Expertise
and insight
for the future

Taneli Virtanen

Comparison of Requirements Management Solutions for Medical Device Development

Metropolia University of Applied Sciences

Bachelor of Engineering

Mechanical Engineering

Bachelor's Thesis

17 March 2020

Author	Taneli Virtanen
Title	Comparison of Requirements Management Solutions in Medical Device Development
Number of Pages	43 pages
Date	17 March 2020
Degree	Bachelor of Engineering
Degree Programme	Mechanical Engineering
Professional Major	Machine Automation
Instructors	Johan Vepsäläinen, Senior V&V Engineer Heikki Paavilainen, Senior Lecturer

This Bachelor's thesis was commissioned by Thermo Fisher Scientific Oy and it aims to study and compare requirements management tools for their medical device development at their Vantaa location. Their medical device portfolio focuses on IVD-devices and medical laboratory automation products. The scope of this thesis was limited to comparing Polarion ALM to the current requirements management solution, evaluating capabilities of the software tools in the extensively regulated medical device industry and identifying possible process improvements.

The identified problems with the current methods and tools were compiled as a problem statement and possible mitigative solutions addressed in the comparisons, evaluating the capabilities of Polarion ALM as the replacement for IBM Rational DOORS. In addition, process improvements and possible changes enabled by adopting a new lifecycle management tool as the requirements management solution were identified and classified based on their impact on the current working processes.

In conclusion, Polarion ALM was found a capable tool and ready to implement to the current working processes relatively seamlessly while simultaneously addressing multiple problems identified with the currently used tools. In addition, the versatile nature of Polarion ALM and its revisioning, traceability and visualization features were identified to facilitate gradual implementation of process improvements towards a more agile and modern environment in software based medical device development.

Keywords	Requirements Management, Product Lifecycle, ALM, Polarion

Contents

List of Abbreviations

ALM	Application Lifecycle Management. Product lifecycle management for computer applications.
IVD	In-vitro diagnostics. Tests done with samples taken from the human body, such as tissue or blood.
R&D	Research and development.
A&A	Analyzers & Automation, a business unit of Thermo Fisher Scientific Clinical Diagnostics Division.
CDD	Clinical Diagnostics Division of Thermo Fisher Scientific.
FDA	United States Food and Drugs Administration
TFS	Thermo Fisher Scientific Oy. A legal entity in Finland.
PLM	Product Lifecycle Management. Software that connects people, processes, and data across the entire product lifecycle.
DHF	Design History File. Documentation containing or referencing to the design history of a finished medical device.
SDLC	Software Development Lifecycle. The development cycle of a software product.
V&V	Verification and Validation. Process of ensuring that a software product meets system requirements and its intended purpose.
API	Application programming interface. Set of functions allowing the creation of applications that access the features or data of an application or other service.
RM	Requirements management. Process of analyzing, tracing, prioritizing and documenting requirements.

ReqIF	XML file format used to transfer requirements and its metadata between software tools. Also abbreviated as RIF.
PDM	Product Data Management. Software tool used for managing product and project data.

1 Introduction

Modern product development in the global market requires unprecedented control of the development process to ensure product and process quality for all markets. Medical device development is a highly regulated field and imposes further controls for the development process in all products.

The modern global market has changed the way how products are developed in many high-technology companies. This is especially apparent in the medical field, including both in-vitro diagnostic (IVD) and non-IVD devices, where there are globally accepted industry standards in place, but regional regulations can impose specific requirements on medical devices sold in those markets.

This creates a need for systems that track, process and present information for all relevant persons in different phases of product development. The nature of requirements for a medical product can vary from regulatory expectations and industry standards to specific customers' needs and affect every phase of the product development process. These requirements are used as design input in different phases of development and create responsibilities for multiple parties involved.

The implementation of requirements into a complete product can only be carried out by a coordinated effort of a team of specialists from different disciplines. The development process must also be traceable and visible to responsible parties and stakeholders outside of the actual implementation team.

1.1 Thermo Fisher Scientific Oy

This thesis was commissioned by Thermo Fisher Scientific and it was done during employment as a Verification and Validation Engineer trainee at their Vantaa location.

Thermo Fisher Scientific is the world leader in serving science, with revenues of more than $24 billion and approximately 70,000 employees globally. The Vantaa location is a key R&D and manufacturing site for multiple divisions. The Vantaa site employs over 600 people in R&D and manufacturing with a strong emphasis in R&D and it is an FDA

registered site. This work was done for Analyzers & Automation business unit in Clinical Diagnostics Division, later referred to as A&A and CDD. (About Us, 2020; Finland, About Us, 2020)

The product development environment related to this thesis work produces a vast range of products used in automated medical laboratory practices. These can be interface modules for analyzers or sample handling products such as an automated entry, exit and storage solutions.

1.2 Objective and scope

The objective of this study for Thermo Fisher Scientific is to replace the current requirements management (RM) software IBM Rational Doors with a more modern and comprehensive solution that is better suited for medical device development. The decision to evaluate Polarion ALM for this use was done before this thesis was started. The goal for Thermo Fisher Scientific is to evaluate a new alternative for their current requirements management software, in order to enhance the entire development process of new products. The company's decision was to evaluate Polarion ALM by Siemens for this purpose

The objective of this study is to understand the current development and requirements management process and evaluate the suitability of Polarion ALM as a part of the process. In addition, possibilities to improve the working process by using the capabilities in Polarion ALM are considered and evaluated. Possible improvements can be related to the automation of processes and documentation, easier management of conformance with relevant regulation and good development practices. In addition, improvements in upwards visibility of development efforts to project management will be considered. This thesis is conducted with the objective of using the findings from this study as a basis for official evaluation and testing phase for the software. The evaluation and comparisons are used by TFS and the medical template provider to facilitate the onboarding process with Polarion ALM.

The scope of this thesis was limited to the pre-testing phase of the project. The pre-testing phase contains comparison of the functionalities and capabilities in the new software tool with the current working process. The formal evaluation for the software takes

place at a later. Due to this, the usability evaluation and possible process improvement possibilities found in this study will be considered for use by the persons responsible for the implementation of the new ALM software and the provider of the medical template for Polarion ALM.

This thesis is structured to provide background information regarding the intended use of a software product like Polarion ALM and the background information related to the medical device development and requirements engineering. In addition to general knowledge, the initial chapters create a baseline for the additional intricacies demanded by the medical aspect of device development regarding software lifecycle management (SDLC) and requirements engineering. These industry specific details in medical device development are critical to understand when comparing the features and possibilities of ALM software in requirements management and related processes in later chapters. Findings based on the study are concluded in the final chapter.

2 Medical Device Development

Management of devices, applications or processes that operate in a medical field naturally comes with increased responsibility, regulatory and otherwise. As a globally regulated market, the medical device industry is a prime example for needing tools to meet the increasingly complex regulatory expectations and adequate product quality. The areas of software development lifecycle most heavily impacted by industry specific process control are quality management, requirements management and the need for formal testing. As an example, International Organization for Standardization in its 2016 publication for medical device quality management systems ISO 13485:2016 states that *"design and development changes shall be reviewed, verified, validated and approved"* creating testing and documentation overhead not applicable for most industries. (International Organization for Standardization, 2016)

Classification of medical devices is used to group devices based on their intended use. These classifications are based on the potential risk or harm caused by the device. According to the European Medical Device Directive, medical devices are grouped into 4 classes (I, IIa, IIb and III). (Council Directive, 1993) The classification of a given medical device determines the mandatory design controls for the device manufacturer and

concludes the scope of deliverables to reach compliance. Although harmonized to a certain degree, the classifications differ between regulatory bodies and it is the device manufacturer's responsibility to set the correct class for their device in any market they wish to operate in.

Medical devices can be assumed to have a longer lifecycle than most consumer electronic devices. Medical laboratories in hospitals are a major market for medical devices and often operate under governmental agencies. When public funding is used for equipping those laboratories, the contracts are a result of competitive bidding process. Considering the applicable regulatory control in transferring medical devices to market, it is often more cost-effective to update the existing products than it is to bring new devices to market. The cost-effectiveness is emphasized in upgrading software focused products, where a new software version can improve performance and reliability. This extended lifespan of products and the underlying reasons for it makes the industry unique and creates a need for specialized tools.

A key area for organizations operating in this field is producing proof of compliance with the desired market's regulatory body. Even in the absence of a global administrative body for regulation, a majority of regional regulations are harmonized globally. One such effort is the medical device single audit program (MDSAP) process. MDSAP is intended to make the regulatory oversight less taxing by allowing a single audit to satisfy the regulatory requirements of the participating regulatory bodies. And therefore, it is essential to research every market you wish to enter with your medical device. This further emphasizes the importance of project planning in bringing a medical device to market. Proof of compliance is often determined by regular on-site auditing. Providing the required documents or deliverables on demand during audits is the norm when operating in the medical device industry. The importance of these documents makes them a mandatory part of nearly every project in a medical device development project. In practice, this proof of compliance drives the need for design control in all stages of product development. Therefore, the industry specific rules and regulations in developing medical devices create an external need for traceability. (Joachim Rossberg, 2014). For instance, in the United States Food and Drug Administration (FDA) Code of Federal Regulations Title 21 part 820 (21CFR820) Quality System Regulation, one notable regulatory expectation is maintaining a complete Design History File (DHF).

The DHF is a collection of documentation that contains or refers to documents that prove the device's design was developed in accordance to an approved plan and set of requirements. This process is visualized in Figure 1, where the rigorous review and approval process needed for upkeeping a complete DHF is apparent. This DHF must be maintained for each device produced. (Quality System Regulation, 2011)

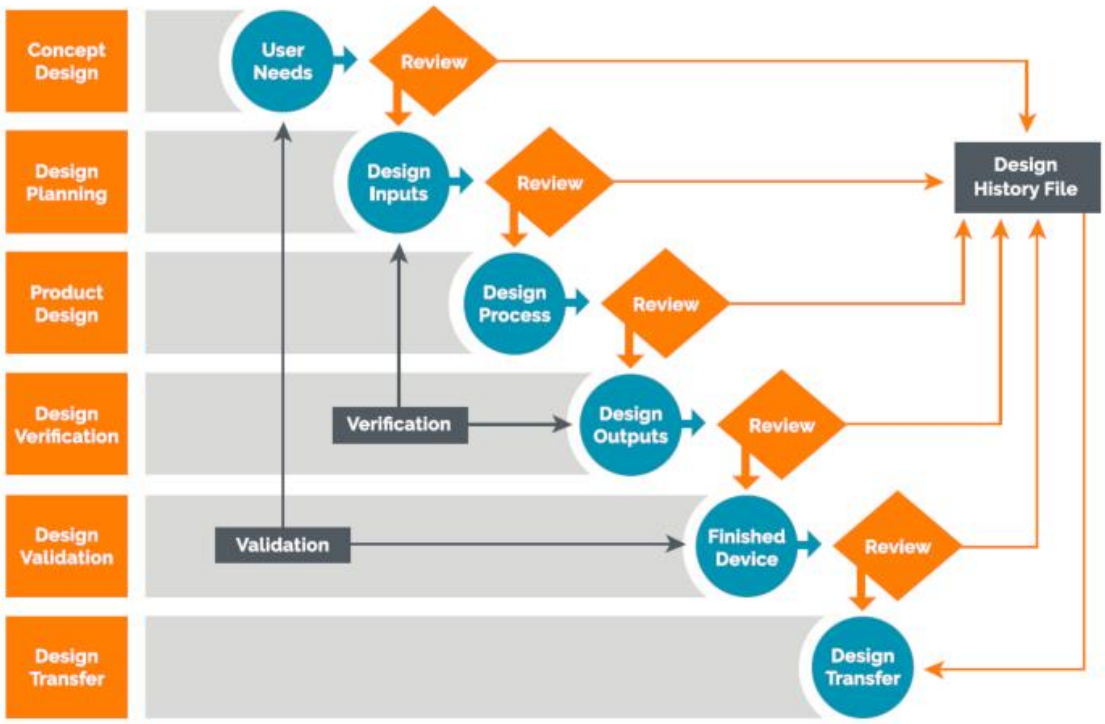

Figure 1 - DHF in development process (Altered from: Cognidox, 2020)

Based on this approach to product development, as required in medical device development, the upkeep of a DHF and constant auditing processes is a strenuous task. This degree of developmental control creates a need for product and project management tools.

3 Requirements Engineering

The most important part of a development effort, after the initial phases of concept and feasibility studies, is to achieve approval for entering development phase is to produce a set of design inputs for the project. These inputs are set as requirements for the product under development. Requirements originating from different stakeholders can impact aspects of the project on a system level, be specific to a single technical feature, or any other aspect of the final product such as documentation or certificates.

The origin of a requirement can be for example a need set by the end user of the product. This user need is then refined into one or more specific requirements to be implemented in the product. For example, if the product is to be operated by humans there is a need for it to be safe to operate. Safety related needs are often risk control for an identified risk. A user need like this can be refined into requirements for either, or both, mechanical and software safety features as well as documentation for safety instructions, labeling and marking of the final product.

The business itself can be another source of requirements. These can be related to product performance, compatibility with current intended use environment or anything else that drives to increase competitiveness of the product in the market. Technical limitations from existing products, current standards and applicable regulations also contribute to the collection of requirements.

Requirements define both functional and non-functional deliverables for every feature of the end product. Functional requirements (FR) can describe what features are in a product, what the product does and how the product behaves. The output of a FR can be a task for a robot to move an object from point A to point B, as well as limitations in doing so. Deciding how to achieve the transition is not a part of the requirements engineering process. However, it is a part of the development output provided by the development teams. Non-functional requirements (NFR) such as quality and safety requirements often come from applicable standards and regulations. An example of such a requirement can be mandatory labeling, marking of products or contents of the user manual.

These requirements are a basis for development activities in every project, and they are used to refine stakeholders' needs in a potential new system to implementable

functionalities. The requirements are used in multiple phases of development, which emphasizes the importance of well thought and refined requirements for systems. For example, the basis for planning, risk management, trade off, acceptance testing, and change control are derived from requirements. (Dick, Hull & Jackson, 2017)

This inevitably leads to the conclusion that in order to develop a product, the development teams must have access to good requirements. Good requirements are hard to define as they can differ between defining teams, but common characteristics for good requirements can be deduced from problems that have been known to occur due to bad requirements. The *Naming the Pain in Requirements Engineering* (NaPiRE) survey by Méndez Fernández et al. (2016) collected experiences of problems in requirements engineering. This survey was conducted in 228 companies and data related to the 10 most common RE problems faced is illustrated in Figure 2 with their relation to project failure.

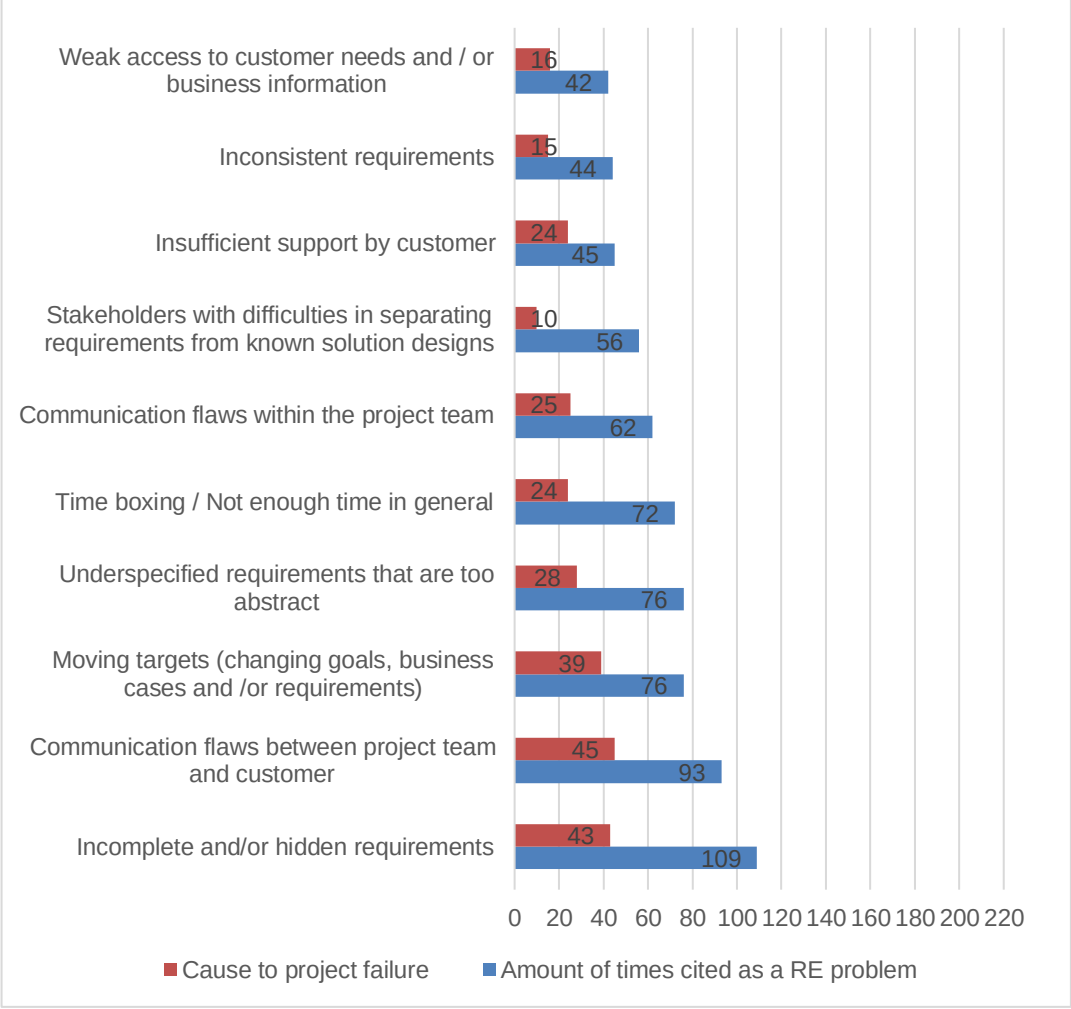

Figure 2 - Frequency of RE problems and their relation to project failure (Méndez Fernández et al., 2016)

As seen in the top 10 most common problems found in the study, problems with requirements were present in many surveyed companies. Using the data from the survey to generalize problems in the requirements management process, it can be seen that the most common problems in the requirements management process have similar characteristics. At least four of the top 10 are related to communication problems. Communication within the project team, between the team and the customer, insufficient support from the customer and weak access to the customer are all related to communication. Another common problematic characteristic to occur as the result of an ineffective requirements elicitation process. These problems related to the elicitation of requirements, incomplete and/or hidden, underspecified or too abstract and inconsistent requirements.

While not always the most common, the most critical problems causing project failure in the study seem to be customer and business related. The risk of project failure notably being over 50% with moving targets and insufficient support by the customer.

The data collected in the NaPiRE survey research (Méndez Fernández et al., 2016) can be used to conclude and evaluate the causes and impacts of the discovered RE problems. In Figure 3, the causes for the most common problem *Incomplete and/or hidden requirements*, observed in 48% of total companies surveyed, are presented in a probabilistic cause-effect diagram. The causes are divided into groups based on their origin and the closer they are to the horizontal arrow the more commonly they were reported in the survey results. From interpreting the cause-effect diagram problems with RE process leading to the most common problem of incomplete or hidden requirements are divided by their origin: Input, People and Method in the RE process. The identified problems originating from project inputs are often related to the customer's ineffective communication or processes. These problems highlight the importance of customer relations and can be alleviated with clear communication and a higher level of co-operation between the development teams and the end customer. People-related issues that are identified in the Figure 3 can be difficult in nature, as the most common problems are related to lack of experience and weak qualifications. These issues are most effectively addressed on a project team level by providing support for inexperienced employees and implementing peer-reviews to processes. Method-related problems impacting the state of requirements in a negative way are causes of inefficient processes. These can be alleviated with implementing review and knowledge sharing processes that involve the development team, people responsible for the RE process and the stakeholders.

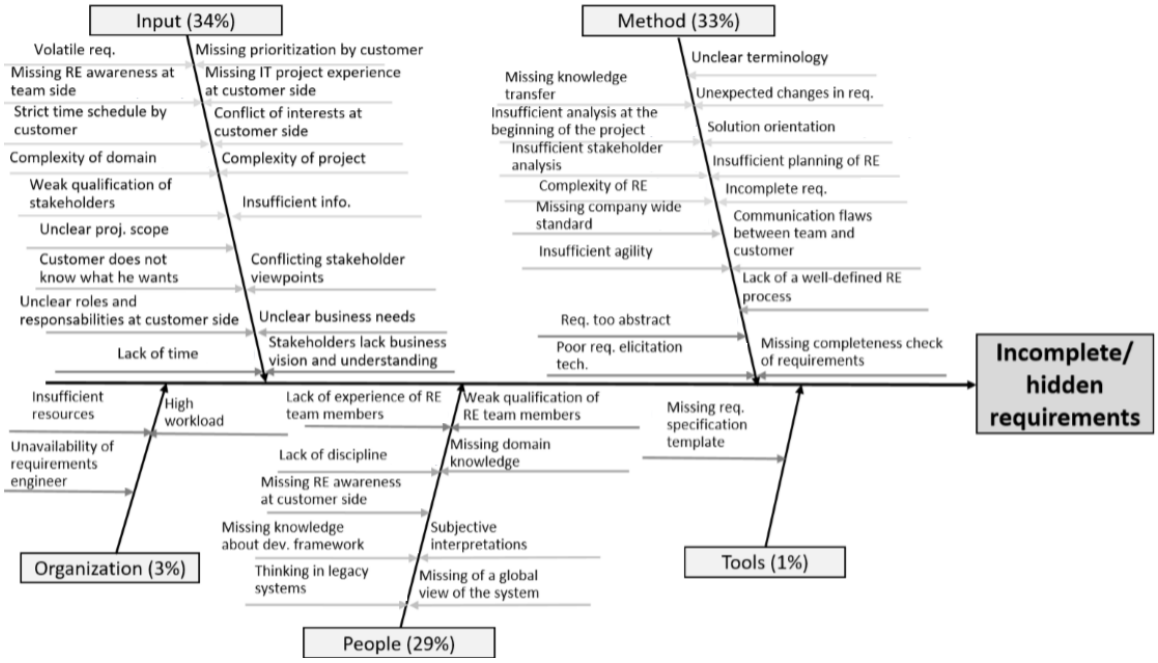

Figure 3 - Probabilistic cause-effect diagram for *Incomplete and/or hidden requirements* (Altered from: Méndez Fernández et al., 2016)

Using the findings from this research, it can be said that the RE process requires heavy involvement of both stakeholders and development teams to elicit clear and nonvolatile requirements essential for product development, and achieving the capability to produce them is a complex task for an organization.

3.1 Requirements in medical device development

Management of requirements in the medical field introduces mandatory deliverables concerning documentation that are not present in other industries. A survey conducted by Denger et al. in organizations that develop software for medical devices and medical information systems concluded that 63% of the companies surveyed felt that the requirements engineering activities caused most of the problems related to software. (Denger, Feldmann, Höst, Lindholm & Shull, 2007)

As stated in General Principles of Software Validation (GSPV), published by the FDA, documented requirements provide a baseline for verification and validation (V&V) of the software. (General Principles of Software Validation, 2002) Verification and validation are terms that have different meanings outside of the medical field, but in medical device development verification and validation are clearly defined terms. V&V activities are used

to ensure product quality by proven confidence in the device performance and features. Verification ensures that design inputs, i.e. requirements, are met by the design outputs, i.e. the product and its implementation. The verification process produces actions to formally test the implemented features in a medical device as well as the accompanied documentation. In practice, this means that requirements are implemented into features of the product, and the functionality of the implementation is formally tested and documented. Validation ensures that the product, with a certain level of confidence, conforms to its intended use in its intended environment and the user needs set for it. (ISO, 2016; Quality System Regulation, 2011)

An important source for requirements is from mitigative actions to identified risks. Failure mode and effect analysis (FMEA) and residual risk management, as visualized in Figure 4, are essential to remove and mitigate possible risks in the operation of a medical device. The FMEA process aims to analyze the root causes for risks, in attempt to find mitigations that are formulated into requirements to be implemented. In addition to risks concerning patient health or safety, risk categorization can focus on patient safety, operator safety, and patient security with regard to personally identifiable information and its misuse. According to ISO 14971:2019 sections 4.4 and 4.5, risk management is a planned action that produces a risk management file that is maintained throughout the product's lifespan. Risk management produces requirements that act as mitigations for risks identified for the device. As stated in ISO 13485:2016 section 7.3.3, the outputs of risk management, i.e. the mitigations for the said risks, are to be used as development inputs and documented as such. (ISO, 2016; ISO, 2019)

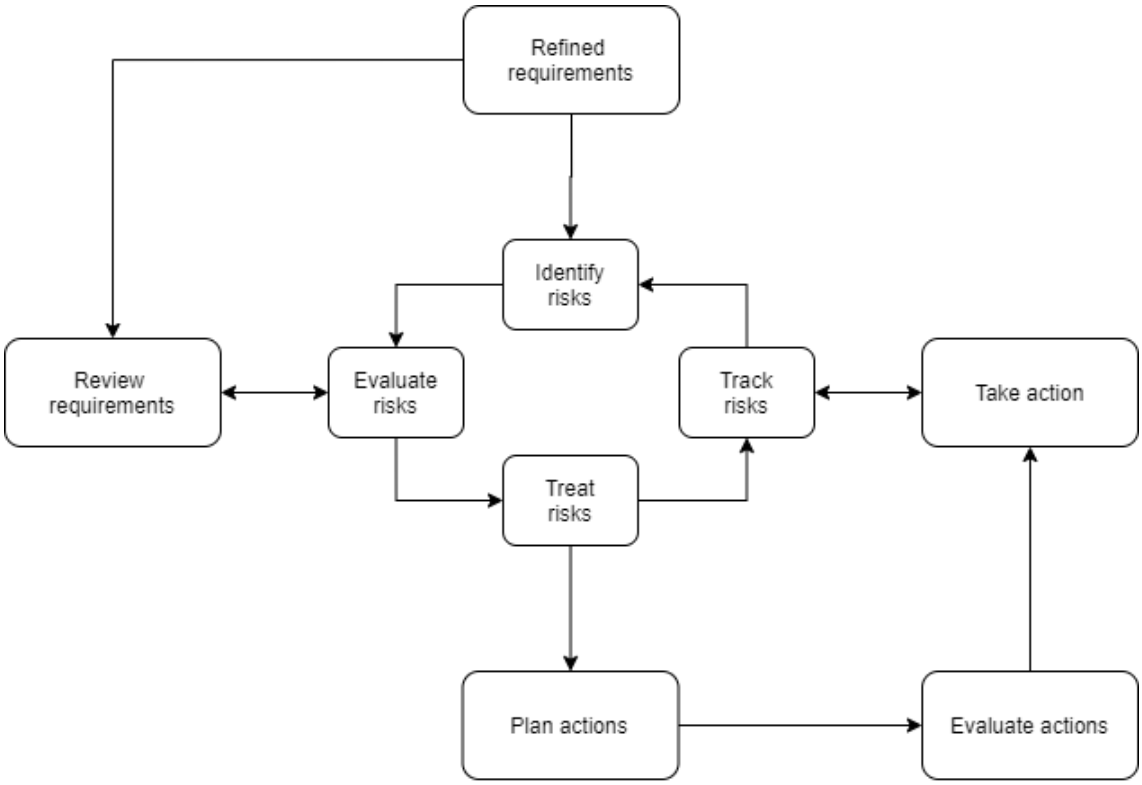

Figure 4 - Risk management cycle (Altered from: Doernemann, 2002)

Traceability of requirements becomes an integral part of the documentation process due to the regulatory nature of the medical field introduced in previous sections. The concept of traceability is visualized in Figure 5. The traceability analysis in the FDA document Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices (2005) is defined as following: *"Traceability among requirements, specifications, identified hazards and mitigations, and Verification and Validation testing."* This means in practice, that the device manufacturer has the ability to provide proof of V&V testing activities for the identified development artefacts with trace to software development status, i.e. software version, date and time and acceptance of the test result. This is carried out to ensure that the device manufacturer has proven confidence in the device's ability to operate as intended. (Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices, 2005)

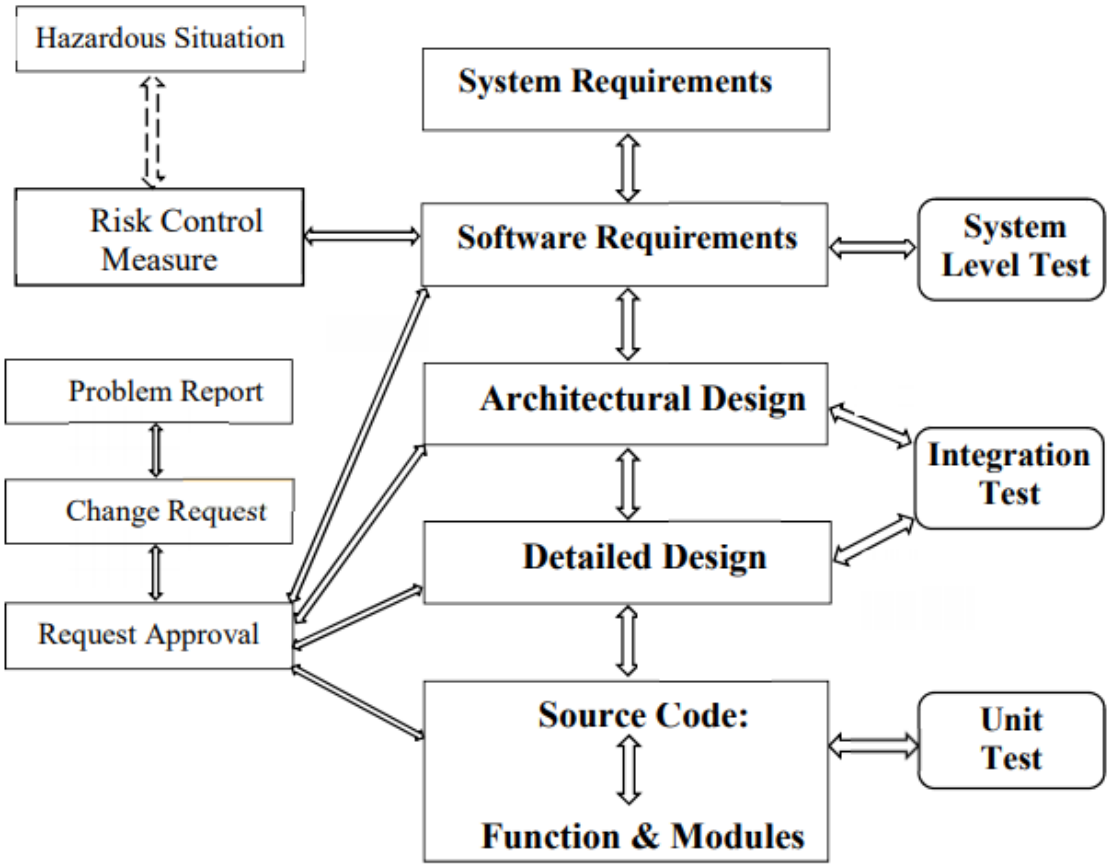

Figure 5 - Requirements for traceability defined in International Standards for medical devices (Altered from: Regan, Mc Caffery, Mc Daid & Flood, 2013)

4 Lifecycle Management

The lifecycle of a product contains all phases of development from initial concepts and feasibility studies to post-release maintenance and end-of-life activities. These phases contain activities that require and produce different types of data. For this data to be available to the people responsible for each phase in the product's lifecycle, they are often collected to a lifecycle management tool that is used as a data repository. As the type of data required for e.g. launching a product to market is different from the data required for evaluating which motors to use in the product, the capabilities of a lifecycle management tool are used to centralize and distribute the data. In addition, versioning and control of project documentation for all phases of its lifecycle are often conducted in such a tool.

Being able to deliver timely and complete products is crucial in a highly competitive and global marketplace. This is can be supported by tools that aid in the management of complex and, with increasing frequency, geographically dispersed development processes. These lifecycle management practices have been widely implemented with Product Lifecycle Management (PLM) tools, which are offered by multiple vendors. However, most PLM solutions are intended solely for product data management. A product data management tool can be useful, but for development of complex software focused products its functionalities are insufficient. In an agile development environment with cross-functional teams, the flow of information has to be constant, readily modifiable and horizontal, thus a more comprehensive tool is needed. (Gao, Gao, Fan & Zheng, 2009)

The concept of ALM is described by Kääriäinen and Välimäki as the coordination of activities and the management of artefacts during a SW product lifecycle, such as requirements, designs, source code, development builds, documents, etc. (Kääriäinen & Välimäki, 2008). The objective of such tools, visualized in Figure 6, is to centralize project information, provide visibility between project management and development teams, and enhance management capabilities for complex development processes. Compared to PLM, which is often used as a centralized location for project documentation and design files, ALM solution offers a more agile solution with real time linking of project artefacts like requirements to their code, testing or risk management related artefacts within the project. This gives the teams working in product development an easier access to specific artefacts and work objects to help the implementation of features.

In return, the upwards visibility provided by an ALM solution with real time development metrics and traceability is hard to achieve using PLM systems. Integration between the ALM and commonly used development tools such as Microsoft's Azure platform can be achieved using application programming interfaces (API), allowing for data mapping and creation of individual artefacts, as well as automation and control of working processes within a project. To summarize the comparison, ALM has been created to cater to the needs of agile development processes most common in software development, where PLM systems face difficulties adapting to rapidly evolving workflow.

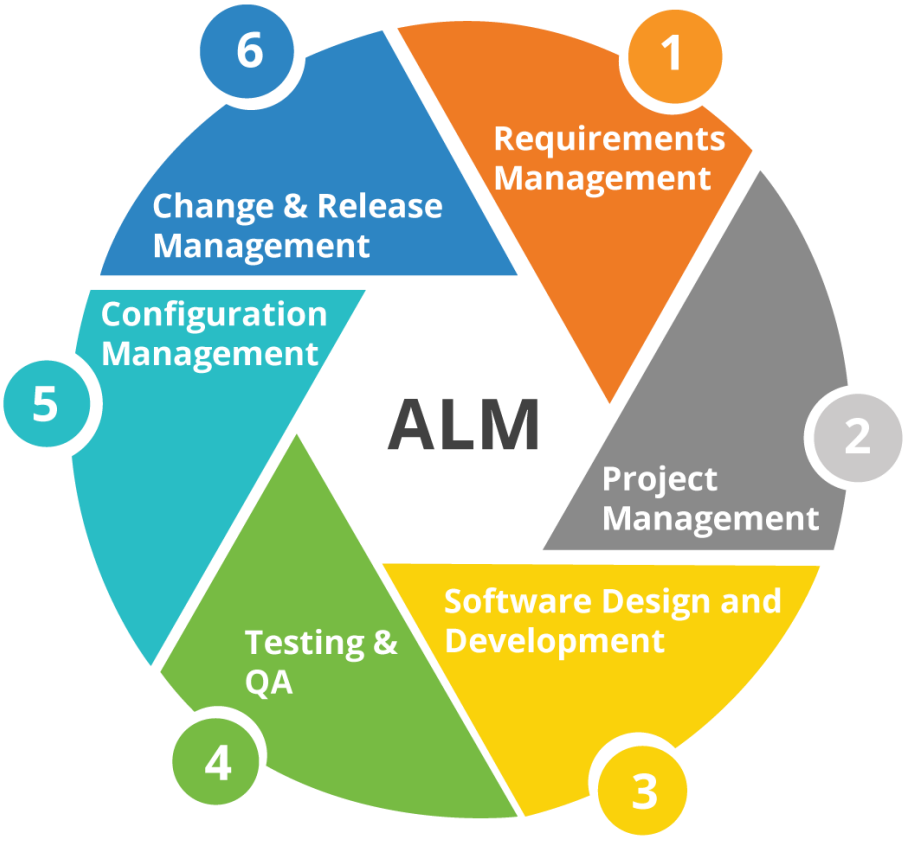

Figure 6 - Visualization of ALM tool function (Microgenesis Application Lifecycle Management, 2020)

Kääriäinen and Välimäki adopted an ALM framework that defines the key elements of ALM on software development. (Kääriäinen & Välimäki, 2008)

These are as described:

- Creation and management of project artefacts

- Traceability of lifecycle artefacts
- Reporting of lifecycle artefacts
- Process automation and tool integration

Another important benefit of an ALM solution is the possibility to provide real time visibility of the project development progress to management by using the data from connected artefacts to provide project metrics. Having access to relevant metrics with less overhead caused by following multiple project development tools, enables more effective and accurate planning by the project management. Planning activities in complex projects can be hard to complete accurately without data from previous or current similar projects, which are readily available when using an ALM tool. In addition, an ALM solution facilitates agile development by facilitating downwards flow of information. The development teams can benefit from having access to changing project goals or needs in real time, thus being able to react and plan accordingly.

5 Comparison of Development Lifecycle Management Tools

The comparison in this thesis work is conducted between Polarion ALM and the currently used tools: IBM Rational DOORS for risk, requirements and test management and Oracle Agile PLM software used for documentation, distribution and maintaining the DHF for medical products. The following focus points are used to define the scope of the comparison to the most relevant features where Polarion ALM is expected to perform better than the current solution.

Development control and project management

Management of development, requirements engineering, evaluation of risks and producing high level documentation are key aspects of a centralized solution for lifecycle management. This focus point aims to compare the capabilities in terms of development control, visibility of development progress and project management.

Medical device development

This focus point aims to evaluate the capability of Polarion to produce deliverables that can be used to show compliance with applicable regulations. Medical device development emphasizes some aspects of development differently from other industries. These points include comparisons focusing on the review and approval process, traceability, auditing, version control and the upkeep of DHF.

Agile development practices

This focus point concerns the suitability of Polarion ALM for supporting agile development practices, where development teams take responsibility for adhering to collaboratively decided working methods. Comparisons will be performed in the of ease of use, compatibility, intuitiveness of use, compatibility with current development tools such as Microsoft Azure DevOps platform and the current tools used for testing and low-level documentation. In addition, usability of the tools is evaluated for working with multiple projects and products.

5.1 Current methods and problem statement

The currently used processes include a collection of requirements on a header level to a module in DOORS. Creation of links between objects in DOORS requires a link module that contains information about the links made between modules that is subdivided into linksets. Link modules can be used to restrict link target and source relations by specifying allowed in and out modules. The elicited requirements are later exported from DOORS to text documents, which are approved and revised in a separate PLM tool that stores the DHF. Outside of DOORS, the development activities are conducted in Microsoft Azure DevOps platform, which has no connection to DOORS or the PLM tool.

A requirements management DOORS module contains in-links from risks and out-links to formal testing modules. A risk management worksheet (RMW) for a project exists in a separate DOORS module with identified and categorized risks, their mitigations and other information including risk severity and likelihood estimates. Risks and their mitigative actions from RMW are linked to the requirements. Requirements are linked to test modules containing test cases for product verification. For an existing product, DOORS module for version history changelog archiving with regression analysis is used to create

a trace between changes in software versions and impacted formal testing. All DOORS modules are exported to text documents in each revision and approval process for project deliverables exists entirely in the PLM tool.

The current requirements management process requires the use of multiple different tools that are not ideal for agile development needs. DOORS, the currently used requirements management tool, does not provide sufficient capabilities for the company's needs. An example of the currently used link and requirement process structure in DOORS is presented visually in Figure 7. The requirements derived from regulations, risk mitigations and other stakeholders grow as the desired control over medical device development grows more complex. This increasing complexity with requirements management in new products and updating the existing products derives from changing methods of storing requirements and their relations to other artefacts. In addition, creating new requirements modules for software changes cause the updated product to lose the connection with previous versions and therefore, the updated products requirements are not representative of the complete product. Constructing a trace between all product iterations or different products using DOORS requires a convoluted linking structure that causes a heavy overhead to manage, in addition to being hard to navigate.

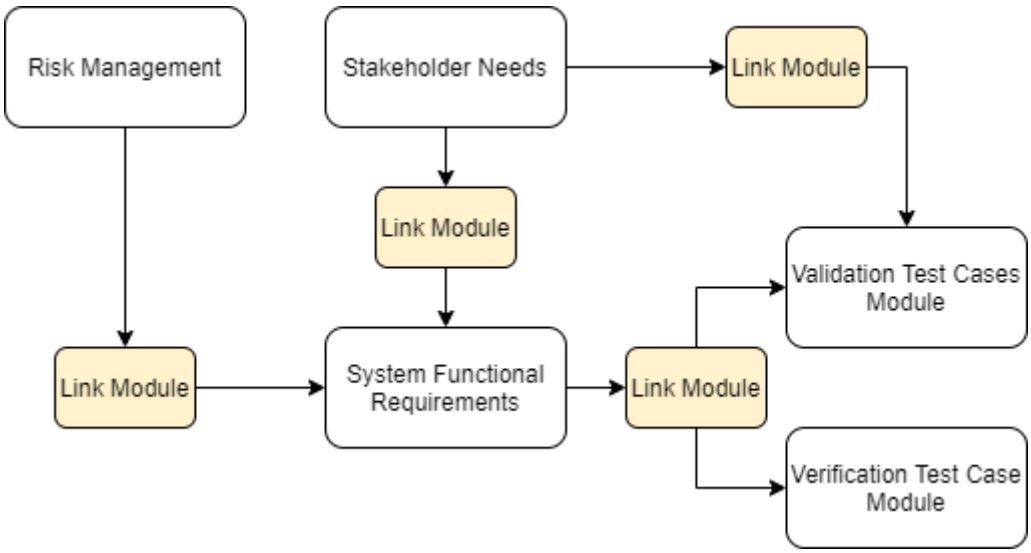

Figure 7 - Example of a DOORS link structure

Currently the requirements are on a singular hierarchical level in the requirements management tool, independent of their origin. The origin and source for stakeholder needs is not always present in the same software tool currently. This can disconnect developers

from seeing the complete picture and cause difficulties with implementing the stakeholder's needs. Even if the source for requirement, such as a risk that it mitigates, is present in the tool, retrieving the data requires specific attributes and configurations in the source module and the destination module. As the data is visible in separate modules, there is a continuous need to switch between them when e.g. a test case description is being drafted. Defects, such as a test failing due to a bug or a misunderstood requirement are currently handled on a different platform from requirements making their relation invisible within DOORS. Linking of defects to requirements is done in defect reports outside of the tool used for controlling requirements, which makes the process of retrieving related data impossible through links.

As development and planning are carried out within Azure DevOps, independently from requirements management in DOORS, the linking of requirements to development actions is not complete or clearly visible. Handling of traceability is carried out with linking of objects in DOORS and retrieving link data from multiple link modules, containing separate linksets that connect specific parts of the trace, which is completely independent from the development environment. Whereas the linking process is fairly straight forward, the modules holding the said requirements vary greatly in their style of content. The templates used to create DOORS modules hold different attributes and they are not interchangeable without losing the functionality of the module. The differences in DOORS modules and linking them to tests, risks, changes and functional descriptions differs between projects thus making the process unnecessarily complex and convoluted. Suspecting links, where the link is suspected after one of the linked objects is changed, does not provide enough or relevant information regarding the change. In addition, the objects in a DOORS module cannot be transferred between modules without having to manually recompile the trace with a new set of links.

The current working process from user needs or risks to functioning and tested software release requires the use of multiple tools. The current workflow with project deliverables is presented in Figure 8. The user need is refined into requirements in DOORS and exported to PLM for approval once the requirements are finalized. The requirements are then refined into implementable features, implemented and integration testing is performed. Simultaneously formal test cases are written and approved for each implemented requirement. Once approval for formal verification is reached, the approved test case is executed and reported. Validation and identified regression testing are performed

after all implemented requirements have been verified and finally the software is released in combination with its accompanying documentation.

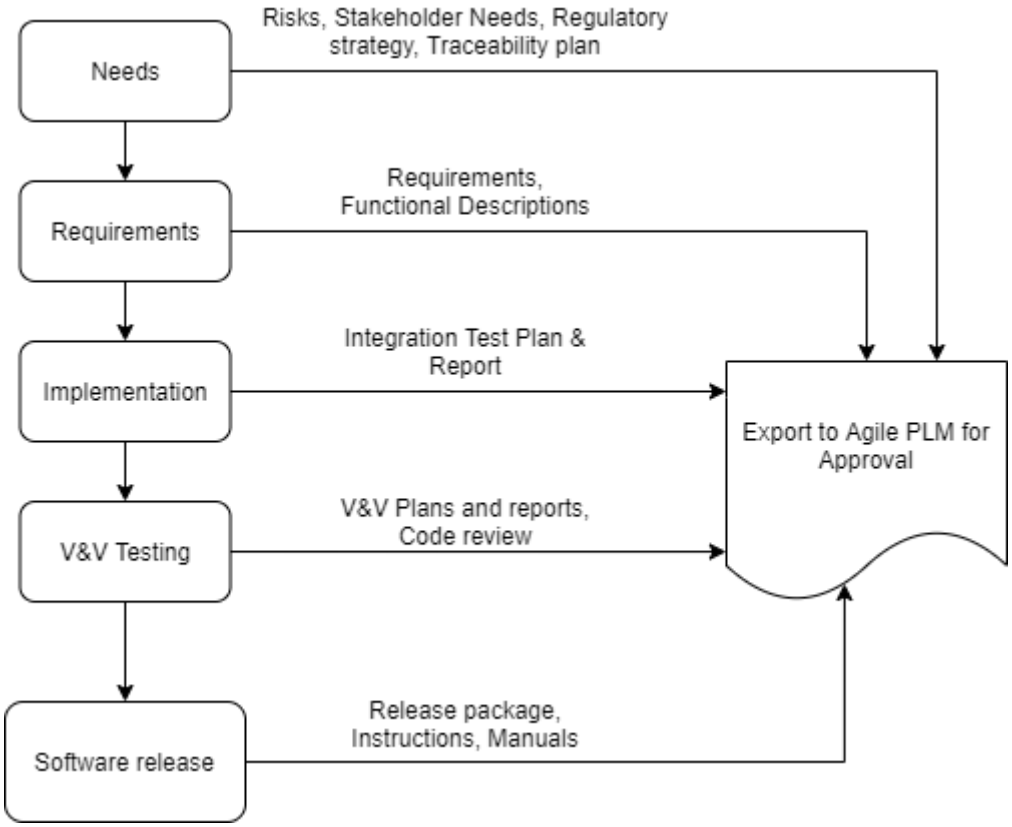

Figure 8 - Current working process and common project deliverables

DOORS supports a very shallow versioning capabilities resulting often uncertainty in the completeness of requirements during development, unless retrieved from the external PLM solution. The approval of requirements is done in a singular approval effort where they are exported to a separate PLM solution. This creates a situation where changes to requirements during development of the functionality, or other reasons requiring a new revision of approved requirements, creates uncertainty within and between implementation teams as DOORS does not inform stakeholders changed requirement clearly. The views inside of a DOORS module are made for specific purposes and a complete view is too cluttered for intuitive use, often hiding relevant information or making it harder to find. In addition, the table style formatting of the modules and objects drives a thinking process that is not desirable, where a filled table cell is thought to represent a completed step.

Object editing in DOORS is severely limited: The layout is unintuitive, and the balance between relevant information being shown in a single view and diminishing text visibility as the number of columns increases is difficult to manage. Even with the table format, exporting data out of the tool does not meet the needs and often requires manual editing before use in formal documentation. Importing data to DOORS from external worksheets or documents is seen to require more effort than its worth, creating a situation where manual data entry is the more convenient solution. Module objects do not support the desired peer-review process, there is no visible history of the reviewing or modification activities and the version control does not support baselines of individual objects inside the module. Furthermore, as DOORS is module based, the cross-module or project level search query functions become important when searching for e.g. regression impact in test case modules. The folder search query capabilities are not ideal for this purpose, as opening the module where your search input is found only points to the first appearance. Finding all relevant objects is tedious and requires constant use of separate modules.

The reusability of DOORS objects is poor, and there is no functionality to branch requirements that apply to multiple products and projects, such as those of regulatory origin, while keeping their traceability intact. This lack of reusability causes a workload of duplicating information, which in nature is not productive. Creating new modules, even out of templates, in a changing development environment creates problems where some attributes might be lacking. The impact of attributes is not clear or visible for the user and lack of a specific one might be hard to find, even though its impact in the module functionality is critical. DOORS modules are always only individually editable, and the software does not have the capability of tracking changes made by multiple people, thus locking others out while a specific object is being edited.

The DOORS application itself has been identified to be problematic. First of all, the login information is currently not synchronized with the rest of the corporate sign-on service, which was found problematic. Secondly, the application can shut down without prior warning to the user, leading to the loss of any un-saved data from the session. This can occur when changing between different wireless or wired networks while moving can lead to a loss of unsaved data as the connection to a license server is interrupted momentarily. Additionally, as there are a limited number of floating licenses shared between the entire R&D department, this can lead to temporary lack of access to important information.

It is clear that the issue with the current tools is that they do not facilitate effective data-flow or workload, and in some cases even force the user to perform unnecessary extra steps to fulfill their tasks.

5.2 Polarion ALM

Siemens Polarion ALM, previously developed by Polarion Software, was the company's choice of requirements management software to replace IBM Rational DOORS. Polarion is a complete ALM solution that integrates development implementation, requirements management, testing and provides agile methods of producing documentation. As seen in Figure 9, Polarion software incorporates functionalities for all aspects of software product development. The three key aspects of Polarion ALM according to the software provider are collaboration, traceability and re-use. These are all key aspects of agile development. (Polarion ALM Factsheet, 2017)

Figure 9 - Polarion ALM key features (Polarion ALM Factsheet, 2017)

Polarion provides a single repository of data, from which it is possible to retrieve specific data to product lines, products and projects. The accessibility of data is controlled through a role-based hierarchy, where information can be set available and modifiable for only relevant people in a specific project. Polarion provides version control and

traceability in its LiveDoc format, where documents can be modified simultaneously by multiple people, and all changes are recorded in the version history. In addition, Polarion software supports extensions made by both Polarion developers and community contributors. These extensions range in functionality from templates and workflow tools to connectors used to synchronize information between development tools.

5.3 Requirements management

Comparison of requirements management capabilities in Polarion to the currently used processes and tools is based on the identified issues with current tools and processes presented in the problem statement. Comparisons address the issues and present alternative or improved ways of working, using the capabilities of Polarion.

Polarion requirements documents that hold work item objects are in their proprietary LiveDocs format, meaning that they are simultaneously editable by multiple people and the user can see the final document in real time. In DOORS documents can only be edited simultaneously by setting individual objects as shareable, which is documented to significantly affect system performance negatively as module size increases. Unlike Polarion, DOORS does not offer methods to edit a single object with visibility of its linked items within a single view. The document text fields are all treated as individual objects, which makes this work tree format, seen in Figure 10, and other views possible and easy to read.

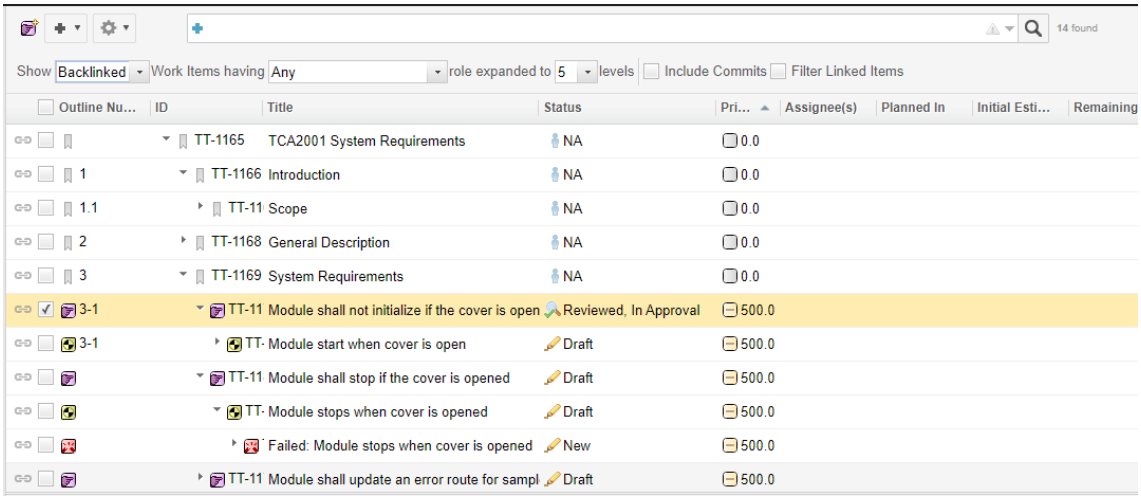

Figure 10 – Example of a requirements document in work tree format

The worksheet objects and their linked or backlinked objects are shown with unique IDs, status, priority and other chosen attributes. As the linked objects are available and editable on the same document, it is easier to grasp the impact of the requirement and its origin while working with it. Being able to see statuses of connected work objects, their approval and review comments and original source with possible attachments the requirements management effort is made considerably more seamless in Polarion.

The visibility of the requirements management process can be achieved using widgets that can be embedded on the home page for all or select users. In Figure 11, two widgets representing user needs and system requirements by status can be used to transfer knowledge of progress to project and product management in real time. Comparing this functionality to the current process where the progress can be invisible before exporting data for approval it is clear that this data presentation can be leveraged on a managemental level to react faster to project progress.

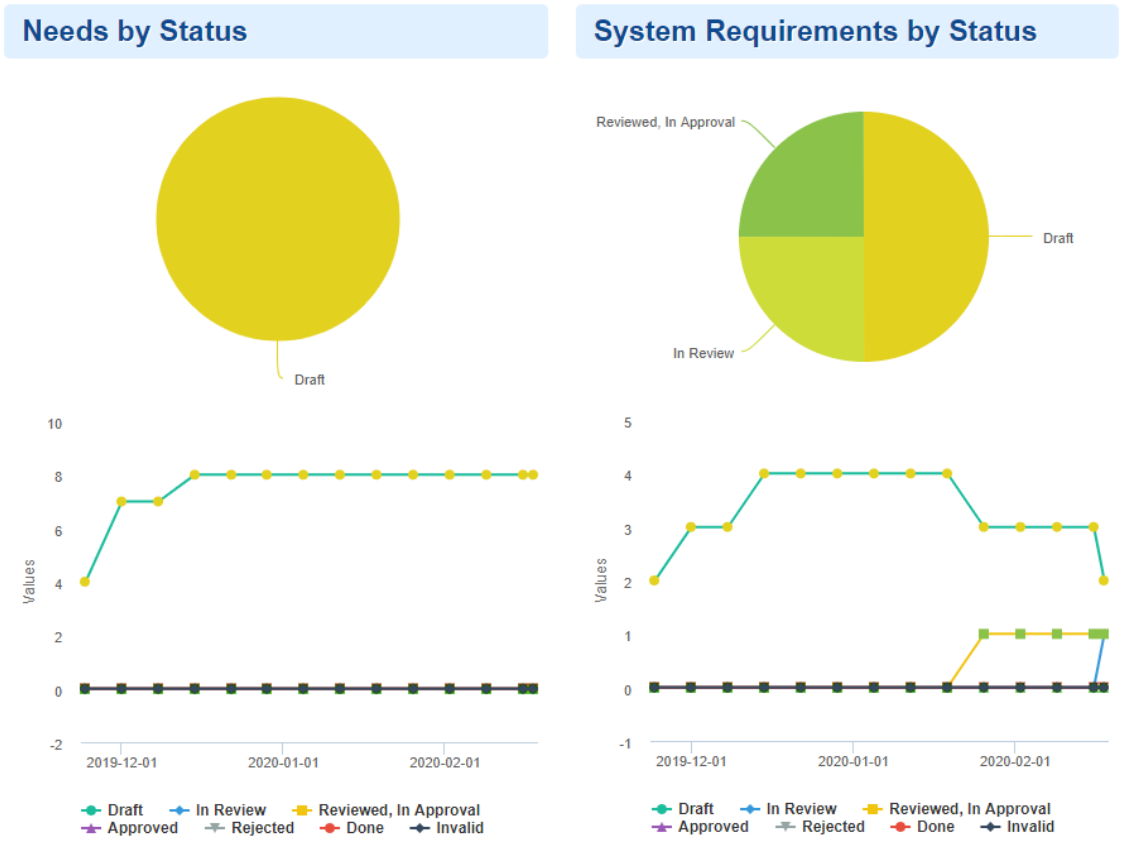

Figure 11 - Needs and requirements visualization in Polarion

The creation and configuration support of requirements in Polarion includes all of the currently used features in DOORS requirements modules. In addition, Polarion has a more comprehensive and highly modifiable list of attributes for requirements management, some of which are seen in Figure 12. These additional features consist of an original source, comprehensive and visible linked work item view, description field supporting rich text format, approval and comments fields and capability to attach documents or other requirement specific information directly to the requirement work item. The linked work items are visible in the requirement title in addition to their own field, providing a better glance-value of their impact and origin.

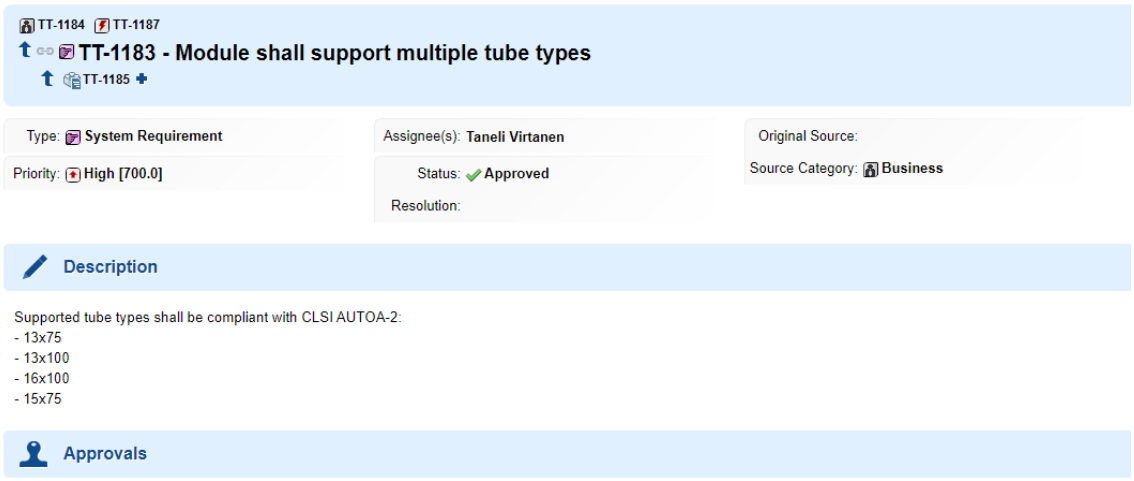

Figure 12 - Example of a Polarion requirement configuration

Polarion offers smoother transition between requirements as they are not stored in modules that have to be opened and locked for editing separately. This accelerates and consolidates the process of eliciting requirements by providing easier access to all relevant details. In addition, retroactively modifying requirements based on, for example, changes in user needs or regulations is easier due to readily available impact of the change.

The review and approval processes are not currently possible in DOORS requirements modules due to limited versioning and lack of electronic signatures, required by regulatory agencies for the upkeep of the DHF. In Polarion, each object can be reviewed and approved independently, with electronic signatures for object approval. This can be included to the process to reduce exporting of data and making the process more flexible, as work item-based approvals can facilitate starting of implementation for the highest priority features. The Polarion approval process is visualized in Figure 13. The process updates the document history with a new revision after every saved modification, keeping a complete history for each object. This combined with the electronic signature -verified approval method within Polarion can be leveraged to reduce external documentation during the development process while ensuring compliance with applicable regulation and best work practices.

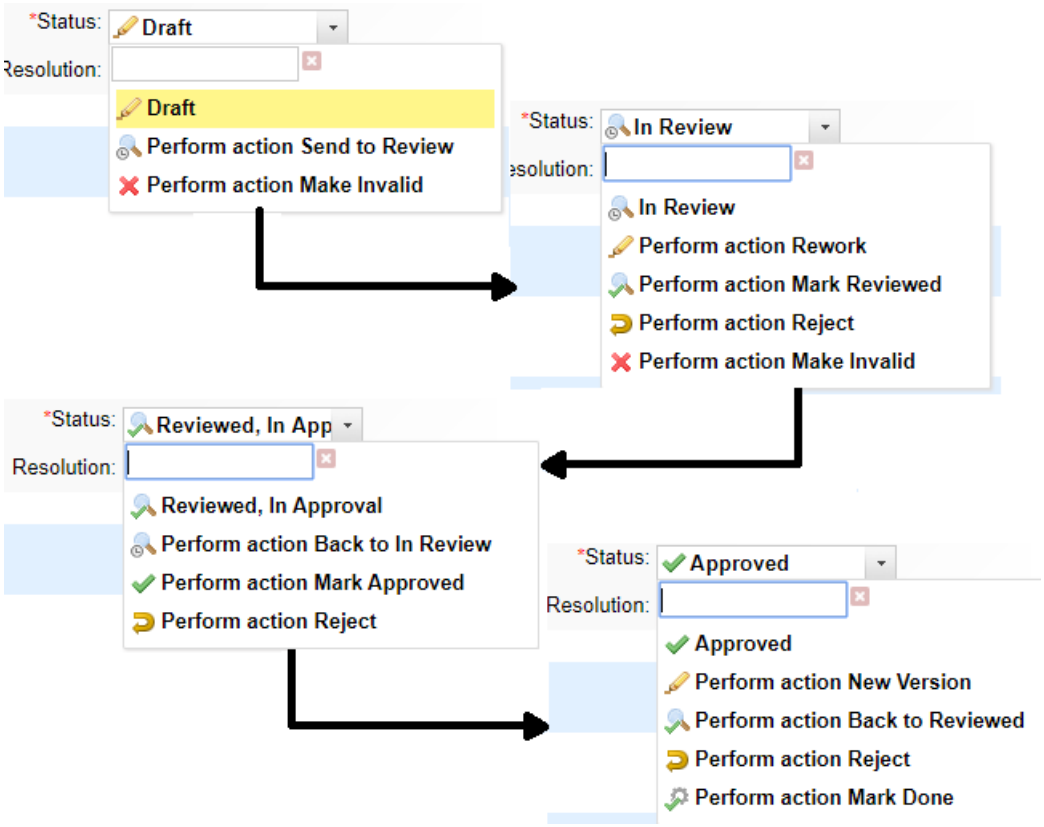

Figure 13 - Example of a Polarion approval process flow

Work items in Polarion can have assignees that can be used to clearly indicate responsibility and make it easier for developers to identify whom to contact in case the requirement is unclear or not refined. In conclusion, features in Polarion compare favorably to DOORS in addressing multiple problems identified and presented in the problem statement.

In addition to comparing advantageously to DOORS features in requirements management, Polarion is comparatively much more intuitive to use. This intuitiveness is based on user a friendly interface and browser-based approach that addresses issues found with a separate client.

5.4 Traceability

When comparing traceability capabilities in DOORS and Polarion, the act of linking to and from project artefacts is a significant factor to consider. In DOORS, the linking is done by either creating links inside a link module or by opening two modules and dragging one object to the other. A link module and a linkset is automatically generated if one

does not exist and, as seen in Figure 14, the linkset contents are presented in a matrix, where the links can be created as well.

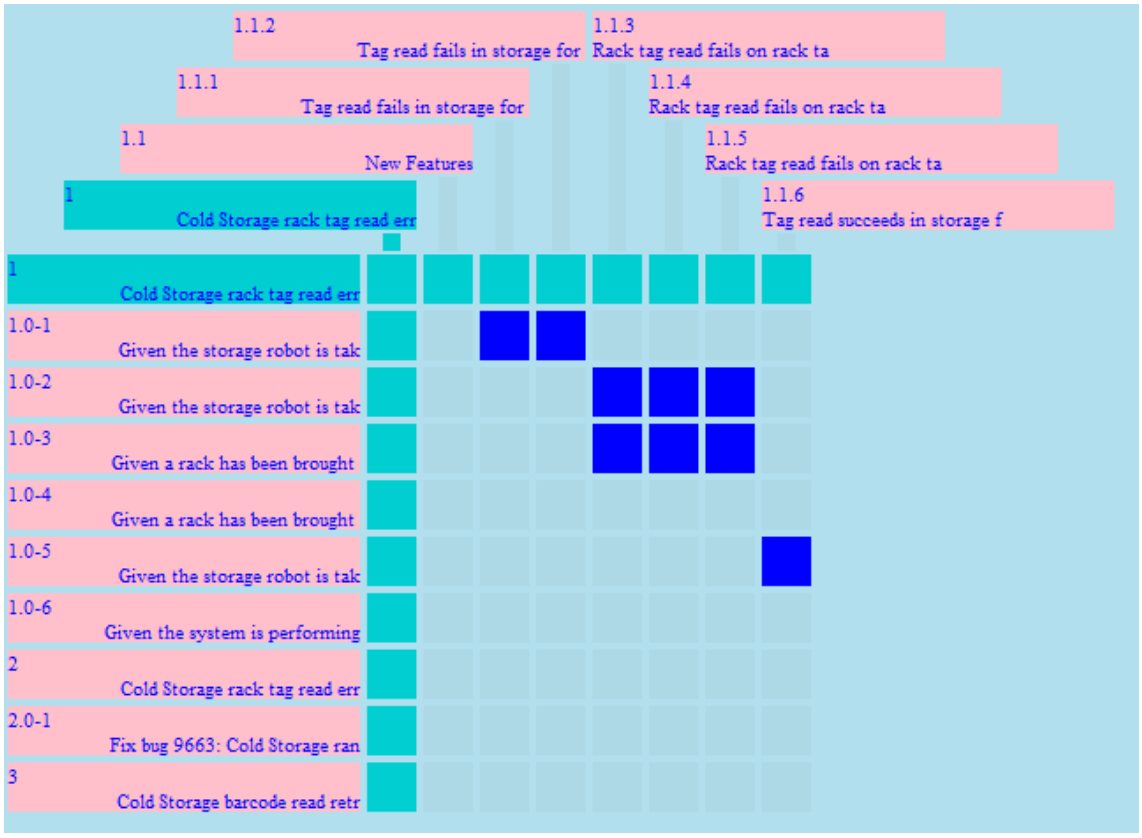

Figure 14 - Example of DOORS Linkset contents

In Polarion, linking can be managed directly from the work item itself. Work items share a ”Linked Work Items” section, where linked work items can be chosen on a project or a repository level. This allows linking objects across products and project, if found useful. The role of the link and suspecting of the link can be chosen for each link. In addition, the template can be configured to automatically suspect all created links in the document. Figure 15 presents the end user view of creating and managing links. In addition to creating and suspecting links, the view shows the status, revision and project of the linked work item.

When comparing these two approaches, Polarion’s link roles allow for a more thorough linking process while still ensuring clear relations between linked work items. This reduces the overhead when compared with browsing DOORS modules when using requirements, risks or other work items as input for development actions. The complete

relations visible in Polarion's linked work items allow for a more complete overview of the impact and origin of the work item. In addition, this fills the visibility gap in the current implementation that does not make linked work item status and revision apparent within the tool.

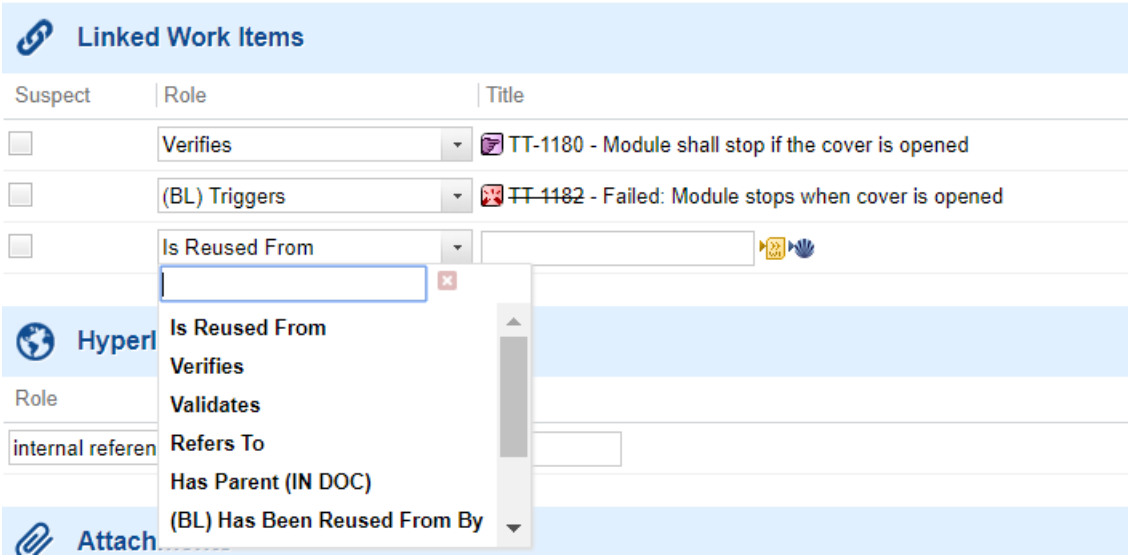

Figure 15 - Example of Polarion Linked Work Items

In producing and maintaining traceability, both tools can be effective. However, Polarion's advantage related to traceability comes from both the ease of reusing and branching previous documents in new projects, and the intuitive way of setting up and maintaining links to those previous documents and new work items. Copying DOORS modules, as templates or with object data, invalidates the links done through link modules. In contrast, branching for e.g. a regulatory requirements document between projects in Polarion maintains the linked parent work item relation. This feature in Polarion can be leveraged especially with common design inputs such as regulatory requirements, or in case of a new project for an existing product.

Polarion also compares advantageously when considering traceability deliverables. DOORS exports in Microsoft Excel or Word formatted tables are often clumsy and require specific views to export only relevant fields. However, in Polarion a traceability report of any project document can be retrieved easily, showing the work items, their parents and child work items in a table format. This table can also be integrated to an export-ready document layout so that no extraneous tools are necessary when project data is

transferred to e.g. DHF in a separate tool. This functionality can be used to e.g. monitor that test cases in a verification plan are linked to requirements. A requirement traceability report in Polarion can be created by choosing the starting level and dependent work item types. This report, pictured in Figure 16, can be exported and used as a traceability deliverable or it can be used to e.g. monitor project testing coverage in real time.

Starting Level	Dependent Items	Test Case	Latest Test Result
TT-1158 - Operator must be able to load and unload samples to the automation track		Test Case Missing!	
TT-1184 - Customer requires support for CLSI AUTOA-2 tube types		Test Case Missing!	
	TT-1183 - Module shall support multiple tube types	Test Case Missing!	
	TT-1185 - Capped tubes shall be gripped below the cap, not touching the cap	TT-1186 - Verifies: Capped tubes shall be gripped below the cap, not touching the cap	*No Records*
TT-1172 - It must be possible to configure error routes for samples that can not be processed		Test Case Missing!	
	TT-1171 - Module shall update an error route for samples that have not been processed correctly	Test Case Missing!	
TT-1179 - Operator must be able to load and unload samples to the automation track		Test Case Missing!	
TT-1157 - Operator must have a module that is safe to operate		Test Case Missing!	
	TT-1170 - Module shall not initialize if the cover is open	TT-1164 - Module start when cover is open	Passed (2020-01-22 10:17) (Deviations) 20191129-1533 - Verification Test Run for TT-11
	TT-1180 - Module shall stop if the cover is opened	TT-1181 - Module stops when cover is opened	Passed (2020-01-22 10:04) (Deviations) 20191129-1533 - Verification Test Run for TT-11
TT-1178 - Operator must have a module that is safe to operate		Test Case Missing!	
	TT-1180 - Module shall stop if the cover is opened	TT-1181 - Module stops when cover is opened	Passed (2020-01-22 10:04) (Deviations) 20191129-1533 - Verification Test Run for TT-11
TT-1156 - Operator must be able to load and unload samples to the automation track		Test Case Missing!	
TT-1155 - Operator must have a module that is safe to operate		Test Case Missing!	

Figure 16 - Example of a Polarion requirement traceability report

Polarion's capabilities in producing visually effective traceability deliverables sets it apart from DOORS where the link data retrieval is unintuitive and not suitable for actively monitoring the project status. While DOORS is a very capable tool for managing traceability, it has disadvantages when managing multiple projects that are spread across a vast number of separate modules. Additionally, the visibility of linked work item roles and status in Polarion compared to DOORS provides a much more immediate overview of the specific work item in an earlier phase of development.

5.5 Documentation and process automation

Automating processes can be seen as a key factor in increasing productivity and reducing monotonous and repetitive tasks for employees. As an example, automatic generation of documents such as plans and reports, leaves more time for testing and development activities. Automating documentation is especially desirable for the medical device industry as a regulatory need for design and development control manifests often in documentation.

The current documentation control is conducted outside of any development tools, using exported data from DOORS and document templates stored in another tool. Templates have to be heavily modified based on the subject matter and exporting large amounts of data in a table format often causes problems regarding the legibility of the final document. Polarion ALM can be leveraged in this case by using the easily modifiable templates that can either be approved inside Polarion with a digital signature or exported as a complete file to an external PLM or PDM tool.

Polarion templates are a modular way of building documentation in a consistent format, which makes project data clear and uniform for management across all projects. As seen in Figure 17, templates are stored and configured in the same tool. The Configure Template functionality allows Polarion templates to be configured based on an existing template with simple to use functionality of selecting the target template and the template you decide to modify. In addition, this allows for creation of project specific templates with relatively low effort if needed and storing them within the project.

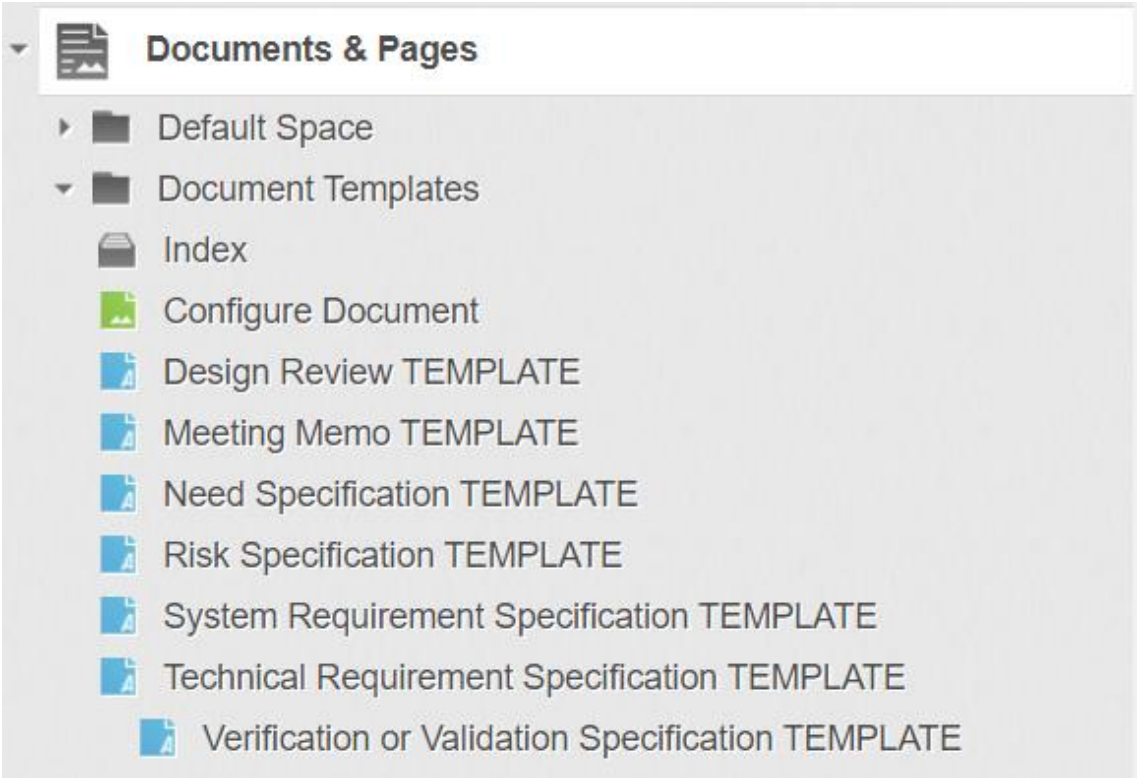

Figure 17 – Project document templates in Polarion

Templates in Polarion can be branched out or reused. Currently the DOORS modules for requirements, test cases, etc. have to be created individually, which often results in duplicative work. In Polarion, branching and reusing allow for easy creation of documents between projects by using an existing one. Reusing a document creates either a new stand-alone document with an option to link it to the original, or a document with derived fields from the original document. Branching on the other hand duplicates the document for iteration in a new location. These functionalities of branching and reusing documentation are seen as highly valuable, especially for the medical device development, as the requirements from regulatory control are the same for most projects. In addition, branching of documents allows for keeping the upwards traceability to stay intact while duplicating documentation, thus not necessarily requiring a new approval process for the document.

Revisions in DOORS are made with manually created baselines as a part of the exporting process, where the data is taken for approval in an external tool. Having only a few effortlessly retrievable revisions of constantly evolving documents, which are modified by individuals within a cross-functional team, or even multiple teams, contrasts poorly to the desired level of control.

Using Polarion, a document is revisioned automatically after each action even with simultaneous changes. These revisions can be approved, baselined and compared in the same tool, making the control of documentation intuitive. As Polarion approvals use electronic signature and are timestamped in the document revisions, the approval process can be flexible on a work item basis or to the entire document. The approval workflow with using Polarion could resemble the example in Figure 18, where approvals made in Polarion remove the need to repeatedly export versions to the external PLM compared to the current process in Figure 9.

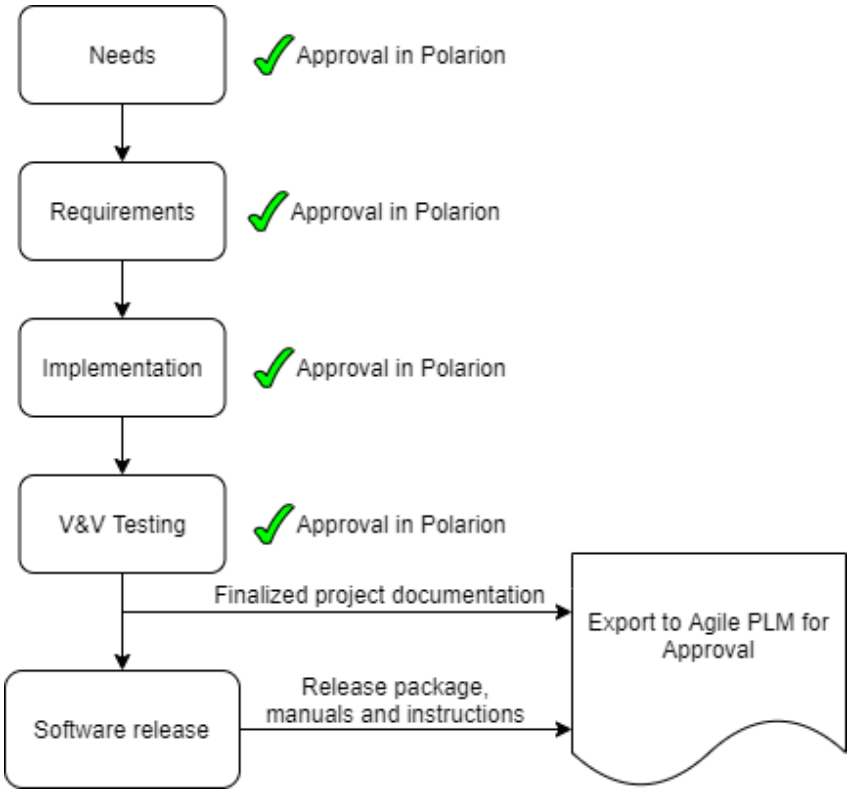

Figure 18 - Possible project workflow using Polarion approvals

In addition, Polarion supports ReqIF and Microsoft Word document importing and exporting. In a situation where the DHF is held in a separate software tool this enables keeping up to date approved documentation in Polarion as well, even if the decision is to keep the external approval process for project deliverables. This change in the working process removes the need to use separate tools to acquire important information during development.

Reporting of project artefacts is conducted currently by exporting data from DOORS to a general template, usually in the form of a table. In comparison, Polarion has configurable report templates that, when exported, can be used as they are. These report templates can be used to present real time data inside of the tool in the form of e.g. a defect report that collects and visualizes open and closed defects in flowcharts or risk traceability report with automatically created risk index matrices before and after mitigative actions, as seen in Figure 19. In addition, Polarion has a built-in capability to create traceability reports for any project document selected.

Figure 19 - Example of a risk traceability report

Using the Microsoft Azure DevOps connector extension for Polarion further sets the current tools apart in automating the information flow between the development platform and requirements, risks and test cases. The connector can be set to use bi- or uni-directional synchronization to duplicate work items made in DevOps tool to Polarion, while keeping the links between work items intact. This functionality allows for easier tracking of development efforts through Polarion, synchronizing defect status and visibility between the requirements management and development and providing more visibility of testing progress from Polarion to DevOps.

6 Results

Results from comparing requirements management tools are separated into two different categories. Categorization for comparison in capability to perform as a part of the current development and management processes conclude how Polarion can be used to effectively replace DOORS in the current working processes. Process improvements made possible by taking Polarion in use are collected under a separate chapter to emphasize the capabilities not currently present in any form or to highlight their impact on the current operating procedures.

6.1 Polarion as part of current processes

Polarion's browser-based approach and user-friendly features reduce the amount of time required to be able to start effectively using the tool leaving more time for productive actions. As the tool is easier to adapt to, it is a better solution for environments where projects and products change regularly. This intuitiveness of use facilitates the onboarding process for new and current employees to effectively work with Polarion. In addition, Polarion can be integrated to the current working processes with a relatively low initial effort, while still having a high degree of customizability and a variety of features that can be used for continuous improvement of processes.

Polarion provides better visibility of the development progress and its artefacts compared to DOORS, which can be leveraged for refining the working process from project planning during development to individual work objects. Ability to present useful visual information such as defect flow for products or development relation to planned milestones provides immediate visual feedback of the progress. Polarion provides visibility of the related work item's statuses and allows the user to generate visual representations of work item relations on a document level, features which are not present in DOORS.

Configurable templates in Polarion enable automatization and improve efficiency in producing documentation. These features can be used in a specific project or deployed repository wide. Polarion LiveDocs documents eliminate the current post export modification process to tailor report and plan contents for each specific case, when the document can be created in its entirety in Polarion and exported as is. In addition to improved

generation of documents, Polarion documents are modifiable simultaneously by multiple people further reducing time lost due to the tool in use.

The current review and approval processes can generate more useful information by using Polarion. The review process leaves a history of revisions with a comments section for work items that improves the visibility of the peer review feedback and alleviates the need to use separate communication channels for that purpose, therefore allowing others to benefit from the feedback as it is widely available. In the required approval process, which is not supported by DOORS, be conducted in Polarion with ability to approve with electronic signatures.

The reusability features of Polarion provide an advantage in requirements management when compared to DOORS, while effortlessly conforming to current processes. The creation of requirements with improved templates and being able to reuse or branch requirements for new projects facilitate the requirements management process. Software changes can be handled in Polarion by branching the original requirements to a new project for a new iteration that retains its connection to the original, whereas, the current implementation of DOORS requires an independent software change specification module for the changed requirements. This simplifies the process while improving the regression risk analysis and finding impacted test cases, which in return can be simply reused for the new project.

The advantages of Polarion implementation to the current processes are visualized in Figure 20. In addition to simplified working process, Polarion enables the possibility of using an API connector to synchronize data such as testing and defect status between the implementation platform and Polarion, which is not possible with DOORS.

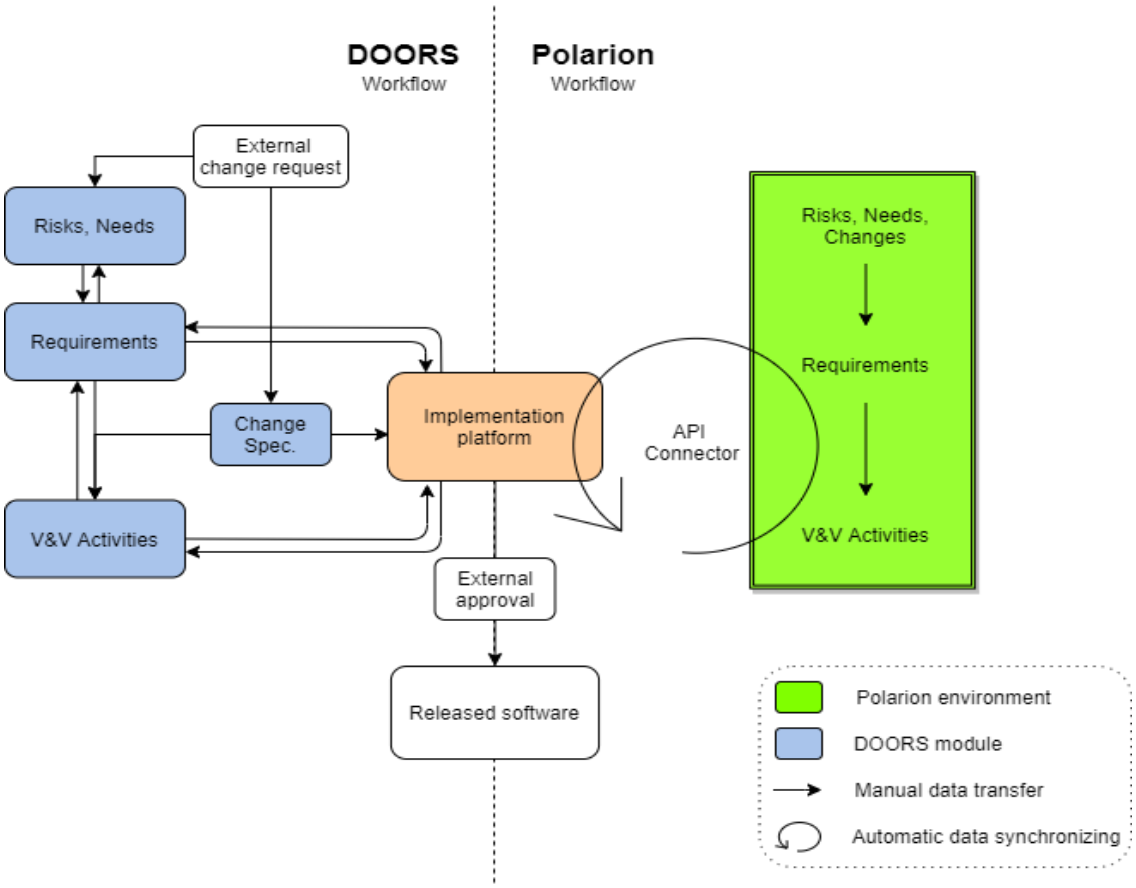

Figure 20 – Examples of Polarion and DOORS implementations in current processes

Automating document creation in an industry that requires heavy developmental control is a clear advantage for Polarion compared to the currently used methods and tools. In addition to automating documentation, Polarion allows for heavily customizable and re-usable templates and intuitive ways of building documents by allowing the user to insert a select revision of a work item to a document. Polarion can be used to automate the creation of project metrics for management purposes and enables effortless retrieval of select revisions of any document without exiting the tool. The ability to react fast with the customizability of project documentation can be leveraged depending on the company's needs and is a valuable asset in an evolving and controlled development environment.

6.2 Opportunities for process improvements

Improvements to the current working processes can be minor or major, the difference being in their impact and amount of work required for implementation. Major process improvements can require more cross-functional management and changing large portions of operating procedures and working instructions, triggering more reformative work than is often desired. Minor improvements can be seen as reducing, simplifying or automating undesired tasks or other quality of life improvements to the process.

Minor process improvements include e.g. the ability to reduce collecting work item related data to network drives by adding them as attachments to their related work item. Attachments for requirement type work items are not currently used in DOORS, and their integration to the work item template allows for centralizing data or information related to the work item. Automatic defect triggering in case of a failed test run and possible synchronizing of defect information to the development platform can be used to reduce manual reporting of defects. In addition, Polarion defect statuses can be visualized with built-in tools inside the platform to track project defect handling and provide immediate visual feedback of the progress.

Another minor improvement is the ability to keep high level links intact even across projects, which creates a more comprehensive trace. As Polarion tracks time used for test runs, the data collected from testing during an integration phase can be used to more effectively evaluate time the consumption of formal verification activities.

Improvements with documentation come from the project specific templates and instructions folders. In Polarion, they are located under the applicable project in the repository improving the accessibility by not having to collect them from separate tools or network drives. Templates can be modified and iterated as a project matures and composed in a more project specific style if desired. Automated generation and real-time checking of report documents with LiveDoc format make following the project's progress more convenient on a work item basis. In addition, Word and ReqIF round-trip feature allowing exports of documents for customer review, evaluation and comments and their importing afterwards can be used to involve customers in the development process more flexibly.

Possible major process improvements that would require a heavier workload to integrate are related to the project workflow presented in Figure 18, leveraging the approvals and revisioning capabilities in Polarion. Considering this would require changes to the current development milestone and gate process, it would make sense to evaluate its actual usefulness after the tool has been taken into use. Similarly, major changes to the working process could be made to allow the implementation and release of features in smaller increments by using revisioning and branching capabilities in Polarion to facilitate more agile development. This would reduce the time-to-market of incremental software versions by relying on features in Polarion to uphold conformance with applicable regulations. The agile incremental release process, visualized in Figure 21, would require substantial changes to current procedures and a thorough impact analysis prior to implementation. In addition, its value for updating offline systems requiring a manual update process must be evaluated.

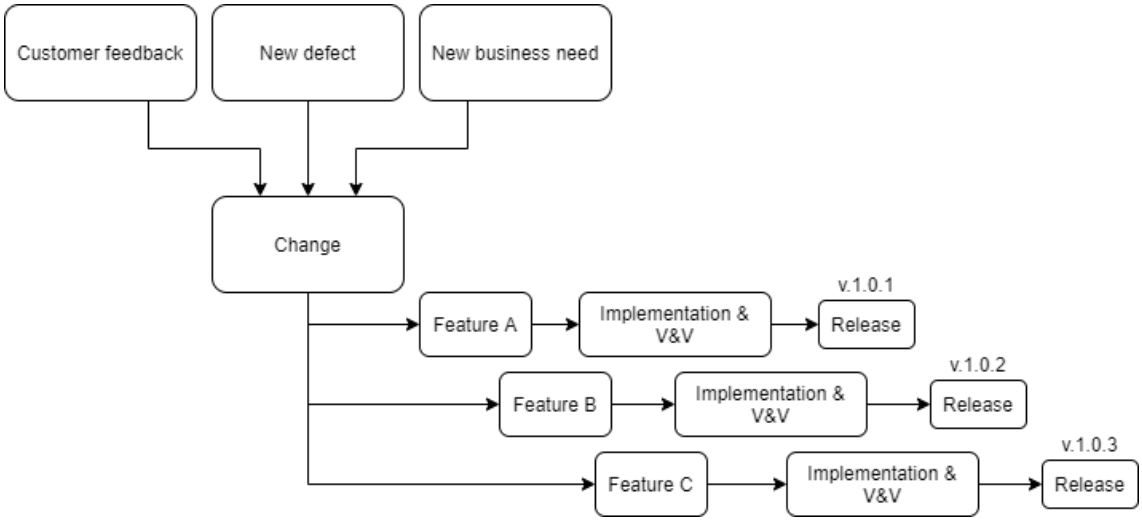

Figure 21 - Example of an incremental release workflow

Another possible, but major, process improvement would be taking Polarion's electronic signature feature to use for integrating the upkeep of the DHF into Polarion. The current DHF is maintained in an external PLM solution, as electronic signatures for approvals are required by FDA, causing additional exporting of documents. Integrating DHF to Polarion would require major process changes for development teams and quality assurance. In addition, this would require validating Polarion as an electronic archive.

A key improvement using Polarion's features is the centralizing of processes and their management to the same tool as the work is conducted in. This integration would include minor and major changes to the current working processes. Polarion could be used for change management, risk management, test management, software development and even project management, in addition to requirements management. Implementing Polarion to be used for all of the earlier mentioned activities to enables its traceability features to produce a complete trace over the product's entire lifecycle. Integration of processes to Polarion should, however, be considered on a process by process basis with their impact and benefit evaluated before actions are taken.

As a result, it is regarded as a good practice to first integrate the new requirements management solution to the current working processes in a pilot project, then, gradually implement more features, projects and process improvements. The use of Polarion features must be validated for its intended use by the company, thus requiring multiple validations for incremental adaptation of its features.

7 Conclusions

This thesis, commissioned by Thermo Fisher Scientific, consisted of a comparison between two requirements management solutions and their impact on working processes in software based medical device development. The company's decision was to evaluate Polarion ALM for this purpose. The software tools were compared as a part of the current working process. To begin with the comparisons, it was necessary to collect and analyze applicable theoretical knowledge, regulatory control and industry practices with medical device development and software development. Possible process improvements with integration of a new tool were identified based on their impact on the process.

In conducting the study, assistance from co-workers and the instructors was found essential for collecting a comprehensive problem statement for current methods and tools. In addition, the accessibility of the company instructor's guidance in industry specific intricacies was crucial.

The results of this thesis conclude that implementing Polarion ALM for its intended use by Thermo Fisher Scientific Oy is possible and provides improvements to the current

working processes even in early stages of implementation. In addition, multiple process improvements enabled by Polarion's features were identified and their implementation process was evaluated. The findings from comparisons will be used in the onboarding process for Polarion and conducting the formal testing required for the implementation of a new software tool.

References

C. Denger, R. L. Feldmann, M. Host, C. Lindholm and F. Shull, (2007) "A Snapshot of the State of Practice in Software Development for Medical Devices," First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, 2007, pp. 485-487. doi: 10.1109/ESEM.2007.54

Cognidox, (2020) 'Medtech development guide' [Online] https://www.cognidox.com/medtech-development-guide (Accessed: 15 January 2020)

Dick, Jeremy & Hull, Elizabeth & Jackson, Ken. (2017). 'Requirements Engineering' pp. 2

European Union. (1993) 'Council Directive 93/42/EEC of 14 June 1993 concerning medical devices'

FDA – TGA – ANVISA – HPFB. (2018) 'Cooperation in the Medical Device Single Audit Program (MDSAP)' [Online] Available at: https://www.fda.gov/international-programs/cooperative-arrangements/fda-tga-anvisa-hpfb-cooperation-medical-device-single-audit-program-mdsap (Accessed: 14 February 2020)

FDA, (2002) 'General Principles of Software Validation; Final Guidance for Industry and FDA Staff' pp. 11.

FDA, (2005) 'Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices' CDRH, Rockville.

FDA. (2011) 'Quality System Regulation' 21 C.F.R. § 820, sec. 820.30 - 820.40.

Gilbert Regan, Fergal Mc Caffery, Kevin Mc Daid, Derek Flood. (2013) 'Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model' Computer Standards & Interfaces, pp. 3-9.

H. Doernemann. (2002) 'Tool-based risk management made practical' Proceedings IEEE Joint International Conference on Requirements Engineering, Essen, Germany, pp. 192-.

International Organization for Standardization. (2016) 'Medical devices — Quality management systems — Requirements for regulatory purposes' (ISO standard no. 13485:2016, sec. 7.3.6-9)

International Organization for Standardization. (2019) Medical devices – Application of risk management to medical devices (ISO standard no. 14971:2019, sec. 4.4-4.5)

Joachim Rossberg. (2014) 'Beginning Application Lifecycle Management' (1st. ed.). Apress, USA, pp. 20.

Kääriäinen J., Välimäki A. (2008) 'Impact of Application Lifecycle Management — A Case Study.' In: Mertins K., Ruggaber R., Popplewell K., Xu X. (eds) Enterprise Interoperability III. Springer, London, pp. 55-67.

L. Gao, J. Gao, Y. Fan and J. Zheng, (2009) 'Research on PLM System Architecture Model and Process,' 2009 WASE International Conference on Information Engineering, Taiyuan, Chanxi, pp. 491-495.

Méndez Fernández, Daniel, et al. (2016) "Naming the pain in requirements engineering: contemporary problems, causes, and effects in practice.".

Microgenesis. (2020) 'Application Lifecycle Management' [Online] https://www.mgtech-soft.com/application-lifecycle-management-alm/ (Accessed: 14 January 2020)

Siemens. (2017) 'Polarion ALM Factsheet' [Online] Available at: https://polarion.plm.automation.siemens.com/hubfs/Docs/Fact-sheets/Polarion_ALM_Fact-Sheet.pdf (Accessed: 3 February 2020)

Thermo Fisher Scientific. (2020) 'About Us' [Online] Available at: https://corporate.thermofisher.com/en/about-us.html (Accessed: 13 January 2020)

Thermo Fisher Scientific Finland. (2020) 'About Us' [Online] Available at: https://jobs.thermofisher.com/global/en/finland-about-us (Accessed: 13 January 2020)