UTRECHT UNIVERSITY

Master Thesis

Customisation of enterprise software using the RE4SA model

Author: Tjerk Spijkman

Internal Supervisors: prof. dr. Sjaak Brinkkemper, dr. Fabiano Dalpiaz

External Supervisor: Anne-Fleur Hemmer.

Business Informatics, Faculty of Science

Department of Information and Computing Sciences

July 2019

Table of Contents

Chapter 1

Introduction

1.1 Problem Statement

Enterprise application software**,** is a type of software that is ready-made and designed to be easily implemented in existing systems. While there is no universally accepted definition of an enterprise application, enterprise applications typically meet one or more of the following characteristics (Hohmann, 2003):

• They are designed to support the needs of a business, at either a departmental or larger organizational unit;
• They are relatively expensive to build or license ($50,000 - $5,000,000+);
• They have complex deployment and operational requirements;
• While they can be operated as an independent application, they needs of the business are often best served when they are integrated with other enterprise applications.

In Enterprise application software, like enterprise resource planning (ERP) software, customisations are often necessary to increase the value for customers and to ensure that the *ERP* meets the company's needs (Zhang, Lee, Huang, Zhang & Huang, 2005). Panorama consulting (2014) reports that for ERP systems, 90% of the systems have at least minor customization. A glossary of the important terms used in this research can be found in Appendix B.

As the software is updated it might include some of the functionalities that were previously added through the customisations. This means that functionalities might be duplicated, or that the ERP needs to be customised again. Light (2001), found that customisations require additional maintenance, which in turn leads to additional maintenance challenges. For example, source code for the customisations might be lost over time, customisations need to be re-written, re-tested and re-implemented. One cause of additional maintenance is that the user exits (points in the software to which add-ons are linked) that add-ons rely on might change in an update of the software (Ng, 2012). This means the add-on needs to be changed to work on the new version, ensuring that customisations work on a new version of the software is known as retrofitting.

Software customisation can be divided in three categories, functionalities can be:

1. Modified: changes made to existing functionalities in the software product; for example if the software only supports a single type of currency like dollars, this currency can be modified to better fit the context.
2. Extended: additional functionalities added to an existing module; for example if the finance module only supports manual invoice processing, this functionality can be extended to support automatic invoice processing.
3. Introduced: additional modules added to the software product; for example introducing satellite localisation of employees or products to an enterprise software.

Light (2001) identified the first scenario as the biggest maintenance risk. An architecture, inspired by documentation of JDE (Oracle, 2015) is displayed in Figure 1. This figure visualizes how these categories could impact the software architecture. It also shows a scenario where part of the Real

Estate functionality that was originally added as a customisation is now included in the new version of the ERP system (4). This can create conflicts between the customisation and the ERP software, and the company now needs to make a choice to lose some of the functionality of the customisation, not update the ERP, or have the customisation altered to work with the newest ERP release.

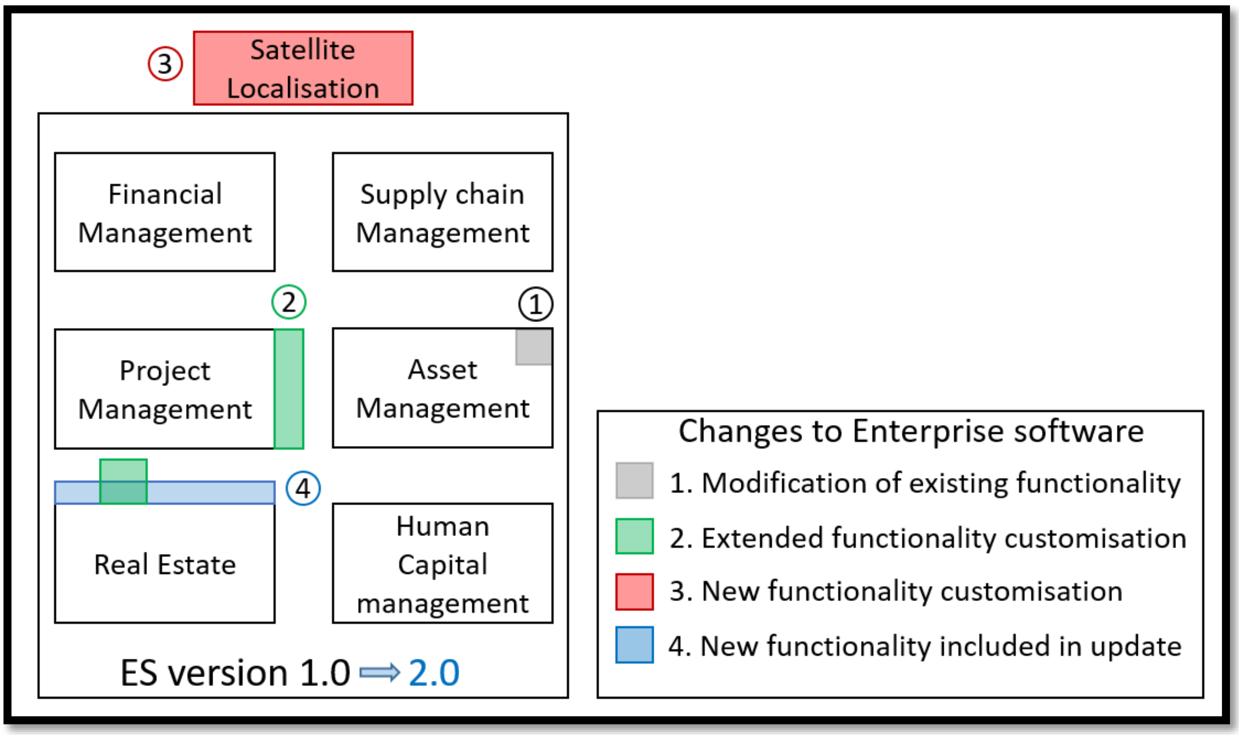

③
Satellite Localisation
Financial Management
Supply chain Management
②
①
Project Management
Asset Management
④
Real Estate
Human Capital management
ES version 1.0 ⟹ 2.0
Changes to Enterprise software
1. Modification of existing functionality
2. Extended functionality customisation
3. New functionality customisation
4. New functionality included in update

Figure 1 – Categories of customisations and updates in software architecture of enterprise application software.

It is often difficult to localise where exactly these incompatibilities will arise when a software update is released. These difficulties occur as the documentation to see which aspects are visualised is limited or non-existent. This means developers often have to scan the source code to detect changes. Furthermore it is not always clear on which software elements one specific module is dependant. In this study the requirements engineering for software architecture (RE4SA) model will be applied in order to improve the documentation, development and maintenance of enterprise application software.

The RE4SA model shows how different abstraction levels in requirements engineering relate to software architecture concepts (Jansen & van Rhijn, 2016). The RE4SA model shows the relation between the epic story & user story concepts on the requirements engineering side, and the module and feature concepts from Software Architecture (*Figure 2*). This model will be discussed in more detail in chapter 3.3.

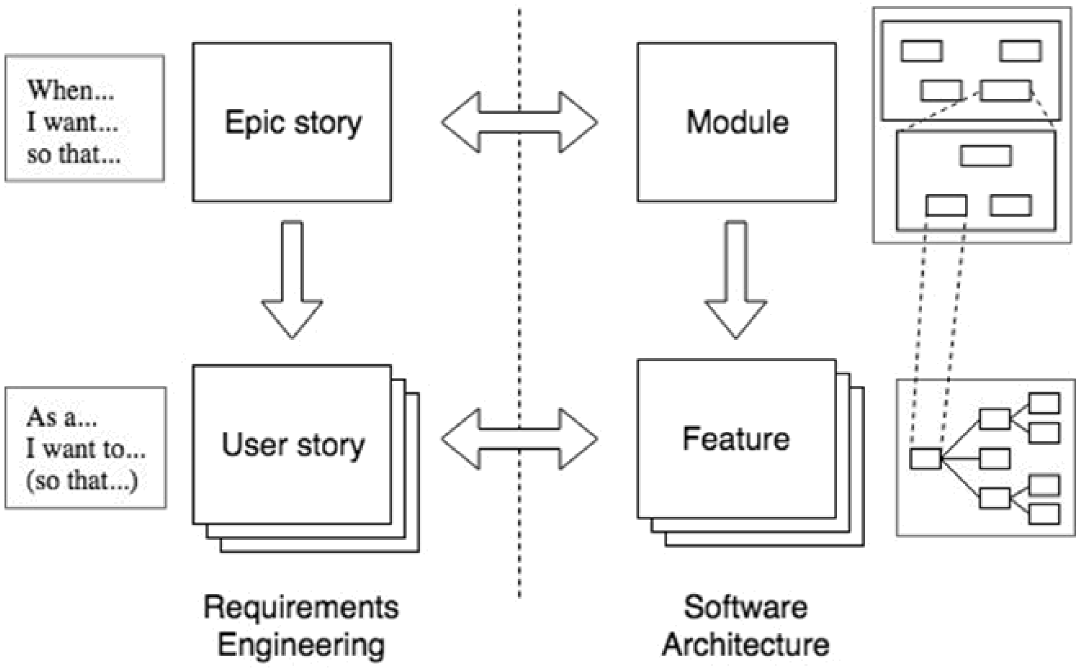


Figure 2 – The requirements engineering for software architecture (RE4SA) model (Molenaar, et.al. 2019)

User stories have become a very popular method for requirements engineering (Kassab, 2015; Lucassen, Dalpiaz, van der Werf, & Brinkkemper, 2016b; Cao & Ramesh, 2008), in a survey with 247 participants, Kassab (2015) found that User Stories had an adoption rate of around 45% in 2013. User stories are defined by Cohn (2004) as a description of a functionality that will be valuable to a user or purchaser. This functionality often matches a certain feature in a software product, for example in the user story "*As a student, I want to save my files as pdf, so that I can hand them in in the correct format*" could match a "save as" feature within a "document management" module. Cohn (2004) defined three aspects that user stories are composed of:

- A written description of the story used for planning and as a reminder
- Conversations about the story that serve to flesh out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete.

User stories are mainly used in agile development methods, but also have a place in other methods. Lucassen, Dalpiaz, Werf, and Brinkkemper, (2016b) found in a survey with 182 valid responses, that 99% of respondents indicate that they used user stories when they work with scrum. For waterfall methods only 21% of the respondents that work with the waterfall method used user stories.

An *Epic story* contextualises high-level jobs that a product needs to fulfil (Klement, 2006). Klement introduced the template as Job Story, however due to the existing notion of epics in Scrum development they were renamed to epic story for research within Utrecht University. An epic story has a larger scope than a user story and can be split into two or more user stories. (Cohn, 2004) Epic stories can be created to represent a module that should be included in the software, once work begins on this module the epic story can be refined by creating the corresponding user stories.

On the software architecture side, the RE4SA model uses modules and features. The modules can be visualised in a functional architecture diagram (FAD) (Brinkkemper & Pachidi, 2010) which shows how the modules interact with one-another and what their functionalities are within a software product. On the more detailed level, the model uses feature diagrams (Jansen & van Rhijn, 2016) to show all user visible elements (features) in the software, and the hierarchy of the features.

1.2 Research Objective

The RE4SA model (*Figure 2)* will be applied to the enterprise application software case in an attempt to improve the documentation, development and maintenance of customisations. User stories could potentially improve these elements because they document more than what is expected from a customization; User stories document who it is for, what is expected from the system and why it is important (Lucassen, Dalpiaz, van der Werf & Brinkkemper, 2016b). This provides more information for developers to base the customisations on, but still let them decide how to develop the functionality. The tested method can also improve the traceability of customisations through the creation of a clear overview of concepts using ontology tooling. This can help detect possible conflicts in features when a new update of the ERP is released, providing information that can detect which customisations require attention based on the new functionalities included in the update.

In this research a method will be constructed that includes the use of user stories and epics for the creation and maintenance of customisations in enterprise application software. This method will then be evaluated based on a case study.

Chapter 2

Research Approach

This chapter will focus on the approach chosen for this research. First the main research question is introduced. This is followed by a set of six sub-questions that together should provide an answer to the main research question. Next, the research method for the project is elaborated. Then the case company Forza IT Group, the case software JD Edwards and the SCANMAN customisation will be introduced.

2.1 Research Questions

Based on the problem statement and research objective stated in chapter 1, a main research question is formed that should provide a (partial) solution to the challenges in customisation of enterprise application software. Such a question is often too broad in scope to give a clear answer. Therefore, a set of sub questions is created to guide the research, and as a whole provide an answer to the main research question. The main research question that will be answered in this study is:

> *RQ: "How can the RE4SA model be applied for customisation of enterprise application software, and what are its benefits".*

The research question is broad in scope as the application can be done for different scenarios within enterprise application software, and these scenarios might have different benefits. Therefore the first sub question is similar but with a much narrower scope:

> *SQ1: "How can RE4SA be applied for customisation of enterprise application software?"*

In this question it will be important to analyse how the different aspects of the RE4SA model can be applied to the creation and implementation of customisations in enterprise application software. This part will consider the literature about *user stories* & *epics*, and look at the creation process of an ERP customisation. A Product deliverable diagram (PDD) will be applied to this scenario to see how the RE4SA model can be implemented in the customisation of an ERP system.

> *SQ2: "How can RE4SA be applied for release updates of (customised) enterprise application software?"*

The second question will be a continuation of the first question. And will adapt the method modelled in the first sub question to be used when the enterprise application software is updated. As discussed in the problem statement, a release update can cause issues in the customisations and require manual actions to ensure that customisations function.

> *SQ3: "How can ontology tooling assist release upgrades and customization in enterprise application software?"*

The third question will analyse the benefit that ontology tooling can have within the method defined in the first two questions. This research question is based on the assumption that ontology tooling can provide visualisations that can be utilized to improve interpretation of a complete set of user stories.

SQ4: "How can RE4SA be applied to improve internal (developers) and external (customers) communication and documentation in enterprise application software?"

The fourth question looks at a different aspect introduced through the RE4SA method. By applying user stories and epics in the implementation, customisation and maintenance of enterprise application software the way of communication and documentation is also changed. The aim is to detect how this change impacts internal documentation and communication with developers, and external documentation and communication with customers.

SQ5: "How can method fragments of RE4SA be integrated into the process used in the case company?"

The fifth question compares the process used in Forza IT Group for implementation, customisation and maintenance of the ERP to the method that is analysed in the first two questions. Besides comparing the steps and analysing differences, fragments of the method suggested for the RE4SA model will be integrated into the process used by Forza to create a new method.

SQ6: "How effective is the situational method for the customisation of enterprise software"

The final sub question is an evaluation of the method created in SQ5, this evaluation will be done through expert interviews based on the case studies in which the method has been applied.

2.2 Research Structure

To solve practical problems the engineering cycle in the design science method by Wieringa (2014) can be applied. Design science is the investigation of artifacts in context (Hevner, March, Park, & Ram, 2004), in this research we will investigate the use of a method based on the RE4SA model in the context of enterprise application software, or more specifically ERP customisation. The engineering cycle (*Figure 3*) has four phases: it starts with investigation of the problem, followed by design of a treatment / solution, which is then validated and implemented. The outcome of the final phase, can be evaluated and be the start of a new iteration of the engineering cycle.

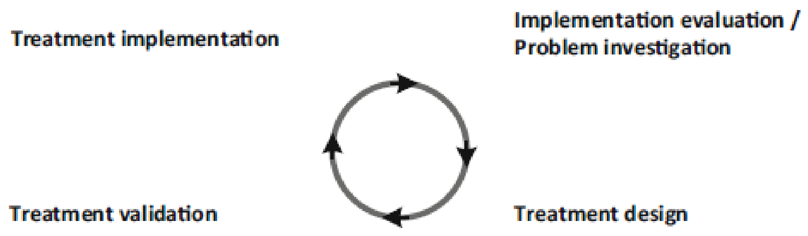


Figure 3 – The engineering cycle by Wieringa (2014)

2.2.1 Problem Investigation

In this phase SQ 1-4 will be investigated. In this study the problem that we will try to solve is that managing requirements is a difficult task, and poor requirement management is the most important cause of project failure (Smith, Bieg & Cabrey, 2014). While the RE4SA model is a promising treatment, it is unclear if it can be effectively deployed in the context of customisation of enterprise software. In this phase a literature study will be performed to detect what the challenges are for ERP customisation, and identification of the current state of the aspects in the RE4SA model. This will first be done in a general context, and will then be specified to analyse how it could fit within the enterprise application software context.

The functionalities of the ERP system and customisations will also need to be analysed to get an overview of the context in which the RE4SA model will be applied. This will be done through a literature search, existing documentation of JD Edwards (ERP system), documentation available at the case company and through manual analysis.

2.2.2 Treatment Design

Once the problem is sufficiently investigated and enough knowledge is gathered, a treatment will be designed, in the engineering cycle this is an artifact such as an algorithm, method, technique or framework that interacts with the problem context (Wieringa & Morali, 2012). The treatment in this study, will be a method in which we apply the RE4SA model to the current customisation method used at the case company. This phase will lead to an answer to SQ5.

2.2.3 Treatment Validation

Once the treatment is designed it should be validated to ensure that it contributes to the stakeholder goals before implementing it. There are different factors that need to be validated before the treatment can be considered to be representative for the population of interest. In this research we will consider three of the four types of validity threats included in the experiment principles model by Wohlin (2012) shown in *Figure 4,* specifically Internal validity, construct validity and external validity.

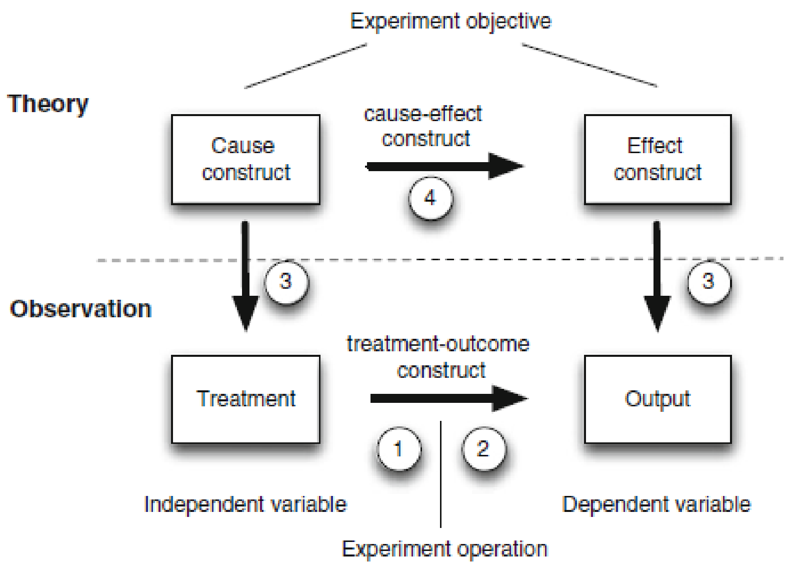


Figure 4 – Experiment principles by Wohlin (2012).

The four validity threats as defined by Wohlin (2012) are:

1. *Conclusion validity*: This validity is concerned with the relationship between the treatment and outcome, and requires a statistical significance in the results. As we will perform a case study and qualitative evaluation instead of quantitative evaluation this validity is deemed not relevant for this study.

2. *Internal Validity:* This validity is about ensuring that the outcome is the result of the treatment and not the result of a factor that we do not control or have not measured.
3. *Construct validity:* This validity is concerned with the relation between cause and effect. In order to adhere to this requirement we need to ensure that the treatment corresponds to the cause and that the outcome of the study reflects the effect we are interested in.
4. *External Validity:* In this threat we consider whether or not the results from the case study represent can be generalised outside of the scope of this research. So in other words we will discuss if the results are relevant for cases different from the one considered in the case study.

These impacts validity threats will be minimised through the creation of a case study protocol, based on guidelines available in literature. In the final chapter of the thesis, the remaining validity threats will be discussed.

2.2.4 Treatment Implementation

In the implementation phase we apply the treatment to the problem context. In this research we will apply the method created as an answer to SQ5 in four case studies provided by Forza IT Group. The first and second case study will be a retrospective case study, each based on one project already finished by Forza IT Group. This first case study matches the field study type (Stol & Fitzgerald, 2018), which means that the natural setting is intruded on in the least amount possible to maximise the realism of the setting. The third and fourth case study will be an action research, this matches the field experiment study type. In action research the researcher intervenes in a setting in an attempt to improve the current situation by making changes, observing the resulting situation and making further changes. This means there will be some level of intrusion but it allows for a better way to evaluate the method compared to the first case study. By performing both these case studies, this research will obtain results for both fields in the natural settings quadrant identified by Stol and Fitzgerald (2018), this will provide real world data for the effectiveness of the RE4SA model.

Using the created method the requirements of the case will be documented using epic stories and user stories. The software architecture of the SCANMAN in the context of this case will be documented using functional architecture diagram and feature diagram notations. All these concepts are further explained in chapter 3.

2.3 Case Company – Forza IT Group

The research project is supported by Forza IT Group, a company that specialises in services relating to Oracle NetSuite, Oracle JD Edwards and cloud applications.

Forza provides support for the implementation of ERP systems, application management and create and maintain customisations for customers. When they implement software at a customer, they specify the required configuration and customisation for the context of that customer. This means that each customer gets a personalised version of JD Edwards to suit their needs. Forza also develops products that extend the functionality of the Oracle ERP systems.

Forza will be a valuable source of information for this research. They will provide information about their current method, documentation on previous and ongoing projects, practical expertise in the context area and expert opinions on the suggestions in this study. The working environment for this research will be mostly within Forza IT Group, this results in constant communication with employees to obtain information and opinions from them, and a form of quality control for the research. They also provide their company knowledge, and access to the Oracle Support website for information about new releases and online documentation of Oracle products. Their internal knowledge will be crucial to determine how the RE4SA model can be applied to enterprise application software, and will be necessary for modelling their internal process. However, since this information is only accessible from the intranet of Forza, and in some cases can be confidential, not everything will be included in this thesis. When a specific part is deemed relevant for the purposes of the research, it will be discussed whether it can be included, and in what form (e.g. anonymised, partially, not at all).

2.4 Case Software – JD Edwards & SCANMAN Add-on

JD Edwards was an ERP software company founded in 1977, they were purchased by PeopleSoft, Inc. in 2003, which in turn was purchased by Oracle in 2005. The JD Edwards ERP is still sold and supported by Oracle, has a broad scale of functionalities and is used in more than 10,000 companies (idatalabs.com, 2018). Oracle describes the software as follows:

> *"JD Edwards EnterpriseOne offers a powerful, fully integrated ERP software suite that provides more choice of databases and deployment options, including on premise, private cloud, public cloud or hybrid cloud for maximized flexibility and low TCO. With over 80 application modules, end-user reporting, and personalization capabilities, JD Edwards EnterpriseOne combines business value, standards-based technology, and deep industry functionality into a solution that will transform your business. In addition to an array of out-of-the-box mobile applications, JD Edwards EnterpriseOne leverages Oracle's Mobile Platform to accelerate business execution and provide a complete enterprise mobile solution from deploying to building and extending mobile applications." (Oracle, 2016)*

As mentioned in the first chapter, ERP systems are often customised to add functionalities required by the customer. One such customisation of JD Edwards is SCANMAN. This add-on increases the efficiency for the financial processes, as it automates invoices processing in JD Edwards. It then provides a workflow for approval of payments. SCANMAN is fully integrated within JDE, and requires no external applications. SCANMAN uses OCR to automatically scan and process values from received invoices. These are then presented to users for verification, after which they can enter the approval process.

The JD Edwards ERP system, and the SCANMAN customisation will be the main focus for the context of this research.

Chapter 3

Theoretical Foundation

For the literature review there are five different categories in which the findings can be split. The chapter will start with an overview of literature findings in the context of ERP customisations. Next the Requirements engineering section of the RE4SA model is reviewed, which covers user stories, epic stories and jobs to be done. This is followed by a review of the software architecture concepts included in the RE4SA model. Once all these concepts are discussed, we can finally discuss the model as a whole and this is done through an example to show how the different aspects of the model are interlinked. Finally ontology tooling is discussed to see how it can be used in a method based on the RE4SA model.

The literature review is done using the snowballing method (Wohlin, 2014). This method refers to using the reference list of a paper, and the referenced by lists to identify additional papers that are relevant for the research. This literature review was started by collecting the papers on user stories and the Grimm project by going through the papers by Lucassen, Dalpiaz, Brinkkemper, and scanning the papers they referenced. An unpublished book (Brinkkemper, S., Dalpiaz, F., & Lucassen, G., 2018) written by students as part of a university course on user stories was also used as a starting point for the snowball method. Other papers outside the snowball literature search were also used, for example papers that were recommended by others, or literature used in university courses.

3.1 ERP Customisation

An Enterprise Resource Planning (ERP) system is used to store information that was previously fragmented in many different systems in a single comprehensive data repository (Laudon & Laudon, 2017) and can be defined as:

> *"Software providing integrated functions for major business functions such as production, distribution, sales, finance and human resources management." (Chaffey, 2009)*

An ERP combines the different sources of information of a business and integrates it into a single collection of inter-linked processes that makes up the business. For example, a purchase in the order entry module passes the order to the manufacturing application, which sends a materials request to the supply-chain module which gets the parts from suppliers and uses a logistics module to get them to the factory (Gupta, 2000). The major goal of an ERP system is to increase operating efficiency and decreasing costs (Seo, 2013; Nah, Lau & Kuang, 2001; Beheshti, 2006). Additionally Light (2005) identified the following reasons for adoption of ERP packages:

- Correction of existing problems;
- Predictability;
- Business benefits;
- Social influences;

While an ERP can improve the business processes of a company, it also requires a big investment of time and money to implement it. ERP implementations often take longer than their estimated schedule and end up at a higher cost than estimated (Ehie & Madsen, 2005; Helo, Anussornnitisarn & Phusavat, 2008). When an implementation fails, this can have high impact on the company and going back is extremely difficult (Bingi, Sharma & Godla, 1999). An example of such a failed implementation is FoxMeyer Drug, a $5 billion pharmaceutical company that went bankrupt after a failed implementation of SAP. Implementation of an ERP system comes with a large amount of change for the organisation, they need to reengineer their business processes to align with the ERP, train employees, and integrate the ERP in the company culture. Other research also reports on the risks of ERP failure:

- "Many ERP systems still face resistance, and ultimately, failure" (Aladwani 2001)
- "Between 50 percent and 75 percent of U.S. firms experience some degree of failure … One recent survey revealed that 65 percent of executives believe ERP implementation has at least a moderate chance of hurting their business." (Umble & Umble 2002)
- "Three quarters of the ERP projects are considered failures and many ERP projects ended catastrophically" (Rasmy et al 2005)
- "Failure rates estimated to be as high as 50% of all ERP implementations" (Muscatello & Parente 2006)

There are different stages in the implementation of an ERP system, Ehie and Madsen (2005) suggest a five-stage implementation process based on a literature review they performed. These stages are:

- *Project preparation stage*: The planning phase in which a project team is formed, targets and objectives are set;
- *Business blueprint stage*: The current processes are analysed to select the ERP system that best fits the company, and a project team is trained in the configuration and functionality of that ERP;
- *Realization stage*: The actual implementation, customisation and data conversion is done in this phase;
- *Final preparation stage*: The process is integrated, the system is tested, and end users are trained to use the ERP;
- *"Go live" and support stage*: The ERP is used in daily activities and is improved and expanded.

Although an ERP, and Enterprise Application Software in general have the benefit of providing a standard solution in the software landscape when compared to custom software, they still need to be customised to meet the company's needs (Zhang et al., 2005). We define customisation as follows based on the definition by Light (2001):

"Customization is any change or addition to the functionality available in the standard software product."

The need for customisation can also been seen in the price of ERP implementation. According to Shin (2006) the majority (up to 60%) of ERP project cost is devoted to setup, installation and customization of the software, services typically provided by outside consultants (Dolmetsch et al., 1998; Österle, et al. 2000). The Financial Time (2003) reports an even higher rate for implementation factors:

> *"Prof Brynjolfsson and colleagues found that of the $20m total cost of an enterprise resource planning (ERP) system, only about $3m goes to the software supplier and perhaps $1m towards the acquisition of new computers. The $16m*

balance is spent on business process redesign, external consultants, training and managerial time" (Financial Time, 2003)

This shows that customisation of an ERP plays a big role in the use of the software within a company. These customisations also introduce new challenges. Light (2001) identified different levels of maintenance needs based on the type of customization. Customisations that change the existing functionality and code of an ERP are most likely to cause conflict in future updates of the ERP. For added functionalities, it is important that the link between the customisation and the ERP is maintained throughout different versions of the software. For automation of processes, and changes in the reports generated, the maintenance need is expected to be lower.

Updates of the ERP software can cause certain customised functionalities to stop working. Hohmann (2003) calls these "ripple upgrades", which he defines as "an upgrade that forces you to upgrade otherwise stable system components". For example the update of an ERP might change the way taxes are calculated to improve performance, which in turn causes issues for customisations that rely on this information.

Somers and Nelson (2001) note minimizing customisation as a factor for successful ERP implementations. Based on a survey they observed that 41% of companies re-engineer their business to fit the application, 37% choose applications that fit their needs and make small customisations, and 5% customise the application to fit their business. Somers and Nelson (2001) mention that customization should only be requested when essential or help achieve competitive advantage, because they often increase the cost of the system, increase implementation time, and limit the benefit from software maintenance and upgrades. The customisation in the paper by Somers and Nelson seems to only consider modifications. Add-ons might be valuable in cases when modifications is not recommended, as they have a reduced risk at increasing maintenance and upgrade costs. Furthermore, add-ons are often supported by the software vendor that provides the add-on.

The combination of the high costs of ERP implementation and customisation, and the high risk of failure, show that improvements to the process are desirable. In the next paragraph, we will consider user stories, epic stories and software architecture. Then in chapter 4 these concepts will be analysed for usability within the ERP context.

3.2 Requirements Engineering

3.2.1 User Stories

User stories date back to 1999, where Beck described them as:

"The amount of a use case that will fit on an index card" (Beck, 1999)

In this paper he also suggests to make shorter development cycles in improve development. The definition for user stories by Cohn (2004) in his book "User stories applied for agile software development" details the following aspects of a user story:

- A written description of the story used for planning and as a reminder
- Conversations about the story that serve to flesh out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete

Cohn notes that the acceptance tests should be written by the customer whenever possible, since they are the ones that have a desired functionality in mind. In his book, Cohn also lists the following advantages for the use of user stories compared to other requirement elicitation techniques. User stories:

- Emphasize verbal communication;
- Are comprehensible by everyone;
- Are the right size for planning;
- Work for iterative development;
- Encourage deferring detail;
- Support opportunistic design;
- Encourage participatory design;
- Build up tacit knowledge.

The requirements engineering technique of user stories has gotten more popular over the years, in 2013 user stories were used in 45% of software companies compared to 32% in 2008 and 1% in 2003 (Kassab, 2015). Lucassen et al. (2016b) found that in practise 70% of the practitioners that use user stories, make use of the Connextra template. This template is as follows:

"As a <Role>, I want to <Goal>, so that <Benefit>."

Cohn (2004), suggests that the benefit part of this template is optional. However, Lucassen et al. (2016b) found that this part of the story is essential to their respondents. The benefit part answers why a functionality is required. The main benefits that Lucassen et al. found for the *why* are: it alleviates confusion among stakeholders, reduces the amount of discussion necessary, and provides developers with autonomy in their work. Figure 5 shows the Connextra template as a conceptual model, this model shows that a user story has one role, one means (goal) and no, or multiple ends (benefit).

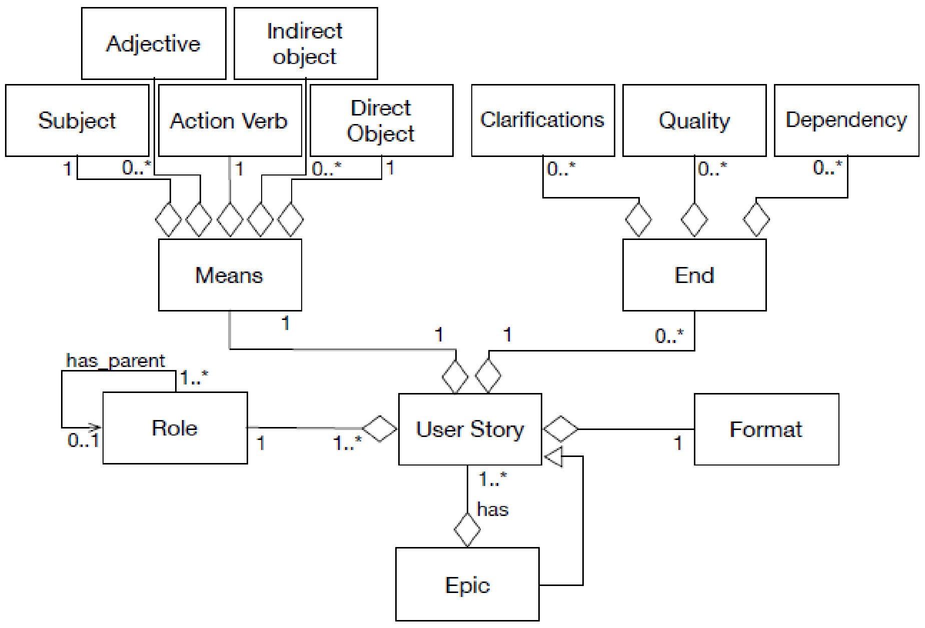


Figure 5 – linguistic model of the Connextra user story template (Lucassen et al., 2015)

An example of a user story that uses the Connextra template is:

"As a financial manager, I want to send an invoice to a colleague for acceptance, so that I can process the invoice."

Based on the book by Cohn (2004), *Figure* 6 shows an activity flow with the activities required for the creation and use of user stories. The process starts with deciding which roles should be included, this is often done through brainstorming and conversations with the customer. Once the roles have been decided, user stories need to be gathered, this can be done through interviews, questionnaires, observation and similar techniques. When the set of user stories is gathered, acceptance criteria need to be specified, Cohn suggests that the customer should specify these tests. After this, the effort to perform the tasks required to satisfy the user story should be estimated. Once this is all done, a release can be planned, the team selects a set of user stories that should be processed and decides on a release window.

The process in the previous paragraph explained the activities Cohn (2004) identified for applying user stories. In chapter X, a more specified method for the enterprise application software context will be discussed.

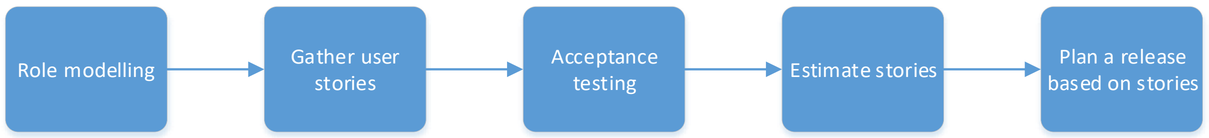


Figure 6 – activity flow for the creation of user stories

In order to be valuable, user stories need to be of an adequate quality. However, the available methods for assessing and improving user story quality is limited (Lucassen et al., 2015). The INVEST (Independent-Negotiable-Valuable-Estimable-Small-Testable) heuristics by Wake (2003) can be used to assess the quality of a user story. These heuristics are used to test whether:

- The user stories do not overlap in concept,
- The user stories are not too specific in detailing how to achieve the result,
- The user stories have value for the customer,
- The level of effort for handling the user stories is estimable,
- The user stories are not too big in scope,
- The user stories can be tested to determine if the results meet the requirements.

In 2015, Lucassen et al. introduced the Quality User Story framework (QUS). This framework as seen in *Figure* 7 considers 14 criteria to assess user stories, in syntactic, semantic and pragmatic categories. They also introduced a natural language processing tool (NLP) that can analyse a set of user stories based on the QUS framework, called Automatic Quality User Story Artisan (AQUSA). This tool can uncover up to 90% of easily preventable defects in user stories (Lucassen, Dalpiaz, van der Werf, & Brinkkemper, 2016a).

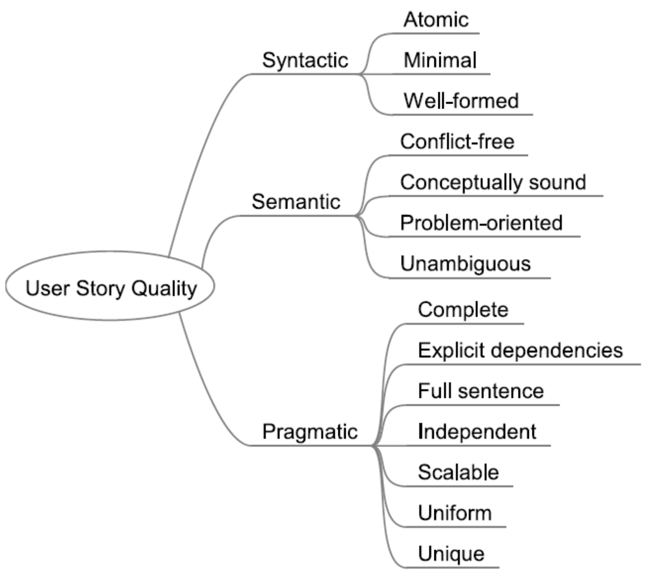


Figure 7- Quality User Story (QUS) Framework (Lucassen et al., 2015)

3.2.2 Epic Stories

An Epic story is a user story that is too big to be implemented and can be deconstructed in multiple smaller user stories (Cohn, 2004), this is also represented in the conceptual model of the user story *Figure* 5. Epics can be useful for describing a functionality on a bigger scale than user stories, and as a placeholder for a functionality that still needs to be defined. It could also be used to create a group of user stories to adhere to a single epic. Klement (2013) introduced the epic story template:

> *"When <problematic situation>, I want <motivation>, so that <expected outcome>."*

For this template Lucassen et al. (2018) also defined a conceptual model similar to that of the Connextra template (*Figure 5)* this conceptual model is shown in *Figure 8.*

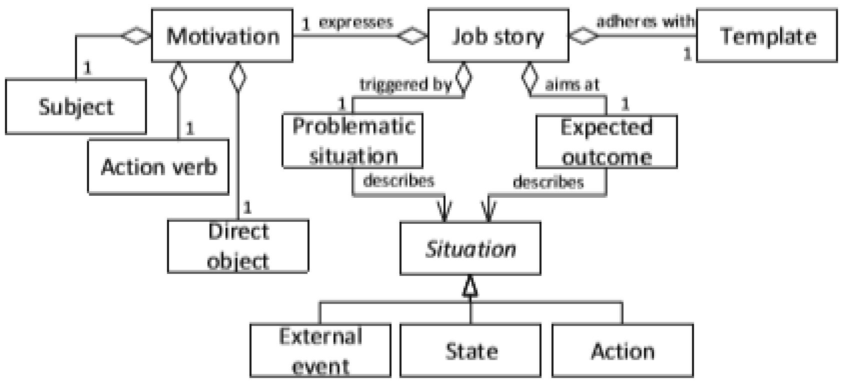


Figure 8 – Linguistic model of epic stories (Lucassen et al., 2018)

Klement (2013) notes that this epic story format prevents assumptions made in a user story, and keeps the options open to define a solution. An example of an epic in the SCANMAN context could be:

> *"When I approve invoices, I want to quickly have access to all the information for invoices that I need to approve, so that I can work more efficiently. "*

This epic could have multiple subtasks, if it was used during the development of SCANMAN, it would have to be decomposed into multiple smaller US that can be implemented. For example one of the smaller user stories could be:

> *"As a financial manager, I want to filter the invoices to see only those that require my attention, so that I don't waste time on invoices that are not my responsibility."*

This decomposition should be done in collaboration between team members to discuss different possibilities and to make sure that all relevant aspects of the epic are covered by the set of user stories.

3.2.3 Jobs to be done

On a higher level than Epics are Jobs to be done (JTBD theory). A Job to be done represents the customers overall goal for wanting a change in his or her situation. A Job to be done is often written as:

> *"Help me <verb> <noun phrase>".*

Klement defines these jobs to be done as:

> *"A Job to be done is the process a consumer goes through whenever she aims to transform her existing life-situation into a preferred one, but cannot because there are constraints that stop her." (Klement, 2016)*

People often hire products to do a specific job for them (Christensen, Cook, & Hall, 2005), this is also the case for a software product. For example, a job to be done in the context of the SCANMAN customisation could be:

> *"Help me spend less time on invoice processing."*

Klement (2016) mentions that innovators often think they are studying customer needs, when they are actually observing what they don't like about the products they currently use. By creating a Job to be done, the customer's goals are considered instead of their frustrations. Furthermore, the increase in available data causes firms to focus on correlations. A job to be done helps to shift this focus from the correlations to what the customer hopes to accomplish (Christensen, Hall, Dillon, & Duncan, 2016). According to Klement (2016), using the job to be done as the basis of a new product, designers can prevent limiting themselves to the needs and expectations that are linked to current products. This should allow for more innovation in the solution that fulfils the customers goal.

Blessinga (2018), included these Jobs to be done in his thesis as an extension of the RE4SA model, however for the purpose of this research they were deemed to be out of scope. Since the application already exists in a customisation scenario, the job to be done should already be determined for the initial application design. When a new job would be included in the existing software, this will likely lead to scope drift of the application, and would likely be more suited to be a new product in the company's portfolio.

3.2.4 The Reference Method for User Stories

The concepts described above can be used to change the development process of software. One such method, the Reference Method for User Stories (*Figure 9)* is defined by Bik, Lucassen & Brinkkemper (2017). This method starts with gathering requirements, based on the gathered requirements user stories and epics (which are later split into user stories) are written and validated. The method then assigns an indication of how much effort they will take to develop and a selection of user stories is made to plan the next sprint, after which development can begin. This method only includes the user story and epics concepts, but not the Jobs-to-be-done or the software architecture side of the RE4SA model. Therefore this method can serve as a baseline for the method that will be created to answer sub-research question five.

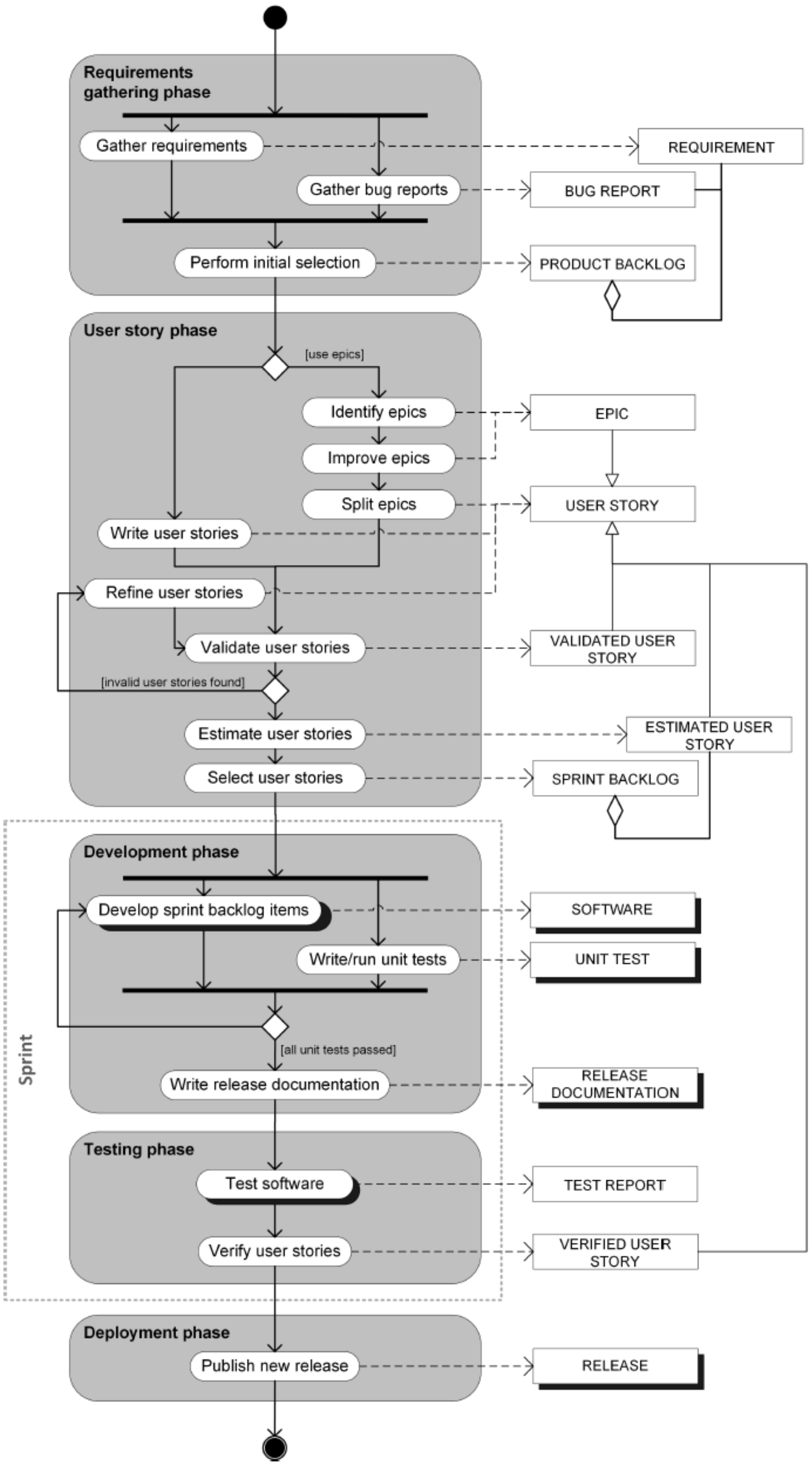


Figure 9 – The Reference Method for User Stories (Bik et al., 2017)

3.3 Software Architecture documentation

Bass, Clements & Kazman (2013) define Software architecture as:

> *"The set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."*

Software architecture enables communication between stakeholders, and helps ensure modularity in software development. In software architecture, a module is a part of the software with a well-defined functionality that can be developed independently (Parnas, 1972). Examples of modules could be the filtering or save management modules. Brinkkemper and Pachidi (2010) presented the Functional Architecture Model (FAM), which is a method to model the functionality of a software product, consisting of its main functions and supportive operations. The FAM created for an ERP by Brinkkemper and Pachidi can be seen in Figure 10.

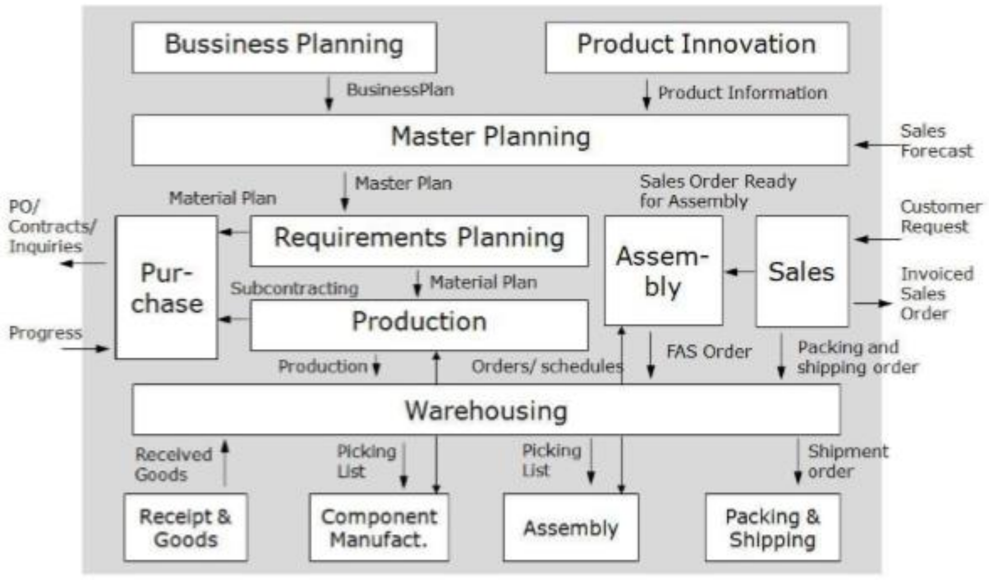


Figure 10 – Functional Architecture of an ERP product (Brinkkemper and Pachidi, 2010)

Brinkkemper and Pachidi (2010) also describe the Functional Architecture Diagram (FAD) (*Figure 11*), which contains modules that resemble the functions of a software product. These functions are implemented in the form of modules, and the interactions between modules can be seen in a FAD by flows. The FAD visualises how the product will be used, and it indicates which modules need to be created for the intended functionality of the product.

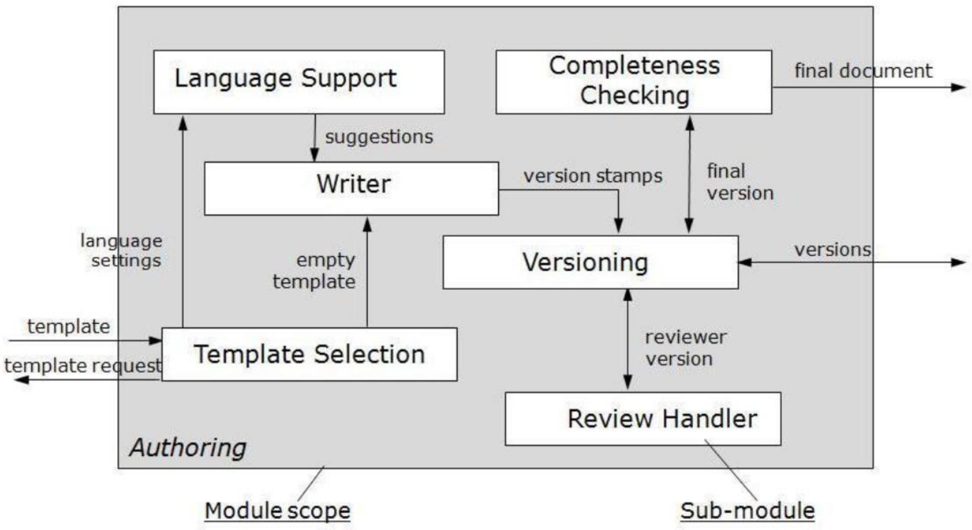


Figure 11 – Functional Architecture Diagram (FAD) of an Authoring module (Brinkkemper and Pachidi, 2010)

A feature is a smaller scale element in software architecture and can be defined as:

> *"A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems" (Kang, Cohen, Hess, Novak, & Peterson, 1990).*

In the module "document management" one of the features could be "save as pdf". Features can be modelled with a feature diagram, this model is a hierarchically arranged set of features (Riebisch, Streitferdt, & Pashov, 2004), and displays if features are mandatory or optional. This model gives an overview of all features that need to be included in a module.

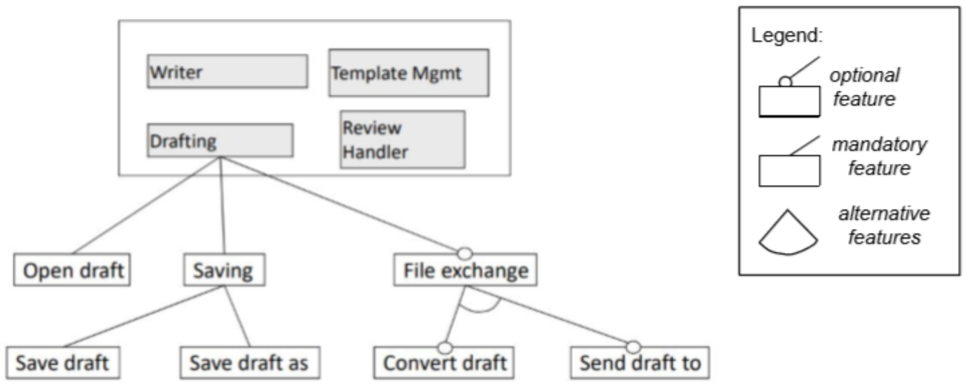


Figure 12 – Example of a feature diagram (Jansen & van Rhijn, 2016)

3.4 The RE4SA Model

As described in chapter 1, the Requirements Engineering for Software Architecture (RE4SA) looks at the correlation between concepts in the requirements engineering and software architecture domains. The importance of the relation between requirements and software architecture is discussed by Nuseibeh (2001), he introduced the twin peaks model that describes how requirements and software architecture are interlinked and should be consistent. The work on the twin peaks model is continued by Lucassen et al. (2015), who state that this is link is important for handling the ever changing requirements set of a software product. The requirements of a software product are continuously generated because the product is provided to multiple customers. Since the requirements are constantly being generated, this also has impacts on the software architecture, which is extended to fulfil the new requirements. The RE4SA model (Figure 13) describes the relationship between specific concepts in both requirements engineering (RE) and software architecture (SA). By linking these concepts, traceability is created between the concepts. This traceability supports the maintainability, changeability, and sustainability of a software system and its artifacts (Gayer et al, 2016). It should also be effective for handling new requirements, as they can easily be placed in the software architecture.

Blessinga (2018) & Van de Keuken (2017), also found that project briefs as defined by Intercom, (Traynor, 2016) are a vital addition to the RE4SA model for ensuring an exhaustive design. The template for the project briefs can be seen in Figure 14.

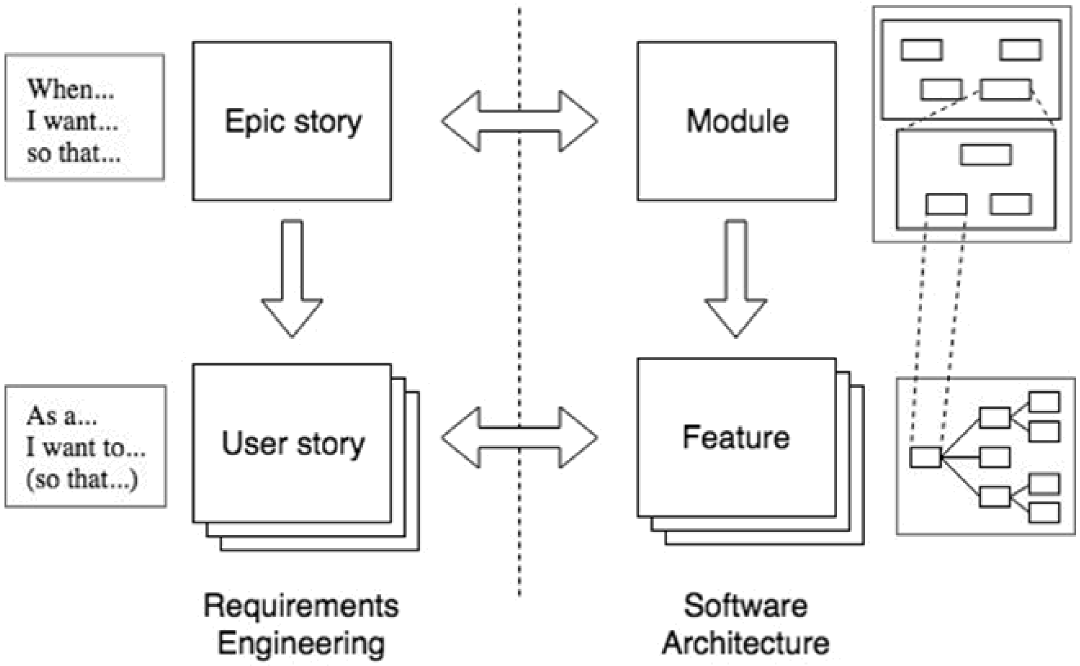


Figure 13 – RE4SA model (Molenaar et al. ,2019)

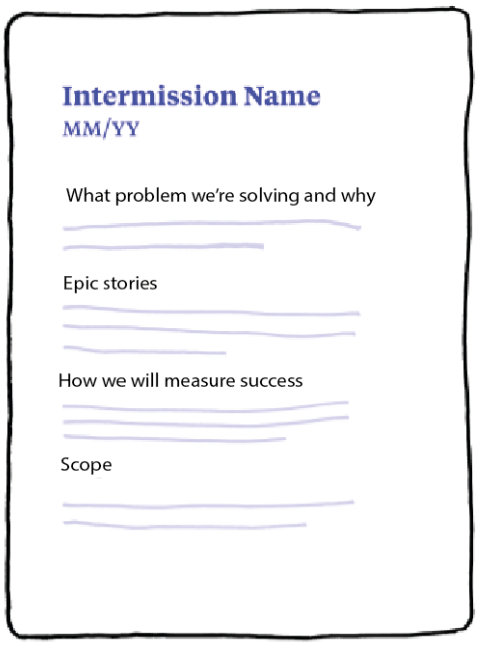

Intermission Name
MM/YY
What problem we're solving and why
Epic stories
How we will measure success
Scope

Figure 14 – Project brief template, adapted from Traynor (2016)

The RE4SA model is currently being researched within Utrecht University, as a part of the GRIMM project. The GRIMM research is an attempt to help practitioners use user stories, provide a framework for user stories and to analyse their usefulness (Lucassen, 2016). The GRIMM research group consists of; a project group of (master) students, and two professors at Utrecht University researching different aspects of the RE4SA model. As discussed previously, this research will test if the model is applicable in the context of ERP customisation, providing a multi-case study to provide findings for the application of the model in practise.

Geneca, a software publishing firm, published findings on a study of software development (Geneca, 2017). They found that 75% of their 600 respondents say their projects were doomed from the start, they also report that 78% of the respondents feel that the business is out of sync with project requirements and business stakeholders need to be more involved in the requirements process. Applying the RE4SA model could help improve requirements, and involve stakeholders, and hopefully results in a lower percentage of failed software projects.

3.4.1 Connection of the RE4SA Concepts in an Example

In order to demonstrate the idea behind the RE4SA method, an example of how the RE4SA might be applied to a route planning tool will be described. As an input for the RE4SA method we can use the job-to-be-done, this shows us what we're trying to achieve by creating a new application. The job-to-be-done in this context could be:

"Help me plan a route"

From this point the designers and developers should communicate with stakeholders to elicit requirements and decide what needs to be included in the application. Based on their findings they can create epic stories and user stories to document these requirements. In order to keep this example simple, only one epic- and user story will be discussed, in an actual project there would be a collection of both. One of the epic stories the developers found is:

"When I plan my trip, I want a planned route so that I can find my destination"

From the set of epics, the developers can model the modules in the application through a Functional Architecture Diagram (Brinkkemper & Pachidi, 2010). Blessinga (2018), found in his thesis that an epic story can indeed be linked to a module and form the foundation for the creation of a software architecture. The Epic story presented above can be traced to the circled "route planner" module in Figure 15.

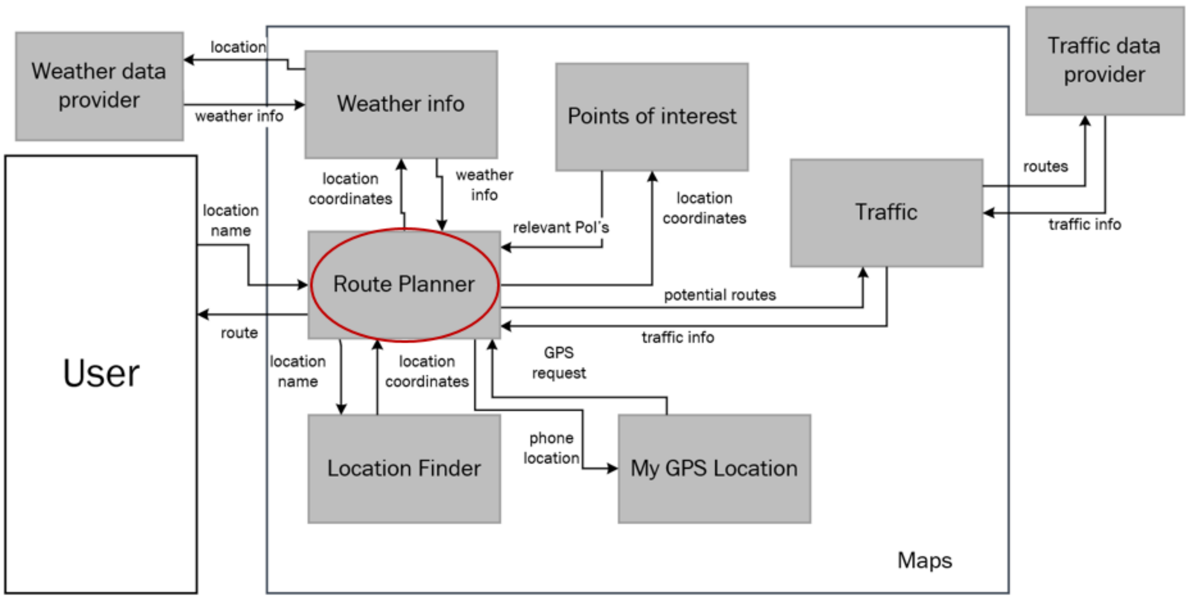


Figure 15 – Functional Architecture Diagram based on Maps

These modules as seen in the figure above all contain features that can be determined from the set of user stories gathered during requirement elicitation. The user story we will consider in this example is:

> *"As a consultant, I want the fastest route to my destination, so that I can efficiently visit customers"*

Just as the epic stories can be linked to the modules in the software, the user stories can be linked to features. The feature model below shows where the user story is linked to a specific feature.

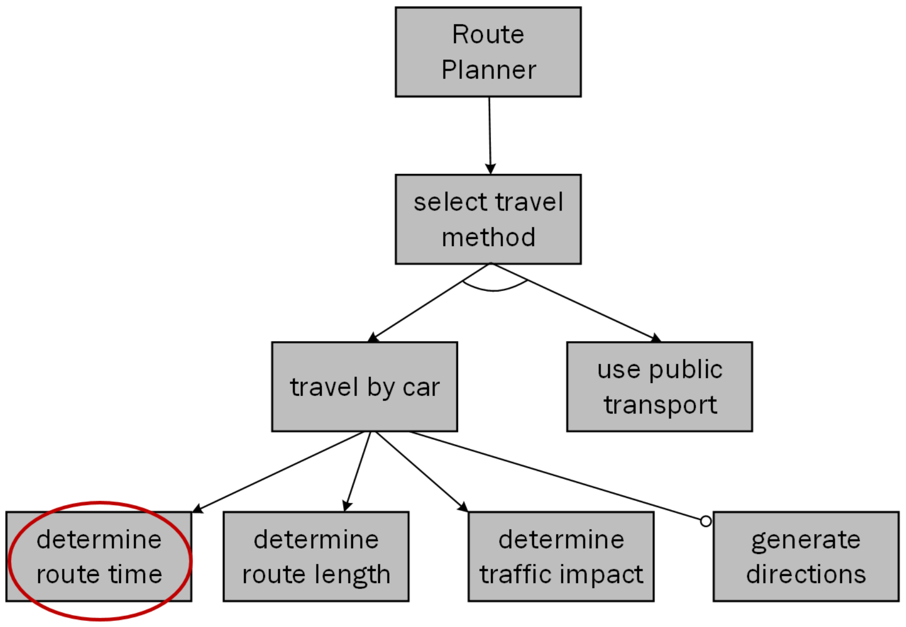


Figure 16 – Partial feature diagram of the "route planner" module

In this research the RE4SA model will be applied in a similar way to investigate its use in the context of enterprise application customisation.

3.5 Ontology Tooling

Ontologies are meant to specify content-specific agreements for knowledge sharing. A popular definition for ontology is the one by Grüber (1995) as:

> *"An ontology is an explicit specification of a conceptualization: the objects concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them"*

This is a general term for an ontology, and details design choices for the terminology and concepts used in a single area of interest. This could be any area from the concepts used in a certain research project, or a company. The benefit of an ontology is that it can visualise a domain and support conversations. Martens specified the term Product Domain Ontology (PDO) to describe the ontology of a specific software product:

> *"A Product Domain Ontology is a domain ontology that explicitly specifies the conceptualization of a certain software product." (Martens, 2016)*

This specialisation of an ontology is especially useful in the context of this research, as it can theoretically support user stories by visualising the concepts mentioned in a set of user stories of a software product. This is further enabled through natural language processing tools that are currently being developed as part of the GRIMM project, including the REVV tool (Dalpiaz, van der Schalk, & Lucassen, 2018) and Visual Narrator Tool (Lucassen, Robeer, Dalpiaz, van der Werf, & Brinkkemper, 2017b). In the following sections we will discuss the following ontology tools:

- The REVV Tool: a tool for ambiguity detection within an ontology
- Visual narrator tool: a tool that extracts a conceptual model from a set of user stories

- The Interactive Narrator Tool: a tool that combines the conceptual model extracted by the visual narrator tool with data from an issue tracking application (like JIRA)

3.5.1 The REVV Tool

The Requirement Engineer Verification and Validation (REVV) tool, aims to give an overview of ambiguities and missing requirements in a set of user stories. The REVV tool creates a conceptual model of the set of user stories, and presents this in a Venn diagram from the viewpoint of each role in the user story set. The algorithm tags possible synonyms, which can aid in both improving user stories by removing ambiguity and identifying missing requirements. Additionally it can improve the Product Domain Ontology by ensuring a single term is used for a single concept. By applying the REVV tool in a situational RE4SA method, we can improve the quality of the user story set, which in turn should improve the naming of features.

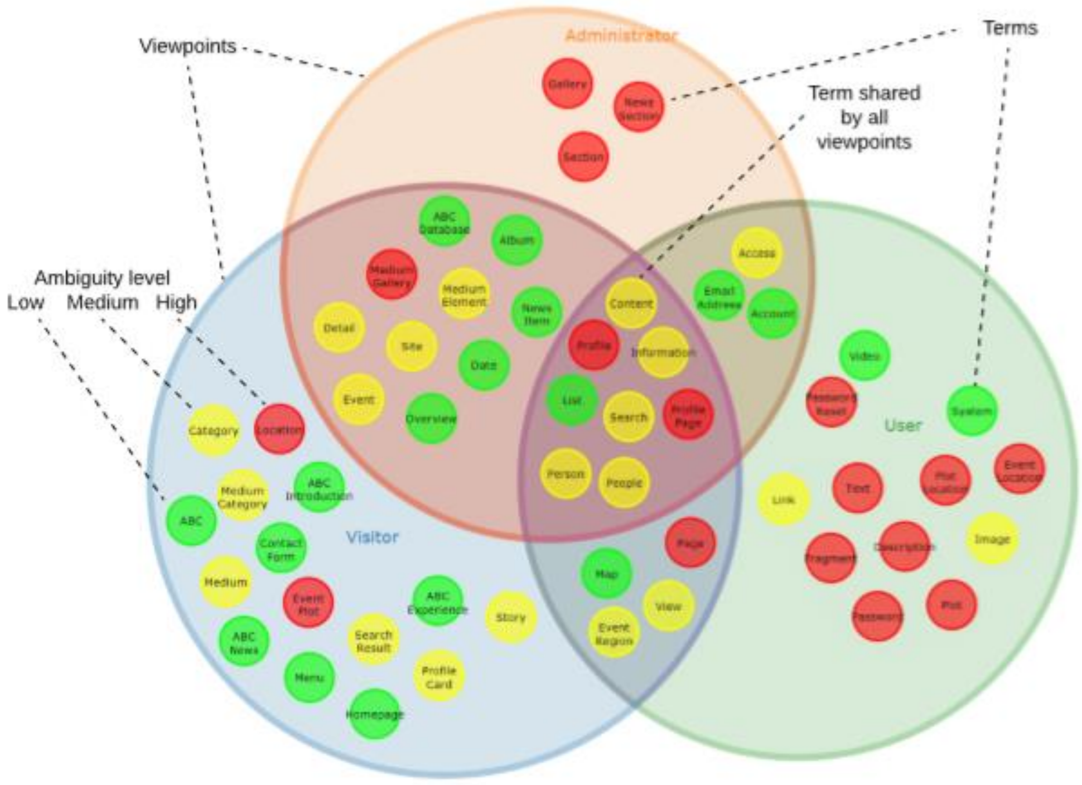


Figure 17 – The REVV tool with three roles and ambiguous terms (Dalpiaz et. al, 2018)

3.5.2 The Interactive Narrator Tool & Visual Narrator tool

Based on a set user stories, the Interactive Narrator tool can extract all relevant entities and the relationships between these from the user story set (Slob, Dalpiaz, Brinkkemper, & Lucassen, 2018). The interactive narrator tool utilizes the concept model extracted from a set of user stories by the visual narrator tool. The interactive narrator tool then combines this information with data from an issue tracker, such as epics, themes, and sprints.

This is then represented as seen in Figure 18, where the entities are shown as nodes, and the relationships are labelled based on the nouns and verbs in the user stories. The interactive narrator tool allows users to inspect the model interactively, and includes zoom and filter functionalities. The goal for the tool is to support practitioners in the creation and communicating about high-quality user stories. This tool could be used to support users in getting an overview of the user story set, and help with the detection of missing user stories, or erroneous user stories. If this tool were to be included in the situational method it could improve the user story sets, and result in a more complete feature diagram to form a basis for development. This could limit the number of missed requirements and functionalities.

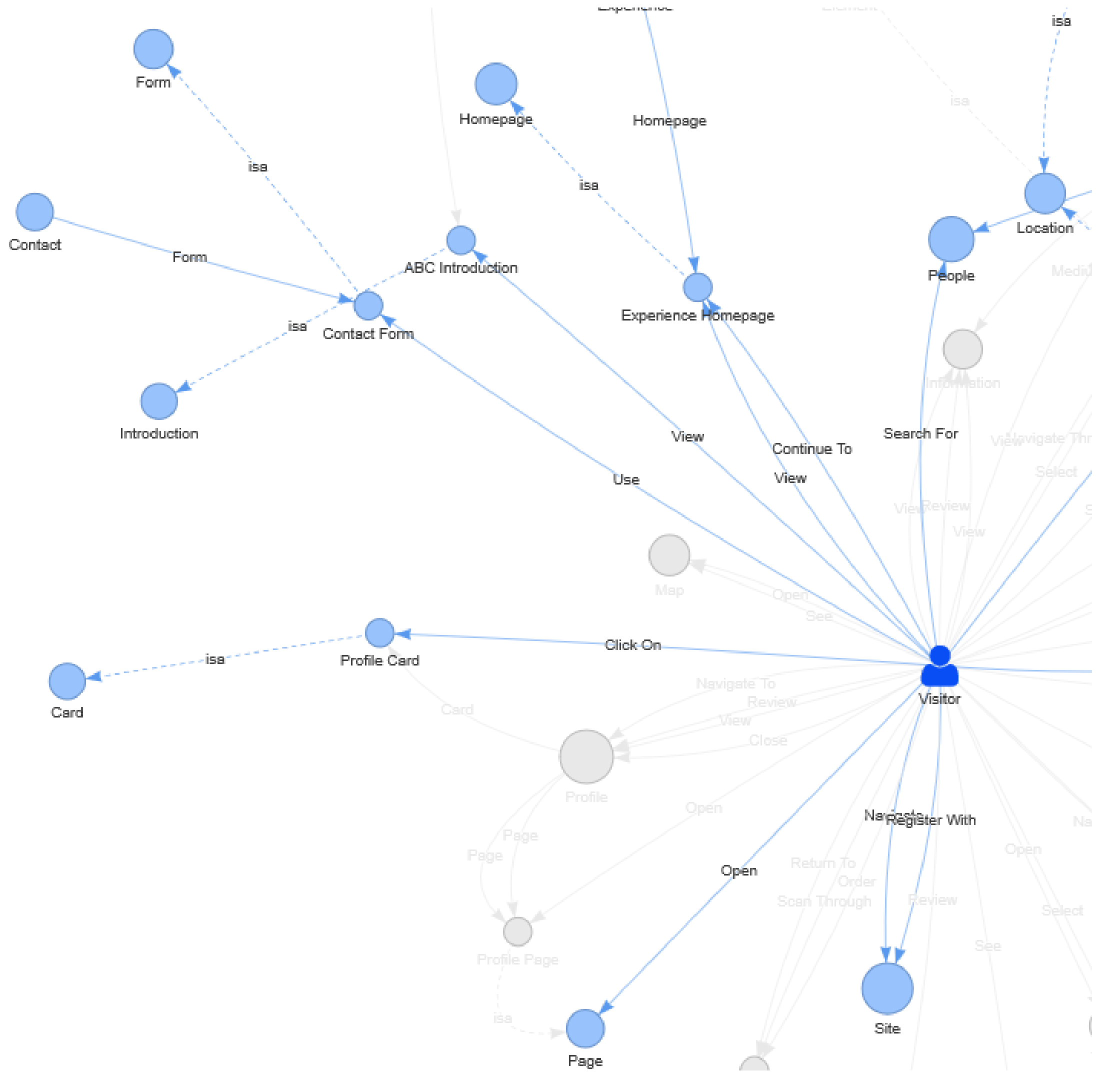


Figure 18 – The Interactive Narrator tool (Slob, et al., 2018)

3.5.3 Ontology Tooling in the Scope of this Research

Although ontology tooling could be a valuable asset to applying the RE4SA model, due to a limited number of available tools, and current issues with the tools discussed above, it will not be possible to include ontology tooling in the case studies for this research. While they will not be included in this research, ontology tooling could provide additional benefits for future applications of the RE4SA model.

Chapter 4

RE4SA for Enterprise Software

In this chapter the RE4SA model will be analysed to see if it can be applied to enterprise application software. This will be based on the findings in literature discussed in chapter 3, information gained from interviews and internal documents at Forza and their customers. This chapter will first discuss the use of user stories and epic stories to document requirements and contrast this to how requirements are currently documented. Then the benefits of software architecture based on the RE4SA model will be discussed. The model will then be considered in this context by applying a SWOT analysis. Finally the method used by Forza will be modelled in a product deliverable diagram (PDD), and how the concepts of the RE4SA model can be introduced in this method.

4.1 Requirements Engineering

Research has long since shown that poor requirements engineering can lead to problems in system design. Jenkins (1984) found that over half of the systems they investigated had to revisit the requirements at some stage of design, PMI (Smith, Bieg, & Cabrey, 2014) reported in 2014 that 47% of unsuccessful projects fail due to poor requirements management. These issues are the cause for 5.1% of wasted money on projects and programs (Smith et al., 2014). Alvarez & Urla (2002) state that for both system build and ERP selection accurate determination of requirements is important to both researchers and practitioners.

One difficulty in requirements engineering is that requirements are spread over many sources, such as multiple problem owners, stakeholders, documents and other systems. There are also many techniques to elicit requirements (Zowghi & Coulin, 2005) for example interviews, questionnaires, focus groups, task analysis, observation, prototyping. For documentation of requirements there are also multiple methods (Cziharz, Hruschka, Queins, & Weyer, 2005), like user stories, scenario modelling, information structure modelling, goal modelling, and unstructured documentation.

Because there are many different information sources, techniques, and documentation methods, there is no specific method that is used in practice. It can differ for each company, and even within a company due to differences in individual approaches. There are models that provide a method for requirements engineering, for example Daneva (2004) used the Accelerated SAP model to evaluate requirements engineering practises for 67 subprojects, based on this evaluation she discussed 14 lessons learned about requirements engineering in ERP systems. The first lesson identified by Daneva (2004), is to reduce barriers to cooperation, for this suggestion the RE4SA concepts can prove very useful as they use a natural language description of the problems. This issue can also be mitigated by visualising the design in an ontology and in the software architecture. A high number of companies, including Forza, use JIRA to keep track of open tasks, issues and features to be developed. JIRA already allows for the use of user stories, and epic stories could be used as an alternative to the SCRUM epics.

In the interviews it appeared that there was no specific way of documenting requirements at Forza, and it was mentioned that it depended on the person, and on the project. The interviewees were enthusiastic about the use of user stories and epic stories in the context of ERP customisation. However, it does appear that jobs to be done are not really suited to the implementation of enterprise software, as the application is already made to fulfil one or more jobs. For example the job mentioned in 3.2.3 for SCANMAN ("Help me spend less time on invoice processing.") is already considered when the customer chooses to order the add-on. This means that the Jobs to be done would only be relevant when a new add-on is created, but not when an existing add-on or enterprise system is implemented. When new jobs are identified during an implementation, it can be very beneficial to keep these documented so they can be used to come up with new software products to add to the company portfolio.

New epic stories can still arise during implementation when the customers want to find a solution to a problematic situation that falls in the context of the problem solved by software but is not included, for example a prepare for audit module. User stories can effectively show the smaller scale customisations that need to be done, for example when a customer wants to be able to edit a field in a view that is not initially supported.

4.2 Software Architecture

There is very little literature on software architecture in ERP systems. Somers and Nelson (2001) include the architecture in their overview of critical success factor for ERP implementation, and mention that architecture considerations should occur early in the implementation process. They state that this is important in ensuring compatibility of existing tools and identification of addons such as data warehouses. Rozansky & Woods (2012), explain that as software systems get bigger, good software architecture design facilitates more effective collaboration, allowing bigger teams to work on an application under the supervision of a software architect. Enterprise software are generally large software systems with many functions, which would indicate that software architecture is important for these systems. Oracle describes the JDE EnterpriseOne architecture as:

> *"A complete suite of modular, preintegrated, industry-specific business applications designed for rapid deployment, ease of administration, and low total cost of ownership. It is ideally suited for organizations that manufacture, construct, distribute, service, or manage products or physical assets.*
>
> *JD Edwards EnterpriseOne Tools is the runtime architecture and integrated development environment at the heart of the EnterpriseOne architecture. JD Edwards EnterpriseOne Tools is the technology layer that integrates with database systems, web application servers, reporting tools, and other third-party solutions. JD Edwards EnterpriseOne Tools provides an abstraction layer that allows the JD Edwards EnterpriseOne applications to function without a direct dependency on the platforms and middleware beneath them." (Oracle, 2015)*

The only publicly available architecture figure for JDE that could be located is an overview of the application suites in JDE (Figure 19). This is one possible way to show the functional design of the system on a high level of abstraction, functional architecture diagrams would also show the interactions between the modules within one application suite, and between the application suites. Feature diagrams could show the features of the application suites on a much higher level of detail.

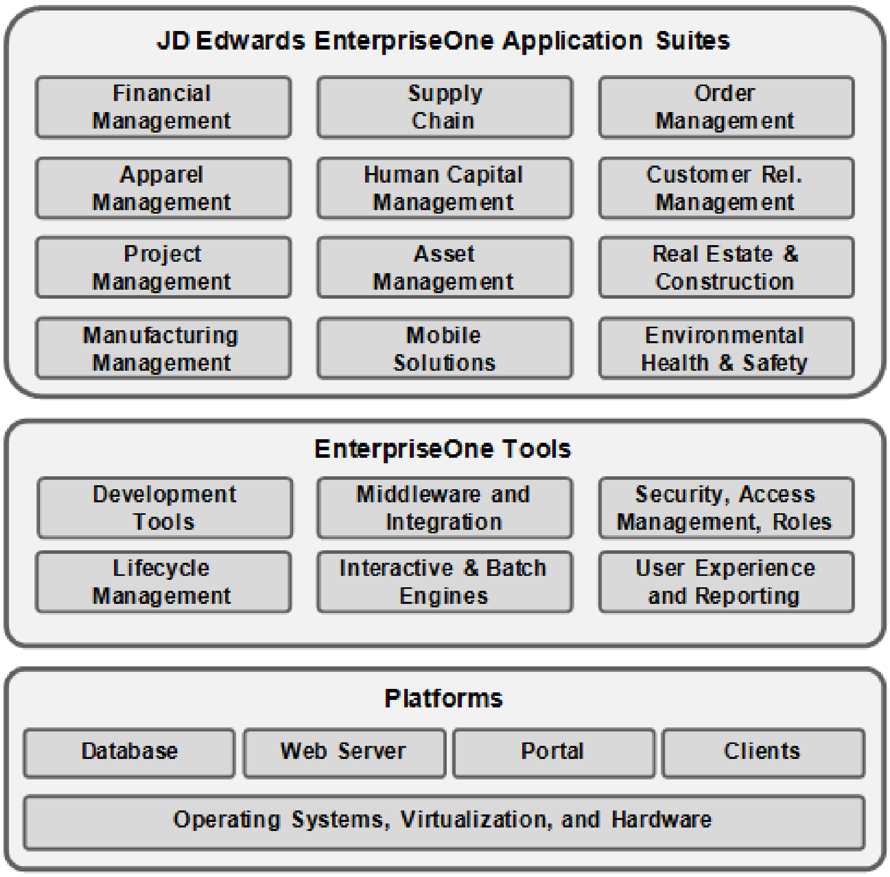


Figure 19 – JD Edwards EnterpriseOne Application Suites (Oracle, 2015)

While software architecture is getting more mature, in practise there is room for improvement in the documentation of software architecture (Clements et al. 2003). Designing a good architecture can be a challenge, and architectures should be documented from different viewpoints (Bass et al., 2013; Clements et al. 2003; Rozansky & Woods, 2012). The viewpoint catalogue by Rozansky and Woods (2012) can be seen in Table 1. While the software architecture concepts included in the RE4SA model cover the functional viewpoint, and the user stories and epic stories cover the context viewpoint, the other viewpoints are not included in the RE4SA model. This means that other documentation might need to be created to create a full overview of a systems architecture.

Viewpoint	**Definition**
Context	Describes the relationships, dependencies and interactions between the system and its environment.
Functional	Describes the system's runtime functional elements, their responsibilities, interfaces, and primary interactions.
Information	Describes the way that the system stores, manipulates, manages, and distributes information.
Concurrency	Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts

	of the system that can execute concurrently and how this is coordinated and controlled.
Development	Describes the architecture that supports the software development process.
Deployment	Describes the environment into which the system will be deployed and the dependencies that the system has on elements of it.
Operational	Describes how the system will be operated, administered, and supported when it is running in its production environment

Table 1 – Viewpoint catalogue by Rozanski & Woods (2015)

Keeping track of the software architecture can be especially valuable in the context of customisations of enterprise software, as it can create a clear view of what is actually changed for the specific customer. For each customer a template with the feature diagram for the application can be used as a starting point, and the customisations for the customer can be added to this template in a different colour. This overview can be very useful to make retrofitting more efficient, instead of detecting the customisations from tags in the code or by manually looking through documentation of the issues, a feature diagram can directly show which customisations have been made and might need to be included in a version upgrade.

When a new module is identified and should be added to the application, it's interactions with the existing modules can be shown by including it in a Functional Architecture Diagram. This allows developers to see how the new module uses information already contained in existing modules and helps them develop the new module more efficiently. When a new module needs to be developed for a customer, this module should also be considered for inclusion in a new version update of the application.

4.3 RE4SA Method for Enterprise Application Software

In this paragraph, the model as a whole will be discussed. First, the use of a method using the RE4SA model will be discussed through a SWOT analysis. Next, the method currently used at the case company will be modelled using a PDD, and analysed to see how parts of the RE4SA model can be integrated in the current method to try and improve it. This method will then be used in the next chapter to perform the case studies.

4.3.1 SWOT Analysis

In a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis, a company or product is analysed to detect both internal (Strengths & Weaknesses) and external (Opportunities & Threats) positives and negatives. In this section we have considered the SWOT concepts from the viewpoint of the RE4SA method, this means the internal factors will detail the positives and negatives of the method, and external factors will consider the use of other methods in the same company. This SWOT Analysis is based on theory, and was done to identify both positive and negative aspects before starting on the case studies. The results of this analysis can be found in the table below.

Strengths	Weaknesses
• More documentation, in the same format for every project. • Software architecture allows for more efficient coding and communication • More room for interpretation of requirements allows consideration of context. • Natural language based requirements. • Natural language reduces barrier to cooperation. • Easier communication with customers • Single structured method for every project. • Increase developer motivation by making them part of the story as opposed to giving them a task list. • More effective creation process due to higher focus on requirements gathering. • Traceability between user stories, epics, modules and features. • More goal focused.	• More time spent on documentation. • More time spent on gathering requirements. • Granularity issues can make it difficult to decide what is a module / feature or epic story / user story. (See barista problem (Blessinga, 2018)) • Natural language format can cause ambiguities. • New employees need to learn the method • The method is new and not thoroughly tested, so benefits might turn out lower than expected. • The RE4SA method might not include enough details on the how of a solution, as it focuses on who, what and why.
Opportunities	**Threats**
• Use upcoming NLP techniques that support user stories. • Ability to assess the quality of requirements (QUS / AQUSA). • Automatic generation of artifacts from code (currently being researched). • Crowd sourcing requirement elicitation through user stories (currently being researched). • As it is an active field of research new findings and opportunities will be reported over time.	• Other methods might provide better results. • Employees might prefer other methods, causing adoption failure. • The focus on FAD and feature models might limit the use of other visualisations for modules and features.

Table 2 – SWOT analysis of a method based on RE4SA

4.3.2 Creating the Situational RE4SA Method.

In order to design the method that will be tested in the case studies, it was first necessary to get an overview of the method currently used at the case company, Forza IT group. In order to model the current method, staff members involved in the process were asked for their insight using semi-structured interviews. In total ten interviews were conducted to determine the current method, these covered the following functions:

- Two developers
- Three functional consultants
- One technical consultant
- Two project managers
- Two customers

In these interviews the interviewees were asked to define the kinds of customisations done for customers, preferably with examples. They were also asked about their current process for creation of customisations, and of implementations in general, what information they were supplied in such cases, and their opinions on the RE4SA model in this context.

After the interviews, internal documentation was analysed to further gather information about the current method. Based on this information, the method was modelled using a product deliverable diagram (PDD). A PDD combines a UML activity diagram and UML class diagram. In a PDD the processes are modelled on the left and the deliverables on the right side (van de Weerd & Brinkkemper, 2009). The PDD is explained in two tables, one describes the activities, and the other table describes the concepts. The customisation section of the current method can be seen in Figure 20. For the PDD tables, see appendix C.

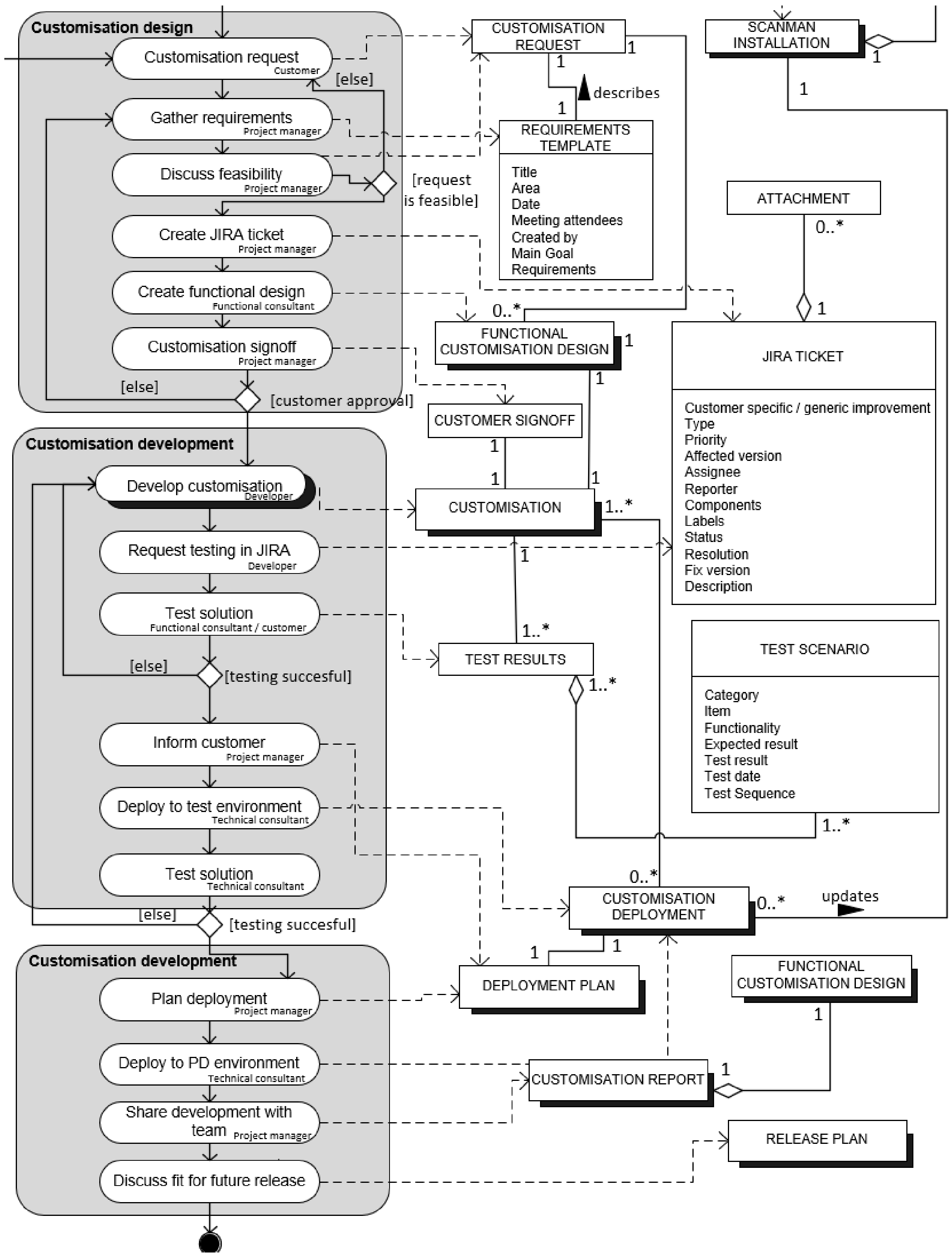


Figure 20 – partial PDD of the current customisation method at Forza IT group.

After the current method was modelled, it was verified with three employees at the case company, and modified to what is seen in Figure 20. From this point, it became possible to analyse where in the current method fragments of the RE4SA method could be integrated. Based on findings in the interviews and analysis of the current method, it was concluded that the RE4SA model would not be suited for the implementation process. The implementation process is mostly a configuration using pre-existing options, in this scenario the specific settings are deemed more efficient than figuring out what the customer wants from the product. For the customisation however, the RE4SA concepts can be integrated, since for developing new functionality, the advantages listed in the SWOT analysis can potentially improve the method.

In order to integrate the RE4SA method into the current method, the situational method engineering approach for method construction (Van de Weerd & Brinkkemper, 2009) was applied. This means that we analyse the current method to see where the needs are for improvement, and then consider how method fragments from the candidate method (RE4SA) can be implemented to meet these needs. From the inclusion of these method fragments, a new method is assembled. This means that the proposed method is based on the current method, minimising the required change in business processes.

As mentioned before, the RE4SA fragments should be included in the customisation section of the method. This means that the analysis will focus on locating what fragments of the current customisation method should be replaced with fragments based on the method formalised by Blessinga (2018).

The high level version of the resulting RE4SA method designed for the case company can be seen in Figure 21, with the blue coloured activities representing introduced activities. The more detailed version of the proposed method and the accompanying tables can be found in Appendix D. This method will be used in the case studies described in the next chapter to obtain real world data which can show the effectiveness of the RE4SA method. The biggest changes to the current method are in the creation of the functional design, as this will be done through the RE4SA concepts. Furthermore, we introduced the option to move towards development sprints to further utilise the possibilities of user stories. In smaller scale customisations, the user story set could simply include all user stories.

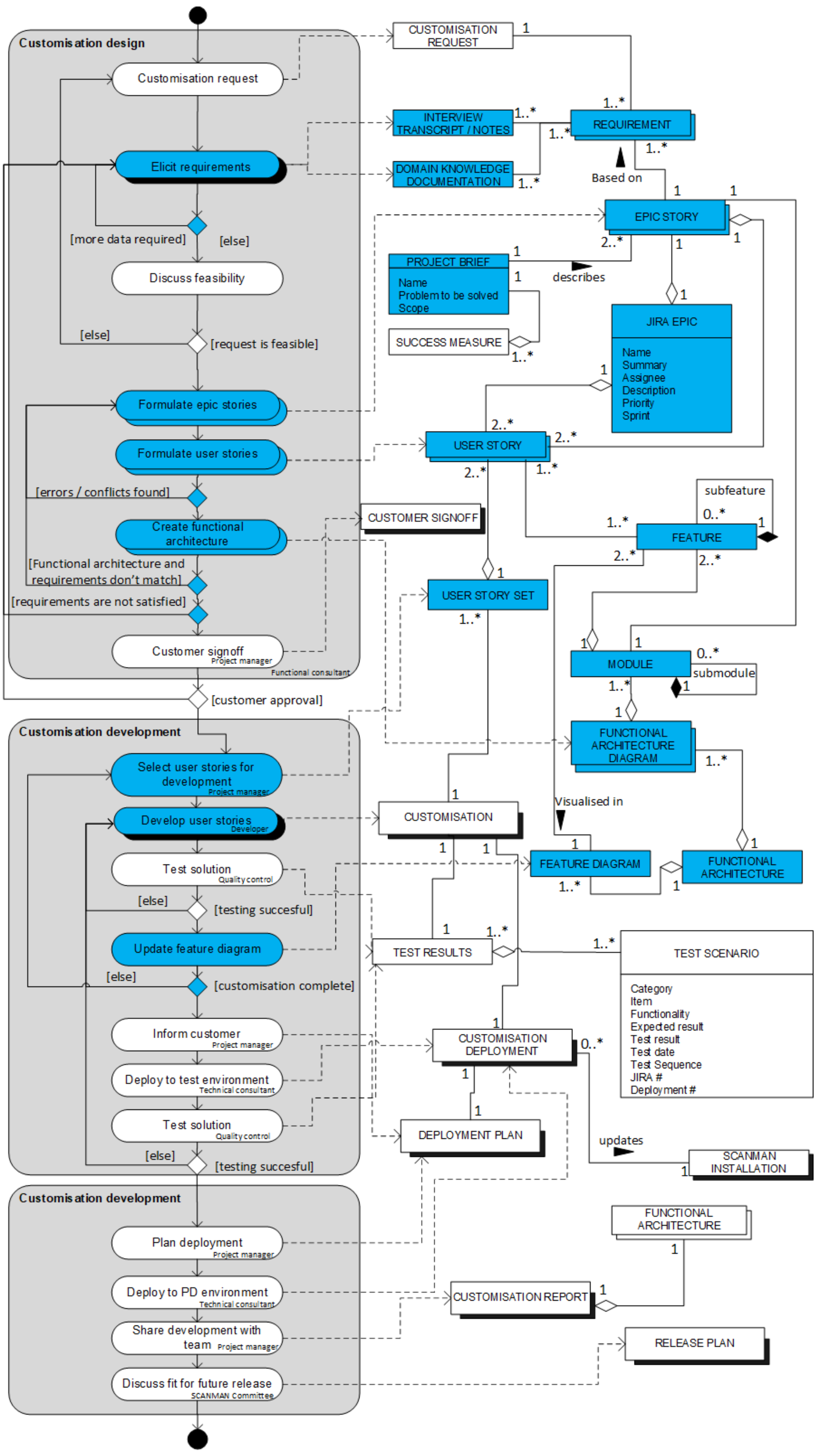


Figure 21 – The situational RE4SA method (short version).

Chapter 5

Case Studies on customisation

In this chapter, the main findings of four case studies will be discussed. First the reconstruction of the software architecture will be discussed, as having an up to date functional architecture of the current standard version was required to perform the case studies. Then the case study protocol will be discussed at a high level, for the specific information of the protocol refer to Appendix E. For each of the four cases some of the key findings will then be discussed, the case study reports will be included in Appendix F.

5.1 Case Study Protocol

In order to ensure that the case studies are valuable and their results are as accurate as possible, a case study protocol was set up (Appendix E). This case study protocol is based on the guidelines described by Runeson and Höst (2009), and is written based on the template defined by Pervan and Maimbo (2005). Besides the qualitative data gained from the interviews with employees (Chapter 6), the case studies will also result in content metrics, as described in the case study protocol.

A total of four case studies were performed, two of these were based on projects that were already completed and were based on the available documentation. The other two case studies focused on project that were still ongoing during the case study, for these cases the current method and situational case study method were performed side by side with the current method being leading in development.

The case study is multi-case theory testing, to identify how the RE4SA model can be applied, and to analyse the benefits of applying the model in the context of enterprise software customisation, and management. Each of the case studies de-scribes the application of the model to a project for the SCANMAN software product. By performing these case studies we test the applicability of the model for a range of cases, and use the outcomes as an input for expert evaluations of the model. An overview of the interviews per case can be seen in Table 3. For the case studies, the design phase of the method defined in 4.3.2 was applied, and the process was analysed. The outputs of these case studies will be evaluated in chapter 6, and the case study reports can be found in appendix F.

Case study	Description	Number of interviews	Roles
1	Customisation on feature level	2	Developer, C&C expert
2	Customisation on module & feature level	3	Functional consultant (x2), developer
3	Version update	3	Developer (x2), management
4	Recreation of the ES addon	3	Project manager, functional consultant, developer

Table 3 – Overview of interviewee's per case

5.2 Architecture Recovery

For the case studies it was necessary to first recover the functional software architecture of SCANMAN. This was done through reverse engineering the implemented version of SCANMAN.

> *Reverse engineering is the process of analysing a subject system to identify the components, interrelationships and create representations of the system at a higher level of abstraction. (Chikofsky & Cross, 1990)*

This reverse engineering was done through an analysis of the user manual, current version of the application, input from employees, and other available documentation. Sticking to the concepts of the RE4SA model, the architecture was visualised in a functional architecture model (FAM) (Brinkkemper and Pachidi, 2010) and a feature diagram (Jansen & van Rhijn, 2016). These recreated architecture models are supposed to function as a template that can be used when a clients version is changed from the standard version, or to identify focus areas for certain scenarios (e.g. an update of the ERP at a customer). Figure 22 gives an indication of how we can reconstruct the software architecture based on the user interface.

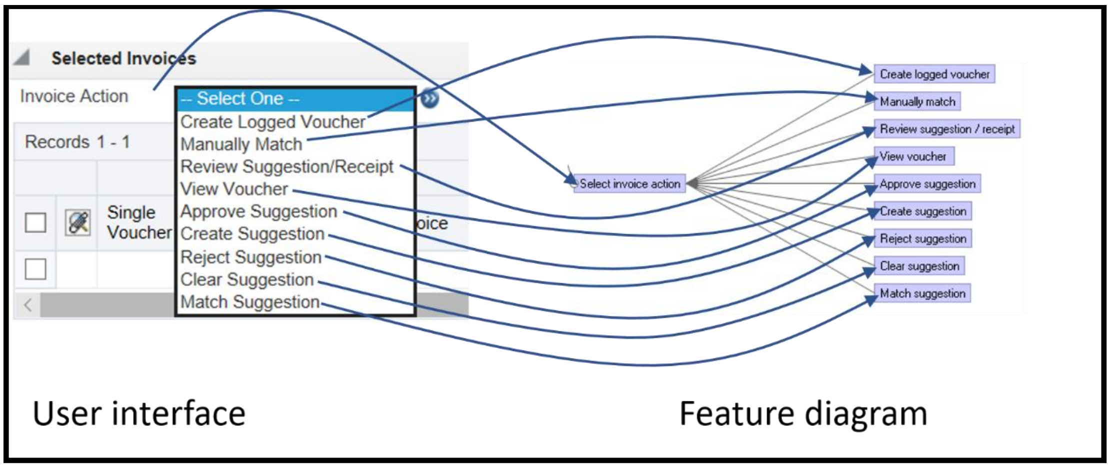


Figure 22 – Architecture reconstruction based on the user interface

After gaining context information about SCANMAN, the application was analysed and the interactive functions were modelled in a feature diagram, using Eclipse and the feature IDE plugin (Eclipse, 2019; FeatureIDE, 2019). Both the user visible features of SCANMAN, and the features of JDE on which SCANMAN relies were modelled. This resulted in a feature diagram containing more than 1300 features. In order to give provide business value with such a large figure, it was necessary to find a way in which the information could effectively be conveyed and to prevent information overload. This was done using the following visualisation techniques / choices:

- Collapse and expand features: Using the eclipse platform, users can collapse and expand the branches of the feature tree that are important for them. This causes the model to adhere to the Information-Seeking Mantra (Schneiderman, 2003): overview first, zoom and filter then details on demand.
- Colour coding to highlight and categorize features: this allows viewers to quickly identify the features that are important for their use case when they open the model. The colours were chosen based on how much they needed to stand out, but the choice was limited as

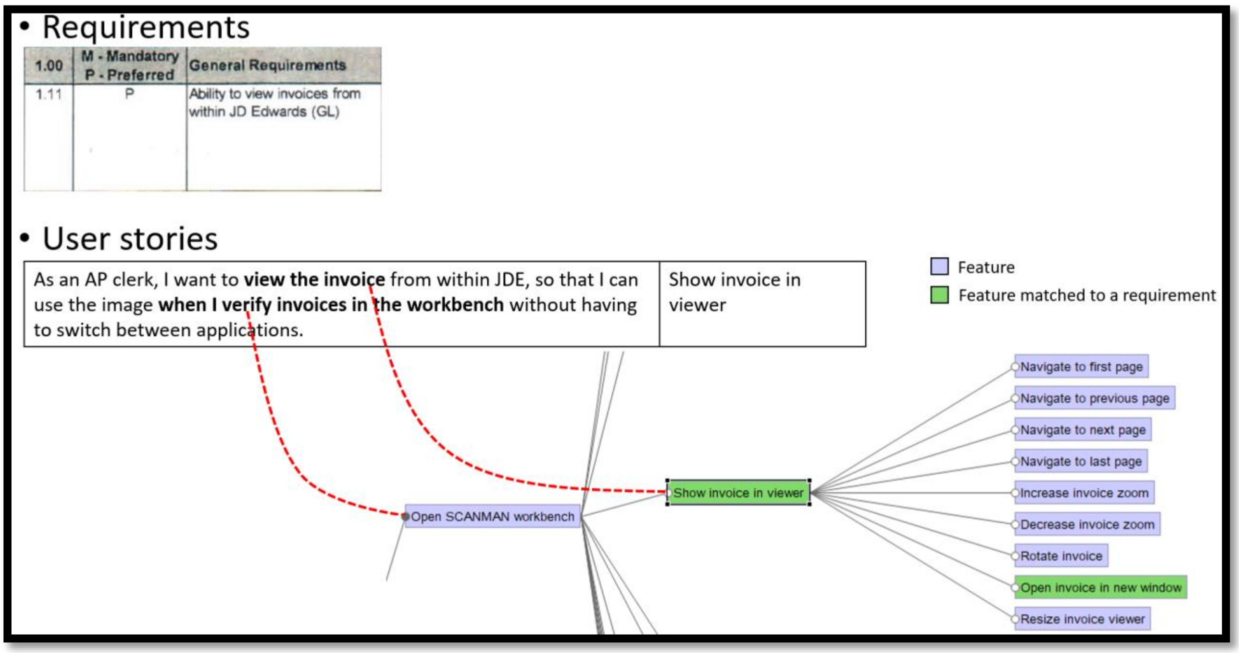
FeatureIDE only supports ten colours. It should not matter which colours are used, as long as the model is consistent, and a description of what the colours signify is documented.

- Customisations will be colour coded using red, to have increased visibility in the overview, allowing a quick insight in which features were changed or added for a specific installation.
- Features that were part of JDE on which SCANMAN is reliant were colour coded cyan. This can be important to detect which features should be checked for compatibility when JDE releases a new version.
- Features that did not work during the analysis were colour coded orange, these features might be reliant on context missing during the analysis, or require bugfixes.
- For the admin setup, options that are valid for all categories are modelled on the same level, and colour coded light grey (this colour is chosen as it does not need to stand out in the overview of features, as opposed to customisations)

- When relevant, non-interactive features were modelled as abstract features to group a set of features to prevent explosive expanding of the model

5.3 Case Study 1: Tracing Customer Requirements to the Software Architecture

The first case study was chosen because it was fairly recent, and had a clear documentation of the requirements of the client. This project was a small scale customisation for a customer that had only just bought SCANMAN. They supplied a list of 46 requirements, which were rewritten as 32 functional user stories, and 7 quality requirements. Only two of their requirements (or 3 user stories) required development as they were not currently in the standard version. Due to the small-scale of the requirements, all documentation was done on a user story level, and no new epic stories were formulated. All functional user stories that were already fulfilled in the standard version were matched to the feature diagram, and these were colour coded (Figure 23). This visualisation could be used to show the customer how their requirements are already met, and by which feature.



- Requirements

1.00	M - Mandatory P - Preferred	General Requirements
1.11	P	Ability to view invoices from within JD Edwards (GL)

- User stories

As an AP clerk, I want to **view the invoice** from within JDE, so that I can use the image **when I verify invoices in the workbench** without having to switch between applications.	Show invoice in viewer

Figure 23 – Visualisation of matching a requirement to an existing feature based on concept matching

Due to the large size of the feature diagram, providing the developer or customer with the full feature diagram is not very effective. This because the relevant information is hard to find in the total model, colour coding does make this process somewhat more effective as the search & zoom tactic can be used. However, providing sections of the relevant parts of the model can be an effective way for communication, in Figure 23 the feature diagram is used to locate requested functionality and displayed in a fragment of the feature diagram. This provides a limitation of the information overload, and highlights the important information.

For the new functionalities, the feature diagram could be used to locate an appropriate position to include the new features in the application. The process is similar to the matching of requirements to features. In order to do this, we use the wording in the requirement to locate the relevant modules and features. An example of this can be seen in Figure 24.

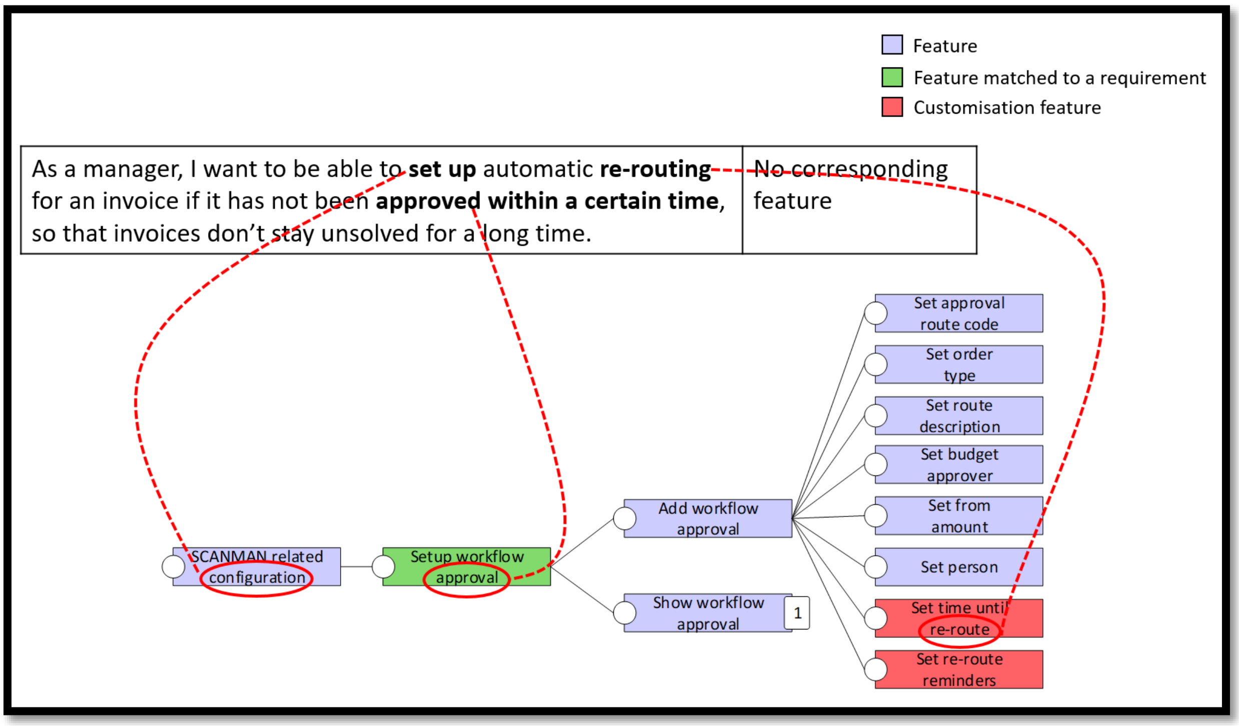


Figure 24 – Placement of new features from a user story based on concept matching

5.4 Case study 2: Customised Voucher Match Report

The second retroactive case study is a retroactive application of the situational method on the customisation for a different customer. This case study was chosen because, as opposed to the first case study, this case has a new functionality that is on an epic story level. Information was gained from staff members at Forza, JIRA issues and from observing a knowledge transfer session. This case study resulted in 7 unlinked user stories, and 1 epic story that contained 7 user stories (14 user stories total). As opposed to case study 1, we only considered new functionalities in this case. The epic story for this case is:

> *When I receive an invoice that does not match an existing purchase order, I want to receive a report when it exceeds the tolerance set in the ERP, so that the invoice can be rejected.*

The epic story details the error detection of a voucher match process. While the application would already automatically reject invoices when they don't match the purchase order, it would not explain the reasoning for rejecting such an invoice. The new functionality is the generation of a report of the rejected invoice, it will detail on what line item the issue is, and what the status of the VMA check is. For example, it will report "matched out of tolerance" when the difference between amounts is bigger than the tolerance set up in the ERP.

Mapping the epic story, and its user stories to the architecture revealed a limitation of the feature diagram. Since the feature diagram only models the user visible features, the customisation was modelled as part of the reporting module. However, most of the changes in the ERP to generate this report are done in the background, and are not visible to the users. This means that a developer unfamiliar with the customer environment can see that this customer specific report is generated, but where the logic to generate the report is located is not visible in the model.

The added module contained 23 features (Figure 25), while we had only 7 USs in the ES. We identified a set of explanations for the difference in the number of user stories and features:

- Features can be added based on gut feeling from the developer
- Features might have been considered 'common knowledge' and therefore do not need a user story to be included in the solution
- The included information fields already existed within the ERP so all relevant fields might have been selected instead of only the fields required for the use case
- Limited context knowledge for the case study might have caused missing user stories

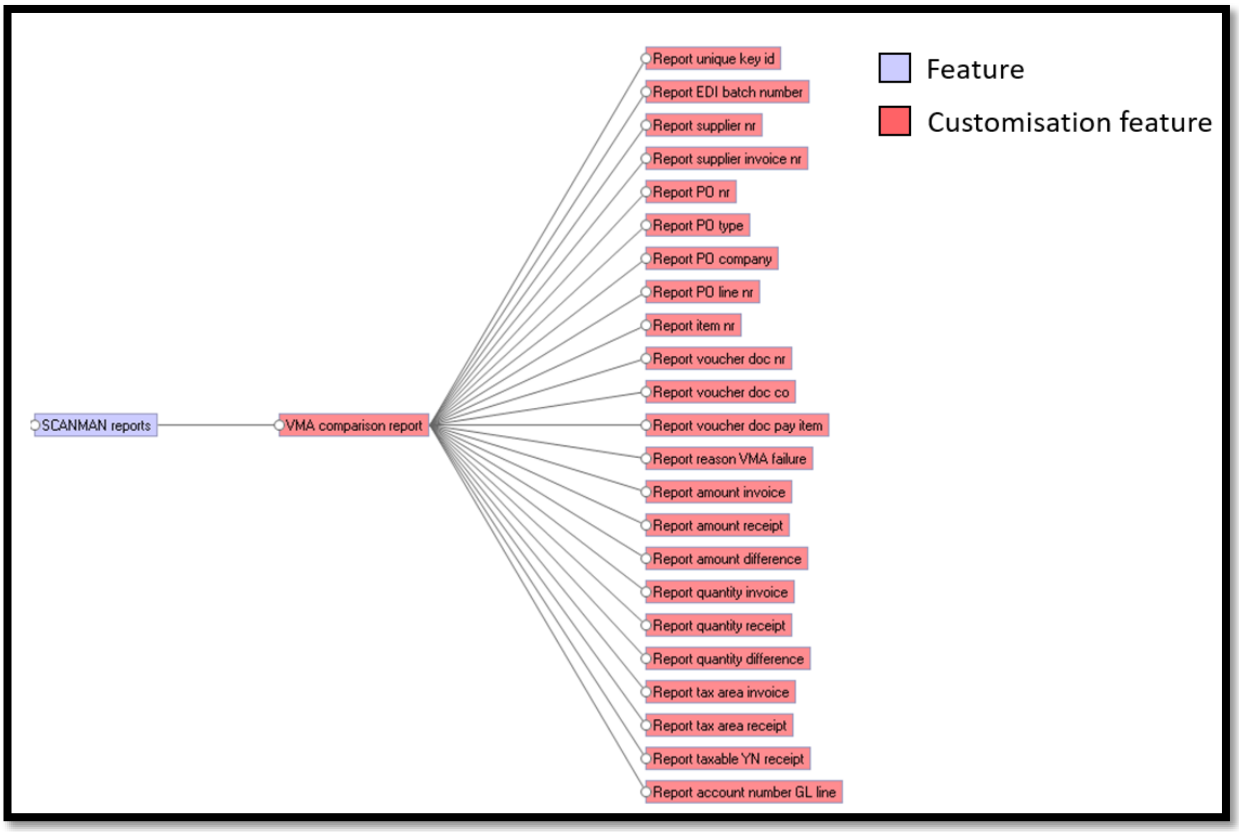


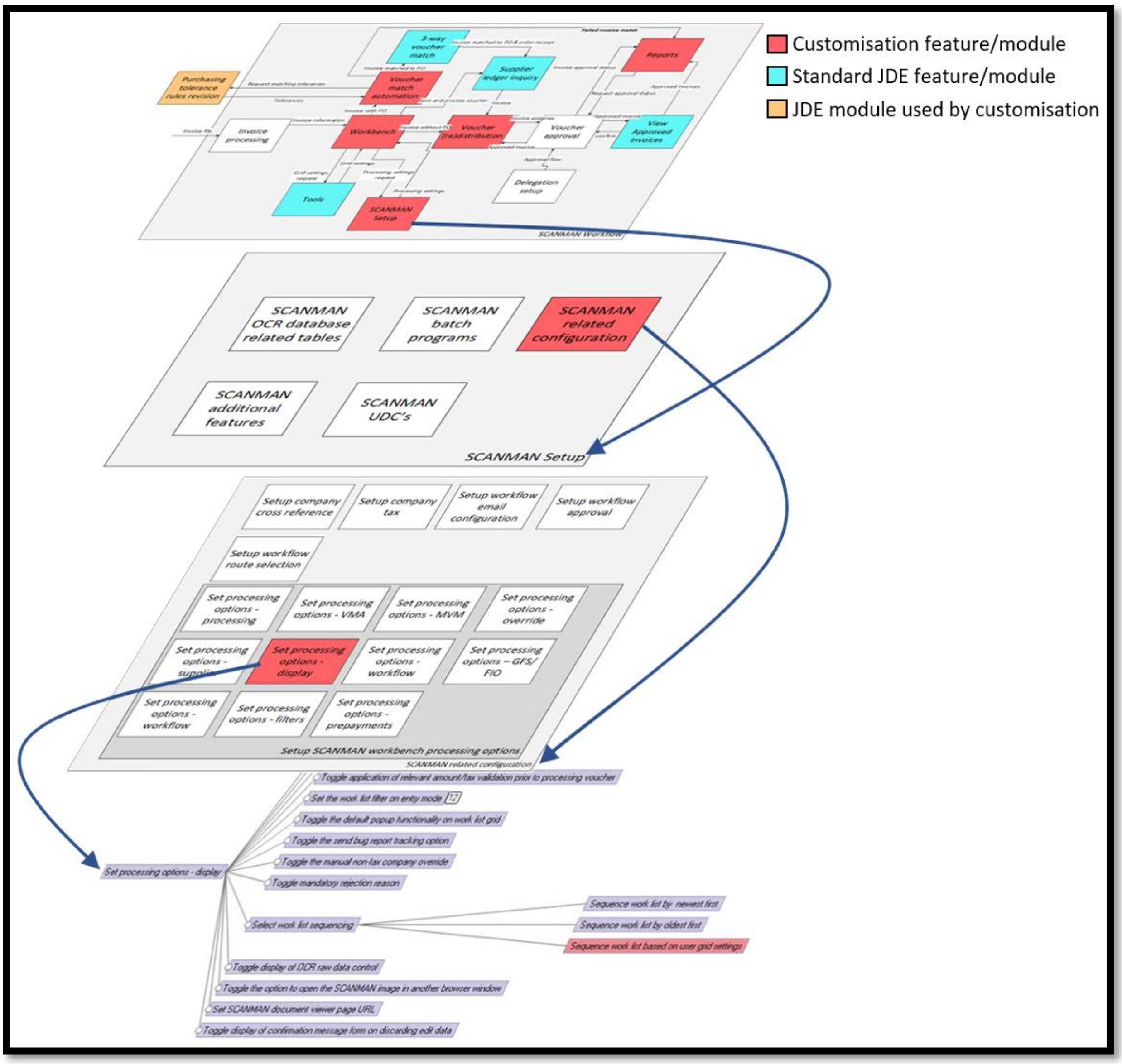
Figure 25 – Feature diagram of the added module

In this case study we extended the colour coding to the FAD so that we can provide an overview of which modules are modified. This supports the information seeking mantra as it allows the user to see which module is changed and more easily locate the customised features in the feature diagram. However, in doing this we were had to deal with the "barista problem" as described by Blessinga (2018). This barista problem describes the granularity issue in using ES and US (and jobs-to-be-done), as an ES can be written as a US by changing the wording slightly. This is complicated as on the software architecture side we can have multiple levels of both (sub)features and (sub)modules. It remains up to the requirements engineer to decide what is on the epic story level, and what is on the user story level. Blessinga suggested to determine the level of granularity of a user story as it is the most specific level of detail, and scaling up from there.

However, in this case our issue was on the software architecture side. In order to support the information seeking mantra, we wanted to allow for effective drill down from the high level FAD, to locating the customised feature. One such customised feature was on the sixth level of depth in the feature diagram (including modules). This made it hard to locate when moving from the FAD to the feature diagram, as even the smallest submodule (which could also be considered a feature based on the granularity decision) contained 194 features. In order to guide the user to this feature we first considered including this in the colour coding of the feature diagram. However, this solution would cause an exponential increase in colour coded aspects in the feature diagram and would diminish the highlighting ability. Therefore the solution we came up with was to add additional levels to the FAD to support the drilldown navigation. Although this meant that we modelled aspects that we considered features in a FAD, it was done based on the following reasoning:

Using the proper level of granularity for US and ES remains a difficult task. However, since each of the figures is a different view of the same information, it should not be an issue if certain aspects are contained in multiple views. These views can be added when they are needed.

The drill-down navigation through the different levels can be seen in Figure 26. The third FAD was added for the purpose of locating this customised feature without having to go through all features. Currently this drill down view can only be achieved by transitioning from figure to figure based on the naming, ideally this would all be possible through a single interface which automatically links the RE and SA elements. To reflect on the barista problem here, it is up to the requirements engineer & software architect to decide whether an element is a (sub) module / feature. For this case study we decided to model aspects that we considered features in a FAD.



Customisation feature/module
Standard JDE feature/module
JDE module used by customisation
3-way voucher match
Failed invoice match
Invoice matched to PO & order receipt
Reports
Supplier ledger inquiry
Invoice approval status
Invoice matched to PO
Purchasing tolerance rules revision
Request matching tolerances
Tolerances
Voucher match automation
Save and process voucher
Invoice
Approved invoices
Request approval status
Invoice with PO
Invoice information
Invoice file
Invoice processing
Workbench
Invoice without PO
Voucher (re)distribution
Invoice assignee
Approved invoice
Voucher approval
View Approved invoices
Approval flow
Delegation setup
Grid settings
Grid settings request
Processing settings request
Processing settings
Tools
SCANMAN Setup
SCANMAN Workflow
SCANMAN OCR database related tables
SCANMAN batch programs
SCANMAN related configuration
SCANMAN additional features
SCANMAN UDC's
SCANMAN Setup
Setup company cross reference
Setup company tax
Setup workflow email configuration
Setup workflow approval
Setup workflow route selection
Set processing options - processing
Set processing options - VMA
Set processing options - MVM
Set processing options - override
Set processing options - supplier
Set processing options - display
Set processing options - workflow
Set processing options – GFS/FIO
Set processing options - workflow
Set processing options - filters
Set processing options - prepayments
Setup SCANMAN workbench processing options
SCANMAN related configuration
Set processing options - display
Toggle application of relevant amount/tax validation prior to processing voucher
Set the work list filter on entry mode
12
Toggle the default popup functionality on work list grid
Toggle the send bug report tracking option
Toggle the manual non-tax company override
Toggle mandatory rejection reason
Select work list sequencing
Sequence work list by newest first
Sequence work list by oldest first
Sequence work list based on user grid settings
Toggle display of OCR raw data control
Toggle the option to open the SCANMAN image in another browser window
Set SCANMAN document viewer page URL
Toggle display of confirmation message form on discarding edit data

Figure 26 – Drill down navigation to locate customised feature.

Figure 26 also includes a module that is colour coded orange, this was done because it is a JDE functionality on which a customisation relies, but was not changed. For this we considered three different options:

- It could be kept cyan as it wasn't changed and is part of JDE functionality and this would match the legend, but would not highlight the module as a potential risk.
- It could be colour coded red, as it is linked to a customisation. This would catch the attention, but would not match the legend.
- It was colour coded orange as this colour is often associated with warning signs. While this might lose clarity on whether or not it is a JDE element, as it is not coded cyan, this was deemed less important than the increased attention from the colour choice.

5.5 Case study 3: Version Update

For the third case study the situational method was applied to an ongoing project for the new version update of SCANMAN. The version update collects requirements from many different customers that are to be included in the standard version of the case software. Some of these additions are already developed in specific customer environments as customisation, and need to be moved to the 'standard' software product. This is another scenario in which the SA models can be applied to identify the location of these customisations.

For the update the requirements were obtained through analysis of the issues and comments logged in the JIRA for the update. Based on these issues, and by attending meetings within the project team, a total of 3 epic stories and 40 user stories were formulated. In total the updated software architecture contained 104 new features, and 4 new modules (of which 3 were sub-modules). The difference in ES and modules can easily be located, as the QA epic:

> *When there is an issue with an invoice, I want a way to ask another JDE user a question, so that the issue can be resolved by the relevant user, or additional information can be supplied.*

Was solved by adding two new modules, one submodule for an invoice specific QA form within the workbench, and a module that provides an overview of all QA forms which was added to the high level FAD.

Like in the previous case study, we did not have a 1:1 relation between user stories and features but a 1:2.6 ratio. The reasons for this can be the same as hypothesised in case study 3. In this case we also found a few examples in which a user story is solved by a collection of features, as seen in Figure 27. The ratio being different than the expected 1:1 (Brinkkemper et al., 2019) is not necessarily a sign of bad requirements engineering or development, but instead is linked to the goal focus of user stories. Since developers have more freedom in determining a solution to fulfil the requirements, they might come up with innovative solutions by reusing existing features, or solving the requirement through a combination of features.

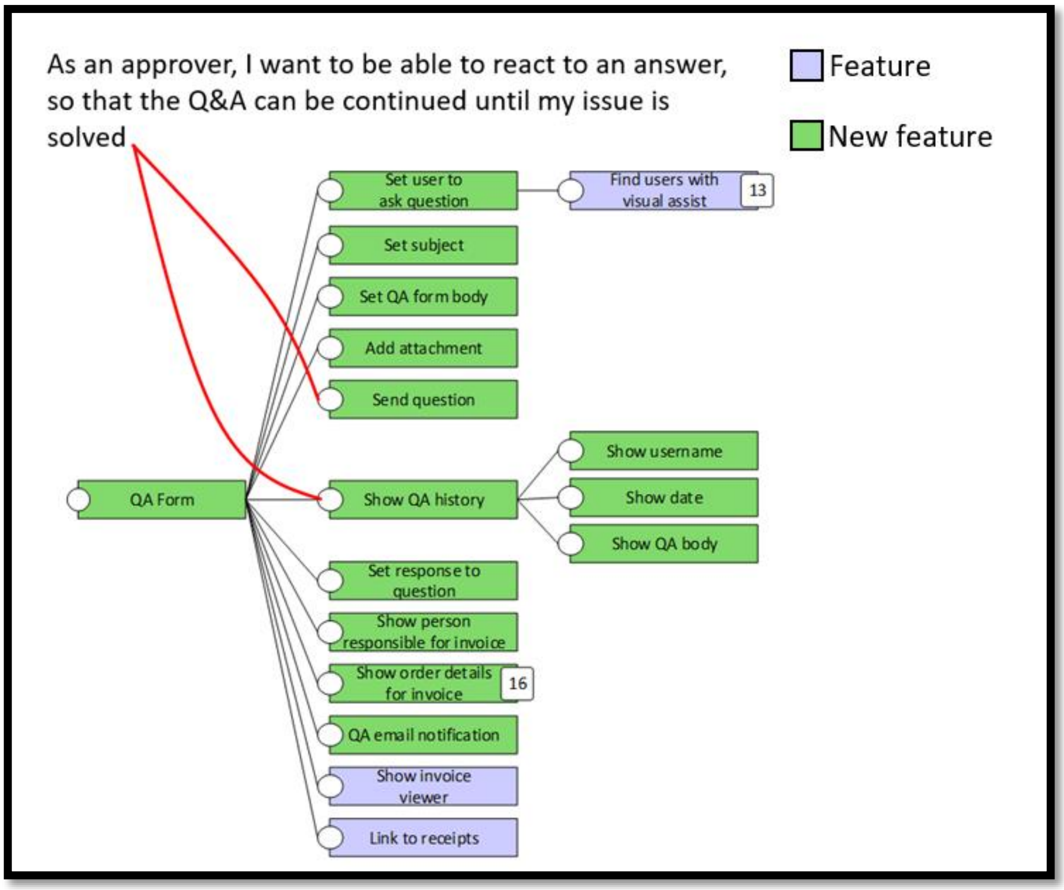


Figure 27 – An example of a user story that is fulfilled by a combination of two features

In order to create the project briefs, we would need to identify the related epic stories for each of the ESs. However, in the release update this is a bit problematic, as existing modules do not have a documented epic story. This meant that it would be necessary to recover the requirements for each module in the software, which would require a high amount of manual work for very little payoff. Therefore the problem was instead solved by relying on the link between ESs and modules as described in the RE4SA model. So instead of listing relevant ESs in the project brief, we just included the relevant modules in that section. This still enables detection of relevant parts of the software of which we need to be aware, but removes the need to completely recover the requirements for sections of the software that already exist.

When the new features were mapped on the feature diagram, we detected aspects that were not (correctly) modelled in the recovered architecture (in 5.1). This implies that by applying the RE4SA situational method over multiple projects or iterations of a single software product, we can improve the software architecture. In applying this method, we can ensure that the difference between the architecture models, and the actual software architecture is minimised. This also leads to alignment between the software architecture and requirements which should facilitate cooperation between software project management, and software architects.

5.6 Case study 4: Recreation of the Case Software for a Different ES

For the final case study we applied the method to a somewhat different case. The case study focused on the recreation of the functionalities of the case software (SCANMAN) in a different enterprise software environment (Oracle NetSuite). This case was selected as it could allow for comparison of the architecture of this intended solution to the architecture that was created in the earlier cases. Furthermore it would allow for a broader generalisation of the research, as we broaden the scope by applying the situational method to a project which customises an enterprise software product through an add-on solution, instead of customising an add-on solution and ERP for customer specific needs.

During this case study an initial design was created for the NetSuite add-on. In total this design consisted of 10 ESs, and 70 USs. Which were translated to 10 modules and 69 features for the intended software architecture of the design. The reason for the near 1:1 ratio of the concepts in this case study, is most likely because we compare it to the intended design instead of the actual architecture like in the other cases. The actual architecture could not be used for the comparison, as the add-on has not yet been created as of the time of writing. The findings of this case study do however indicate that the method can be applied to transform RE concepts into an intended architecture, on which the development can be based. During development, certain features will likely be changed, added or requirements will be solved through different means. Therefore, the intended architecture should be updated after development to reflect the actual architecture.

The design for this case study is supposed to fulfil the same business requirements as the JDE SCANMAN application. However, this design only has about 5% of the features of the reconstructed architecture of JDE SCANMAN (excluding the linked JDE features). There are multiple hypothesised causes for the difference:

- Since we defined the architecture up front, the design is likely more modular and loosely coupled which could result in a more efficient application that needs fewer features to achieve the requirements
- The design only describes the minimal viable product, while the JDE version has been used and expanded over the lifetime of the product.
- The compared numbers are for an intended and an actual architecture. The number of features for the actual implemented NetSuite add-on might result in a higher number of features. For example, the NetSuite UI elements will also be included if we recover the SA based on the UI.

The SCANMAN for NetSuite app is still pending development as of the time of writing. Therefore, it is impossible to include the comparison of the actual SCANMAN NetSuite architecture to actual SCANMAN JDE architecture in this thesis.

5.7 Case Study Reflection

Overall, the situational method was effectively applied in each of the case studies, and resulted in a clear documentation and extensive functional architecture. While it did take a high amount of time to recover the architecture, once this initial recovery was done, updating the architecture for the case studies was quickly done. For the fourth case, the architecture construction from scratch was effective, as it could directly be translated from the identified requirements. This indicates that the situational method or a similar method could be effective for both new and existing software products throughout their lifetime, but does take a time commitment if the product is already created but lacks a software architecture documentation. It should be considered here that the case software is part of an ERP application, which makes it likely to have more features then other applications as there are many low level features (table fields etc.).

Case	User stories	Features	Epic stories	Modules
1	32	21	0	0
2	14	28	1	1
3	40	104	3	4
4	70	69	10	10

Table 4 – cardinalities of RE4SA concepts in the cases

Chapter 6

Evaluation

The case studies have been evaluated through expert interviews with employees of Forza IT group, who were involved with the projects of the case studies. In this chapter the findings of these interviews will be discussed. Additionally the case studies were analysed to obtain a number of content metrics. The extended introduction of the protocol, and the question list can be found in appendix E.

6.1 Expert interviews

Each of the cases was evaluated in two or three expert interviews with experts involved with the projects of the case studies (11 total, see table 3 for details). No expert was interviewed for more than one case study. The experts had varying roles, so that the results would not be limited to a single viewpoint. For the evaluation interviews, the experts were first presented with an explanation of the RE4SA model through the maps example, and then the results of the case studies. After this they were asked a total of 11 questions (and 5 sub questions). The interviews were semi structured in order to allow for further questions when interesting topics were raised by the interviewees, or to ask for more information.

The experts had limited knowledge and experience with user stories, three of the experts had worked with user stories before. This limited knowledge is likely because the company currently doesn't use user stories. However, one of their new partners does and through this partnership, user stories are also known within the organisation from another source than this research. Despite the limited knowledge, overall opinions were very positive. All experts felt that the documentation through the RE4SA concepts adequately captured customer requirements and provide a very detailed overview of the software product. They also thought that the situational method would be suited to working with changes in customer requirements, as the modular nature allows for additions where needed. The experts viewed the resulting documentation as very detailed. Compared to current documentation it is more thorough and also captures the why, and not just the how of the solutions.

While all but two interviewees thought that the situational method outputs contained enough information to start development, they did mention that it would most likely be dependent on the developer and his familiarity with the software product. The developers that were interviewed did think the information would be enough for them. It was also mentioned that it would likely be less effective if development was outsourced, which ties into the familiarity aspect, but could also be dependent on the cultural challenges that often accompany outsourcing. This issue could potentially be mitigated by enforcing the solution identified in the intended architecture (through feature diagrams). This topic would be interesting for future research, but was out of scope for the case studies in this research.

Having the software architecture models available was positively received by all the interviewees, but their expected use cases for the models differed. The interviewees mentioned it was very nice to have an overview of the software product, and how the modules are linked. For new employees the SA models could help get an overview of the product and its features. Functional experts liked being able to zoom from the high level FAD to the detail level of the feature diagram, as they are generally focused on the interaction of modules. The developers mentioned that since they often work at a feature level, zooming out from this to the module overview would help them keep the bigger picture in mind. This contrast in identified use, implies that it should facilitate internal communication, as it helps functional experts to have a view of the details, and developers to see the high-level overview.

The SA models could also be used in locating where new features should be placed, as a checklist to see if every feature is included in the actual product, for testing purposes, determining risk for customizations and detecting dependencies with other elements. One use case that was rated to be especially useful by all 11 experts, is to identify customer specific alterations to the software as seen in case 1 and 2. One of the developers mentioned that every time the customers' environment was accessed, it was unknown what changes had been made in that environment. And that having a SA on a customer level (extending a template from the standard version) would save a lot of time. However, in order for the SA models to be useful, they need to be kept up to date.

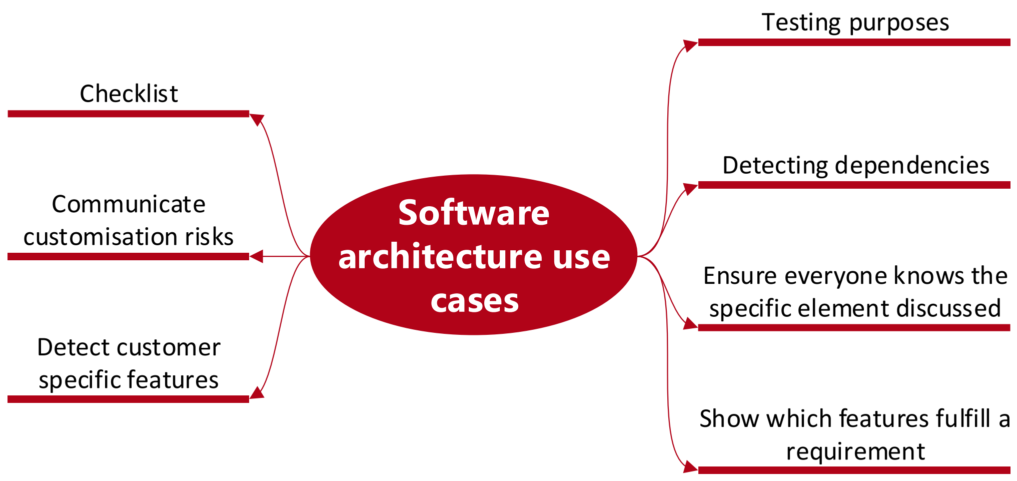

Software architecture use cases
Checklist
Communicate customisation risks
Detect customer specific features
Testing purposes
Detecting dependencies
Ensure everyone knows the specific element discussed
Show which features fulfill a requirement

Figure 27 – Some of the mentioned usecases for the SA models.

The interviewees were asked whether or not they would use the architecture in communication with customers. The response to this question was split, with four of the experts feeling like the models were too complex for communication with customers. They felt that in order to use the architecture models in communication with customers would require additional preparation of the models so only specific parts are shown (similar to the case study figures). The other opinions were more positive, and identified use cases in which they would use the SA models. For example, to ensure that ensure they are talking about the same aspects, to explain a customer how the software works, or to ensure that the features fulfil requirements. It was also mentioned that the figures could be used to visualise and communicate the impact and risks of a requested customisation.

The project briefs were rated as a nice to have by four of the experts, while all others rated it as a must have. These results support the findings by Blessinga (2018) and van de Keuken (2017) that project briefs, or a similar document providing extra context should be included when the RE4SA model is used in practise. The experts that rated them as nice to have did mention they were important, however not especially in their tasks but for different roles in the organisation.

Although all experts said that the situational method would be an improvement over the current method, they also acknowledged some disadvantages of the method when prompted:

- Requires discipline and time to keep up to date, and if not kept up to date the documentation loses a lot of value.
- Due to the size of the application, the feature diagram can be overwhelming which could limit its acceptance within the organisation.
- Increases the time spent on documentation, which frontloads more work, but (hopefully) reduces the need for rework and improves the level of documentation.
- Developers might feel limited by the solutions mapped in the feature diagram. This could however be mitigated by communicating clearly that it is only a suggestion.
- Enforcing the use of the method and ensuring that all employees have the required skills to create the documentation as described in this research, might prove challenging. It would require both support from the management layers and the operational layers of the company.

6.2 Content Metrics

The content metrics section is split in two parts, the first part will discuss the content metrics for the cases, and the second part will be an analysis based on the complete feature diagram. The content metrics can potentially be used for hypothesis forming for future research, and can be used in other research as a dataset for a larger scale empirical research. For the feature diagram analysis, a template was applied that was used in a different GRIMM research group project, this ensures that the research data can be a resource for related future research on content metrics.

6.2.1 Case Study Content Metrics

The first and simplest of the content metrics is the cardinality between the RE4SA metrics. Here we considered the ratio between user stories and features, and the ratio between epic stories and modules (Table 5). This indicates that there is not necessarily a 1:1 relation between either of the concept combinations. There are cases where we have more US than features, and the other way around. In the case studies we already hypothesised some explanations for this. Modules and epic stories are closer to the 1:1 relation with only one exception, however due to the lower number of ES (compared to US) the sample size is also smaller which makes the results less generalizable. The ES: US ratio, can differ from the total US divided by total ES, because there are US that are linked to existing modules in some of the cases. The average amount of user stories per epic in the cases was 7.6; and the average number of features per module 20.3

Case	**User stories (US)**	**Features**	**US : Features**	**Epic stories (ES)**	**ES: US**	**Modules**	**ES: Modules**	**Module: Features**
1	32	21	1.5 : 1	0	N/A	0	N/A	N/A
2	14	28	1 : 2	1	1: 7	1	1 : 1	1 : 28
3	40	104	1 : 2.6	3	1: 8	4	1 : 1.3	1 : 26
4	70	69	1.01 : 1	10	1: 7	10	1 : 1	1 : 6.9

Table 5 – cardinalities of RE4SA concepts in the cases extended

While we have disproven the assumption that there would be a 1:1 relation between the requirements and architecture concepts, this does not necessarily take away value of the RE4SA model. One of the advantages of user stories compared to other requirements management techniques is that they are goal focused. This goal focus means that we do not care how requirements are met, as long as the stakeholder can achieve the intended goals. If we extend this, we can also say that it doesn’t matter how many features are required as long as we reach the goals. One example of this, is the QA module from case 3, from the requirement we could assume that some form of internal chat would be required. However, in the end this was implemented as an overview of all questions and answers for an invoice, allowing other users to access the communication which facilitates knowledge sharing. So while it did take more than a single feature, there are also benefits as it provides more value to the users.

6.2.2 Feature Diagram Metrics

An analysis on the complete feature diagram was also performed, for this the analysis is based on the 3.8 version of SCANMAN and the relevant JDE modules, as this is the most up to date feature diagram of the software. In total this update added 4 modules, 93 atomic features and 10 composite features. Including these new functionalities, SCANMAN has a total of 66 (sub) modules, 135 composite features and 1399 atomic features. For each of the (sub) modules the average and standard deviation of the degree and depth were calculated, the explanation of these terms can be found in Table 6. These concept metrics are based on the Quality in Use Integrated Measurement (QUIM) model by Seffah et al. (2006), these metrics can help determine usability aspects of GUI style applications. Figure 28 visualises the degree and depth of a feature diagram. The values were averaged over all modules to obtain the average values for the application. The metrics can be seen in Table 7.

Concept metric term	***Description***
Composite feature	A composite feature, is a feature that contains one or more sub-features
Atomic feature	An atomic feature is a feature that does not contain any sub-features. These are the final nodes of a feature tree.
Degree	The degree details how many features a modules or composite feature contains. For an example see Figure 28
Depth	The depth of a feature is the number of composite features between the module and the atomic feature. For an example see Figure 28

Table 6 – Explanation of concept metric terms used in this research

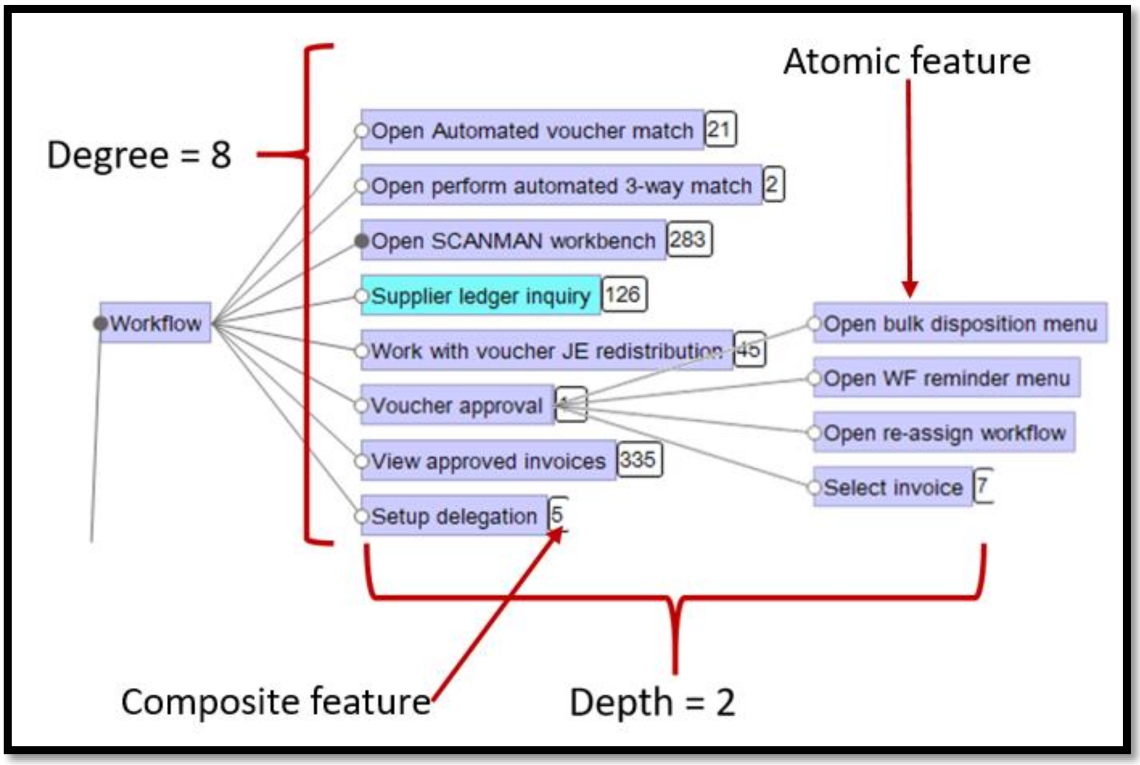

Atomic feature
Degree = 8
Open Automated voucher match 21
Open perform automated 3-way match 2
Open SCANMAN workbench 283
Supplier ledger inquiry 126
Workflow
Work with voucher JE redistribution 45
Voucher approval 4
View approved invoices 335
Setup delegation 5
Open bulk disposition menu
Open WF reminder menu
Open re-assign workflow
Select invoice 7
Composite feature
Depth = 2

Figure 28 – Example of the content metrics

Metric	**Value**
Number of modules	11
Number of submodules	55
Average atomic features per (sub) module	20,6
Average degree	9,2
Max degree	323
Min degree	1
Average standard deviation of degree	9,8
Average degree without grids	5,0
Average standard deviation of degree without grids	2,2
Average depth	2,1
Max depth	6
Min depth	1
Average standard deviation of depth	0,4

Table 7 – content metrics on degree and depth of the feature diagram (all modules)

The standard deviation of degree is rather high. This implies that there is a big difference in the degree between modules. This can be explained due to the nature of the software product, as it stores a lot of information, there are modules that contain a high number of features and show different elements in a grid (e.g. the View approved invoices in Figure 28). If we were to ignore these grids both the average degree and standard deviation of degree is a lot lower, and likely better comparable to other software products.

While the content metrics for SCANMAN can't necessarily be used to define specific aspects of the software product, hopefully the data for this software product and other software products will allow us to define specific characteristics of software based on the feature diagrams. For example, we hypothesise that a low level of depth is likely to indicate that a software product is relatively user friendly, as it requires less clicks to get to a desired features. Or a high degree might indicate information overload, as a lot of features are offered at the same time. Due to confidentiality reasons, not all metrics will be made available in this thesis. However these could be made available on a case to case basis, please contact the author if you are interested in the data.

6.3 Lessons learned

By applying the RE4SA model in the case studies and performing the evaluation, a number of findings could be generalised to indicate some of the lessons that were learned. These lessons were collected in the following table, and should be useful to anyone who intends to apply the RE4SA model, and to guide future research. The lessons have been grouped based on the RE4SA concepts they refer to.

Lessons learned	
User stories & Features	
1	*User stories leading, feature diagram suggestive:* It needs to be stressed in communication with developers that the user stories are the main focus and that the feature diagram is only a suggestion. Otherwise the developers will feel restricted, and you lose part of the goal focus that user stories achieve. This does mean that the feature diagram needs to be updated to reflect the actual architecture after development
2	*Tracing requirements to features:* User stories from a customer can be traced to features in the feature diagram to show that their requirement is already met (and how). This can be assisted through concept recognition, where certain terms in
3	*Placing new features:* New features could be placed in a logical location for the software by first considering the different modules that could contain the new feature, and then looking at existing features within those modules to see if any are similar to the new features. Candidate modules can be identified by looking at the concepts in the user story.
4	*Feature & user story cardinality:* Cardinality of features and user stories can differ. Sometimes a feature is described in multiple user stories, for example when different roles want the same thing. In other scenarios a combination of multiple features can fulfil a single user story.
5	*Undocumented features:* Not all features are described in user stories, this can be because some features are common sense and defining user stories for these can be redundant. For example when a user story defines a customer profile, it can be deemed unnecessary to create a user story to describe the need for a username. Alternatively, a developer might add features that are not described in a user story because he thinks it is important.
Features	
6	*Customer specific feature diagrams:* The feature model could be easily duplicated within the Eclipse modelling tool. This means that it is little effort to create customer specific feature diagrams. These customer specific environments provide a lot of value, as they efficiently document customisations.
7	*Feature diagram comprehension:*

	As feature diagrams grow in size they become harder to comprehend which limits their use in communication. This can be mitigated by using only the relevant section of the feature diagram so the information is in a comprehensible format.
	Features & modules
8	*Colour coding features or modules:* Colour coding can assist in guiding the viewer to relevant features or modules. Colour coding is especially useful to visualise aspects that deviate for an existing software product (e.g. customised or new). Colour coding can also partly solve the issue that the feature diagram becomes incomprehensible due to its size.
9	*Iterative SA refinement:* Applying the situational RE4SA method in multiple projects for a software product allows for refinement of the software architecture models. This means over time the accuracy of the model compared to the actual implementation improves.
10	*Consistency in functional SA documentation:* By creating/updating the feature diagram and functional architecture diagram in parallel the models can be made more consistent and be tested for completeness. This includes ensuring elements have the same names in different models, and are at the same level of depth.
11	*Update intended to actual architecture:* The feature diagram should be updated after development to properly reflect the actual architecture as opposed to the intended architecture.
12	*Drill-down navigation:* In order to properly support the information seeking mantra through drill-down navigation, it would be needed to link the different functional architecture views to each other. However, without tool support this would take a huge amount of manual work.
13	*Zoom out navigation:* While the drill-down navigation might be the most common use case for functional experts, developers might benefit more from being able to zoom-out starting at the feature level. This would allow them to obtain an overview of the functionalities instead of only working on the low-level details.
14	*Version update:* Added features and modules can be modelled in the FAD and feature diagrams. By colour coding the new elements, it can easily be seen which additions were made in the update. This can be used to create the release notes, and to identify in which update a feature / module has been added
	Modules & epic stories
15	*Related modules instead of epic stories in project briefs:* When there are no epic stories available for already created modules, these module names can be used in the relevant epic stories section of the project brief. This prevents the need for requirement recovery in order to use the method.
	User stories, epic stories, features & modules
16	*Identification of new features / modules in SA creation:* During the creation of the software architecture new features and modules can be identified, by using the link between software architecture and requirements, requirements can be recovered based on the architecture.
17	*Barista problem:* Using the proper level of granularity for US and ES remains a difficult task (Blessinga, 2018). This problem was extended to a SA level (what is a composite feature or a sub-module). However, since each of the figures is a different view of the same information,

	it should not be an issue if certain aspects are contained in multiple views. These views can be added when they are needed.
18	*Minimal yet thorough documentation:* Applying the situational method led to more detailed documentation, while still minimizing the amount of text necessary. By utilizing the requirements for documentation purposes we can keep track of the "who, what and why" of a solution. While the SA keeps track of the "how".
19	*Facilitate communication between functional experts and developers:* As a result of lessons 12 & 13, we can support a shared context knowledge between developers and functional experts. As developers can zoom out from their normal focus on the feature level, and functional experts can zoom in from their normal focus on module level.
20	*Customisation risk assessment:* By utilizing the software architecture models, the risk of a requested customisation can be assessed. This can be done by analysing the information requirements from the modules, and the hierarchy of the suggested customisation in the feature diagram.

Table 8 – Lessons learned from the case studies

Chapter 7

Discussion & Conclusions

In this final chapter, conclusions are drawn from the research, the research questions are answered based on the findings. Furthermore, we will address validity threats and discuss interesting topics for future research.

In this research, a method based on the RE4SA model was applied in a total of four enterprise system customisation case studies. For each of these cases we could successfully apply the case study method, and received positive expert feedback. The method resulted in a more detailed documentation of the requirements, increased goal focus and provides both a high level and detailed overview. One of the benefits of a method based on the RE4SA model, is that it allows for the use of software architecture models in practise. Over time it also allows us to achieve alignment between software product management and software architecture, as the software architecture is based on the requirements. Furthermore we have found some initial proof that the RE4SA concepts could facilitate communication with customers.

7.1 Research questions

Before we can provide an answer to the main research question "How can the RE4SA model be applied for customisation of enterprise application software, and what are its benefits", we should first consider the sub questions that were formulated in chapter 2.1.

> *SQ1: "How can RE4SA be applied for customisation of enterprise application software?"*

Customisation differentiates itself from other software application development, in that we already have a "standard" version of the software. This "standard" version is then changed to meet customer specific needs, this means that there are multiple versions of the same software product. In chapter 4.3 we defined the situational method that would be used for the case studies. In this method we introduced the RE4SA concepts in the functional design phase of a customisation, and in the development of a customisation. While this does have the prerequisite that a software architecture of the "standard" version is available, there are not many different requirements to integrate the RE4SA model into a method. Agile development is preferable, but not required to work with the method. Furthermore, employees might need to be trained in the use of user stories, epic stories and functional software architecture if they are not familiar with these concepts.

Once the SA template of the "standard" version is available, this can be extended to create a clear view of a customer specific environment. In order to visualise this, we applied colour coding to allow for quick identification of non-standard features or modules. Colour coding seemed to work for the purposes of this research, and help with placing attention on the relevant aspects. However, we do acknowledge that other visualisation techniques might also be effective and hope to do more research on this. Applying the RE4SA model to customisations will result in a model of each customer specific implementation, allow for detection of dependencies of customisations, and save developers time in scanning code to detect customisations.

SQ2: "How can RE4SA be applied for release updates of (customised) enterprise application software?"

We investigated this sub question in the third case study, in 5.5. A release update can be done using the same method defined for customisations, only instead of changing a customer's environment, the "standard" version is changed to include new features / modules. Furthermore, the SA models of the new version can be utilised to show an overview of new features / modules and be utilized for the creation of release notes.

It can also be combined with the customer specific models discussed before, in this way we can detect dependencies for the new features and investigate if these have been customised at a customer. If a dependency of the new feature / module is customised this is likely to cause issues and the risks should be assessed before upgrading the software.

SQ3: "How can ontology tooling assist release upgrades and customization in enterprise application software?"

As discussed in chapter 3.5.4, due to software limitations the ontology tooling section of this research was left out of scope aside from the literature research. We do see promise in generating ontologies for a project as a way to support application of the RE4SA model. In the earlier phases of the research, experts were enthusiastic about the demos of the tools described in 3.5.

SQ4: "How can RE4SA be applied to improve internal (developers) and external (customers) communication and documentation in enterprise application software?"

While this research is somewhat limited in scope to answer this question, as all of the case studies were performed based on the case company. This means that our only comparison for this sub question is the current documentation at Forza. Besides the comparison we did obtain expert opinions on the communication and documentation through the expert interviews detailed in 6.1.

All of the experts thought that the case study documentation would be an improvement over current documentation standards. The RE artifacts of the case study are more detailed and contain more context information than the current documentation standards. The inclusion of SA artifacts supports the high level overview of the application, and zoom to low level details of the application. This should also improve internal communication as developers can use the models to zoom out to the high level view, and functional experts to zoom in on the detailed level, providing a common knowledge.

For customer communication we obtained differing viewpoints, while some experts said they would definitely use the artifacts in customer communication, others were more reserved. In this research we focused on the practitioners' viewpoint, for future research it would be important to investigate the customer viewpoint.

SQ5: "How can method fragments of RE4SA be integrated into the process used in the case company?"

In the creation of the situational method (chapter 4.3), we changed the current method to include the creation of the RE4SA concepts. The biggest changes were in the creation of a functional design, as the creation of the RE4SA concepts falls within this activity. Besides the functional design creation, we introduced the artifacts as inputs for the development, and added an activity to change the intended architecture to match the actual architecture after development. In general, we tried to make few changes to the current method while introducing the RE4SA method fragments.

> *SQ6: "How effective is the situational method for the customisation of enterprise software"*

While the case studies are in a proof of concept stage in this research, all experts reacted positively to the outputs (6.1). Furthermore, we identified lessons learned that can help use the model in the enterprise software context (6.3) The results of the research were often discussed with management at the case company, and they see a lot of promise to the suggested method. In the future the researcher will be working with the case company to further improve and integrate the situational method in the company culture.

> *RQ: "How can the RE4SA model be applied for customisation of enterprise application software, and what are its benefits".*

Finally, after discussing and answering the sub-questions, an answer can be provided for the main research question. Applying the RE4SA model can be done by introducing the concepts in a company, and including creation of the RE4SA artefacts in the functional design phase. These can then be used as an input for development. Benefits and use cases of applying the RE4SA model for customisation of enterprise software are listed in the lessons learned table in chapter 6.3, and include the following aspects:

- Provide a high-level overview of the software and requirements, a detailed low-level overview of the software and requirements and allow to shift between these views.
- By providing both a high- and low-level overview facilitates communication between functional experts, and developers.
- Results in a detailed documentation of both the requirements and the solution to the requirements
- Applying the RE4SA model allows keeping up-to-date software architecture.
- Can be used to keep track customisations in a specific software environment.
- Allows for risk assessment of requested customisations.
- Can be used to keep track of new functionalities added in a version update.
- Achieve alignment between requirements and software architecture.
- Results in efficient yet detailed documentation well suited for agile development.

7.2 Validity threats

In order to minimalize the validity threats of this research we set up a case study protocol (Appendix E) before starting work on the case studies. This protocol was based on the guidelines by Runeson and Höst (2009). A total of four case studies were performed to ensure data triangulation. Furthermore, this research was built on previous findings on the RE4SA model and its concepts (Blessinga, 2018; Brinkkemper et al. 2019; Lucassen et al, 2018; van de Keuken, 2017).

However, due to the nature of case studies there are a number of validity threats that should be addressed to inform the readers about aspects that should be taken into account for the generalisability of this research. Firstly, as mentioned in the expert interviews section, the experts had limited experience with user stories. This could cause validity threats as they might have a more favourable opinion because they were shown ‘something new’. This is somewhat limited as it has also been introduced from a second source aside from this research (the partnership). Furthermore, the effectiveness of user stories has been proven in previous research, and by the high adoption numbers. It is also possible that due to general human resistance to change, the reactions were less positive than if they had been familiar with US’s (but this seems slightly doubtful due to the positive reactions).

Secondly, although we used data triangulation to make the results more reliable, all of the cases were still based on a single software product, and a single company. This might have caused some confounding factors to impact the findings. No specific confounding factors were identified that might have caused the findings for each case, and the research findings have commonalities with findings in other research within the GRIMM research group.

Thirdly, the research was done in combination with an internship, and the experts knew the interviewer and that they were part of a study. This might have led to more positive reactions due to social conventions.

Finally, Forza did not have a completely structured method for their projects. While they did have a certain way of working, this was not formalised and had some slight differences between projects. This could have led to more positive interview reactions since any structured method might be an improvement.

7.3 Future research

While we did obtain proof that the RE4SA model can be applied effectively in the multi-case study, additional case studies and similar research will help provide more evidence for the findings of this research. Similar findings would also help dismiss the validity threats discussed in the previous section, and thus improve the generalisability of this research.

Additional evidence, content metrics and ontology tools

Besides additional evidence from case studies and practice reports, we also identified a number of open research directions. The content metrics could lead to interesting insights in the use of feature diagrams if a larger data set can be obtained. Eventually these metrics could possibly allow evaluation of software based on the feature diagrams.

In this research the scope was mostly focused on the design phase, in future research this could be continued by analysing development in a method that applies the RE4SA model, and comparing the process to development without the RE4SA concepts.

No-code & citizen development

At the case company a no-code platform was recently introduced for a specific set of projects, this could also be an interesting venue for further testing of the situational method, and to link the RE4SA concepts to citizen development. Due to the natural language format of user stories, and the findings on crowdsourced user stories by Menkveld, Brinkkemper & Dalpiaz (2019) this could be an effective way to support citizen development. We do however feel that the software architecture might be difficult to grasp for citizen developers, but this should be researched and not just assumed.

Outsourced development utilizing feature diagrams

In the interviews, areas for future research were also identified. One area would be to evaluate the use of feature diagrams for detailing outsourced software development. We hypothesised that enforcing a feature diagram could ensure that the outsourced development matches the expectations. By internalising the creation of the software architecture based on the requirements and then supplying this as a design outline to an outsourced company could improve these results. The effects of making use of software architecture in practise could also be further investigated.

Software tools

We also identified the possibility to support the RE4SA model through software tools. Due to limitations the ontology tools were left out of scope for this research, but we still believe the approach to be promising and would like to perform more focused research on this topic. Additionally, it might be possible to automatically generate the architecture models based on the requirements or other artefacts through natural language processing. This would greatly reduce the amount of manual work required for architecture recovery which could increase the adoption of a method based on the RE4SA model. If the RE4SA concepts can be linked to one another in a tool, it would be possible to fully utilize the traceability between the RE and SA concepts.

Automatic reporting

Finally, recent research on automation of medical reporting through speech and action recognition has been performed (Maas et al., 2019). In this Care2Report research, the researcher were able to generate documentation of medical consultations, detecting symptoms, doctor evaluations and treatment plans. For future research, we would like to apply their research to the RE4SA context. If this is successful it is possible to automatically generate requirements documentation from an interview with a customer.

References:

Aladwani, A. (2001), "Change management strategies for successful ERP implementation", Business Process Management Journal, Vol. 7 No. 3, pp. 266-275

Bass, L., Clements, P., & Kazman, R. (2013). Software architecture in practice (3rd ed). Upper Saddle River, NJ: Addison-Wesley.

Beck, K. (1999). Embracing change with extreme programming. Computer, 32(10), 70–77. https://doi.org/10.1109/2.796139

Beheshti, H. M. (2006). What managers should know about ERP/ERP II. Management Research News, 29(4), 184-193.

Bik, N., Lucassen, G., & Brinkkemper, S. (2017). A Reference Method for User Story Requirements in Agile Systems Development. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 292–298). Lisbon, Portugal: IEEE. https://doi.org/10.1109/REW.2017.83

Bingi, P., Sharma, M. K., & Godla, J. K. (1999). Critical Issues Affecting an ERP Implementation. Information Systems Management, 16(3), 7–14. https://doi.org/10.1201/1078/43197.16.3.19990601/31310.2

Blessinga, R. (2018). Designing the Automated Greenhouse. Utrecht University.

Brinkkemper, S., & Pachidi, S. (2010). Functional Architecture Modeling for the Software Product Industry. In M. A. Babar & I. Gorton (Red.), Software Architecture (Vol. 6285, pp. 198–213). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15114-9_16

Brinkkemper, S., Dalpiaz, F., & Lucassen, G. (2018 - unpublished) Draft manuscript: User Stories – Fundamentals of Software Specification

Canal, C., Murillo, J. M., & Poizat, P. (2006). Software Adaptation. L'objet, 12(1), 9–31. https://doi.org/10.3166/objet.12.1.9-31

Cao, L., & Ramesh, B. (2008). Agile Requirements Engineering Practices: An Empirical Study. IEEE Software, 25(1), 60–67. https://doi.org/10.1109/MS.2008.1

Chaffey, D. (2009). E-business and e-commerce management: strategy, implementation and practice (4th ed). Harlow, England ; New York: FT Prentice Hall.

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: a taxonomy. IEEE Software, 7(1), 13–17. https://doi.org/10.1109/52.43044

Christensen, C. M., Cook, S., & Hall, T. (2005). Marketing Malpractice. Harvard business review, 83(12), 74–83.

Christensen, C. M., Hall, T., Dillon, K., & Duncan, D. S. (2016). Know Your Customers' "Jobs to Be Done", 10.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: views and beyond. In 25th International Conference on Software Engineering, 2003. Proceedings. (pp. 740–741). Portland, OR, USA: IEEE. https://doi.org/10.1109/ICSE.2003.1201264

Cohn, M. (2004). User Stories Applied - For Agile Software Development. Addison Wesley Longman Publishing Co.

Cziharz, T., Hruschka, P., Queins, S., & Weyer, T. (2005). Handbook of Requirements Modeling According to the IREB Standard, 115.

Dalpiaz, F., & Brinkkemper, S. (2018). Agile Requirements Engineering with User Stories. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 506–507). Banff, AB: IEEE. https://doi.org/10.1109/RE.2018.00075

Dalpiaz, F., van der Schalk, I., & Lucassen, G. (2018). Pinpointing Ambiguity and Incompleteness in Requirements Engineering via Information Visualization and NLP. In Requirements Engineering: Foundation for Software Quality (Vol. 10753, pp. 119–135). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-77243-1_8

Daneva, M. (2004). ERP requirements engineering practice: lessons learned. IEEE Software, 21(2), 26–33. https://doi.org/10.1109/MS.2004.1270758

Dolmetsch R., Huber, T., Fleisch E. & Österle, H. (1998). 'Accelerated SAP: 4 case studies', Institute for Information Management, University of St. Gallen, Switzerland

Ehie, I. C., & Madsen, M. (2005). Identifying critical issues in enterprise resource planning (ERP) implementation. Computers in Industry, 56(6), 545-557.

Financial Times (2003) Buried treasure. Article by Simon London. Financial Times, 10 December. (As referred by Chaffey, 2009)

Gayer, S., Herrmann, A., Keuler, T., Riebisch, M., & Antonino, P. O. (2016). Lightweight Traceability for the Agile Architect. Computer, 49(5), 64–71. https://doi.org/10.1109/MC.2016.150

Gupta, A. (2000). Enterprise resource planning: the emerging organizational value systems. Industrial Management & Data Systems, 100(3), 114–118. https://doi.org/10.1108/02635570010286131

Helo, P., Anussornnitisarn, P., & Phusavat, K. (2008). Expectation and reality in ERP implementation: Consultant and solution provider perspective. Industrial Management & Data Systems, 108(8), 1045-1059

Hevner, March, Park, & Ram. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75. https://doi.org/10.2307/25148625

Hohmann, L. (2003). *Beyond software architecture: creating and sustaining winning solutions*. Addison-Wesley Longman Publishing Co., Inc..

Jansen, N., van Rhijn, J. (2016). Utrecht Architecture Description Language. (unpublished?)

Jenkins, A. M., Naumann, J. D., & Wetherbe, J. C. (1984). Empirical investigation of systems development practices and results. Information & Management, 7(2), 73–82. https://doi.org/10.1016/0378-7206(84)90012-0

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study: Fort Belvoir, VA: Defense Technical Information Center. https://doi.org/10.21236/ADA235785

Kassab, M. (2015). The changing landscape of requirements engineering practices over the past decade. In 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE) (pp. 1–8). Ottawa, ON, Canada: IEEE.

Klement, A. (2016). When coffee and kale compete. CreateSpace Independent Publishing Platform

Laudon, K. C., & Laudon, J. P. (2017). Management information systems: managing the digital firm. (15th ed). Pearson.

Light, B. (2001). The maintenance implications of the customization of ERP software. Journal of software maintenance and evolution: research and practice, 13(6), 415-429.

Light, B. A., & Papazafeiropoulou, A. (2004). Reasons behind ERP package adoption: a diffusion of innovations perspective. 12th European Conference on Information Systems, Turku, Finland.

Light, B. (2005). Potential pitfalls in packaged software adoption. *Communications of the ACM*, *48*(5), 119-121.

Lucassen, G., van de Keuken, M., Dalpiaz, F., Brinkkemper, S., Sloof, G. W., & Schlingmann, J. (2018, March). Jobs-to-be-Done Oriented Requirements Engineering: A Method for Defining Job Stories. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 227-243). Springer, Cham.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2017a). Improving user story practice with the Grimm Method: A multiple case study in the software industry. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 235-252). Springer, Cham.

Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2017b). Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering, 22(3), 339–358. https://doi.org/10.1007/s00766-017-0270-1

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2016a). Improving agile requirements: the quality user story framework and tool. Requirements Engineering, 21(3), 383-403.

Lucassen, G., Dalpiaz, F., Werf, J. M. E. M. van der, & Brinkkemper, S. (2016b). The Use and Effectiveness of User Stories in Practice. In M. Daneva & O. Pastor (Red.), Requirements Engineering: Foundation for Software Quality (Vol. 9619, pp. 205–222).

Lucassen, G., Dalpiaz, F., Werf, J. M. van der, & Brinkkemper, S. (2015). Bridging the Twin Peaks -- The Case of the Software Industry. 2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture, 24–28. https://doi.org/10.1109/TwinPeaks.2015.13

Maas, L., Geurtsen, M., Nouwt, F., Schouten, S., van de Water, R., van Dulmen, S., olde Hartman, T., Noordman, J., Dalpiaz, F., van Deemter, K., Brinkkemper, S., (2019). The Care2Report System: Automated Medical Reporting as an Integrated Solution to Reduce Administrative Burden in Healthcare. Unpublished paper submission.

Molenaar, S., Brinkkemper, S., Menkveld, A., Smudde, T., Blessinga, R., Dalpiaz, F., (2019). On the Nature of Links between Requirements and Architectures: Case Studies on User Story Utilization in Agile Development. (currently) Unpublished RE4SA submission for IEEE: RE next 2019

Moon, Y. B. (2007). Enterprise Resource Planning (ERP): a review of the literature. International Journal of Management and Enterprise Development, 4(3), 235. https://doi.org/10.1504/IJMED.2007.012679

Muscatello, J.R. & Parente, D.H. (2006), "Enterprise resource planning (ERP): a postimplementation cross-case analysis", Information Resources Management Journal, Vol. 19 No. 3,pp.61-80.

Nah, F. F., Lau, J. L., & Kuang, J. (2001). Critical factors for successful implementation of enterprise systems. Business Process Management Journal, 7(3), 285-296

Ng, C. S.-P. (2012). A Case on ERP Custom Add-On in Taiwan: Implications to System Fit, Acceptance and Maintenance Costs. International Journal of Enterprise Information Systems, 8(4), 44–62. https://doi.org/10.4018/jeis.2012100102

Nuseibeh, B. (2001). Weaving together requirements and architectures. Computer, 34(3), 115–119. https://doi.org/10.1109/2.910904

Österle, H., Feisch, E. & Alt, R. (2000), Business Networking:Shaping Enterprise Relationships on the Internet. New York: Springer.

Parnas, D. L. (1972). On the Criteria To Be Used in Decomposing Systems into Modules, 15(12), 6.

Pervan G, Maimbo H (2005) Designing a case study protocol for application in IS research, The Ninth Pacific Conference on Information Systems pp 1281–1292

Rasmy, M., Tharwat, A. & Ashraf, S. (2005), "Enterprise resource planning (ERP) implementation in the Egyptian organizational context",

Riebisch, M., Streitferdt, D., & Pashov, I. (2004). Modeling Variability for Object-Oriented Product Lines. In F. Buschmann, A. P. Buchmann, & M. A. Cilia (Red.), Object-Oriented Technology. ECOOP 2003 Workshop Reader (Vol. 3013, pp. 165–178). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-25934-3_16

Rozanski, N., & Woods, E. (2012). Software systems architecture: working with stakeholders using viewpoints and perspectives second edition. Upper Saddle River, NJ: Addison-Wesley.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Seffah, A., Donyaee, M., Kline, R. B., & Padda, H. K. (2006). Usability measurement and metrics: A consolidated model. Software quality journal, 14(2), 159-178.

Seo, G. (2013). Challenges in implementing enterprise resource planning (ERP) system in large organizations: similarities and differences between corporate and university environment

Shin, I. (2006). Adoption of Enterprise Application Software and Firm Performance. Small Business Economics, 26(3), 241–256. https://doi.org/10.1007/s11187-005-0215-9

Shneiderman, B. (2003). The eyes have it: A task by data type taxonomy for information visualizations. In The Craft of Information Visualization (pp. 364-371). Morgan Kaufmann.

Slob, G.-J., Dalpiaz, F., Brinkkemper, S., & Lucassen, G. (2018). The Interactive Narrator Tool: Effective Requirements Exploration and Discussion through Visualization, 6.

Smith, A., Bieg, D., & Cabrey, T. (2014). Requirements management. PMI's Pulse of the Profession: Requirements Management — A Core Competency for Project and Program Success

Somers, T. M., & Nelson, K. (2001). The Impact of Critical Success Factors across the Stages of Enterprise Resource Planning Implementations. Th Hawaii International Conference on System Sciences, 10.

Stol, K.-J., & Fitzgerald, B. (2018). The ABC of Software Engineering Research. ACM Transactions on Software Engineering and Methodology, 27(3), 1–51. https://doi.org/10.1145/3241743

Traynor, D. (2016). Intercom on Jobs-to-be-done. Book retrieved from interface.com

Umble, E. & Umble, M. (2002), "Avoiding ERP implementation failure", Industrial Management, Vol. 44 No. 1, pp. 25-34.

van de Keuken, M. (2017). Using Job Stories and Jobs-to-be-Done in software requirements engineering.

van de Weerd, I., & Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In Handbook of research on modern systems analysis and design technologies and applications (pp. 35–54). IGI Global.

Wake, B. (2003). INVEST in Good Stories: The Series, 41.

Wieringa, R., & Moralı, A. (2012). Technical Action Research as a Validation Method in Information Systems Design Science. In K. Peffers, M. Rothenberger, & B. Kuechler (Red.), Design Science Research in Information Systems. Advances in Theory and Practice (Vol. 7286, pp. 220–238). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29863-9_17

Wieringa, R. J. (2014). Design Science Methodology for Information Systems and Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43839-8

Wohlin, C. (2012). Experimentation in software engineering. New York: Springer.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14 (pp. 1–10). London, England, United Kingdom: ACM Press. https://doi.org/10.1145/2601248.2601268

Zhang, Z., Lee, M. K., Huang, P., Zhang, L., & Huang, X. (2005). A framework of ERP systems implementation success in China: An empirical study. International Journal of Production Economics, 98(1), 56-80.

Zowghi, D., & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum & C. Wohlin (Red.), Engineering and Managing Software Requirements (pp. 19–46). Berlin/Heidelberg: Springer-Verlag. https://doi.org/10.1007/3-540-28244-0_2

Grey-literature

Eclipse (2019). Eclipse modelling project. https://www.eclipse.org/modeling/

FeatureIDE (2019). FeatureIDE: An extensible framework for feature-oriented software development https://featureide.github.io/

Forza (2018). About Forza IT Group https://www.forzaconsulting.eu/en/about-us/forza-consulting/ & SCANMAN https://www.forzaconsulting.eu/en/service/solutions/scanman/

Oracle (2015). Overview of the JD Edwards EnterpriseOne Architecture https://docs.oracle.com/cd/E24902_01/doc.91/e50711/ch_ov.htm#EOIHS107

Oracle (2016). JD Edwards EnterpriseOne Product Line https://www.oracle.com/us/assets/jde-product-line-2823274.pdf

Idatalabs (2018). JD Edwards Market Share and Competitors in ERP https://idatalabs.com/tech/products/jd-edwards

Klement (2013). Replacing The User Story With The Job Story https://jtbd.info/replacing-the-user-story-with-the-job-story-af7cdee10c27

Lucassen, G. (2016c). GRIMM method brochure. https://garmlucassen.nl/

Panorama (2014) Biggest ERP Customization Challenges. https://www.panorama-consulting.com/tuesday-poll-erp-customizations/

Techopedia (2018). What is User Exit? - Definition from Techopedia. Accessed at December 5 2018, https://www.techopedia.com/definition/16457/user-exit

Techterms (2018) add-on definition, accessed at December 5, 2018 https://techterms.com/definition/addon

Wikipedia (2018) Upgrade, accessed at December 12, https://en.wikipedia.org/wiki/Upgrade

Wikipedia (2018), Implementation, accessed at January 14, https://en.wikipedia.org/wiki/Implementation

Geneca (2017) Why up to 75% of software projects will fail, accessed at December 12, 2018 https://www.geneca.com/why-up-to-75-of-software-projects-will-fail/

Atlassian (2018) Defining a project, accessed at January 14, 2019 https://confluence.atlassian.com/adminjiraserver/defining-a-project-938847066.html

Appendix

A. Paper submission based on this research

Specification of Requirements and Software Architecture for the Customisation of Enterprise Software:

A multi-case study based on the RE4SA model

Tjerk Spijkman, Sjaak Brinkkemper, Fabiano Dalpiaz, Anne-Fleur Hemmer, Richard van de Bospoort

***Abstract*— Most failed software projects can be traced to bad requirements management. Additionally there is a big gap between state of the art and practice in software architecture. For enterprise software customisation all common software development problems exist, as well as additional challenges. Instead of one standard software product, vendors often have to deal with customised versions with additional maintenance challenges. In this research we apply the Requirements Engineering for Software Architecture (RE4SA) model in a multi-case study to show how the requirements engineering and software architecture disciplines can be linked, and in doing so provide improvements to both. By applying this model in a multi-case study on enterprise software customisation, we achieve improvements in requirements management, documentation and ensuring alignment between the software architecture and requirements.**

***Index Terms*— Requirements engineering, Software architecture, Case study, Enterprise software, Software products, Customization.**

I. Introduction

Requirements engineering (RE) is one of the key processes in the creation and customization of software products, as it addresses the critical problem of designing the right software for the customer [1]. Poor requirement management can complicate software development projects, and lead to project failure. In 2014, PMI conducted a study which showed that 47% of all project failures can be linked to poor requirements management [2]. By placing appropriate focus on requirements engineering, it is possible to prevent project failure, manage project time-frames, effectively plan releases, facilitate communication, and ensure that a solution meets the stakeholder's expectations [1-3].

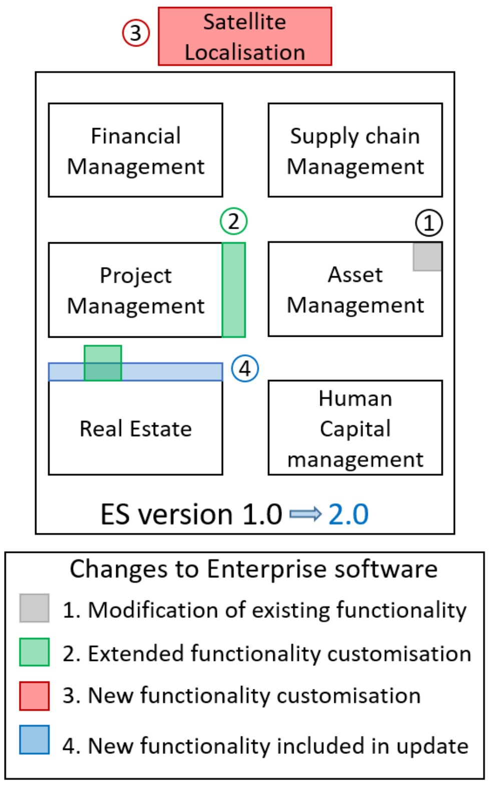
The second concept that we will focus on in this research is software architecture (SA). Bass, Clements and Kazman define software architecture as a representation of a system in which there is a mapping of functionality onto software components, and a concern for the human interaction [4]. Bass et al. also state that software architecture constitutes a common language for all stakeholders, and captures design decisions in the early stages of a software product. As software products get bigger, good architecture design is required to ensure a loose coupling within the software, and to facilitate more effective collaboration [5]. Good architecture documentation is often scarce in practice, and there is a gap between state of the art and state of the practice in software architecture [6]. This indicates an opportunity to improve the use of software architecture in practice.

While these two concepts might seem unrelated at first glance, Nuseibeh introduced the twin peaks model in 2001, where he linked the software requirements to architecture in order to improve both disciplines in a project and to more effectively suit agile development [7]. Applying the model results in progressively more detail in both the requirements and architecture, this allows for both to evolve incrementally. Lucassen et al. extend this approach and introduce the reciprocal twin peaks (RTP) model, and discuss how to achieve alignment between RE and SA in practice [8]. They also discuss how this approach is different for product software compared to tailor-made software, as they receive many requirements from different customers.

These requirements can lead to multiple different implementations of a standard enterprise software product. In 2013, Panorama consulting [9] reported that 90% of enterprise software has at least minor customisations. There are different categories of customisation. Software can be modified, by changing existing functionality; Extended by adding functionality to an existing module; or additional modules can be added to the software product. These customisations can cause issues when a new version of an extension is deployed, or when the enterprise software is updated. Fig. 1 shows an example architecture with each of the customisation categories.

One promising approach to achieve alignment between RE and SA, is the RE4SA model as seen in Fig.2 [10]. This model implies a link between specific concepts in requirements engineering and software architecture. The RE4SA model, and its sub concepts are an active field of research at Utrecht University, where both (Masters) students and professors are researching different aspects of the model. Although initial findings on the use of this model are promising, there is a lack of evidence for its effectiveness in practice. In this research we have used the RE4SA model as a basis for a situational method [11], which was tested by performing a multi-case study on four cases of a specific product software solution.

This research focuses specifically on the RE4SA model applied to enterprise software, as these have to deal with additional challenges. They often lack architectural design and documentation, receive many requirements from different customers, and have differences in customer specific implementations [8, 12].



③
Satellite Localisation
Financial Management
Supply chain Management
②
①
Project Management
Asset Management
④
Real Estate
Human Capital management
ES version 1.0 ⟹ 2.0
Changes to Enterprise software
1. Modification of existing functionality
2. Extended functionality customisation
3. New functionality customisation
4. New functionality included in update

Fig. 1. Customisation categories for a software product

There is a lack of literature on the customisation of enterprise software. With this research we attempt to fill this gap, by testing the RE4SA model and report on its use in practise. We share the results of our case studies, and in doing so propose an improvement to current processes, provide an effective way for requirements management and leverage the opportunities for applying functional software architecture in practise.

The rest of this paper will be structured as follows; in section II we will cover the RE4SA model in more detail. Section III will introduce the context of the case study. In section IV we will describe the main findings of the case studies. Section V will detail the findings from expert evaluations of the case studies. In Section VI we list the lessons learned during the case studies. Finally, in section VII we will discuss the findings, and future research directions.

II. The RE4SA Model

The Requirements Engineering for Software Architecture (RE4SA) model aims to align the RE and SA disciplines, similar to the Reciprocal Twin Peaks model [8]. It differs however in the approach, instead of linking the responsibilities of actors within the RE and SA disciplines, the RE4SA model links specific concepts in both areas. This allows us to base the software architecture directly on the requirements gathered for the product, and ensure that the requirements are met by the software. Additionally this traceability between RE and SA is expected to facilitate communication between team members, and customers.

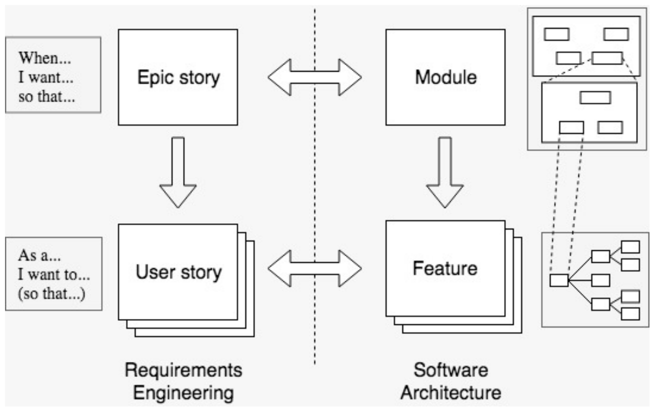


Fig. 2. The Requirements Engineering for Software Architecture (RE4SA) model [10].

A. *Explanation of the Model*

The model is especially promising because it utilizes user stories, a form of requirements that is widely used in practice [13, 14]. Due to the adoption of user stories in practice, this model allows for the use of available documentation to keep an up to date software architecture. In this model we link epic stories (ES) [15] to modules [16] in the software, and user stories (US) [17] to features [16]. ES can be used to detail the requirements that should be solved in functional modules of a software product, while US can be used to detail the requirements for more specific features within a module.

The model has a number of expected benefits; it is expected to facilitate alignment between RE and SA, communication and collaboration, help with release planning, prevent architectural drift, provide traceability from requirements to solutions and provide thorough yet minimal documentation. By utilizing the link between RE and SA, parts of either discipline can be used to improve the other. In the context of enterprise software, we can visualize where customer specific changes are located in the software, saving time when a later project (for example an update) at the same client. Having an overview of dependencies for a customisation, can provide valuable information in the risk assessment for adding a customisation. In this context we can also apply the model to plan future releases.

B. *An Example of the RE4SA Model*

In order to give a clear view on the RE4SA model, we will discuss an example of how all these concepts are linked. This example is based on the case study findings, and highlights a small section of the software product analysed for the case studies.

Consider the following epic story for an invoice automation application: *“When there is an issue with an invoice, I want a way to contact another user, so that the issue can be resolved by the relevant user.”* This requirement was solved in a “QA Form” module in the application. Each epic story contains multiple user stories, one such US for this example could be: *“As an approver, I want to set a subject for my question, so that the person who I ask the question can quickly see what it is about.”* Which can be solved through a "set subject" feature in the QA module. The RE4SA model with the concepts for this example can be seen in Fig. 3, this figure only shows a fragment of the architecture and the requirements to illustrate the concept.

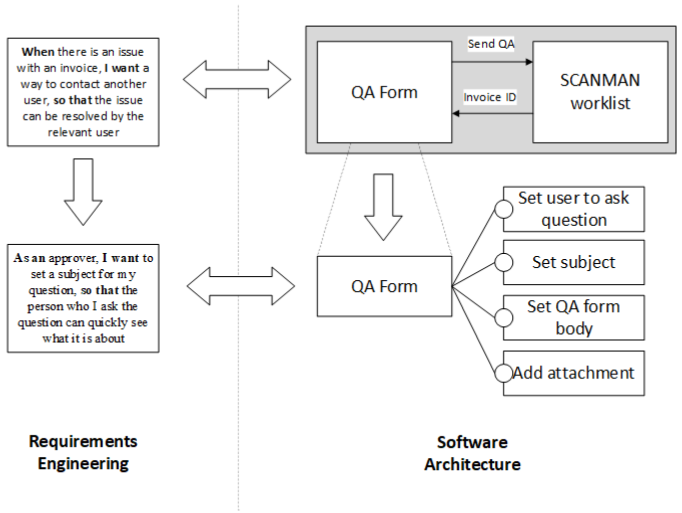

When there is an issue with an invoice, I want a way to contact another user, so that the issue can be resolved by the relevant user
As an approver, I want to set a subject for my question, so that the person who I ask the question can quickly see what it is about
QA Form
Send QA
Invoice ID
SCANMAN worklist
QA Form
Set user to ask question
Set subject
Set QA form body
Add attachment
Requirements Engineering
Software Architecture

Fig. 3. RE4SA applied to the SCANMAN QA example.

III. Case study context

For this research a case study was performed in which we analysed four different cases within a single company. The case study protocol was based on the guidelines as described by Runeson and Höst [18]. For each of the cases we applied the RE4SA model in a situational method. The results from these cases were then evaluated. In this section we discuss the goals for the case studies, and introduce the case company and the software product that is central for each of the cases.

The goal of the case studies was to identify how the RE4SA model can be applied, and to analyse the benefits of applying the model in the context of enterprise software customisation, and management. Each of the case studies describes the application of the model to a project for the SCANMAN software product. For each of the cases, we created all of the RE4SA concepts based on available documentation and evaluated this through expert interviews. By performing these case studies we test the applicability of the model for a range of cases, and use the outcomes as an input for expert evaluations of the model. Through the case studies and evaluation, we wanted to test if the RE4SA model can be applied:

- for multiple projects of a single software product;
- to effectively and efficiently document requirements and software architecture;
- to provide an overview of customer specific functionalities in an environment;
- to facilitate communication between functional and technical experts.

Case study	Description	inter-views	Roles
1	Customisation on feature level	2	Developer, C&C expert
2	Customisation on module & feature level	3	Functional consultant (x2), developer
3	Version update	3	Developer (x2), management
4	Recreation of the ES addon	3	Project manager, functional consultant, developer

Table 3 – Overview of case studies

A. *The Company*

Forza IT group is a company that is mainly focused on supporting their customers in using Oracle enterprise software. The company has offices in Soest, The Netherlands and in Sofia, Bulgaria. They provide consultancy services, perform ERP implementations, and develop extensions to the enterprise software. This means they are involved in both the RE and SA fields, as they set up the ERP environment to match the customers' requirements, and occasionally have to make changes to the software so that it matches the customers' business processes. Besides the standard Oracle software, they also develop their own add on solutions to the software to extend the functionality of the ERP.

B. *The Software*

One of Forza's most successful add-ons is SCANMAN. SCANMAN is an invoice automation application that is fully integrated within the JD Edwards (JDE) ERP system. SCANMAN uses Optical Character Recognition (OCR) to scan incoming invoices, and to automatically enter these values into JDE. These values can then be validated, and the add-on also adds an acceptance flow to JDE, in which users can accept or reject an invoice.

For SCANMAN, Forza obtains a high number of requests for functionality/features from customers. In most cases, these new functionalities/features become part of the standard version of SCANMAN, but some of the additions are customer specific. Especially these customer specific functionalities cause additional challenges. These customisations need to be retrofitted in upgrades, can cause conflicts with new version of the ERP, and cause other maintenance issues. For example, during an upgrade, it first needs to be identified what customer specific changes were made, and then the decision to recreate the customisation or leave it out needs to be made based on the circumstances. Here it would be very beneficial to have an overview of the software architecture so that identifying customizations can be made easier.

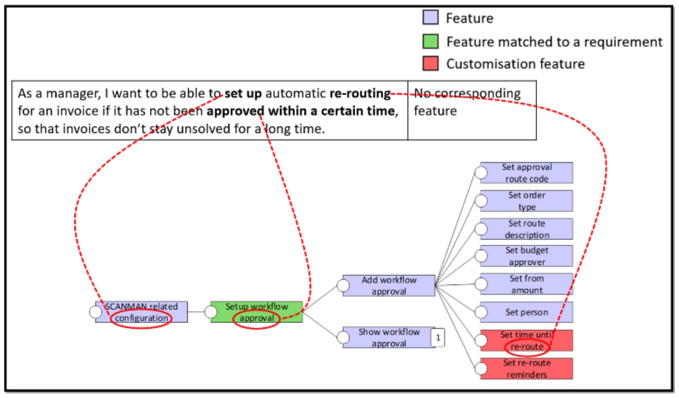
IV. Case Study Findings

In this section we will describe the context, and the main findings for each of the four cases. As a prerequisite for the cases we first had to recover the software architecture of the current version of SCANMAN, and the functionalities of JDE that it relies on. This recovery was done manually by the researchers, through analysis of the tool, documentation and user guides. This resulted in a functional architecture diagram (FAD), and feature diagram. The FAD shows the modules of the application, and how these interact [16]. The feature diagram shows the user-visible aspects of the software system, and was recovered by modelling all aspects of the GUI [19]. Since the feature diagram was recovered from the GUI, the features are likely at a more detailed level than feature diagrams created to define functionalities of a software design. The recovered architecture contained 62 (sub-) modules, and the feature model contained 1340 features. These were extended for the case studies.

A. *Case Study 1: Tracing Customer Requirements to the Software Architecture*

In this retrospective case study we applied the situational method to a requirement set compiled by a new SCANMAN customer. The set of 46 requirements were rewritten to 32 functional user stories, and 7 quality requirements. Only two of the requirements detailed functionality that was not in the current software. None of the requirements were on an ES or module level, so only the lower half of the RE4SA model was used.

For this case study we traced each of the requirements to the corresponding features and marked these green. This colour coding allowed for quick identification of the relevant features, which was a necessity due to the high number of total features. The new features were added to the feature diagram and colour coded red to signify that they were customer specific additions. The RE4SA model proved useful for deciding the appropriate location for the new features, since we could use the user stories to locate relevant features as seen in Fig. 4.


As a manager, I want to be able to **set up** automatic **re-routing** for an invoice if it has not been **approved within a certain time**, so that invoices don't stay unsolved for a long time.

No corresponding feature

Fig. 4. New feature placement, based on a user story.

In this case we found that once the "standard" feature diagram is made, this can easily be copied and extended to capture the customer specific environment. The traceability between requirements and architecture can be utilitzed to detect logical placement for new features. Since the feature diagram is rather complex and has a high number of features, this complicates the use of the model for communication. However, by collapsing irrelevant modules, or using only a small section of the model, like in Fig. 4 can mitigate this issue and makes the model useful for communication purposes. Furthermore, colour coding specific elements, like customer specific features makes it possible to easily spot important features in the model.

B. *Case Study 2: Customer specific module and features*

The second retrospective case was chosen because it had a customization that was on the ES / module level. For this case all JIRA issues concerning the specific customer were analysed and transformed to user stories. This resulted in 1 new ES, and 14 US, these numbers are lower than those of case 1 because only new functionalities were included in the JIRA tickets, and not existing ones like those matched in the first case.

The ES that was written for this case was *"When I receive an invoice that does not match an existing purchase order, I want to receive a report when it exceeds the tolerance set in the ERP, so that the invoice can be rejected."* This resulted in the user visible sub-module VMA comparison report. This shows us one limitation of the feature diagram, as it only models the features visible to the user, and not the processes "in the background" that are required to generate the report. The ES for this case contained 7 user stories, and the VMA report consists of 23 features. This difference in cardinality can have a number of possible reasons: The included information field already existed within the ERP so all relevant fields might be selected which results in a higher number of fields than required, due to limited context knowledge for the case study some customer requirements could have been missed in the US, or some features might be considered 'common knowledge' and do not need a user story to be included in the solution.

In this case we extended the colour coding to the FAD, this allows for a quick overview of which modules have been altered for the customer environment, and this information can be used to facilitate the information seeking mantra [21]. The colour coded FAD for this case, with drill-down navigation to the feature level can be seen in Fig. 5. Where red was used to signify customised modules, cyan for JDE modules that the application relies on, and orange as a warning because that JDE module is required specifically for a modification.

The colour coding allows for zooming from the FAD for the high level modules to sub module(s) to features in the feature diagram. This drill down navigation is currently only possible by manually opening views based on the naming in other views. Ideally this process would be facilitated through a tool which links all the models together, allowing for navigation by zoom on click.

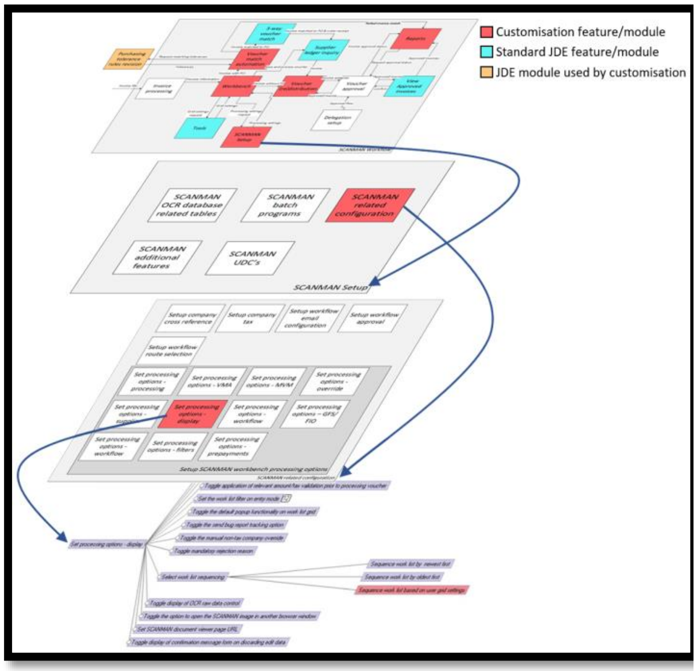

Customisation feature/module
Standard JDE feature/module
JDE module used by customisation
SCANMAN OCR database related tables
SCANMAN batch programs
SCANMAN related configuration
SCANMAN additional features
SCANMAN UDC's
SCANMAN Setup
Setup SCANMAN workbench processing options

Fig. 5. Colour coded Software architecture, with drill-down navigation capability

C. *Case Study 3: Version Update*

For the third case study we applied the situational method side by side with the current method for an ongoing project. The version update collects requirements from many different customers that are to be included in the standard version. Some of these additions are already developed in specific customer environments and need to be moved to the 'standard' software product.

For the update case study, a total of 3 ES (one of these is the QA functionality from Fig. 3.) containing 16, 2 and 6 US respectively were documented. Additionally 18 US that extend already existing modules were identified, which did not belong to an epic story due to this. For the project briefs we had a challenge to solve, as the relevant ES section couldn't be used since the ES for existing modules were never created. To mitigate this issue, we decided to apply the RE4SA model, and considered the link between ES and modules. This meant that instead of having to recover ES for existing modules, we could just list the relevant modules in the project briefs.

While we had a total of 3 ES and 40 US, the feature diagram was updated with 104 new features and the FAD was updated with one new module and three submodules. The difference in ES and module is because two modules were added for the QA epic, a QA overview & specific QA module, this indicates that we either forgot to formulate an ES, or it was added due to goal focused nature of the requirements. For the US to feature cardinality of 1:2.6 there are multiple possible explanations: like in case 2 some features were added based on 'common sense' (e.g. fields in the reports), developers might have added unrequested features, and some user stories can be solved by a collection of features. An example of this can be seen in Fig. 6, this is not inherently a bad thing, as it ties into the US principle of being goal focused [17]. This principle implies that we don't care how we get to our goal, as long as we can get there. Fig. 6 also shows an example of 'common sense' features in the sub-features of the "show QA history" feature.

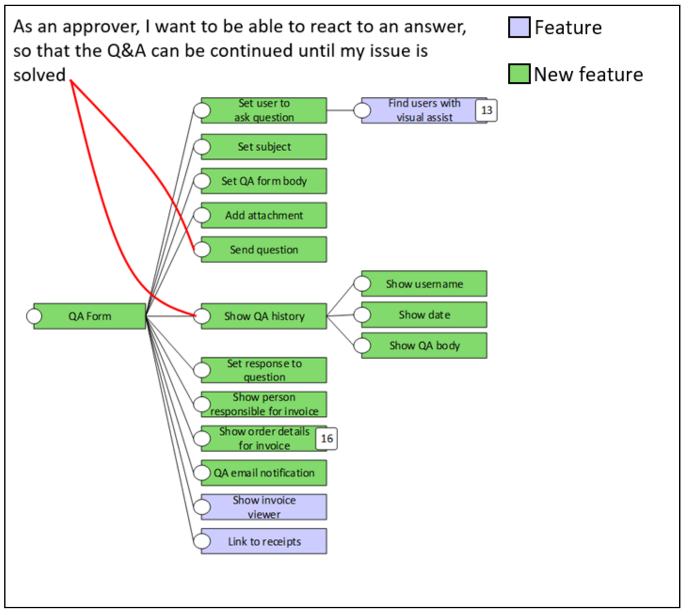


Fig. 6. A single user story which is solved by two combined features.

As the software architecture is revised for new versions and customisations, this also allows for improvements to the architecture models. While adding the new features to the feature diagram, it was noticed that some existing functionalities were not yet included in the reconstructed architecture. For example, in this update new features were added to the invoice processing module, however in the process of adding these features to the feature diagram it appeared that the invoice processing module had not been included in the feature diagram yet. Because these new features extended a module that was not mapped, this error in the feature diagram could be detected and fixed. This implies that the use of the situational RE4SA method in multiple projects also allows for refinement of the architecture model, and increases the accuracy of the models over time.

The case study report was used as a basis for the update notes of the version update. The software architecture provided a quick overview of all added functionalities, which was a lot less effort to analyse than all the logged JIRA issues. By using cut-outs of the relevant parts of the feature diagram (like Fig. 6), it becomes possible to detect the new functions within the software, and combined with a short textual explanation can provide enough information for the update notes.

D. *Case Study 4: Recreation of SCANMAN for NetSuite*

The fourth case study is somewhat different than the others, as it is not focused on the JDE version of SCANMAN, but on recreating the functionality of SCANMAN in NetSuite, another Oracle enterprise application. For this case study we used the software architecture and documentation of the JDE version to discover the core functionalities and to decide what needed to be included for the first version of SCANMAN NetSuite. This case can still be seen as an customisation, as it extends the functionality of NetSuite.

The design for a first version of the NetSuite add-on consisted of a total of 10 ES, and 70 US. This resulted in 10 modules and 69 features. The reason why we mostly have a 1:1 relation between the RE and SA concepts, is because this case study is based on the intended architecture, instead of the actual architecture (like in the other 3 cases). The reason for not using the actual architecture, is that the project is still in progress as of the time of writing. This does indicate that the RE4SA model can be applied to transform the RE sections to an intended architecture. The developer can then refine this intended architecture with the 'common sense' features, and the developer's own additions or alternative solutions to the requirements. Afterwards the intended architecture needs to be updated to reflect the actual architecture.

The initial design for the NetSuite version only has about 5% of features compared to the JDE version. This can be explained, as the initial design only consists of the must haves of the application. It also describes the intended architecture (Fig. 6.) as opposed to the actual architecture (based on GUI) which will most likely have more features. Finally, the difference could be due to the software architecture being created before the application, which would result in a more efficient solution.

Case	1	2	3	4
User stories (US)	32	14	40	70
Features	21	28	104	69
US: Features	1.5: 1	1: 2	1: 2.6	1 : 1
Epic stories (ES)	0	1	3	10
ES: US	N/A	1: 7	1: 8	1: 7
Modules	0	1	4	10
ES: Modules	N/A	1: 1	1: 1.3	1: 1
Modules: Features	N/A	1: 28	1: 26	1: 6.9

Table I – Cardinalities of the RE4SA concepts in the cases

E. *Case study metrics*

To obtain insight on the cardinality of the RE4SA metrics, we noted all the values and calculated the ratio's between the connected concepts. These can be seen in Table I, the values indicate there is not necessarily a 1:1 relation between the RE and SA concepts. There are cases where we have more US than features, and the other way around. Modules and epic stories are closer to the 1:1 relation with only one exception, however due to the lower number of ES, the sample size is also smaller which makes the results less generalizable. The ES: US ratio, can differ from the total US divided by total ES, because there are US that are linked to existing modules in the cases. The average amount of user stories per epic in the cases was 7.6; and the average number of features per module 20.3.

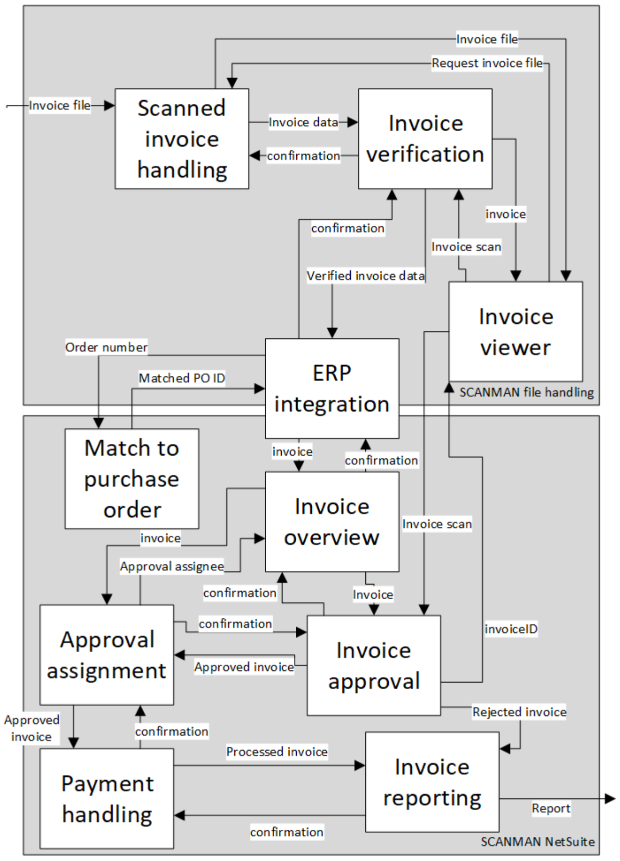


Fig. 7. FAD of the SCANMAN NetSuite intended architecture

We also analysed the complete feature diagram of the application after the update described in case 3. For this the metrics are based on the Quality in Use Integrated Measurement (QUIM) model by Seffah et al. [22] The values from the QUIM model can help determine usability aspects of GUI style applications. The results of this analysis can be seen in Table II. These metrics can be used in future research to determine usability aspects of an application based on the feature diagram.

Metric	Value
Number of modules	11
Number of submodules	55
Average atomic features per (sub) module	20,6
Average degree	9.2
Max degree	323
Min degree	1
Average standard deviation of degree	9.8
Average degree without grids	5.0
Average standard deviation of degree without grids	2.2
Average depth	2.1
Max depth	6
Min depth	1
Average standard deviation of depth	0.4

Table II – Feature diagram metrics

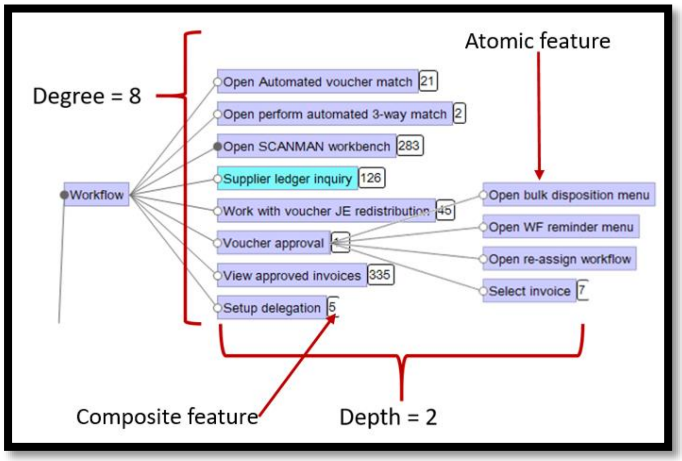

Atomic feature
Degree = 8
Open Automated voucher match 21
Open perform automated 3-way match 2
Open SCANMAN workbench 283
Supplier ledger inquiry 126
Workflow
Work with voucher JE redistribution 45
Voucher approval 4
View approved invoices 335
Setup delegation 5
Open bulk disposition menu
Open WF reminder menu
Open re-assign workflow
Select invoice 7
Composite feature
Depth = 2

Fig. 8. Visualisation of the feature diagram metrics

V. Case Study Evaluation & Lessons learned

A. *Expert interviews*

The case studies were evaluated through interviews with experts involved with the project of the case studies. These experts have different roles within the teams and company; developers, functional consultants, project managers and C&C experts were interviewed. In each of the evaluations, a semi-structured interview was used, the experts were first presented with the results of the case study, and then asked ten questions to obtain their opinion on the cases and situational method. In this section we will discuss the findings of these interviews.

All of the experts said they had limited knowledge and experience with user stories, since the company does not currently use them. And only one of the experts had worked with user stories before. However, even with this limited knowledge, opinions were very positive, all interviewees felt that the RE4SA concepts adequately captured the customers' requirements, that they provide a very detailed overview, and could effectively be applied for changes in requirements. Only one interviewee doubted if they contain enough information for a developer. The other interviewees did think the concepts gave enough information, but were unsure if it would work properly if development was completely outsourced, as this often requires stricter guidelines since outsourced workers lack context information for projects. It was mentioned by multiple interviewees that it would be dependent on the skills and experience of the developer. The interviewed developers all thought the documentation would contain enough information for development, however, one of the developers mentioned that not all requirements can be captured before starting development.

All interviewees were positive about the software architecture figures, but their expected use cases for the figures differed. Most experts said the figures gave a clear overview of the software, and thought that it would be useful for new employees to learn about the application. Other use cases that were mentioned are: as a checklist, for testing purposes, detecting dependencies with other elements.

When asked if it would be used in communication with customers, the opinions were split, some interviewees said they would use (parts of) the SA in communication to show the way the application works, ensure they are talking about the same aspects, or ensure that the features fulfil requirements. It was also mentioned that the figures could be used to visualise and communicate the risks of a requested customisation. The other interviewees thought the figures would be too complicated for use in customer communications, the feature diagram was seen as only useful in this scenario if cut-outs were used as opposed to the full diagram.

When presented with the scenario where customer specific customizations were colour coded, all experts agreed that it would be very useful to have. One of the interviewee's mentioned that every time the customers' environment was accessed, it was unknown what changes had been made in that environment. And that having a SA on a customer level (extending a template from the standard version) would save a lot of time.

While all experts thought the situational method would be an improvement over the current method, they also acknowledged some disadvantages of the method. Three of them mentioned that while the RE4SA concepts provide a very detailed overview of the application, they would quickly become useless if they are not kept up to date. Due to the size of the application, the feature diagram can be overwhelming which could limit the acceptance for using it. It would also increase time spent on documentation, frontloading more effort but (hopefully) reducing the need for rework and improving available documentation. Finally, it could be that developers feel limited due to the specific solutions mapped in the feature diagram of the intended architecture. This could be mitigated by communicating clearly that the initial feature diagram is only a suggestion.

B. *Lessons learned*

From the combination of performing and evaluating the case studies we've gained some insights in the use of the RE4SA model. The most important lessons that we've learned from performing the multi-case study are presented in table III. The lessons have been grouped based on the RE4SA concepts they refer to.

	Lesson learned
	User stories & features
1	**User stories leading, feature diagram suggestive:** It needs to be stressed in communication with developers that the user stories are the main focus and that the feature diagram is only a suggestion. Otherwise the developers will feel restricted, and you lose part of the goal focus that user stories achieve. This does mean that the feature diagram needs to be updated to reflect the actual architecture after development
2	**Tracing requirements to features:** User stories from a customer can be traced to features in the feature diagram to show that their requirement is already met (and how). This can be assisted through concept recognition, where certain terms in
3	**Placing new features:** New features could be placed in a logical location for the software by first considering the different modules that could contain the new feature, and then looking at existing features within those modules to see if any are similar to the new features. Candidate modules can be identified by looking at the concepts in the user story.
4	**Feature & user story cardinality:** Cardinality of features and user stories can differ. Sometimes a feature is described in multiple user stories, for example when different roles want the same thing. In other scenarios a combination of multiple features can fulfil a single user story.
5	**Undocumented features:** Not all features are described in user stories, this can be because some features are common sense and defining user stories for these can be redundant. For example when a user story defines a customer profile, it can be deemed unnecessary to create a user story to describe the need for a username. Alternatively, a developer might add features that are not described in a user story because he thinks it is important.
	Features
6	**Customer specific feature diagrams:** The feature model could be easily duplicated within the Eclipse modelling tool [23]. This means that it is little effort to create customer specific feature diagrams. These customer specific environments provide a lot of value, as they efficiently document customisations.
7	**Feature diagram comprehension:** As feature diagrams grow in size they become harder to comprehend which limits their use in communication. This can be mitigated by using only the relevant section of the feature diagram so the information is in a comprehensible format.
	Feature & modules
8	**Colour coding features or modules:** Colour coding can assist in guiding the viewer to relevant features or modules. Colour coding is especially useful to visualise aspects that deviate for an existing software product (e.g. customised or new). Colour coding can also partly solve the issue that the feature diagram becomes incomprehensible due to its size.
9	**Iterative SA refinement:** Applying the situational RE4SA method in multiple projects for a software product allows for refinement of the software architecture models. This means over time the accuracy of the model compared to the actual implementation improves.
10	**Consistency in functional SA documentation:** By creating/updating the feature diagram and functional architecture diagram in parallel the models can be made more consistent and be tested for completeness. This includes ensuring elements have the same names in different models, and are at the same level of depth.
11	**Update intended to actual architecture:** The feature diagram should be updated after development to properly reflect the actual architecture as opposed to the intended architecture.
12	**Drill-down navigation:** In order to properly support the information seeking mantra [23] through drill-down navigation, it would be needed to link the different functional architecture views to each other. However, without tool support this would take a huge amount of manual work.
13	**Zoom out navigation:** While the drill-down navigation might be the most common use case for functional experts, developers might benefit more from being able to zoom-out starting at the feature level. This would allow them to obtain an overview of the functionalities instead of only working on the low-level details.
14	**Version update:** Added features and modules can be modelled in the FAD and feature diagrams. By colour coding the new elements, it can easily be seen which additions were made in the update. This can be used to create the release notes, and to identify in which update a feature / module has been added
	User stories, epic stories, features & modules
15	**Identification of new features / modules in SA creation:** During the creation of the software architecture new features and modules can be identified, by using the link between software architecture and requirements, requirements can be recovered based on the architecture.
16	**Barista problem:** Using the proper level of granularity for US and ES remains a difficult task [9]. This problem was extended to a SA level (what is a composite feature or a sub-module). However, since each of the figures is a different view of the same information, it should not be an issue if certain aspects are contained in multiple views. These views can be added when they are needed.
17	**Minimal yet thorough documentation:** Applying the situational method led to more detailed documentation, while still minimizing the amount of text necessary. By utilizing the requirements for documentation purposes we can keep track of the “who, what and why” of a solution. While the SA keeps track of the “how”.
18	**Facilitate communication between functional experts and developers:** As a result of lessons 12 & 13, we can support a shared context knowledge between developers and functional experts. As developers can zoom out from their normal focus on the feature level, and functional experts can zoom in from their normal focus on module level.
19	**Customisation risk assessment:** By utilizing the software architecture models, the risk of a requested customisation can be assessed. This can be done by analysing the information requirements from the modules, and the hierarchy of the suggested customisation in the feature diagram.

Table III – Lessons learned

VI. Discussion and future research

In this research, we have carried out a multi-case study where we applied the RE4SA model to a customisation scenario of Enterprise Software. We have shown that the RE4SA model can be applied to link requirements to the software architecture, and ensure that these match one another. By updating the SA based on new requirements, we can ensure alignment between the requirements and architecture. The situational method enables the company to keep an up to date architecture of the software product in a way that minimizes required effort.

We have successfully applied the method to each of the cases and received positive expert feedback. Through this multi-case study we have provided evidence for the use of the RE4SA model in Enterprise Software customisation and management. Application of the RE4SA model improved the design process by increasing the goal focus of requirements, providing a clear overview of customer specific environments, and providing an overview of the location of features and modules in a new software version. Colour coding was applied in the first three cases and allows us to emphasise certain features or models. This was especially useful to visualize customizations, and new features. Furthermore, applying the model results in efficient yet detailed documentation, and we have found some initial proof that it could be used to improve communication with customers.

We also obtained additional proof for some of the hypotheses by Molenaar et al [10]. Case 1 shows that the RE4SA model can support placement of new functionalities in the software. The identified use cases for the SA aspects in the expert evaluation also provides evidence for the hypothesis that the model can be used to guide and support testing activities. Finally the version update case shows that the model can potentially be used for release planning.

In this research the hypothesis [10] that there would be a 1:1 relationship between USs and features seems not to hold true. This can be explained as multiple roles might require a specific feature, which would result in multiple USs per feature. Multiple features could also solve a single US as seen in Fig. 6. For the cardinality between ES and modules our findings were more consistent with the expected cardinality, as in two cases there was a 1:1 relationship and in one there was a 1:1.25 relationship.

A. *Validity threats*

We have tried to minimalize the validity threats by following the case study guidelines [18], building on previous research [10, 15, 21], and performing four case studies, which should increase the precision due to data triangulation [18]. However, as with all case studies, there are validity threats that limit the generalizability of our research. All of the cases were focused on a single enterprise software application, and focused on single company. Forza IT group provided all available documentation, and time of experts which facilitated the research, however since the research was done in combination with an internship this might lead to a bias in the results. Since the interviewees all knew the interviewer, and that they were part of a study, might have led to more positive reactions. As mentioned in the expert interviews section, the experts had limited experience with user stories. This could cause validity threats as they might have a more favourable opinion because they were shown ‘something new’. This is somewhat limited as it has also been introduced from a second source aside from this research (the partnership). Furthermore, the effectiveness of user stories has been proven in previous research, and by the high adoption numbers. It is also possible that due to general human resistance to change, the reactions were less positive than if they had been familiar with US’s. Finally, as Forza did not have a completely structured method for their projects. This could have led to more positive interview reactions since any structured method might be an improvement.

B. *Future research*

While we have obtained proof that the RE4SA can be applied effectively in this case study, it would be very beneficial to have more research and case studies available to provide additional evidence for the findings in this research. One particularly interesting aspect that we would like to perform more research on, is how effective the RE4SA concepts are in communication with customers, as the findings in this research were all based on views from the practitioners. Furthermore, our scope was mostly on the design phase, this research could be continued by applying the model with a higher focus on the requirements elicitation, and development phases.

For outsourcing development, designing the solution with the RE4SA model and enforcing the creation of features as described in the feature diagram might lead to effective results. Outsourcing development often has the issue that the developers lack context information of the application and possibly end up creating the wrong solutions. Further research on application of the RE4SA model on outsourced development could provide proof for this hypothesis.

We also identified a possibility to support the RE4SA model through software tools. By linking all concepts in a tool, the amount of manual work required to use the method can be minimized and the traceability between the RE and SA disciplines can be fully utilized. Utilizing natural language processing it might also be possible to automate creation of the SA based on a requirements set.

Finally, we would like to investigate automation of requirements reporting through speech and action recognition, similar to the research on automation of medical reporting in Care2Report [24].

Acknowledgment

We would like to thank the GRIMM research group at Utrecht University for their feedback and comments throughout the research. We would also like to thank Forza IT group for their assistance, input on the research, and access to artifacts and experts.

REFERENCES

[1] A. Aurum and C. Wohlin, "Requirements Engineering: Setting the Context", in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Red. Berlin/Heidelberg: Springer-Verlag, 2005, pp. 1–15.

[2] A. Smith, D. Bieg, and T. Cabrey, "PMI's pulse of the profession indepth report: Requirements management–a core competency for project and program success," Project Management Institute, Newtown Square, PA, 2014

[3] D. M. Fernández e.a., "Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice", Empirical Software Engineering, vol. 22, nr. 5, pp. 2298–2338, okt. 2017.

[4] P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford, "Documenting software architectures: views and beyond", in 25th International Conference on Software Engineering, 2003. Proceedings., Portland, OR, USA, 2003, pp. 740–741.

[5] N. Rozanski and E. Woods, Software systems architecture: working with stakeholders using viewpoints and perspectives. Upper Saddle River, NJ: Addison-Wesley, 2012.

[6] M. Lindvall and D. Muthig, "Bridging the Software Architecture Gap", Computer, vol. 41, nr. 6, pp. 98–101, jun. 2008.

[7] B. Nuseibeh, "Weaving together requirements and architectures", Computer, vol. 34, nr. 3, pp. 115–119, mrt. 2001.

[8] G. Lucassen, F. Dalpiaz, J. M. van der Werf, and S. Brinkkemper, "Bridging the Twin Peaks -- The Case of the Software Industry", in 2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture, Florence, Italy, 2015, pp. 24–28.

[9] Panorama, "Biggest ERP Customization Challenges." 2014 ERP report retrieved from: https://www.panorama-consulting.com/tuesday-poll-erp-customizations/

[10] S. Molenaar, S. Brinkkemper, A. Menkveld, T. Smudde, R. Blessinga, F. Dalpiaz. "On the Nature of Links between Requirements and Architectures: Case Studies on User Story Utilization in Agile Development," unpublished.

[11] I. van de Weerd and S. Brinkkemper, 'Meta-modeling for situational analysis and design methods', in Handbook of research on modern systems analysis and design technologies and applications, IGI Global, 2009, pp. 35–54.

[12] S. A. Fricker, 'Software Product Management', in Software for People, A. Maedche, A. Botzenhardt, and L. Neer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–81.

[13] M. Kassab, 'The changing landscape of requirements engineering practices over the past decade', in 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), Ottawa, ON, Canada, 2015, pp. 1–8.

[14] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, 'The Use and Effectiveness of User Stories in Practice', in Requirements Engineering: Foundation for Software Quality, vol. 9619, M. Daneva and O. Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 205–222.

[15] G. Lucassen, M. van de Keuken, F. Dalpiaz, S. Brinkkemper, G.W. Sloof and J. Schlingmann, "Jobs-to-be-Done Oriented Requirements Engineering: A Method for Defining Job Stories.", in International Working Conference on Requirements Engineering: Foundation for Software Quality: Cham: Springer International Publishing, 2018, pp. 235-252.

[16] S. Brinkkemper and S. Pachidi, 'Functional Architecture Modeling for the Software Product Industry', in Software Architecture, vol. 6285, M. A. Babar and I. Gorton, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 198–213.

[17] M. Cohn, User Stories Applied: for Agile Software Development. Redwood City, CA, USA: Addison Wesley Professional, 2004.

[18] P. Runeson and M. Höst, 'Guidelines for conducting and reporting case study research in software engineering', Empirical Software Engineering, vol. 14, no. 2, pp. 131–164, Apr. 2009.

[19] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, 'Feature-Oriented Domain Analysis (FODA) Feasibility Study':, Defense Technical Information Center, Fort Belvoir, VA, Nov. 1990.

[20] D. Traynor, 'Intercom on Jobs-to-be-done.' Book retrieved from interface.com

[21] B. Shneiderman, 'A Task by Data Type Taxonomy for Information Visualizations'. In The craft of information visualization. Morgan Kaufmann, 2003, pp. 364-371

[22] Seffah, A., Donyaee, M., Kline, R. B., & Padda, H. K. "Usability measurement and metrics: A consolidated model." Software quality journal, 14(2), 2006, pp. 159-178.

[23] Eclipse modelling tool , 2019 https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-modeling-tools

[24] Maas, L., Geurtsen, M., Nouwt, F., Schouten, S., van de Water, R., van Dulmen, S., olde Hartman, T., Noordman, J., Dalpiaz, F., van Deemter, K., Brinkkemper, S., "The Care2Report System: Automated Medical Reporting as an Integrated Solution to Reduce Administrative Burden in Healthcare. Unpublished paper submission." unpublished, 2019.

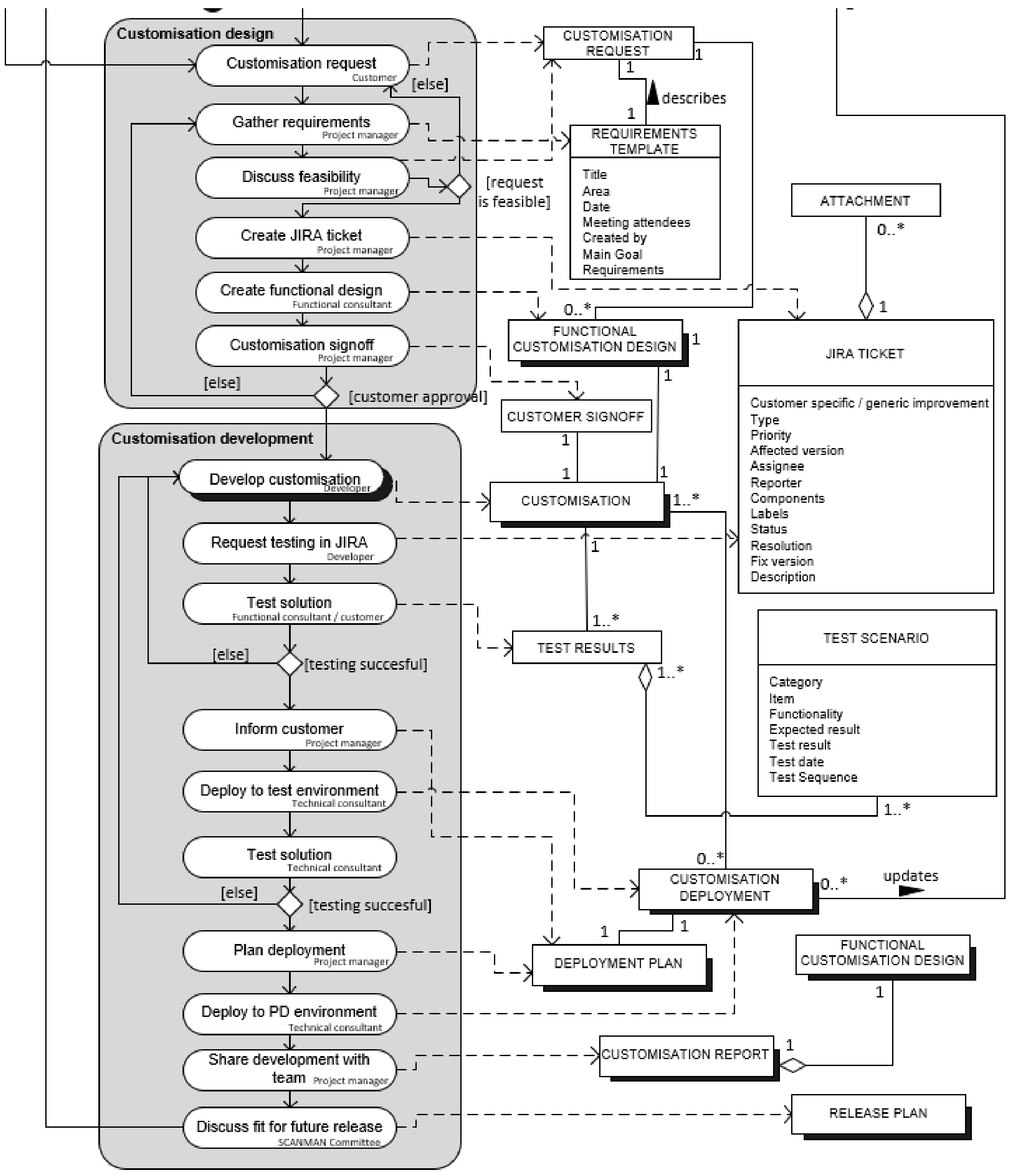
B. Glossary

Term	Definition used in this research
Module	A module is a part of the software with a well-defined functionality that can be developed independently (Parnas, 1972).
Feature	A feature is a smaller scale element in software architecture and can be defined as “A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems” (Kang, Cohen, Hess, Novak, & Peterson, 1990).
Customisation	Customisation is meant to describe changes or additions to the functionality available in the standard software (Light 2001). In this research, customisation is the collective term for modification, add-ons and other types of changes to software to fit the context of a customer.
Add-on	An add-on is a software extension that adds extra functionalities to a program. It may extend certain functions within the program, add new items to the program's interface, or give the program additional capabilities. (Techterms.com) An add-on system consists of modules of program codes that are plugged into the ERP system’s user exits. (Ng, 2012)
Modification	Changes made to existing functionalities in the software product, these are the changes Light (2001), identified as the biggest maintenance risk.
Implementation	Implementation in the information technology context refers to the post-sales process of guiding a client from purchase to use of the software or hardware that was purchased (Wikipedia, 2019). ERP implementation has a life cycle that starts with the decision to use the application and ends with the go live stage. (Moon, 2007)
Update	Release of a new version of software. For example JDE Edwards version 9.2
Upgrade	An upgrade is generally a replacement of hardware, software or firmware with a newer or better version, in order to bring the system up to date or to improve its characteristics. (Wikipedia, 2018)

Retrofit	Recreation of existing customisations when software is upgraded. This way it is ensured that no functionality is lost in an upgrade.
User exits	User exits are routines that allow for the addition of customized programs without affecting the standard program and its features. When executing a software package for a certain predefined event, a program may invoke a subroutine. If a user exit is defined, the default subroutine can be replaced with one customized by the package client to add customized functionality. (techopedia, 2018)
Enterprise application software	A type of software that is ready-made and designed to be easily implemented in existing systems. Enterprise applications typically meet one or more of the following characteristics: • They are designed to support the needs of a business, at either a departmental or larger organizational unit; • They are relatively expensive to build or license ($50,000 - $5,000,000+); • They have complex deployment and operational requirements; • While they can be operated as an independent application, they needs of the business are often best served when they are integrated with other enterprise applications (Hohmann, 2003)
Enterprise resource planning	Software providing integrated functions for major business functions such as production, distribution, sales, finance and human resources management.” (Chaffey, 2009)
User story	A description of a functionality that will be valuable to a user or purchaser (Cohn, 2004)
Epic story	An Epic is a user story that is too big to be implemented and can be deconstructed in multiple smaller user stories (Cohn, 2004)
Functional architecture diagram	A Functional Architecture Diagram (FAD) contains modules that resemble the functions of a software product. (Brinkkemper & Pachidi, 2010)

C. Current method PDD

The non-customisation sections of the PDD have been left out of this appendix, as they contain confidential information that is out of scope for this research.



Activity table of the SCANMAN implementation and customisation process.

Activity	Sub activity	Description
Support		If no other customisation is required by the customer, they move to the support process. In this they can still receive support when they need it for the time determined in the SALES AGREEMENT
Customisation design	Customisation request	If the customer requires additional functionalities, they will communicate this to the project manager.
	Gather requirements	When a customer places a request for additional customisation, the project manager will contact the customer to gather requirements
	Discuss feasibility	When the project manager has a clear overview of the customers' requirements, he will discuss this with relevant staff: - Developers are consulted to determine if the requested customisation is possible, - the SCANMAN committee is consulted to see if it matches the goals of the application - Sales department is consulted to determine if the customer needs to sign a SALES AGREEMENT for the development of the customisation. If the request is not feasible the customer will be contacted and the request will be revised
	Create JIRA ticket	The project manager will create a JIRA TICKET and assigns this to the relevant employee that needs to create the functional design for the customisation.
	Create functional design	The functional consultant creates a functional design for the customisation. Once the design is created this is discussed with the customer to see if he approves of the solution.
	Customisation signoff	The project manager discusses the designed customisation with the customer and obtains a CUSTOMER SIGNOFF.
Customisation development	Develop customisation	The developer creates the CUSTOMISATION based on the FUNCTIONAL CUSTOMISATION DESIGN.
	Request testing in JIRA	The developer assigns the JIRA ticket for the customisation to someone in quality control to have it tested.
	Test solution	The quality control employee tests the customisation to see if it meets the requirements and if the test scenarios can be completed. If the testing is unsuccessful the

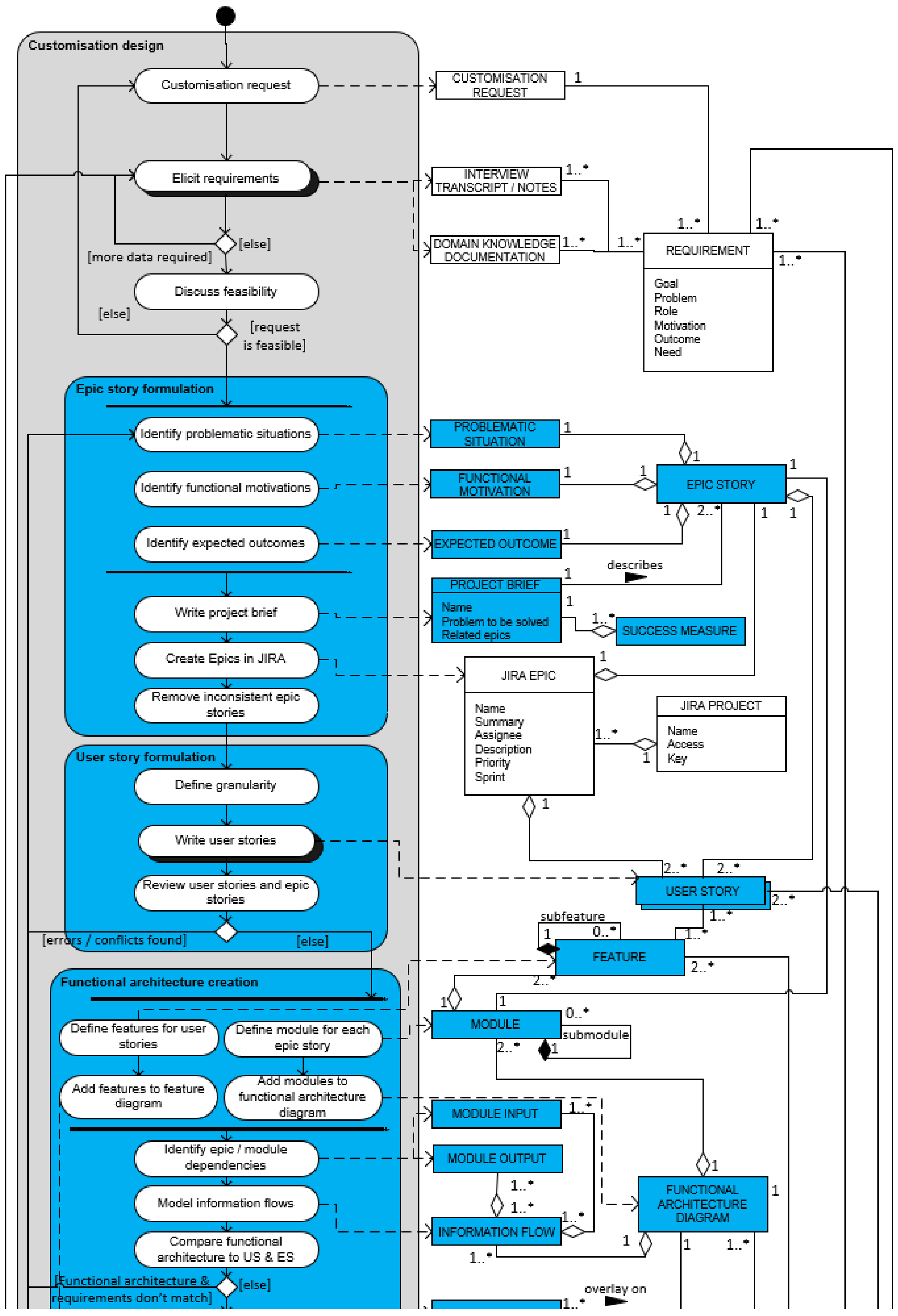
		ticket is reassigned to the developer with an overview of the test scenarios that failed, the developer can then restart the development.
	Validate customisation	The SCANMAN committee validates the customisation to see if they approve and decides if it can be implemented at the customer.
	Inform customer	The project manager informs the customer of the solution and creates a deployment plan for the customisation to the customers test environment.
	Deploy to test environment	The technical consultant deploys the customisation to the customers test environment.
	Test solution	The quality control employee repeats the tests for the customisation on the customers test environment. The testing verifies if the customisation performs as expected in the customers' environment.
	Plan deployment	The project manager contacts the customer to plan the deployment of the customisation to the production environment of the customer.
	Deploy to PD environment	The technical consultant moves the customisation to the customers production environment according to the DEPLOYMENT PLAN
	Share development with team	Once the development is completed and accepted, the project manager shares knowledge gained and the new customisation with the rest of the SCANMAN team.
	Discuss fit for future release	Finally, the SCANMAN committee discusses whether or not the customisation should become part of the standard package in a future release.

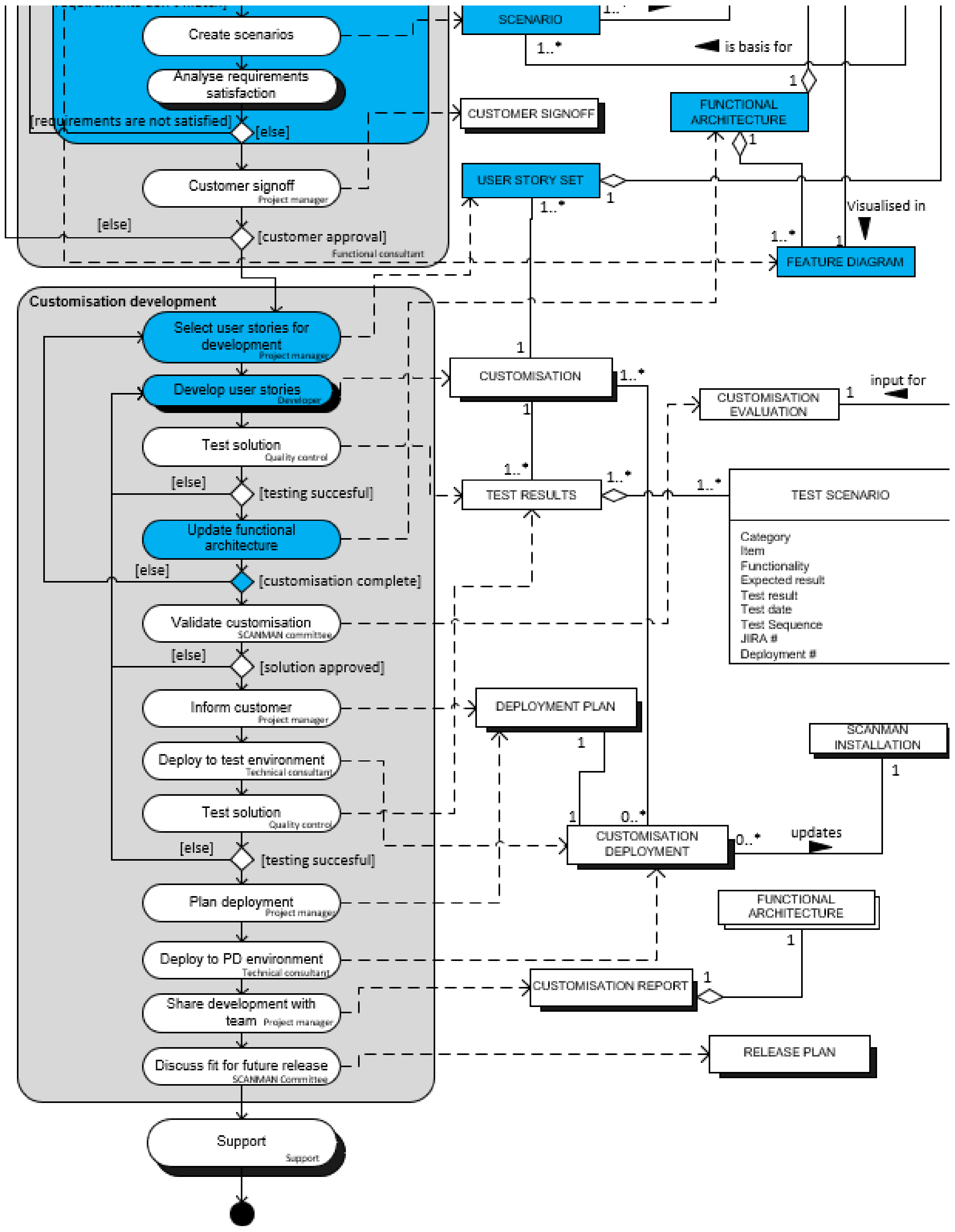
Concept table of the SCANMAN implementation and customisation process.

Concept	**Description**
TEST RESULT	A collection of the results of the TEST SCENARIOs.
TEST SCENARIO	A scenario used to test a specific functionality of the application. The scenario has specific expected results and indicates if this succeeded or failed.
CUSTOMISATION REQUEST	The customer wants a specific functionality to be included in the application, and will place a CUSTOMISATION REQUEST. This can be done through email, meetings or telephone.

REQUIREMENTS TEMPLATE	The template used to document CUSTOMISATION REQUESTs includes information on the goals of the customer and the requirements.
JIRA TICKET	A ticket in JIRA that details a certain issue that needs to be resolved. In this context, there is a JIRA for general development, and one for customer specific development. A JIRA TICKET is assigned to the staff member that needs to act on it.
ATTACHMENT	A file that is (optionally) added to a JIRA TICKET, for example a screenshot included in a bug report ticket.
FUNCTIONAL CUSTOMISATION DESIGN	The FUNCTIONAL CUSTOMISATION DESIGN for a specific customisation that can be used by developers to see what they need to develop. It is currently done in a free format depending on the staff and project.
CUSTOMER SIGNOFF	A formal agreement by the customer confirming that they approve of the solution proposed in the FUNCTIONAL CUSTOMISATION DESIGN.
CUSTOMISATION	The actual developed customisation to fulfil the CUSTOMISATION REQUEST.
CUSTOMISATION EVALUATION	In this evaluation the CUSTOMISATION is evaluated to determine if it matches with the FUNCTIONAL CUSTOMISATION DESIGN.
DEPLOYMENT PLAN	A plan for determining the process for deployment of the customisation in the customer’s environment.
CUSTOMISATION DEPLOYMENT	The actual deployment of the customisation in the customer’s environment.
CUSTOMISATION REPORT	A report that documents the functional design and customisation specific details, to be used for future reference.
RELEASE PLAN	A RELEASE PLAN that details what functionality should be included in a specific future release of the application.

D. RE4SA situational method PDD




Create scenarios
Analyse requirements satisfaction
[requirements are not satisfied]
[else]
Customer signoff
Project manager
[else]
[customer approval]
Functional consultant
SCENARIO
1..*
1..*
is basis for
1
FUNCTIONAL ARCHITECTURE
1
CUSTOMER SIGNOFF
USER STORY SET
1
1..*
Visualised in
1..*
1
FEATURE DIAGRAM
Customisation development
Select user stories for development
Project manager
Develop user stories
Developer
Test solution
Quality control
[else]
[testing succesful]
Update functional architecture
[else]
[customisation complete]
Validate customisation
SCANMAN committee
[else]
[solution approved]
Inform customer
Project manager
Deploy to test environment
Technical consultant
Test solution
Quality control
[else]
[testing succesful]
Plan deployment
Project manager
Deploy to PD environment
Technical consultant
Share development with team
Project manager
Discuss fit for future release
SCANMAN Committee
Support
Support
1
CUSTOMISATION
1..*
1
CUSTOMISATION EVALUATION
1
input for
1..*
TEST RESULTS
1..*
1..*
TEST SCENARIO
Category
Item
Functionality
Expected result
Test result
Test date
Test Sequence
JIRA #
Deployment #
DEPLOYMENT PLAN
1
SCANMAN INSTALLATION
1
1
0..*
CUSTOMISATION DEPLOYMENT
0..*
updates
FUNCTIONAL ARCHITECTURE
1
CUSTOMISATION REPORT
1
RELEASE PLAN

Activity table of the RE4SA situational method

Activity	Sub activity	Description
Customisation design	Customisation request	If the customer requires additional functionalities, they will communicate this to the project manager. In the case that a request is deemed not feasible, the project manager will communicate with the customer to either reject the request, or change the request to something that is feasible.
	Elicit requirements	Domain experts are interviewed to create the INTERVIEW TRANSCRIPT / NOTES and DOMAIN KNOWLEDGE DOCUMENTATION is gathered so that the REQUIREMENTS can be gathered from these sources.
	Discuss feasibility	When the project manager has a clear overview of the customers' requirements, he will discuss the feasibility with relevant staff: - Developers are consulted to determine if the requested customisation is possible, - the SCANMAN committee is consulted to see if it matches the goals of the application - Sales department is consulted to determine if the customer needs to sign a SALES AGREEMENT for the development of the customisation. If the request is not feasible the customer will be contacted and the request will be revised
Epic story formulation (Customisation design)	Identify problematic situations	Based on the gathered REQUIREMENTs the PROBLEMATIC SITUATIONs that need to be solved are identified and included in the first section of an EPIC STORY.
	Identify functional motivation	For each PROBLEMATIC SITUATION a functional MOTIVATION is described. This MOTIVATION describes the change that is needed to solve the problematic situation.
	Identify expected outcome	For each of the PROBLEMATIC SITUATIONs an expected OUTCOME is identified. This sketches the situation that is to be achieved.
	Write project brief	Project briefs are created for each EPIC STORY, describing the problem to be solved, the related EPIC STORIES, the SUCCES MEASURES and the scope.

	Create Epic in JIRA	For the EPIC STORIES a JIRA EPIC is created, which is assigned to the employee who should work on it.
	Remove inconsistent stories	The EPIC STORIES are compared to check for inconsistencies, like duplicate or contradictory stories. If these are detected they are removed.
User story formulation (Customisation design)	Define granularity	The minimum required granularity is set for the definition of USER STORIES, this is the level of detail in which a USER STORY can't be broken up into multiple stories. Stories that can be broken up should be formulated as an EPIC STORY instead.
	Write user stories	For each EPIC STORY, USER STORIES are written to describe the goals of the different stakeholders.
	Review user stories and epic stories	The EPIC STORIES and USER STORIES are reviewed to see if the USER STORIES capture all elements of the EPIC STORIES, and to see if they adhere to the Quality user stories framework. This step can be supported through the use of ontology models generated from the user stories.
Functional architecture diagram creation (Customisation design)	Create module for each epic story	A MODULE (or submodule) will be designed for each EPIC STORY. The MODULE has to provide a solution to the PROBLEMATIC SITUATION. The MOTIVATION component is used to see what functional name the MODULE should be given.
	Add modules to functional architecture diagram	The new modules are added to the existing functional architecture diagram at the correct level.
	Create features for user stories	The requirements in the user stories are mapped to new features that fulfil these requirements.
	Add features to feature diagram	The new features are added to the functional architecture diagram and colour coded appropriately. The resulting feature diagram shows the intended architecture.
	Identify epic dependencies	Based on the OUTCOME and PROBLEMATIC SITUATION of each MODULEs EPIC STORY, identify dependencies between MODULEs.
	Model information flows	Based on the OUTCOME and PROBLEMATIC SITUATION of each EPIC STORY, identify and connect MODULEs with INFORMATION FLOWs. This step can be supported by ontology models generated from the user stories

	Compare functional architecture to US & ES	The intended architecture that was defined, is checked against the requirements to decide if they match.
	Analyse requirements satisfaction	Considering the REQUIREMENTs, the model is analysed to see if all have been considered and satisfied. If this is not the case, another iteration of customisation design is started, improving the current version.
Customisation design	Customisation signoff	The project manager discusses the designed customisation with the customer and obtains a CUSTOMER SIGNOFF.
Customisation development	Select user stories for development	A set of USER STORIES is selected for the first development cycle. This is based on the expected effort of the USER STORIES and their priority.
	Develop user stories	The USER STORY SET is considered, and FEATURES that provide a solution for each USER STORY are developed. These solutions are collected in the CUSTOMISATION that is being developed.
	Test solution	The quality control employee tests the customisation to see if it meets the requirements and if the test scenarios can be completed. If the testing is unsuccessful the ticket is reassigned to the developer with an overview of the test scenarios that failed, the developer can then restart the development.
	Update feature diagram	When the testing is successful and the FEATUREs are considered to be high enough quality to be included, the FEATURE DIAGRAM is updated to include the new FEATURES, colour coded to easily detect which features are part of customisation instead of the standard package.
	Validate customisation	The SCANMAN committee validates the customisation to see if they approve and decides if it can be implemented at the customer.
	Inform customer	The project manager informs the customer of the solution and creates a deployment plan for the customisation to the customers test environment.
	Deploy to test environment	The technical consultant deploys the customisation to the customers test environment.
	Test solution	The quality control employee repeats the tests for the customisation on the customers test environment. The testing

		verifies if the customisation performs as expected in the customers' environment.
	Plan deployment	The project manager contacts the customer to plan the deployment of the customisation to the production environment of the customer.
	Deploy to PD environment	The technical consultant moves the customisation to the customers production environment according to the DEPLOYMENT PLAN
	Share development with team	Once the development is completed and accepted, the project manager shares knowledge gained and the new customisation with the rest of the SCANMAN team.
	Discuss fit for future release	Finally, the SCANMAN committee discusses whether or not the customisation should become part of the standard package in a future release.
Support		If no other customisation is required by the customer, they move to the support process. In this they can still receive support when they need it for the time determined in the sales agreement.

Concept table of the RE4SA situational method

Concept	**Definition**
CUSTOMISATION REQUEST	The customer wants a specific functionality to be included in the application, and will place a CUSTOMISATION REQUEST. This can be done through email, meetings or telephone.
INTERVIEW TRANSCRIPT / NOTES	A recording or set of notes from which the requirements can be extracted
DOMAIN KNOWLEDGE DOCUMENTATION	The documents that contain context information about the customer's processes and practices from which requirements can be extracted. These documents can be in various forms, like manuals, webpages, project documents etc.
REQUIREMENT	A requirement is a functional description of a new functionality that is desired by a stakeholder.
PROBLEMATIC SITUATION	The first part of an EPIC STORY that provides the contextual starting point to understand why the actor has a motivation and goal. It should describe a situation which confronts the actor with a concrete problem. (Lucassen et al., 2018; Klement, 2013)

FUNCTIONAL MOTIVATION	The second part of an EPIC STORY provides the central element of motivation. It should capture the change that needs to occur to reach the expected outcome, the FUNCTIONAL MOTIVATION generally already implies a solution to alleviate the problematic situation. (Lucassen et al., 2018)
EXPECTED OUTCOME	The EXPECTED OUTCOME sketches the desired to-be situation that the actor wants to achieve by satisfying the motivation. The EXPECTED OUTCOME conveys additional context needed to satisfy the motivation in the right way. (Lucassen et al., 2018)
EPIC STORY	An EPIC STORY a formulated requirement on a high level than a USER STORY consisting of a PROBLEMATIC SITUATION, a FUNCTIONAL MOTIVATION, and an EXPECTED OUTCOME
PROJECT BRIEF	A PROJECT BRIEF is a one-page document that describes the problem to be solved, scope, relevant EPIC STORIES and the SUCCES MEASUREs.
SUCCESS MEASURE	A criteria that needs to be met in order to consider the problem as solved.
JIRA EPIC	An entry in a JIRA PROJECT, which describes an EPIC STORY. It contains a summary, assignee, description, priority and a sprint.
JIRA PROJECT	A project in JIRA that collects issues. A project can be used to coordinate the development of a product, and track its progress.
USER STORY	A description of a functionality that will be valuable to a user or purchaser (Cohn, 2004). A USER STORY is a formulated requirement that consists of a role, action, and benefit.
FEATURE	A prominent or distinctive user-visible aspect, quality, or characteristic of a software system. (Kang et. al, 1990)
MODULE	A part of the software with a well-defined functionality that can be developed independently (Parnas, 1972).
MODULE INPUT	The MODULE INPUT is a visualisation of the incoming information flow for a MODULE in the FUNCTIONAL ARCHITECTURE DIAGRAM
MODULE OUTPUT	A visualisation of the outgoing information flow for a MODULE in the FUNCTIONAL ARCHITECTURE DIAGRAM
INFORMATION FLOW	The relation between two modules, containing both MODULE INPUTs and MODULE OUTPUTs.
FUNCTIONAL ARCHITECTURE DIAGRAM	A FUNCTIONAL ARCHITECTURE DIAGRAM

	(FAD) contains MODULEs that resemble the functions of a software product. (Brinkkemper & Pachidi, 2010)
FUNCTIONAL ARCHITECTURE	The FUNCTIONAL ARCHITECTURE represents at a high level the software product's major functions from a usage perspective, and specifies the interactions of functions, internally between each other and externally with other products. The FUNCTIONAL ARCHITECTURE consists of at least a FUNCTIONAL ARCHITECUTRE DIAGRAM and a FEATURE DIAGRAM. It can also contain different documentation describing functions, or other models.
FEATURE DIAGRAM	Features can be modelled with a FEATURE DIAGRAM, this model is a hierarchically arranged set of features (Riebisch, Streitferdt, & Pashov, 2004), and displays if features are mandatory or optional. This model gives an overview of all features that need to be included in a module.
CUSTOMER SIGNOFF	A formal agreement by the customer confirming that they approve of the solution proposed in the FUNCTIONAL CUSTOMISATION DESIGN.
CUSTOMISATION	The actual developed customisation to fulfil the CUSTOMISATION REQUEST.
CUSTOMISATION EVALUATION	In this evaluation the CUSTOMISATION is evaluated to determine if it matches with the FUNCTIONAL CUSTOMISATION DESIGN.
TEST RESULT	A collection of the results of the TEST SCENARIOs.
TEST SCENARIO	A scenario used to test a specific functionality of the application. The scenario has specific expected results and indicates if this succeeded or failed.
DEPLOYMENT PLAN	A plan for determining the process for deployment of the customisation in the customer’s environment.
CUSTOMISATION DEPLOYMENT	The deployment of a customisation to the customer’s environment.
SCANMAN INSTALLATION	The specific version of SCANMAN that is installed in the customers’ environment configured and possibly customised for that customer.
CUSTOMISATION EVALUATION	In this evaluation the CUSTOMISATION is evaluated to determine if it fulfils the REQUIREMENTS.

CUSTOMISATION REPORT	A report that documents the functional design and customisation specific details, to be used for future reference.
RELEASE PLAN	A RELEASE PLAN that details what functionality should be included in a specific future release of the application.

E. Case study protocol

Preamble

The purpose for the case study, is threefold. Firstly, we aim to obtain some real world data for the use of the RE4SA model in practice. In order to achieve this, the situational method was designed that uses fragments of RE4SA and introduces them in the method currently used at the case company. This method will be used to perform the case studies and get data on how the outputs will function, and how they are received. Secondly, the findings from the different cases will allow for refinements to the method, as we detect flaws in the initial design. Thirdly, to obtain opinions from practitioners on the concepts of the RE4SA method. This will also give us an indication for the chance that the model will actually be implemented by a company.

The results of the case studies will be published as a part of a master thesis for the Business Informatics master at Utrecht University. Data will be anonymised so that the client organisations can't be identified. Interviews will be summarised for the findings, and these summaries will be included in the appendix of the thesis. No names will be included in this summary, except for the case company name (not names of individuals who work at the company). When people outside of the case company are interviewed, a short generalised introduction of the company will be included in the introduction of the interview (e.g. Company B is a company that specialises in software development, the interviewee is a product manager at this company). Interviews will be recorded for personal use only, data might be extrapolated from the answers. (E.g. 60% of interviewees were positive about the outcomes of the situational method)

General

In the case study, two types of cases will be investigated. The first type is a retroactive case study, which means the method will be applied to data contained in documentation of completed projects. These cases will be evaluated with employees that were involved with the project, and when possible with the customer that initiated the project. The second type is a field study, where the situational method will be applied to an ongoing project, and will be presented as an alternative documentation / design method. In the field studies, the current method and the situational method will both be performed in parallel to obtain representative information for comparison of the methods, and outputs.

Procedures

Retroactive case study: for the retroactive case study, the following steps will be completed

1. Case selection: The cases that are selected need to fulfil a couple of criteria, first they must detail cases in which customisation was performed. Second, they have to be documented in enough detail to perform a case study without having to guess the inputs (or knowledge needs to be available through employees). Additionally, recent projects will be prioritised,

because they will be the more representative for the current method than older projects, as the current method has likely been improved / changed.

2. Obtain case information: The documentation available for the case will be analysed to determine what the requirements for the customer were, and the JIRA issues for these projects will be considered. When possible employees will be asked for additional information.
3. Complete all steps of the design section of the situational method. This results in a functional design report, containing:
    - An explanation of the customisation request,
    - An description of the source of the requirements,
    - A set of epic stories,
    - A project brief for each epic story,
    - A set of user stories for each epic story,
    - A functional architecture diagram
    - A feature diagram
4. Evaluate the report with two to three people involved in the project, with different roles within the project. When possible, the customer will also be contacted for an evaluation of the design. This evaluation will be done using a semi-structured interview. A summary of the interviews will be created that contains the most important findings, this summary will be included in the appendix of the thesis.
5. Analyse the findings of the case study, this includes the findings from the evaluation, and the relation of the concepts (e.g is there a feature for each user story, how many user stories on average per epic story, etc.)

Field study: for the field study cases, the following steps will be completed

1. Case selection: The cases selected for the field study, should be ongoing projects. They will be selected by discussing the options with steering group and product manager of the case company. Cases are required to involve some development of customisations, a case with just the implementation is not interesting for the study.
2. Obtain case information: Case information will be obtained by attending meetings relevant to the ongoing project, and by discussing the project with employees that are involved in the project. If possible, the customer will also be contacted to give more information on the customisation request.
3. Complete all steps of the design section of the situational method. This results in a functional design report, containing:
    - An explanation of the customisation request,
    - An description of the source of the requirements,
    - A set of epic stories,
    - A project brief for each epic story,
    - A set of user stories for each epic story,
    - A functional architecture diagram
    - A feature diagram

   Note: For the case study it will most likely not be possible to perform development process as suggested in the situational method, as the method is still in the investigation / testing phase. Performing multiple projects using this new development method, will put the company's projects at risk as employees are not yet trained to work with user stories, sprints etc.

4. Evaluate the report with two to three people involved in the project, with different roles within the project. When possible, the customer will also be contacted for an evaluation of the design. This evaluation will be done using a semi-structured interview. A summary of the interviews will be created that contains the most important findings, this summary will be included in the appendix of the thesis.
5. Analyse the findings of the case study, this includes the findings from the evaluation, and the relation of the concepts (e.g is there a feature for each user story, how many user stories on average per epic story, etc.)
6. Verify the intended architecture created in step 3, with the actual developed solution. If the RE4SA outputs were used in the development, this can be used to test how different the intended architecture is compared to the actual architecture. If the outputs were not used, it will be used to see which of the user stories are solved by the actual solution.

Research instruments

The research instruments will be a semi-structured interview, and the outputs of the case study in which the situational method is applied.

Semi-structured interview:

The semi structured interview will contain a list of questions that should be asked in each of the interviews to evaluate the method and obtain opinions from the staff members. The interview is kept semi-structured, in order to allow for more in depth conversation and to ask more about statements made by the interviewee. This means in each of the interviews the following questions will be asked, but not necessarily in the same order, and based on the conversation additional questions might be asked.

The interview will start with a short introduction of the research and an explanation of how the gathered data will be used. The participant will then be presented with the collection of artefacts created in the case study, and an explanation of the intended use. After they have had the chance to look at the epic stories, user stories, feature diagram and functional architecture they will be asked to answer the following questions and prompted for further information when relevant (and why?):

Questions:

1. Are you familiar with user stories?
    a. Have you worked with user stories before?
2. Do you think this way of documenting requirements captures the customer's wishes?
3. Do you think these documents contain enough information to start development?
    a. (If no) What other information would be needed?
4. How would you use the software architecture figures in your job?
    a. Would you use this in communication scenarios with customers?
    b. Would it be beneficial when you have to do another customisation for a customer at a later time?

5. Do you think the link between the epic stories and modules makes sense for the creation of a functional design?
6. Do you think this way of documenting a functional design allows developers more freedom in developing solutions?
7. Do you think the project briefs are necessary, or would just the architecture models and requirement set contain enough information?
8. Do you think the suggested design and development method would be an improvement over the current process?
9. Do you think the use of the user stories, and epics would make communication with customers easier?
10. Compared to the current method, what benefits of the suggested method can you think of?
    a. And disadvantages?
11. Do you think using user stories and epic stories will improve working with changes in the requirement set?

Consent form

Thank you for your willingness to participate in this research. The goal of this interview is to get the required data to evaluate the use of the RE4SA (requirements engineering for software architecture) model. A sound recording will be made for the interview, in order to ensure that no information is missed, this recording will only be used for the purposes of this specific research. The information gained from this interview will be anonymised, summarized and included in the appendix of my thesis. Some of your answers will also be used to generate statistics (e.g. 60% of participants think the documents contain enough information to start development.). The obtained data will only be used for the purposes of this research.

Participating in this research is voluntary. All information gathered will be used confidentially. You can stop the experiment at any time and withdraw your permission given in this form. If you have any questions after the experiment feel free to contact me (tjerk.spijkman@forzaconsulting.eu)

Data analysis guidelines

Data analysis guidelines

The case study evaluation will predominantly result in qualitative information, this means that most findings will be based on opinions from interviewees. However, the architecture will also be analysed to obtain content metrics. The following metrics will be extracted:

- Number of modules
- Number of composite features
- Number of atomic features
- Average atomic features per submodule
- Average degree
- Average depth
- Ratio of epic stories to user stories (multiplicity)
- Ratio of modules to features
- Ratio of features to user stories

F. Case study reports

Case study 1: Tracing Customer Requirements to the Software Architecture

1 Customisation request:

The customisation request stemmed from the implementation of the standard version of SCANMAN JDE at a new customer. This customer had a list of requirements that needed to be met by SCANMAN based on their agreement with the sales department. This case study is a retroactive case study, which means that the requirements were already documented, and a solution was already developed. However, this solution will only be considered at the end of the case study, and will not be used to formulate the ES / US set.

2 Elicit requirements:

The customer supplied a list of requirements and already prioritised them on mandatory and preferred. This list of 46 requirements are analysed to formulate ES and US, which will then be used to check against the current implementation. These requirements will be compared to the current implementation to see if there are features / modules that already fulfil the requirements.

3 Epic story formulation:

None of the requirements were determined to be on a higher level than user story, so no epic stories were formulated.

4 Project briefs

Since there are no new epic stories formulated, no new project briefs were created.

5 User stories

The list of 46 requirements led to 7 non-functional user stories, 32 functional user stories 2 of which require new functions not included in the standard version, and 2 of the requirements were rejected. The user stories were linked to the features in the feature diagram template that was reconstructed for the case studies. The two user stories that could not be linked to existing features were marked yellow in the table

Non-functional:

1. As a manager, I want the application to support processing more than 40.000 invoices, so that I can ensure that we can process all invoices we receive.
2. As a manager, I want to retain invoices for seven years, so that I can access them at all times they are relevant.
3. As a manager, I want to ensure that the application integrates with JD Edwards 9.2, so that it matches our current business practices.
4. As a manager, I want to ensure that the application does not require specific hardware for scanning, so that we don't need to replace our current hardware.
5. As a manager, I want to ensure that the application works without being reliant on browser plugins, so that it works in all scenarios
6. As a manager, I want to ensure that the applications works on all modern browsers (Firefox, Chrome, Safari, IE, Edge), so that employees can use their preferred browser without impacting their work.
7. As a manager, I want to have a separate environment for testing / development, so that it does not impact my production environment.

Functional:

	User story	Corresponding feature
1	As a manager, I want to extract invoice data via OCR from scanned images, so I don't have to hire people to do this manually.	Scan with OCR
2	As a AP clerk, I want to extract invoice data via OCR from scanned images, so I only have to confirm the values instead of adding them manually.	Scan with OCR
3	As a manager, I want to extract invoice data from PDF formatted invoices, so I can process the most common format for invoices.	Process PDF format
4	As a manager, I want to ensure that invoices which don't match a purchase order are held in an error / hold queue, so that they can be reviewed manually.	Error detection
5	As a manager, I want to ensure that invoices that contain data errors are held in an error / hold queue, so that they can be reviewed manually.	Error detection
6	As an AP clerk, I want to see which invoices contain data errors, so that I can review them manually.	Show only entries with error status
7	As an AP clerk, I want to be able to open the invoice image in a pop-out window, so that I can view it on a second screen and efficiently multi-task.	Open invoice in new window
8	As a manager, I want to see who approved an invoice, so that I can contact the correct employee when an invoice is approved that should've been rejected.	Show approver number / name
9	As a manager, I want to see any changes made to an invoice, so that I can recover previous versions.	SCANMAN audit trail
10	As an AP clerk, I want to view the invoice from within JDE, so that I can use the image when I verify invoices in the workbench without having to switch between applications.	Show invoice in viewer
11	As an approver, I want to view the invoice from within JDE when I'm approving invoices, so that I can review the image when I suspect an error in the values.	Show invoice in viewer
12	As an approver, I want to change the invoice GL dates when I approve an invoice, so that I can choose the appropriate date for the invoice to be processed (1.13).	**No corresponding feature**
13	As an AP clerk, I want to select a GL accounts from a list when I need to insert it, so that I don't need to memorise all GL accounts.	Visual assist account selection
14	As a manager, I want to support 3-way invoice matching, so that I can leverage all the information we have for an invoice.	Automated 3-way match
15	As a manager, I want to support 2-way invoice matching, so that I can compare the values on an invoice to a purchase order.	Automated voucher match
16	As a manager, I want to support invoice handling for invoices that don't have a PO, so those invoices can be processed.	Work with voucher JE redistribution
17	As a manager, I want to set up an approval route, so that non-PO invoices are approved by multiple approvers in a set order	Setup workflow route selection

18	As an approver, I want to have an approval route, so that invoices that I approve are automatically forwarded to the next approver.	Setup workflow route selection
19	As an approver, I want to apply each line item to a GL account, so that each item can be mapped to the appropriate account	Edit line item
20	As an approver, I want to see a notification for invoices that need my approval, so that I know which invoices need my attention	Open WF reminder menu
21	As a manager, I want to set a limit on the gross amount that can be approved on an account basis, so that I can manage risks and approval privileges.	Setup workflow approval
22	As a manager, I want to set up an approval flow based on the total amount of an invoice, so that I can assure high cost invoices are given enough attention.	Setup workflow approval
23	As an approver, I want to forward an invoice to the creator of the purchase order, so that he can confirm the invoice matches his intentions.	Forward referral (JDE)
24	As a manager, I want to use approval routes that are defined in JDE, so that I don't need to recreate all these approval routes.	Setup workflow approval
25	As a manager, I want to be able to set up automatic re-routing for an invoice if it has not been approved within a certain time, so that invoices don't stay unsolved for a long time. 1.25	**No corresponding feature**
26	As an approver, I want to receive reminders for invoices that are close to automatically re-routing, so that I can prevent invoice escalation and assure I complete my tasks. 1.25	**No corresponding feature**
27	As a manager, I want to delegate approval tasks to a different employee, so that the process still functions when someone is sick or on holiday	Setup delegation
28	As an approver, I want to delegate my approval tasks to a colleague, so that I don't return from absence to be greeted by a huge backlog.	Setup delegation
29	As an approver, I want to be able to add a comment to an invoice, so that I can elaborate and document why I made a decision.	Edit remark
30	As a manager, I want to be able to track the approval process, so that I can ensure the process goes as planned.	Data browser / approved invoices
31	As a manager, I want to generate customised reports, so that I can easily obtain the information that is relevant for me.	Filter / export grid data
32	As a manager, I want to export reports in CSV format, so that I can perform data analysis using Excel	Export as csv
33	As a manager, I want to generate statistics for the invoice process, so that I can measure our performance.	Supplier ledger inquiry.

Not accepted

As a manager, I want to extract invoice data from text and/or excel formatted invoices, so I can process all formats

> **The invoice document standard is to use pdf or XML format, excel and word formats can be saved as PDF to enable processing in SCANMAN**

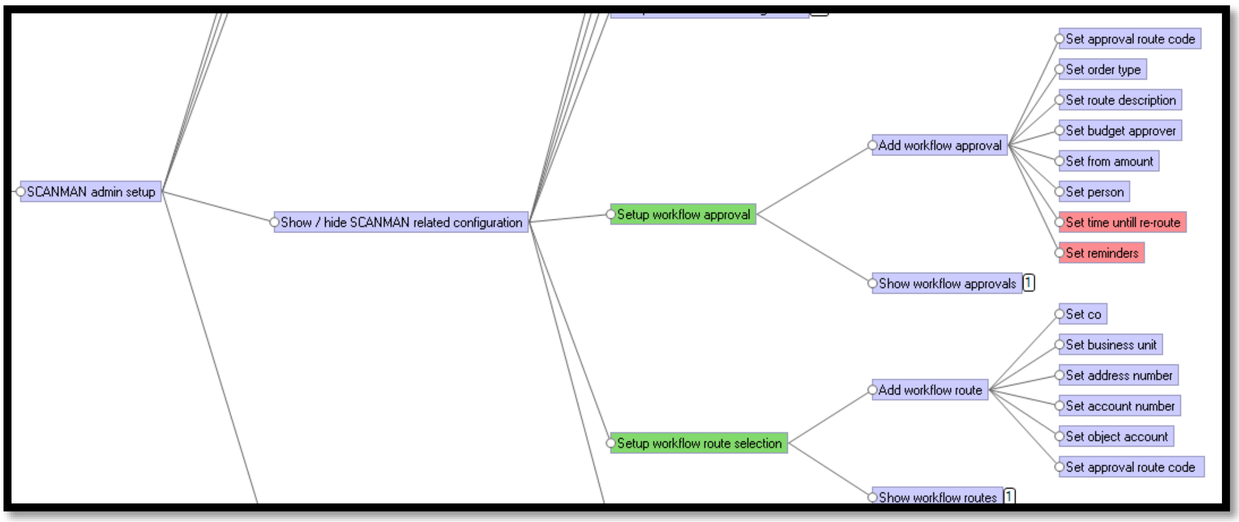
As a manager, I want to enable staff to create internal invoices, so that cheque requisitions can be replaced.

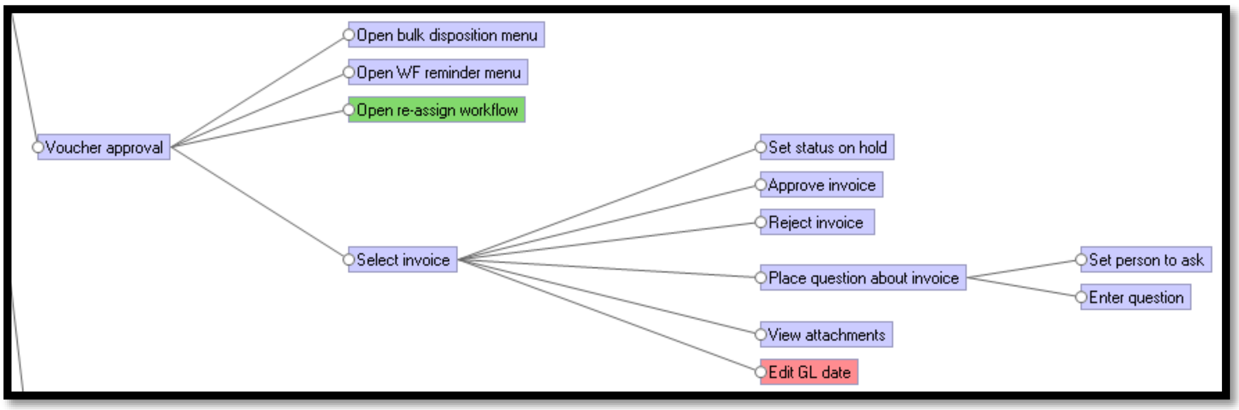
Out of scope for the application.

6 Architecture

Since no epics were identified, no changes were made to the functional architecture diagram. The features identified as solution to the already solved requirements were coloured green in a copy of the feature diagram, and the new features that are part of the customisation for this customer were added in red.

Sections of the feature diagram where new features (red) are to be added.





7 Evaluation

This case study was easy to perform, as the requirements were delivered by the customer in a clear format. The requirements were documented in a table format, with the requirements already prioritised in mandatory and preferred by the customer. Each of the requirements was described in 2-3 lines of text. These requirements could then be transformed into user stories, since this was a retroactive case study, the “so that” part of the user stories was mostly inferred and not formulated by the actual customer.

Matching the user stories to features in the feature diagram was easy to do. When no feature could be located that fulfilled the requirement in the user story, it was concluded that a new feature was required to satisfy the customer. In total 2 of the requirements formulated by the customer needed development, resulting in three new features. This suggests that only three user stories, and an updated version of the feature diagram would be needed to start development.

Existing documentation of one of the two requirements could be located, this document contains three pages of text (1178 words). In this case study, the user stories that needed development would require about ~100 words, and the two figures. Since the development was still in progress, the location of the solution in the architecture could not be compared.

Not all questions of the interview protocol could be asked in the evaluation of this case, as some of them are focused on the epic stories / module level, which was not applicable in this specific case. In order to still obtain the interviewee's opinion on these questions, they were presented with some of the deliverables of the other case studies.

Case study 2: Customised Voucher Match Report

1 Customisation request:

The customisation request stemmed from the implementation of the standard version of SCANMAN JDE at a new customer. The customer had some requirements for the application that it currently did not fulfil. The customer required a new way to automatically match invoices to vouchers, and some additional changes. This case was chosen as it had development on an ES / module level, and relatively high amount of customisation.

2 Elicit requirements:

Since this case study is done retroactively, obtaining the requirements was done by analysing the documentation, observation of a knowledge transfer session and issues logged on Jira. The created functionality was reverse engineered to obtain the set of requirements for this case study. Since the documentation was limited to issues logged in Jira, they only describe functionality that was not included when the issue was logged. This means that opposed to the other retrospective case study, no requirements were matched to existing features except when they were added to the standard version.

3 Epic story formulation:

When I receive an invoice that does not match an existing purchase order, I want to receive a report when it exceeds the tolerance set in the ERP, so that the invoice can be rejected.

4 Project briefs

VMA comparison report

When I receive an invoice that does not match an existing purchase order, I want to receive a report when it exceeds the tolerance set in the ERP, so that the invoice can be rejected.

What problem are we solving and why?

When a purchase order is available in the ERP, this means that it is acknowledged that an invoice will come in for a certain product at a certain price. If this is tested, and the invoice is different from the purchase order, this indicates a possible error in either value. Currently someone has to manually check both files and look long enough until they either find an error in one of the files, or in the ERP entries. The VMA comparison tool and its report will allow users to quickly read what the error is, and in which section of a document it occurs.

Relevant epic stories / modules

Supplier ledger inquiry

Reports

Purchase order

Voucher processing

Invoice line items

Success criteria:

- The tool recognizes when there is a difference between the invoice and PO, and runs when this difference is detected.
- The tool generates a report that shows in which lines of the PO and invoice a difference occurs.
- The tool generates a report that shows how big the differences are between the PO and invoice.
- The tool can be set up to ignore differences if they fall within tolerances.

5 User stories

1. As a manager, I want to ensure that all verified invoices contain a invoice number, so that no invoices can enter the approval process without an invoice number
2. As an approver, I want to be able to access the speed status change when I redistribute vouchers, so that I can update the pay item status for distributed vouchers.
3. As a manager, I want to use logged vouchers in the voucher match automation, so that received invoices can automatically be set to on hold when the invoiced goods are not received yet.
4. As a manager, I want to track freight costs of each invoice, so that transport costs can be properly monitored.
5. As an AP clerk, I want to be able to sort the worklist using any of the available columns, so that I can easily find the invoices I need.
6. As a manager, I want to ensure that record reservations are only active when an invoice is in edit mode, so that the reservation time of a record is minimized and other users can pick it up as soon as possible.
7. As a manager, I want to see the due date of a record in the redistribution application, so that I can redistribute records based on when they are due.

When I receive an invoice that does not match an existing purchase order, I want to receive a report when it exceeds the tolerance set in the ERP, so that the invoice can be rejected.

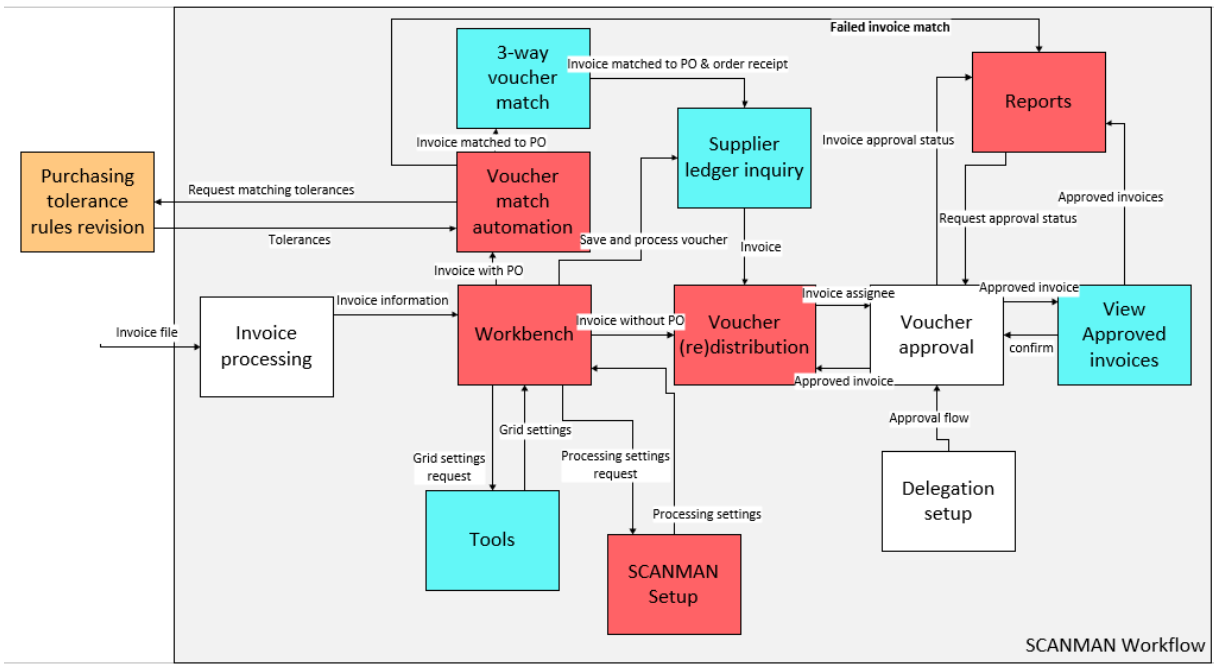
1. As a manager, I want to set percentage tolerances for matching a invoice to its purchase order, so that only invoices that differ more than the percentage are rejected.
2. As a manager, I want to set amount tolerances for matching an invoice to its purchase order, so that only invoices that are below this amount can be processed without a manual check.
3. As an AP clerk, I want to see why the invoice was rejected, so that I can review the rejected invoice and see if I can fix the issue manually.
4. As an AP clerk, I want to see what the difference is between the PO amount and the invoice amount, so that I can see how big the difference is.

5. As an AP clerk, I want to see the amount of the invoice, so that I can check if the ERP value amount matches the invoice.
6. As an AP clerk, I want to see the amount of the PO, so that I can check if the ERP value matches the PO.
7. As an AP clerk, I want to see the invoice ID, so that I can locate the invoice when I need to further act on the matching issue.

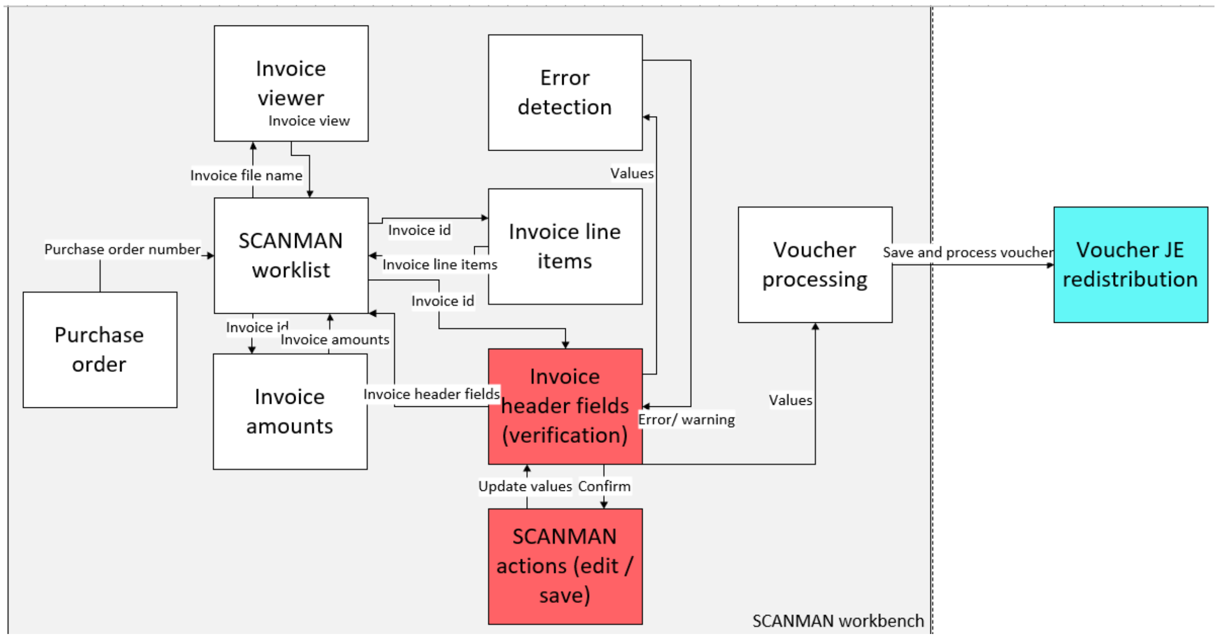
6 Architecture

Functional architecture diagrams

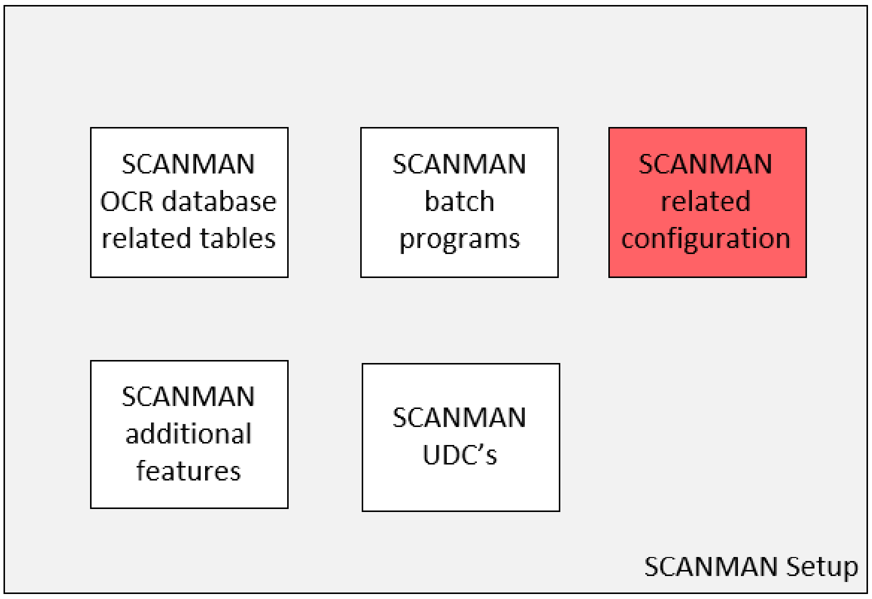
SCANMAN Workflow



SCANMAN workbench

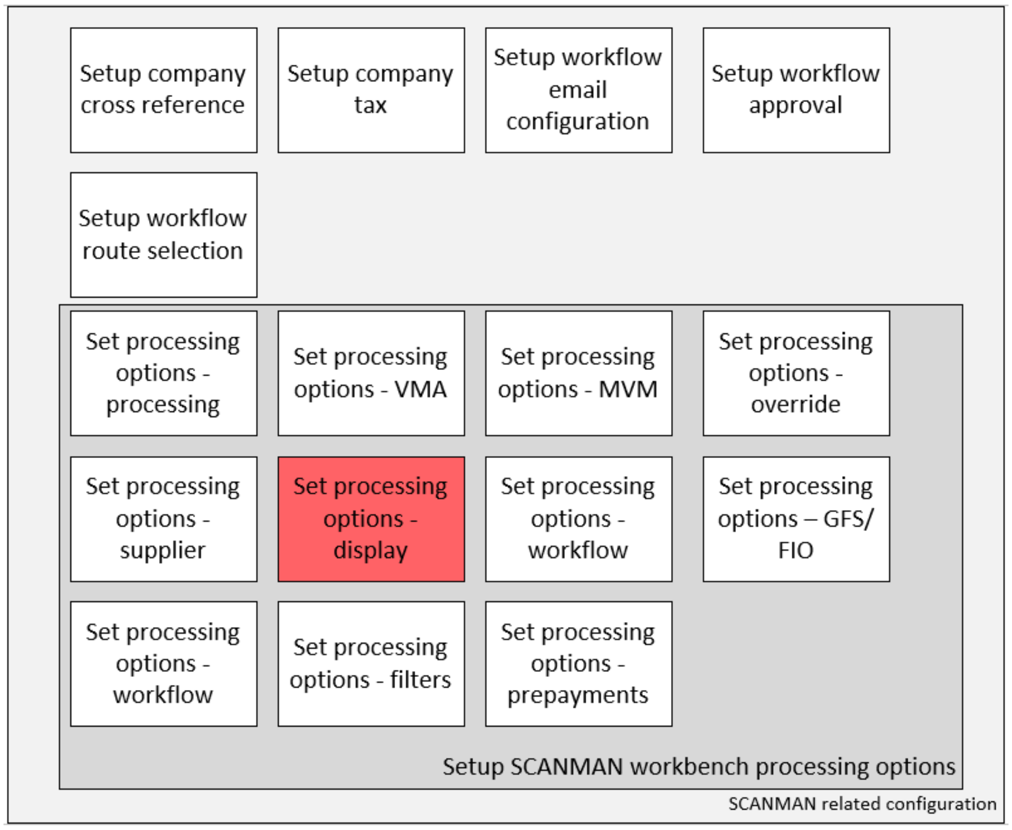


SCANMAN setup



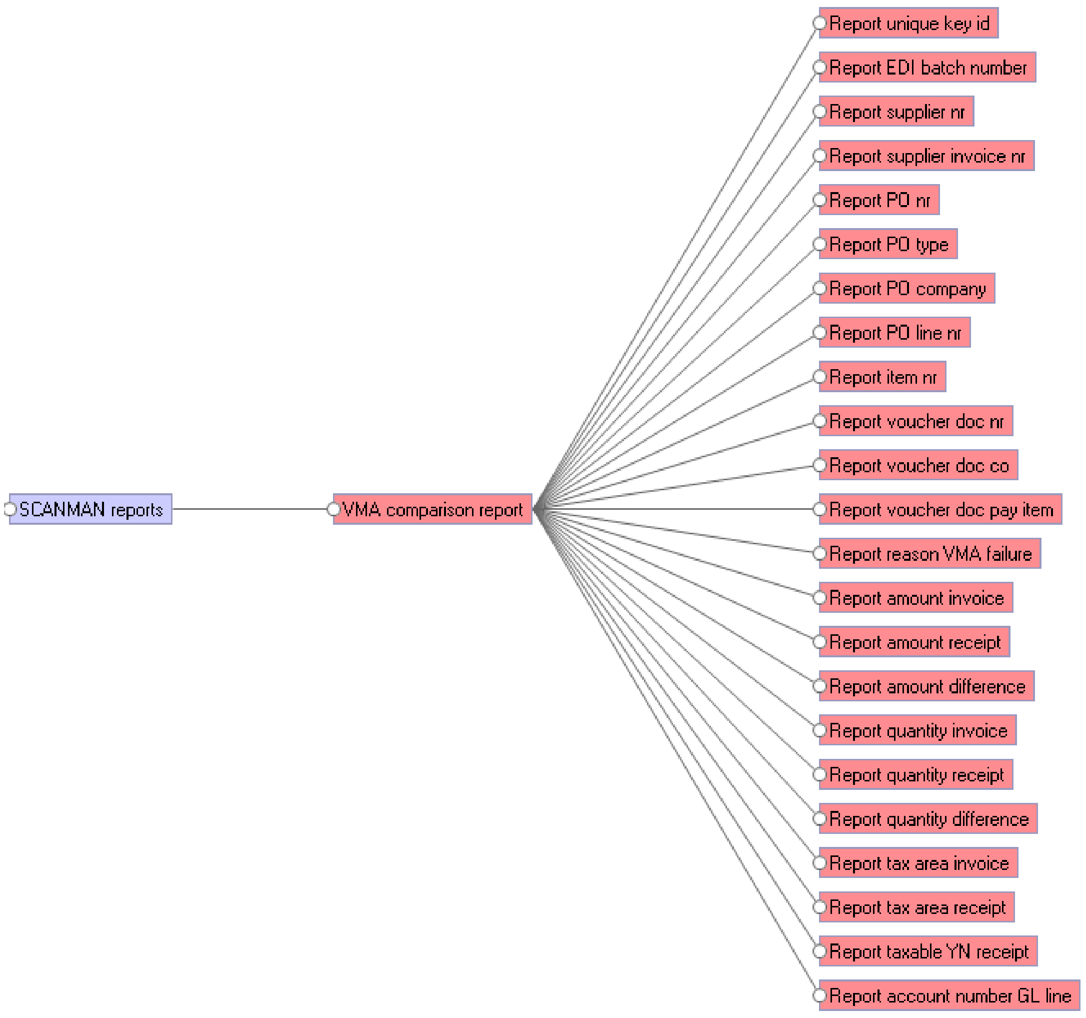
SCANMAN OCR database related tables
SCANMAN batch programs
SCANMAN related configuration
SCANMAN additional features
SCANMAN UDC’s
SCANMAN Setup

SCANMAN processing options

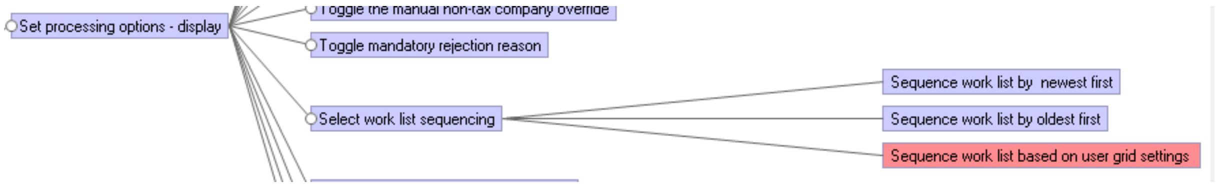

Setup company cross reference
Setup company tax
Setup workflow email configuration
Setup workflow approval
Setup workflow route selection
Set processing options - processing
Set processing options - VMA
Set processing options - MVM
Set processing options - override
Set processing options - supplier
Set processing options - display
Set processing options - workflow
Set processing options – GFS/ FIO
Set processing options - workflow
Set processing options - filters
Set processing options - prepayments
Setup SCANMAN workbench processing options
SCANMAN related configuration

Feature diagrams:

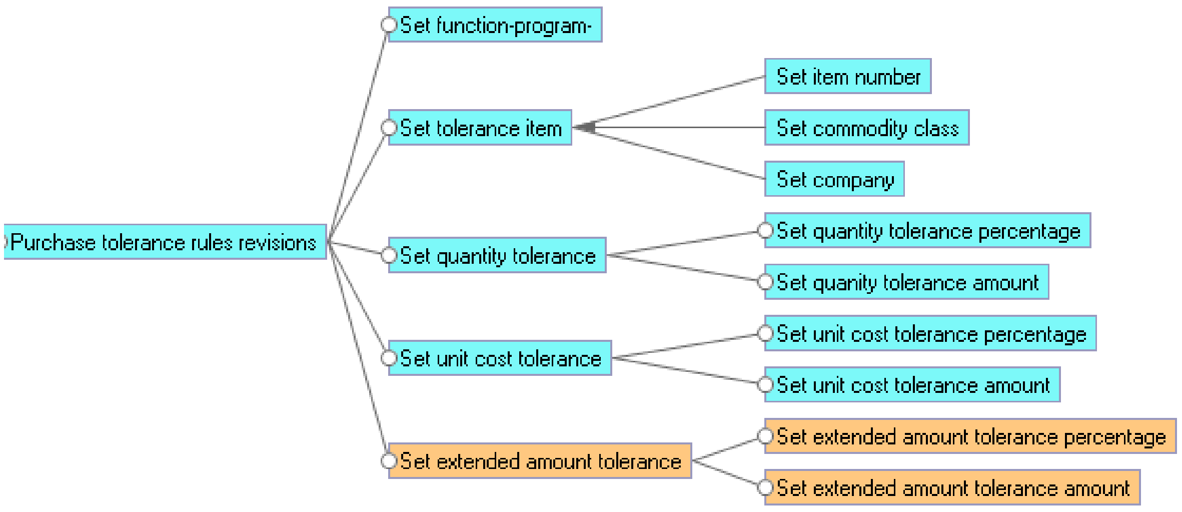
VMA comparison report



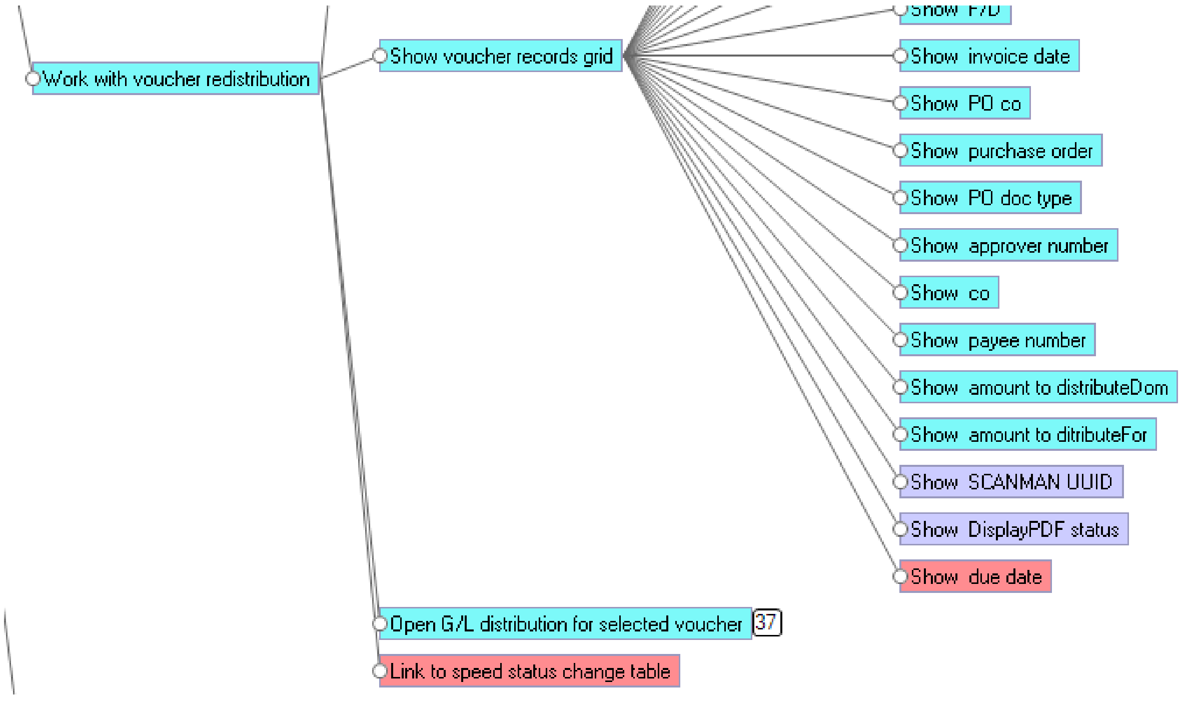
Processing options


Set processing options - display
Toggle the manual non-tax company override
Toggle mandatory rejection reason
Select work list sequencing
Sequence work list by newest first
Sequence work list by oldest first
Sequence work list based on user grid settings

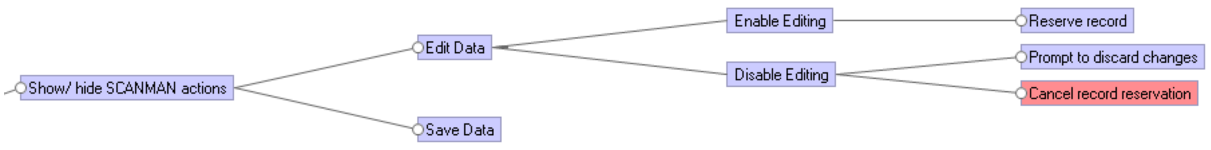
Tolerance rules

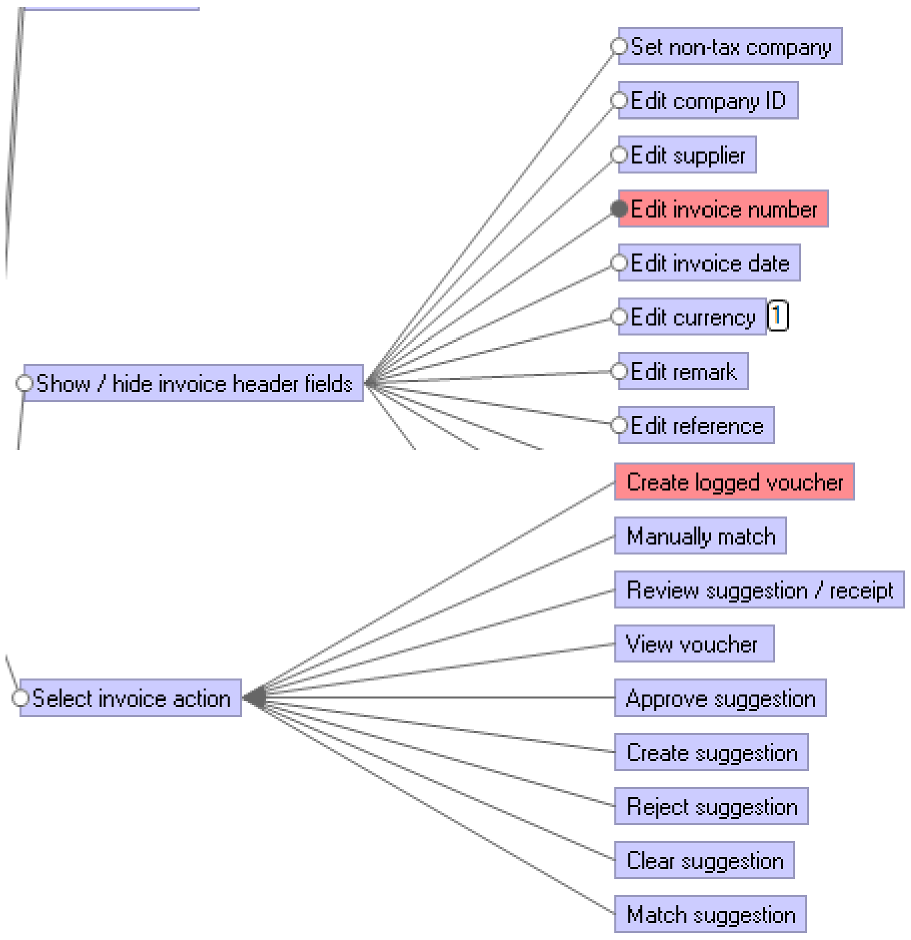

Purchase tolerance rules revisions
Set function-program-
Set tolerance item
Set item number
Set commodity class
Set company
Set quantity tolerance
Set quantity tolerance percentage
Set quanity tolerance amount
Set unit cost tolerance
Set unit cost tolerance percentage
Set unit cost tolerance amount
Set extended amount tolerance
Set extended amount tolerance percentage
Set extended amount tolerance amount

Voucher redistribution


Work with voucher redistribution
Show voucher records grid
Show F/D
Show invoice date
Show PO co
Show purchase order
Show PO doc type
Show approver number
Show co
Show payee number
Show amount to distributeDom
Show amount to ditributeFor
Show SCANMAN UUID
Show DisplayPDF status
Show due date
Open G/L distribution for selected voucher
37
Link to speed status change table

Cancel record reservation


Show/ hide SCANMAN actions
Edit Data
Save Data
Enable Editing
Disable Editing
Reserve record
Prompt to discard changes
Cancel record reservation


Set non-tax company
Edit company ID
Edit supplier
Edit invoice number
Edit invoice date
Edit currency
1
Edit remark
Edit reference
Show / hide invoice header fields
Create logged voucher
Manually match
Review suggestion / receipt
View voucher
Select invoice action
Approve suggestion
Create suggestion
Reject suggestion
Clear suggestion
Match suggestion

7 Evaluation

The MVM comparison report (which is the biggest customisation in this case study) relies on some standard JDE functionality. To properly colour code this was a difficult task, as there were multiple options that had to be considered:

- All standard JDE functions were colour coded cyan in the feature diagram, which would indicate that the amount tolerance should also be colour coded cyan. However, since the customised functionality relies on this specific part of the module, it could be beneficial to colour it differently in order to have it jump out compared to the other JDE components which is what we want to achieve.
- Since it is linked to a customisation, it could be colour coded red. This would achieve the goal of catching the attention in the feature diagram. However, it would also be untrue to the legend for the figure, as these features were not modified, but only used.
- Therefore it was finally decided to use orange colour coding for these features, this colour is often associated with warning signs. While it might lose a bit of clarity on whether or not it is a JDE element (as it is not colour coded cyan) this was deemed less important than having the increased attention that comes with this colour choice.

It could also be decided that this information reliance can be detected from the information flow between the modules in the FAD, and therefore does not need to be included in the feature diagram. But in this case it seemed important enough to highlight, as this allows to detect that updates/ changes in these features from a new JDE version can cause issues in the customisation.

The feature diagram is limited in that it only shows the user interactable features, for this case this shows in the VMA match report. Only this report is a user visible addition, however all the technical changes required to generate this report are not included in the feature diagram since they are not user visible. These user hidden customisations are not visualised in a feature diagram.

The VMA match report has a higher number of features than user stories (23 features, 7 US). This has a number of possible explanation: It could be that all relevant fields were selected from a list of existing categories, which would result in a higher number of features than considering which fields are a must have for a user. Another explanation could be that due to case study limitations context information was not considered during the user story creation, which was considered in the actual functionality creation.

Although the feature diagram becomes hard to comprehend in full due to the size of the application, and number of features, colour coding the FAD can help us locate where customised features are. By checking the FAD, it becomes possible to see which modules in the feature diagram should be checked to detect what customer specific changes were made, which allows the user to quickly zoom to the relevant section of the feature diagram. For example, the user could see in the top level FAD that the setup was customised, then in the SCANMAN setup FAD it shows that the SCANMAN related configuration was altered.

Tooling support to automatically allow for drill down navigation between the different FAD's and into sections of the feature diagrams would be very useful. This was now done by linking them in PowerPoint for a proof of concept, but this is not sustainable in a work environment. In order to most effectively make use of the RE4SA concepts it would be needed to automate the connections between the different concepts, where clicking on a module would lead to its submodules or its features depending on the levels of depth. In this case this would be particularly useful due to the depth of the customised process options.

Case study 3: Version update

1 Customisation request:

While this case does not directly describe a customisation for a single customer, it does detail a version update. This update collects requirements from many different customers that are to be included in the standard version of SCANMAN. This case will provide further evidence of using the situational RE4SA method for designing additional functionality of the case software which should prevent the need for customer specific customisation.

2 Elicit requirements:

The requirements were obtained through analysis of all the issues and comments logged in the product JIRA for version 3.8. These issues, designs, comments and solutions were used to formulate the epic stories and user stories. The issues originate from multiple sources: specific customer requests deemed to be an overall product improvement, internal suggestions and bug reports. Since the RE4SA model is focused on creation of new functionality, bug reports were not included in this case study.

3 Epic story formulation:

When <problematic situation>, I want <motivation>, so that <expected outcome>

When there is an issue with an invoice, I want a way to contact another JDE user, so that the issue can be resolved by the relevant user.

When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of invoices that completed the approval process

When invoices are not approved after a set amount of days, I want to automatically re-assign / escalate these invoices, so that I can ensure they are handled in time

4 Project briefs

4.1 Question and Answer

> *When there is an issue with an invoice, I want a way to ask another JDE user a question, so that the issue can be resolved by the relevant user, or additional information can be supplied.*

What problem are we solving and why?

Communication between JDE users is mostly done outside of JDE. For example through email, chat, phone calls etc. It would be valuable to facilitate the asking and answering of questions within JDE, so users can immediately post a question without interrupting their work. This way less questions go unasked, and the knowledge in the answers is stored within JDE.

Relevant epics / modules

Workbench (worklist / Invoice header fields (verification) / user information / invoice amounts / invoice viewer)

Voucher approval

How will we measure success?

- The application allows users to ask a specific user a question
- The application notifies users when a question is asked of them
- The application allows users to answer a question
- The application notifies users when their question is answered

4.2 Invoice reporting

When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of invoices that completed the approval process

What problem are we solving and why?

Management needs to keep track of invoices, and an overview is required for them to obtain information about the invoices. For example if an employee has rejected all invoices they received in the last three weeks, management needs to be able to find this information so they can contact the employee and obtain more information about the situation.

Relevant epic stories / modules

Invoice approval

Approval assignment

How will we measure success?

- A report can be generated, which shows invoice acceptance metrics (# accepted, rejected, open, invoice amounts, etc.) on a specific user basis.
- A report can be generated, which shows an overview of all monthly invoices and their status plus the number of invoices in each status.

4.3 Invoice escalation

When invoices are not approved after a set amount of days, I want to automatically re-assign / escalate these invoices, so that I can ensure they are handled in time

What problem are we solving, and why?

Sometimes invoices are kept in the backlog for too long, this causes issues as invoices are not paid and suppliers might get annoyed. The escalation module will allow managers to set up escalation paths, which ensure that if an invoice is not approved in time, a different employee will receive the invoice and it can still be handled.

Relevant epic stories / modules

Invoice approval

Approval assignment

How will we measure success?

- The application escalates invoices to a specified user when the escalation date has passed.
- The application sends notifications as set up, when closing in on the escalation date.
- The application tracks which escalations have occurred.

5 User stories

5.1 When there is an issue with an invoice, I want a way to contact another JDE user, so that the issue can be resolved by the relevant user.

1. As an approver, I want to ask a question to the person who verified the invoice, so that I can ask them to have another look at invoices that have mismatches.
2. As an approver, I want to set a subject for my question, so that the person who I ask the question can quickly see what it is about.
3. As an AP clerk, I want to receive an email notification when someone asks me a question, so that I know when my input is required
4. As an AP clerk, I want to see the original file relevant to the question, so that I can easily see which entry the question refers to.
5. As a manager, I want to be able to see all questions and answers in an overview, so that I can track which users use it, and what kind of questions are asked.
6. *As an AP clerk, I want to see if my answers to questions have been read, so that I know when the issue is concluded*
7. As an approver, I want to receive an email notification when my question is answered, so that I know when I can further process the invoice
8. As an approver, I want to be able to react to an answer, so that the Q&A can be continued until my issue is solved.
9. As an approver, I want to edit my question, so that I can add further info or remove errors.
10. As an approver, I want to be able to delete my question, so that an issue that has been solved before receiving an answer can be removed.
11. As an AP clerk, I want to edit my answer, so that I can add more information after committing my first answer.
12. As an AP clerk, I want to be referred to the relevant worklist entry for a question, so that I can obtain the information I need to answer the question.
13. As an approver, I want to add an attachment to my question, so that I can effectively communicate what I'm asking about.
14. As an AP clerk, I want to add an image to my answer, so that I can visualise my answer
15. As an AP clerk, I want to see the timestamp of my answer, so that I can show that I answered a question before it was edited.
16. As an AP clerk, I want to see who asked the question, so that I can contact them outside JDE when it is more appropriate to answer in person.

5.2 When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of invoices that completed the approval process

1. As a manager, I want to receive reports that show key information for approved invoices, so that I don't have to manually check all approved invoices.

2. As a manager, I want to receive reports that show information about the status for open invoices, so that I can have an overview of all invoices that haven't been processed.

5.3 When invoices are not approved after a set amount of days, I want to automatically re-assign / escalate these invoices, so that I can ensure they are handled in time.

1. As a manager, I want to be able to set up automatic re-routing for an invoice if it has not been approved within a certain time, so that invoices don't stay unsolved for a long time.
2. As an approver, I want to receive reminders for invoices that are close to automatically re-routing, so that I can prevent invoice escalation and assure I complete my tasks.
3. As a manager, I want to set up escalation profiles, so that I can reuse specific settings for invoice escalation.
4. As a manager, I want to edit escalation profiles, so that I can change profiles as processes or needs change.
5. As a manager, I want to receive reports of escalations that occur, so that I can contact the employees that have not met their deadline.
6. As a manager, I want to see an overview of escalation rules that are set up, so that I can see what options are available.

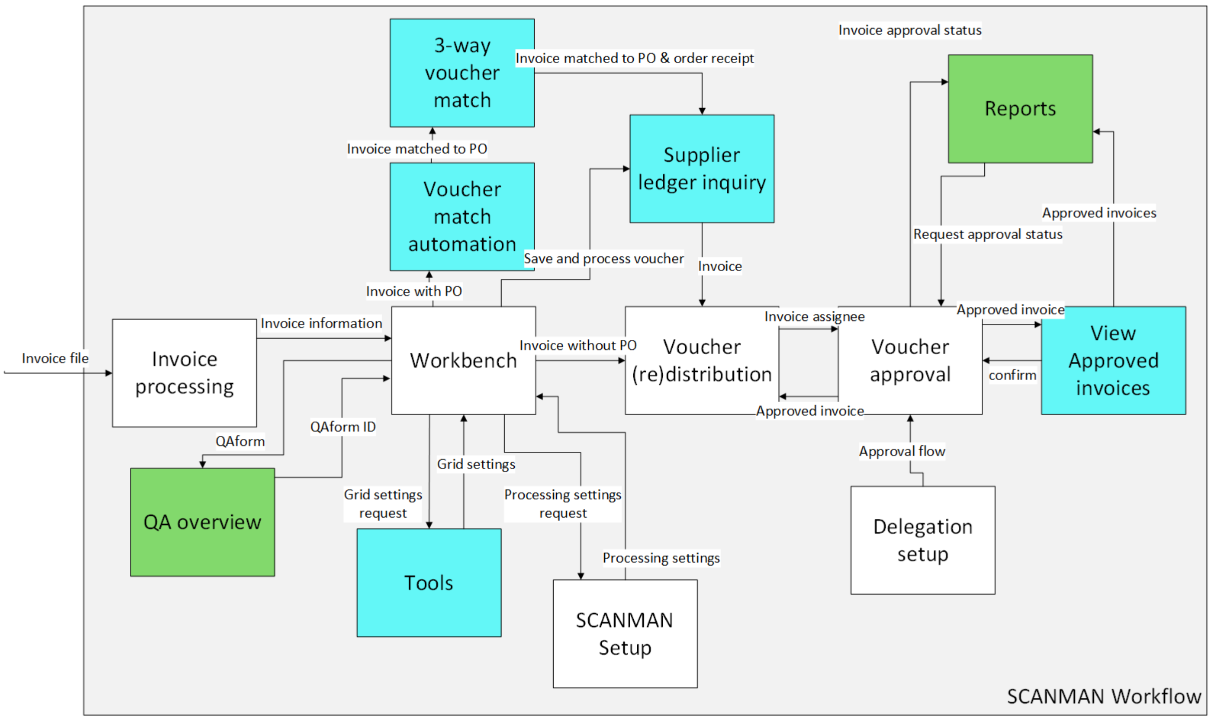
User stories not belonging to a new epic

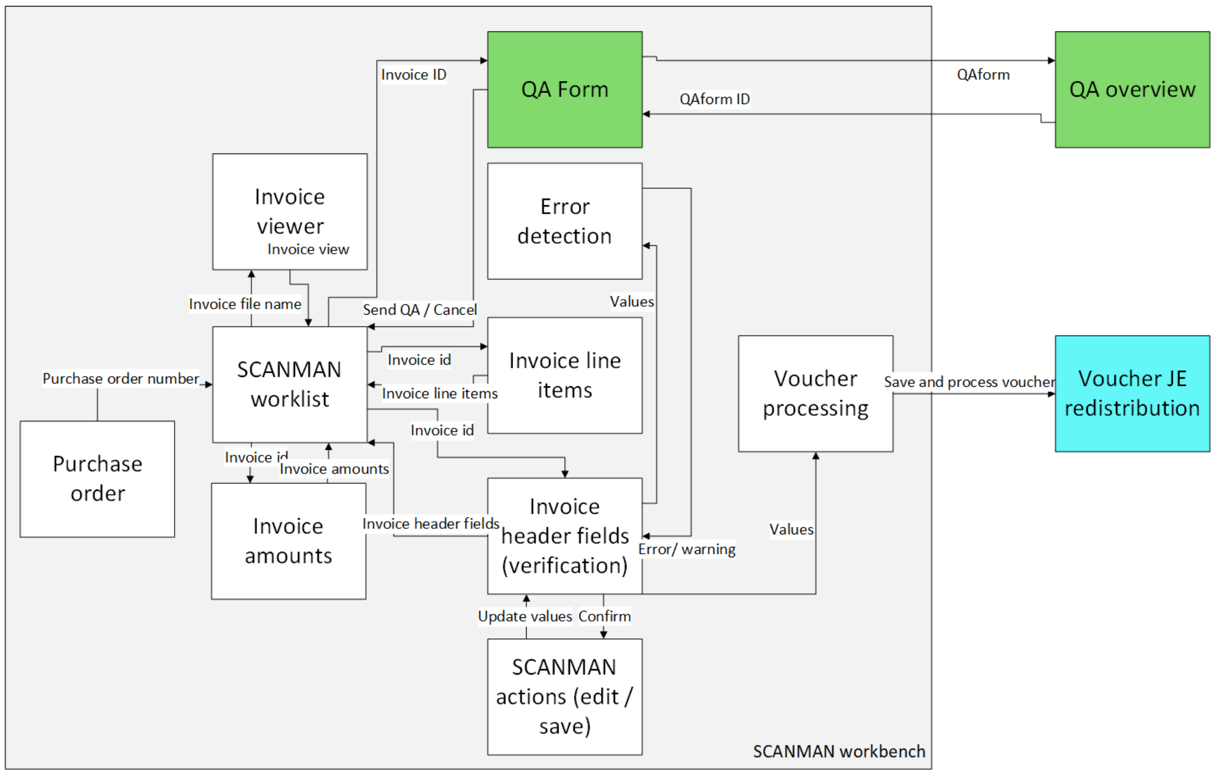
1. As an AP clerk, I want to set the frequency of notification emails, so that my mailbox doesn't get spammed with notifications
2. As an approver, I want to set the frequency of notification emails, so that my mailbox doesn't get spammed with notifications
3. As an approver, I want to have the OCR recognize the currency used in the invoice, so that I don't have to do this manually.
4. As a manager, I want SCANMAN to use the appropriate values for the currency, so that limits set up in the application are checked against the converted value and not against the absolute value.
5. As a manager, I want to search suppliers in a customisable way, so that I can manually set up search categories for my suppliers.
6. As a manager, I want to automatically approve an invoice if it is below a certain value, so that small expenses can be automated without requiring manual labour.
7. As a manager, I want to ensure that record reservations are only active when an invoice is in edit mode, so that the reservation time of a record is minimized and other users can pick it up as soon as possible.
8. As an AP clerk, I want to retain manual edited values on an invoice when it is rejected, so that I don't have to start the verification process from scratch.
9. As a manager, I want to ensure that no invoices can be processed that have a General Ledger date in a period that is already processed, so that I can ensure payment for all processed invoices.
10. As an approver, I want to disable approval actions for invoices that have already been approved, so that I can easily see that the invoice is already approved.
11. As a manager, I want to be able to re-assign an invoice, so that I can send it to the correct user if I make a mistake in the initial assignment

12. As a manager, I want to set up approval for invoices that correctly match an approved purchase order, so that I can assure that the invoice meets all conditions for payment.
13. As a manager, I want to ensure that invoices enter an approval route after manual voucher match, so that I can assure that the invoices go through the correct approval procedure.
14. As an AP clerk, I want to be able to enter an approval route after matching an invoice to a PO, so that the invoice enters the correct approval process, and doesn't remain unapproved.
15. As a manager, I want to have the invoices matched to tax areas, so that they can automatically be assigned the correct tax code.
16. As a manager, I want to ensure that approvals can only be delegated to employees with appropriate approval limits, so that employees don't approve invoices above their restrictions, and that all delegated invoices can be processed.

6 Architecture

6.1 Functional architecture diagram Update




QA Form
QA overview
QAform
QAform ID
Invoice ID
Invoice viewer
Invoice view
Error detection
Invoice file name
Send QA / Cancel
Values
SCANMAN worklist
Invoice id
Invoice line items
Invoice line items
Purchase order number
Voucher processing
Save and process voucher
Voucher JE redistribution
Invoice id
Purchase order
Invoice id
Invoice amounts
Invoice amounts
Invoice header fields
Invoice header fields (verification)
Error/ warning
Values
Update values
Confirm
SCANMAN actions (edit / save)
SCANMAN workbench

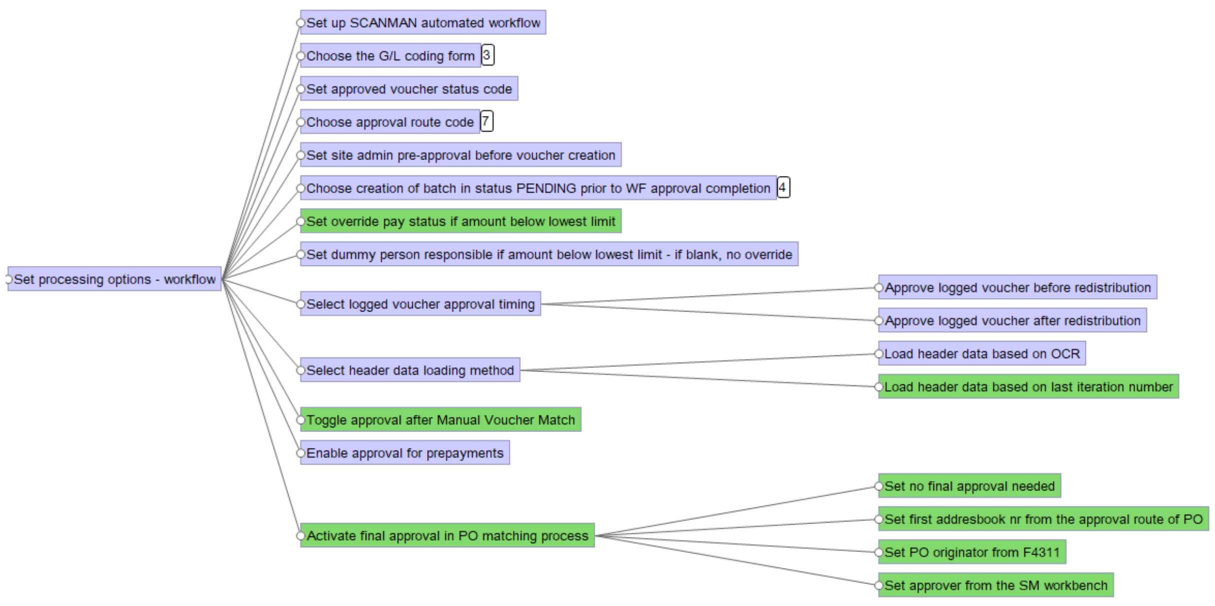
6.2 Feature diagram update

Orange - to be removed? (based on comments on the Jira)

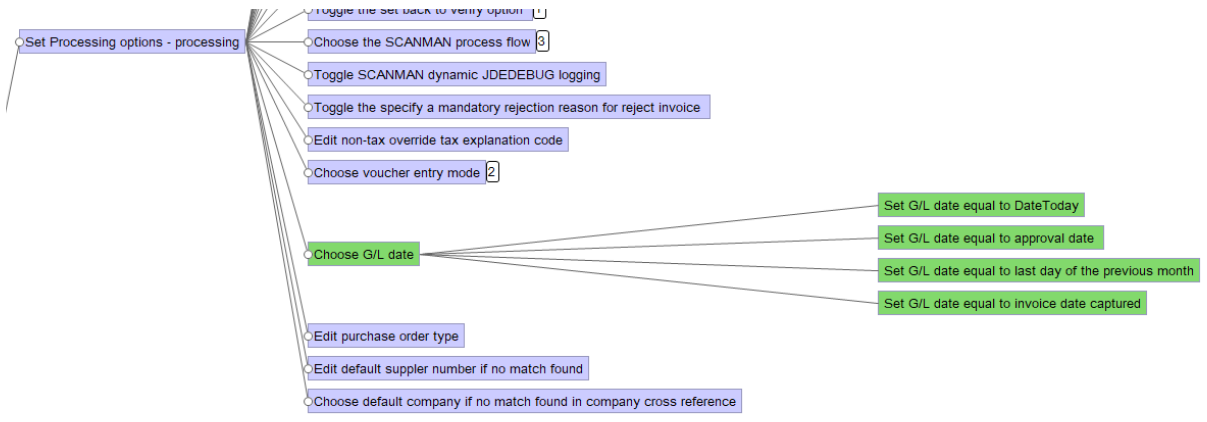
Green - new features

Grey - shared features for that level of degree

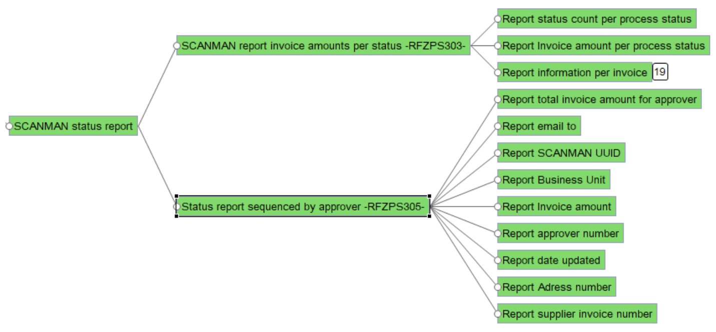
New workflow processing options


Set processing options - workflow
Set up SCANMAN automated workflow
Choose the G/L coding form 3
Set approved voucher status code
Choose approval route code 7
Set site admin pre-approval before voucher creation
Choose creation of batch in status PENDING prior to WF approval completion 4
Set override pay status if amount below lowest limit
Set dummy person responsible if amount below lowest limit - if blank, no override
Select logged voucher approval timing
Approve logged voucher before redistribution
Approve logged voucher after redistribution
Select header data loading method
Load header data based on OCR
Load header data based on last iteration number
Toggle approval after Manual Voucher Match
Enable approval for prepayments
Activate final approval in PO matching process
Set no final approval needed
Set first addressbook nr from the approval route of PO
Set PO originator from F4311
Set approver from the SM workbench

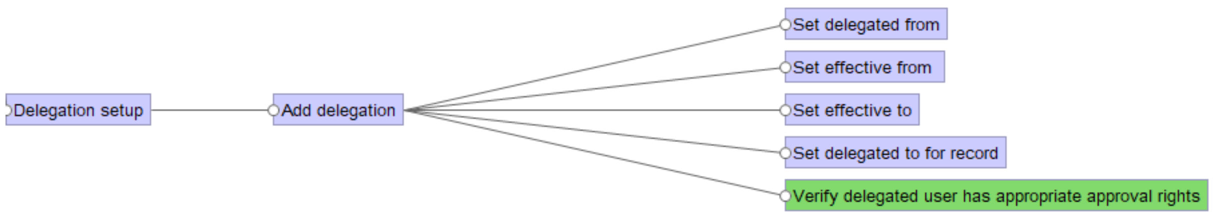
G/L date settings


Set Processing options - processing
Toggle the set back to verify option 1
Choose the SCANMAN process flow 3
Toggle SCANMAN dynamic JDEDEBUG logging
Toggle the specify a mandatory rejection reason for reject invoice
Edit non-tax override tax explanation code
Choose voucher entry mode 2
Choose G/L date
Set G/L date equal to DateToday
Set G/L date equal to approval date
Set G/L date equal to last day of the previous month
Set G/L date equal to invoice date captured
Edit purchase order type
Edit default supplier number if no match found
Choose default company if no match found in company cross reference

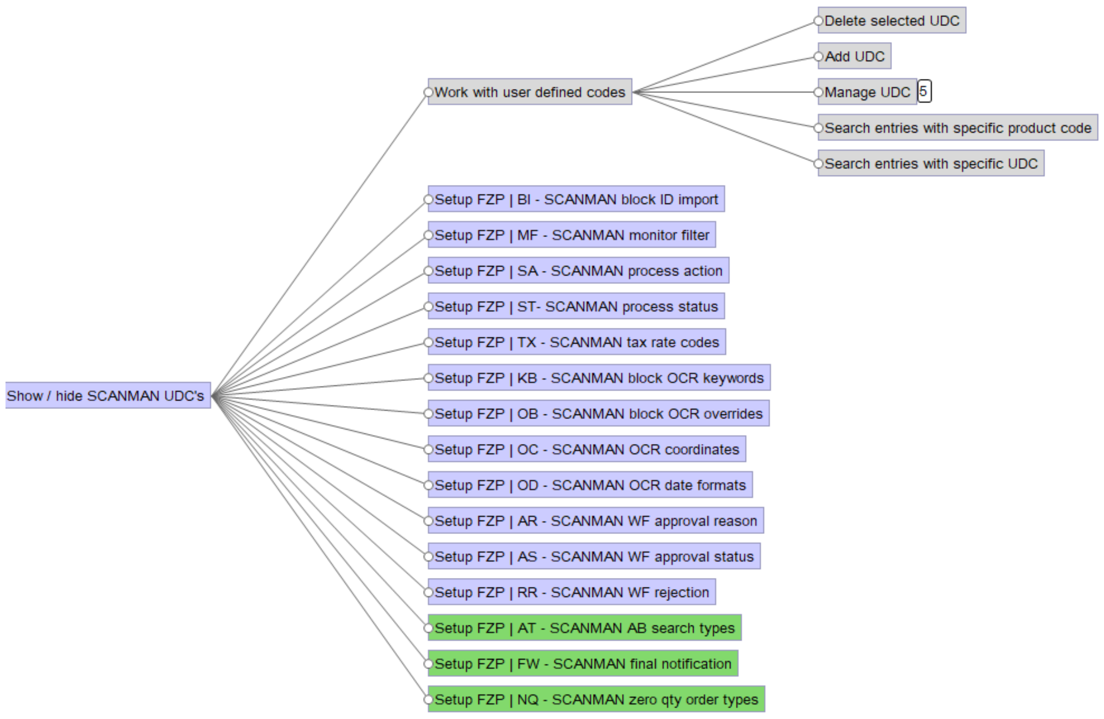
New SCANMAN reports


SCANMAN status report
SCANMAN report invoice amounts per status -RFZPS303-
Report status count per process status
Report Invoice amount per process status
Report information per invoice 19
Status report sequenced by approver -RFZPS305-
Report total invoice amount for approver
Report email to
Report SCANMAN UUID
Report Business Unit
Report Invoice amount
Report approver number
Report date updated
Report Adress number
Report supplier invoice number

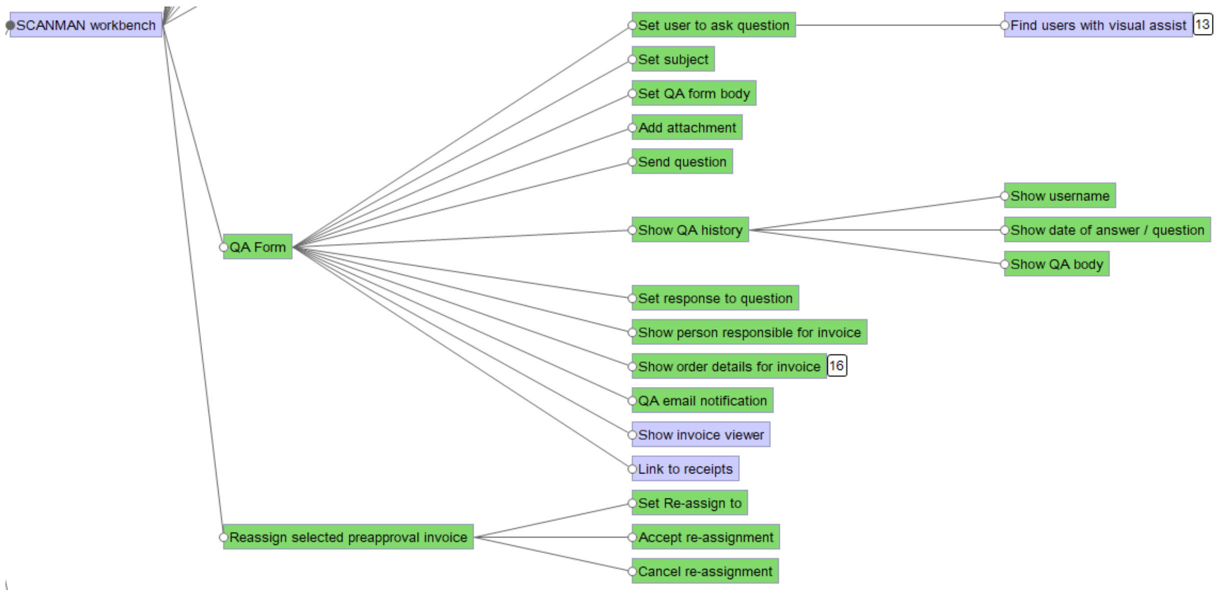
Delegation verification



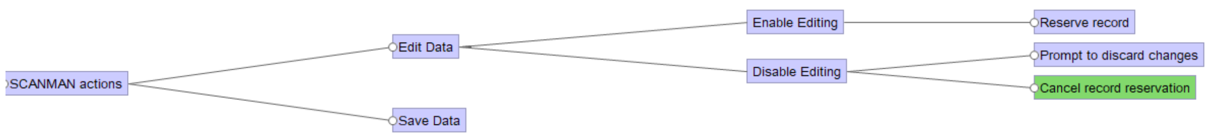
New User Defined Codes, allows custom set up for search types


Show / hide SCANMAN UDC's
Work with user defined codes
Delete selected UDC
Add UDC
Manage UDC 5
Search entries with specific product code
Search entries with specific UDC
Setup FZP | BI - SCANMAN block ID import
Setup FZP | MF - SCANMAN monitor filter
Setup FZP | SA - SCANMAN process action
Setup FZP | ST- SCANMAN process status
Setup FZP | TX - SCANMAN tax rate codes
Setup FZP | KB - SCANMAN block OCR keywords
Setup FZP | OB - SCANMAN block OCR overrides
Setup FZP | OC - SCANMAN OCR coordinates
Setup FZP | OD - SCANMAN OCR date formats
Setup FZP | AR - SCANMAN WF approval reason
Setup FZP | AS - SCANMAN WF approval status
Setup FZP | RR - SCANMAN WF rejection
Setup FZP | AT - SCANMAN AB search types
Setup FZP | FW - SCANMAN final notification
Setup FZP | NQ - SCANMAN zero qty order types

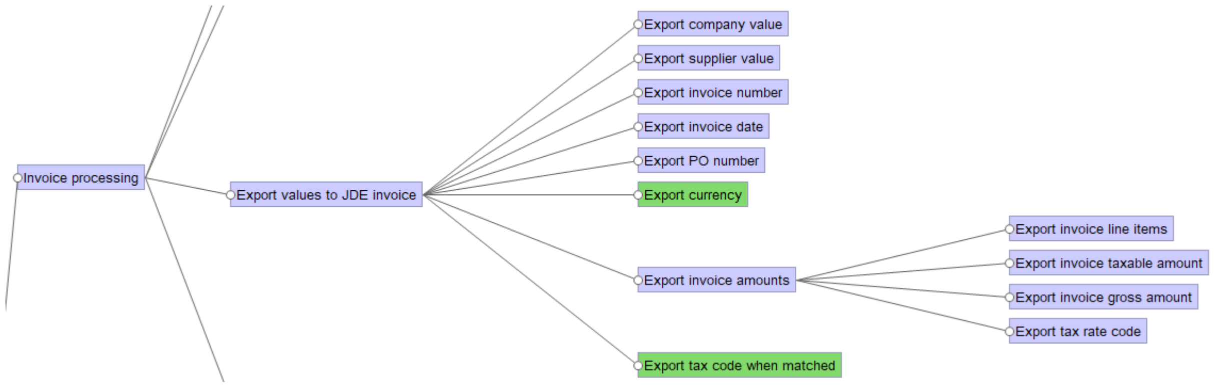
Q&A form functionality and Invoice re-assignment



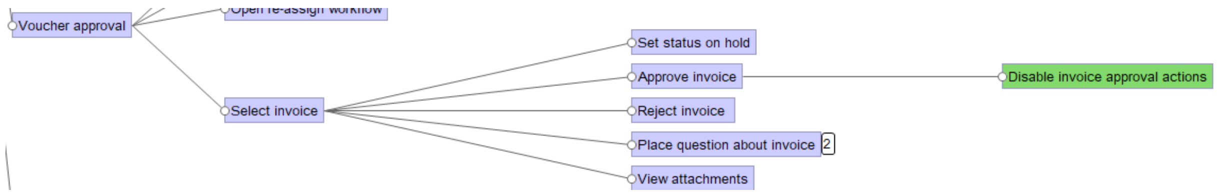
Record reservation minimisation



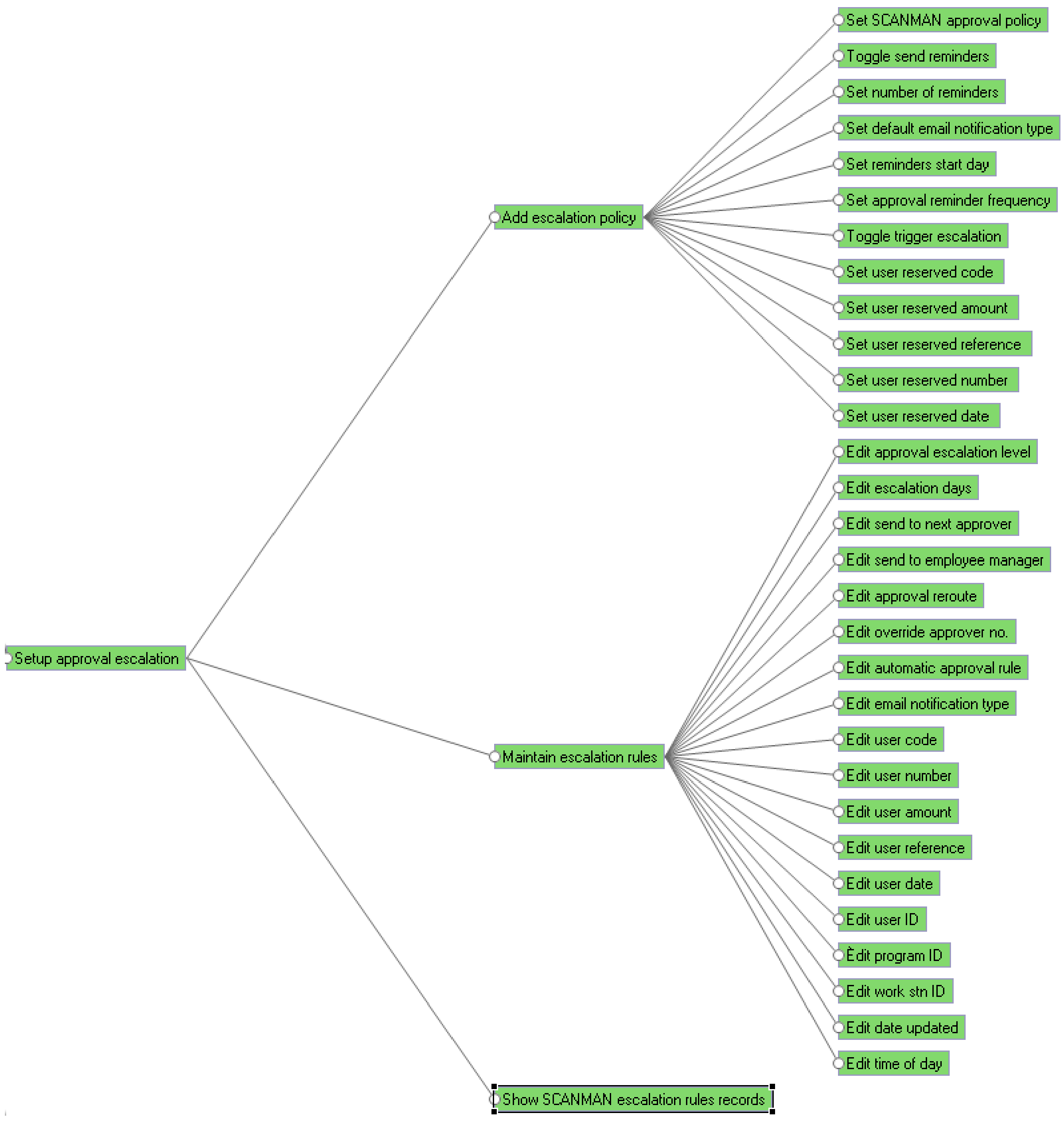
New invoice processing / OCR functions



Disable invoice approval actions



Automatic approval escalation


Setup approval escalation
Add escalation policy
Maintain escalation rules
Show SCANMAN escalation rules records
Set SCANMAN approval policy
Toggle send reminders
Set number of reminders
Set default email notification type
Set reminders start day
Set approval reminder frequency
Toggle trigger escalation
Set user reserved code
Set user reserved amount
Set user reserved reference
Set user reserved number
Set user reserved date
Edit approval escalation level
Edit escalation days
Edit send to next approver
Edit send to employee manager
Edit approval reroute
Edit override approver no.
Edit automatic approval rule
Edit email notification type
Edit user code
Edit user number
Edit user amount
Edit user reference
Edit user date
Edit user ID
Edit program ID
Edit work stn ID
Edit date updated
Edit time of day

7 Evaluation

Most of the time spent on this case study was to properly analyse the JIRA issues to detect what exactly changed in this version update. This was then formulated in 3 epics and 40 user stories. In total the feature diagram was updated with 70 new features, and the FAD was updated with 4 new modules. There are multiple explanations why the feature to user story relationship is 1.75 to 1, the first explanation is that some features are added based on common sense, and do not have to be documented in user stories. For example on the new report information per invoice feature, many of the specific fields that are included in the report were not documented in the user stories.

A second reason, is that in this case the development was not completely based on the user stories, but mostly on the issues (from which the user stories are derived). This means that the internal validity on this case study is relatively low, and that the cardinality relationship could be different if the developers had used the user stories instead of the JIRA issues.

The third reason, is that some user stories can be solved by a collection of features, for example the user story:

> *" As an approver, I want to be able to react to an answer, so that the Q&A can be continued until my issue is solved."*

Was solved by a combination of the QA history feature (which allows users to see previous questions and answers) and the send question feature, to add a new question to this overview. This can be linked to the increased goal focus, and developer freedom that are the result of user stories.

During the creation of project briefs for the new epic stories in this system update, it was found that the relevant epic stories section was difficult to include as these epics were never created. This would either create a lot of extra work to recreate all the epic stories for the application, or mean that the information would not be available in later stages. In order to solve this issue, the relationship between epic stories and modules was utilised. Instead of recovering the epics that did not exist, the module names were utilised to show which existing parts of the software the new epic was reliant on. When the information flows had to be modelled for the FAD update, this information served the same use as the relevant epics in the NetSuite case study.

As the software architecture is revised for new versions and customisations, this also allows for improvements to the architecture models. While adding the new features to the feature diagram, it was noticed that some existing functionalities were not yet included in the reconstructed architecture. For example, in this update new features were added to the invoice processing module, however in the process of adding these features to the feature diagram it appeared that the invoice processing module had not been included in the feature diagram yet. Because these new features extended a module that was not mapped, this error in the feature diagram could be detected and fixed. Using the situational RE4SA method in multiple projects also allows for refinement of the architecture model, and increases the accuracy of the models over time.

Case study 4: Recreation of the Case Software for a Different ES

1 Customisation request:

In this case the customisation request for NetSuite came from Forza internally. The goal is to create a solution that can solve the same business function that the company provides through SCANMAN for JD Edwards.

This means that the current JDE application needs to be considered to identify the problematic situations that it solves, and to decide whether they should and can be re-created in NetSuite. Since the development language in NetSuite is Java, and in JDE it is mostly based on event rules.

2 Elicit requirements:

The requirements were mostly obtained from the internal staff who are experienced in SCANMAN, documentation available of SCANMAN and the knowledge gained when reverse engineering the application. Furthermore a design for the NetSuite application was already in progress, and was used to determine the epic stories and user stories.

3 Epic story formulation:

The epic stories in this document have been numbered in the order that they were added. Requirements change and get added over time, since the numbers are referenced to in later steps it would take a lot of effort to change the references every time an epic / user stories is added at a later time. Each of the epic stories are written in the following format:

When <problematic situation>, I want <motivation>, so that <expected outcome>

1. When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.
2. When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.
3. When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don't have to keep track of files or do it manually.
4. When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.
5. When I receive an invoice, I want to have it matched to the relevant purchase order if one exists, so that the values can be compared for consistency.
6. When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of processed invoices.
7. When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.
8. When I verify an invoice, I want to be able to edit fields, so that inconsistencies can be removed.
9. When I have to approve an invoice, I want to see the information I need to decide whether to approve the invoice or not, so that I can make a valid decision.
10. When an invoice is fully approved, I want the payment of the invoice to be processed, so that the accounts payable invoice handling process can be completed.

4 Project Briefs

4.1 Project Brief: Scanned invoice handling

> *When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.*

What problem are we solving, and why?

Character recognition is at a stage where it is difficult to ensure that all of the values extracted from a document are correct. Especially when invoices have exceptions, like handwritten values, inconsistent placement of values etc. A user would want to have the assurance that the automatic processing doesn't make mistakes, the ability to verify that the values are correct manually, or that possible errors fall within acceptable margins.

Relevant epic stories

3 When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don't have to keep track of files or do it manually.

4 When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.

8 When I verify an invoice, I want to be able to edit fields, so that inconsistencies can be removed.

How will we measure success?

The application fulfils at least one of the following:

1. The OCR application makes no mistakes when it processes an invoice.
2. The application can provide a reliable estimate on how correct its interpretation is, and can compare this to a value set by the user.
3. The application requires manual verification of the generated values.

- All received invoices are processed

4.2 Project Brief: Invoice overview

> *When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.*

What problem are we solving, and why?

Employees might not be aware of what their task list is, and can have trouble locating the invoices they need to approve. Therefore, it is important that the application can filter out all the relevant invoices that require a specific user's attention.

Relevant epic stories

3 When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don't have to keep track of files or do it manually.

4 When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.

6 When I finish approving invoices, I want to receive a report of all the invoices that were processed, so that I can keep track of the total expenses that have been processed

7 When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.

How will we measure success?

- All invoices that require the user's attention are collected
- All collected invoices are presented in a single view

4.3 Project Brief: ERP integration

> *When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don't have to keep track of files or do it manually.*

What problem are we solving, and why?

Received invoices that need to be added manually to the system, might be lost when the email is buried in a mail box, or a file isn't uploaded. If this is processed automatically, it removes the chance for human error and increases the efficiency of the process.

Relevant epic stories:

1 When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.

2 When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.

4 When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.

5 When I receive an invoice, I want to have it matched to the relevant purchase order if one exists, so that the values can be compared for consistency.

How will we measure success?

- The application can process invoices from at least two different sources.
- The application can process invoices that are in pdf, xml or doc format.

4.4 Project Brief: Invoice viewer

> *When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.*

What problem are we solving, and why?

An AP clerk needs to verify whether or not the values scanned by the OCR are the same as the original invoice. Therefore, it is desirable for the AP clerk to have access to the OCR information, the original invoice and the option to change the values.

Relevant epic stories:

1 When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.

How will we measure success?

- The application can show the original values of the invoice, the scanned values and allows the user to change values. Preferably all on a single screen.
- All the fields that are scanned by the OCR should be included in the verification form.

4.5 Project Brief: Match to purchase order

> *When I receive an invoice, I want to have it matched to the relevant purchase order if one exists, so that the values can be compared for consistency.*

What problem are we solving, and why?

If the ERP already contains a purchase order for the invoice, this information can be used to confirm the values that the OCR recognizes from the invoice. When there is a discrepancy, there is a chance that either the OCR made a mistake, or the value on the invoice / purchase order is incorrect. This will most likely need manual checking.

Relevant epic stories:

1 When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.

3 When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don’t have to keep track of files or do it manually.

How will we measure success?

- The application can compare values between a scanned invoice, and a existing purchase order.
- A tolerance can be set that notifies the user when a discrepancy between PO and invoice is bigger than this tolerance.

4.6 Project Brief: Expense report

> *When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of invoices that completed the approval process*

What problem are we solving and why?

Management needs to keep track of invoices, and an overview is required for them to obtain information about the invoices. For example if an employee has rejected all invoices they received in the last three weeks, management needs to be able to find this information so they can contact the employee and obtain more information about the situation.

Relevant epic stories:

2 When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.

7 When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it

10 When an invoice is fully approved, I want the payment of the invoice to be processed, so that the accounts payable invoice handling process can be completed.

How will we measure success?

- An overview of invoices that have completed the entire approval flow is generated
- Information about the invoices is generated and visualised
    - Ratio of approved to rejected invoices
    - Stats per employee

4.7 Project Brief: Approval assignment

> *When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.*

What problem are we solving and why?

Most companies have an approval flow, this means that more than one employee needs to approve an invoice. Often this has to be in a specific order of people. This needs to be supported in the application.

Relevant epic stories:

2 When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.

9 When I have to approve an invoice, I want to see the information I need to decide whether to approve the invoice or not, so that I can make a valid decision

10 When an invoice is fully approved, I want the payment of the invoice to be processed, so that the accounts payable invoice handling process can be completed.

How will we measure success?

- When an approval flow can be set-up, by configuring an automatic flow (person 1 -> 2 -> 3) Or by allowing a user to assign it to someone after they complete their approval.
- The employee who gets an invoice assigned to them, has to be able to see that it requires their attention.

4.8 Project Brief: Invoice verification

> *When I verify an invoice, I want to be able to edit fields, so that inconsistencies can be removed.*

What problem are we solving and why?

Scanned values and actual values might differ. It needs to be possible to manually verify and edit values, so that the correct values can be processed.

Relevant epic stories

4 When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.

4.9 Project Brief: Invoice approval

> *When I have to approve an invoice, I want to see the information I need to decide whether to approve the invoice or not, so that I can make a valid decision.*

What problem are we solving and why?

An invoice can be approved or rejected based on specific criteria. In this project brief, we attempt to create an overview of the information that the approver needs to decide whether to approve or reject an invoice.

Relevant epic stories

2 When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.

7 When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.

How will we measure success?

- A financial expert should approve of the information included in the application.
- The fields displayed should be configurable by the user.

4.10 Project Brief: Payment handling

> *When an invoice is fully approved, I want the payment of the invoice to be processed, so that the accounts payable invoice handling process can be completed.*

What problem are we solving and why?

An invoice that is fully approved is still not completely processed. As long as the invoice is not paid, the company that sent the invoice will not be satisfied. Once an invoice is fully approved for payment, this payment should also be handled at the designated date.

Related epic stories:

6 When I finish approving invoices, I want to receive a report of all the invoices that were processed, so that I can keep track of the total expenses that have been processed

7 When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.

How will we measure success?

- An invoice that has completed the approval flow is added to the payment handling process
- An invoice that is in the payment handling process, is processed at the specified payment date

5 User stories:

For the writing of user stories, the following roles were identified and considered:

- AP clerk: (Scans, verifies and codes the invoice before it goes to the company)
- Approver: (Approves or rejects invoices)
- (Financial) Manager: (Is responsible for the payment of invoices, and will check if anything goes wrong in the invoice project)
- QA tester: Has to be able to test the application so that it can be assured that it works as intended

For each of the epic stories identified in step 3, as many user stories as possible are written to include all the requirements for each of the epics.

5.1 When I automatically process an invoice, I want to ensure that the values are correct, so that I can prevent errors occurring in our accounts payable invoice processing.

1. As an approver, I want to see that the invoice was verified, so that I know that there should be no inconsistencies.
2. As an AP clerk, I want to have the values on the invoice scanned, so that I don’t have to add them manually.
3. As an AP clerk, I want to see confidence rating that the OCR scanned all values correctly, so that I can decide how much effort I need to spend validating the values.
4. As a QA tester, I want to see the confidence rating of the OCR, so that I can evaluate if it matches my manual analysis.
5. As a manager, I want to make sure that all received invoices are processed, so that I don’t have to tell clients we lost their invoice.
6. As a manager, I want to make sure that files other than invoices are not added to the ERP as an invoice, so that I can ensure that all “invoices” in the ERP are actually invoices.
7. As an AP clerk, I want to automatically add invoices received by email to the ERP, so that I don’t have to manually download the attachment and upload it to the ERP
8. As an AP clerk, I want to automatically add invoices that are placed in a folder to the ERP, so that I automatically store them in a set location
9. As an AP clerk, I want to be able to manually upload invoices, so that I can use this method when the other options fail.
10. As a manager, I want to ensure that invoices that are in both the invoice folder and mailbox are combined into a single entry, so that they don’t get added as two separate invoices.
11. As a QA tester, I want to manually upload invoices, so that I can more easily test specific scenarios

5.2 When I have to approve invoices, I want to have an overview of all invoices that are relevant to me, so that I can see what I need to do.

1. As an approver, I want to sort invoices from oldest to newest, so that I can focus on emptying my backlog.
2. As an approver, I want to sort invoices from newest to oldest, so that I can keep track of new invoices that need my attention.
3. As an approver, I want to search for invoices on invoice ID, so that I can help a colleague when he has an issue or find an invoice specified in a ticket.
4. As an approver, I want to filter invoices on the error status, so that I can resolve invoices that have an error in the approval process.
5. As an approver, I want to filter invoices with unapproved status, so that I can work on approving these invoices.
6. As an approver, I want to filter invoices with approved status, so that I can edit them when I realise I made an error in my approval.
7. As an approver, I want to sort invoices on due date, so that I can approve the invoices that are nearly due.
8. As a manager, I want to sort invoices on gross amount, so that big expenses can be processed first.
9. As a manager, I want to filter invoices on rejected status, so that I can contact the client and solve the issue.
10. As a manager, I want to filter invoices on assignee, so that I can see the number of processed and open invoices of a specific approver.
11. As a manager, I want to filter on assignee ID, so that I can check each entry of my assignee list and see the status of each assignee
12. As a manager, I want to search for invoices on company, so that I can easily find an invoice when communicating with a client.

5.3 When I receive an invoice, I want it to automatically be added to my ERP environment, so that I don't have to keep track of files or do it manually.

1. As a manager, I want files to be transferred to my ERP system, so I have all the relevant information in one secure location
2. As a AP clerk, I want the values that I verified to be transferred to NetSuite, so that approvers can complete their tasks.

5.4 When I have to verify an invoice, I want to see a scanned version of the invoice, so that I can compare the values.

1. As an AP clerk, I want to see what fields the OCR scanned, so that I can easily identify which sections of the invoice the generated fields are based on.
2. As an AP clerk, I want to zoom in / out on the invoice, so it is easier to read and compare values.

3. As an AP clerk, I want to open the pdf reader in another window, so that I can display it on my second screen and optimise my workspace.

5.5 When I receive an invoice, I want to have it matched to the relevant purchase order if one exists, so that the values can be compared for consistency.

1. As an approver, I want to be notified when an invoice I need to approve deviates from the purchase order, so that I can analyse what causes this difference
2. As an approver, I want to be notified when an invoices matches a purchase order, so that I can quickly accept it since it was already approved as a purchase order.
3. As a manager, I want to have the tool compare an incoming invoice to an existing purchase order, so that the invoice is checked against the values of a purchase order

5.6 When invoices are approved (or rejected), I want them to be added to an overview, so that I can keep track of invoices that completed the approval process

1. As a manager, I want to see statistics of the completed invoices, so that I can easily detect where my attention is required
2. As a manager, I want to see statistics about specific employees, so that I can use this information to evaluate my employees.
3. As a manager, I want to be able to generate reports based on selected variables, so that I can more easily monitor the processes.
4. As a QA tester, I want to see an overview of completed invoices, so that I can test whether or not the process is completed.

5.7 When I approve an invoice, I want to assign it to another user, so that the next person in the approval flow can approve or reject it.

1. As an approver, I want to add a note to an invoice, so that I can include my reasoning for a decision.
2. As an approver, I want an easy way to assign it to the correct user, so that I don't have to remember who exactly I need to assign it to, or know all user id's by heart.
3. As a manager, I want to set up approval routes, so that a standard approval flow can be set-up for specific situations.

5.8 When I verify an invoice, I want to be able to edit fields, so that inconsistencies can be removed.

1. As an approver, I want to ensure that all required values are included when an invoice is verified, so that no uncomplete invoices enter the approval flow.
2. As an approver, I want to see the amount to be paid without tax, so that I can see the amount of money it costs after deducting tax.
3. As an approver, I want to automatically calculate the exchange rate, so that I can use up to date exchange rates when approving foreign invoices.

4. As an AP clerk, I want to be able to edit values, so that I can remove inconsistencies between the invoice and the imported values.
5. As an AP clerk, I want to edit the currency used, so that I can ensure that it matches the currency on the invoice
6. As an AP clerk, I want to be able to edit the amount per line of the invoice, so that wrong amounts can be replaced with the correct value
7. As an AP clerk, I want the gross amount to be checked against the sum of line amounts, so that I can't verify the invoice when the values don't match
8. As an AP clerk, I want to see which fields require additional attention after I edit a field, so that I can prevent errors.
9. As an AP clerk, I want to be able to mark an invoice as erroneous, So that I'm not forced to forward invoices to approvers that are not correct.
10. As an AP clerk, I want to be able to edit the tax rate, so that I can change this when the field is scanned wrong, or when an item has a different tax rate.
11. As an AP clerk, I want to be able to use a tax code, so that I can easily handle the tax use in an invoice by using the correct pre-set.
12. As an AP clerk, I want to edit the subsidiary, so that I can match it to the invoice
13. As an AP clerk, I want to edit the vendor name, so that I can match it to the invoice
14. As an AP clerk, I want to edit the vendor bank account, so that I can match it to the invoice
15. As an AP clerk, I want to edit the vendor email, so that I can match it to the invoice
16. As an AP clerk, I want to edit the purchase order ID, so that I can match it to the invoice
17. As an AP clerk, I want to edit the tax amount on an invoice, so that I can match it to the invoice.
18. As a manager, I want to customise which fields are verified, so that I can adapt the application to my business processes.
19. As an AP clerk, I want to see an overview of all invoices that I need to verify, so that I can keep track of my open tasks.

5.9 When I have to approve an invoice, I want to see the information I need to decide whether to approve the invoice or not, so that I can make a valid decision.

1. As an approver, I want to see when the gross amount is different than the sum of amounts, so that I know to check all amounts on the invoice.
2. As an approver, I want to be able to send a notification to the client, so that I can inform them that there is an issue with the invoice.
3. As an approver, I want to see the amount for the invoice, so that I can use it to decide if the invoice should be approved
4. As an approver, I want to see the amount without tax, so that I can see how much the cost is after deduction
5. As an approver, I want to see the tax amount, so that I can see how much is deductible
6. As an approver, I want to be able to see the original invoice file, so that I can reference it for more context information.
7. As an approver, I want to automatically calculate the exchange rate, so that I can use up to date exchange rates when approving foreign invoices.
8. As an approver, I want to be able to edit values, so that I can replace incorrect values with the correct value before I approve the invoice.

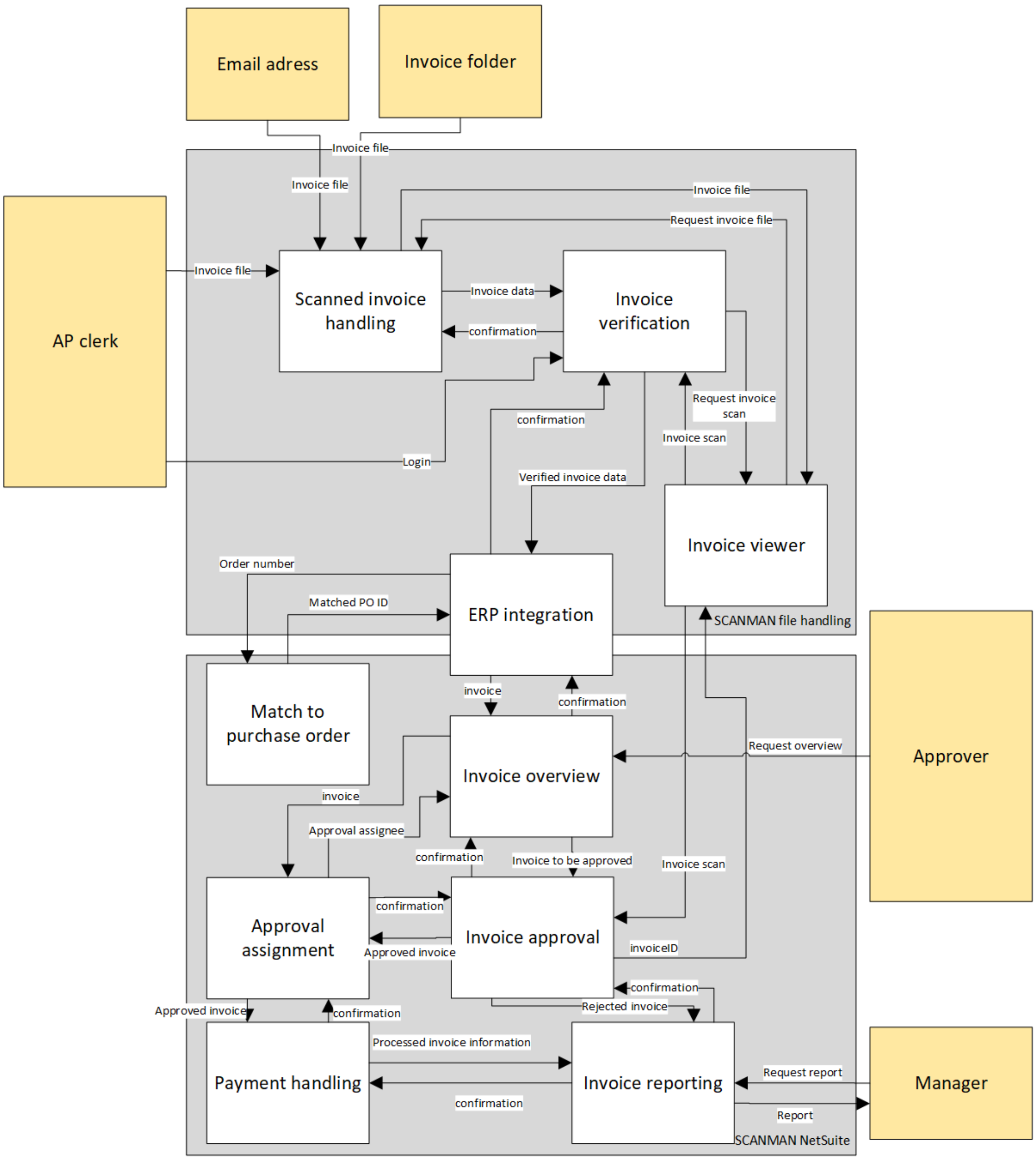
9. As a manager, I want to be able to undo data edits, so that previous versions can be restored.
10. As a manager, I want to customise the information fields that are shown, so that I can adapt the application to my business processes.

5.10 When an invoice is fully approved, I want the payment of the invoice to be processed, so that the accounts payable invoice handling process can be completed.

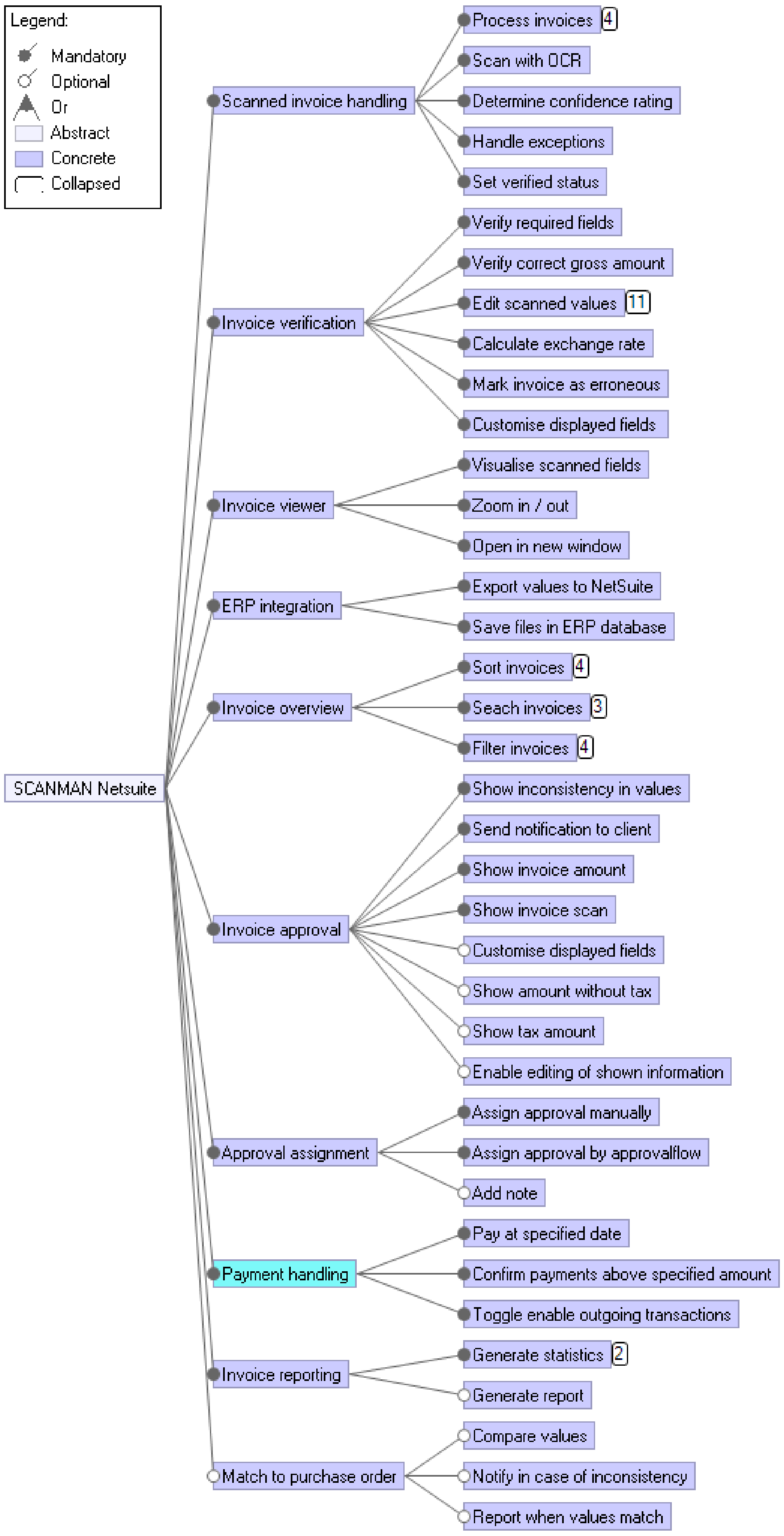
1. As a manager, I want to ensure that approved invoices are paid at a specific date, so that I can manage our monthly expenses
2. As a manager, I want to manually confirm the payment of invoices over a specified amount, so that I can personally keep up with big invoices.
3. As a QA tester, I want to disable the outgoing transactions, so that I can test the process without fully processing invoices.

6 Architecture:

Functional architecture diagram



Feature diagram


Legend:
Mandatory
Optional
Or
Abstract
Concrete
Collapsed
SCANMAN Netsuite
Scanned invoice handling
Process invoices 4
Scan with OCR
Determine confidence rating
Handle exceptions
Set verified status
Invoice verification
Verify required fields
Verify correct gross amount
Edit scanned values 11
Calculate exchange rate
Mark invoice as erroneous
Customise displayed fields
Invoice viewer
Visualise scanned fields
Zoom in / out
Open in new window
ERP integration
Export values to NetSuite
Save files in ERP database
Invoice overview
Sort invoices 4
Seach invoices 3
Filter invoices 4
Invoice approval
Show inconsistency in values
Send notification to client
Show invoice amount
Show invoice scan
Customise displayed fields
Show amount without tax
Show tax amount
Enable editing of shown information
Approval assignment
Assign approval manually
Assign approval by approvalflow
Add note
Payment handling
Pay at specified date
Confirm payments above specified amount
Toggle enable outgoing transactions
Invoice reporting
Generate statistics 2
Generate report
Match to purchase order
Compare values
Notify in case of inconsistency
Report when values match