The selection process of model based platforms

Dré Hendriks — S4130626

July 13, 2017

First supervisor/assessor:
dr. S.J.B.A. Hoppenbrouwers
stijnh@cs.ru.nl

Second assessor:
dr. P. van Bommel
P.vanBommel@cs.ru.nl

Radboud Universiteit

Abstract

In this research we searched for the most important characteristics of both model based platforms and organizational situations and how they should be matched to each other. We performed interviews with industry experts to find out how the industry looks upon the matching of situations and platforms. We found that the most important characteristic is the main functionality the platform provides or the situation needs. But a lot more characteristics must be taken into account.

After we found the characteristics were found a proof of concept was performed to see if a framework could be created to help with the selection of a platform for the situation. We created a framework using the characteristics found earlier and asked industry experts to provide the information about the platforms and the proof of concept seems to work. So with some further development this framework could be used to support the platform selection process.

Acknowledgements

I would like to say a special thank you to Stijn Hoppenbrouwers for all the guidance and feedback he provided during the process. I would also like to thank the experts that participated in the research as without them it would be impossible to do this. Last but not least I want to thank everyone that have listened to me and provided me with feedback.

Contents

1 Introduction

Software development gets more important in this world of rapid IT progression. In the last few decades software engineering went from being only for the professional developers to the situation that even kids are making software. This means that the community of software developers became a lot bigger and more diverse, the community now consists of both the professional developers and the civilian developers making software. This diversity has as a consequence that the software development community has a wide variety of needs, as the professionals have different, often more technical, needs than the civilians. The civilian programmers often don't have a deep understanding of the technical background of the tools they use, they just know how to use the tools and create applications with them. Now this group of civilian developers gets bigger and bigger, the market is seeing opportunities in this particular section of the software development community. This part of the community is in need for flexible, simple, efficient, *non-technical* tools for developing applications.

We see the market reacting to this by increasing effort in SaaS, PaaS and IaaS. SaaS (Software-as-a-Service) is software offered to their customers in *the cloud*. An example of this would be Google Drive, which is an online environment for keeping files and editing them. This is *software* that can be accessed via a browser, which means the software doesn't have to be installed on the computer, but users can use it anywhere using any device. This can also be done with professional software used in organizations. A level below SaaS is PaaS (Platform-as-a-Service) which will be the main part of this research, it offers developers an online platform to help them develop, run and manage applications. At the lowest level is IaaS (Infrastructure-as-a-Service) which offers people infrastructure to run applications or algorithms on, an example of this could be a data scientist which has to run a lot of calculations once in a while. The data scientist can rent the infrastructure it needs and thus does not have to invest heavily in his own hardware.

In October 2016 Gartner released a paper predicting what the year 2017 will

bring to IT (Hilgendorf & DeBeasi, 2016). In this paper they highlight three elements that will be important in the coming years: *sense*, *adapt* and *scale*. Platforms will play a role in this development of the market. With regards to *sense* Gartner writes about the Internet of Things (IoT), equipping everyday things with sensors and use that as a sensing object for it's surroundings. Platforms play a limited role in this, platforms do enable the *everyday person* to build an application. In that regard IoT is boosted a bit by the platforms as a lot of IoT projects are hobby projects of people making cool stuff in their home for themselves.

The next element that is going to be important according to Gartner is *adaptability*. With all this data coming in from all the sensing, organisations must analyse this data quickly and adapt their business on the outcome. Applications must be built in order to transform data from system to system, this is where the platforms come in, as many platforms focus on enabling their users to get the data to flow from one (cloud)application to the other successfully.

The last and probably the most important of Gartner's elements from a platform perspective is *scalability*. Scalability is made easy by the rapid growth of the cloud and the cloud platforms that have emerged the past years. Technically a cloud application can start as a small project with only a few users and scale up to millions of users in a matter of seconds as cloud providers have almost infinite capacity. This quality makes cloud applications a good option to start an application in, because providers provide scaling so that the application is scaled for the amount of users it as, instead of too much or too little, this makes making a big hardware investment obsolete.

There has been done a lot of research whether the advantages of SaaS and PaaS solutions outweigh the disadvantages (Bhardwaj, Jain, & Jain, 2010), (Mital, Pani, & Ramesh, 2014). Those articles mainly conclude that the advantages of cloud solutions do indeed outweigh the disadvantages in most cases. There are also situations in which the advantages of the cloud platform do not really provide any further value, or provide less value than the disadvantages outweigh. Cloud platforms are in the centre of attention and are being hyped. This enables the cloud market to rapidly grow as a lot of companies move to the cloud to keep up with the pace of the market. This enables a lot of organizations to start

providing cloud services to their customers.

To illustrate the growth of cloud based software one can look at the following statistic from Taylor (2016). This statistic shows that in 2008 12% of the businesses using Customer Relationship Management software (CRM), used it in the cloud. In 2016 that was 87%, which is of course a huge growth. This example shows the rapid growth of the cloud market and thus the rapid replacement of software on premises by software that is hosted in a cloud environment.

We see that this move to the cloud is spreading from non-business critical applications like Customer Relationship Management systems to business critical applications, for a software development company this could be software development platforms. Although this market is just starting, there are already plenty of organizations in the market offering cloud development platforms or PaaSs, ranging from small start-ups to IT giants like Google and Amazon. The question that remains unanswered in the research that is already done is: "On what basis can we decide which provider to choose". How does one know which provider offers the best solution for the situation. In this thesis we will make a start in developing a framework for helping to make this choice. The focus will be laid upon strongly specific business applications. As these applications are so specific for a certain situation, the applications can not be bought off-the-shelf but have to be *tailor-made*. This framework will **not** be a tool that provides the answer to the decision of a platform. This framework will help to see what platforms fits in the situation of the organization, but will not provide a definite answer. This framework will show you a score that certain platforms have regarding the organization's specific situation but this does not mean that the platform with the highest score is always the best choice, it means that the probability of platforms with high scores are good platforms for the situation is high. Not every aspect of the selection process can be measured in the framework, which makes the framework imperfect by definition. This means that the choice should always be made by a human that knows the imperfections of the framework and can see that in perspective.

In this thesis we will help to get insight in what should be expected from

a cloud platform, and what the differences are between them. The answer our model gives will not result in a single name of a platform which you can choose to solve your problems, but it will provide guidance and a summary of all the pros and cons of different cloud platforms in different organisational situations. This is the gap that we are hoping to fill in this thesis.

This all results in the following research question: "What are the characteristics of organisational situations and the characteristics of IS platforms that can help to find the match between situations and platforms?"

2 Theoretical Background & Concepts

There are two concepts that are connected to this thesis, that would be good gain some insight in before starting the research. The first concept is *software procurement*, because platform selection lies very close to the field of software procurement. This is because in software procurement a big step is the selection of the software to procure, which is similar to the selection of an online platform, only an online platform has some other aspects to keep in mind.

The second concepts that needs background information before the research starts is an *information system platform*, because this background information is needed before we can start to look at the best way to select a platform. We will look at what a platform is exactly, what the industry generally sees as the pros and cons of working with a platform.

2.1 Software Procurement

There are three different methods of acquiring new software (Finkelstein, Ryan, & Spanoudakis, 1996).

1. Buying "off-the-shelf" software.
2. Buying software and modify it.
3. Building new software.

These options all have their pros and cons, this makes each procurement a challenge to even pick the best way of procurement. Buying off the shelf software is generally a cheap and labour inexpensive way of procuring software, but without customization it may lack certain functionality or, also a disadvantage, have too much functionality. This is a disadvantage because 1) the functionality has been made, so it is included in the price, hence you pay for something that you don't actually use. 2) It may make the software excessively complicated to use. Users have to know which functions do what and which to use and which to leave. Plus there is an increased chance of making errors.

With the second option, buying software and modifying it, point 2 is mainly negated, but on top of the paying for unused functionality, there has to be paid

to exclude that functionality from the software and adding functionality to the software that is not normally included in the software, resulting in a higher price.

The third option has as an advantage that you only pay for what you get and what you need. The downside is that is the price of the software might be relatively high because it is made for a very specific context (that of the organization that the software is being made for). This means that the market for the software is smaller, so it might only be sold to the organization that the software is being made for, so they have to cover all the building costs. The positive side is that the software stays within the organization and if competitive advantage is gained, that too stays within the organization.

All these options need to be considered when an organisation is thinking about the method of software acquisition. The question of which method to choose is quite different relative to the situation an organization is in. For the small-to-medium enterprises (SMEs) the choice has a whole other context than for large enterprises as they often have already invested a lot in their IT departments. This makes it easier for a large enterprises to build their own software because most SMEs don't even have an IT department, so having custom software would require hiring a software company to build their custom software.

The field has found 10 factors that play a role in the choice for the software acquisition method (Daneshgar, Low, & Worasinchai, 2013; Hung & Low, 2008):

- *Strategy & competitive advantage*; Software should be built if it is of strategic value for an organisation or when it is the core competence of an organisation.
- *Cost*; It should be considered whether it is cheaper to build or buy the software. Maintenance should be taken into account
- *Scale & complexity*; Complex systems should be bought, because the expertise of specialised parties could be used.
- *Requirements fit & commoditization & flexibility & change*; Very unique software should be built, standard software (like bookkeeping systems) should be bought.
- *Time*; Building software costs more time than buying software, this should be taken into account.

- *In-house information systems expertise*; An organisation should have the information system expertise if they would like to build their own software
- *Risk*; All sorts of risks should be considered. Buying takes away certain risks, like performance and cost/budget risks. Building your own software takes away the risk of lack of support and the vendor being still in business.
- *Support structure*; The amount of support you get with the product should be considered.
- *Operational factors*; Organisations with a lot of experience in in-house development, might be more likely to choose the build method.
- *Intellectual property*; Usually the party that builds the software has the intellectual property of that software. If the organisation wants to have the intellectual property of certain (strategic) software, they might want to build it themselves.

If the choice is made to buy the software, there are still different procurement strategies possible. The software procurement process is about managing risks, the process is optimised to acquire software in a systematic way to try to make the risk of buying or building the software as small as possible. Strategies are formulated to guide the software procurement process. Clemons, Reddi, and Row (1993) formulated the *move-to-the-middle* hypothesis predicting that business will move to a state where non-core activities are outsourced and that organizations will engage in long-term relationships with fewer suppliers. By outsourcing an organization is able to transfer a part of its risks to another organization. We see this, for example, in an ERP system like SAP. By buying the SAP framework, the organization will not have the risks that accompany building an ERP system. This risk will be transferred to another, more specialised, organization, in this example SAP.

2.1.1 Unified procurement

Like in every other field, software procurement is also prone to trends. One of these trends is the unified procurement strategy, described by Kauffman and Tsai (2009). Unified procurement means that instead of buying software from a multitude of vendors, software is bought from a single vendor, supporting the

move-to-the-middle hypotheses mentioned above. A critical element in this strategy is the transferred risk that it provides. This means that the risks that come with the technology and integration thereof are transferred to the software vendor instead of the software buyer.

Also the coordination costs will go down. The coordination costs are all costs that are related to the coordination with the other parties including maintaining a good relationship with them. If a multi-vendor strategy is maintained a project with 3 vendors would be possible. This means that partnerships with 3 vendors have to be maintained (excluding possible relationships within the vendors). This will take significantly more resources than maintaining a partnership with just one vendor, which is the case in unified procurement.

Another big factor with unified procurement is vendor lock-in. Because everything is bought from one vendor, it will give the vendor more power increasing the risk that they might misuse this power in a way that is disadvantageous for the buying party. Kauffman and Tsai (2009) says that by using a unified procurement strategy a buyer actually gets more power in negotiations, because the vendor has a lot to win or lose by maintaining the relationship. If the buying party is going to procure a new software product, they are likely to first go to their single vendor. This creates a mutual dependency for both the buying as the selling party.

We also see this move to one "provider" with development platforms. It is more beneficial to stay with one platform, as this saves you costs like training employees to be able to work with the platform and through the economy of scale the licence will be cheaper per application.

2.1.2 App store model

Wenzel, Faisst, Burkard, and Buxmann (2012) made a framework which would help decide whether it's worth to start an extensive consulting process for the sales process or if a transactional process is enough. This framework takes things like price, complexity, users and implementation in consideration. If it concerns a transactional process, there are increasingly many Business-to-Business software

marketplaces. An example of these marketplaces is Salesforce's AppExchange where extensions to Salesforce.com are as easy to buy as an app in the AppStore. In the article they note that the market of these marketplaces is growing and that there are increasingly more applications or extensions to applications on these marketplaces. This trend does not apply to development platforms as those are often a big purchase and a consulting process should be started to see what viable options there are in the market and to see what platform would be a good fit. Unfortunately this consulting process is not always as well performed as it would be ideally. But often a consulting process is started because there are a lot more things like training to take into account with the purchase.

2.2 Platforms

Before we start looking at platforms specific, we'll take a look at where the move to the cloud started and what is expected to still follow. Dubey and Wagle (2007) predicts this move to the cloud back in 2007 in a way that would prove to be highly accurate. They see some important advantages of SaaS over on premises software.

- **Payment strategy:** A lot of organizations are fed up with the traditional way of buying a software product, paying a maintenance contract and having time-consuming and expensive upgrades made to their software. With SaaS organizations are able to pay a monthly fee and be free of all these pains, plus altogether it is often cheaper to pay for SaaS than on-premises software.
- **Application hosted:** The software provider takes care of all the back-end troubles and hosts the software for you, which generally has a higher availability than on premises hosting through the scale on which they operate. They host the software of all of their customers instead of an organization hosting only it's own.
- **Vendor lock-in:** SaaS offers a low level of vendor lock-in. As most providers use a monthly payment, if a provider does not meet the expectations or needs of the customer he can decide to move to another provider each month, with less hassle than to start a whole procurement cycle with a new provider and all the contractual struggle that comes with it.

- **Maintenance/updates:** It is easier for the provider to update and maintain the product on their own premises than it is for them to roll out an upgrade with all of their customers. The product is upgraded and maintained continuously. This releases the customer of a mayor update every 3 to 5 years and making sure all the old machines and applications still work with the new upgrade. The small and continuous upgrades make this process a lot more gradual and less complex.

Dubey and Wagle (2007) also predicted the order certain types of applications would move to the cloud. They predicted the following four waves of SaaS adoption:

1. Tools that are not mission critical, have low data security and privacy concerns, have a distributed user base and does not need a lot of integration with on-premises software. Examples of this kind of software are HRM applications like CRM and payroll administration. These are prime examples thanks to their very distributed user base and as they are not mission critical, the competitive advantage of such a software product is not great. The software just has to meet certain standards, there is not a lot of advantage to be taken.
2. If organizations notice that the SaaS products work well in their organizations they will probably be more motivated to try it in other parts of their business products. This is still a step behind mission critical applications that provide organizations with competitive advantage, but one could think of non-mission critical applications supporting communications with external parties. Tools that help with interactions with other organizations. For example the interaction between buyer and seller, logistics and supply chain management.
3. Organizations will start to have a growing trust in their cloud software and see that the interaction with external parties are even simplified using cloud software, more business critical software will be moved towards the cloud. For example software development environments might move to the cloud, as cloud software is now growing more and more popular the software that enables organizations to develop software will also move to the cloud and provide these advantages on the meta level of software development.

4. Then they expect a new level of functionality to be developed, as in the previous three waves of cloud adoption, this was mainly deploying the old functionality to the cloud to gain some advantages in the business process. This new wave will be about offering new functionality that is made possible through the use of cloud software. For example a spam- and virus filter application that filters the traffic before it is inside the organization's firewall.

Right now, in 2017, we're in the third wave. Specialised business critical software is being moved to the cloud. The example Dubey and Wagle (2007) gave back in 2007 is quite accurate actually, we now see online software development platforms rising and we call them PaaS systems.

Now that it's clear where the move to the cloud comes from, a definition of what we consider to be a platform has to be made. In this thesis we will use the following definition:

"A platform is an (online) environment which provides the opportunity to ***develop***, ***run*** *and* ***manage*** *applications or information systems."*

There is two different groups of platforms (van Stokkum, 2016), one is leaning towards model-based development. These platforms use models as the centre of the application. These platforms often are no-code or low-code platforms, so no or a little bit of coding or programming is needed to create the application. Simplified this means that a model is created in the platform environment, segments are dragged-and-dropped to link it all together and the application is created. As Wim van Stokkum said: *"These platforms target the new smart developers, not the COBOL developers. You can make anything with these. These [model based platforms] target business process managers and business architects."* The platform will generate the code based on what the user has modelled the model or use existing code to run the model itself as an application. If a user for example links an input field to a specific column of a database, the platform will make sure that everything end-users put in that input field, is transferred to the database. This extracts a lot of technical knowledge away from the user into the

framework, which means that the user needs less technical knowledge in order to create an application in this way.

The other group of platforms is more tech-heavy and is more like a new kind of (online) development environment. It is still a coding environment and technical knowledge is heavily relied on. Such platforms often enable the user to use one or more programming languages to create their application. This way more technical knowledge is required, but it has less restrictions than the *model-based* platforms.

Working in a PaaS-based environment has a few major advantages in comparison with the traditional way of development. Important advantages are (Richardson & Rymer, 2014):

- **Easy:** Often developing using a PaaS or SaaS solution is easier than traditional developing. PaaS providers are usually used for a specific niche of the the market. If you choose the right PaaS, the provider has added a lot of options and functionality that make the development of your specific kind of application easier. Also the PaaS providers have already made several decisions for the developers like network communications. Not having to deal with all this technical background makes the development more efficient and decreases the time-to-market of the applications made with these platforms.
- **Scalability**: Using PaaS also makes scalability less of an issue. Most providers offer their customers the ability to run their applications on the hardware of the provider as well. By doing this, if the application gets used a lot the provider will (automatically) scale up the hardware on which the application is running. This enables cloud providers to use the economy of scale of their computing power as many organizations (all with their different peak hours) use the computing power of the cloud provider. This enables the cloud provider to create this computing power less costly than if all organizations would create their own through the economy of scale.
- **Cost:** By using PaaS often payment is done only for what is actually used. there is no licence of a certain software package all year if you don't use it all year. Organizations don't have to have a lot of (expensive) servers running day and night to be able to cope with the peak hours during the day, as the PaaS providers will provide the scaling for you. This makes the cost

structure lean a lot more towards variable costs and less to constant costs, this also means that big investments are not needed as the application gains more usage, as payment is only done on the basis of what is used.

- **Continuous integration** Because most PaaS providers focus on enabling to build cloud based applications, they are structured in a way that relieves the programmer of taking care of the back-end of the system. This enables the programmes to work on a higher level of programming abstraction, which means that they are mainly working on functionality and less on complicated back-end instructions. This reduces the development cycle, which means that it takes less time to build an application. This is a very big advantage for using online platforms like these, because the market is demanding this speed of development. This is what is called continuous integration: the product is continuously, almost real-time adapted to the present needs of the business.(Marko, 2017)

Of course platform based development is not all positive, it has some negative sides as well (Lawton, 2008): vendor lock-in, reluctance to trust third party/internet with business critical application development or sensitive information.

- **Vendor lock-in:** As there are so many different PaaS providers, choosing the best one for a certain project is a difficult task to start with. Therewithal business have to keep future projects or elaborations of the same project in mind as well. If the choice for a certain platform is made, there is no easy way to change platforms and the same platform *has to be used* for the elaboration as well, otherwise the invested time and money in the platform/application are lost. This means the choice is either recreating the application in an other platform, thereby throwing away the made investment or staying with the same platform and be forced to work in an inefficient manner.
- **Trust:** Another negative side of an *online* PaaS platform is the reluctance of people to store/process important information on the internet. Businesses are not keen on storing sensitive information on the internet, as there is a lot of legal trouble when for example personal information of European citizens is stored outside of the European Union. Also users of the platform have less/no influence on the security measures taken by the

PaaS provider, they however will have to deal with the loss of face when data leaks. This data can be personal data which is bad but can also be business critical information which provides the organization with a competitive advantage over other businesses. This competitive advantage may decrease or even disappear if other businesses in the industry get access to this critical information.

Lindström (2011) tries to create a basic framework on which organization can base their security considerations in a systematic way, in this article it is pressed that the (security) requirements should always be checked to see if using a or a particular cloud service provider will not violate the requirements that have been set. If the requirements aren't met either the platforms should not be adopted or the security requirements should be changed.

He also states that it is best to start with non critical business processes in the cloud than to start transferring business critical ones at once. By starting with non-critical processes experience will be gained, which can be used in the future. It is less disastrous to make a mistake in a non-critical process.

2.3 concepts

In figure 1 is a diagram of the concepts that play a role in this match. We see that the *match* is central in the diagram. This match has *meaning*, the meaning of the match is that the situation and the platform work well together and that the platform fits well in the organizational situation the organization is in. On the left side of the diagram is the *platform* and on the right side is the *organizational situation*. The platform and the organizational situation both have *characteristics* that should be a good fit with each other. On the left side we see that the platform's characteristics consists of among other things the *functionality, costs* and *support* the platform provider provides.

On the right side of the diagram wee see that the characteristics of the organizational situation are the *IT landscape*, in the *present* form as well as how the IT landscape is expected to be in the *future*. Another characteristic of the organizational situation is the *goal* they have with acquiring a development platform. This goal could have a *operational* (short term), *tactical* (middle term) and a *strategical* (long term) component. The organizational situation also has cer-

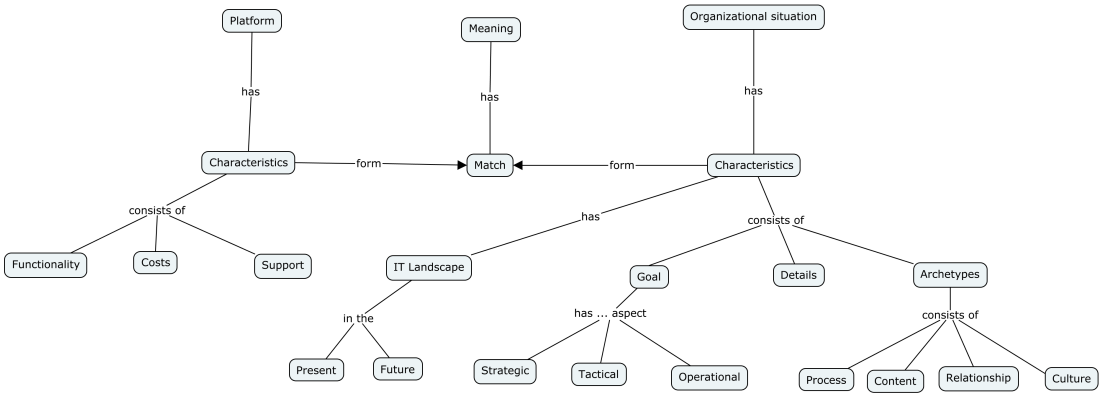

Figure 1: concepts relations

tain *archetypes*, this archetype is consists of the four components, being *process, content, relationship* and *culture*. More about these archetypes can be read in 4.1.2. Then only the *details* of the organizational situation are left, which covers everything else, like how much scalability is needed and how much focus is on development speed.

3 Method

As mentioned before the research question of this thesis is:
"What are the characteristics of organisational situations and the characteristics of IS platforms that can help to find the match between situations and platforms?"

In order to ensure that the whole research question we divide it in multiple sub-questions. When these are answered, the answer to the research question should be clear and answered in full.

We divide the research question in the following sub-questions:

1. (a) What is an IS platform?
   (b) What is an organisational situation?
2. What is the meaning of a match between an organisational situation and an IS platform?
3. (a) What are characteristics of an IS platform to consider in the platform selection process?
   (b) What are characteristics of an organisational situation to consider in the platform selection process?
4. (a) Do the characteristics we found support a match between an organisational situation and an IS platform?
   (b) How can this match be made

These sub-questions together will answer the research question in fully as definitions and context are made to ensure a strict scope of research, the characteristics that are important are determined and are tested when we find out how these will help in the selection process.

The first step of this research will be defining both the interpretation of an IS platform and an organisational situation (SQ1). In order to do so, online academic sources will be used to see what the generally used definitions are in the academic community. This is especially important in order to have a well defined research scope and that the same terminology is used in comparison with other academic findings, this will decrease the chance of confusion or misunderstanding the findings in this thesis.

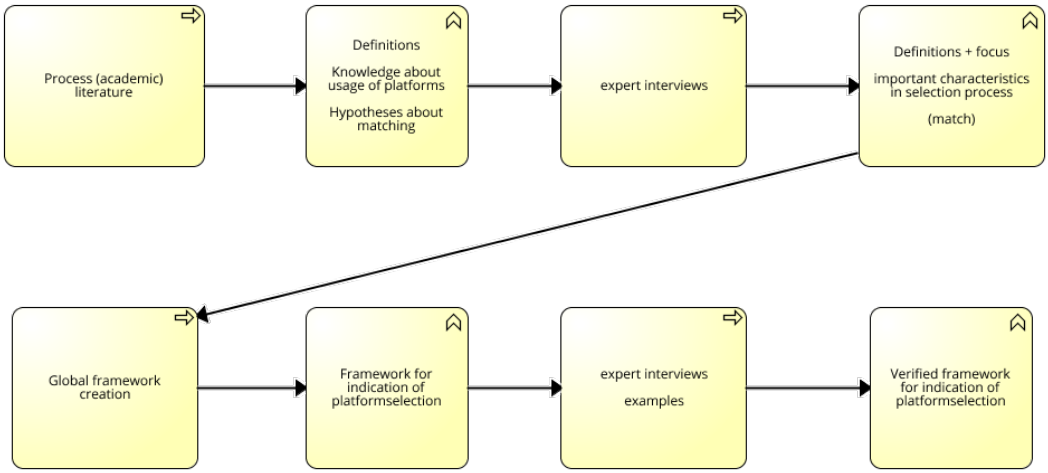

Figure 2: Method flow

After the concepts are clearly defined and the scope is set, hypotheses will be made with regards to the meaning of the match (SQ2). When can you say that there is a good or a bad match? Also the characteristics of the platform and situation (SQ3) will be searched for. Both academic and non-academic online sources will be used to do so. By using the non-academic online sources, we hope to get a better view on how the industry looks upon this topic as the industry experts are usually not very present in the academic world. Because using non-academic literature in an academic research can be tricky a lot of effort will be put in verifying the expertise of the author of the source to make sure that it is a serious source. This in combination with the results from the academic literature will give a good idea as to the characteristics that are important in the process of selecting a platform.

These hypotheses will be checked and elaborated upon in the expert interviews. These interviews will be mainly focussed on SQ2. We will try to find out when an organisation thinks they've made a good choice en when they think there is a mismatch. The interviews will be semi-structured, this will provide that the right topics are handled and that the interviews can deviate a bit from the topics if needed to get further information gain. The interviews will be recorded in order to ease the information processing.

When we are on the same page as the industry and are working on something that will actually help the industry, we will start the creation of a model which will state what kind of characteristics of situations match with the characteristics of platforms. By creating this model, the author is forced to apply the characteristics to situations. As the characteristics are used in the model the semantics of the characteristics and the model will be thoroughly checked. By using them in practice, they will be approached in a practical matter, which will provide the best results for the business.

This model will be checked in further interviews with experts from the industry to make sure this research is really reflecting the state of the industry instead of the opinion of one specific expert. The framework will be put prone to different business cases where there is a demand for a platform. The framework will then be tested to see what platform(s) will be recommended, finding these results and the recommendations that the framework makes will be topic of discussion in the interviews.

4 Analysis

4.1 Definitions

4.1.1 Platform

In this section it will be elaborated upon what kind of platforms are considered in this research. The focus will be laid down so that everything that is said is considered in the right context and confusion will be avoided as much as possible. In the section theoretical background there already is a definition of platforms. The focus of this thesis will be laid towards what is known as low-code platforms, because speaking about all PaaS platforms would be too broad and too complex for this master thesis. The basis of these platforms lays very close to Model Driven Development (MDD). MDD focusses on on separating the specification of the system functionality from the implementation of that functionality (Ormsc et al., 2001). This is done by creating models in order to understand, design, construct, deploy, operate, maintain and modify a system (Sharma & Sood, 2011). Forrester (Richardson & Rymer, 2014) made the first definition for low-code platforms that we know of:

"Platforms that enable rapid application delivery with a minimum of hand-coding, and quick setup and deployment, for systems of engagement."

This definition will be used as the basis, some focus will be added to be able to make the research more specific and manageable. As this is an exploratory research it's good to try the concept of the research on a smaller and more manageable set of platforms. The subset will consist of platforms which focus on developing information systems in the form of internal or external facing enterprise applications. This combined with the definition of platforms used in the theoretical background, will form the following definition of platforms considered in this research:

"An online environment that enables rapid application delivery with a minimum of hand-coding and provides opportunity for deploying, running and managing these applications."

Forrester (Richardson & Rymer, 2016) divides low-code platforms in five different categories using their background and **functionality**. These categories are: General purpose platforms, Process application platforms, Database application platforms, Request-handling platforms and Mobile application platforms.

In this thesis extra focus will be added by leaving out the last two of those categories, because those both have a too specific and distinctive functionality which will not fit nicely in the comparison. By filtering the platforms like this there will be a set of similar platform functionality but the platforms differentiate enough to make a reasoned choice for what platform fits the best in the particular situation. Besides that the request-handling platforms and mobile application platforms will be ignored, we'll make another modification to Forrester's categorization. Instead of setting up mutual exclusive categories as Forrester does, this thesis will use not mutual exclusive categories but will allow platforms to be a member of multiple categories to a certain extend. For example platform A is focussing on database applications, so they are involved in the database application platform category with a high score. But they support some basic process modelling as well, so it also is in the process application platform category but to a lesser extend. This allows us to score a platform on multiple focus points, instead of labelling them *general purpose* without any further specification.

As general purpose will not really help us in terms of scoring a platform, this category will be split in the other functionality of platforms that came forward in the interviews: Rule based application platforms and Document application platforms. This will bring our total of categories to the following four categories:

- Process application platforms
- Database application platforms
- Rule based application platforms
- Document application platforms

Obviously this is a very important characteristic to keep in mind while selecting a platform. These categories represent the main functionality of the application created with the platform. For example if the structure of a business process is to be improved, it would be wise to choose a process application platform over a database application platform. "The main functionality is often visible in

the history of the platform provider" says Wim van Stokkum in the interview "Platforms that are really process based are often from companies that are really process centred." The choice of this main functionality does not only relate to the nature of the project. It is also subject to the skills and preferences of the team. For instance Jilt Sietsma often has a data based view of the problem: "I'm not going to look at how the offer is moving through the process, I do not really care about the process and the work flow. I look at the data that is concerned." and "You start with making a data model, entities with attributes and relationships, and later on you add rules to the data. For instance: if the date of creation is 2 weeks ago, make a notifications with these characteristics."

The main functionality may be a very important characteristic to keep in mind when selecting a platform, but it is far from the only characteristic. In a previous chapter an explanation was given about the benefits of using a Cloud Platform in contrast to using a traditional development environment. A platform should be sufficient in providing these benefits. This means that **scalability** and **continuous integration** should also be included in the platform of choice (Haddad, 2011). Scalability is something that usually is covered in the back-end of a system, which means that the platform usually will take care of this problem, but it should be checked upon to make sure that it really is supported. Continuous integration is mostly about the speed of development. This is influenced by the speed in which a team can update the application. This is influenced by both the speed of development of the team and the way updates can be rolled out provided by the platform. If a platform supports the higher level of programming abstraction mentioned in the previous sector about platforms, the development team will be provided with the possible speed of development that is needed for continuous integration, as there is a lot of back-end managing that is done by the platform that will automatically be integrated in the application, removing this duty from the obligations of the development team, meaning the development cycle will shorten, improving the ability to quickly deliver the next version of the software. Jeffrey Kwee said: "People choose for using a platform because they want to develop faster, if you look at OutSystems or Mendix, in specific circumstances these platform can offer quicker change."

Of course there a lot more characteristics that need to be considered when selecting a platform. For example the current and predictions of the future **IT landscape** of an organization. With so many different platforms for different functionality, there is a risk of creating a spaghetti-infrastructure. If each division of an organization is able to use their own platform an organization could end up buying a lot of different platforms for different units of the organization, as when this decision would be centralised there could be a lot more knowledge sharing between developers. Developers would get the chance to specialise in a certain platform and the benefits could be extended. Problem with trying to centralise these decisions is that "buying a platform" is cheap enough that central management doesn't even have to approve the expense. This means there could arise a growth of the platform portfolio, where enterprise architecture professionals are unaware of, meaning they can not do their job properly, which is gaining strategical advantage using the benefits of the platforms and creating a platform portfolio where synergy between the different platforms can be formed, instead of all business units using their platform separate from each other.

Software procurement is a lot different for small organizations than it is for large organizations, looking at IT landscape we see a difference in selection method here, because the needs of small and large organizations are so different here.

Small organizations often start by buying a single platform with which they will try to solve multiple problems because they do not have the financial capacity to invest in multiple platforms for different functionalities. By having this need they would be interested in what Forrester calls *general purpose* platforms, translated into our framework that would be a platform that scores high on the multiple functionalities that the organization needs now and in the (near) future. This means that a smaller organization has to keep in mind both the functionality needed to solve the "immediate" problem they want to fix with a platform as well as the functionality needed to solve other problems within the organization. Not having to buy two separate platforms (including personnel, training) will be a huge financial relief for a smaller organization.

Larger organizations can afford to use multiple platforms for different functionality. The key thing they should take care of is that the platforms in their

portfolio complement each other instead of providing redundant functionality. It would be a waste to have two different platforms for displaying data from the ERP while this can be done with just one platform, using a single platform will mean less platform costs, less training costs and more knowledge sharing possibilities. As Jeffrey Kwee noted: "The cost tag often is pretty important." A well selected platform should add business value instead of just solving the immediate problem.

Most organizations already have an IT infrastructure before they start working with a platform. The applications that are built with the platform are built around the existing infrastructure. This makes it very important that the platform supports the use of **interactions** with other systems, as it often is an application to support the core infrastructure it should have easy ways of interacting with the core infrastructure. The other end of the application often needs to have interaction with a person, this can either be a client or an internal user, in the form of a web interface. This separation of functionality of the core application and side application is an aspect of a new trend in IT development: bimodal IT. Gartner (2015) defines this as:

"The practice of managing two separate, coherent modes of IT delivery, one focused on stability and the other on agility. Mode 1 is traditional and sequential, emphasizing safety and accuracy. Mode 2 is exploratory and nonlinear, emphasizing agility and speed"

This means that an organisation deploys two quite different strategies for different purposes. This is summed up nicely in Figure 3 from Horlach and Drews (2016). Platforms are mainly used in the mode 2 of bimodal IT. Because of the speed of development, possibility to work in small iterations and keep the business close in those iterations. In this way the exploration of the business' needs can be experimented upon and changed rapidly based on the feedback provided by the business/customers. This provides the opportunity to quickly see if an experiment is viable or not.

Of course the **cost** of a platform play a huge role in the selection process. As mentioned before this is a bit more relevant to the smaller organizations that are

Traditional IT (mode 1, industrial / core IT)		**Digital IT (mode 2, agile IT)**
Stability	*Goal*	Agility & speed
IT-centric	*Culture*	Business-centric
Remote from customer	*Customer proximity*	Close to customer
Performance and security improvement	*Trigger*	Short term market trends
Performance of services	*Value*	Business moments, customer branding
Security & reliability	*Focus of services*	Innovation
Waterfall development	*Approach*	Iterative, agile development
Systems of records	*Applications*	Systems of engagement
Slow	*Speed of service delivery*	Fast

Figure 3: Characteristics of mode 1 and mode 2

looking to buy a platform, but in the end the value for money has to be right, even for the bigger organizations.

Some platforms might be of a better quality or may have provided better for the desired functionality but if it is too expensive it might still be a bad choice.

Not every platform is equally present in different regions of the world. This also means that the **support** capabilities of the platform providers are not equally present in all regions of the world. We see certain platforms that are mainly active in North America, they will focus their support efforts mainly on North America because there is the most business for them. This might mean that there is worse support in other countries.

4.1.2 Organizational situation

Before we go into the details we should have a clear vision of what we consider to be the organizational situation. This is everything an organization can vary in, that influences how a platform is used in an organization, thus influencing the choice for a platform. We define the organizational situation as follows:

"The organizational situation consists of all characteristics of the situation

that an organization is in that can influence the choice for a platform"

The first thing that needs to be determined for the platform selection is the **goal** of the adoption of the applications. This can be goals like: improving efficiency, improving quality, decreasing costs, improve flexibility etc. An organization can have countless different reasons to adopt a platform. It is hard to directly use the goal as an aspect of choice for a specific platform, but it is good to keep this in mind while making the decision. For example if an organization wants to improve the uniformity of their help desk to improve the quality of the service they can provide, it might not be such an issue to use a more expensive platform in contrast to when an organization is trying to reduce as many costs from their help-desk to reduce overhead costs. Wouter van den Berg emphasizes that from a process point of view it is better to be involved from the beginning of the project: "It is bothersome to join a client halfway a project where they've already thought of a solution. Then you think if we could have thought with them from the beginning we would have offered a different solution using Blueriq than they thought of without knowing exactly what Blueriq can do."

There is a lot of difference concerning the goal of the acquiring of a model based platform whether the project started in the IT department or from the business itself. As Wouter van den Berg says in the interview: "If the demand comes from the IT they often ask for a BPM platform but they do not know that there are different categories of these platforms. If the demand comes from the business they often ask for specific functionality like: Can we make this specific call centre-interaction with it."

Then we're going to categorize the organizational situation on what van Stokkum (2016) calls the organizational **archetypes**. This is how the organizational situation is on four different aspects. For all four the aspects the situation is described to get a good view of how the organization works and what is important for the organization. The four aspects of the organizational situation are:

- *Process:* An important factor is the predictability of the process. If a process is very predictable, another platform might be preferred than when

a process is very unpredictable. A predictable process might just need an online form people can fill in and be provided with an answer. If a process is very unpredictable, just an online form will not do because there are too many factors to take into account. These unpredictable processes are often harder to fully automate, but the processes often can be made more efficient with the use of business rules platforms. Also there might be certain regulations in an industry. For example regulations about the documentation of the process that has to meet certain standards in the financial business.

- *Content:* Another major important aspect is the content that is subject to the new application. Is the required data always the same or do certain choices require different kinds of data and how does the process change by those choices. How much knowledge is needed to handle the data, is the data only used for a simple calculation or is deep expert knowledge needed to be able to handle the data? If more knowledge is needed to handle the data, it often is harder to automate it into an application but if it possible to do that, it can be more valuable than it is to automate a simple calculation.
- *Relationship:* The relationship the owner of the application has with the user of the application is important as well. For example if the application is for employees of an organization, a lot more knowledge and training can be assumed than if the application is used by customers. It makes sense that an application for applying for assisted living facilities should be easier to use than an application to check your study loan as young people are expected to be a lot more self-reliant in the digital world. But it is not only about what can be expected from the users. It is also how the application owner wants to treat the user. If there is a high level contact in a big organization, it is probably not desired to have them fill in an online form, instead personal contact is the preferred way of communication. This means that the application needs to support the process in a different way.
- *Culture:* With culture things that matter are things like the amount of trust the owner of the application has in the user of the application. Some things to consider are what is there to gain for the users of the application and what kind of fraud can they perform to have personal gains.

4.2 Match

4.2.1 Definition

It is hard to define a match between a platform and a organizational situation because it is an abstract thing. There isn't a real solution to solve this problem. Several platforms might fit in an organization, however some might be better than others, it is hard to say which one is best, partially because we can not see in the future or know how the situation would have evolved if a different platform was chosen. To be able to make an argued decision there has to be a decision making process that is as objective as possible. It is important to make the notion that a decision making process like this in this situation can never be truly objective because choices have to be made by humans and those choices can never be made 100% objectively. The process we will use to approach objectivity is Multiple Criteria Decision Analysis. This is a process in which decisions dealing with multiple criteria are analysed in an as objective as possible manner. MCDA tries to provide the option that maximizes welfare (Guitouni & Martel, 1997). They also point out that MCDA supposes that the situation is isolable and has strict boundaries, this is the only way to *calculate* what the best option is, as too much interference from factors that are not considered in the MCDA process, will make the calculation less reliable. In our case this is not fully correct as an organizational situation can never be captured in a limited amount of factors as there are limitless factors influencing the situation. Still we chose to use this technique as the answers it will provide do not have be viewed as the answer to the question: "Which platform to select?", but should be viewed as an substantiated ranking of options, which can be used as an indication of which options might be best, considering the situation.

Multiple Criteria Decision Analysis is a process that consists of multiple actions that need to be taken (Dodgson, Spackman, Pearman, & Phillips, 2009).

1. **Determine decision context:** First of all the decision context has to be determined. The context is needed for the decision because it provides value to the decision. Why is this decision being made? The decision has to have a goal, it has to be made to provide value to the organization. It is important to know how the choice is going to help the organization. Which aspects

of the organization are important in this decision? Also the stakeholders have to be determined, they might be able to help in the decision process. They should also be involved in the decision making process to improve involvement in the decision which is highly likely to improve the support for the decision, because they themselves were a part of the decision.

2. **Identify options:** Here all viable options are selected. Every option, in our case each platform that is on the list to be considered are searched for and selected for the scoring step, this can also be the combination of multiple platforms and creating synergy through that. Two platforms might complement each others functionality towards the level that it exceeds a singular, more complete, platform.
3. **Identify criteria:** Identifying the criteria means to find the characteristics on which the scoring will be done. The criteria selected should be all criteria that play a role in the decision. This might be just a little role, but everything that matters in the decision should be identified in this step. All the criteria together should give a complete view of the option as a whole. All stakeholders can be used to propose criteria that influence the decision to try to get an as complete as possible view of the option. This step might be visited later on in the process if something is forgotten in the first iteration. This completeness will make the goal and a complex thing like a platform more or less tangible and through this workable. The criteria can also be clustered under the different objectives of the process in order to ensure that the criteria are kept in a organized fashion.
4. **Score options on criteria:** In this step the actual scoring happens. All options are scored on how well they perform on all of the criteria. If an option scores poorly on a criterion, it will get a low grade. If an option scores well on a criterion it get a high grade. These scores can be obtained from actual measurements, but as this is not always possible it might also be an (argued) estimation or studies performed by third parties. It is important to score the criteria in such a way that they are comparable to the scores of other criteria in order to make the final analysis.
5. **Weight criteria:** In this step an analysing party can prioritize the criteria by putting a weight on all the criteria. If a criterion is very important to the organization it will give the criterion a very hight weight. This is where

the context determined in the first step comes in. To determine the priority of the criteria, one must know where the priorities of the organization lay, therefore the context determined before is used in the weighting process. It is possible that multiple sets of weights are made by different stakeholders in the decision process as for example the IT department might have different priorities than the board of directors.

6. **Combine weights and scores:** This is where the final scores of all the options are calculated. The scores are multiplied by the respective weights and those are all added together. By multiplying the higher prioritized criteria have a bigger share of the total final score and by adding them all together, there is an easy way to compare the options as a whole.
7. **Examine results:** Here the results are transformed in a way that they can be evaluated. The best way to do this is not to just come up with the option with the highest ranking but indicate the ranking of the options so people can see the scores and not just the best option. This provides a more complete image of the analysis than just pointing out the winner.
8. **Perform sensitivity analysis:** Then a sensitivity analysis is performed to see if the results make any sense and if nothing is neglected or prioritized wrongly. In this step priorities can be tweaked if some priorities are laid down wrongly. Looking at the separate weighted scores of criteria, one can notice that a certain criterion has more influence on the end result than it should, then the weight shall be lowered. What are the advantages and disadvantages of the good options? If all the results are to everybody's liking a recommendation can be made. In order to provide an as complete as possible image of the analysis, all the assumptions and uncertainties that are noticed in the whole MCDA are to be included in this recommendation.

5 Results

5.1 Interviews

For this research there we two different kinds of interviews held with four different experts. The experts are Wim van Stokkum, Wouter van den Berg, Jilt Sietsma and Jeffrey Kwee. An overview of the interviews can be found in table 1.

The first interview was held with all the experts and was about platform selection in general, the goal of this interview was to get a general idea of how platforms are treated in practise and how businesses approached them. Also the way industry is selecting which platform they're going to buy or use for their projects. The interview was semi-structured to make sure that the needed information was not to be forgotten about and the experts also had the room to add information they thought was relevant. There were two different sets of questions for the different experts. There was a set for the PaaS providers and a set for the PaaS procurement (the users). The following questions were the basis of the interviews.

PaaS provider

1. Do you have a specific target niche focussed for your platform.
   (a) What is the niche?
2. Why did(n't) you focus on a niche?
   (a) Was there a gap in market? Was there a gap in functionality of other platforms?
3. What are the needs of the niche/market? What kind of projects do you expect them to have?
4. How is the platform designed with these needs in mind?
   (a) What platform characteristics did you make/use/focus on?
5. How do customers (with certain characteristics) react to those specific characteristics in the platform?
   (a) Do the platform characteristics really match with situational needs?
6. How do customers (with certain characteristics) react to the platform as a whole?

PaaS procurement

1. Do you use (a set of) specific PaaS providers? Which?
2. Why do you use this specific platforms? Do they have specific characteristics that you're interested in?
3. What does your customer base look like? Big/small companies? How are their technical skills developed?
4. What (classes of) situations/projects do you expect with your customers?
5. How do you match a platform with a customer?
6. What are generally good/bad choices with situational characteristics and platform choice?
7. Do you have any specific examples of good/bad matches?

The second interview was only done with Wim van Stokkum and Jeffrey Kwee. By the time this interview took place, the first version of the framework was already created. The goal of this interview was to check if the concept of this framework was the right one and if the right results would be gotten. The interview was constructed of two parts.

The first part of the interview was filling the platform scores for a few platforms that the experts were familiar with. These scores would be the backbone of the framework and would eventually determine the scores of the platforms on the organizational situation. The overlapping platforms were PEGA, Mendix and OutSystems. Wim van Stokkum also provided the values for Blueriq, although it would be better to have a second opinion on those scores to decrease

Table 1: Interviews

Date	**Expert**	**Subject**	**Length**	**Remark**
07-12-2016	Wim van Stokkum	Platform selection	90 min	
22-12-2016	Wouter van den Berg	Platform selection	80 min	
05-01-2017	Jilt Sietsma	Platform selection	70 min	
06-01-2017	Jeffrey Kwee	Platform selection	70 min	
13-06-2017	Wim van Stokkum	Framework testing	60 min	
16-06-2017	Jeffrey Kwee	Framework testing	70 min	Telephone

the chance of those scores being consciously or unconsciously biassed, we made the decision to use these scores in the framework as well, because Mendix and OutSystems are very similar platforms and it is preferable to have a few different kinds of platforms in the framework. This provides the opportunity to test the framework in a better way than if there would only be two genres of platforms in the framework. The exact scores the experts provided can be found in appendix A

In the second part of the interview we took a look at some fictive cases. Before the interview three cases were prepared. All of those cases were focussed on different platforms, to test if the framework would come up with these/similar platforms. These cases are further elaborated upon in section 5.3. For these cases scores of the needs of the organizational situation were filled in in the framework. The framework made the calculations and the final scores came out. The scores will be discussed in section 5.3. All the scores can be found in appendix B.

5.2 Matching Framework

Table 2 is the framework used for scoring the situation and the platforms. The framework consists of multiple columns.

1. **Characteristic:** The name of the characteristic that is being scored.
2. **Score:** This is the score that the customer can fill in to indicate the situational need. If someone for example wants a platform with a high development speed, they should score *Development speed*-characteristic with a high score. In table 2 this column is filled with the possible scores of that characteristic.
3. **Weight:** This metric indicates the priority of the characteristic. If some characteristic has a high weight, the influence of that characteristic on the result will be higher, due to the multiplication of the score and weight. We recommend that users of the framework modify the weights as this is a big factor of the final score and this is very different for every situation. For instance some companies might have a lot less financial means than others, they should increase the weight of the *Cost*-characteristic.
4. **Platform score:** The score a specific platform scored for this characteristic, this will be filled in by experts that have an understanding of how

the platforms work and what their advantages and disadvantages are with respect to other platforms.

5. **Platform score*Weight*Score:** This is the platform score multiplied with the weight and the score. This calculation is made for all the different platforms that are taken into consideration. If this amount is added up for a specific platform, the total score will be acquired.

In the rows of the framework are the different characteristics that the situation and the platforms are scored on. The following are all the characteristics.

- **Process enablement:** This metric is about how much working with processes is enabled. Things to take into consideration here are: is it possible to model out your processes? how easy is it to transform those processes into working functionality?
- **Data enablement:** Data enablement is about how much working with data is being enabled by the platform. Can you create a data model? What can you do with these models?
- **Rule enablement:** This is about the enablement of business rules in the platform. Is it possible to create business rules? What other functionality do you get with these rules?
- **Document enablement:** This is for how much the platform enables the use of documents. Often when this metric is high the documents are the centre of the application. Documents are changed or added to and sent on into the system again for further use.
- **Development speed:** This is about how fast the development goes. How long does it take to create an application using this platform? A high score means a high development speed.
- **System integration:** This metric is about how much external interaction is supported by the platform. How easy is it to connect with other systems? How many possibilities are supported to interact with external systems.
- **Variable cost system:** This is about how the cost system of the platform is set up. Are there a lot of variable costs (c.q. do organizations only pay for what they use)? Or are there a lot of constant costs?
- **Technically unique:** This metric is about how much technically different things are possible with the platform. Or can you basically only make the

same thing in a different skin with different data?

- **Non-technical knowledge only:** How far can a person with limited technical knowledge get with the platform? Is it possible for someone to make a simple application with it?
- **Business model unique:** How many different kind of applications can someone make with the platform?
- **Scalability:** This is about how well the applications created by the platform scale to many users.
- **Version control (reversibility):** Is there a version control system in place? Can the application be rolled back to a previous state and continue working.
- **Support:** How well is the support organized? How easy is it to get help with the development in the platform from the provider.?
- **Costs:** This metric is about the total costs of using the platform. A high score means that the costs are low.

Table 2: Matching framework

Characteristic	**Score**	**Weight**	**Platform score**	**Platform score*Weighted*Score**
Functionality				
Process enablement	1-5	7		
Data enablement	1-5	7		
Rule enablement	1-5	7		
Document enablement	1-5	7		
Cloud characteristics				
Development speed	1-5	3		
System integration	1-5	3		
Scalability	1-5	2		
Variable cost system	1-5	2		
Development				
Technically unique	1-5	3		
Non-technical knowledge only	1-5	3		
Business model unique	1-5	3		
Beauty				
Version control (reversibility)	1-5	1		
Support	1-5	2		
Costs	1-5	5		
			SUM	**SUM**

5.2.1 Modifiability

As we give openness about how the framework works and all the calculations are quite easy and understandable, the framework is very flexible towards being changed if desired, we recommend at least changing the weights, because priorities with different situations are always different as well. But someone can change a whole lot more to the framework if desired, for instance someone could change the scoring scales. We chose for a 1-5 scoring scaling because it offers enough diversity between the different scores, this way meaning can be given towards the different scores. One has to keep in mind thought, that if not all the scores are changed in the same way the impact that the characteristics have will change as well. Basically if the possible score goes up, intrinsically the weight has gone up with it. For example if the score of 1-3 is changed to 1-7, an average situation would score 2 on the original score and 4 on the modified score, so basically the weight has been intrinsically doubled. To prevent this problem from occurring is to either change all the scores in the same way or to modify the weight so that the score has a similar impact, in the example given above the weight would have to been halved.

But this is not everything that can be changed, people can leave out characteristics if they want to. People might also want to add certain characteristics and they can if they know the platform values for these characteristics themselves. Another thing users might want to change is the platform scores on the

characteristics even though they are provided by experts. Although this of course should be done with care as these scores are carefully set by experts.

5.3 Proof of Concept

5.3.1 Cases

Case 1: PEGA

The first case is a case of Rabobank, one of the big banks in the Netherlands. The bank noticed big technological innovation taking place in the banking system. They think that if they want to keep up with this innovation they need to make more applications. These systems will be linked to their ERP, in order for it to have access to the data used in these applications. The applications they want to add are pretty complicated and mostly data and rule based. They have enough financial means to train people in the use of the platform. They already have employees that are trained in modelling. It is important for the bank that the applications are scalable, because they have about 7.5 million clients. The situation can be summed up as follows:

1. Big organization: Rabobank
2. Want to add more applications to their ERP
3. Data and rule based
4. Enough budget to train experts internally
5. They have modelling experts
6. Needs to be scalable

Case 2: Blueriq

This case is about the Dutch tax authorities. They have enough financial needs but they want don't want to spend too much on the project, because there is some pressure from the government to keep the costs low, because of the expensive, failed government projects. They don not have the technical knowledge yet, but that can be trained. They want to make an application that will help start-ups to determine how much tax they have to pay, which is a very knowledge-intensive application.

1. Dutch tax authorities
2. Have medium budget

3. Technical knowledge can and have to be trained
4. Decision support for start-ups.
5. knowledge-intensive

Case 3: Mendix

This is the case of a start-up called SpaceShare, which rents office and meeting space to people. They want to create an app-like self service application, which enables their customers to rent spaces and create monthly invoices. This is mainly data based and a bit document based for the invoices. Development speed is pretty important for them, because currently this takes up a lot of time and effort of their employees and customer often complain that it takes too much effort to reserve space. Costs are pretty important for SpaceShare as they don't have a lot of financial means yet.

1. Start-up: SpaceShare
2. Space-renting
3. Self service application
4. Technical knowledge can be trained
5. Mostly data based
6. Development speed has some priority
7. Costs are important

5.3.2 Case results

In the interview with Wim van Stokkum we scored the platforms: Blueriq, PEGA, Mendix, Filenet and Drools (a simple open-source rule engine). He later provided the scores on OutSystems by email on request.

Case 1 (focussed on PEGA) resulted in the framework recommending PEGA with a score 17% higher than Blueriq (the second highest score).

The second case (focussed on Blueriq) resulted in the framework recommending PEGA followed by Blueriq. PEGA was 9% higher than Blueriq.

The third case (focussed on Mendix) resulted in the framework recommending Mendix followed by PEGA. Mendix only had a 1% higher score than PEGA. In this case there was quite some focus on costs and with PEGA being very expensive, the customer should probably not take PEGA into consideration, simply

because they can not afford it. Also PEGA got high scores on support, but they mainly have good support in the United States, so a European organization could choose to lower the support score for PEGA, which would induce a decrease in PEGA's total score.

In the interview with Jeffrey Kwee we scored the platforms: Mendix, OutSystems, BeInformed and PEGA. During the interview we decided to change some things. We changed some names of the characteristic, either to clarify them or to make them a better fit with the concept that was behind the name. We also decided to take out the characteristics *Uses engine*, which was a score on how good the engines of the different platforms were, and *Supports evolutionary applications*, because for a customer this does not really matter when making a choice for which platform to acquire.

In the first case, focussed on PEGA, the framework recommended PEGA followed by Mendix and then OutSystems. PEGA was 4% higher than Mendix and 7% higher than OutSystems.

In the second case, focussed on Blueriq (which was not scored by Jeffrey) the framework recommended using PEGA followed by Mendix. PEGA was 11% higher than Mendix. We considered this a good result because PEGA is the platform that comes the closest to Blueriq.

In the third case, focussed on Mendix, the framework recommended Mendix followed by OutSystems. Mendix was 6% higher than OutSystems.

Then we took the platform scores together by averaging the scores both experts gave and looked at the results again. We left Blueriq, PEGA, Mendix and OutSystems in the framework for scoring.

In the first case, focussed on PEGA, PEGA was recommended by the framework, followed by OutSystems. PEGA scored 9% higher than OutSystems.

In the second case, focussed on Blueriq, PEGA scored the highest followed by Blueriq and OutSystems. PEGA scored 9% higher than Blueriq and 12% higher than OutSystems.

In the third case, focussed on Mendix, Mendix ranked the highest followed by OutSystems. Mendix scored 8% higher than OutSystems.

If we look at the results we got out of the framework we see that in case 1, that is focussed on PEGA in all three different frameworks (Wim van Stokkum's, Jeffrey Kwee's and the combined framework) PEGA is the highest ranking platform, which is a good result because it is the platform we hoped to have the highest ranking.

In the second case PEGA was recommended in all different frameworks as well, this is because PEGA is a very complete platform but it is very expensive. This results in PEGA having equally high or higher scores on every characteristic than Blueriq except for the cost aspect. This makes it very hard for a platform like Blueriq to compete with PEGA in this framework. This is were it becomes visible that the MCDA framework will not provide the one true answer of what is the best platform, but will give an indication of what platforms would work well in the particular situation. This is where a human comes into play to make the right decision, the person could just exclude PEGA because of the price difference in combination with the not that much higher score PEGA gets and they would select one of the following platforms.

In the third case, focussed on Mendix, all three frameworks resulted in Mendix getting the highest score, followed by OutSystems. This is what we would have expected because the case was focussed on Mendix, and OutSystems is a similar platform so would gain a high score as well.

Overall this means that the MCDA framework is working pretty well. The platforms that should gain high scores are generally ranked first or in Blueriq's case second. With the help of a human person in the selection process, which always should be included, the right choice can be made more easily than if that person did not have the framework to back them up in the selection process. First of all investigating all platforms would take a lot of time, now a person can focus on the highest scoring platforms to investigate if that platform would fit their situation.

6 Conclusions

The general opinion is that platforms can help to simplify making applications and improving efficiency. This improvement in efficiency enables the idea of continuous integration by being able to produce a business application in matter of weeks instead of months. This in turn enables organizations to test a first concept of an application often within a few days or a week. In this way an organization can test that concept before building the real application. If traditional programming was used, building this proof of concept alone would take a few weeks. In the popularization of bimodal we see that (model based) PaaS systems are generally used to make mode 2 systems, because of the speed and flexibility the platforms provide, which is exactly the core goal of these mode 2 applications. Using model based platforms is also a way of staying close to the business as the development cycle progresses. As Jeffrey Kwee said it: *"Mendix developers need a different skill set, they need to be closer to the business and more communicative"*. He also mentioned that while developing in a model based platform it is not uncommon to take the model to the customer if there is uncertainty about something. And you can change the model in the customers office and show them the result of the change. This is also the closeness to the business he meant platform developers need and is central in mode 2 application development. There is also a trend in increasingly complex application moving over to cloud platform development/-mode 2 development. As Wouter van den Berg said: "These platforms are used more for the primary process of the company. Of course there is functionality that is always needed, like an integration of administration systems in the ERP or archiving documents. But the composing of this generic functionality to your primary process is what a platform is used for."

If an organization decided to use platform based development for a certain project, the decision of which platform they would use, often does not get the attention it deserves. Often a platform is selected because someone in the management heard of it or there is someone who is fond of for example Microsoft and pushed the use of a Microsoft platform, while that platform might not be the best fit in the organizational situation. By choosing the right platform and thus creating a good fit, the development costs could drastically decrease. So

if an unfitting platform is chosen, the development costs are likely to be higher than necessary because an application is developed in a illogical way, this will probably not only increase the costs, but also decrease the quality and maintainability of the product. This is because if the right platform is selected, you can use the platform in the way it is supposed to be used, which is a lot more logical and therefore a lot more understandable for other people (who might need to maintain the product). Choosing the right platform is an important step of the preparation of an project and is currently being undervalued, this should be changed.

As we knew that creating a platform selection tool with one "perfect" solution was not going to work, both because there are too many platforms to be able to do this and because organizational situations and the platforms themselves are to complex, resulting in a tool never being able to take all characteristics into consideration. That is why we decided to only take the most important characteristics and use them to determine a score. In this way an organization would get an "opinion" on every platform taking the situation in consideration, instead of coming up with one platform as an answer to the situational needs. A scoring system can also be used more as an advice instead of a solution, this is preferred because the complexness of the decision can never be fully captured in the tool.

An organization should first thoroughly analyse their situation. What are exactly the needs they have for the platform? What characteristics should the platform exceed in and what are characteristics that are less important? If the organization has finished analysing their situation, this can be converted into scores and weights that can be put into the framework. The framework will then come up with what platforms match their situation. In our example cases we saw that the framework is capable of selecting the platforms that would fit the situation, so the proof of concept is successful. It may need some tweaking to make it more accurate and to prevent it from things like always scoring platforms like PEGA quite high.

While creating the MCDA model we came across two difficulties we needed to overcome. We needed to fill the model with values of the different platforms

and we needed to find a way to find out the weights that we were going to use.

Platforms providers need to provide more openness to what their platform is all about. Most of the platforms only note the general characteristics of an PaaS platform and the benefits of working in the cloud. By looking through a lot of PaaS websites we find that the most used words are: *fast, easy* and *flexible*. All of these are just non-distinctive cloud characteristic buzzwords. The industry should provide the information that is important to the procurement staff of organizations, e.g. What the main functionality of the platform is, is it an platform that specializes in business rules management or a data driven platform which can be used to link a database to a business application? If this information is be provided, organizations can more easily substantiate their selection process and start by researching only the platforms that specialize in the kind of applications they need to have. Of course this would not superfluous the selection process but at least organizations would have a head start.

The weights are in a different category of problems. As the weights should be customized to the organizational situation, this should be determined by the party that is going to make the decision. For example if an organization's priority lays with the cost of the platform, this could be the case in for instance a start-up with limited financial resources, the weight of the cost characteristic should be higher than an organization that does not have to worry about the costs of the platform that much. This also has to do with the MCDA result not being a perfect result and that the highest ranking platform is not necessarily the perfect platform for the situation. Because of this customization of the weights of the MCDA model, it is important to have a clear view of the organizational situation before MCDA is used, this is the only way to make sure the weights are correct for the situation an organization is in.

7 Future research

Important to know, for readers, is that the research done in this thesis is just a proof of concept. So it is not at all a platform selector, it is merely a tool that will help understand what kind of platform would be a good match and what kind of platform would not fit that well. That doesn't mean that the project would a definite failure if a platform is used which has a low score and it also does not guarantee that if a project is done with a platform with a high score it will be a great success.

Also the model needs to be validated with more than just the few platforms that are in the system now. The model may work well with just a few platforms, but what happens when the model consists of dozens or even hundreds of platforms, will the scores still give a significant difference between each other or will it just be a cluster of scores that is gradually decreasing. If this would be the case, how bad would this be, if it still supports the decision making process, it still isn't a bad thing per se. However if this is the case, research should be done to what part of the top platforms are really a good fit. It needs to be determined where the boundary goes from good to neutral platforms.

The model should also depend on a more objective way of scoring the platforms. The platforms for instance should be able to proof that they provide in certain measures to be able to get the matching scores for the different characteristics. For example the process based functionality should support a certain amount of process modelling languages to earn points in the process based functionality characteristic. If this is neglected and the model would be used on a bigger scale, platforms could easily try to get higher scores than they should be getting by fiddling with their scores. Also an objective way of scoring prevents experts of being biassed, they may (un)consciously give platforms they prefer themselves a higher score. By having an objective way of scoring the chance of this will decrease.

8 References

Bhardwaj, S., Jain, L., & Jain, S. (2010). An Approach for Investigating Perspective of Cloud Software-as-a-Service (SaaS). *International Journal of Computer Applications*, *10*(2), 44–47. doi: 10.5120/1450-1962

Clemons, E., Reddi, S., & Row, M. (1993). *The impact of IT on the organiztion of economic activity - the move to the middel hypothesis.pdf* (Vol. 10). doi: 10.1016/S1873-1503(06)01001-4

Daneshgar, F., Low, G. C., & Worasinchai, L. (2013). An investigation of 'build vs. buy' decision for software acquisition by small to medium enterprises. *Information and Software Technology*, *55*(10), 1741–1750. Retrieved from `http://dx.doi.org/10.1016/j.infsof.2013.03.009` doi: 10.1016/j.infsof.2013.03.009

Dodgson, J. S., Spackman, M., Pearman, A., & Phillips, L. D. (2009). *Multi-criteria analysis : a manual* (Vol. 11) (No. 1-3). Retrieved from `http://eprints.lse.ac.uk/12761/1/Multi-criteria{_}Analysis.pdf` doi: 10.1002/mcda.399

Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *6*(May), 1–12. Retrieved from `http://ai.kaist.ac.kr/{~}jkim/cs489-2007/Resources/DeliveringSWasaService.pdf` doi: 10.1021/cr068365a

Finkelstein, A., Ryan, M., & Spanoudakis, G. (1996). Software package requirements and procurement. *Proceedings of the 8th International Workshop on Software Specification and Design*(1984), 141–145. Retrieved from `http://discovery.ucl.ac.uk/153752/` doi: 10.1109/IWSSD.1996.501156

Gartner. (2015). *It glossary - bimodal it.* Retrieved 20-02-2017, from `http://www.gartner.com/it-glossary/bimodal`

Guitouni, A., & Martel, J.-m. (1997). Some Guidelines for Choosing an MCDA Method Appropriate to a Decision Making Context. , 1–29.

Haddad, C. (2011). White Paper Selecting a Cloud Platform : A Platform as a Service Scorecard White Paper About the Author. , *06*, 1–28.

Hilgendorf, K., & DeBeasi, P. (2016). *2017 Planning Guide Overview: Architecting a Digital Business With Sensing, Adapting and Scaling.* Gartner. Retrieved 25-11-2016, from `https://www.gartner.com/doc/3471558`

?refval={&}pcp=mpe

Horlach, B., & Drews, P. (2016). Bimodal IT : Business-IT alignment in the age of digital transformation Bimodal IT : Business-IT Alignment in the Age of Digital Transformation. *Multikonferenz economic computer science (MKWI)*(April), Ilmenau, Germany. Retrieved from https://www.researchgate.net/publication/287642679{_}Bimodal{_}IT{_}Business-IT{_}alignment{_}in{_}the{_}age{_}of{_}digital{_}transformation

Hung, P., & Low, G. C. (2008). Factors affecting the buy vs build decision in large Australian organisations. *Journal of Information Technology*, *23*(2), 118–131. doi: 10.1057/palgrave.jit.2000098

Kauffman, R. J., & Tsai, J. Y. (2009). The Unified Procurement Strategy for Enterprise Software: A Test of the "Move to the Middle" Hypothesis. *Journal of Management Information Systems*, *26*(2), 177–204. doi: 10.2753/MIS0742-1222260208

Lawton, G. (2008). Developing Software Online with Platform-as-a- Service Technology. *Computer*, *41*(6), 13–15.

Lindström, J. (2011). Areas and problems to consider within information security and digital preservation during procurement and use of cloud services.

Marko, K. (2017). *To choose the right paas vendor, know thyself.* techtarget. Retrieved from http://searchmicroservices.techtarget.com/feature/To-choose-the-right-PaaS-vendor-know-thyself (Accessed 2017-03-24)

Mital, M., Pani, A., & Ramesh, R. (2014). Determinants of choice of semantic web based Software as a Service: An integrative framework in the context of e-procurement and ERP. *Computers in Industry*, *65*(5), 821–827. Retrieved from http://dx.doi.org/10.1016/j.compind.2014.03.002 doi: 10.1016/j.compind.2014.03.002

Ormsc, A. B., Burt, C., Dsouza, D., Duddy, K., Kaim, W. E., Frank, W., ... Wood, B. (2001). Model Driven Architecture (MDA) Document number ormsc / 2001-07-01. , 1–31.

Richardson, C., & Rymer, J. R. (2014). *New Development Platforms Emerge For Customer-Facing Applications* (Tech. Rep.).

Richardson, C., & Rymer, J. R. (2016). *Vendor Landscape: The Fractured, Fertile Terrain Of Low-Code Application Platforms* (Tech. Rep.). Retrieved from ForresterResearchdatabase

Sharma, R., & Sood, M. (2011). Cloud SaaS and Model Driven Architecture. *International Conference on Advanced Computing and Communication Technologies*, *11*(August), 978–981.

Taylor, M. (2016). *18 crm statistics you need to know for 2017.* SuperOffice. Retrieved from http://www.superoffice.com/blog/crm-software-statistics/ (Accessed 2017-04-16)

van Stokkum, W. (2016, December 7). personal communication.

Wenzel, S., Faisst, W., Burkard, C., & Buxmann, P. (2012). New Sales and Buying Models in the Internet: App Store Model for Enterprise Application Software. *Multikonferenz Wirtschaftsinformatik*, 639–651.

A Platform scores

Table 3: Wim van Stokkum's platform scores

	Blueriq	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	4	4	5	3
Data based	2	3	2	2
Rule based	5	5	1	4
document based	1	1	1	2
Cloud characteristics				
Development speed	3	4	5	4
Supports external interaction	4	4	4	4
Variable cost system	2	3	3	2
Uses engine	2	5	1	4
Development				
Technical knowledge (possible) inhouse	3	4	2	3
Non-technical knowledge only	2	2	4	2
Business model unique	4	5	3	3
Supports evolutionary applications	3	4	1	4
Beauty				
Scalability	4	5	4	4
Version control (reversibility)	4	3	2	3
Support	2	5	4	2
Costs	3	1	3	3

Table 4: Jeffrey Kwee's platform scores

	Mendix	**OutSystems**	**BeInformed**	**PEGA**
Functionality				
Process enablement	4	4	3	4
Data enablement	4	4	1	3
Rule enablement	3	3	5	5
Document enablement	3	4	3	4
Cloud characteristics				
Development speed	5	5	3	4
Supports systems integration	3	4	2	4
Variable cost system	2	3		
Uses engine				
Development				
Technical knowledge (possible) inhouse	4	5	2	4
Non-technical knowledge only	4	3	4	2
Business model unique	4	5	3	4
Beauty				
Scalability	4		1	4
Version control (reversibility)	4	4	3	5
Support	3	3	3	4
Costs	4	3		1

Table 5: Combined scores

	Blueriq	**PEGA**	**Mendix**	**Outsystems**
Functionality				
Process enablement	4	4	4.5	3.5
Data enablement	2	3	3	3
Rule enablement	5	5	2	3.5
Document enablement	1	2.5	2	3
Cloud characteristics				
Development speed	3	4	5	4.5
Supports external interaction	4	4	3.5	4
Scalability	4	4.5	4	4
Variable cost system	2	3	2.5	2.5
Development				
Technically unique	3	4	3	4
Non-technical knowledge only	2	2	4	2.5
Business model unique	4	4.5	3.5	4
Beauty				
Version control (reversibility)	4	4	3	3.5
Support	2	4.5	3.5	2.5
Costs	3	1	3.5	3

B Cases

B.1 Case 1: Rabobank

Table 6: Situational needs scoring

	needs score	**weight**	**weight*score**
Functionality			
Process based	2	4	8
Data based	5	7	35
Rule based	5	7	35
Document based	2	4	8
Cloud characteristics			
Development speed	3	4	12
Supports external interaction	4	4	16
Scalability	5	3	15
Variable cost system	1	1	1
Development			
Technical knowledge (possible) inhouse	5	3	15
Non-technical knowledge only possible in house	2	3	6
Business model unique	4	3	12
Beauty			
Version control (reversibility)	3	1	3
Support	2	2	4
Costs	1	5	5

Table 7: Wim van Stokkum's scores

	Blueriq	**PEGA**	**Mendix**	**Outsystems**
Functionality				
Process based	32	32	40	24
Data based	70	105	70	70
Rule based	175	175	35	140
Document based	8	8	8	16
Cloud characteristics				
Development speed	36	48	60	48
Supports external interaction	64	64	64	64
Variable cost system	2	3	3	2
Uses engine	8	20	4	16
Development				
Technical knowledge (possible) inhouse	45	60	30	45
Non-technical knowledge only possible in house	12	12	24	12
Business model unique	48	60	36	36
Supports evolutionary applications	12	16	4	16
Beauty				
Scalability	60	75	60	60
Version control (reversibility)	12	9	6	9
Support	8	20	16	8
Costs	15	5	15	15
	607	**712**	**475**	**581**

Table 8: Jeffrey Kwee's scores

	Mendix	**OutSystems**	**BeInformed**	**PEGA**
Functionality				
Process based	32	32	24	32
Data based	140	140	35	105
Rule based	105	105	175	175
Document based	24	32	24	32
Cloud characteristics				
Development speed	60	60	36	48
Supports external interaction	48	64	32	64
Variable cost system	2	3	0	0
Uses engine	0	0	0	0
Development				
Technical knowledge (possible) inhouse	60	75	30	60
Non-technical knowledge only possible in house	24	18	24	12
Business model unique	48	60	36	48
Beauty				
Scalability	60	0	15	60
Version control (reversibility)	12	12	9	15
Support	12	12	12	16
Costs	20	15	0	5
	647	**628**	**452**	**672**

Table 9: Combined scores

	Blueriq	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	32	32	36	28
Data based	70	105	105	105
Rule based	175	175	70	122.5
Document based	8	20	16	24
				0
Cloud characteristics				0
Development speed	36	48	60	54
Supports external interaction	64	64	56	64
Scalability	60	67.5	60	60
Variable cost system	2	3	2.5	2.5
Development				
Technical knowledge (possible) inhouse	45	60	45	60
Non-technical knowledge only possible in house	12	12	24	15
Business model unique	48	54	42	48
				0
Beauty				0
Version control (reversibility)	12	12	9	10.5
Support	8	18	14	10
Costs	15	5	17.5	15
	587	**675.5**	**557**	**618.5**

B.2 Case 2: Tax authorities

Table 10: Situational needs scoring

	needs score	**weight**	**weight*score**
Functionality			
Process based	2	4	8
Data based	2	4	8
Rule based	5	9	45
Document based	2	4	8
Cloud characteristics			
Development speed	4	3	12
Supports external interaction	2	3	6
Scalability	5	3	15
Variable cost system	2	1	2
Development			
Technical knowledge (possible) inhouse	5	3	15
Non-technical knowledge only possible in house	2	3	6
Business model unique	3	1	3
Beauty			
Version control (reversibility)	4	3	12
Support	2	2	4
Costs	4	2	8

Table 11: Wim van Stokkum's scores

	Blueriq	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	32	32	40	24
Data based	16	24	16	16
Rule based	225	225	45	180
Document based	8	8	8	16
Cloud characteristics				
Development speed	36	48	60	48
Supports external interaction	24	24	24	24
Variable cost system	4	6	6	4
Uses engine	4	10	2	8
Development				
Technical knowledge (possible) inhouse	45	60	30	45
Non-technical knowledge only possible in house	12	12	24	12
Business model unique	12	15	9	9
Supports evolutionary applications	12	16	4	16
Beauty				
Scalability	60	75	60	60
Version control (reversibility)	48	36	24	36
Support	8	20	16	8
Costs	24	8	24	24
	570	**619**	**392**	**530**

Table 12: Jeffrey Kwee's scores

	Mendix	**OutSystems**	**BeInformed**	**PEGA**
Functionality				
Process based	32	32	24	32
Data based	32	32	8	24
Rule based	135	135	225	225
Document based	24	32	24	32
Cloud characteristics				
Development speed	60	60	36	48
Supports external interaction	18	24	12	24
Variable cost system	4	6	0	0
Uses engine	0	0	0	0
Development				
Technical knowledge (possible) inhouse	60	75	30	60
Non-technical knowledge only possible in house	24	18	24	12
Business model unique	12	15	9	12
Beauty				
Scalability	60	0	15	60
Version control (reversibility)	48	48	36	60
Support	12	12	12	16
Costs	32	24	0	8
	553	**513**	**455**	**613**

Table 13: Combined scores

	Blueriq	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	32	32	36	28
Data based	16	24	24	24
Rule based	225	225	90	157.5
Document based	8	20	16	24
Cloud characteristics				
Development speed	36	48	60	54
Supports external interaction	24	24	21	24
Scalability	60	67.5	60	60
Variable cost system	4	6	5	5
Development				
Technical knowledge (possible) inhouse	45	60	45	60
Non-technical knowledge only possible in house	12	12	24	15
Business model unique	12	13.5	10.5	12
Beauty				
Version control (reversibility)	48	48	36	42
Support	8	18	14	10
Costs	24	8	28	24
	554	**606**	**469.5**	**539.5**

B.3 Case 3: SpaceShare

Table 14: Situational needs scoring

	needs score	**weight**	**weight*score**
Functionality			
Process based	3	7	21
Data based	3	7	21
Rule based	1	3	3
Document based	0	2	0
Cloud characteristics			
Development speed	4	5	20
Supports external interaction	3	3	9
Scalability	3	2	6
Variable cost system	2	2	4
Development			
Technical knowledge (possible) inhouse	4	3	12
Non-technical knowledge only	3	3	9
Business model unique	1	3	3
Beauty			
Version control (reversibility)	2	1	2
Support	2	2	4
Costs	5	5	25

Table 15: Wim van Stokkum's scores

	Blueriq	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	84	84	105	63
Data based	42	63	42	42
Rule based	15	15	3	12
Document based	0	0	0	0
Cloud characteristics				
Development speed	60	80	100	80
Supports external interaction	36	36	36	36
Variable cost system	8	12	12	8
Uses engine	8	20	4	16
Development				
Technical knowledge (possible) inhouse	36	48	24	36
Non-technical knowledge only possible in house	18	18	36	18
Business model unique	12	15	9	9
Supports evolutionary applications	12	16	4	16
Beauty				
Scalability	24	30	24	24
Version control (reversibility)	8	6	4	6
Support	8	20	16	8
Costs	75	25	75	75
	446	**488**	**494**	**449**

Table 16: Jeffrey Kwee's scores

	Mendix	**OutSystems**	**BeInformed**	**PEGA**
Functionality				
Process based	84	84	63	84
Data based	84	84	21	63
Rule based	9	9	15	15
Document based	0	0	0	0
Cloud characteristics				
Development speed	100	100	60	80
Supports external interaction	27	36	18	36
Variable cost system	8	12	0	0
Uses engine	0	0	0	0
Development				
Technical knowledge (possible) inhouse	48	60	24	48
Non-technical knowledge only	36	27	36	18
Business model unique	12	15	9	12
Beauty				
Scalability	24	0	6	24
Version control (reversibility)	8	8	6	10
Support	12	12	12	16
Costs	100	75	0	25
	552	**522**	**270**	**431**

Table 17: Combined scores

item	**Blueriq**	**PEGA**	**Mendix**	**OutSystems**
Functionality				
Process based	84	84	94.5	73.5
Data based	42	63	63	63
Rule based	15	15	6	10.5
Document based	0	0	0	0
Cloud characteristics				
Development speed	60	80	100	90
Supports external interaction	36	36	31.5	36
Scalability	24	27	24	24
Variable cost system	8	12	10	10
Development				
Technical knowledge (possible) inhouse	36	48	36	48
Non-technical knowledge only	18	18	36	22.5
Business model unique	12	13.5	10.5	12
Beauty				
Version control (reversibility)	8	8	6	7
Support	8	18	14	10
Costs	75	25	87.5	75
	426	**447.5**	**519**	**481.5**