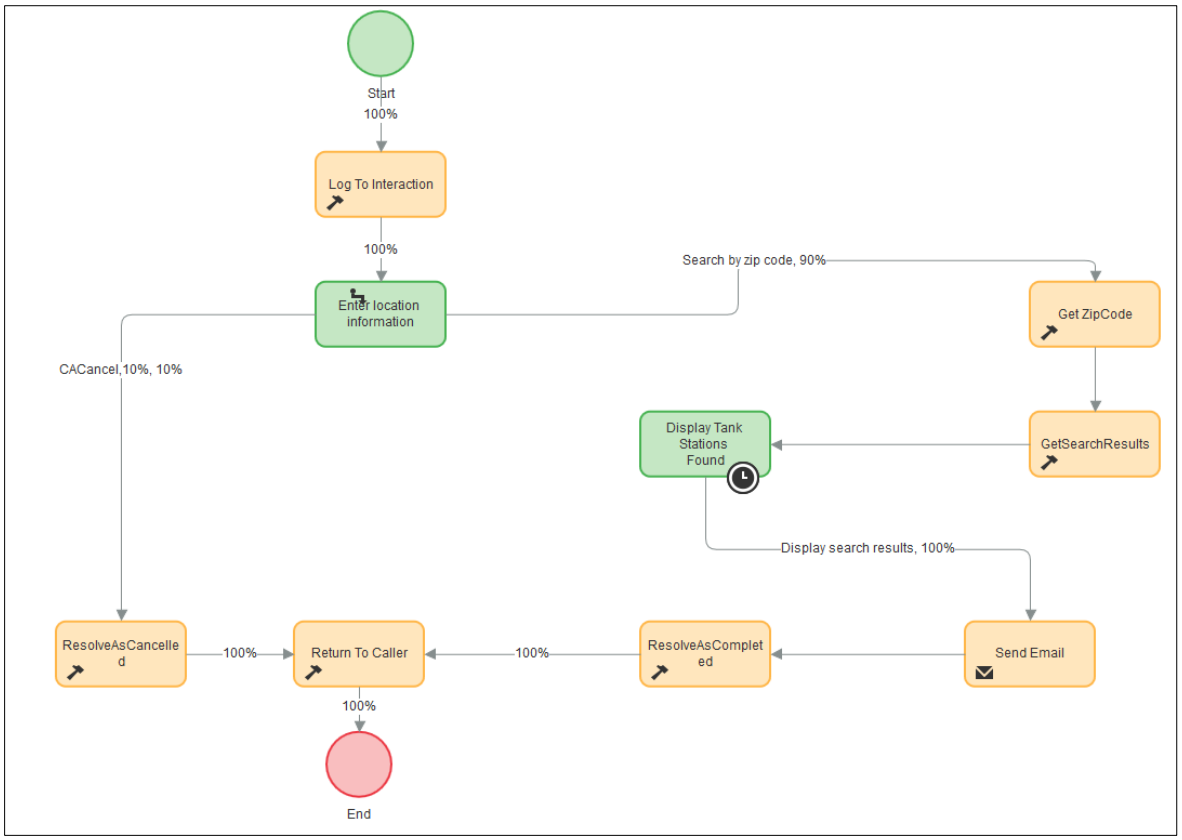


Figure 41 Locate Gas Stations process flow

7.2.3 User Interface

In this section we present the user interface of the implemented prototype through the description of two use case scenarios.

Driver Desk officer scenario

First, we present below the screens that a Driver Desk officer sees when she executes her tasks and serves a driver. After she logs-in to the dashboard, she sees the following screen of Figure 42.

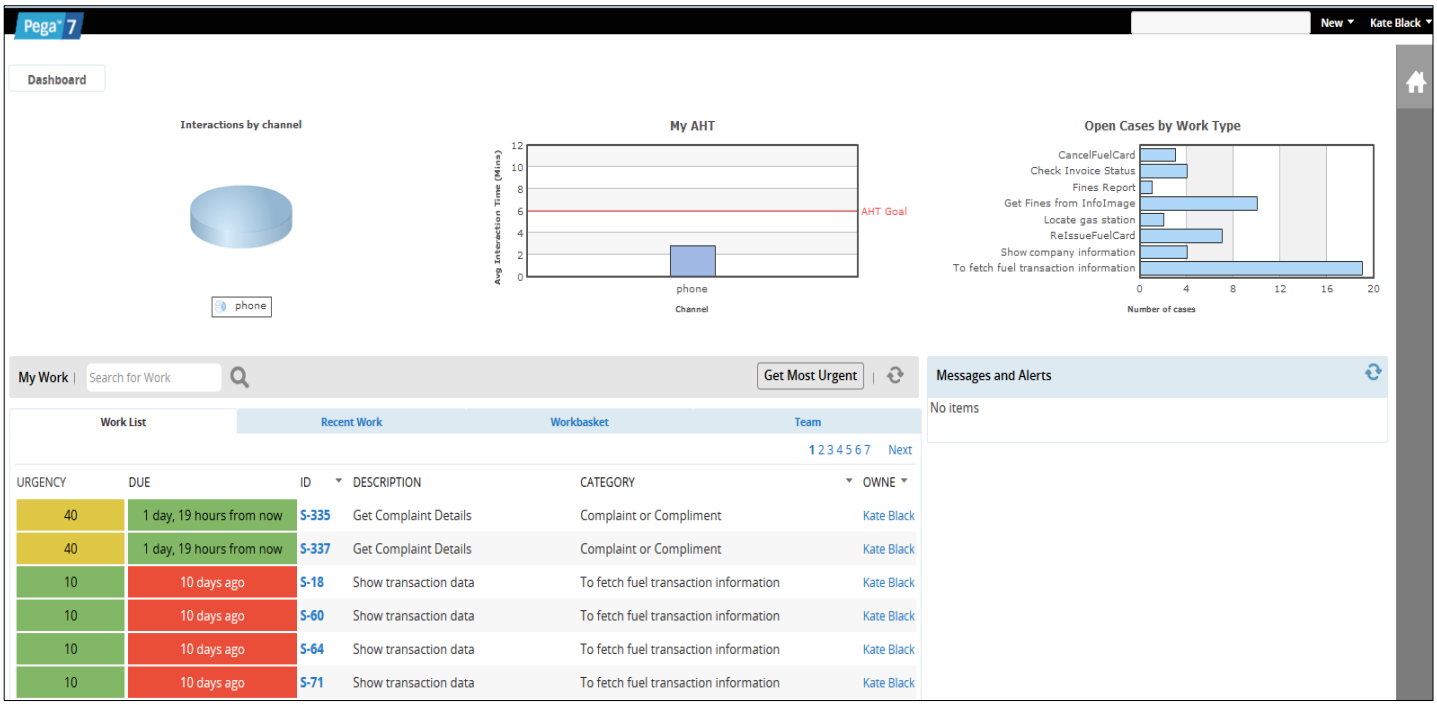


Figure 42 Initial dashboard screen - Driver Desk Officer

In that screen she can view a list of cases and their urgency. Statistics show individual objectives for her workload. In case of an incoming call (we have simulated that scenario by clicking New > Phone Call on the up right corner of the dashboard), she gets a screen where she can look up the driver after asking his details. This screen is shown in Figure 43.

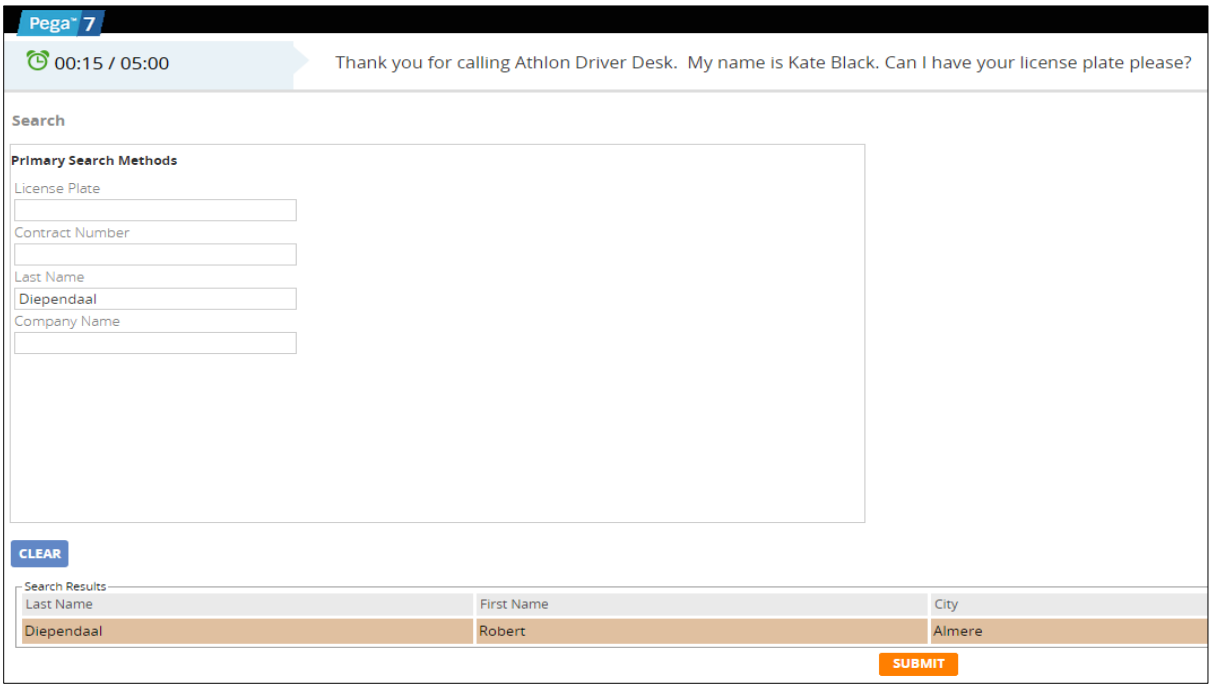


Figure 43 Incoming call screen

By entering vehicle's license plate or driver's name and/or company name, she retrieves his contact. After selecting that, she gets the main screen of the dashboard with all the information she needs to serve that driver. This is presented in Figure 44 below.

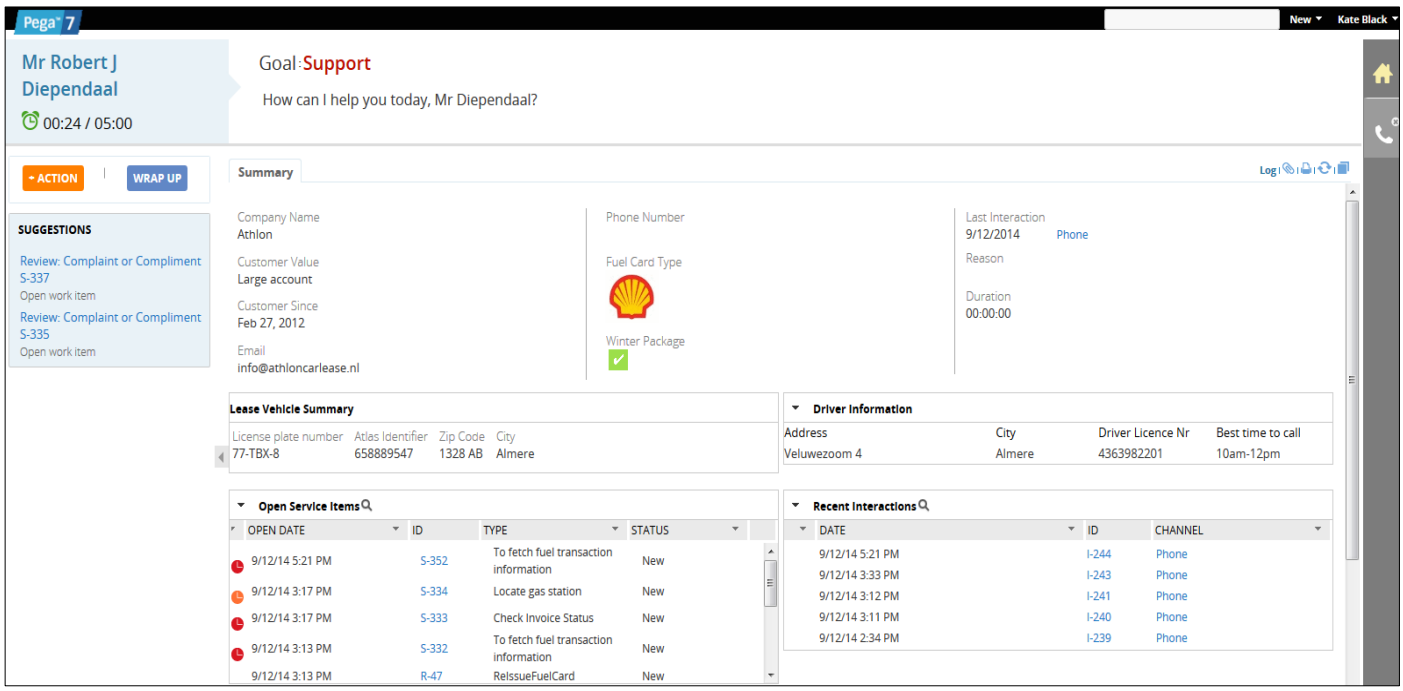


Figure 44 Main screen for serving a driver

Information about driver's details, company's details in which he belongs, vehicle details and contract information are presented. A list with open cases for that specific driver and a list with past interactions are also available in order to allow the Driver Desk officer to serve the driver as efficiently as possible. After reviewing driver's data and getting his inquiry, a list of actions are available, corresponding to the services that Driver Desk can offer. This is available on the up left part of the dashboard and can be seen in Figure 45.

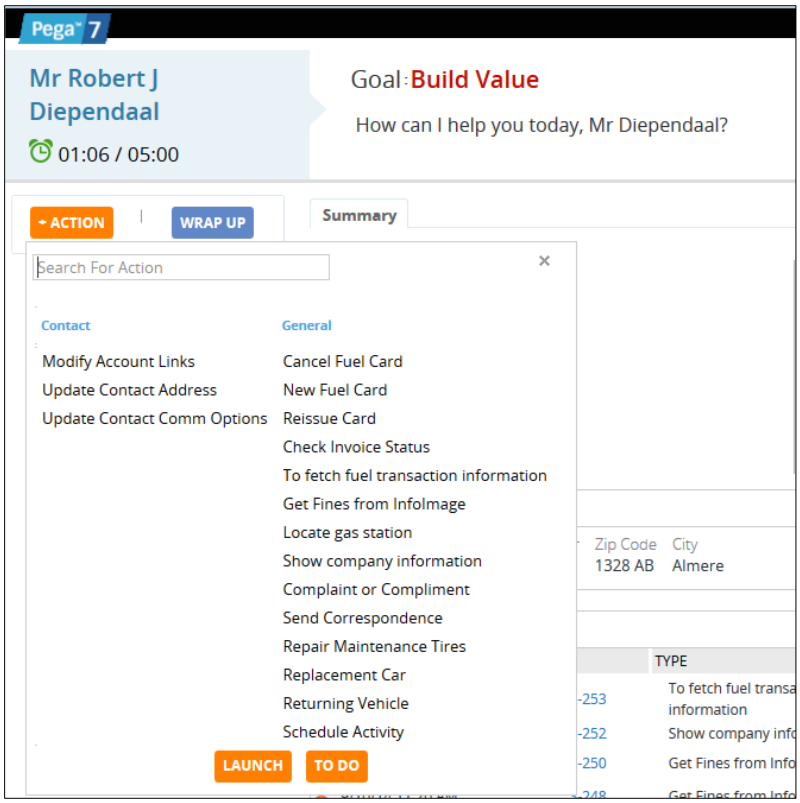


Figure 45 List of supported actions - services

Assuming that the driver requests a list of his fine tickets, the officer launches the service which makes a service call to the Document Management System and gets the screen of Figure 46. In that screen, by clicking "Show" she sees the related .pdf file and by clicking "Submit" the file is attached to an email and is sent to the driver.

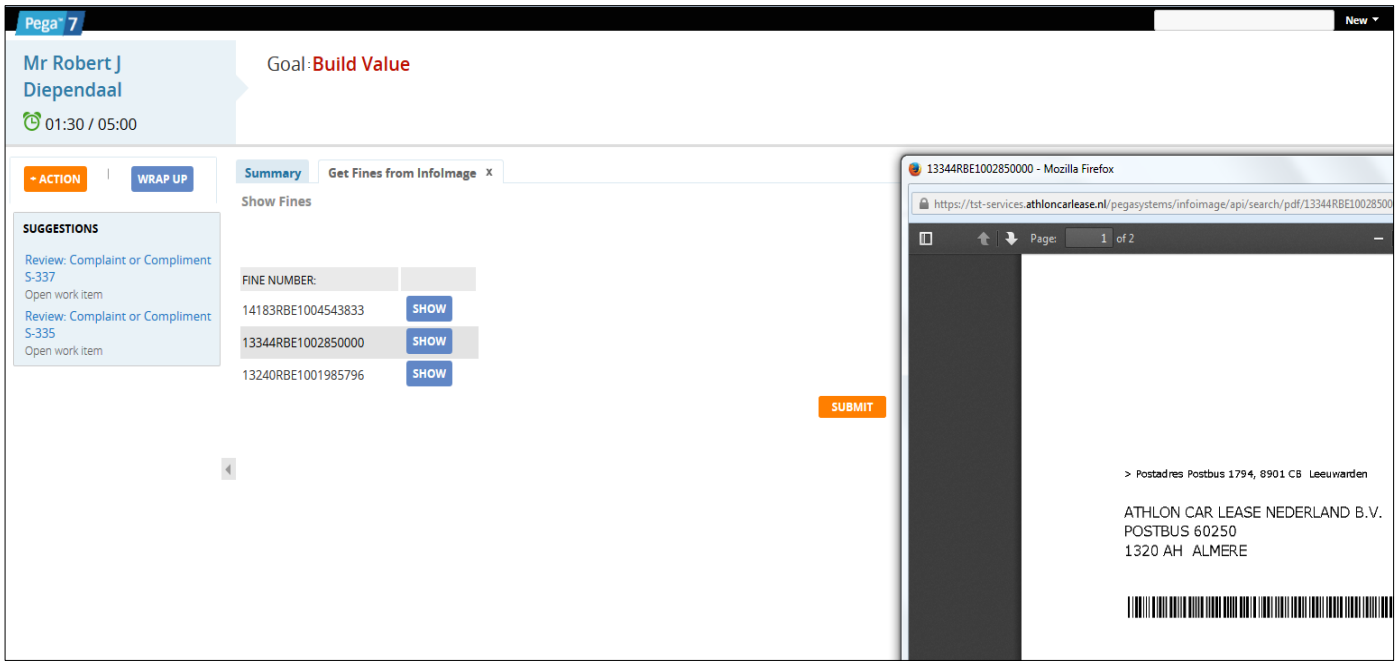


Figure 46 Fine Tickets Results

After this inquiry is resolved, the driver is still on the phone requesting a new service. For example, he wants to know a list of fuel transactions on a predefined period. The Driver Desk officer gets the screen of Figure 47 where she enters the dates (license plate is already prefilled).

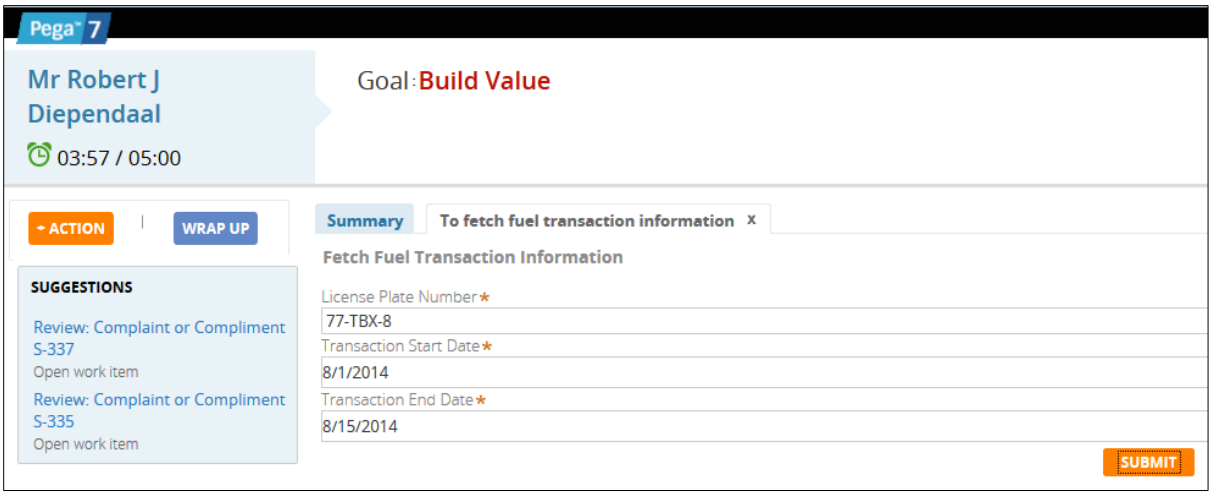


Figure 47 Fuel Transactions input screen

The service makes a direct call to MTC, an external partner which provides that information. The results are shown in Figure 48.

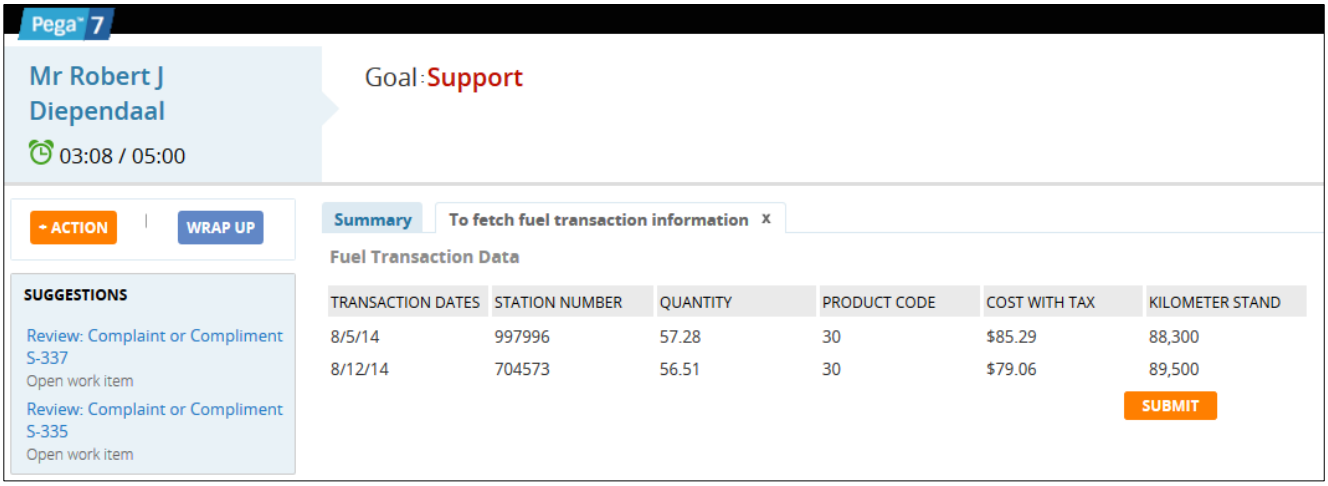


Figure 48 Fuel Transactions results

When the driver completes his inquiries and the call is about to end, the driver desk officer can ask his feedback on the offered services. A "Wrap Up" functionality of the dashboard can capture driver's satisfaction in order to be taken into account for future interactions with the same driver, improving customer intimacy. This can be seen in Figure 49.

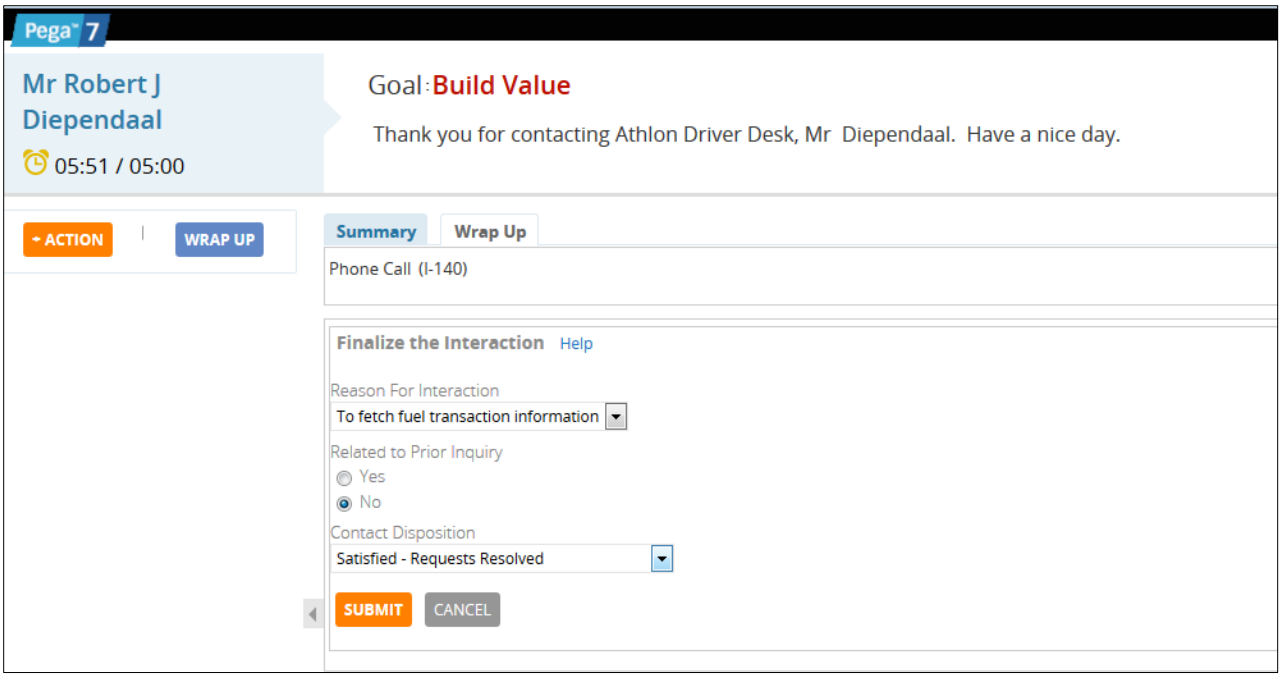


Figure 49 Wrap-up of a call, capturing driver's feedback

Driver Desk Team Leader scenario

In case of a Driver Desk Team Leader, the screens he can see are the same as an officer plus the ones where he can manage his team, review statistics and reports and change business rules of the processes. Below we assume the scenario in which a team leader desires to change some Service Level Agreement (SLA) values in a decision table.

After he logs-in to the system, he sees a screen like the one in Figure 42 but now data about statistics of the cases for the whole team are presented. Also, there are new tabs for Manager Tools and Report Browser. This is shown in Figure 50.

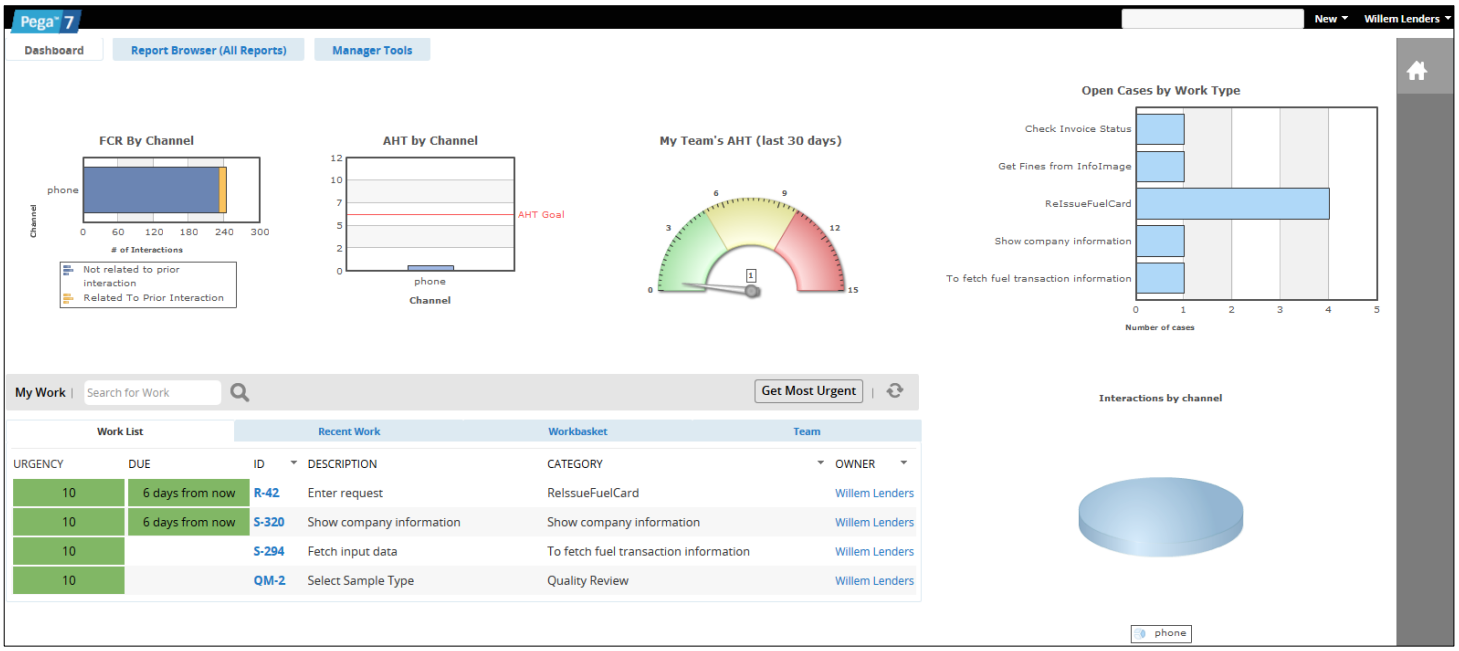
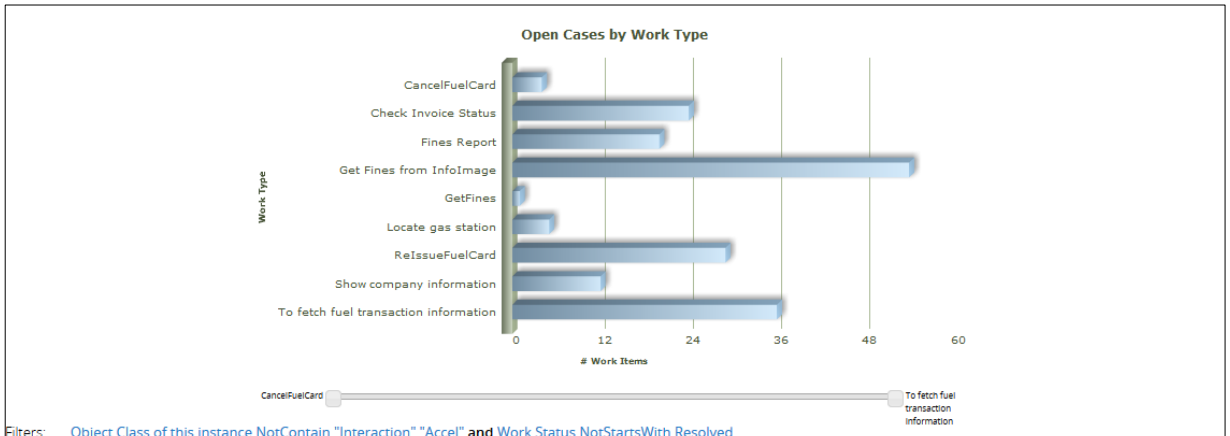


Figure 50 Initial dashboard screen - Driver Desk Team Leader

An example of a report can be seen in Figure 51 below. Here, information per case type is presented, giving insights on which processes are mostly asked and may require adaptations.



WORK TYPE	# WORK ITEMS CREATED	WORK ID
⊟ ALL	185	185
CancelFuelCard	4	4
Check Invoice Status	24	24
Fines Report	20	20
Get Fines from InfoImage	54	54
GetFines	1	1
Locate gas station	5	5
ReIssueFuelCard	29	29
Show company information	12	12

Figure 51 Report – Open Case by Work Type

Assume now that the Team Leader has to change a value, based on a new SLA. In his menu he can see a list of business rules he can changes. The scenario here is based on the fact the delivery time of reissuing a new fuel card varies with the card providers. If a card is an MTC card, then a standard delivery time of maximum 3 business days is required. In cases of

other fuel cards (e.g. Shell, Texaco) more than 3 days are needed to deliver the card and the driver has to be informed (The decision element in the process of Figure 39 shows that business rule). This can be shown in the decision table of Figure 52.

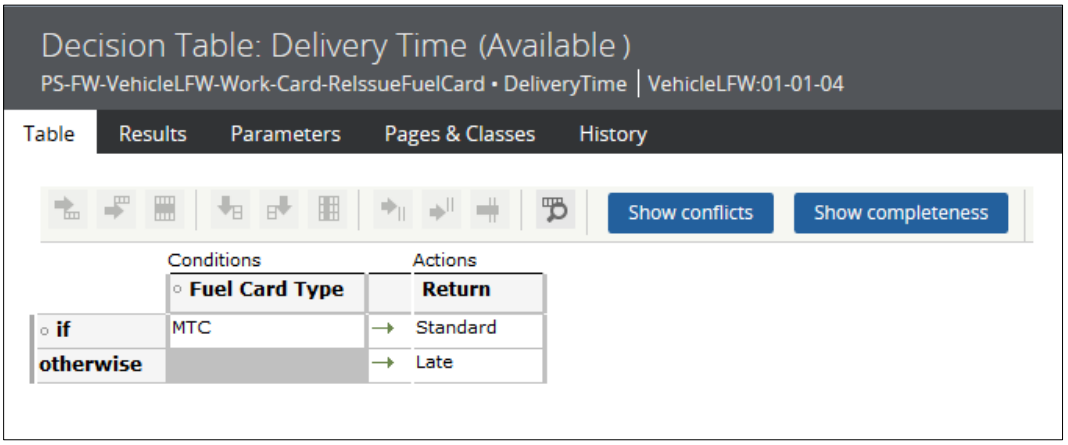


Figure 52 Decision Table

In the case in which processes in Shell have resulted in a standard delivery, the Team Leader can now edit that table by himself. The new decision table can be seen in Figure 53.

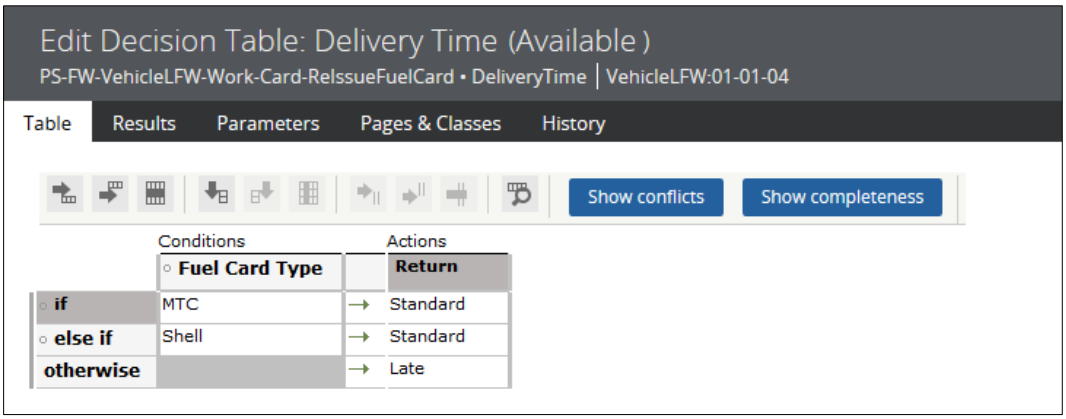


Figure 53 Decision Table changed by Team Leader

Changes are applied immediately on run-time, without any need for re-deployment of the application.

7.3 Evaluation

In this section we present the evaluation of the prototype implemented in Pega tooling. The aim is to assess the perception and the opinion of users who experienced the functionality of it. In Section 7.3.1 we discuss the evaluation of some quality attributes of the prototype. In Section 7.3.2 we assess Pega 7, whether it matches with the requirements we had set for a business process management tool in Section 2.1 and also how it fits to the architectural components of BASE/X.

7.3.1 Prototype evaluation

The evaluation of our prototype is done based on a model for assessing the quality of software, the ISO 9126-1 Software Quality Model [45], [46]. Six software quality characteristics are considered in this model, namely Functionality, Reliability, Usability, Efficiency, Maintainability, Portability. Each of these characteristic can be evaluated by a number of aspects. In Figure 54 below we see an overview of ISO 9126-1 Software Quality Model. The characteristics and aspects with grey colour are the ones considered in the evaluation of our prototype. Basically, this selection takes into account the fact that the implemented application is a simple prototype and not a commercial product. The goal of the prototype is to give people insights on service compositions and aspects like suitability and understandability are the most relevant in terms of assessing this Proof of Concept.

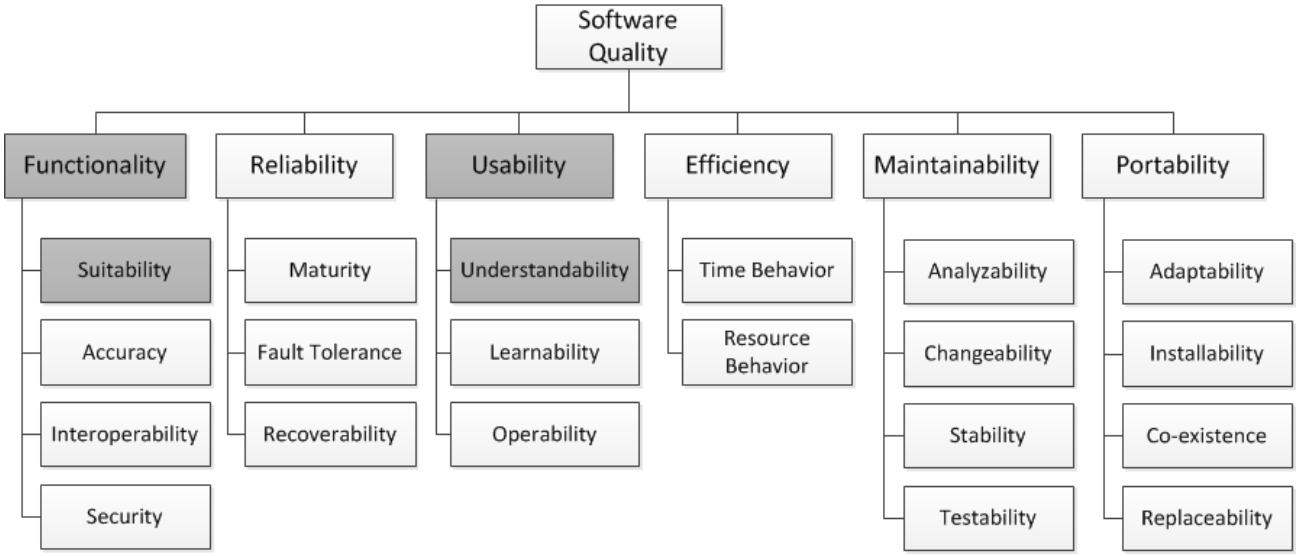


Figure 54 Overview of the ISO 9126-1 Software Quality Model, adapted from [47]

Functionality determines the capability of the application to provide functions which satisfy stated and implied needs when it is used under specified conditions (what the software does to fulfil needs). One aspect to evaluate functionality is:

> ***Suitability:*** the capability of providing the appropriate functionality for the required tasks.

Usability determines the capability of the application to be understood, learned and used by a stated or implied group of users (the effort needed for use). One of the aspects to evaluate usability is:

> ***Understandability:*** the capability of enabling users to understand the functionality provided in the application.

The approach we were planning to follow in order to assess the above aspects involved a workshop in which we would explain to people from DLL and Athlon (two enterprise architects, two Driver Desk team leaders and two Driver Desk users) how the prototype works and let them experience the application based on the predefined use case scenarios we saw in Section 7.2.3. We would then gather their feedback through questionnaires. These questionnaires had already been prepared but it proved to be rather difficult to organize

such a workshop due to limited availability of people. However, we had a few discussions with a Driver Desk team leader and enterprise architects about the suitability aspect. Regarding the understandability aspect, we present here the questionnaire and how can this be interpreted into meaningful results, leaving the gathering of feedback for future work.

7.3.1.1 Suitability aspect

In order to assess the suitability of the prototype, the following questions were posed to a team leader of Driver Desk, who has specific needs and requirements of such a tool (also, an enterprise architect replied to the same questions from a business perspective):

- Does the application provide a single point of contact for drivers to solve their inquiries about leased vehicles quickly and efficiently?
- Does the application provide the right functionality to execute all tasks and services that Driver Desk is responsible for?
- Is the application capable of providing insights on processes that fail to meet drivers' needs?
- Does the application allow performing of changes on business rules and decisions easily?

All the answers per interviewee can be seen in Appendix E.1. The average results are shown in the graph of Figure 55 below. The solid line represents how positive the answers are, regarding to a sub-question. In general, team leaders showed enthusiasm about the tool and believe that a fully production application with all the functionality of Driver Desk included will meet their expectations of solving drivers' inquiries efficiently and effectively. Changes on business logic is a point of discussion since the interviewees think that it requires some training on the tool for performing any of them. We believe however that this is a matter of skills in order to exploit the capabilities of the application and the tool itself to provide flexibility on changes.

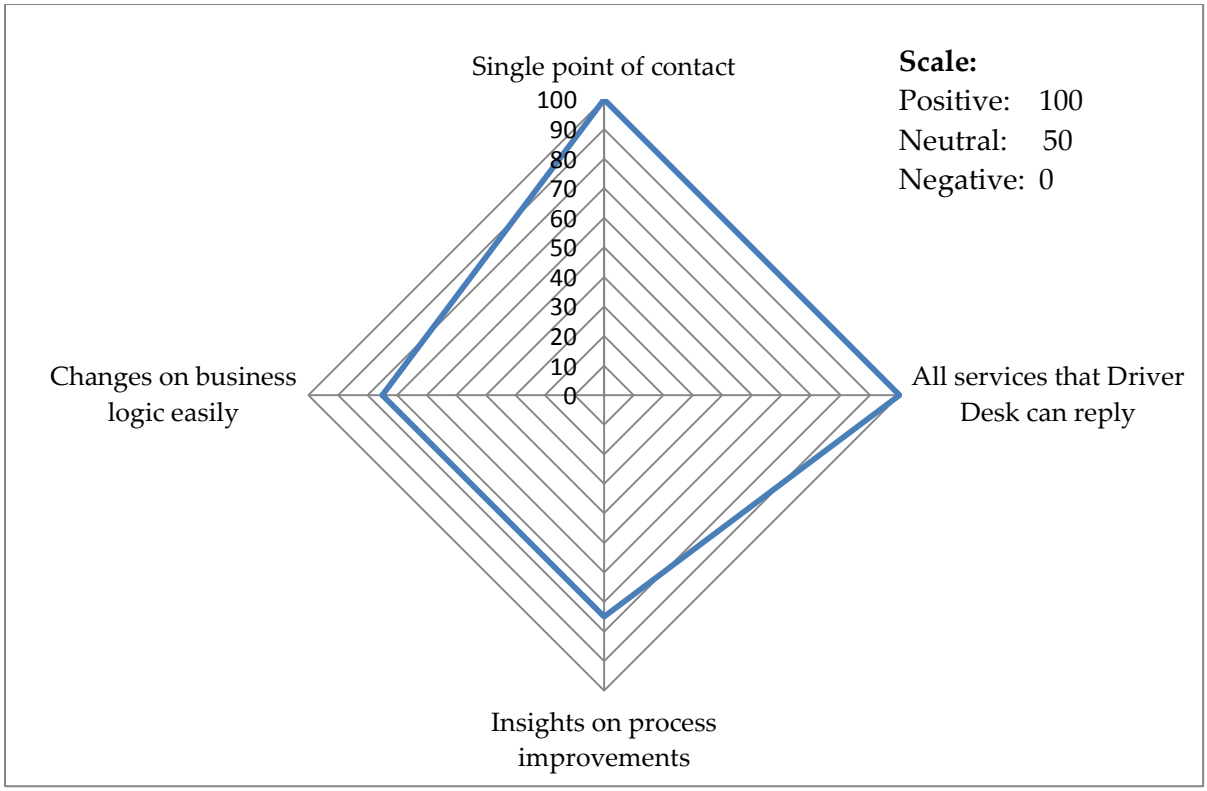


Figure 55 Feedback on suitability of prototype by team leaders.

7.3.1.2 Understandability aspect

In order to assess the understandability of the prototype, we can use the IBM Computer Usability Questionnaire [48] (we consider though a small representative subset of the nineteen questions that are there, relative to the understandability aspect). A typical question is "Overall, I am satisfied with how easy it is to use this system", where the interviewee has to specify his agreement or disagreement in a 7-point scale, from "Strongly Agree" to "Strongly Disagree". The complete questionnaire that can be posed to Drive Desk officers can be seen in Appendix E.2.

As we explained previously, participants for answering that questionnaire were not available. However, we discuss here how to interpret any received feedback. The results of the questionnaires can be analysed with descriptive statistics. We can treat each question separately (ordinal data), so we can measure the variability of a response, summarized by the interquartile range (IQR)[28]. Central tendency of the responses summarized by the mode is also an option [49]. With these measures, we can get an impression of the results of each question.

[28] The IQR is defined as the range of the values extending from the 25th percentile (Q1) to the 75th percentile (Q2). By fist computing the median, i.e. 50th percentile (Q2), a box with the IQR can be represented in a plot. The interval [Q1 - 1.5·IQR; Q3 + 1.5·IQR] can also be plotted to indicate any possible outliers (values outside that interval).

7.3.2 PEGA 7 tool evaluation

In Chapter 2, in which we discussed about BPM tools, we had set a list of high level requirements that an appropriate tool should meet in our effort to orchestrate business processes and provide new service compositions. Now that we implemented a prototype to prove these concepts, we can evaluate whether Pega 7 is a good choice for our purposes.

Through the course of the implementation and after discussions with experts from Pega, we can comment on which of the high level requirements of Table 1 are met. The ones we had marked with a "+" (based on market analysis) are present in Pega 7, as we experienced through the implementation of the prototype. Therefore, we focus below on the requirements that we had indicated with a "-" or "+/-" mark.

Model-driven composition environment: Pega's technology has been entirely designed to replace traditional way of development through coding in Cobol or Java by model-driven development. This is done with the Direct Capturing Objective (DCO) capabilities where business people can design case stages and business rules which help them understand the process models. Thus, Pega 7 platform replaces the more technical and cumbersome workflow design with the stage-based case design, leveraging the model-driven development.

Import process descriptions from other tools (e.g. ARIS, MS Visio): Pega has in the past provided a way to import models from ARIS tool but there was no meaningful usage of these models. The reason is that, even after importing a process description successfully into Pega environment, a step-by-step configuration is still required. That is because process descriptions in tools like ARIS are designed in a high-level, containing no executable logic, user interfaces, decision rules, etc. Moreover, workflows from other tools may be difficult to understand and maintain, making the import to a BPM tool useless. Also, since Pega uses the concepts of cases and subcases for inheritance and reusability, it is difficult to find that logic in any imported process models. All in all, Pega considers that the functionality of importing models from other tools has no business value. We can also agree on that since further configuration after importing models from a different tool will add extra effort.

Minimizing need of hard coding: In Pega they think that coding ends up in applications that are difficult to be maintained, thus their goal is to provide a revolutionary way to create applications without the need of coding. Coding should be used only in some user interface development. Capgemini[29] has performed a study concluding that Pega's way of designing applications is more than six times faster than using standard Java development tools[30]. However, in the implementation of the prototype we realized that the connection to legacy systems and external web services requires coding by technical people.

Comprehensibility and easiness of learning: Even though this is a subjective requirement to assess, we do believe that a steep learning curve is required to exploit all the capabilities of

[29] http://www.nl.capgemini.com/

[30] http://www.pega.com/sites/default/files/Productivity-Comparison-Pega-7-vs-Java-EE-FINAL.pdf, retrieved September 10th 2014.

Pega 7 platform. However, with extensive documentation and training material, Pega aims that even business people with no IT background can easily use their technologies.

Simulation capabilities: Pega's experts, with extensive experience in BPM tools, believe that simulation capabilities were a hype in the nineties, when simulation of a process in order to identify bottlenecks was seen as a nice way to implement the last step of Six Sigma[31] approach. However, nowadays, simulation is not an easy exercise and companies are even way busy to find time to do that exercise. Even standard simulation tools that were sold as Original Equipment Manufacturer (OEM) on top of BPM tools, were not actually used by customers. What Pega offers instead, is the ability to simulate changes in a real time environment, believing that this is the best way to adopt the Six Sigma concepts. With the use of rule-versioning concept, changes can be implemented in a production environment to be only visible initially by testers. Once changes are accepted, they can be easily promoted to all users.

The main conclusion regarding Pega 7 is that it is a rules-driven tool (everything in the tool can be considered as a rule) with an underlying architecture approach, called Situational Layer Cake[32]. BPM assets (business rules, process fragments, policies, integrations, UIs etc.) are organized into layers, with the most generic assets to be in the bottom layer and the specialized assets per situation are built on top of them (the specialization can be per line of business, customer segment, geographical location, etc.). This structured approach enhances reuse and customization, allowing for quick and agile application development.

The integrated platform offers capabilities for development of robust business applications with intuitive user interfaces. Changes can be applied on run-time without re-deployment, accelerating process adaptations. Extensive case management capabilities with a lot of features, model-driven design of process flows and predictive analytics render Pega 7 a powerful tool that enables an organization to deal with agility and dynamism. However, in order to reap all its potentialities it requires an investment of time to learn how to use it, both for developers and business people.

In addition to the above evaluation, in order to assess the suitability of the application built in Pega tool with respect to fit with BASE/X architectural components and architectural principles of the organization, the following questions were formulated to be posed to enterprise architects from DLL:

- Is the tool capable of hosting service compositions (mash-up or process type)?
- Is the tool capable of storing a business service repository where the business capabilities of the organization can be found?
- Can the tool integrate services from different ISs of the organization (internal) and external as well? Is the use of ESB supported/needed?
- Is the cloud environment able to support integration of systems in an agile way?
- Can the tool provide controlled access to services, adhering to security policies?

[31] Six Sigma is a leading methodology by which companies manage and improve their business processes. It was born out of work performed by the Motorola Government Electronics Group in the 1980's to apply statistical methods and models to analyze process defects and subsequently improve and control the processes.

[32] http://www.pega.com/resources/the-situational-layer-cake, retrieved September 10th 2014.

- Does the tool allow for a distinct separation of roles between business and IT people?
- Does the tool, both the design environment and the implemented application, allow for agile changes on business processes and quick implementations driven by business needs?

The above list takes into account main components of the BASE/X framework like the business service repository, service compositions, integration of legacy systems as appear in the 3-pyramid model of Figure 3 and its concepts in general, the Reference Architecture of BASE/X (Figure 7) and the Standard Architecture we designed in Chapter 6. It also derives from the agility that BASE/X aims to bring into organizations. Also, we should consider security aspects that the organization cares about and organizational aspects (separation of roles between IT and business people so as to know who should be responsible for what).

One Enterprise Architect from DLL answered the above questions, the complete answers of which can be seen in Appendix E.1. The results are shown in the graph of Figure 56 below (in case more enterprise architects had answered our questions, average results could be shown in a similar graph). The solid line represents how positive the answers are, regarding to a sub-question.

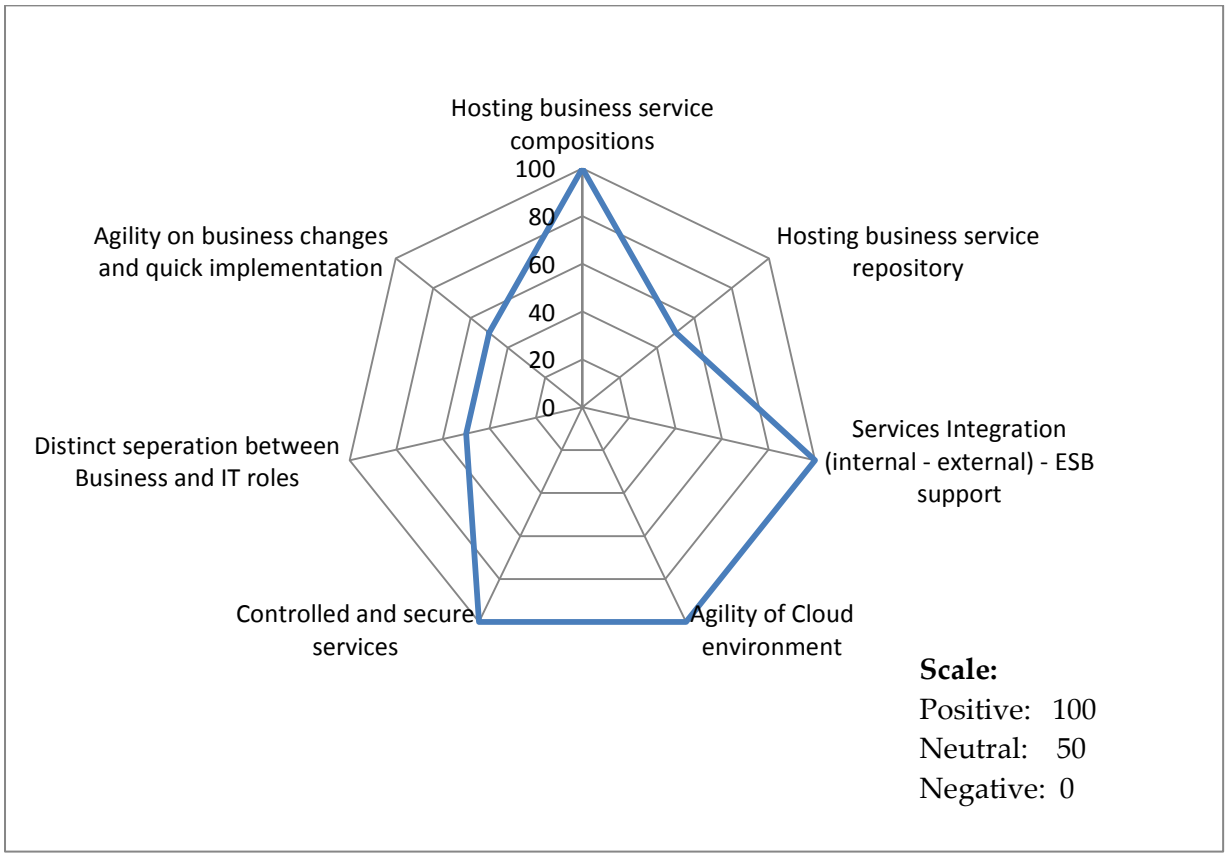


Figure 56 Feedback on suitability of prototype with respect to fit with BASE/X

In general, Pega 7 is a platform that can fit with BASE/X concepts in a great level. Service compositions can be hosted as shown by the prototype. Integration to legacy systems and to systems of external partners is also feasible, providing a right infrastructure to host service compositions. Even though in the prototype we did not make use of an ESB (services were

called directly), Pega supports many "servitization" standards (e.g. SOAP, JMS) meaning that the use of an ESB can also be supported.

One component of BASE/X, namely a business service repository, that Pega 7 does not out-of-the box support, may not be a hampering issue. Possibilities of developing such a repository within Pega exist, but the discussion is whether it is so necessary to have a service catalog within a BPM tool. It seems more reasonable to position it to an "integration" layer which can offer (SLA) management and overview of these services. However, even in that case, the BPM tool should have access to that repository to get any data needed such as technical specifications of a service, i.e. a URI.

Important aspects related to flexibility to perform changes and develop new applications due to new business requirements may be an issue, especially if IT and business people are not well trained to use the tool. However, we believe that Pega 7 allows for agile implementations and quick changes and it should be better if business and IT people collaborate so they can help each other. It is true that it will require time and efforts from both groups to learn the tool but since you get the right skills, the offered capabilities will leverage your agility.

Chapter 8

Conclusions

This thesis is in the frame of the research that Eindhoven University of Technology conducts along with industry on how organizations should shift to a service-dominant world. Having developed BASE/X, a business engineering framework for service-dominant business, its concepts have to be tested in practice. Our study was aiming at that direction and more specifically to apply the concepts of service compositions.

In this chapter we conclude our work. In Section 8.1 we summarize the findings of our research. In Section 8.2 we discuss in more details any limitations we faced throughout the project. In Section 8.3 we propose our recommendations for future research and finally in Section 8.4 we reflect on the entire course of the project.

8.1 Research Conclusion

The goal of the current project was defined as follows:

Goal of the project: *Design of an enterprise standard architecture for service management in order to show how the service composition should be realized.*

The approach we followed to design the targeted architecture was first to select appropriate conceptual tools and frameworks that would lead to final architectural models. With a number of steps we explained how business concepts are translated into technological realizations. We used SOA principles to show how legacy systems can be "servitized" in order to expose functionality and services. In that way, IT constraints can be overcome, allowing organizations to adapt flexibly in business changes by easily exposing services.

The drawn architectural models were presented to experts from De Lage Landen and we received positive feedback, even though some concepts are subjected to discussion (mainly how we define business capabilities and how building blocks capabilities can be combined). The fact that the designed architecture is perceived as being in accordance to BASE/X concepts makes us believe that the project achieved its goal. Of course, adaptations and improvements can be made, as discussed in the sequel of this chapter.

The concepts of the designed standard architecture were applied in a working prototype of business process automation and compositions, where "servitization" techniques and integration of legacy systems (mainly through web services) resulted in an application where new service compositions could be offered to customers. After analysing a specific business model and the business services that Athlon Car Lease offers, we used Pega 7 as a tool with business process management and case management capabilities to develop a prototype which orchestrates services with the aim to demonstrate how business agility can be addressed. The prototype achieved to combine services, both internal and external to Athlon, into a single application, and especially with a flexible and quick way of

development. Use case scenarios were also presented to demonstrate how easily changes in business logic could be performed.

We finally presented the approach of how to evaluate the prototype based on quality characteristics, i.e. functionality and usability. After discussions with both architects and business people on how they perceive the essence of such an application, the positive and encouraging feedback (even though we could not get statistically significant results) showed that the prototype worked on the right direction and acted as a successful proof of concepts of the BASE/X framework.

Through the entire course of the project, a few sub-goals had also been set. One was to discover what efforts are required to expose services in order to be called by a BPM tool. Even though the prototype was scoped to simple scenarios, we managed to get functionality lying on Information Systems like Document Management System and the cloud system for database management, in a rather short time of development.

The other main sub-goal was to assess the chosen BPM tool. Pega 7 seems a powerful tool with a lot of functionality and its own architectural approach (situational layer cake) supporting it. Despite the fact that it does not fully support all concepts and components of BASE/X, mainly due to the absence of a service repository and the considerable amount of effort to learn how to perform business changes, we believe that it can help any organization in the direction of business process automation and service management. It is true that requires a lot of effort to be fully learned, but once you understand its approach and capabilities, you can quickly build new applications.

8.2 Limitations

A main omission of the current project was the evaluation of the designed standard architecture to check if the models are functional, suitable for the organization, interoperable, modifiable and in general whether they meet any desirable quality attributes. The essence of evaluating software architectures is that it saves development efforts, time and costs [50] when designing an IS architecture. There are many methods for software architecture quality evaluation, like Architecture Tradeoff Analysis Method (ATAM) [51], Software Architecture Analysis Method (SAAM) [52] and Active Reviews for Intermediate Designs (ARID) [53]. However, in DLL there is an ongoing change in integration strategy and on the one hand it would be not wise to try to fit our designed models with old architectures, while one the other hand it would not be easy to map the designed architecture with the new architecture models that the organization builds.

Another constraint in our project was the fact that we had rather limited amount of time to implement the prototype and not many human resources were available to help in creating a fancy application. Integration with the main ERP system of Athlon, ATLAS, proved to be hard for our time constraints and in order to get any data of that system we had to find workarounds (extracting relevant data with some tools and enter them in Pega 7 environment, instead of calling ATLAS directly). Also, Pega 7 is a hard tool to learn in a short period and work of experts from Pega was vital for the completion of a functional prototype.

One more thing that this study did not touch upon is technical specifications of the implementation and aspects like security or performance issues. Regarding the technical details, it was our decision not to mention such details in the report since our aim was to prove in a high level how service compositions will provide flexibility to an organization. With respect to security issues, especially since the application was built in a cloud environment, many aspects were taken into account by experts from DLL, Athlon and Pega, securing a safe environment with all the right authorizations. For performance issues, it was left out of the scope of our study since the outcome was a prototype and not a production application.

8.3 Future Work

The next logical step after the design of an enterprise standard architecture is the detailing of these models in order to end up with more concrete architectures, reaching the back bottom right cell of the three-dimensional design "cube", as depicted in Figure 6 (of course before designing detailed project architectures, there should be first an alignment between the standard architecture and the existing architectures of an organization as we explained in Section 8.1 above). These concrete architectures can be designed for each line of business of the organization, transforming the entire organization into a service-dominant player in its markets.

Now that we have shown how service compositions can be realized, a possible future work can be how to realize a cost/benefit analysis of a composition with respect to its corresponding business model. As suggested in [1], this can be done by assigning a cost of a function of a business service (as a monetary figure per function call). With such a Quality of Service (QoS) parameter, we can measure the total cost/benefit of a service composition. However, this suggestion has to be applied in practice. Moreover, the financial evaluation of service compositions can be related to the time that is needed to provide new profitable offerings to the market due to changes in business models.

Further research can be conducted on how organizations can expose business services to be used by other business partners of the same business network. In this study we took as granted that DLL is the focal organization that uses internal and external services to compose them into new offerings. But what if DLL wants to expose a business capability to its partners? How does it affect the service management? We do believe that technically this can be done with the same concepts of "servitization" we handled in the current thesis, but investigation on the business viewpoint should be done (which services should be exposed and whether this can result into profit).

As we discussed Section 8.1, we did not evaluate the main artifact of the project, the enterprise standard architecture. But we did evaluate the other artifact, the implemented prototype of service compositions, which we believe adheres to concepts of our architectural models. However, that evaluation was limited to assess some quality attributes of the final application and to do that we mainly had discussions with a very few business and IT people. These results need to be generalized by asking more experts or compare opinions from other organizations where such initiatives take place.

Finally, since BASE/X offers also a set of business design tools for each of the four layers of the business pyramid, we assume that it would be handy to develop software applications of these conceptual tools so that business people can experiment. Such a conceptual tool is the Business Service Composition Blueprint as proposed in [3] for the identification of service compositions based on the business model and the underlying business service. A corresponding software application in which business people can simply drag and drop services for making new compositions can be used for the design of service compositions before the actual implementation in a BPM platform.

8.4 Reflection

The project can be considered as a multifaceted one, dealing with concepts like Architecture Designing, Service Oriented Architecture, Business Process Management, both from a business perspective and a technical perspective. Starting such a project combining all these, it seems wise to have performed a literature study in advance in order to see how concepts are related to each other. This study was performed during the course of the project but it would have saved some time from the actual execution of the project if it was done prior to its beginning.

The description of the use case of the Driver Desk that we used as a starting point in our discussions with Pega for the implementation of the prototype could have been documented in more details. In a limited implementation period, it is more efficient to have the most complete documentation of what you desire to develop as possible. However this cannot always be possible since you need to have a first experience with the BPM tool in order to see what is required. In our case, Direct Capturing Objectives sessions took place in the beginning of the implementation and hopefully we managed to capture the functionality and expectations of the prototype.

One more issue related to the implementation of an application, the design of which has preceded, is that it requires an iterative approach. The design of business processes and use case scenarios may require adaptations based on difficulties or emerging needs during the implementation. Good planning has to be taken into account in order to avoid long delays in the deliverables.

Moreover, resource planning for helping in the implementation of the prototype was a bit cumbersome causing a few delays. Many people, both business and IT should be involved and the alignment of their agendas was not an easy task. The same goes for the final presentations for the discussions on the deliverables, leading to the conclusion that such a project may take extra time to be completed.

Finally, we should mention that since the project required a lot of collaboration with experts and professionals from many organizations, extra work was done to discuss many aspects of the projects (setting the goals, the scope and approach, explaining concepts of BASE/X to people who were not aware of, presenting mid-term results, taking decisions on certain stages). This work was not depicted in the report but had to be expected, scheduled and performed in addition to the writing of the report.

References

[1] P. Grefen, E. Lüftenegger, E. van der Linden, C. Weisleder; *BASE/X Business Agility through Cross-Organizational Service Engineering*; Eindhoven University of Technology, 2014.

[2] P. Grefen; *Business Information System Architecture*; Eindhoven University of Technology, 2014.

[3] E. Lüftenegger; *Service-Dominant Business Design*; Dissertation, Eindhoven University of Technology, 2014.

[4] T. Skarabahataya, A. Khramava, J. Li; *Databased design for De Lage Landen*; Eindhoven University of Technology, course1VD50: Business Service Tool Design, 2013.

[5] T. Claessens; *Developing a prototype for the service composition layer of the BASE/X framework*; Bachelor Thesis, Eindhoven University of Technology, 2014.

[6] R.G. Lee, B.G. Dale; *Business process management: a review and evaluation*; Business Process Management Journal, Vol. 4 Iss: 3, 1998; pp. 214-225.

[7] W.M.P. van der Aalst, A.H.M. ter Hofstede, M. Weske; *Business Process Management: A Survey*; International Conference on Business Process Management (BPM 2003); pp. 1-12.

[8] M. Hammer; *What is Business Process Management*; 2010.

[9] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers; *Fundamentals of Business Process Management*; 2013.

[10] P. Loucopoulos, V. Karakostas; *System Requirements Engineering*; 1995.

[11] Gartner; *Magic Quadrant for Business Process Management Suites*; Oct. 2010.

[12] Gartner; *Magic Quadrant for Intelligent Business Process Management Suites*; Sept. 2013.

[13] Forrester; *The Forrester Wave: BPM Suties*; Q1 2013.

[14] N.Ward-Dutton; *BPM Tehcnology Review 2013: Vendor Comparison report*; MWD Advisors, July 2013.

[15] Steve Craggs; *Comparing BPM from Pegasystems, IBM and TIBCO*; August 2011.

[16] Avio Consulting; *BPM Product Analysis: A Comparison of Pegasystems PegaRULES Process Commander and Oracle BPM Suite*; August 2013.

[17] R. Hilliard; All About IEEE Std 1471; 2007; available at http://www.iso-architecture.org/ieee-1471/docs/all-about-ieee-1471.pdf.

[18] J. Truijens, A. Oosterhaven, R. Maes, H. Jägers, F. van Iersel; *Informatie-infrastructuur: een Instrument voor het Management*; Kluwer Bedrijfs-wetenschappen, 1990 (in Dutch).

[19] P. Kruchten; *Architectural Blueprints—The "4+1" View Model of Software Architecture*; IEEE Software 12 (6); November 1995, pp. 42-50.

[20] R. Wieringa, H. Blanken, M. Fokkinga, P. Grefen; *Aligning Application Architecture to the Business Context*; Proceedings 15th International Conference on Advanced Information Systens Engineering; Klagenfurt/Velden, Austria; 2003; pp. 209-225.

[21] P. Grefen; *Mastering e-Business*; Routledge; 2010.

[22] S. Brahe; *BPM on Top of SOA: Experiences form the Financial Industry*; 5th International Conference on Business Process Management (BPM 2007); pp 96-111.

[23] A. Almonaies, JR. Cordy, TR. Dean; *Legacy system evolution towards service-oriented architecture*; International Workshop on SOA Migration and Evolution, pp. 53-62, 2010.

[24] Erl T; *SOA: principles of service design*; Upper Saddle River, NJ; Prentice Hall, 2007.

[25] J. Magretta; *What Management Is*; New York: Free Press, 2002.
[26] A. Osterwalder, Y. Pigneur; *Business Model Generation*; Wiley, 2010.
[27] T. Kohlborn, A. Korthaus, T. Chan, M. Rosemann; *Identification and analysis of business and software services—a consolidated approach*; Services Computing, IEEE Transactions on, 2(1):50-64, 2009.
[28] V. De Castro, E. Marcos, R. Wieringa; *Towards a service-oriented mda-based approach to the alignment of business processes with it systems: from the business model to a web service composition model;* International Journal of Cooperative Information Systems, 18(02):225–260, 2009.
[29] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard ;*Web Services Architecture*; W3C Working Group Note, 11 February 2004, http://www.w3.org/TR/ws-arch/.
[30] M.P. Papazoglou, W.J.van den Heuvel; *Service oriented architectures: approaches, technologies and research issues*; The VLDB Journal 16, pp. 389-415, 2007.
[31] M. Gudgin, M. Hadley, N. Mendelsohn, J.J. Moreau, H.F. Nielsen, A. Karmarkar, Y. Lafon; *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*; W3C Recommendation, 27 April 2007, http://www.w3.org/TR/soap12-part1/ .
[32] R. Chinnici, J-J. Moreau, A. Ryman, S. Weerawarana; *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*; World Wide Web Consortium, 27 March 2006, http://www.w3.org/TR/wsdl20/ .
[33] OASIS Standard; *Web Services Business Process Execution Language v2.0 (WS4BPEL)*; April 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf .
[34] T. Freund, P. Niblett; *ESB Interoperability Standards*; 02 June 2008.
[35] R.C. Seacord, D. Plakosh, G.A. Lewis; *Modernizing Legacy Systems*; Carnegie Mellon, SEI, 2003.
[36] H.M. Sneed; *Wrapping Legacy Software for Reuse in a SOA*; 2008.
[37] W.M.P. van der Aalst, M. Weske, D. Grünbauer; *Case Handling: A New Paradigm for Business Process Support*; Data and Knowledge Engineering 53(2), pp. 129–162, 2005.
[38] W.M.P. van der Aalst, P.J.S. Berens; *Beyond Workflow Management: Product-Driven Case Handling*; In S. Ellis, T. Rodden, and I. Zigurs, editors, International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001), pp. 42–51, ACM Press, New York, 2001.
[39] I. Vanderfeesten, H.A. Reijers, W.M.P. van der Aalst; *Product Based Workflow Design with Case Handling Systems*; 2007
[40] D.M. Strong, S.M. Miller; *Exceptions and exception handling in computerized information processes*; ACM Transactions on Information Systems, 13(2), pp. 206–233, 1995.
[41] F.C. Lunenbrug; *Organizational Structure: Mintzberg's Framework*; International Journal of Scholarly, Academic, Intellectual Diversity, Volume 14, Number 1, 2012.
[42] W.M.P. van der Aalst, M. Adams, A.H.M. ter Hofstede, M. Pesic, H. Schonenberg; *Flexibility as a Service*; Database Systems for Advanced Applications, DASFAA 2009 International Workshops: BenchmarX, MCIS, WDPP, PPDA, MBC, PhD, Brisbane, Australia, April 20-23, *2009,* pp. 319-333.
[43] The Workflow Management Coalition Specification; *Workflow Management Coalition: Terminology & Glossary*; Document Number WFMC-TC-1011, February 1999.
[44] M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede; *Process-Aware Information Systems: Bridging People and Software through Process Technology*; 2005.

[45] ISO/IEC 9126-1:2001 – *Software Engineering – Product Quality – Part 1: Quality Model*, 2001.
[46] F. Losavio, L. Chirinos, N. Lévy, A. Ramdane-Cherif; *Quality characteristics for software architecture*; Journal of Object Technology, 2(2):133-150, 2003.
[47] J.T.S. Ribeiro; *Multidimensional Process Discovery*; 2013.
[48] J.R. Lewis; *IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use*; Technical Report 54.786, 1993
[49] D. Bertram; *Likert Scales are the meaning of life*; CPSC 681-Topic Report, http://poincare.matf.bg.ac.rs/~kristina/topic-dane-likert.pdf, retrieved September 10th, 2014.
[50] P. Clements, R. Kazman, M. Klein; *Evaluating a Software Architecture*; December 2001.
[51] R. Kazman, M.H. Klein, P.C. Clements; *ATAM: Method for architecture evaluation*; August 2000.
[52] R. Kazman, L. Bass, G. Abowd,M. Webb; *SAAM: A Method for Analyzing the Properties Software Architectures*; Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, May 1994, pp. 81-90.
[53] P.C. Clements; *Active Reviews for Intermediate Designs*; August 2000.

Appendix A: Project Plan

The project officially started in February 1st, 2014 and the initial project plan was presented on February 17th, 2014 in the kick-off meeting. The scheduling is presented in Table 4.

Table 4 Project Planning

Phase	Name	Deliverable(s)	Duration	End Date
0 – Set up	Project Planning	Project Plan	2 weeks	week 7 (15th Feb)
1 – Preparatory	Charting current architecture	Architecture specifications	9 weeks	week 16 (19th Apr)
	Exploring BPM and Service Dominance	BPM platform requirements BPM platforms selection list Conclusion of BPM choice Specifications of services ready for SD Specifications of business processes based on Business Model		
2 – Design	Architecture Designing	Standard architecture model	7 weeks	week 23 (7th Jun)
3 – Prototyping	Prototype Implementation	Prototype Documentation of prototype	7 weeks	week 30 (26th Jul)
	Evaluation	Evaluation Questionnaires		
4 – Finalizing	Results Presenting	Presentation of architecture and prototype MSc Thesis	2 weeks	week 32 (9th Aug)

Appendix B: Car Leasing Business Model Canvas

<table>
<tr><th>Key Partners</th><th>Key activities</th><th>Value Propositions</th><th>Customer Relationship</th><th>Customer Segments</th></tr>
<tr>
<td rowspan="2">Dealer network, Vereniging ROB
Network responsible for RMT
(Repair, Maintenance, Tyres)

Damage Repair network
Responsible for damage repair

Multi Tank Card
Fuel

Insurance companies
Insurance of vehicles, liability

International partners
Athlon subsidiaries abroad and other companies

NS (Dutch Railways)
Public transport

XXIMO
Prepaid mobilities (debet card)

Car traders
For remarketing purposes

Achmea
Supplier of E-driver platform</td>
<td>Car Purchasing
Car Maintenance
Insurance
Damage & Claims
Fuel management
Rental cars
Remarketing
Reporting
Logistics</td>
<td rowspan="2">Mobility, in terms of cars and other means of transportation

Hassle free

Cheaper, through economies of scale

Better cashflow, no investment required

Professional fleet management

Purchase service

Remarketing service

Relief vehicle, rentals

Mobility advice</td>
<td>Personal contact (fleetmanagers)

Online ordering
Online reporting
After sales (Drivers desk)
Road assistance</td>
<td rowspan="2">B2B (companies and their employees)

B2C (consumers, private lease)

International customers

Large accounts, Key accounts, SME's</td>
</tr>
<tr>
<td>Key Resources

Athlon members (Sales, Backoffice activities)
Funding (money)
Vehicles (assets)
Premises (parking ground, office)
IT (IT services, both Athlon / DLL)</td>
<td>Channels

Face2Face

Brokers

Online
- Private lease (new customers)
Athlonline (existing customers)</td>
</tr>
<tr><th colspan="3">Cost Structure</th><th colspan="2">Revenue Streams</th></tr>
<tr>
<td colspan="3">Cost of personel (Athlon members)
Cost of funding (interest)

Invoices for Vehicles
Invoices for Fuel
Invoices for RMT
Invoices for Insurance
IT costs</td>
<td colspan="2">Lease rates, covering:
- interest and depreciaton
- RMT
- Insurance
- Relief vehicles
- Management fee

Rental income
Asset sales (Remarketing)
Kickback bonusses on:
- vehicles
- tyres
- glass</td>
</tr>
</table>

Figure 57 Car Leasing Business Model Canvas

Appendix C: Functionality of prototype services

In Table 5 below we present the list of functional requirements that the prototype should meet, along with related specifications.

Table 5 Functional requirements and specifications for the prototype

S/N	Requirement	Specification
1	Driver Desk shall provide the ability to be reached by customers/drivers through the following channels: phone/email/web form (social media are out of scope of the prototype).	In case of a phone call, the following driver's data will be required: - Vehicle's License Plate - Driver's Last Name - Company's Name In case of an inquiry submitted by email or through the current available web form, extra fields will be required: - Gender - Address - Email - Telephone - Inquiry Type - Inquiry Description
2	The application shall provide a search field for retrieving cases.	The driver desk officer will enter the case id or the vehicle's plate number to retrieve related cases.
3	The application shall provide a list of past cases related to the current driver in order to provide a solution.	For each driver's inquiry, a list of past cases shall be generated with the following elements: - Case ID - Case Type - Case Description - Enter Date - Vehicle's License Plate - Driver's name - Given solution - Officer handled that case
4	The application shall provide graphical information related to metrics and statistics.	The following graphical information shall be visible: - Percentage of cases closed successfully. - Average time to provide a solution.
5	The dashboard shall provide all the right information to the caseworker related to the current case/inquiry/customer. These are driver's personal details, case details	The dashboard shall provide the following information for each case/inquiry/customer: - Case ID

	and information on how to handle a case.	- Case Type - Enter Date - Case Description - Vehicle's License Plate - Contract No - Driver's Name - Driver's Telephone - Driver's Email - Driver's Address - Related documents (Contract, Driver's license)
6	The application must prevent the driver desk officer to provide an answer to a driver in case that is restricted by his contract agreements.	A specific field/flag will indicate to the driver desk officer that he cannot respond to the driver's inquiry according to the regulations of his contract. In that case he should inform the driver to refer to his fleet manager.
7	The application shall be able to create a report with a list of fines.	The driver will request a report with a list of his fine tickets indicating the period. The report will contain the fine ticket type, the date and the amount.
8	The application shall provide services for (re-)issuing a fuel card or cancelling and existing one.	The driver will provide the following information: - Fuel Card No (if known) - Fuel Card Type - Vehicle's License Plate - Reason for replacement - Driver's Name
9	The application shall provide information on nearest gas stations (specific radius), given the location of the driver.	Given driver's zip code, the application shall return a list of gas stations in a specific radius with information of gas prices.
10	The application shall generate a report with the fuel transactions of a vehicle.	Give a vehicle's license plate and the fuel card number, the application shall generate a report with a list of transactions containing information on date of transaction, type of fuel, gas station name and address and amount.
11	The application shall be able to provide answers to inquiries related to repair, maintenance and tires (RMT) of a vehicle.	There is no specification defined since the requirement is out of scope of the current prototype.
12	The application shall be able to provide answers to inquiries related to damages	There is no specification defined since the requirement is out of scope of the current prototype.

13	The application shall be able to provide answers to inquiries related to returning vehicles.	There is no specification defined since the requirement is out of scope of the current prototype.
14	The application shall be able to provide answers to inquiries related to replacement vehicles.	There is no specification defined since the requirement is out of scope of the current prototype.
15	The application shall be able to answer questions on the status of an ordered service or product.	The application shall provide the information of the current status of an ordered service or product by retrieving the corresponding case.
16	The application shall be able to register and handle general complaints of customers/drivers.	For each generated case, an input field for registering the complaint shall be available.

Appendix D: Mock-up Driver Desk dashboard

The following mock-up interface was designed to explain to people from Pega the required functionality of the prototype.

From the list with cases on the left, the Driver Desk officer picks one and reviews the details. Then he can initiate the corresponding service and a result is being generated with the status of the service. A response is also generated to be sent to the driver. There is also a search function. Statistics and graphs show information about objectives. These are shown in Figure 58 below.

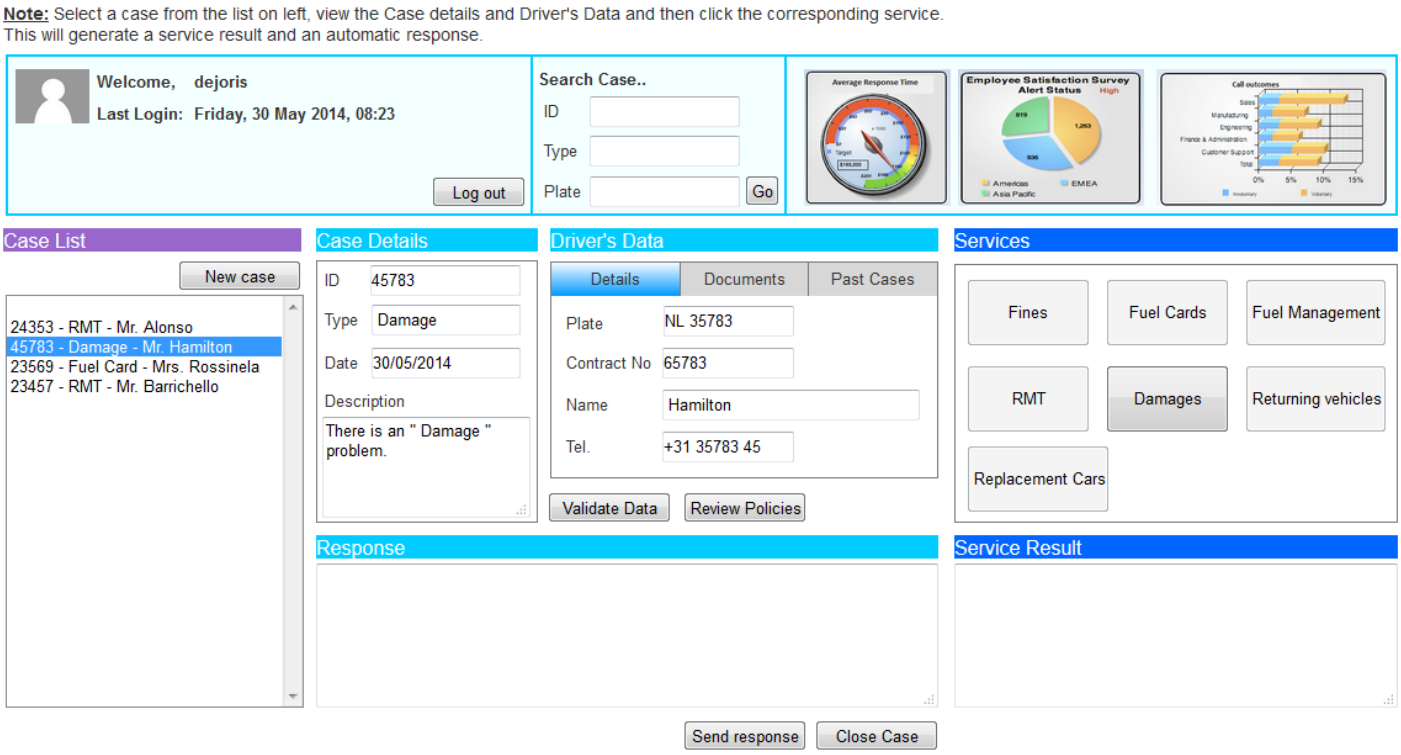


Figure 58 Driver Desk dashboard, mock-up interface

Appendix E: Evaluation questionnaires

In this Appendix we present the questionnaires and any raw results we got after the discussions with people of the company in the frame of the project evaluation. In Section E.1 we present the questions and answers related to suitability aspect of the prototype. In Section E.2 we present the subset of IBM Usability questionnaire.

E.1 Suitability aspect of prototype

Table 6 Suitability of prototype - Interviewee A

Evaluation on Suitability of prototype		
Interviewee: Angelo Jitbahadoer, Driver Desk Team Leader		Date: 09-09-2014
Q1	Does the application provide a single point of contact for drivers to solve their inquiries about leased vehicles quickly and efficiently? (Also, rate your final answer with a negative, neutral or positive indication).	
	○ Negative ○ Neutral ◉ Positive	
Q2	Does the application provide the right functionality to execute all tasks and services that Driver Desk is responsible for? (Also, rate your final answer with a negative, neutral or positive indication).	
	○ Negative ○ Neutral ◉ Positive	
Q3	Is the application capable of providing insights on processes that fail to meet drivers' needs? (Also, rate your final answer with a negative, neutral or positive indication)	
	I think it will be capable if you have a complete application. ○ Negative ◉ Neutral ○ Positive	
Q4	Does the application allow performing of changes on business rules and decisions easily? (Also, rate your final answer with a negative, neutral or positive indication)	
	○ Negative ○ Neutral ◉ Positive	

Table 7 Suitability of prototype - Interviewee B

Evaluation on Suitability of prototype	
Interviewee: Robert Diependaal, EA, from a business perspective	Date: 09-09-2014
Q1	Does the application provide a single point of contact for drivers to solve their inquiries about leased vehicles quickly and efficiently? (Also, rate your final answer with a negative, neutral or positive indication).
	Within the context of the PoC, the answer is yes. Though, one should realize that not all steps in the business process are supported (due to time constraints) and that some functionality is blended into the current process, for the sake of integration. Overall, the dashboard does show potential to fully support the Drivers Desk process. ○ Negative ○ Neutral ◉ Positive
Q2	Does the application provide the right functionality to execute all tasks and services that Driver Desk is responsible for? (Also, rate your final answer with a negative, neutral or positive indication).
	No, merely because that is not in scope of the PoC. However, the functionality as defined for the PoC has been implemented, in an acceptable way. ○ Negative ○ Neutral ◉ Positive
Q3	Is the application capable of providing insights on processes that fail to meet drivers' needs? (Also, rate your final answer with a negative, neutral or positive indication)
	Yes, it is. The 'Wrapup' of each call enables the Driver Desk Officer to record to what extent the driver is satisfied, including additional free text. That, in combination with the fact that all the functions performed in the call are recorded, offers enough base for reporting and generating insights. ○ Negative ○ Neutral ◉ Positive
Q4	Does the application allow performing of changes on business rules and decisions easily? (Also, rate your final answer with a negative, neutral or positive indication)
	Not sure about the application, the development environment does. As stated before, untrained business members should not do any changes in the development environment. But trained members are able, to a certain extent, to change business rules and decisions. ○ Negative ◉ Neutral ○ Positive

Table 8 Suitability of prototype - Interviewee C

Evaluation on Suitability of prototype		
Interviewee: Robert Diependaal, Enterprise Architect		Date: 09-09-2014
Q1	Is the tool capable of hosting service compositions (mash-up or process type)? (Also, rate your final answer with a negative, neutral or positive indication).	
	The mashup type is part of the Driver Desk dashboard, the PoC has shown that combining different services into a single, coherent business process is possible. The process type occurs in the supporting (back-office) systems; these systems have not been touched during the PoC nor are they supported by Pega. Based on the supplier's information regarding Pega, support for the process type is on board as well. ○ Negative ○ Neutral ◉ Positive	
Q2	Is the tool capable of storing a business service repository where the business capabilities of the organization can be found? (Also, rate your final answer with a negative, neutral or positive indication).	
	A business service repository may be developed within Pega but is not on board as such, by default. During the PoC, discussions on a business service repository resulted in the conclusion that such a repository is best positioned in the integration layer, not in the BPM layer. "Better tools are available for that". As integration claims governance, (SLA) management and monitoring of these services, the overview of all available services (both internal and external) indeed belongs more to integration. ○ Negative ◉ Neutral ○ Positive	
Q3	Can the tool integrate services from different ISs of the organization (internal) and external as well? Is the use of ESB supported/needed? (Also, rate your final answer with a negative, neutral or positive indication)	
	The PoC shows that services from different ISs can be integrated. The use of the ESB has not been tested in the PoC (direct integrations were used) but is required in future use. The future ESB will be exposing services using market standards (SOAP, e.g.), Pega is supporting these standards. Conclusion is that use of an ESB is both required and supported. ○ Negative ○ Neutral ◉ Positive	
Q4	Is the cloud environment able to support integration of systems in an agile way? (Also, rate your final answer with a negative, neutral or positive indication)	
	Yes. Apart from the normal integration issues (IP restrictions, firewalls and so on), the integration with the MTC organisation was implemented in less than a day, based on a (simplified) use case.	

	○ Negative ○ Neutral ◉ Positive
Q5	Can the tool provide controlled access to services, adhering to security policies? (Also, rate your final answer with a negative, neutral or positive indication)
	Yes, topics such as IP restrictions, firewalls, uid/pwd were part of the PoC and have been implemented. Furthermore (not tested but based on the supplier's information), numerous security possibilities, as to granting and revoking access to parts of the application, are available. ○ Negative ○ Neutral ◉ Positive
Q6	Does the tool allow for a distinct separation of roles between business and IT people? (Also, rate your final answer with a negative, neutral or positive indication)
	Based on supplier's information, yes. ○ Negative ◉ Neutral ○ Positive
Q7	Does the tool, both the design environment and the implemented application, allow for agile changes on business processes and quick implementations driven by business needs? (Also, rate your final answer with a negative, neutral or positive indication)
	Yes. Although process flow, for one, is a topic that is merely addressed in the design environment, not in the implemented application. Pega states that it is possible to design wizards that help business people to overcome technical issues while designing. And part of the building blocks (not all) within the Pega design environment may be changed by knowledgeable business members. Without proper training, however, business people are not likely to perform any changes in the environment. Pega stresses on the collaboration between business and IT to achieve agility. ○ Negative ◉ Neutral ○ Positive

E.2 Understandability aspect of prototype

The subset of questions from IBM Usability questionnaire is presented below.

Table 9 Understandability of prototype

Evaluation on Understandability of prototype – IBM Usability Questionnaire		
Interviewee: Name, Driver Desk Team Member		Date:
Q1	Overall, I am satisfied with how easy it is to use this system. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q2	It is simple to use the system. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q3	I can effectively complete my work using this system. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q4	I feel comfortable using the system. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q5	The system gives error messages that clearly tell me how to fix problems. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q6	The organization of information on the system screens is clear. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	
Q7	The interface of this system is pleasant. Strongly Agree Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7	