TRANSFoRm

Translational Research and Patient Safety in Europe

Milestone 13

Software Quality Plan: A Review Report

Work Package Number:	**3**
Work Package Title:	**Technical Frameworks, Policy and Integration**
Nature of Deliverable:	Report
Dissemination Level:	Public
Version:	2.0
Delivery Date From Annex 1:	M63
Principal Authors:	Noel Carroll and Ita Richardson
Partner Institutions:	University of Limerick
Internal reviewers:	Vasa Curcin, King's College London; Nikolaos Mastellos, Imperial College London

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 247787 [TRANSFoRm].

Contents

1. Purpose

The underlying concept of TRANSFoRm was to develop a 'rapid learning healthcare system' driven by advanced computational infrastructure that would improve both patient safety and the conduct and volume of clinical research in Europe. This report examines the various software quality processes throughout the TRANSFoRm project and highlights how the Software Quality Plan supported the success of the TRANSFoRm project. Building on *Work Task 3.5 – Software Quality Assurance Framework* this report reviews the implementation of the TRANSFoRm Software Quality Plan. The TRANSFoRm Software Quality Plan outlines the processes which should be undertaken so that when software output from this project moves towards commercial exploitation, potential partners will understand the level of quality currently built into the prototype which has been developed by this project. This report also describes the subset of Software Process Quality Plan employed by various partners in the TRANSFoRm project. Finally, it summarises the main findings from various partners and their experience with implementing the Software Quality Plan.

The main objectives of undertaking this review is to learn how various practices are employed as part of a Software Quality Plan, to identify key strategies which contributed towards software improvement, and to report on the various 'lessons learned' throughout the software development team's experiences.

2. Introduction

Software engineers must have a specific goal to deploy software to address users' requirement and must achieve a certain level of quality to support its functionality. Therefore, software engineers are responsible for quality requirements. The various processes associated with software quality are normally incorporated in the software development process. Software quality is defined as "the degree to which a system, system component, or process meets customer or user needs or expectations" (*Kalaimagal and Srinivasan, 2008)*. Quality encapsulates the totality of all the features and characteristics associated with software which are designed to address a specific need. Therefore, evaluating software quality requires "*a systematic examination of the software capability to fulfil specified quality requirements*" (*Dukic and Boegh, 2003)*. We employed Work Task 3.5 – Software Quality Assurance Framework to provide a systematic evaluation approach. In doing so, we have identified key quality processes and examine how software developers within the TRANSFoRM project have incorporated the Software Quality Plan to successfully develop software products. We carried out a number of interviews (see Appendix A) across three locations (UK, Ireland and Poland) using a semi-structured interview technique. We also gathered key software development documentation resources to examine how key elements of the quality plan were employed. Our results indicate that the Software Quality Assurance Framework proved to be a core resource for the TRANSFoRm project providing a common goal and vocabulary for all of the TRANSFoRm team to focus on throughout the software development process. We examine this in further detail throughout this report.

3. Overall Experience with the TRANSFoRm Project

This section provides a brief overview on the software developers' overall experience with the TRANSFoRm project (for more details see Section 4 – 'Software Quality Plan'). The interviewees (*n=6*) discussed the concept of 'quality' and their overall experiences throughout the project in maintaining high levels of quality. Documents used throughout the TRANSFoRm project, for example, Description of Work (DoW), deliverables, academic publications, presentations, meeting minutes and software development packages were also gathered and analysed to support this research. They highlighted the demand which quality assurance places on a team – often requiring skill to adhere to the complexity of quality and support the numerous procedures around governance and sharing developments between various organisations. They also discussed the challenges faced by a

team to achieve consistency across all of the project team and indicated the necessity to address such challenges which were not unique to TRANSFoRm.
Overall, respondents felt that they successfully adopted the Software Quality Plan based on resources and expertise available within their team. To adhere to their Software Quality Plan, the TRANSFoRm team adopted the V-Model (Figure 1) and engaged in an agile-like/incremental software development approach which enabled them to respond to various changes requirements. The general consensus within the TRANSFoRm team was that they had adopted an agile philosophy (yet not necessarily following the structured agile process or guidance), thus affording the team greater control over quality through various phases. An agile methodology may have been adopted internally as part of individual team structures but was not explored as part of this review.

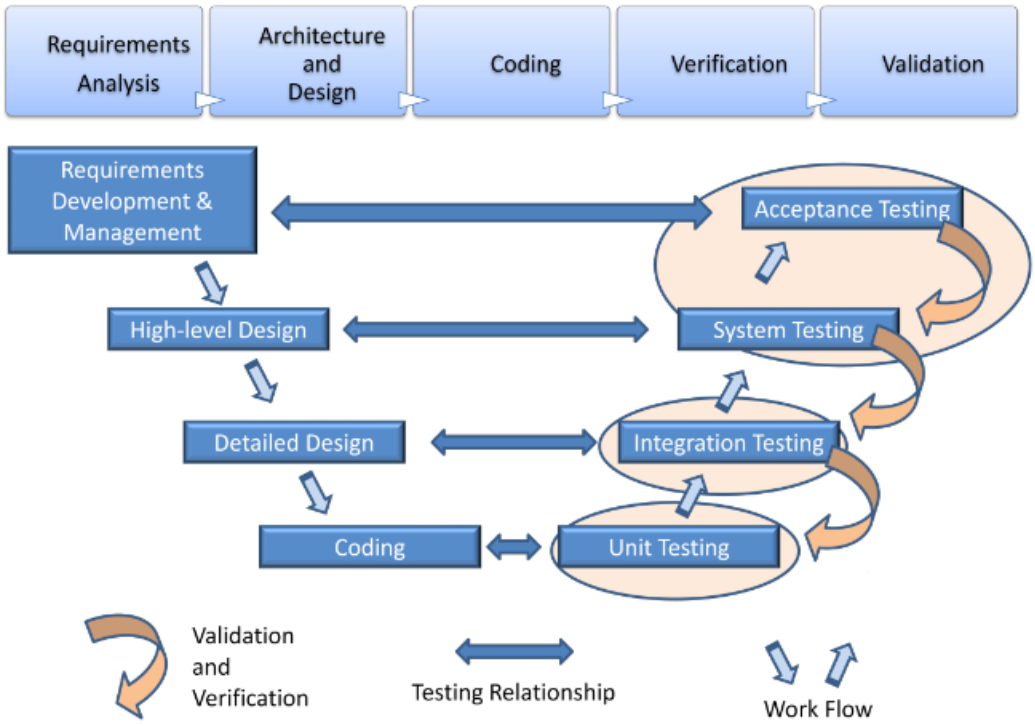


Figure 1 V-Model of TRANSFoRm Software Testing

Some members of the TRANSFoRm team reported that they were not involved in the project at its very conception which caused some challenges in developing a thorough understanding on the scope of the project and its goals. However, through the formation of new collaborative relationships between all partners, this challenge was addressed. It was also reported that staff retention presented possible issues for software quality. To overcome this issue, the TRANSFoRm team implemented an effective software documentation strategy which supported newly recruited staff by providing the documentation as an instructional and informative resource. For example, the TRANSFoRm team provides some examples of the various documentation outputs included:

- ***Design documentation***: technical design documentation which was created at various stages of software development. Initial requirements analysis was performed with particular emphasis given to input received from other TRANSFoRm development teams whose tools will integrate with the distributed infrastructure developed. Documents produced are human readable documents targeted at end users to enable collaborative design decisions.
- ***Software documentation***: constructed throughout development in the form of comments and standardised comment blocks within the source code. In typical development scenarios, a JavaDoc style system of annotation is used for code. Inline comments are used to explain algorithms and other complex code blocks. All source code is annotated with the author,

name of the code block (class or otherwise), description of the functionality of the code and all input/output fields.

- ***Installation guides***: created to facilitate other TRANSFoRm development teams to use the software components. These human readable documents provide detailed step by step description for installation and integration of software in local environments.
- ***To do lists***: used until release and any requested updates to be completed.

The development teams were based across a wide geographically distributed environment, for example, University of Birmingham (UB), University of Warwick (UW), Royal College of Surgeons in Ireland (RCSI), University of Dundee, NIVEL, **Error! Bookmark not defined.Error! Bookmark not defined.**Trinity College Dublin (TCD), Imperial College London (ICL), King's College London (KCL) and Wroclaw University of Technology (WRUT). Because of the large number of partners (user groups and developers) involved in TRANSFoRm (22 partners in total), it was reported that some of the team found it challenging to learn about new members and learn how to work across a distributed environment. The team comprised of the following partners:

1. King's College London (UK)
2. University of Birmingham UK)
3. Trinity College Dublin (IE)
4. Imperial College London (UK)
5. Heinrich Heine University Dusseldorf (DE)
6. University of Dundee (UK)
7. Royal College of Surgeons in Ireland (IE)
8. St George's - University of London (UK)
9. Mediterranean Institute of Primary Care (MT)
10. NIVEL (NL)
11. University of Limerick (IE)
12. University of Antwerp (BE)
13. Karolinska Institute (SE)
14. University of Crete (EL)
15. GPRD (UK)
16. Quintiles (UK)
17. INSERM (FR)
18. University of Warwick (UK)
19. Wroclaw University of Technology (PL)
20. CSIOZ (PL)
21. Custodix (BE)

Therefore, team work was a critical success factor in TRANSFoRm and this is evident throughout the software development lifecycle. The Project Management (Figure 2) played a key role in ensuring the development, planning and management of TRANSFoRm is achieved successfully.

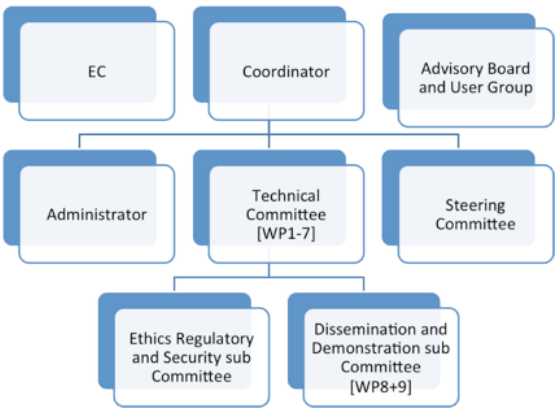
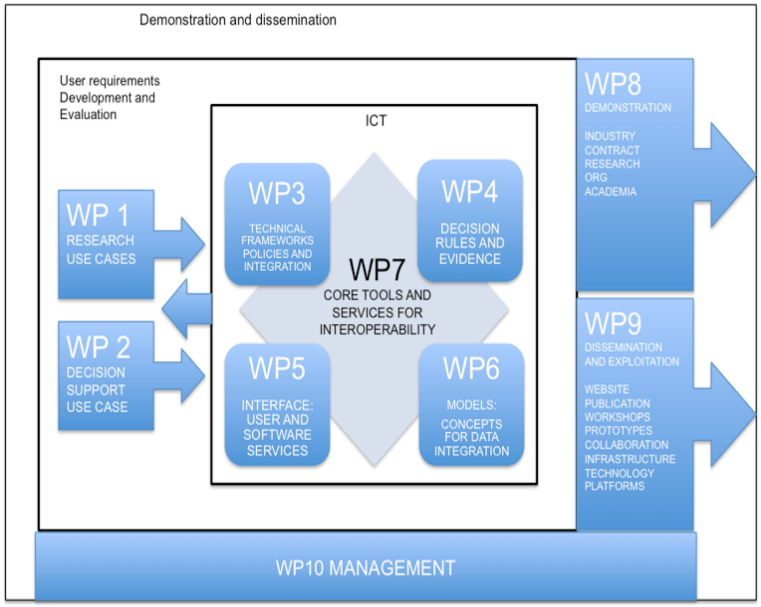


Figure 2 Overview of TRANSFoRm Project Management Structure

Based on collaborative efforts, mock-ups were also produced to define components and interfaces with other software tools. These enabled discussion regarding the integration of separate software components. Discussion of these mock-ups allowed the Task Teams to develop collaborative relationships and share expertise for further refinement of requirements. Such efforts were an essential part of the software specification design. Feedback from the TRANSFoRm partners would suggest that through the adoption of an agile methodology, whereby change requests from new feature requests and requirements were implemented in short timeframes; Task Teams were encouraged to engage in software design and development as a more cohesive team – further ensuring the success of the TRANSFoRm project (see work packages overview Figure 3).



Work Packages:

- WP1 Research Use Cases
- WP2 Decision Support Use Cases
- WP3 Technical Frameworks, Policies and Integration
- WP4 Decision rules and evidence
- WP5 Interface: User and Software Services
- WP6 Models: Core ICT Concepts for System and Data Integration
- WP7 Core Tools and Services for Interoperability
- WP8 Demonstration and Exploitation
- WP9 Dissemination and Exploitation
- WP10 Management Activities

Figure 3 Overview of TRANSFoRm Work Packages

Considering the unique aims and exploratory nature of TRANSFoRm, the Task Teams indicated that the goals of the project were described as challenging. Adding to the challenges was a sense that the main objective of TRANSFoRm had to be a little flexible as research by other partners progressed. Furthermore, there were some modifications to the project following European Union reviews. This is expected within any research project, but makes it more difficult for the development teams to develop project requirements. The distributed infrastructure was described as the product of an original 'wish list' which was sent out to all partners. Due to the novelty of TRANSFoRm, it was difficult to clearly identify what the requirements ought to be. The Software Quality Plan provided some structure on guiding and documenting the decision-making process to manage the requirements. This supported the wider team in identifying what was deemed necessary and sufficient for the project functionality. This was achieved by presenting a vision for TRANSFoRm between management and development teams which encouraged greater participation in refining the requirements. In addition, the prototype development phase of the project became a core element of the project, supporting the building of collaborative relationships and enabling research to evolve between partners interactions. In addition, within the Software Quality Plan, the management of expectations was also a key process towards the success of TRANSFoRm building knowledge. This also supported the development of a rapid learning healthcare system driven by an advanced computational infrastructure.

4. Software Quality Plan

Building on the TRANSFoRm Software Quality Plan (*Work Task WT 3.5 – Software Quality Assurance Framework*), this report examines how the plan supported the TRANSFoRm software developers to build in a level of quality into their tools. The various stages of the plan were devised based on published software processes and Medical Device Regulations. We revisit this Software Quality Plan with the aim of reviewing how it was implemented, how it contributed towards the success of the TRANSFoRm project and whether there are any valuable 'lessons learned' from the software development team who implemented this plan. Specifically, we interviewed the development teams located in the UK, Ireland and Poland which constitute as key members of the distributed project team. Within this report, these teams are presented under the umbrella of the TRANSFoRm development team. In addition, we surveyed the TRANSFoRm Work Task team using the Software Quality Plan (see feedback in Appendix B) to examine how they implemented the plan to successfully meet the projects aim and objectives.

4.1 Managing Software Requirements

The Requirements Development stage presented the TRANSFoRm team with an opportunity to elicit, develop and analyse customer, product and product component requirements. This allowed them to understand and document customers' needs, expectations and constraints and examine how this would influence the project deliverables and overall product.

The TRANSFoRm Work Task team firstly determined appropriate methods to elicit requirements which best suited their situations, concerns and abilities (as partners). These methods ranged from structured interviews, surveys, use case studies, face-to-face meetings or online exchanges, e.g. email and WebEx. The nature of the TRANSFoRM project was considered to be dynamic which contributed towards the change in software requirements as the research project evolved. This also presented challenges for the team from time-to-time when they had to keep up with the pace of change. To minimise any potential issues surfacing as a result of the pace of change, the team adopted a test-driven design to managing software requirements. Our findings suggest that although all partners were not involved in the requirements gathering and management, (some partners joined when the project was more mature), based on individual experience and competencies, they still had some influence on altering them. Being involved in various tasks and sub-tasks ranging from, for example, mobile, web application, and decision-support developments enables the TRANSFoRm Work Task team to contribute to predefined requirements which had been set out in the DoW. Collaboration with other team members and partners was pivotal to satisfy all requirements. Yet, this was a lengthy process to develop a holistic understanding of the projects and how various components fit together to provide a "*complete jigsaw using all of the pieces*".

The TRANSFoRm Work Task team felt that the requirements development process was the most challenging part of the overall project. There were a number of factors which contributed to this, for example, new exploratory research area, communication between partners at the start of the project, rapid changes at the beginning of the project due to the dynamic nature of the requirements development, and presenting requirements at face-to-face meetings making changes slow to address. In addition, this required some iteration of the requirements management process. The DoW supported the management of software requirements and provided a '*requirements commitment strategy*'. This also sustained relationships through email and also added a valuable link for traceability purposes (i.e. tracking changes). The TRANSFoRm Work Task team explained that there was a need to agree on requirements and agree on updates or changes on a weekly basis through face-to-face meetings using various documentation (for example, meeting minutes) to monitor changes. If a change in requirements was being considered, the TRANSFoRm Work Task team were involved in assessing the impact this would have on the entire project.

Obtaining information from stakeholders about their various needs and expectations, while understanding product constraints and interfaces to develop requirements for the product, was a key

process which shaped the overall TRANSFoRm project. As the interviewees reported, requirements was not a static process but rather dynamic and evolved with the development of the research and changing needs of the stakeholders. Members of the TRANSFoRm Work Task team explained that this was largely due to the range of different expertise and sometimes the differences in the conceptualisation of the project requirements. This would sometimes give rise to conflicts in opinion. Conflicts were typically resolved via face-to-face meetings and through email exchanges to address any inconsistencies. Requirements were managed and inconsistencies addressed using some of the following processes:

- Managing all changes to the requirements;
- Maintaining the relationships among the requirements, the project plans, and the work products;
- Identifying inconsistencies among the requirements, the project plans, and the work products;
- Taking corrective action.

Our findings also indicate that product and product component requirements were generated from the customer requirements and the local Principal Investigator's (PIs) experience/expertise. The various discussions regarding new and/or changed requirements required that the technical team were ultimately able to translate design and functionality requirements into software products. Additional requirements and constraints were also obtained from medical and health standards and legal requirements. The process of functional analysis allowed the Task Teams to define the required functionality with various actions, sequences of events, different inputs, outputs, actors and other information associated with the functional architecture.

The team reported that they were involved in requirements through a research-based approach, i.e. discovering what the requirements ought to be. They developed a study on the data collection system to examine methods (i.e. software requirements). This enabled them to capture data directly from the source while adopting industry standards to do so (e.g. CDISC[1]). Managing the software requirements were also influenced by the PI's who provided key requirements information from a clinical perspective. The software developers collaborated with the clinical experts to establish the projects requirements. Our findings suggest that there may have been some mismatch on the clinical expectations and their requirements against the nature of technical changes and the resources available to realise and implement complex and large technical developments. One core issue which surfaced from our findings is that clinical expectations often lacked insight on how technical standards would influence requirements and ultimately impact on the scope of the project. Thus, while the interdisciplinary nature of TRANSFoRm did add to the richness of the outcomes, it also introduced a complexity in finding a balance between clinical needs and technical capabilities. As this demonstrates, to successfully manage software requirements at this complex level, significant efforts across the TRANSFoRm Work Task team were required. Across the research Task Team, when requirements were agreed upon, it was considered that there would be an immediate commitment to the development of the requirements, which was difficult for the Work Task team.

The Work Task Team aligned efforts with medical and healthcare industry practice (e.g. FDA, 2002[2]) which states that "*success in accurately and completely documenting software requirements is a crucial factor in successful validation of the resulting software*". The requirements formed the basis on which the system was designed and coded. The process of verification and validation (discussed later in this report) was assessed against the requirements and needs of the customer. A key focus of the requirements included the need for accuracy, currency and traceability throughout the development lifecycle. Many of the TRANSFoRm Work Task team members reported that they employed a simple agile development philosophy with a standard structured release and deployment workflow. Software was developed as part of an iterative process. New features were prototyped during an evolving process independently of core software products. This process used a typical *Development - Staging - Production* workflow for all software and hosting of services. Separate

[1] http://www.cdisc.org/standards-and-implementations
[2] http://www.fda.gov/downloads/RegulatoryInformation/Guidances/ucm126955.pdf

machines (not networked) were used by each developer, with integration between developers taking place at the staging level using source repository frameworks such as Subversion[3]. Traceability was achieved through the list of requirements and various email records exchanged between the team. However, they would have benefited from a formal direct link between developers in distributed locations i.e. a formal tracking system.

The integration process presented many challenges. One of the TRANSFoRm Work Task team members explains that "*in theory, the integration process works well in practice but it did present some challenges such as compatibility*". In addition, collaborating with various stakeholders presented challenges, for example, clinical investigations which were "*long on-going processes as the project matured*". However, this was necessary to achieve a high standard of quality. The TRANSFoRm project was very exploratory in nature which sometimes contributed to the challenges in decision-making and longer time periods to reach specific decisions. To overcome these challenges, the categorisation of requirements was not always completed by the Work Task team until decisions were fully agreed on.

In TRANSFoRm, requirements changes took place throughout the lifetime of the project (due to changes in circumstances or needs, changes in technology available, changes in priorities, etc.). Changes were justified and documented across the software lifecycle (which provided greater traceability of change sources throughout the project). Conflict resolution was not a major concern for TRANSFoRm but rather some discussion on '*difference of opinion*' which was always welcome for a project of this scale and complexity. Conflicts were viewed as an approach to refine to quality of TRANSFoRm and invite constructive criticism to enhance the research outcomes. It challenged various pre-conceptions held by the research team on specific clinical and technical developments. It was important, however, to determine whether the differences stemmed from 1) *different conceptual understandings* or b) *the lack of available resources* to realise certain software requirements (for example, technical resources, funding, or expertise). It was suggested that there may have been an imbalance in power since industry owned the data sources which presented some initial challenges. These were overcome through increased access and demo applications for TRANSFoRM research requirements.

The various customer requirements include the needs and expectations of the stakeholders, but also an elicitation of what is required and acceptable. These requirements included technical, business and clinical needs of the various stakeholders. Our findings suggest that it was important to manage some technical expectations. To do so, this required face-to-face meetings with stakeholders to explain the complexities around meeting clinical expectations and the technical difficulties which some requirements had presented. For example, it was important to manage expectations on security of the library, technical infrastructure and clinical requirements and to find a balance between all stakeholders' necessities. The balance of requirements would hinge on expertise (e.g. web service developers and knowledge in the domain) and time (balancing requirements and their changes over time). Other factors also influenced expectations, for example, regulations (e.g. privacy of patient data) and resources available to the TRANSFoRm Work Task team.

Maintaining consistency in software requirements in TRANSFoRM was considered to be a difficult task since the requirements were described as 'dynamic' catering for numerous stakeholders' requirements and/or expectations which changed throughout the project. Consistency was achieved through numerous meetings and discussions which were controlled by the integration manager of the project who also managed stakeholder expectations. Again, this highlights the level of cohesiveness within the TRANSFoRm Work Task team to foster collaborative relationships and ultimately achieving consistency of requirements.

Through various collaborative relationships, the requirements criteria were guided by the use of various techniques, for example, using UML in Year 1 to develop many use cases and test

[3] https://subversion.apache.org/

requirements through various scenarios. Within the project strategy, a key phase included the establishment of operational concepts and scenarios. The operational concepts and scenarios detailed the interaction of the product components through using various use cases or user stories and regardless of engineering discipline. The TRANSFoRm Work Task team explained that operational concepts were supported through the use of various scenarios to examine flows into the middleware. This was often influenced by the DoW and provided a central focus for discussions around the development of scenarios. To build the scenarios, the team initially developed simplistic overviews, for example, using PowerPoint, and then progressed the discussion to more technical specifications the use of sequence diagrams.

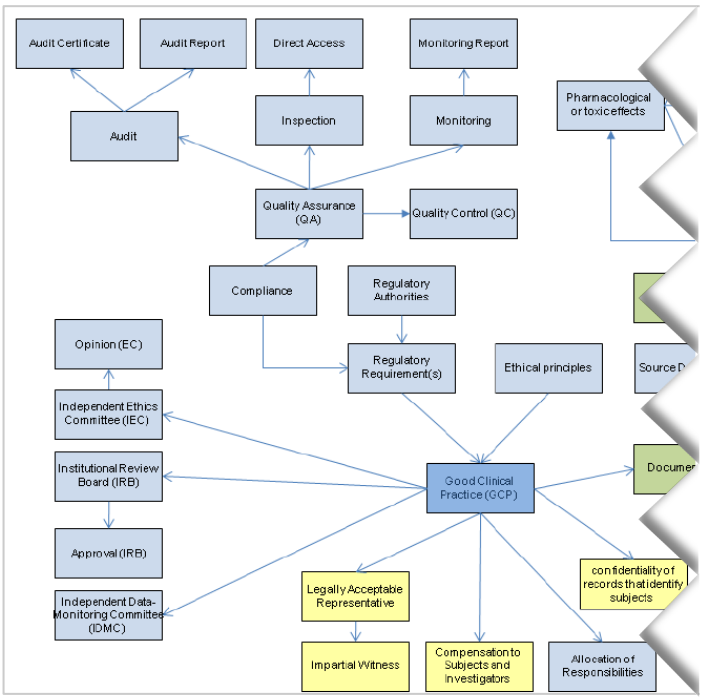


Figure 4 Snippet Sample of a Model used to Define Requirements

An agreement on the use of requirements criteria allowed the TRANSFoRm Work Task team to follow a specific model, for example; clinical-based information models using UML (see Figure 4). Such models supported the analysis to give meaning to certain software requirements and illustrated how they fitted into the UML model. Various issues that arose from discussions could be traced back to the UML to support the outcome from disagreements.

Managing requirements' change required a general process which categorised the change into '*minor*' and '*major*'. A minor change was considered easy to implement such as a change in the API. A major change was considered a more complex task such as adding a new feature. Major changes often required the involvement of senior stakeholders, for example, key clinical stakeholders. To minimise this from occurring, the team would engage in a task-driven approach which were directly linked to specific scenarios or cases. Some of the TRANSFoRm Work Task team members suggested that there were no major changes implemented in the project although minor changes were significant enough to enhance the quality of the project's outcomes. Therefore, traceability of software requirements proved to be an important process. 'Major' requirements have greater traceability but may not have had a formal process. For example, the use of email served as a key tool to search communication history on various discussion exchanges. The TRANSFoRm Work Task team explains that "*it is important to have a record and be careful of changes in the project….after all it was a five year project!*" However, some inconsistencies of work and requirements did occur but to minimise this, the TRANSFoRm Work Task team followed the DoW which indicated changes in 'individual' tasks. If there were some deviations from tasks within the DoW but there was justification that the changes would improve the overall output of the project, the DoW was updated accordingly. While the DoW

was considered a key source for requirements, it was equally important to be aware of how other requirements were linked to specific work tasks. Understanding the scope of other linked requirements was time-consuming, for example, to determine the interface or a product and how it could impact on other requirements (i.e. defining inputs and outputs). Additional constraints on project requirements stemmed from the use of various standards. Overall, the project reflected the dynamic reality of software development lifecycle. How requirements change was considered an evolutionary reflection of a dynamic and exploratory research project. The requirements were described as "*a natural fit*" between the project and current practice which already existed across the IT community.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Providing clarity on customers' needs, expectations and constraints can influence the project deliverables;
- It is important to accommodate for change within a dynamic project and establish strategies to support a team to embrace change rather than resist it;
- As part of the requirements management activities, a collaborative protocol should be established to encourage team members to exchange information and ideas on the project;
- To ensure that all team members are provided with a holistic understanding of the project, expectations and how various deliverables are integrated to meet the project's objectives;
- Establishing a management software requirements decision-support tool can support to alleviate decision-making tasks on a distributed team throughout the project lifecycle;
- Developing a requirements change and monitoring protocol can be a significant support for auditing tasks throughout the lifecycle of the project;
- Industry standards (for example, medical device and healthcare) play a very influential role in shaping the 'end product' to ensure quality and compliance is adhered to;
- A clear management structure can also provide greater transparency on how to resolve conflicts relating to distributed project developments and changes;
- Establishing operational concepts and scenarios provides greater clarity on requirements, workflow and expectations for a distributed project team;
- Introducing a shared records management collaboration system for distributed projects can prove to be a valuable tool for projects developing over a number of years.

4.2 Requirements Management Process

Our findings suggest that requirements did change and were altered in response to the users' needs and as the project evolved based on partners experience and competencies as would be expected within a research project of this scale. Therefore, the Requirements Management process identified in the Software Quality Plan became a key phase of the development cycle for TRANSFoRm. It enabled the TRANSFoRm Work Task team to manage the requirements of the project's products and product components. It also afforded the team an opportunity to identify inconsistencies between those requirements and the project's plans and work products. The product requirements were influenced by the clinical overview specification provided at the initial stage of TRANSFoRm. This was achieved using numerous tools, for example a Wiki (Figure 5) which enabled various groups to undertake specific software components of the product development for the project.

Figure 5 TRANSFoRm Wiki Tool

Our findings suggest that, as the project developments evolved, some team members gained more responsibilities for implementing requirements as they became more experienced in the project. This also added increased workloads and pressures to implement new requirements to develop software components which were prioritised. This was particularly challenging for the software developers to implement clinical requirements because clinical changes could be introduced without any thorough consideration being given to the long-term impact on the project development and how they may influence other technical requirements. Therefore, flexibility was a necessity throughout the projects development lifecycle. To allocate various components requirements, Query Workbench was used to develop specific software components. The query results were returned and enabled the TRANSFoRm Work Task team to focus on finding data sources from a library which integrated with services and were sent to the middleware. The software components were typically defined through face-to-face meetings which offered the most practical and efficient method to discuss various opportunities for the development process. Particular focus was placed on identifying the different interface requirements between the different functions/objects/product components and this played a key role in the requirements management process. As part of this process, the interface requirements were developed and were typically provided by the associated PIs. The TRANSFoRm Work Task team used various mock-up techniques and tools which presented and demonstrated various interface design options. For example, some of the TRANSFoRm Work Task team demonstrated how they used Balsamiq (for example, see *D.5.3 – Query Formulation Workbench*). Mock-ups were produced on completion of the requirements analysis, shared across the Work Task team and were used as a starting point for development of user interfaces and software workflows (Figure 7). Balsamiq allowed the team to build simple diagrams of future software interfaces in a format suitable for end users. As well as enabling design decisions for refining user interface requirements, mock-ups are used to enable discussion of integration of separate software components. In this case non-UI diagrams such as block diagrams are used to describe any required interactions. This also allowed various team members to comment on the user interface and discuss interface options with project team members.

Some members of the TRANSFoRm Work Task team explained that the interface requirements played an important role to develop an understanding of the end-user and the factors which influenced usability, for example, colours, fonts, patterns and questions. The interface also had to meet specific output requirements which influenced the input requirements. In some cases there was a lack of clear guidance or structure on specific requirements (see Figure 6). In such cases, while it may be presumed that this presents developers with a certain level of freedom to create novel software components, specific IT standards had to be adhered to and components had to integrate with other tasks.

Objective 5:

WT 5.5 (NEW) Mobile eHealth application for the delivery of Patient Related Outcome Measures

This work task will develop a mechanism for patients and clinicians to use the data captured from mobile patients under the GORD use case. It will record patient related outcome measures triggered by an index event in the EHR (consultation):

1. To develop a software application for mobile devices (such as mobile phones and smart phones) that will be used to fill by distant patients in questionnaires used in clinical trial (Patient Related Outcome Measures – PROM). Application will enable capturing:
   a. Quality of life or patient evaluations of health care scores such as SF-36 Health Survey, SF-12 Health Survey and EuroQoL (EQ-5D),
   b. Symptom scores - sets of categorized responses.

The delivered mobile part of the platform will operate on mobile devices and will be able to provide information that is individually-tailored and relevant to recent health condition. The work task will meet open standards for mobile eHealth, e.g. "Open mHealth".

This work task will be carried out by WRUT-leader (36M) and CSIOZ (6M). Months 30-48.
D5.5

Figure 6 Sample of Requirements Brief

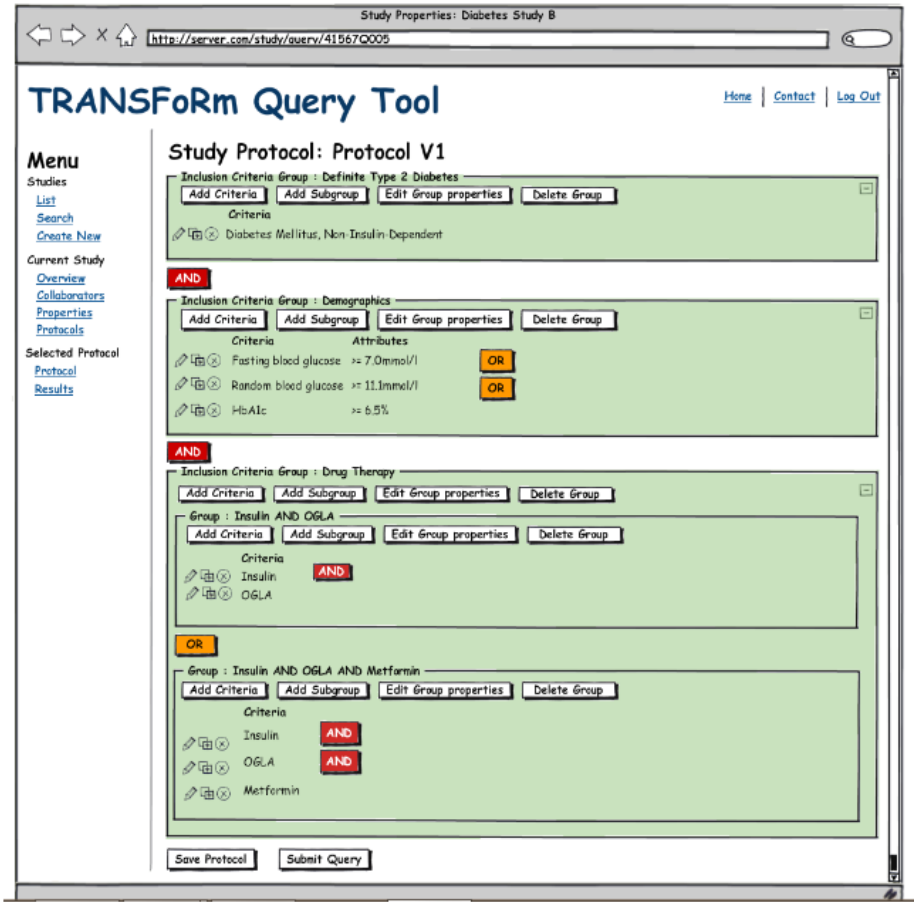


Figure 7 Use of Balsamiq to Develop Visual Mock Ups

The interface requirements were largely considered to be an iterative process since it was often based on what the middleware required and thus components needed to be decoupled. For example, Camel[4] workflow and Workbench[5] provided an interface between various components. However, requirements changed over the lifetime of the project as needs evolved and conflicts were resolved but this was largely considered to be an 'organic' development considering the exploratory nature of the

[4] http://camel.apache.org/aws-swf.html
[5] https://www.mysql.com/products/workbench/

research. Senior project partners played a key role in shaping the requirements and changes which occurred throughout. However, the conceptual architecture (Figure 8) broadly remained the same which provided a good overarching idea of what components would be required to fit into the overall architecture. This also influenced the interface and data gathering developments within the project.

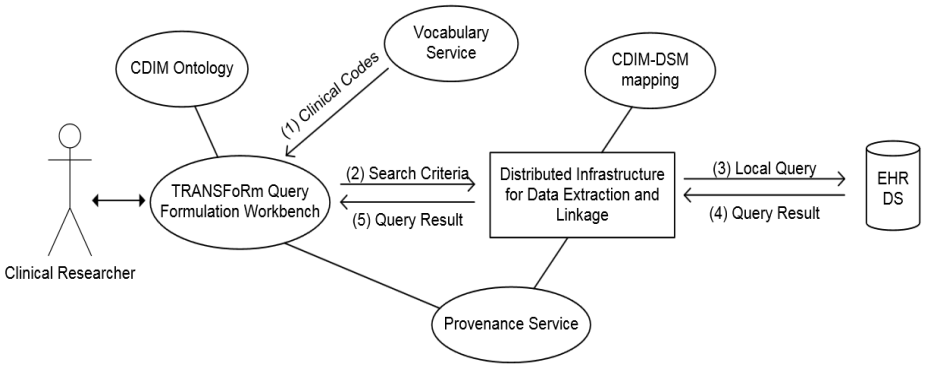


Figure 8 TRANSFoRm Conceptual Architecture

As part of the requirements management process, it was important to analyse requirements for a design perspective to ensure that specific requirements were necessary, sufficient and achievable. Progress milestones were met to track the progress of software development plan and discussed at various meetings throughout the project. Management of requirements allowed the TRANSFoRm team to quickly identify risks to the project allowing them to plan ahead and mitigate risks to the project.

The TRANSFoRm Work Task team explained the importance in achieving a balance between functionality required by the stakeholder, identifying what is absolutely necessary and exploring whether specific changes could have been addressed within a timeframe, within a specific cost structure and address performance constraints which were put in place. Therefore, requirements which were considered unnecessary were recorded but not included in the technical solution. To achieve a consistent high level of quality within TRANSFoRm, the validation of requirements (further discussed in Section 4.6) would typically occur through a collaboration process with other team members to learn through various workflows, prototypes and tried and tested methods of "*what works and what needs to change*". The validation process occurred at various levels using different plans to enhance the quality of the software output. Each software component used a validation and testing approach to simulate/test the software solutions. For example, some of the TRANSFoRm Work Task team used Smart Bear® SOAP UI for local validation. The Test Centre Integration was also used to assess the overall workflow (end-to-end). Validation was supported though a basic high-level understanding of the project which also guided the requirements management process. The basic ideas of the project did not change, i.e. the general use-cases were stable throughout the project lifecycle but some details were changed without any formal documentation, but this was discussed over meetings (e.g. WebEx).

At the development level, virtual machines or other development architectures were used by individuals to produce early versions and prototypes of software. At this stage, all developers on the team focused on their individual codebase, allowing team members to make changes without impacting the staging or production versions of their output on the entire project. During the allocation of product component requirements, requirements were outlined and individual product components were created by various team members. Simple prototypes were developed based on the interpretation of requirements and then tested, negotiated and possibly extended with other teams. Many of the components which were selected were considered to be a natural fit for the middleware used in TRANSFoRm. Key software tools used during development included, among others:

- Maven as the project management tool;
- Eclipse, or other similar integrated Java development environment;

- MySQL;
- Apache Camel was used as routing infrastructure;
- Subversion server software such as VisualSVN was used for collaborative software development. Client programs such as TortoiseSVN were used by all developers;
- Office software, such as Microsoft Office, were used when writing end user documentation;
- VirtualBox, VMware or other virtualisation platforms.

The TRANSFoRm Work Task team used real systems to test the various products and used test results to examine how components aligned with requirements. The closer the project got to the end-user the more emphasis was placed on the testing process and the more expensive the testing process became.

Managing responsibilities that directly linked team members to specific requirements was a resource-dependent task. It required prioritisation of tasks within a team management structure although it was difficult to determine the availability of developers in the distributed setting. This presented some challenges to align work schedules with other requirements development. However, the DoW did present some support though the use of the Gantt chart to prioritise tasks (for example, see Figure 9). It was suggested that greater transparency on workload and project management tasks would have been beneficial here to identify how to delegate responsibilities. To support the management of the requirements, operational concepts became an important factor, especially for some partners who had used all the TRANSFoRm components and connected remotely via servers. The TRANSFoRm Work Task team highlighted the need for better documentation to detail how the connection of components between partners was achieved. The key factor was to determine the success of the product's functionality. Functionality was based on a high-level use and understanding of the various cases. This was achieved using the mock-ups to develop well-defined interfaces. This was considered to be an individual task using individual testing methods which operated successfully internally, rather than a wider TRANSFoRm Work Task team process.

	Tool	Start	End	Year 1	Year 2	Year 3	Year 4
WT3.3	Security framework (v1)	1/03/2010	1/06/2013				
WT3.3	Security framework (v2)	1/06/2013	1/06/2014				
WT4.3	Ontology service for diagnostic evidence	1/09/2010	1/12/2012				
WT4.4	Web-based evidence service (v1)	1/07/2012	1/06/2013				
WT4.4	Web-based evidence service (v2)	1/06/2013	1/06/2014				
WT4.5	Data mining tools	1/06/2012	1/06/2013				
WT5.1	Data quality tool (v1)	1/12/2011	1/06/2012				
WT5.1	Data quality tool (v2)	1/06/2012	1/12/2012				
WT5.2b	Triggering tool	1/12/2012	1/12/2013				
WT5.3	Query and data extraction workbench (v1)	1/12/2011	1/06/2012				
WT5.3	Query and data extraction workbench (v2)	1/06/2012	1/06/2013				
WT5.4	Provenance service	1/03/2011	1/06/2012				
WT7.1	Model-based semantic mediation service	1/12/2011	1/12/2012				
WT7.2	Terminology service	1/03/2010	1/09/2011				
WT7.3	Metadata repository	1/12/2011	1/12/2012				
WT7.4	Infrastructure for deployment of eCRFs into EHRs (v1)	1/12/2012	1/06/2013				
WT7.4	Infrastructure for deployment of eCRFs into EHRs (v2)	1/06/2013	1/12/2013				
WT7.5	Infrastructure for extraction/linkage (distributed infrastructure) (v1)	1/03/2011	1/06/2012				
WT7.5	Infrastructure for extraction/linkage (distributed infrastructure) (v2)	1/06/2012	1/06/2013				
WT7.5	Infrastructure for extraction/linkage (distributed infrastructure) (v3)	1/06/2013	1/06/2014				
WT7.6	Middleware for mobile e-Health and CPR derivation	1/06/2012	1/06/2014				

Figure 9 Example of Gantt chart used to prioritise tasks

Determining whether requirements were necessary and sufficient was a challenge. The DoW played an important role since it provided the scope of the project development. This guided the TRANSFoRm Work Task team to examine whether specific requirements fell within the scope and offered them a tangible approach to managing requirements. If requirements added some form of functionality, this would typically indicate that changes were more acceptable. It was suggested that some of the TRANSFoRm Work Task team may not have tested ideas and strong personalities may have hampered some ideation of requirements changes, i.e. preferring a risk-adverse approach to the project development. However, change of requirements was considered inevitable and the timeframe and budget was flexed by the various users and stakeholders having new ideas as research progressed. The TRANSFoRm Work Task team had to demonstrate whether the solution already existed and they were required to justify minor/major changes in the project's DoW.

Some of the key lessons learned about this phase of the Software Quality Plan in TRANSFoRm for future projects include:

- As part of the project management process, changes in requirements should be factored into the project plan and specific management protocol needs to be established to cater for changes;
- Adopting collaborative tools, for example Wikis, within a distributed team environment provides greater cohesion between team members and presents greater visibility on work tasks;
- It is important to manage the level of responsibility and pressure placed upon individual team members to achieve certain work task objectives. Fostering an open and honest working environment can enhance the overall team moral and encourages more transparency on workloads across team members;
- Within a distributed team environment, face-to-face meetings offer a more practical and efficient method to discuss various opportunities for the software development process;
- Using diagram tools to develop mock-ups of software components provides a very effective communication and presentation technique to demonstrate how specific requirements can be achieved;
- Developing a conceptual architecture offers a team a core focus and provides an overarching overview of a project and the various factors which influence how deliverables are achieved;
- Prioritising testing events in the management plan is important. For example, the closer the project gets to the end-user the more emphasis should be placed and the testing process and the more expensive testing process becomes;
- Linking responsibilities to team members (based on staffing resources, time, and expertise) needs to be made visible for all team members to understand who is linked to specific deliverables of a distributed project;
- Project management tools, such as Gantt charts, offer greater visibility of the overall project developments and expectations amongst team members.

4.3 Software Architecture and Design Process

The Software Architecture and Design Process were concerned with the implementation of the requirements into software. Individual TRANSFoRm Work Task teams managed the architecture and design for the software they were developing but collaborated towards achieving the TRANSFoRm project goals. The software architecture was typically tested using a choice of web services. The testing process was often guided by industry trends whereby particular frameworks were chosen based on the availability of documentation and tool support for the TRANSFoRm Work Task team. The team also documented product component solutions which had recorded how a solution was discussed, developed and implemented. This would also indicate how capabilities of the team were aligned with the requirements whereby decisions were considered to be 'obvious', driven by the team's competence and the market's technological offerings. In addition, as part of the development

and design tasks, various requirements and standards (e.g. HTML5) would have influenced the key tasks. For example, in the various mobile developments, IOS apps differ from standards available some years ago and new standards would have influenced how the interfaces or security constraints were developed.

The TRANSFoRm Work Task team also described how the technical solution was an essential part of implementation of the requirements into software. The high-level architecture of the product was designed in the project scope while lower-level components of the design were aligned with specific software requirements using, for example, UML diagrams, class diagrams, interaction diagrams, design patterns chosen for implementation (see Figure 10).

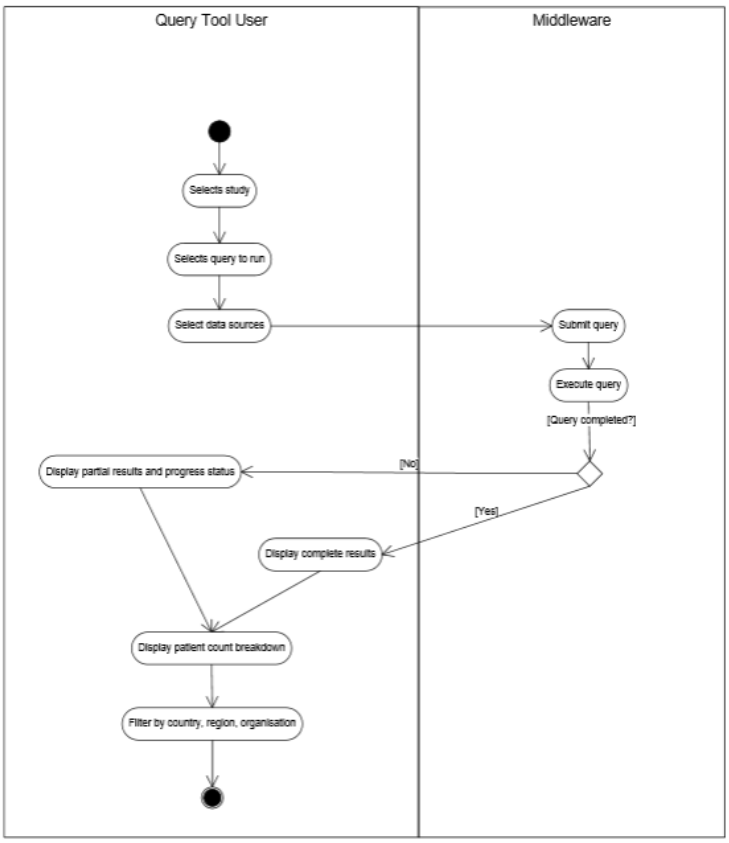


Figure 10 Typical Workflow for Running a Query

The presentation layer provides an Ajax web client implemented using JQuery (Figure 11). The Ajax application was delivered to clients browsers and supported by the Spring MVC framework. The Ajax client included a vocabulary search interface for users to quickly search the TRANSFoRm integrated vocabulary service. In order to reduce network latency and improve throughput, the Ajax client was designed to connect directly to the vocabulary service through its RESTful interface. The design was recorded in a '*technical data package*' that was used to document the architecture definition and provide the development team with a comprehensive description of the product. The technical data package was developed using a Java prototype (RESTful technology) since the infrastructure needed to be a synchronous solution and there was experience within the team using this technology.

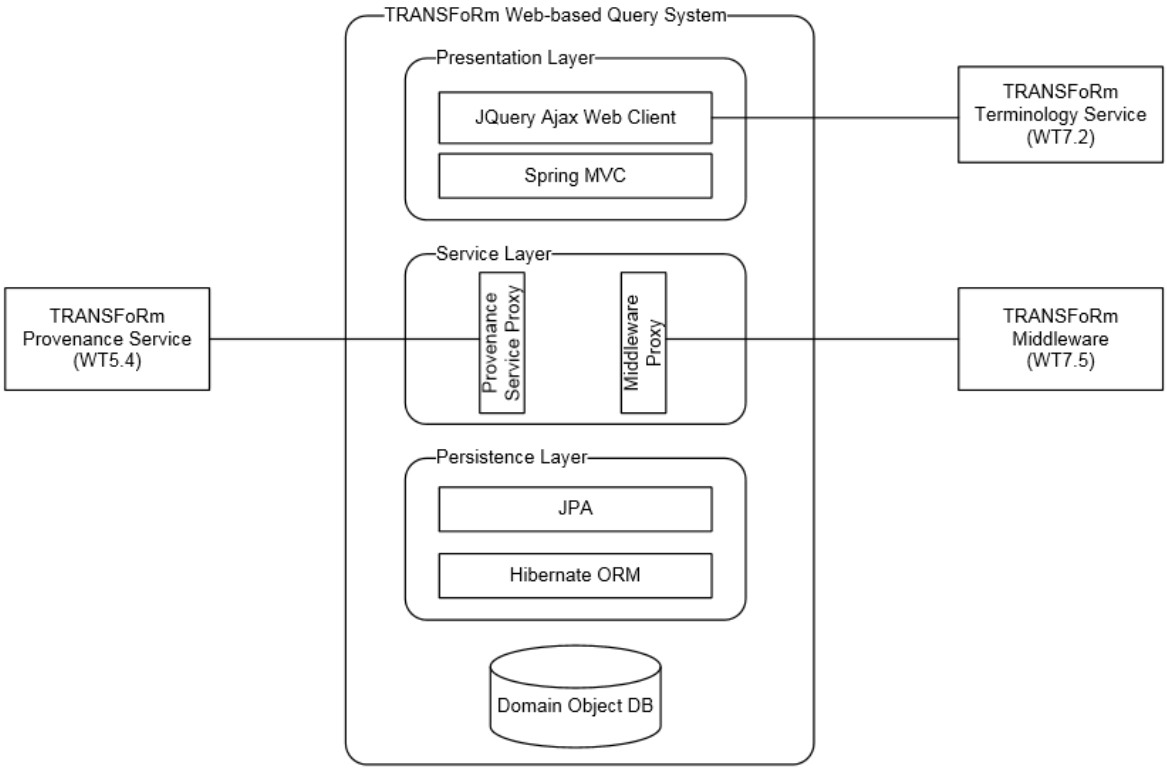


Figure 11 Layered Application Architecture

As part of the software architecture and design process, alternative solutions often emerged and options were often influenced by the availability of funding vs. cost, option of scalability, security and installation of local software components. Alternative solutions were considered and were categorised into high-level and low-level alternatives. High-level alternatives were overarching solutions which were considered but had no major change on the overall project. Low-level alternatives had a direct impact on the functionality of the project development and required some investigation and alignment with expertise. Alternative solutions were driven by meetings on various components. A history of the meetings was documented through the use of PowerPoint and documentation on different solutions. The decision of "*Make or Buy?*" was briefly considered during the design stages but considering the exploratory nature of TRANSFoRm, there was a lack of suitable software vendors with ready-to-use products to address the unique requirements of the project. The selection of the various software components were typically defined by the DoW. The component solutions were evaluated using various criteria such as key requirements, design issues and the availability of commercial off-the-shelf (COTS) solutions. Through the assessment of architectural features and solutions, COTS solutions were evaluated on whether they were 'fit for purpose'. The selection criteria to evaluate alternative solutions examined factors, such as:

- Cost of development;
- Time required to develop;
- Complexity of the solution;
- Ability to grow the solution;
- Technology limitations;
- Robustness in terms of privacy, security and other quality concerns;
- Risk involved;
- Dependence on 3rd party COTS solutions.

Depending on the time in the TRANSFoRm project when specific team members joined, they would have to typically comply with existing use of technologies and in some case lacked freedom to inject

new, e.g. specific open source tools and technologies, into the development process. There was an effort to use open source in order to encourage the reuse of these developments at both an EU and international community into the future. The TRANSFoRm Work Task team were conscious that they must assess the market for emerging technologies to ensure, where possible, that their technical development would be still 'relevant' into the future. It was also considered a positive by various reviewers that by minimising the number of operating systems, this would support the interoperability of the software products throughout the project lifecycle. Throughout the project, the design process influenced the technical implementation and provided a baseline for other activities, such as maintenance, procurement, modification and installation. A baseline defined a set of work products, components and specifications, which have were formally reviewed and agreed on for all future development and delivery within the TRANSFoRm project.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Offering team members a choice of testing tools can enhance the overall reliability of the architecture and design process;
- Adopting current industry standards and frameworks can support the documentation and guidance of architecture and design processes;
- Examining how team capabilities can be aligned with the project's requirements can mitigate the risk of failures within a project, thus requiring effective human resource management techniques;
- Identifying diagrammatic techniques to present conceptual requirements into technical solutions is a critical support to a distributed software development team;
- As part of the overall project learning process, it is important to be open to new or unexpected solutions using various categories of impact (for example, high, medium, low);
- Assessing commercial off-the-shelf (COTS) solutions should be considered as a means to survey existing solutions and best practice and considering how project resources may be best utilised.

4.4 Managing the Product Design and Coding Developments

The core purpose of coding was to enable the TRANSFoRm Work Task team to implement the identified requirements. As outlined above, these requirements were architected at a high level and designed at a lower level by the TRANSFoRm team. When the code was implemented, the team reflected and discussed the overall design whereby individual work task teams managed the software development for their work task.

The product design followed various software engineering industry standards and internal procedures of other good practice (based on previous or on-going projects within the team). As part of the management process, coding was aligned with the product design. Managing the design and coding developments allowed the team to reuse the expertise and technologies. Good practice included the SPREE[6] Framework (an open source e-commerce solution based on Ruby on Rails) was used to manage the product design and coding developments. The product component design comprised of two iterative key stages: *preliminary* (high-level based on functional and performance requirements) and the *detailed* design (defined developments such as coding algorithms). The TRANSFoRm Work Task team indicate that developments and coding of various components would have been achieved using prototypes, for example, web services to fulfil end-user needs. The prototype would be used to determine that its functionality met specific requirements based on rigorous peer-review processes. Therefore, in some cases, for specific elements of the code, peer-reviewing was prioritised over other elements. Coding was also tested by external competencies, interoperability capabilities and on the outcomes of other systems. One example of availing of external competencies included the use of

[6] https://spreecommerce.com/

hackathons. The hackathons were events set up by the TRANSFoRm team to invite technical people from outside the TRANSFoRm community to apply their computing skills to test the overall reliability and security of the TRANSFoRm components and system.

Our findings indicate that an over emphasis on technical detail could 'clutter' one's view of the overall goal of the project, i.e. losing sight of the high-level objective. Teamwork between the TRANSFoRm Work Task team proved to be critical throughout this phase. The product design and development underwent numerous peer-review and unit testing processes internally within individual partner teams. The peer-review process often enhanced the overall quality, particularly where crucial developments were required to support the advancement of the overall project. In addition, setting up a robust testing environment consumed some time especially as the team needed to run a SVN on one machine for testing purposes. The technical lead designed the basic architecture using diagrams to linked components and this demonstrated how various elements fit to the overall architecture. The TRANSFoRm Work Task teams at different locations used different systems which had to be connected to a central architecture to complete the project. Face-to-face meetings every couple of months were critical to provide a high-level abstraction of coding development. In addition, weekly teleconferences provided an important collaborative support link between the development teams to discuss issues and overall progress within the project. Key people, such as the PI's and senior researchers within the TRANSFoRm team provided an interdisciplinary 'bridge' between clinical and technical team members which proved to be crucial as a mediator and resolving conflicts on critical feature developments.

The design experience i.e. the user interface (UI) was not a process about which all the development team had experience. However, the TRANSFoRm Work Task team borrowed best practice design principles from other similar projects. Firstly, they evaluated similar projects (including functionality) which are available in the public domain and identified what could be implemented in the TRANSFoRm project, e.g. UI best practice. When the TRANSFoRm team explored various new functionalities, the question of "Make or Buy?" did surface from time-to-time. However, because of the novelty and exploratory nature of the TRANSFoRm project, software functionality which was not available on the market was required. However, it may have been possible to buy smaller components of the overall program. For example, consider the complexity of the Clinical Data Management System (CDMS). Such systems are available but there are many issues associated with acquiring a licence for such software that would prevent the "buying" decision from going ahead. Therefore, the option of buying software components was not considered a real option for the project's development.

Reviewing of code largely depended on the availability of resources. Resources also dictated how each team member was responsible for each component of the overall product. Documentation of the software quality was generated at various levels. For example, the various deliverables offered high-level explanations of the design and development. The team also developed a WiKi, a user manual and various publications to discuss and disseminate the more technical aspects of the software development. This supported the management of product design and code. For example, installation was a key part of the project and the migration of services was largely supported using the available installation documents. Many of the deliverables also served to document and support installation, configuration and maintenance tasks (for example, *D5.5* and *D7.3*). The documentation was kept up-to-date to reflect changes which were made in the requirements, design, software interface and functionality. In addition, when the design was established, the product or component implementation was a key process which included unit testing and peer-reviews were held prior to product integration and documentation. Various refinements and verifications were made to enhance the overall quality of the product. This supported both the learning process amongst the TRANSFoRm Work Task team and the overall success of the TRANSFoRm project.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Adopting industry standards ensures that a team follows best practice throughout the project lifecycle;
- Product component design can comprise of two iterative key stages: *preliminary* (high-level based on functional and performance requirements) and the *detailed* design (defined developments such as coding algorithms);
- Peer-reviewing is a key management activity of product design and coding;
- Adopting innovative techniques to avail of external competencies, such as conducting hackathons, can prove to be a valuable resource for a distributed software project;
- Eliminating much of the technical detail can improve a team's ability to understand the overall goal of a complex project;
- Devising methods to encourage teamwork within a distributed team proves to be critical throughout project lifecycle;
- Regular face-to-face meetings are critical to provide a high-level abstraction of coding developments;
- Teleconferences provide an important collaborative support link between the development teams to discuss issues and overall progress within the project;
- Ensure there are interdisciplinary team members who can act as the 'bridge' between different disciplines to act as a mediator and resolve conflicts.

4.5 Product Integration Process

Product integration was a key process for the TRANSFoRm Work Task team to collaborate and integrate products together, thus forming more complex components of the TRANSFoRm architecture and the entire TRANSFoRm system itself. Product integration also supported the TRANSFoRm Work Task team to validate and verify the internal and external interfaces. Rather than adopting a considerably more risky single integration stage, the team benefited from their use of an agile and iterative approach towards product integration.

Testing proved to be a key process when doing product integration. The TRANSFoRm Work Task team developed internal versions to test the integration process. Developers from teams other than the team who carried out the integration tested and commented on various components of the software whereby the integration was an iterative process. Within TRANSFoRM, the product components were integrated and tested to assess and assembly integration sequence, i.e. to identify the best integration sequence in accordance with the overall design. The integration sequence largely depended on the various components employed. For example, one tactic employed by the TRANSFoRm Work Task team was to commence at the interface level and go through the complete workflow to identify which components were causing issues. Addressing the issues became a task for individual developer as each component had been originally assigned to that person. Agreeing on the integration process (Figure 12) was time-consuming and decisions were documented.

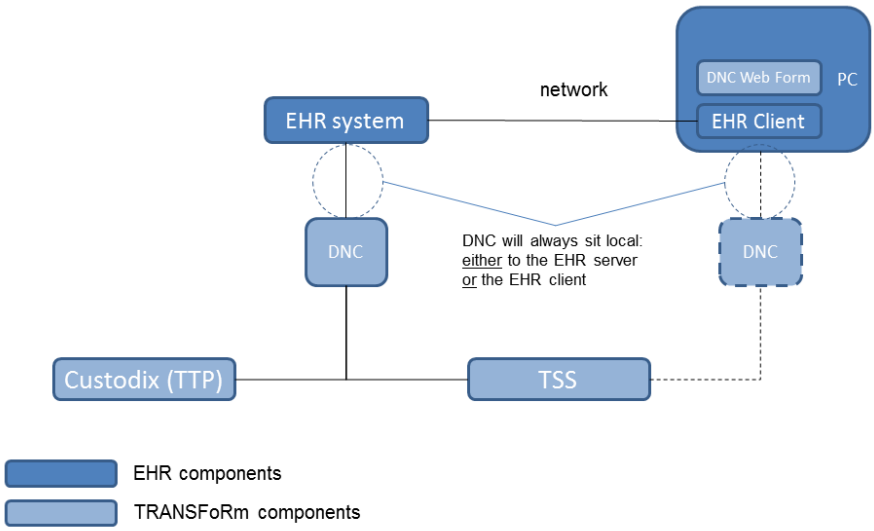


Figure 12 Relationships and Topology of Integration Process

Some of the TRANSFoRm Work Task team suggested that it may have been useful to have some form of integration software to test and identify problems throughout the whole workflow in a faster and/or automated manner. The integration testing was performed on the code-base at the staging level as the nature of the software produced necessitated integration with a number of other TRANSFoRm development teams. On acceptance of software coding at the staging level by partner teams, the SVN was used to push the same code to a production environment. This was only achieved through the recommended changes at a development level based on feedback received during integration. In all cases, any changes based on requirements were first tested at a development level before pushing to the staging environment for further integration testing. The DoW was considered a key resource which listed the TRANSFoRm requirements. However, the sufficiency of various requirements was tested using various prototypes which would have initiated discussions with developers and clinicians on the necessity for requirements based on their various functionalities.

The TRANSFoRm Work Task team also explained that there was no formal structure put in place for the product integration process but the team made progress in 'bursts' which was informally scheduled, often integrating components within short timeframes. To alleviate the spikes of integration, meetings were conducted to overcome potential problems associated with the integration process. While this was considered to be an informal process, a tacit agreement was put in place to indicate when the integration would occur. A formal process which provides a specific procedure or criteria would have been of benefit to the integration process within TRANSFoRm. The TRANSFoRm Work Task prioritised product independence and integration based on other development task scheduling across the wider TRANSFoRm team. For example, as part of the integration process, some team members would examine the inputs and outputs to determine how the structure 'fits' within the structure of other components. Thus, the integration process required both a technical perspective but also, considering the interdisciplinary nature of the TRANSFoRm project, a holistic view and awareness of other on-going developments within the TRANSFoRm Work Task team. To support the Work Task team in gaining a holistic perspective of the project, weekly teleconferences on 'neighbouring tasks' during the integration process provided a shared vision of this process. The meetings also informed the team of development readiness and where changes were required. This enabled the team to be proactive and identify potential issues before the integration process. In addition, the hackathons proved to be another good source of external feedback on the integration readiness which enabled the TRANSFoRm Work Task team to identify problems quickly.

Interface compatibility was expected to be a complex task, for example, using the final product across a number of browsers did initially cause an issue. There were a number of technical solutions to

overcome this (for example, implementing JQuery for increased compatibility) which required the need for an online browser compatibility testing environment. To test the browser compatibility became a task for the UI team as part of the integration process. There was much discussion with various parties within the team on methods, prototypes and testing techniques on key element of the interface compatibility both the front-end interface and back-end functionality to align development efforts across the team. As part of the integration process, it was important to assess readiness. To confirm readiness of the overall project, the TRANSFoRm Work Task team used pilot tests to examine the various functionality and feedback on the overall system. This also formed part of the validation and verification process. Readiness was also 'confirmed' using methods to simulate the workflows and comparing results against the 'ideal' state (based on the requirements). Teams tested readiness internally and compared results against the requirements (which were shifting throughout the project lifecycle). Therefore, to avoid ambiguity, it was important to clarify requirements from time-to-time. In summary, the overall design of the interface was described as an organic process which emerged through various discussions with partners and evolving changes in requirements. Collaboration and openness amongst the TRANSFoRm Work Task team proved to be a critical success factor for the product integration process.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Product integration supports a distributed team to validate and verify the internal and external interfaces within a final product;
- Identifying the best integration sequence in accordance with the overall design is a key task before the product integration process commences;
- Exploring whether suitable integration software may be used to test and identify problems throughout the whole workflow may prove to be more efficient in a distributed team environment;
- Establishing feedback mechanisms during the integration process is important to identify required changes to ensure software is successfully integrated;
- Implementing a formal product integration process structure may reduce uncertainty and increase resource efficiency if integration processes and timeframes are clearly established;
- Identifying software compatibility tools and techniques as part of the project strategy can improve readiness assessment and increase the success of product integration throughout the project lifecycle;
- Collaboration and openness amongst the distributed team proved to be a critical success factor for the product integration process.

4.6 Managing Quality Assurance (Verification & Validation)

The quality assurance process provided the TRANSFoRm Work Task team with the knowledge that the software products and components which were built by the team met the specified requirements. Testing proved to be a core element of the software product development lifecycle for the TRANSFoRm Work Task team. For example, some of the team members explained that while they had adopted an agile software development methodology, all software was tested with existing software at the development stage before it is pushed to Apache Subversion (SVN), particularly for integration testing. Comments on each commitment to the SVN were required for all changes, no matter how small. The SVN was used as a software versioning and revision control system distributed to maintain current and historical versions of files such as source code, web pages, and documentation.

Verification, which took place throughout the lifecycle, ensured that selected work products met their specified requirements and was used to identify corrective action. The team used live testing, for example, within the GORD validation study in Poland, using five GP practices, from three GP practices and 10 patients which enabled them to gather and extract data. Therefore, verification was a

key part of each component during which its functionality was verified against requirements and how they integrated with the middleware. In addition, verifying that the Query Workbench was fully functional was a key part of the project. Peer-reviewing was also a key process in the verification stage of the Software Quality Plan. Methods of verification included inspections, peer reviews (for code, design or architecture), simulations and testing. The main source of verification criteria were the product and component requirements identified by the TRANSFoRm research team. Within the TRANSFoRm Work Task team, individual members participated in the peer review, preparing and updating materials, review criteria and checklists and schedules to successfully improve the various outcomes of the project. In addition, validation was key to ensuring that the product was fit for purpose. During the validation phases of the selected work products, validation was based on how well the work products met the user needs. Therefore, requirements were also validated at various stages of the requirements development process to remove any potential inconsistencies and to ensure efforts remain on the right track to meet stakeholder needs and expectations. This was achieved using simple analysis, simulation, prototyping and demonstration tools. The TRANSFoRm Work Task team analysed and validated requirements based on a "*discuss-build-test*" structure which enabled their team to easily adapt to change based on the requirements. This simplified the validation process to ensure the right product was designed to work the way it was intended to and overall streamlined the quality assurance management process.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Testing proved to be a core element of the software product development lifecycle;
- Software versioning and revision control systems within a distributed environment proved to be important to maintain current and historical versions of developments;
- As part of the verification process, live testing proved to provide critical feedback on the software functionality;
- Peer-reviewing was also a key process in the verification stage;
- To adapt to any inconsistencies in the software products, requirements were assessed using a "*discuss-build-test*" structure which enabled their team to easily embrace change.

4.7 Configuration Management Process

As the proposed software development projects in TRANSFoRm are widely distributed across several different teams in Europe, several layers of complexity were added into the process. Configuration management addressed some of the complexities involved in managing the complete project, i.e. maintaining the integrity of the work products. Individual teams within TRANSFoRm managed the configuration of their own software and documentation process. Appropriate source control software was used to manage configurations of the software.

The configuration management process supported the configuration identification, configuration control, configuration status accounting and configuration audits to ensure it all products and process were in place and functions as identified by their requirements and its 'intended use'. Configuration could be categorised into 1) code and prototype configuration and 2) software configuration, enabling developers to track and control changes using the SVN. Tracking components of the system allowed the TRANSFoRm Work Task team to monitor how the system functioned and met specific requirements. The TRANSFoRm team established a baseline which enabled them to manage the evolution of the product. While change was not formally documented, it was clearly traceable from documentation of various meetings and change proposals. To support the configuration management process, the team comprised of software development and testing members. To test components of the product, a request was sent typically out to the development team. The development of any software was documented using various templates, and was shared across the TRANSFoRm team using existing tools, such as Dropbox. This played a key role in the configuration process. The evaluation and implementation of the software was then supported using these documents. Since there were

many TRANSFoRm Work Task team members, it was challenging to coordinate all of the configuration processes across all of the groups.

For example, referring back to *D3.5 Software Integration Plan*, we can examine how TRANSFoRm was described as a digital infrastructure organised around three configurations: Epidemiological Study (supporting the Diabetes use case), Randomized Control Trial (supporting the GORD use case), and Diagnostic Support (evaluated with the industrial partners). These three configurations were supported with an updated list of all planned software components and descriptions. Since TRANSFoRm is a model-based system, the underlying models were also described. The three basic configurations of the tools were evaluated in the project. For example, the Epidemiological Study configuration (Figure 13) was used in the Diabetes use-case. It consisted of tools and frameworks for secure, provenance-enabled design and execution of epidemiological studies. This included the use of query design, for phenotypic and genotypic data retrieval from heterogeneous data sources. As part of the configuration, queries were formulated using standardised elements and using information about suitable practices obtained from the data quality tool. These queries were sent to the data sources using a secure data transport mechanism that communicates with TRANSFoRm connector tools on the data source side.

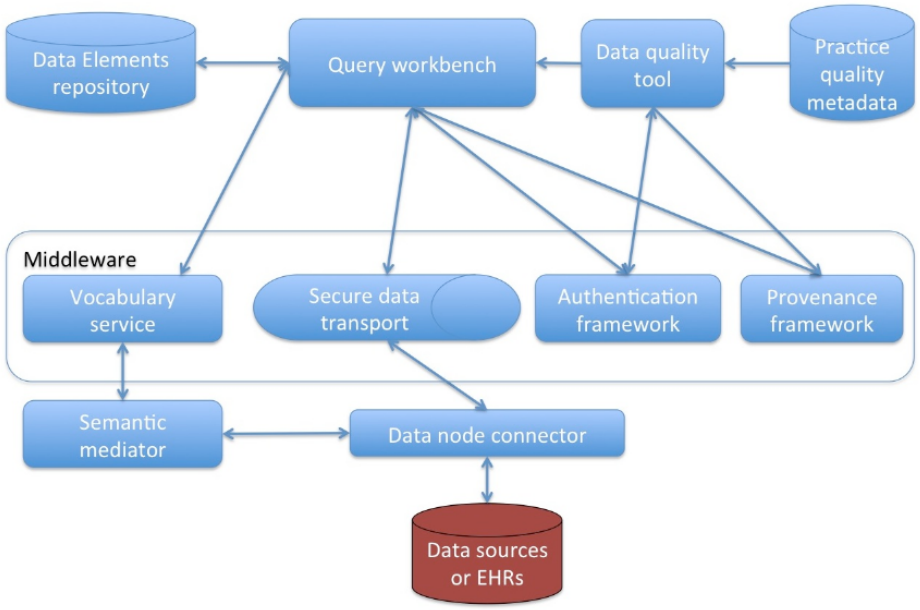


Figure 13 Epidemiological Study Configuration

Configuration management was successfully achieved through clear lines of communication and testing whether the product provided an end-to-end solution. A formal process, such as an agile approach, may have been useful although it was not considered necessary for the successful delivery of the TRANSFoRm solution. One of the reasons for this included the reliance placed upon software experience, skills and competencies rather than operational strategies or procedures. Changes in the configuration were typically captured using email and various meeting minutes. A mechanism to formally track changes would have added greater traceability or visibility during the various phases of the project development.

After confirming the quality of newly developed software in a development environment, the SVN was used to push all changes to a staging environment. This was considered the first "public" showcase of any work and a repository for all stable code. No changes were made to software at the staging level but continued development of software components was allowed at the development level. From a TRANSFoRm projects' perspective, there were key items which were included in the deliverables for other team members. For example, establishing configuration items, identification and selection of work products, assigning unique identifiers to each of these items and identifying the

owners responsible for each of the items were important characteristics of each item. In addition, integrity was established by employing the initial criteria used for baseline identification and then tracking and controlling various changes. To ensure that the system was being correctly utilised, periodical configuration audits were performed within the team, issues discussed at meetings and configuration assessed using various design and requirements documents. All of these efforts within the TRANSFoRm Work Task team ensured that the Software Quality Plan played a key role in the configuration management process and, ultimately in the success of TRANSFoRm.

Some of the key implications from this phase of the Software Quality Plan in TRANSFoRm which should be considered for future projects include:

- Distributed teams of various competence and capabilities can add new layers of complexity to both project management and software development;
- Tracking configuration management processes enables a distribute team to identify how the system functions and monitor whether specific requirements are being met;
- The utilisation of collaborative tools, such as Dropbox, does support the configuration process by engaging all the team and sharing results in the process;
- Coordinating all of the configuration processes across the distributed teams/groups is a challenging task;
- Configuration management was successfully achieved through clear lines of communication and testing whether the product provided an end-to-end solution;
- Implementing a mechanism to formally track configuration changes would add greater traceability or visibility to the project development;

5. Lessons Learned

The underlying concept of TRANSFoRm was to develop a 'rapid learning healthcare system' driven by advanced computational infrastructure that would improve both patient safety and the conduct and volume of clinical research in Europe. To achieve this, the Software Quality Plan played a critical role.

Similar to the work of Derntl et al. (2015), we also discovered that it is a major challenge to establish a developer community that align requirements and development processes with other stakeholders. While projects such as TRANSFoRM are heavily research-orientated and attempting to break new ground, the exploratory nature of new software developments in a unique healthcare context exposes a significant risk of failing. Common challenges include introducing new partners to one another and developing new working relationships, exploratory nature of research, politics between working groups, decision-making and empowerment, visibility and transparency of activities, impact of development on other tasks and the overall project, availing of support competencies and tools, industry commitment, staff retention, and availing of correct expertise within the team. Our research indicates that the project team's collaborative effort to ensure success indicates a significant human commitment rather than a technological know-how. Thus, going forward with other large EU projects, establishing project management and collaboration structures will play a significant role in the success of distributed software development projects.

This report examines the various software quality processes throughout the TRANSFoRm project and highlights how they supported the success of this project. While the results of the TRANSFoRm project are clearly documented across various deliverables and software development products, there are a number of 'lessons learned' which should be shared. Initially, there were difficulties associated with the coordination of the project across the dynamic and interdisciplinary team environment. Changes to staff retention presented some challenges in the university environment – all of which reflected reality either in industry and/or academic environments. It is evident that there is a need for all teams to meet on a social level to introduce all team members to one another and establish a working relationship. This ensures that team members are familiar with one another and strengthens

the level of trust and collaboration across the team. A recommendation would be to improve mechanisms for communication and documentation within EU projects. However, research indicates that this is a complex task. For example, Richardson et al. (2012) identify the pressures on software to successfully manage teams in a global environment. They present the Global Teaming (GT) process to address specific problems relating to temporal, cultural, geographic and linguistic distance which will meet the complex and changing needs of global software management. Such approaches would also help to reduce misunderstandings between Work Task team across Europe. It would also be important to integrate all partners as soon as possible in the early stage of the project's lifecycle. Since this team worked within a distributed environment, there were many cultural factors to consider including language and interpretation of colloquial language. The nature of TRANSFoRm was a rapidly changing environment often requiring strong project management and time management skills to adapt accordingly. A distributed working environment can give a sense of a lack of transparency on progress of other teams, and this was also the case within TRANSFoRm when considering the DoW requirements. In addition, the requirement gathering process was a critical process which requires a formal process to structure and document the stakeholder requirements. Various market standards and the skills of the TRANSFoRm Work Task team proved to be a critical influential factor in the development of TRANSFoRm. This would directly support and improve change management factors within an exploratory and complex EU project, such as TRANSFoRm, and document decisions being signed-off on from a holistic perspective. The Software Quality Plan proved to be a critical resource for the TRANSFoRm Work Task team to employ and guide key processes through the software development lifecycle and similar quality plans should be incorporated in projects at this scale, i.e. on an EU level. Based on the findings, we have identified a number of recommendations for future research projects operating within a distributed environment. Of particular importance, and an underlying theme throughout the findings, is the need for a work package to set up the software development strategy and management policy within a project. This needs to be brought a step further than the Software Quality Plan whereby people have access to a centralised technical and collaborative infrastructure which is managed on an on-going basis. As a requirement of EU projects, the work package should be introduced to set up the development processes. While the Software Quality Plan did support the development processes, there needs to be a strong collaborative effort to establish a clear strategy of how to achieve the projects goals as part of the proposal and initial phase of the project.

6. Conclusion

The TRANSFoRm Software Quality Plan outlined the processes which should be undertaken so that software output from this project will be commercially exploitable. In this review report, we summarise our findings of the semi-structured interviews to examine how key elements of the quality plan were employed. Our results indicate that the Software Quality Assurance Framework proved to be a core resource for the TRANSFoRm project which provided a common goal and vocabulary for all of the TRANSFoRm team to focus on throughout the software development process. Through the use of the Software Quality Plan, TRANSFoRm could successfully meet the project objectives and maintain a high quality of standard in capturing clinical data, achieve a distributed interoperability of data and developed unique software tools and services. Our findings also indicate that additional emphasis must be placed on strategy, not technology, at the beginning of a distributive software project since this ultimately drives the digital transformation and ensures that technical milestones are met.

References

Derntl, M., Renzel, D., Nicolaescu, P., Koren, I., & Klamma, R. (2015). Distributed Software Engineering in Collaborative Research Projects. In Proceedings of 10th International Conference on Global Software Engineering, ICGSE 2015. Los Alamitos, CA: IEEE

Dukic, L.B., Boegh, J.: COTS Software Quality Evaluation. In: Erdogmus, H., Weng, T. (eds.) ICCBSS 2003. LNCS, vol. 2580, pp. 72–80. Springer, Heidelberg (2003)View Article

Kalaimagal, S., Srinivasan, R.: A Retrospective on Software Component Quality Models. SIGSOFT Software Engineering Notes 33(5) (November 2008)

Richardson, I., Casey, V., McCaffery, F., Burton, J., & Beecham, S. (2012). A process framework for global software engineering teams. Information and Software Technology, 54(11), 1175-1191.

Appendix A: Semi-structured Interviews

Overview of Questions

Describe the overall Project Management Process: the experience?

Describe How You Managed Requirements Development Process?

- What is your experience with the Requirements Development Process?
- How did you develop customer requirements?
  a. How did you elicit the various needs for various stakeholders?
  b. How did you manage to pool/categories all of the various requirements?
  c. How did you (if any) resolve conflicts during the requirements gathering?

- How did you manage requirements and remain consistent throughout the project?
- How did you obtain an understanding of the various requirements: what criteria did you use?
- How did you obtain commitment to requirements: what agreement strategy was used?
- How did you manage requirement changes throughout the lifetime of the project?
- How did you maintain bi-directional traceability of requirements (i.e., from source to higher detailed requirements)
- How did you identify inconsistencies between project work and requirements?

Describe How You Developed the Product Requirements?

- How did you establish the product/product components requirements?
- How did you allocate product component requirements (with project timelines etc.)
- How did you identify various interface requirements?
- Describe how you analyses and validated the requirements
- Did you establish operational concepts and scenarios?
- How did you establish a definition of required functionality?
- How did you analyse the various requirements to ensure they are necessary and sufficient?
- How did you achieve a balance between requirements – between functionality and necessity within a specific timeframe and budget?
- How did you validate requirements at various stages of the development?

Describe the Architecture and Design Process.

- How did you select product component solutions?
- How did you develop alternative solutions and selection criteria? What factors influenced decisions (e.g. cost, time, complexity, ability, technology limitations, risk etc.)?
- How did you select product component solutions and how was it documented?
- What criteria were used to support the development and design of the product(s)?
- How did you establish a technical data package to define the architecture for the development team?
- What criteria were used to support the design of the interface(s)?
- What evaluation process did you use to perform a “make or buy” analyses?

How did you Manage the Product Design and Coding Developments?

- How did you implement the product design (e.g. peer reviews of codes, unit testing, refactoring?(?
- How did you develop the product support documentation to support installation, operation and maintenance?

Describe the Product Integration Process.

- How did you prepare for product integration?
- How did you determine the integration sequence?
- How did you establish a product integration procedures and criteria?
- How did you ensure interface compatibility (completeness, manage interfaces, assemble product components?)
- How did you confirm readiness of the products components for integration?

How did you Manage the Product Verification Process?

- How did you prepare for verification?
- Did you preform peer reviews? How did you select products for verification?

How did you Manage the Configuration Management Process?

- How did you establish baselines to ensure a useful configuration?
- How did you track and control changes?
- How did you establish integrity?
- Did you perform configuration audits?

Other general comments: lessons learned?

Appendix B: Summary of Checklist Feedback

Requirements Development Checklist

Colour coding of findings:

Full compliance from all teams	Some compliance from teams (depended on primary role of team)	No compliance from teams (was not considered necessary by teams)

When **Requirements** have been developed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are stakeholder needs being recorded?	Y	Y	Y	See Description of Work and GORD Protocol	
Have business goals being documented?	Y	Y	Y	See Description of Work	
Have any relevant legal requirements been elicited?	Y	Y	Y	TRANSFoRm Study System (TSS) had to meet requirements of Bioethical Committee in each country where it will be used. Additionally TSS had to meet Good Clinical Practice (GCP) requirements	
Is there any missing information which at the end of the requirements consolidation which needs to be addressed? There should be none.	N	N	N		
Are all conflicts resolved, and the decisions and reasoning documented?	Y	Y	Y	Although this is done informally a lot of the time, mostly over email chains. It is an on-going process considering EHR vendors which will be running the GORD Study. At this point all conflicts were documented and resolved during the GCP Validation and Test Study.	
Are requirements being written in technical terms necessary for product and component design?	N	Y	Y	Some are technical and others are more high level. See Technical Overview of GORD use case documentation	
Are requirements derived from design and architectural decisions being recorded?	Y	Y	Y	All of the requirements were delivered to the development team in form of notes, minutes, documents created/presented on Face-2-Face (F2F) meetings	
Is there traceability between customer requirements and product requirements?	Y	N	N	There is no specific documentation for that. It is possible to trace that based on documentation related to various part of the project	
Are the new requirements managed in a change management and requirement allocation process?	N	N	Y	We have an internal change management process but there is no formal process across the whole project. Not suitable for our case a more informal approach was used. We are using applications like Trello, Redmine.	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are all the requirements allocated to different functions?	Y	Y	Y	See deliverable 7.1, 7.3 and 5.5	
Are all the requirements allocated to product components?	Y	Y	Y		
Are design constraints allocated to product components?	Y	Y	Y		
Are any derived requirements allocated appropriately?	Y	Y	Y		
Are the relationships between all the allocated requirements documented?	Y	Y	N	There is no formal documentation for that	
Is there clear dependencies highlighted where changes in one requirement may affect other requirements?	N	Y	N	There is no formal documentation. Teams talk to each other about dependencies. These are highlighted but not all in one place, emails used a lot.	
Are all internal interfaces identified?	Y	Y	Y		
Are all external interfaces identified?	Y	Y	Y		
Are all identified interfaces developed into requirements?	Y	Y	Y		
Are different possible operational concepts being defined?	Y	N	Y	Not applicable for our components	
Are different usage scenarios being defined?	Y	N	Y	Not applicable for our components	
Are these looking at functionality, performance, maintenance, support and disposal?	Y	N	N	Development focused more on functionalities than anything else. It was the biggest challenge in this project	
Is the environment (including borders and constraints) in which the system will operate defined?	Y	Y	Y		
Are the operational concepts and scenarios refined based on chosen solutions and reviewed and accepted?	Y	N	Y	Not applicable or suitable in our case.	
Is the functionality required by the end user quantified and analysed?	N	Y	Y	Difficult to quantify See deliverable 7.1, 7.3, 5.5 and Technical Overview of GORD use case documentation	
Are the requirements analysed and subdivided into logical or functional	Y	Y	Y		

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
partitions?					
Are any time critical functions being analysed for sequencing within the project?	Y	Y	Y		
Are the customer requirements allocated to functional partitions, objects and people?	Y	Y	Y	Our customers are other TRANSFoRm components	
Are the functional and performance requirements being allocated to functions and sub-functions within the design solution?	Y	N	Y	Were allocated to this degree	
Are any conflicts in requirements analysed?	Y	Y	Y		
Are the requirements organised into related areas?	Y	Y	Y		
Are a set of requirements analysed to see if they satisfy the objectives of the higher level requirements for that set?	Y	Y	Y		
Are the requirements complete, feasible, realisable and verifiable?	Y	Y	Y		
Are key requirements which may have a large impact on cost, budgets, timeliness, risk or performance identified?	Y	Y	Y		
Are technical performance measures used to track progress identified?	N	N	Y	No global measures defined in the project Functional performance used instead Key parts of the software are monitored	
Are the organisational concepts and scenarios analysed and the customer's needs, constraints and interfaces refined and new requirements developed?	Y	Y	Y		
Has a risk assessment been carried out on all the requirements?	Y	Y	Y	Not formally documented but risk discussed and understood	
Has a risk assessment being carried out on the functional architecture?	Y	Y	Y	Not formally documented but risk discussed and understood	
Is there a document recording requirements which has been identified as having a	Y	N	N	Not done as part of our team Main focus was put on time constrains because of complexity of created software and integration of	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
significant effect on project costs, timeliness and performance?				various components	
Is there documented analysis that uses proven models to define the balance of stakeholder needs and constraints?	N	N	N	No formal models adopted in the project Not suitable for our project team task Stakeholders needs were changing continuously. All work was done within constrains presented by the stakeholder	
When validating the requirements, are they being checked for adequacy and completeness?	Y	Y	Y		
Are requirements being analysed to determine risk that the resulting product will not be appropriate for its intended use?	Y	Y	Y		

Requirements Management Checklist

When **Requirements** have been completed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are there selection criteria for evaluating who is a valid requirements provider?	Y	N	Y	Not formally specified. Was somewhat organic surrounding who our components interfaced with.	
Are there evaluation and selection criteria to determine what a valid set of requirements are?	Y	N	Y		
Are all agreed to set of requirements being recorded?	Y	Y	N	Requirements are still changing because of system integration with new EHR vendors. See documentation for the Data Node Connector	
Are all requirements being assessed and assessments documented?	N	N	N	No formal documentation about assessments Assessments not documented.	
Are all changes to requirements as part of assessment documented?	N	Y	N	No formal documentation about assessments Email chains etc.	
Is commitment to requirements agreed on and documented?	Y	N	N	Was not our experience on the project	
Are all agreed-to	Y	Y	Y	See Description of Work,	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
requirements being recorded?				deliverable 7.1, 7.3, 5.5 and Technical Overview of GORD use case documentation	
Are all changes to existing requirements being recorded?	Y	Y	Y	There is no formal documentation for this. Minutes, notes, email conversations do have that included	
Are decisions made for requirements being recorded?	Y	Y	Y	There is no formal documentation for this. Minutes, notes, email conversations do have that included	
Is the status of the requirements being recoded?	Y	Y	Y	There is no formal documentation for this. Minutes, notes, email conversations do have that included	
Is traceability amongst all requirements, can every low-level, decomposed requirement be traced back to its original source requirement/change request?	Y	N	N	Not suitable for our work on TRANSFoRm	
Can the different work products of a project be traced back to its source requirement, be it a design doc, piece of code, or test cases?	Y	Y	Y		
Can requirements be traced to the people working on and project tasks created for those requirements?	Y	Y	Y		
Is there a requirements traceability matrix?	Y	Y	N		
Are there inconsistencies between requirements and projects/work products being recorded? Do these records include the source and cause of the inconsistency?	Y	Y	Y	Again not formally but recorded in email trails etc. See GCP Validation and Test Study documentation	
Are corrective actions being recorded for inconsistencies found?	Y	Y	Y	See GCP Validation and Test Study documentation	
Are there changes being initiated in the project/work products to overcome these inconsistencies? Is this data being provided to the appropriate parties?	Y	Y	Y	See GCP Validation and Test Study documentation	

Architecture & Design Checklist

When **Architecture & Design** has been completed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are there screening criteria for alternative solutions?	Y	N	N	Not suitable for our task	
Are there formalised evaluation criteria for alternative solutions?	Y	N	N		
Are there reports for alternative solutions proposed and COTS solutions evaluated?	Y	N	N	One of the requirements was that software must be created without use of COTS	
Are the selection decisions documented?	Y	N	Y	Not applicable in our case See deliverable 7.1, 7.3, 5.5	
Has traceability between requirements and selected product component solutions been identified?	Y	Y	Y		
Have low level requirements and interface requirements been identified and documented after the selection of a solution?	Y	Y	Y		
Are there criteria against which the design and architecture can be evaluated?	Y	Y	Y		
Are there design standards and criteria to which the design must adhere to?	Y	N	Y	No, not hard standards. Criteria was based on needs of other teams.	
Is the design documented?	Y	Y	Y		
Is there a traceability matrix between requirements and design?	Y	N	N		
Are the detailed design descriptions based on requirements, architecture and high level designs?	Y	Y	Y		
Is the rationale for major decisions documented?	Y	Y	Y	Email and meeting minutes, presentations	
Are interface criteria identified?	Y	Y	Y		
Is there a matrix documenting all interfaces with internal and external product components?	Y	N	N		
Are the interface designs and rationale for the	Y	N	Y	These were devised based on discussion	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
design documents?					
Are there documented criteria for the reuse of component designs?	Y	N	Y		
Are there criteria identified for make or buy analysis, taking long-term maintenance into consideration?	Y	N	N	Long term maintenance is taken into consideration. Make or buy decisions are left out since software development has ended when it comes to incorporating new functionalities in the software	
Is there documented rationale for the decisions to make, buy or reuse?	Y	N	Y	See Description of Work, deliverable 7.1, 7.3, 5.5 and Technical Overview of GORD use case documentation	

Coding Checklist

When **Coding** has been completed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are design patterns being utilised?	Y	N	Y	Not formally done	
Is the code being documented?	Y	Y	Y	Everything is documented through comments inserted into the code	
Are unit tests written for all the individual units of code?	Y	Y	N	For significant units	
Is the code being reviewed by peers?	N	N	N	No enough resource Not suitable for a team of our size	
Is the implementation revised based on unit tests and code reviews?	Y	Y	N		
Are there documentation standards identified and utilised?	Y	Y	Y		
Are the documents being reviewed by stakeholders?	Y	N	Y	See Developer Assessment documentation	
Is the documentation being kept updated to reflect changes in the product?	Y	Y	Y	There is a Change Control document maintained	

Roles and Responsibilities Checklist

If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are the components to be integrated identified?	Y	Y	Y		
Are the tests to be performed during integration identified?	Y	Y	Y		
Are there alternative product integration sequences identified?	Y	N	N	N/A on the project	
Is the product integration sequence revised since the previous iteration?	Y	N	Y	N/A on the project	
Are the decisions being made and their rationale documented?	Y	N	Y	Through non-formal documentation. Minutes, notes, presentations, email conversations	
Are the requirements for product integration environment identified?	Y	Y	Y		
Are there verification criteria and procedures for the product integration environment?	Y	Y	N		
Are product integration procedures established and maintained?	Y	Y	Y	Yes but not formally done. Tacit agreement with other teams	
Are product component integration and evaluation criteria established and maintained?	Y	Y	Y	Same	
Are validation criteria for delivery of the integrated product established and maintained?	Y	Y	Y	Yes, although this simply resolved around the ability of a query and result to flow See GCP Validation and Test Study documentation	
Is the interface data complete and ensures coverage of all interfaces?	Y	Y	Y		
Are the interfaces updates to remain compatible with any changes, non-compliances or conflicts	Y	N	Y	Not our experience	
Is the evaluation of assembled product components documented?	Y	Y	Y	See GCP Validation and Test Study documentation	

Verification Checklist

When Verification has been completed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Is there a traceability matrix to identify requirements for work products?	Y	Y	N		
Are the verification methods utilised appropriate for the product?	Y	Y	Y		
Are the verification environment, equipment and tools adequate for testing and reflective of real usage of the products?	Y	Y	Y		
Are there verification criteria created to ensure that all requirements are tested in different ways	Y	N	Y	Just the way they were intended to be used. Perhaps an oversight.	
Are there expected results recorded for all the different tests and criteria?	Y	N	Y	Not formally.	
Are there requirements for data collection at the time of the peer review?	Y	N	Y	Peer review not done so the next few are N/A	
Are there entry and exit criteria for a peer review?	Y	N	Y		
Are there criteria to determine is a subsequent peer review needs to be done?	Y	N	Y	See GCP Validation and Test Study documentation	
Are there checklists for what needs to be reviewed during a peer review?	Y	N	Y		
Are there defects being identified based on checklists and documents?	Y	N	Y	This is done based on testing	
Is the peer review data being recorded?	Y	N	Y	N/A	
Are there actions identified and communicated appropriately?	Y	N	Y	N/A	
Is the review data being analysed?	Y	N	Y	N/A	
Are there medium-term and long-term actions identified to provide continuous improvement?	Y	N	Y	N/A See GCP Validation and Test Study documentation	
Is verification of products taking place against the product requirements?	Y	N	Y	N/A for the middleware components	
Are verification results being recorded?	Y	Y	Y	Results of test communicated to other Transform members	
Are actual results being compared to expected	Y	N	Y	N/A in our case	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
results to determine if criteria are being met?					
Are action items being generated as required from verification procedures?	Y	N	N	N/A in our case	
Is information being provided on the defects were detected, logged and possible solutions on how it could be resolved.	Y	N	Y	No, a more informal process used.	

Validation Checklist

When Validation has been completed, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are key phases of validation for each product or component identified?	Y	Y	Y		
Are the validation requirements, constraints and methods reviewed by stakeholders?	Y	Y	Y		
Are third party software being validated?	Y	N	N	Not applicable It is outside of the scope of the project	
Are validation environment requirements identified?	Y	N	Y	Not formally done See GCP Validation and Test Study documentation	
Are tools and test equipment identified?	Y	Y	Y		
Are validation resources which can be reused identified?	Y	N	N	Not applicable to us	
Is there a detailed availability plan for resources, if resources are not dedicated?	Y	N	N	All of the resources are dedicated. There is documentation related to resources used	
Are the validation environment, procedures, operational environments and criteria for validation documented?	Y	Y	Y	Mostly done	
Are subsequent changes in requirements and design being reviewed for	Y	N	Y	Not on the process itself no	

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
possible impact on validation processes?					
Are products and components performing according to validation criteria?	Y	Y	Y		
Are the results being recorded?	Y	Y	Y	Yes as part of bigger validation	
Are the results analysis and issues identified being documented?	Y	N	Y	Not as part of our team	

Configuration Checklist

When the Configuration Management system has been set up, the Work Task leader should answer each of these questions on this checklist. If the answer to any question is 'No', the Work Task leader must give a short explanation as to why.

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
Are the selection criteria for configurable items documented and items identified?	Y	N	Y	Not applicable for our team	
Are there unique identifiers and specific owners assigned to the configuration items?	Y	N	Y	Not applicable for our team	
Are there different levels of control and mechanism to manage them within the configuration management system?	Y	N	Y	Not applicable for our team	
Can you store and recover archived versions of configurable items in this system?	Y	N	Y	Not applicable for our team	
Is the baseline created purely from previously identified configuration items?	Y	N	Y	Not applicable for our team	
Are the configuration items within the baseline documented?	Y	N		Not applicable for our team	
Can you ensure that the current baseline readily available?	Y	Y	Y	Assuming this refers to the MW components, yes	
Are all change requests being documented?	Y	Y	Y	But again not a formal process in place	
Are all the change requests analysed for impact?	Y	Y	Y		
Are relevant stakeholders involved in the analysis and	Y	Y	Y		

Question on Checklist	Team A	Team B	Team C	Explanation (if any)	Overall rating
acceptance decisions of change requests?					
Are change request acceptance decisions documented with reasoning?	Y	Y	Y	Email chains etc	
Are configuration items checked-out and checked-in appropriately?	Y	N	Y	N/A	
Are changes to the baseline and configuration items being recorded?	Y	N	Y		
Is the current content and status of each configuration item known?	Y	N	Y		
Can a previous version of a configuration item be recovered?	Y	N	Y		
Can the differences between subsequent baselines be described?	Y	N	Y		
Is the integrity of each baseline evaluated?	Y	Y	Y	Using a query life-cycle test	
Is the configuration management system being used in accordance with defined criteria?	Y	N	N	There was no configuration management system planned and it is not mentioned in the requirements	
Are the controls put in place for tracking and controlling changes being followed and documented?	Y	N	Y	Not applicable or suitable in our context. There is a Change Control document maintained	