From reference model and system requirements to architecture design recommendations: an expert system approach

M.Sc. Thesis
Carlos Acosta

July 2017

From reference model and system requirements to architecture design recommendations: an expert system approach

Carlos Acosta
July 2017

M.Sc Thesis

The University of Amsterdam

Supervisors: Dr. Zhiming Zhao
Dr. Paul Martin

Table of contents

Abstract

When developing distributed systems like research infrastructures, requirement gathering and architecture design are often difficult and time consuming. The ENVRI Reference model abstracts generic patterns from environmental research infrastructures[1] and provides an ontological framework for facilitating the communication between infrastructure developers and domain scientists; however, deriving application patterns from specific design requirements are still challenging due to lack of user friendly tools.

In this thesis we tackle this challenge by proposing an expert system based approach to bridge the gap between requirements and system architecture design, studying the interaction usability of the prototyped expert system. We investigated several dialog methods and analysed the differences between them. Later on, we identified different patterns in the participants interactions. We also investigated how to profile the expertise levels and background of the participants based on their input, which contributes to autonomous customization of the interaction interface.

[1] "About | Research Infrastructures - Research & Innovation - European ...." 17 Jan. 2017, https://ec.europa.eu/research/infrastructures/index_en.cfm?pg=about. Accessed 9 Aug. 2017.

Acknowledgments

I would like to thank my supervisors Dr. Zhiming Zhao and Dr. Paul Martin for being helpful and guiding me throughout the process in completing this project. I would also like to thank the ENVRI community for joining the survey.

1 Motivation

A software architect must manage a constellation of issues, which range from requirements gathering, architecture design or data flow modelling to selecting proper technical candidates. When developing a complex distributed system like a big data infrastructure, they often must excel in interpersonal conversations and understanding of users from various communities. The process is time consuming and costly when requirements change during development.

During the past years, research infrastructures attracted lots of research interest from domains like environmental and earth sciences, humanity, biomedical and high energy physics. Different technologies have been prototyped, however, a proper solution has to meet several design constraints, such as dependencies among relevant components, metadata standards and API's.

A reference model abstracts generic patterns of certain types of system and it is often used as a guideline to design system architectures. A typical example is the ENVRI[2] Reference Model (RM), a reference model for building research infrastructures in environmental and earth sciences. The ENVRI project gathers many of the EU ESFRI[3] and other environmental research infrastructures to find solutions to common problems. The results, including the ENVRI Reference Model, will accelerate the construction of these infrastructures and improve the interoperability among them. The experiences gained will also benefit the building of other advanced research infrastructures. On the other hand, designing a customized architecture based on requirements currently require lots of input from human experts.

In order to bridge the gap amongst reference model, the requirements and the architecture design; it was crucial to develop an intuitive tool. During this thesis we will refer to such tool as portal.

Expert systems can help developers to efficiently check those constraints and choose a proper solution.

Expert systems have three different components within its core[4]. First of all it contains a knowledge base providing all the facts and rules about a specific topic.

---

[2] "ENVRI community – Studying the environment today to tackle the ...." http://envri.eu/. Accessed 5 Aug. 2017.
[3] "esfri.eu." http://www.esfri.eu/. Accessed 5 Aug. 2017.
[4] "Expert Systems/Components of Expert Systems - Wikibooks, open ...." https://en.wikibooks.org/wiki/Expert_Systems/Components_of_Expert_Systems. Accessed 10 Aug. 2017.

Secondly, it has an inference engine, which is an artificial intelligence protocol for navigating through the different rules and data inside the knowledge base. Finally, we have the Graphical User Interface (GUI) which links both the logic and data to the user. This provides a way for the user to input information and also a way for the user to visualize and understand the information retrieved from the knowledge base.

The aim of this project was to study state-of-the-art expert system technologies and usability in software engineering, and to investigate the behaviour of users based on their input data. To test it we developed a user friendly graphical interface (portal) that linked the user to a knowledge base that had ingested a machine readable form of the reference model in order to allow querying of the model's definitions. This would automate the process of discovering architectural solutions upon receiving functional requirements as an input. Furthermore, we also investigated the way in which the portal would communicate with the reference model, depending on the most popular type of requirement input by the participants. For this it was necessary to study and deepen my knowledge about requirements interpretation and classification.

Lastly, a study carried out internally by members of the ENVRI community about characterisation of the people that used the reference model, motivated the idea to research about how requirements could be used to classify users into a certain 'level of expertise'.

In this thesis, we focus on the following questions:

1. What interaction interfaces provided by expert systems are preferred within the ENVRI community? Specifically, we focus on natural language interface approach and question-based.
2. Within the natural language domain, what do users prefer, structured-text input method or free-text input method?
3. How to effectively discover architecture patterns for given requirements based on a reference model?
4. How to profile users based on their knowledge on the 'reference model'?

The thesis is organized as follows. First, we review the state of the art and discuss the research approach we proposed in the thesis. After that we present the ENVRI RM architecture recommender (portal), and discuss its system requirements, technology considerations and prototype walkthrough. Furthermore, we describe the usability study and experimental procedures, followed by the experimental results. Finally we analyze the results against theory and provide a conclusion.

2 Approach

In this section we introduce the ENVRI RM and cover the state of the art of key issues involved in expert systems, including its usability, knowledge bases and natural language interfaces. Furthermore, we describe the approach taken in this research and state how it differentiates from other researches performed in the past.

2.1 ENVRI RM introduction

A reference model abstracts generic patterns of certain types of system and it is often used as a guideline to design system architectures. A typical example is the ENVRI[5] Reference Model (RM), a reference model for building research infrastructures in environmental and earth sciences. The ENVRI project gathers many of the EU ESFRI[6] and other environmental research infrastructures to find solutions to common problems.

We distinguish three viewpoints inside the reference model that are used in the research to classify users depending on their requirements; the science, information and computational viewpoints [1][2]. The science viewpoint captures the requirements for an environmental research infrastructure from the perspective of the people who perform their tasks and achieve their goals as mediated by the infrastructure. The information viewpoint provides a common abstract model for the shared research data handled by the infrastructure. It focuses in the data, without considering any platform-specific or implementation details. Finally, the computational viewpoint accounts for the major computational objects that can be found within an environmental research infrastructure, as well as the interfaces by which they can be invoked, and by which they can invoke other objects in the infrastructure. In my research we focus on the computational viewpoint as the system shows recommendations for computational objects.

Within the computational viewpoint, the structure of research infrastructures is divided into sub-systems. It helps break down the complexity in analysis. Each sub-system follows a data life-cycle to identify functions and computations.

[5] "ENVRI community – Studying the environment today to tackle the ...." http://envri.eu/. Accessed 5 Aug. 2017.
[6] "esfri.eu." http://www.esfri.eu/. Accessed 5 Aug. 2017.

The life cycle[7] is; acquisition, curation, publishing, processing, use. There is already identified a core set of functions. These are defined as interfaces which encapsulate operations and services that act upon an object. An object is a real world entity which contains a behaviour and a state. The interactions between the objects are supported by the corresponding interfaces. For example, *Register Service* would be the behaviour performed by a *Service Provider* to make the service visible to *Service Consumers* by registering it in a service registry[8].

2.2 State of the art

Expert systems technology provides the functionality for building institutional or corporate memory of firms. They are being used to preserve or document knowledge so that once the individual retires, their knowledge would not be lost [3]. Applications of expert system knowledge in different fields of discipline has been done. One of these fields is *design and planning,* most relevant to my research*,* where the aim is to shorten the time taken to achieve a solution, acting as human experts. Two examples of this type of expert system are COMEX[9] [4] and CAKES-ISTS[10] [5]. Currently there are many studies and development of expert systems for various different fields like medicals, militaries, chemistry, engineering, manufacturing, management, etc. Expert systems aim to provide better alternative solutions and assist companies that struggle to thrive due to the competitive market challenges. In terms of the optimization, it hopes to prevent losses and wastes, production time and labor invested to manufacture a product [6].

Expert system usability

Usability makes it possible for interactive GUIs to be easy to learn, effective to use and enjoyable from the user's perspective. This results in the goals of effectiveness, efficiency, safety, utility, learnability and memorability [7]. Previous usability research performed by experts, provide the definition and quality components of usability [8].

---

[7] "Model Overview - ENVRI Collaboration and ... - EGI Confluence." 9 Dec. 2016, https://confluence.egi.eu/display/EC/Model+Overview. Accessed 5 Aug. 2017.

[8] "ENVRI Reference Model - EGI Confluence - EGI.eu." 9 Nov. 2016, https://confluence.egi.eu/display/EC/ENVRI+Reference+Model. Accessed 5 Aug. 2017.

[9] "COMEX: A Cost Management Expert System." http://icit.zuj.edu.jo/icit11/PaperList/Papers/Artificial%20Intelligence/518_daniela.pdf. Accessed 5 Aug. 2017.

[10] "A causal knowledge-based expert system for ... - ACM Digital Library." 1 Aug. 2012, http://dl.acm.org/citation.cfm?id=2181431. Accessed 5 Aug. 2017.

There are:

1) Dix et al highlights an expert system as a system that can help users solve their problems [9]. Such system should be :
• Useful: functions as desired by the user.
• Usable: is easy to operate
• Used: motivates the user to use, appealing to the user, fun, and etc.

2) Usability is a quality attribute that assesses how easy user interfaces are used. The word "usability" also refers to methods for improving ease-of-use during the design process. Usability is defined by Jakob Nielsen as five quality aspects [10]:
1. Learnability: The degree of ease with which users accomplish basic tasks the first time they encounter the interface.
2. Efficiency: How quickly can they perform tasks once the user have learned the design of the system?
3. Memorability: After a period of inactivity (the user not using the system for a significant period of time) how easily can they reestablish proficiency?
4. Errors: The amount of errors users make and the ease with which they can recover from them.
5. Satisfaction: How pleasant is it to use the system?

3) Palmer highlights the quality attributes of usability as: the download time, navigability, interactivity, responsiveness, content quality. [11]

Furthermore, Gaines and Shaw came up with several standard dialog guidelines specific to the design of effective human-computer dialogs [17]. These can be applied to expert systems. The most relevant guidelines to my research were:

1. *Programs create the reality experienced by the users computers*. The computer is a tool for simulation. This power should be used to create worlds that are simple for the user and natural to the task. Most expert systems currently operate through keyboard input and textual output. However, the combination of expert systems and simulation techniques to provide knowledgeable environments is extremely powerful and might be an increasing basis for significant applications.

2. *Users already have expectations about computers.* Take into account the possibility that the user's expectations of the computer will affect his interpretation of any dialog with it. The dialog should be designed to minimize confusion arising from these prior expectations.

3. *Use the vocabulary of expert and user.* Design the dialog using the normal vocabulary that an expert and a user would use. The vocabulary in expert systems is very important. The expert specifies his conceptual framework and inference rules, and it is assumed that the user can communicate and understand facts within that framework.

Knowledge base

Currently we can distinguish two types of knowledge bases. A *domain-specific* knowledge base that captures concepts, instances, and relationships of a domain of interest and a *global* knowledge base which attempts to cover the entire world. Examples of domain-specific knowledge bases include DBLP[11], Google Scholar[12], DBLife[13], echonest[14], and product KBs being built by e-commerce companies. Examples of global knowledge bases include Freebase[15], Google's knowledge graph[16], DBpedia[17], and the collection of Wikipedia infoboxes[18].

This distinction is important because depending on the target applications, we may end up building one type or the other. To power most of real world applications, it's enough to build a few large global knowledge bases.On the other hand, while global knowledge bases are still very important, there is also an increasing need to build domain-specific KBs, and in fact, we have seen this need in many domains. Consequently, it is important to develop efficient methodologies to help domain experts build such knowledge bases as fast, accurately, and inexpensively as possible [12][13].

Natural language interfaces

Not all expert systems provide natural language interfaces, but when they do, an important issue is addressed. Expert systems use a semantic mechanism that is powerful enough to translate user statements into facts. This semantic approach is based on verb categorization, which is often structured hierarchically and equipped with parsing algorithms. Some issues in the construction of the complete semantic

---

[11] "dblp: computer science bibliography." http://dblp.uni-trier.de/. Accessed 5 Aug. 2017.
[12] "Google Scholar." http://scholar.google.co.uk/. Accessed 5 Aug. 2017.
[13] "DB Life." http://www.dblife.today/. Accessed 5 Aug. 2017.
[14] "The Echo Nest." http://the.echonest.com/. Accessed 5 Aug. 2017.
[15] "Freebase - Google Developers." https://developers.google.com/freebase/. Accessed 5 Aug. 2017.
[16] "Knowledge Graph - Google." https://www.google.com/intl/es419/insidesearch/features/search/knowledge.html. Accessed 5 Aug. 2017.
[17] "DBpedia." http://wiki.dbpedia.org/. Accessed 5 Aug. 2017.
[18] "Wikipedia:List of infoboxes - Wikipedia." https://en.wikipedia.org/wiki/Wikipedia:List_of_infoboxes. Accessed 5 Aug. 2017.

module are still being investigated, for example; partial matching, i.e. what to do with inputs that only match part of a fact, instantiation of variables in the facts, as well as derivation of goals from user queries. Natural language modules are used to add facts to the data base of the underlying expert system in an unconstrained manner, thus placing extra requirements on the underlying expert system. The inference engine uses a combination of forward chaining and backward chaining so as to efficiently use facts entered by the natural language interface. It makes available descriptions of its rule base and allows for a limited form of user control over its backward chaining mechanism. This facility allows the user to ask questions about the information contained in the rules but not normally supplied by expert systems. These attributes of inference engines allow a knowledgeable user to arrive at a solution to his query in the most efficient and least time consuming way, while maintaining a focused dialogue. This approach is also useful in domains where the decisions have to be made quickly, or where user queries are expensive, such as expert systems designed for use by busy professionals, such as accountants and doctors, as well as systems that work in hazardous environments, such as nuclear reactors [14].

User profiling within ENVRI Community

Previous research identifying different expertise levels within the ENVRI community had been performed [28]. During the research, users were interviewed and categorized into three different classes: RI Engineer/Scientist, CS Engineer/Scientist and Manager.

Summary

1. Expert systems usability has been investigated in the past in a broad scope. They were based on attributes such as learnability, memorability or satisfaction that encapsulated the expert system as a whole (input & output); in contrast to the focus that we wanted to give to the input methods detailed in this thesis.

2. Natural language input methods state of the art take into account issues related to functionality rather than usability. The distinction between structured-text and free-text are not covered for usability.

3. Knowledge bases play a key role in expert systems, but the communication with the front-end varies. In this thesis we wanted to investigate the communication between these two entities depending on the requirements input by the user. Due to the lack of such a domain-specific knowledge base,

we can not arrive at any conclusion through the state of the art for these particular question.

4. Profiling members of the ENVRI community has been performed in the past, although this entitled executing face-to-face interviews with each member. In this thesis we wanted to investigate the profiling of users through the requirements they input.

2.3 Proposed approach

To answer the research questions, we will first build the test system (step 1). Secondly, we will identify methods to query the knowledge base and finally we will design the experiments.

Step 1: Expert System Design
Firstly, the approach planned to design the test system was to make use of a decision tree or an inference engine.

Decision trees are widely used up until the present day and are one of the main ways to create the logic base of an expert system. Starting from the root as the first question, it traverses the tree by asking questions to the user. From each of these nodes there are n possibilities (with n>1) of answers with always only one possibility achievable in one time. This means that there is no backtracking in a decision tree and will only stop once it gets to an outer node.

In contrast, an inference engine such as the *pyke*[19] implementation for python contains rules that are triggered by user input. These rules can be backtracked by a simple process. If it succeeds at providing an assumption, the flow proceeds down to the next assumption in the list. Trying to continue down the last assumption in the list causes the rule to succeed. If on the other hand it fails at providing an assumption, the flow backtracks up to the prior assumption in the list and tries to find another solution for it. This process repeats until it succeeds [15]. In figure 1 we can see a backtracking example flow.

[19] "Welcome to Pyke." http://pyke.sourceforge.net/. Accessed 5 Aug. 2017.

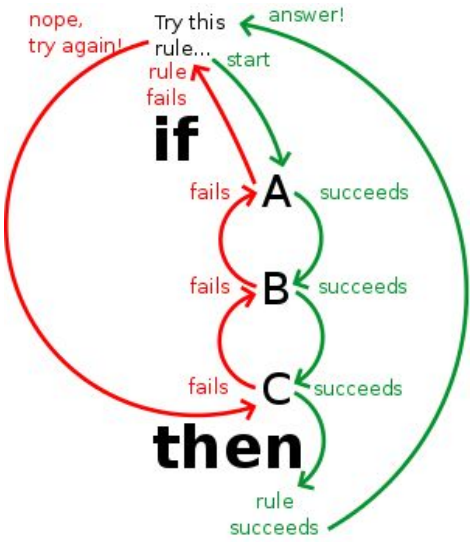


Figure 1: Backtracking example flow.

Both these approaches were dropped for another approach later in development due to several drawbacks. The decision tree would have had difficulties with scalability [16], both with addition of new nodes and with traversal, since the biggest the tree, the longer it would have taken to get an output. The inference engine with the *pyke* implementation was dropped when we realised it would have future integration problems with the reference model.

The final approach used a remote knowledge base. This was better than its predecessors for several reasons. It was more efficient and less resource consuming since it could be queried for any entity directly without going through all the other nodes from a root node. The knowledge base could be updated with new information and facts at any time without touching any of the already deployed data. Lastly, it allowed for an easier integration with the portal.

Step 2: Identifying methods for querying the knowledge base

Components and design patterns are useful ways to capture certain requirements. The role of models such as the ENVRI Reference Model are to try and describe useful design patterns that research infrastructures can implement to support certain behaviours.

Initially we thought about using quality attributes as a means to query the knowledge base. Due to this we conducted further research about architectural components and design patterns classification into different quality attributes. This research can be found in appendix B.

Another way of querying the knowledge base was identified in a later date. This consisted on using behaviours in order to classify requirements. A behaviour is a process engaged by one or more actors acting in certain roles. In the ENVRI Reference Model we can find five different sub-systems; data acquisition, data curation, data publishing, data processing and data use. Each of these subsystems have different behaviours associated with them. For example in data acquisition we find the behaviour of *data collection*. This involves components whose main purpose is to obtain digital values from a sensor instrument, and associate consistent timestamps and necessary metadata.

There are many behaviours that are identified and listed in the ENVRI Reference Model documentation[20].

Step 3: Experiments design
In order to perform the usability tests on the system, a tool that recorded the user interactions with the system was used. This made it possible for members of the ENVRI community located in other countries to participate concurrently in the experiment, since it was possible to review all the recordings without me being physically present or interviewing them. At the same time, to classify users into different expertise levels and find out the best way to communicate with the reference model, a questionnaire was designed through 123contactform[21]. Members of the ENVRI community were kindly asked to participate in the survey.

2.4 Novelty

Expert systems have been tested for usability since their earlier days. However, using expert systems to facilitate reference model based architecture pattern discovery and guided design is not yet broadly studied. In this thesis we focus on the actual communication method between user and machine, taking into account user feedback.

Furthermore, although the classification of members of the ENVRI reference model into the three expertise levels was already investigated by performing interviews, we decided to take another approach and use requirements to classify them.

[20] "SV Community Behaviours - ENVRI Collaboration ... - EGI Confluence." 9 Nov. 2016, https://confluence.egi.eu/display/EC/SV+Community+Behaviours. Accessed 5 Aug. 2017.
[21] "123ContactForm." https://www.123contactform.com/. Accessed 10 Aug. 2017.

3. ENVRI RM architecture recommender

In this section we discuss the key technologies and theories in prototyping the architecture recommendation systems. We start by showing the system requirements. Secondly, we talk about the system architecture and technical considerations. Here we focus on the technology used to create the system and the reasons behind the choices. In addition to this, we also show how each component of the system communicates with each other through several diagrams.
The section finishes with an explanation of the portal prototype.

3.1 Requirements

During the development of my project, the requirements did not suffer considerable changes worth mentioning here. These were, for the most part, clear since the beginning of development. This was due to the fact that the problem that the system had to solve was solid and well established from the start.

Functional

1. The system must contain several methods for the user to input requirements: such as through natural language and through questions.
2. The system must be able to read user requirements as an input and interpret them.
3. The user should be able to input as many requirements as it wants through natural language.
4. The output (recommended system) should contain a visualisation and this must be manageable.

Non-functional

1. The system should have an easy to use design, minimising unnecessary components.

3.2 Architecture and technical considerations

The portal is developed with the idea of being easily deployable and accessible in every environment possible. Due to this idea a web based approach was chosen. For the website hierarchy see diagram 2. Javascript is used to develop the interactive part of the portal. The react[22] library is used to implement the text input functionality and filtering of keywords to ease with reusability and simplicity of code. In order to query the remote knowledge base, which is a *Jena Fuseki RDF triple store*[23], *SPARQL*[24] queries embedded in HTTP 'get' requests are used. The use of a remote knowledge base was due to the fact that it could be accessible online anywhere and that it had the ability to efficiently use existing knowledge resources (since the knowledge base already encoded lots of information about the reference model, so it was not necessary the replication of the information locally).
In appendix A you can find activity diagrams of the different input methods of the portal. Diagram 1 shows a sequence diagram with all the interactions between the user, GUI and knowledge base.

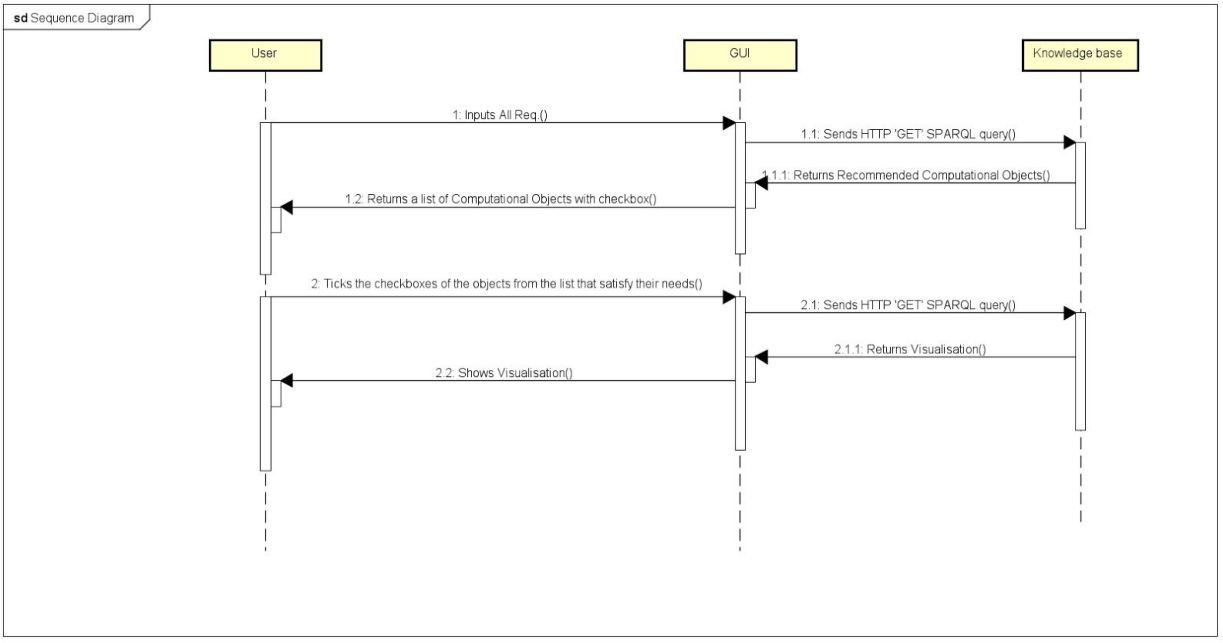


Diagram 1: Sequence diagram showing software component interactions for natural language input method.

[22] "React - A JavaScript library for building user ...." https://facebook.github.io/react/. Accessed 5 Aug. 2017.
[23] "Apache Jena - Fuseki: serving RDF data over HTTP." https://jena.apache.org/documentation/serving_data/. Accessed 5 Aug. 2017.
[24] "SPARQL Query Language for RDF - World ...." 15 Jan. 2008, https://www.w3.org/TR/rdf-sparql-query/. Accessed 5 Aug. 2017.

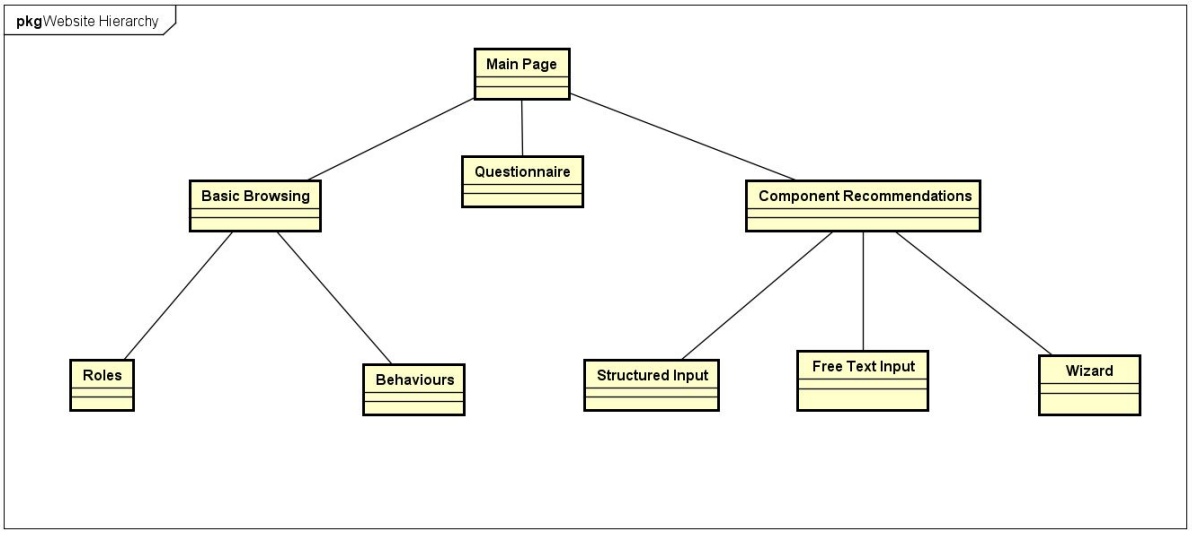


Diagram 2: Website hierarchy

3.3 Prototype

The portal is separated into two sections (shown in figure 2): The *basic browsing* section and the *component recommendations* section. In basic browsing you can examine closely the reference model's roles and behaviours per research community. In component recommendations, a generated architecture design is modelled through user requirements. The user may choose between three different input methods: Structured-text, free-text and wizard. Both the structured-text and the free-text are natural language input methods. The wizard input being in the "question-based" category.

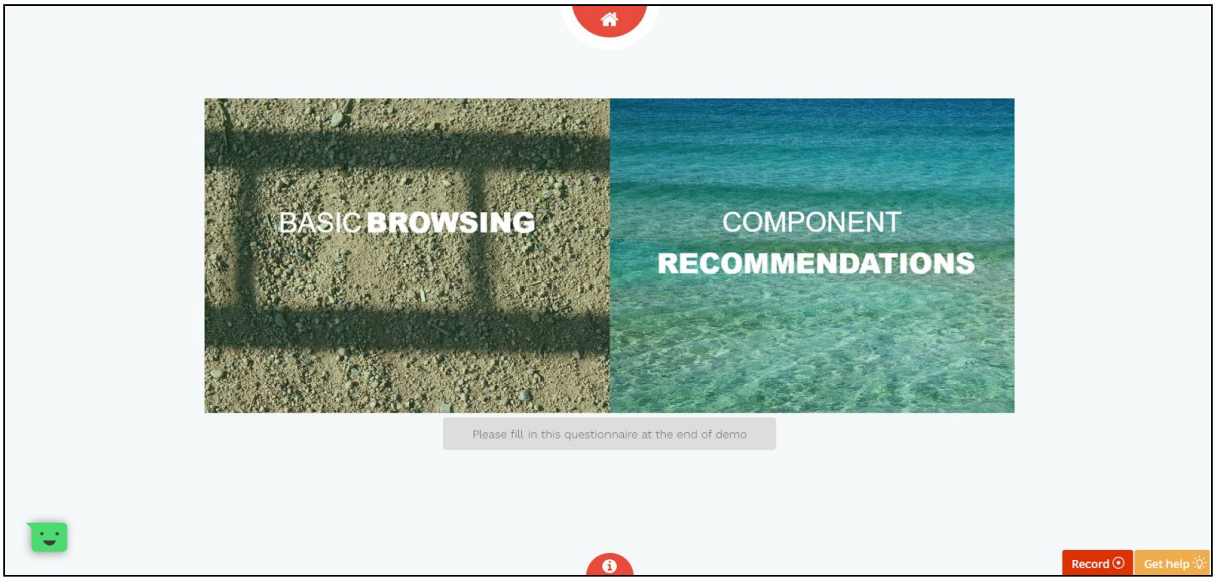


Figure 2: Portal main menu.

In the structured-text input option (shown in figure 3), the user must tell the system the number of requirements they want to input and follow a predefined requirement structure, more precisely, they must input their requirements using the following structure: “The software shall…”. This was not a random choice. It was necessary to find a schema that would be understandable by everyone and suitable for representing generic functionality, so a schema defined by the *Intel Corporation*[25] to define ubiquitous requirements was used.

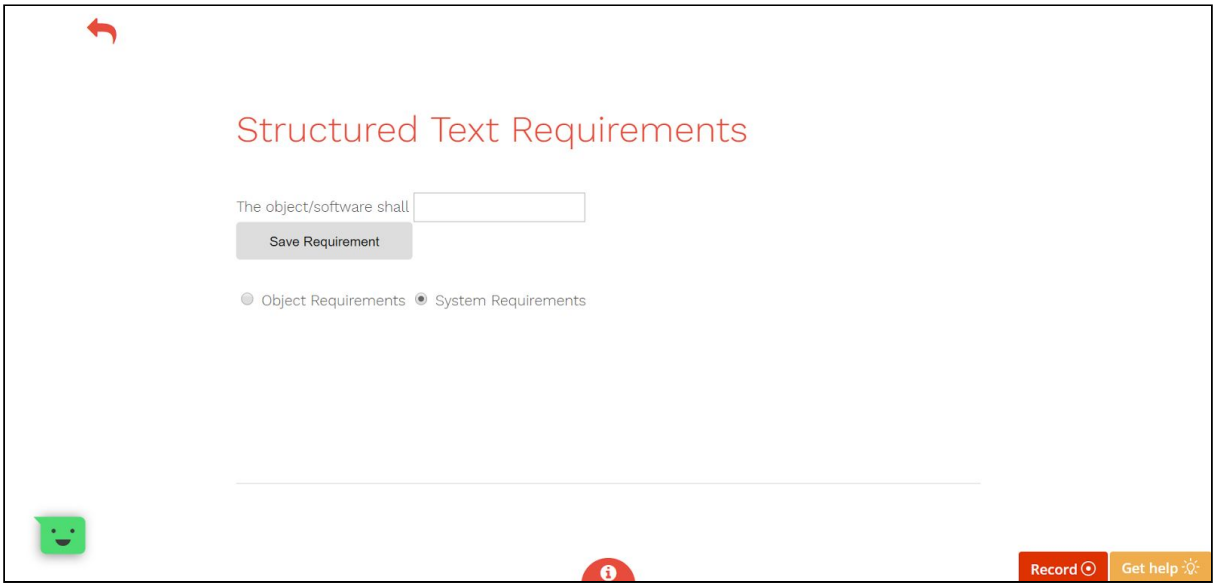


Figure 3: Structured input option. 125% zoom.

In free-text the user is given full control on the structure of the requirement he wants to input. They also don’t need to specify the number of requirements they want, they just write any number of them and sends them to the system when finished. In both the structured and the free-text input methods, once the requirements are sent to the system, keywords are extracted and compared against the computational objects descriptions, returning a table of recommended components (shown in figure 4). Each component can be checked for dependencies with other components and marked up for analysis. Once all the marked components are sent for analysis, the system returns a visualisation of them, along with their dependencies (shown in figure 5).

[25] *"EARS: The Easy Approach to Requirements Syntax ... - iaria." 21 Jul. 2013, https://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf. Accessed 5 Aug. 2017.*

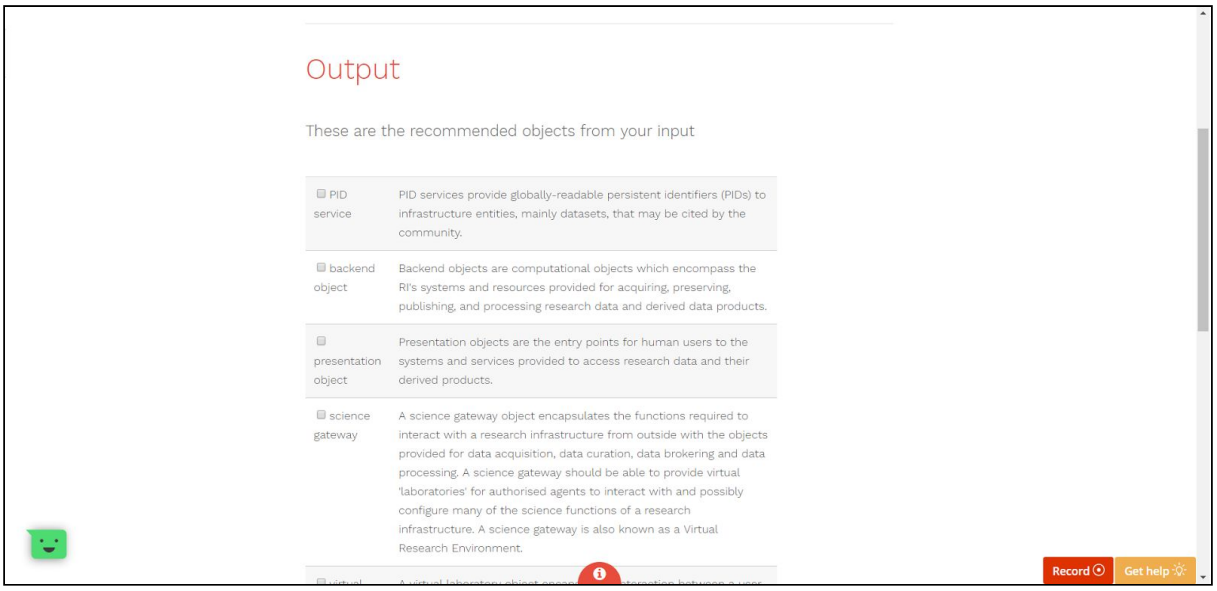


Figure 4: Table of recommended components.

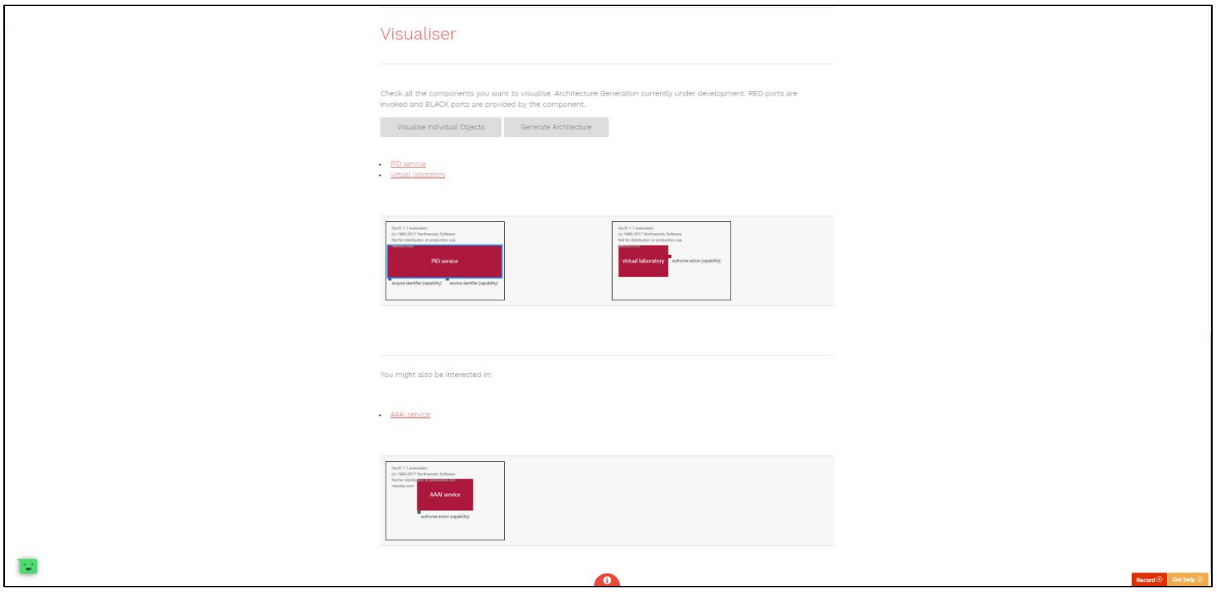


Figure 5: Component visualisation. 55% zoom.

The wizard option (shown in figure 6) works by asking the user for “Yes-No” type of questions and depending on the outcome of the answers it would return different component visualisations.

Figure 6: Wizard input option.

4 Usability study

To answer our research questions enumerated in section 1, we set up a number of experiments to study the interaction between user and software, and to analyse their natural behaviour when using different input methods. Furthermore, we describe the procedure for investigating the communication methods between the reference model knowledge base and the portal, tagged under the following subsection: *architecture pattern query*. This is followed by a subsection describing the profiling of users procedure.

The validity and reliability of the experiments and the way the data is collected are defined, along with the data that had to be balanced to make it a fair test. Also possible points of error are identified.

4.1 Requirement description interface study

We first investigate different interfaces for describing architecture requirements: natural language or question based. Then we specifically focus on two modes in the natural language approach: structured text or free text.

In both experiments we had to balance several aspects of the procedure to increase its validity:

- All participants have to describe the same amount of systems.
- The system to be described is the same for all candidates.
- The provenance of the participants are from CS and/or engineering backgrounds.

4.1.1 - Natural language or question-based?

We start by comparing two basic description approaches for system requirements in the expert system: classical question based approach and natural language based input. The basic assumption is that users will prefer the flexibility of the natural language approach in contrast to the question based approach. On the other hand, this hypothesis could be wrong since the users might value more the simplicity of the question based approach.

Procedure

Each participant starts by going to the main page of the demo and going through a guided tour of the portal, showing how everything works (shown in figure 7). The

purpose of this is to give the participant an insight to the system and all its functionality.

Secondly, we perform the experiment by asking the user to fulfil a specific task using the input method of their choice, in this case a *Data Collection* system. The text displayed to the user can be read in figure 8. We ask them to use at least two methods (ie. wizard and structured-text or free-text and wizard). This way we make sure they use both a natural language and a question-based method.
Data is collected through two ways; demo screen recordings and a survey. By using *hotjar*, a well known tool used often by user experience specialists, we can record and store the actions from any participant that uses our website. This way we can check which input method they use first, if natural language or question-based. The first choice that the user will make is commonly the one that he thinks will perform better for a certain task, but this does not mean it will be the correct one. This is why we ask the user in a survey to specify which input method he preferred from the two chosen ones and to explain why.

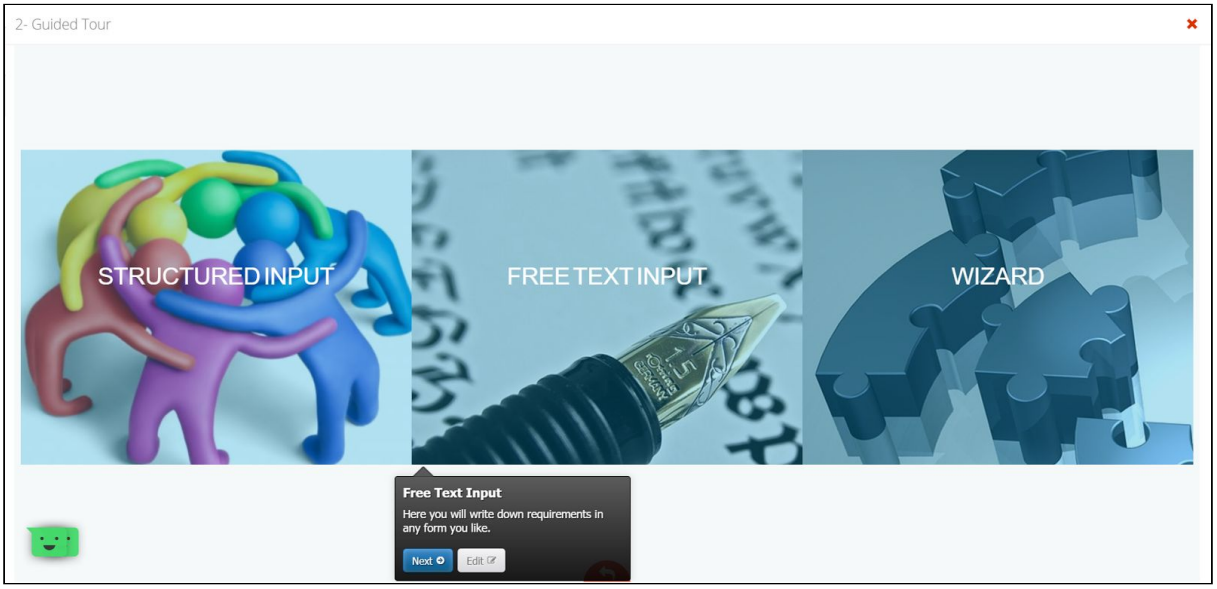


Figure 7: One of the steps in the guided tour describing the *free-text* input method.

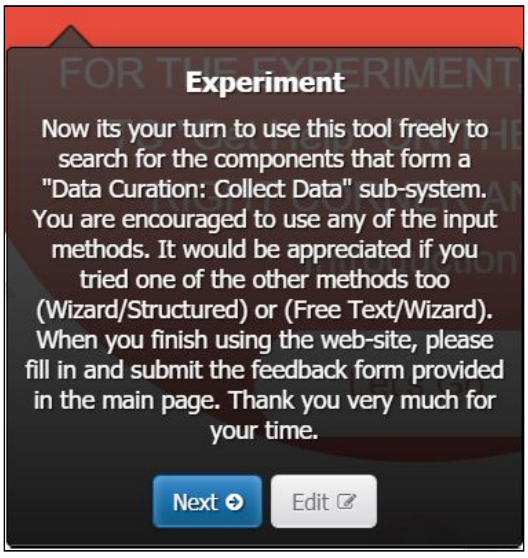


Figure 8: The experiment description displayed in the portal.

Since there are two natural language input methods and only one question-based method, it was necessary to treat the two natural language input methods as one. Furthermore, a possible source of error could be the difference in the output visualisation between the natural language approaches and the question-based due to technical limitations; making it possible to influence the participants choice of input.

4.1.2 - Free-text or structured-text?

This experiment shares many similarities with the previous one, hence why it has been treated as a sub-experiment. The difference is that the experiment is performed within the natural language input methods to test whether participants prefer free-text or structured-text.

Data is collected in the same way as the previous experiment, but only registering the preferred input method. The expected result is that users will prefer the free-text over the structured-text since it provides a higher degree of freedom when expressing requirements. On the other hand, structured provides much more help since it guides users that do not have so much experience with requirements.

4.2 - Architecture pattern query

The hypothesis we wanted to test was: When describing requirements, developers address system functionality/behaviour more often than quality attributes. In other words, they use more functional requirements than non-functional requirements to generate a system. Currently the reference model computational objects do not support quality attributes so the only way of querying it was through behaviours, but would it be better to query it depending on the qualities of the system? For this to be feasible the user would need to input requirements that directly influenced any of the quality attributes identified so far. To test the users nature with requirement input it was necessary to check the frequency with which users addressed quality attributes in their requirements when using the system, and if this was high enough, introduce the possibility for a quality attribute querying implementation within the reference model. It was not specified to the user whether to input functional or non-functional requirements since this would have influenced negatively in the results.

Data was collected through the requirements that participants input into the system. Subsequently, they were analysed to distinguish between the ones that addressed behaviours and the ones that addressed quality attributes. To make the test fair, all participants were asked to design (through requirements) the same systems.

The participants in this experiment were from different backgrounds since, despite being able to implement this technology on the ENVRI Reference Model, was not specific to it. We wanted to capture the general atmosphere of architectural design and requirement input of the participants.

4.3 Profiling based on user input

This experiment aims to find out if it is possible to classify RM users into specific classes depending on the requirements input. The classes proposed by an external study were RI Engineer/Scientist, CS Engineer/Scientist and manager. This would be possible by analyzing the requirements from people from one class in search of which viewpoint was referenced the most.

Data was collected through a form with two sections. The first one checks which viewpoint a user is more familiar with. The second one checks which viewpoint is more relevant to the requirements input by the user. There are in total four systems that the participants had to design through requirements. It must be noted that the

nature of this systems did not influence the experiment negatively since each system could be addressed by using any of the three viewpoints.

The form has been sent to several users from the ENVRI community. In order to make it a fair test and increase validity, expertise of individual users with the RM has been taken into account. To measure it, participants are asked to indicate their level of expertise through a scale from one to five (five being proficient). Since we wanted to test precisely the ENVRI reference model, only participants with an understanding of three to five were included in the research. Furthermore, a certain class would only be taken into account if there were five or more participants. Consequently, since there is a scarce amount of manager participants , they are not considered. The amount of systems and type are all the same for each candidate.

A possible degree of error in this experiment is the fact that some requirements referenced two viewpoints. The approach here was to address both viewpoints since we could not know which viewpoint the participant referred to and it would had been bad practice to ignore them. Another point of error could be the anchoring effect influence exerted by the first section of the form into the second section. Furthermore, it must also be noted that qualitative analysis is always prone to interpretation differences and can lead to erroneous results.

Procedure

First it was necessary to know which viewpoint was the one each class had the most expertise with. This was possible by bringing together five objects from each viewpoint into one question and asking the participant to identify which objects they were more familiar with. This was the first question of the survey and labeled as *term expertise* test. To test the validity of this question, we asked a second question to the participants consisting of describing a specific system by using only the objects from this pool of objects. This question was labeled as *scenario expertise* test. Depending on which objects were identified by the users within a class, we would find out which one was the most referenced viewpoint per class.

Secondly, in order to test the hypothesis, the user is asked to describe four systems using free text requirements. These are analysed qualitatively per class to check for the most referenced viewpoint. This part of the experiment is labeled as the *requirement referencing* test.

If both parts of the experiment did match on the viewpoint used and the difference between the first and second most referenced viewpoints were significant, the

hypothesis would be true. It would be possible to group users into different classes by reading their requirements with a high level of affinity.

4.4 Summary

1. The *requirement description interface studies* share many features of the procedure. Users follow a walkthrough when participating in the experiment and data is collected by recording their actions. To clarify some aspects of the experiment, the user also submits a survey

.

2. The *architecture pattern query* experiment studies the way in which users reference the RM knowledge base in their requirements. Data is collected by analysing the different requirements input by the users into the system.

3. The *profiling based on user input* experiment aims to find out if it is possible to classify RM users into specific classes depending on the requirements input. Data was collected through a form.

5 Results

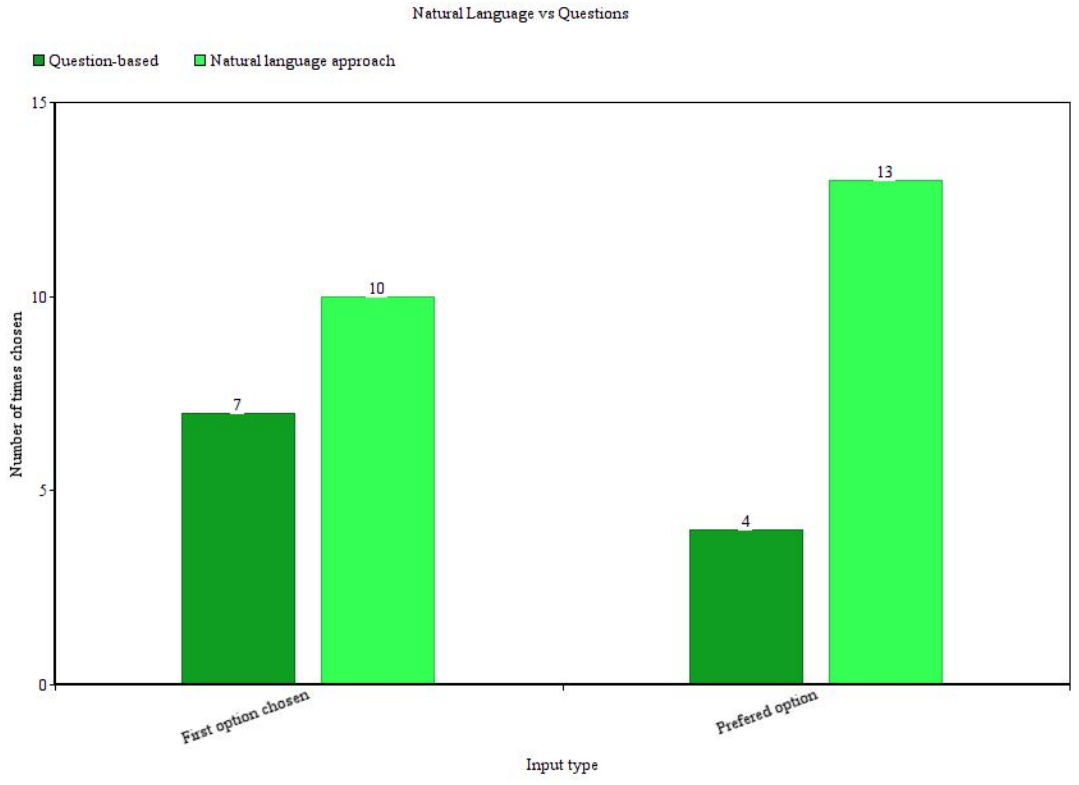
In this section we analyse the results, pointing out the quantifiable data and the interesting trends and patterns within each experiment. These results are later discussed in section 6 of the report, and finalised in several conclusions.

5.1 Requirement description interface study

First we show the results for the investigation between the natural language and the question based approach. After this we show the results for the investigation between the natural based interfaces: Free text and structured text.

5.1.1 Natural language or question-based?

The results for this experiment are plotted in graph 1. There are in total 17 people participating in the experiment. For each of them, the first input method they used was registered. After using the system they were also asked which one they preferred. Participants admitted that they started using the question-based approach since it seemed to be the easiest option to achieve the output, seeking for speed and efficiency.

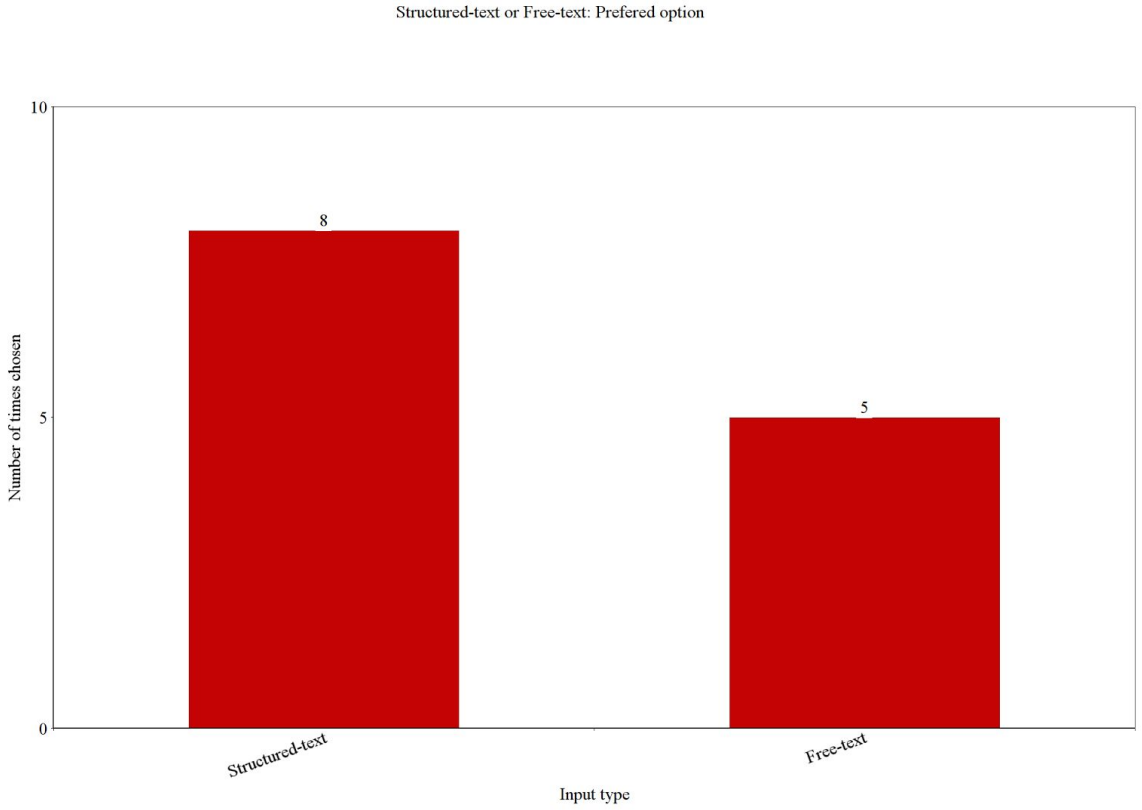


Graph 1: On the left is the data for the first option chosen by the user. On the right is the data for the preferred input method.

After trying the other options, they stated that it lacked the flexibility and vastness that the natural language approach had; hence why the 76% of participants chose a natural language approach as their favourite. All but two participants that started using the question-based approach switched to natural language, whilst one still thought it was the best option. Only two participants switched from natural language to question-based.

5.1.2 - Free-text or structured-text?

From the results we do not see a significant difference between options due to the number of total participants, although we can still identify that more users preferred the structured input in contrast to the free-text. The most common explanation to why they chose structured over free-text was that their knowledge about system requirements was not advanced enough and that it helped having a template in which to write the requirements. Another less common explanation, although still interesting, was that through structured-text the user could perform actions faster.



Graph 2: Graph representing the preferred natural language input option. On the left the structured-text and on the right the free-text.

5.2 Architecture pattern query

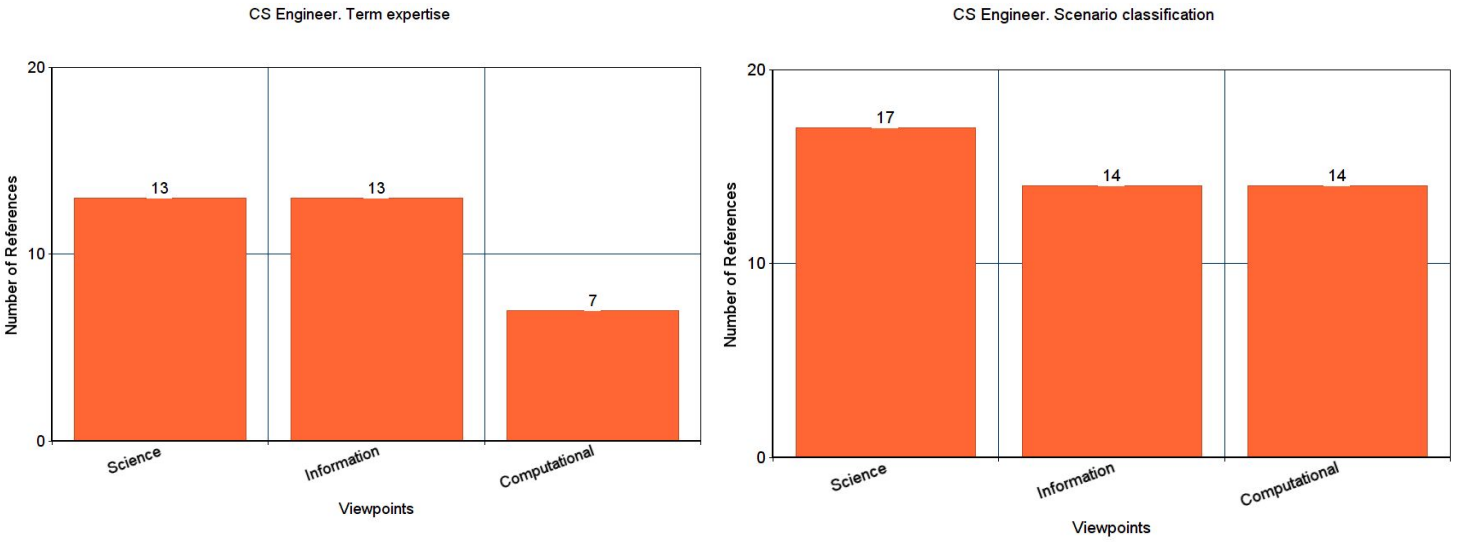
From 167 requirements, only 11 addressed quality attributes in the system. This means that 94% of requirements addressed behaviours in contrast to quality attributes. There was no correlation between the expertise of the participants with the reference model and the use of quality attributes since both, proficient users and average users provided these requirements. The accessibility of data was particularly mentioned, probably emphasizing the need for data sharing amongst scientists.

Requirement	**Quality attribute**
Accessible data repository	Accessibility
Interoperability	Interoperability
Improve data accuracy	Integrity
Accessibility of data	Accessibility
Ensure data protection	Safety
Verification of reusability of data (reuse Already processed data)	Reusability
Ensure that data are accessible from the platform	Accessibility
Ensure that data are accessible from outside the platform	Accessibility
Optimise data transfer for staging	Efficiency
To have a long-term preservation policy for sustainability.	Sustainability
Optimise data transfer for results	Efficiency

Table 2: On the left, requirements that satisfy quality attributes. On the right, the quality attribute that it satisfies.

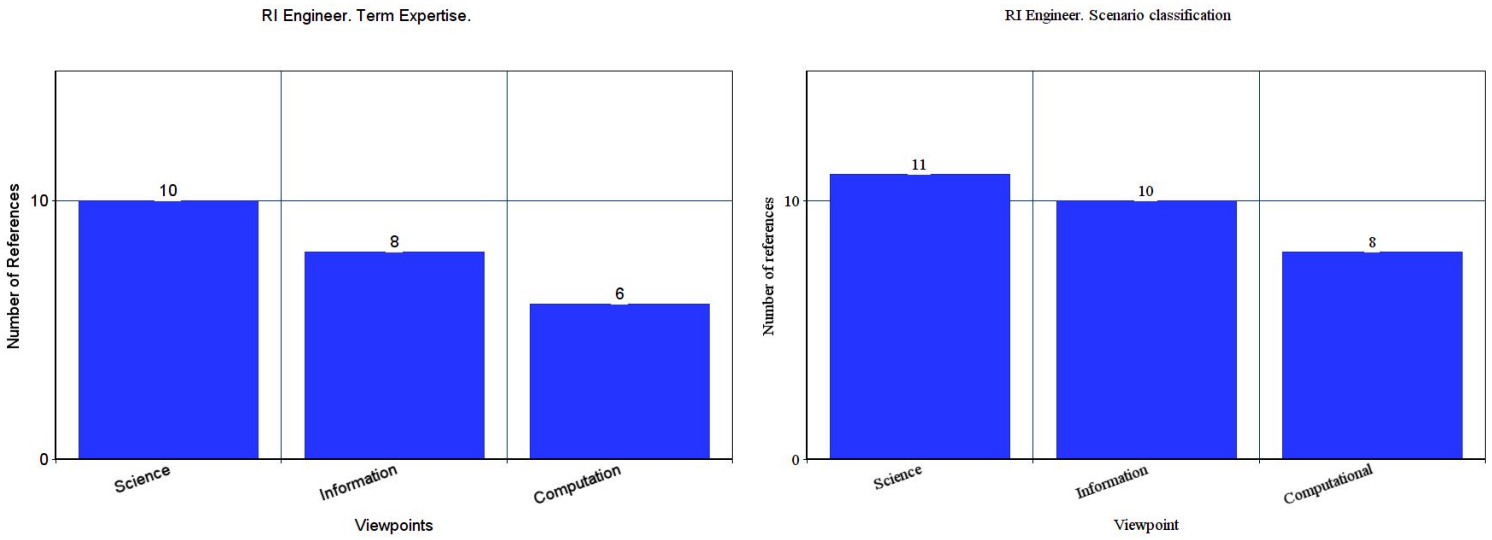
5.3 Profiling based on user input

Graph 3.1 illustrates the results from the term and scenario expertise questions for CS Engineers. The viewpoints are plotted against the number of references to each of them in the experiment. It can be seen from the *term expertise* question that there was an equal level of proficiency with the science and the information viewpoint components, followed by a decreased number of references to the computational viewpoint. Furthermore, the *scenario expertise* question tilted the balance in favor of the science viewpoint. It also must be pointed out how the participants referenced more components than the ones they initially pointed out as being proficient with.



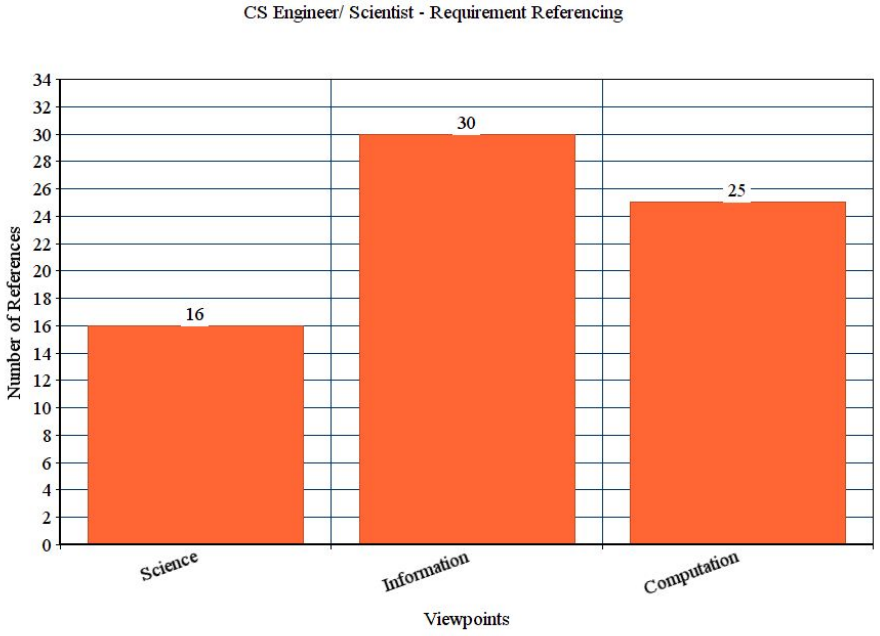
Graph 3.1: Data from the CS Engineer class. On the left from the *term expertise* question. On the right from the *scenario expertise* question.

The RI engineers came out to be more familiarised with the science viewpoint, followed by the information viewpoint and the computation viewpoint last. As with the CS Engineers, they also referenced a higher number of components when asked to describe a system. This can be visualised in graph 3.2.



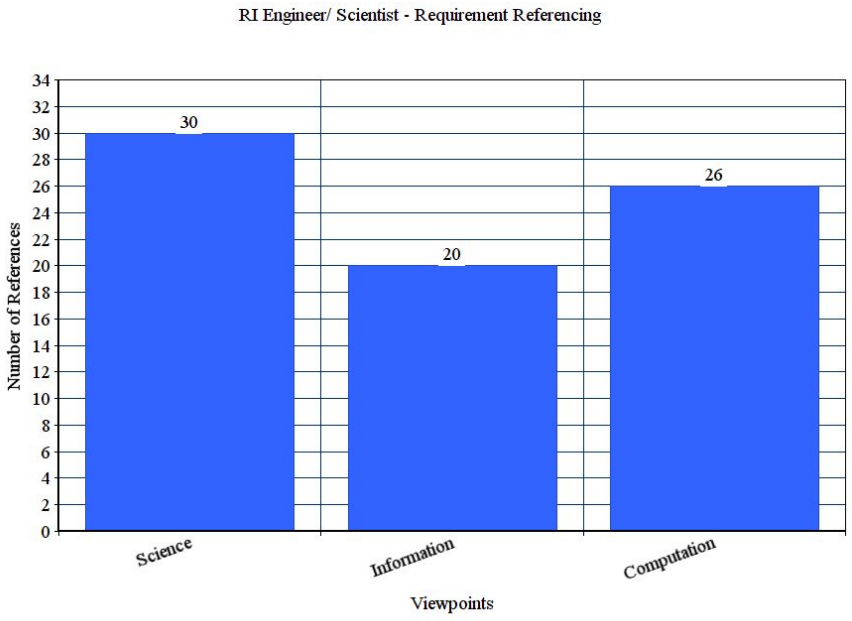
Graph 3.2: Data from the RI Engineer class. On the left from the *term expertise* question. On the right from the *scenario expertise* question.

Furthermore we state the results for the requirements input by the participants in the *requirement referencing* test. Graph 3.3 illustrates the number of references per viewpoint from the CS Engineers. The information viewpoint stands out as being the most referenced one by the requirements, followed by the computation viewpoint and the science viewpoint.
There were a total of 83 requirements for CS Engineers, with 71 referencing the reference model and 12 not. It must be noted that some requirements referenced two viewpoints simultaneously, such as ‘data storage’ since it is addressed in both the science viewpoint and the computational viewpoint.



Graph 3.3: Data from the CS Engineers requirement referencing.

It stands out from graph 3.4 that the RI engineers use components from the science viewpoint to describe the different systems. References to the computational viewpoint is also high and nearly equal to the science viewpoint. The information viewpoint concepts were significantly less referred to than their peers. There were a total of 84 requirements, with 76 referencing the reference model and 8 not. Appendix D can be checked for a full list of the requirements.



Graph 3.4: Data from the RI Engineers requirement referencing.

The fact that the CS engineers seem to be more confident with the computational viewpoint but reference more the informational viewpoint with their requirements, addresses the following point: the expertise of a user with a specific viewpoint is not directly proportional to the number of references to this viewpoint in the requirements.

5.4 Summary

From the results, we can observe:

1. Many participants started their experiments by using the question-based approach, since this approach seemed to be the easiest option to achieve certain output. However, after trying the natural language approach, most participants felt that the question based approach lacks flexibility and vastness compared to the natural language approach. Furthermore, we notice that more participants preferred the structured text input method rather than the free text input when using natural language input. One explanation given by participants is that a template based structure does help them to write the requirements.

2. The results for the *architecture pattern query* experiment showed that 94% of people use system behaviours in contrast to quality attributes as requirements.

3. The results for the *term & scenario expertise* questions from the *profiling based on user input* experiment showed that both CS and RI engineer/scientists were used to concepts in the science viewpoint of the ENVRI RM. However, in the *requirement referencing* test, the CS engineers/scientist referred to more concepts in the information viewpoint while the RI engineers/scientists referred to more concepts in the science viewpoint. This showed that the expertise of a user with a specific viewpoint is not directly proportional to the number of references to this viewpoint in the requirements.

6 Discussion

In this section we discuss the different trends and patterns identified from the experimental outcomes. We will discuss the experimental limitations of the system and provide possible further solutions . The common points of error identified in section 4 are revisited in this section, stating how it could have influenced the results.

Question vs natural language input method

One of the main properties an expert system should have is the ability to simulate a real human expert in every part of the usability cycle. This means that, in the domain of our research, the input methods should be *close to identical* to the ones used in a meeting face-to-face with an expert. We depart from the assumption that natural language might be more suitable for a user, since practically many requirements are initially written on paper as free text. Based on this assumption, we then compare it with a question based input.

Take into account the probability that the users expectations of computers would affect their interpretation of any dialog with it [17]. When using an expert system, one often expects the system to behave like a simulation of a human expert, capable of listening alike. This would explain why still a reduced amount of participants chose the question-based approach, meaning that the user subconsciously felt more comfortable, believing they were talking to a real expert. A limitation of the portal that could have influenced the result is the lack of effective information visualisations. With a greater number of output visualisations, the participant would have had a more immersive experience. Another limitation of the current experiment was the type of questions used. Due to the fact that only yes/no answers could be given to the portal, the precision and complexity of the questions were constrained. In reality, a software architect would ask more precise questions such as the ones found in appendix C.

From the experiments, we can see most of the participants chose the natural language approaches, which is compliant to the following principle of expert systems: Dialog vocabularies should be designed according to expert and user vocabularies [17]. Natural language allows the user to input the requirements by using any type of vocabulary they want, something that was lacking in the question-based approach.

It should be noted that, although the scope of the experiment depends on the choice of input method by the user, the output visualisation inevitably influenced the results. Some participants expressed different opinions referring to how good or bad the visualisation was. The output visualisation of the question-based input method

showed the architecture design of a system, with all the components linked to each other. On the other hand, the output visualisation of the natural language input methods showed the individual components without links. If more time could be spent on developing the portal, a unified output visualisation for both input methods could be prototyped to increase the validity of the results.

To sum up, a question-based approach would be an appetizing option at first sight, but after users familiarize with all its capabilities, they become aware of the limitations it imposes on the ability to express requirements. According to one of Gaines and Shaw's guidelines <‘*Programs create the reality experienced by the users computers’*>, combining different knowledge bases and simulation techniques is extremely powerful to provide a knowledgeable environment to the user. Due to this, it is important to combine these two approaches, having the question functionality that makes users feel identified with a human expert and the natural language approach as a way for the user to express their answer.

Structured vs free input methods

The portal was designed with the idea to implement a free-text input method for the user to express their requirements. After some early recordings of people using the system, it came to our minds the possibility to introduce an alternate input method. Users often have difficulties in interacting with the free-text input, when they do not know what requirements to write or how to write them.

The results we collected are not exactly compliant to the hypothesis mentioned in section 4.1.2. We hypothesised that users would prefer free-text over structured-text due to the high degree of freedom it provided when expressing requirements. In reality, more users preferred structured-text input because it provided them with a template, helping new users that were not experienced in requirements engineering; although it doesn't stop there. In addition to experience issues, users also reported other less common properties; speed increase, reduction of errors and consistency of inputs. This behaviour can be further explained by referencing three benefits of template usage [18]. A template can speed up the requirement input since it decreases the time spent by the user writing unnecessary entry lines such as: “I want the system to...”, “I would like...”, etc. Instead, the user spends time in the actual requirement data that will be analysed by the expert system. It reduces errors by implying to the user the type of requirement it wants to read. In the case of the

expert system, it's the schema proposed by the *intel corporation*[26]. Consistency of inputs is achieved since it does not leave every user to individually decide on the display of the requirement, instead, all requirements are the same and makes it easier for the expert system to locate important aspects.

We can assert that structure input was favoured over free-text, but can't solidly assume that it is a better input method due to the small difference between the number of references. On the other hand, we can affirm that structured input is more relevant for novice users that are not experienced with requirement input and as a way to increase speed, consistency and reduce errors of data input into the system.

Architecture pattern query

When participants are not told what type of requirement to input, their past experiences could have influenced on describing the systems after object behaviours instead of their qualities [19]. In addition to past experiences, there are several cognitive biases that could also have influenced the decision making ability of the participants. These include, but are not limited to: belief bias, the over dependence on prior knowledge in arriving at decisions; omission bias, generally, people have a propensity to omit information perceived as risky; and confirmation bias, in which people observe what they expect in observations [19].

During the experiments, we did not find many quality attributes enhanced requirement descriptions from the participants. But what if there was a method to extract quality attributes and behaviour from a single requirement, allowing the system to query the reference model with both pieces of data? Something we did not take into account was the possibility that participants implied quality attributes in their functional requirements. Although we did not explicitly get any examples in our results about it, knowledge engineers have investigated the relation between functional requirements and quality attributes. They assert that in late stages of software development, quality attributes are integrated with functional requirements and that it is possible to provide support for separation of crosscutting functional and non-functional properties minimising conflicts arising due to tangled representations [20]. This is done by processing three main activities: identify, specify and integrate requirements. Firstly, the system could identify all the requirements input by the user and select the quality attributes (if any). Secondly, it would specify functional requirements, using a use case based approach, and describing quality attributes using special templates, identifying those that crosscut functional requirements. In

[26] *"EARS: The Easy Approach to Requirements Syntax ... - iaria." 21 Jul. 2013, https://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf. Accessed 5 Aug. 2017.*

the final step, a set of models would be used to represent the integration of crosscutting quality attributes with the functional requirements. In the scope of the thesis, we have not yet investigated this issue.

Profiling based on user input

Both the RI engineers and the CS engineers seem to be more confident with the science viewpoint in the *term & scenario expertise* part of the experiment. The amount of entities that the participant can choose from to answer these questions is limited to five entities per viewpoint. Since each viewpoint has many more concepts, this part of the experiment congests the viewpoints into a very tight area of choice. Therefore, the use of other arbitrary concepts could have had altered the results.

The *requirement referencing* test provided the opportunity for users to express their requirements freely. However, the previous section could have influenced the users choice of requirement by applying a cognitive bias known as the anchoring effect[27]. This is a tendency to rely too heavily on the first piece of information perceived when making decisions. Fortunately, after testing it for this effect (appendix E), it was seen that there was not a significant difference between viewpoints within each class. In other words, the final result was not altered by the phenomena since each viewpoint was affected in an equal way. On the other hand, the RI engineers results were 24% more influenced by the anchoring effect than the CS engineers. This could be due to the fact that 83% of CS engineer participants were proficient (level of 4 or 5) with the reference model, in contrast to the 40% of RI engineers. This would address the possibility that proficient users had more variety of entities in their descriptions since they know the reference model better and have a broader extension of objects from which to choose from. On the other hand, this is debunked by the following observation: Participants made use of more of the keywords provided when describing the system than when asked to identify the ones they had more expertise with. This tells us that even though person A could understand the utility of a component better than person B (due to higher proficiency with the reference model), it does not mean that person B would not address the component in a requirement specification.

The unpredictability of the users choice of object from the different viewpoints would make it difficult to classify a user in a particular profile, although an approximation could certainly be done. When the RI engineers participants input requirements in a free manner, they address the science viewpoint and the CS engineers address the information viewpoint. The differences between each viewpoint within a profile is not significant so the approximation might not be always correct.

[27] "Anchoring - Wikipedia." https://en.wikipedia.org/wiki/Anchoring. Accessed 8 Aug. 2017.

It should also be mentioned the human error produced by a qualitative analysis and the problems a system could face analysing natural language. Ambiguity of data could arise since readers and writers may not interpret words in the same way. Over-flexibility is also common, meaning that the same thing has many types of ways to be expressed. Requirements could be confusing since they might address many things at the same time or mix functional with non-functional requirements.

7 CONCLUSIONS

Based on the experiments, we can conclude:

1. Expert systems usability has been investigated in the past in a broad scope. They were based on attributes such as learnability, memorability or satisfaction that encapsulated the expert system as a whole (input & output); in contrast to the focus that we wanted to give to the input method in this thesis. Therefore, we set up experiments to test the actual communication methods between user and machine. The results showed that both the question-based and natural language input methods had some functionality that the participants appreciated using. We deducted from both the results and Gaines and Shaw's guidelines (section 2.2) that the question-based approach provided the user with a closer personification of an expert architect and the text-based with a way for the user to express themselves. As stated by one of the usability guidelines, combining different knowledge bases and simulation techniques is extremely powerful to provide a knowledgeable environment to the user. Due to this, we suggest combining the two input methods together into one, since they would complement each other in providing a better knowledgeable environment.

2. Natural language input methods take into account issues related to functionality rather than usability. The distinction between structured-text and free-text have not been investigated for usability in previous investigations. Therefore, in this thesis we test this feature to find out which natural language input method is the preferred one by users. The results showed that the structured input was favoured over free-text. We can affirm, due to the feedback received, that structured input is more relevant for novice users that are not experienced with requirement input. Other less common attributes addressed by participants, that at the same time are benefits of template usage [18], were; the increase in speed, consistency and reduction of errors in the data input into the system.

3. Knowledge bases play a key role in expert systems, but the communication with the front-end varies. In this thesis we wanted to investigate the communication between these two entities depending on the requirements input by the user. The results showed that querying the RM in a quality attribute driven approach would have not worked as expected. What can be done is implementing a complementary search for non-functional requirements or differentiating crosscutting functional and non-functional properties from single requirements.

4. Profiling members of the ENVRI community has been performed in the past, although this entitled executing face-to-face interviews with each member. In this thesis we wanted to investigate the profiling of users through the requirements they input. After performing the experiment, we can conclude that an approximation of the classification of the user into a particular profile can be done. Users that reference the science viewpoint the most can be classified into the RI Engineer profile and users that reference the information viewpoint can be classified into the CS Engineers profile. Must be emphasised that this is an approximation due to two factors. First, the difference between the first and second most referenced viewpoint is not significant. Second, the expertise of a user with a specific viewpoint is not directly proportional to the number of references to this viewpoint in the requirements; increasing its unpredictability.

8 Future Work

In the future, there are a number of action points that we would like to work on. Currently the visualisation for the natural language input method returns individual component recommendations without links between them. In the future a full architectural model could be generated in order to add value to the expert system.

The portal analysis requirements through key word querying. In the future a more sophisticated way to analyse requirements could be developed to achieve more complex output recommendations, simulating an expert architect with a higher degree of accuracy.

A login system with customisable GUI could be implemented. Each user would be able to save architectural designs generated by the system for further referencing. This would make the system more engaging and immersive than the current prototype where every user gets the same GUI.

By using the current profiling of users approach, it could be possible to provide specific recommendations for each user profile. Again this would make the system feel more *personal* with the user, increasing user satisfaction.

References

[1] ENVRI Reference Model Documentation. 2016. 15-76

[2] Chen, Y., Hardisty, A., Preece, A., Martin, P., Atkinson, M., Zhao, Z., Magagna, B., Schentz, H. & Legre, Y. (2013). Analysis of Common Requirements for Environmental Science Research Infrastructures. ISGC 2013. 6-8.

[3] Jay Liebowitz. Expert systems: A short introduction, Engineering Fracture Mechanics. 1995. 601-607.

[4] Grahovac, D. and Devedzic, V. COMEX: A cost management expert system. Elsevier Ltd. 2010.

[5] Lee, K.C. and Lee, S. A causal knowledge-based expert system for planning an Internet-based stock trading system. Expert Systems with Applications. 2012. 8626-8635.

[6] C.F. Tan, L.S. Wahidin, S.N. Khalil, N. Tamaldin, J. Hu and G.W. M. Rauterberg. The application of expert system: A review of research and applications. ARPN Journal of Engineering and Applied Sciences. 2016. 2448-2449.

[7] Rogers Y, Sharp H, Preece J. John Wiley Sons. Interaction Design: Beyond Human-Computer Interaction. 2011. Chapter 1, 8, 12.

[8] Jurusan Teknik Elektro dan Teknologi Informasi. Paulus Insap S. Bahan Kuliah Teknik Komputer Interaktif. Fakultas Teknik Universitas Gadjah Mada. 2010.

[9] Dix et al. Interaction design basics. In: Human-Computer Interaction, Prentice-Hall. 2004.

[10] Usability 101: Introduction to Usability. Jakob Nielsen. 2014 Jan 2012. Available from: http://www.useit.com/alertbox/20030825.html.

[11] C. P. C. Munaiseche and O. E. S. Liando. Evaluation of expert system application based on usability aspects . Department of Information Technology and Communication Education Faculty of Engineering, Universitas Negeri Manado. 2016. 2-11.

[12] Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramania, Anand Rajaraman, Venky Harinarayan, AnHai Doan. Building, Maintaining, and Using Knowledge Bases: A Report from the Trenches. University of Wisconsin-Madison. 2-10.

[13] Reid G. Smith. Knowledge-Based Systems, Concepts, Techniques, Examples. *Knowledge acquisition*. 8-16. 38-49. 1985

[14] Galina Datskovsky Moerdler, Kathleen R. McKeown and J. Robert Ensor. Building Natural Language Interfaces for Rule-based Expert Systems. Columbia University.

[15] Jocelyn Ireson-Paine. How the inference engine works. 1996.

[16] Greg Barish. Scalable and high performance Web Applications - Measuring scalability. 2002.

[17] Brian R. Gaines. Designing Expert Systems for Usability. Knowledge Science Institute. University of Calgary. 1-10, 24-32.

[18] The benefits of using a template. Available from: "http://www.sage.com/us/Sage-Advice/Articles/18369/2015/8/13/The-benefits-of-using-templates". SAGE.

[19] Dietrich, C. Decision Making: Factors that Influence Decision Making, Heuristics Used, and Decision Outcomes. 2010. 1-8

[20] Ana Moreira and Joao Araújo, Isabel Brito. Crosscutting Quality Attributes for Requirements Engineering. 2002. 1-9

[21] K. Zeroual. KBRAS: A Knowledge-based Requirements Acquisition System. 38-45.

[22] Ian Sommerville . Software Requirements. 2003. 1-9.

[23] Malcolm Atkinson, Alex Hardisty, Rosa Filgueira, Cristina Alexandru, Alex Vermeulen, Keith Jeffery, Thomas Loubrieu, Leonardo Candela, Barbara Magagna, Paul Martin, Yin Chen, Margareta Hellström. ENVRI Plus - Deliverable 5.1: A consistent characterisation of existing and planned RIs. 2016. 46-59. 114.

[24] Scott M. Angelo. Security Architecture Model Component Overview. 2001.

[25] Len Bass and Bonnie E. John. Supporting Usability through Software Architecture. 2001.

[26] Edward Crawley, Olivier de Weck, Steven Eppinger, Christopher Magee, Joel Moses, Warren Seering, Joel Schindall, David Wallace, Daniel Whitney (Chair). The influence of architecture in engineering systems. The ESD Architecture Committee. 2004. 18-23.

[27] Yevgeniy Brikman. Hello, Startup: A Programmer's Guide to Building Products, Technologies, and Teams. 2015. Chapter 7.

[28] Aurora Constantin, Malcolm Atkinson. Investigating users' needs to improve the use of the ENVRI RM. ENVRI week 4, Grenoble. 16 May 2017.

Appendix A

An activity diagram showing the requirement gathering process for the structured-text input method of the portal.

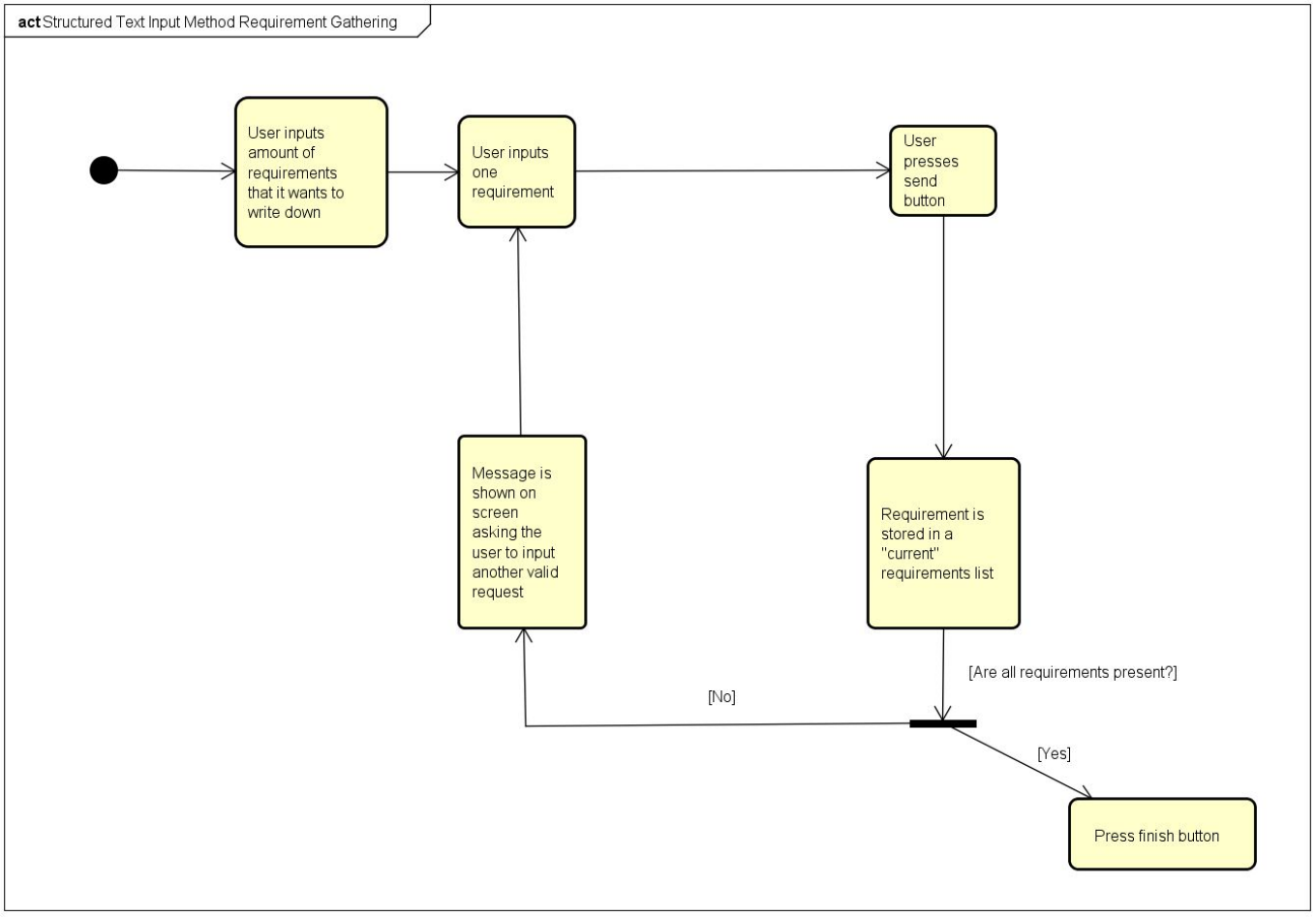


Diagram 3: Activity Diagram for the structured-text input method.

An activity diagram showing the requirement gathering process for the natural language input method of the portal.

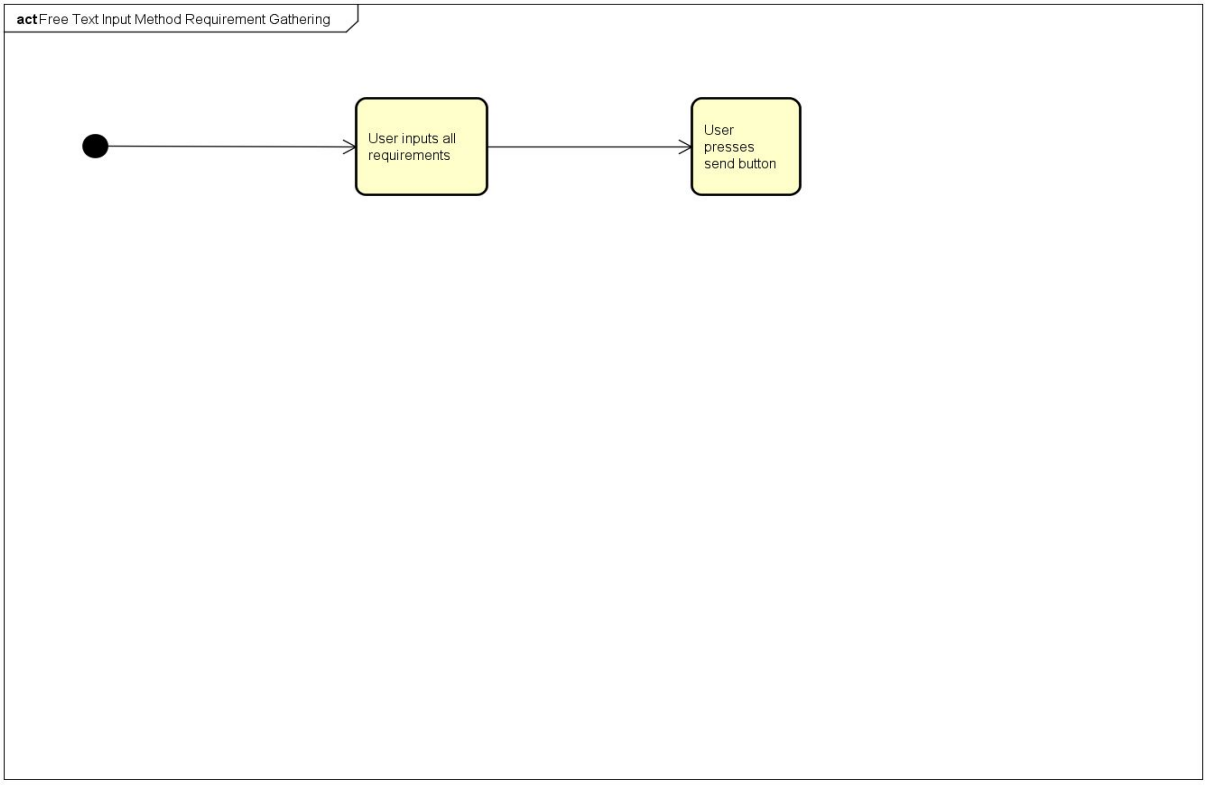


Diagram 4: Activity Diagram for the natural language input method.

An activity diagram showing the requirement gathering process for the question-based input method of the portal. A final node is one that contains an output visualisation (product of a set sequence of responses to the answers).

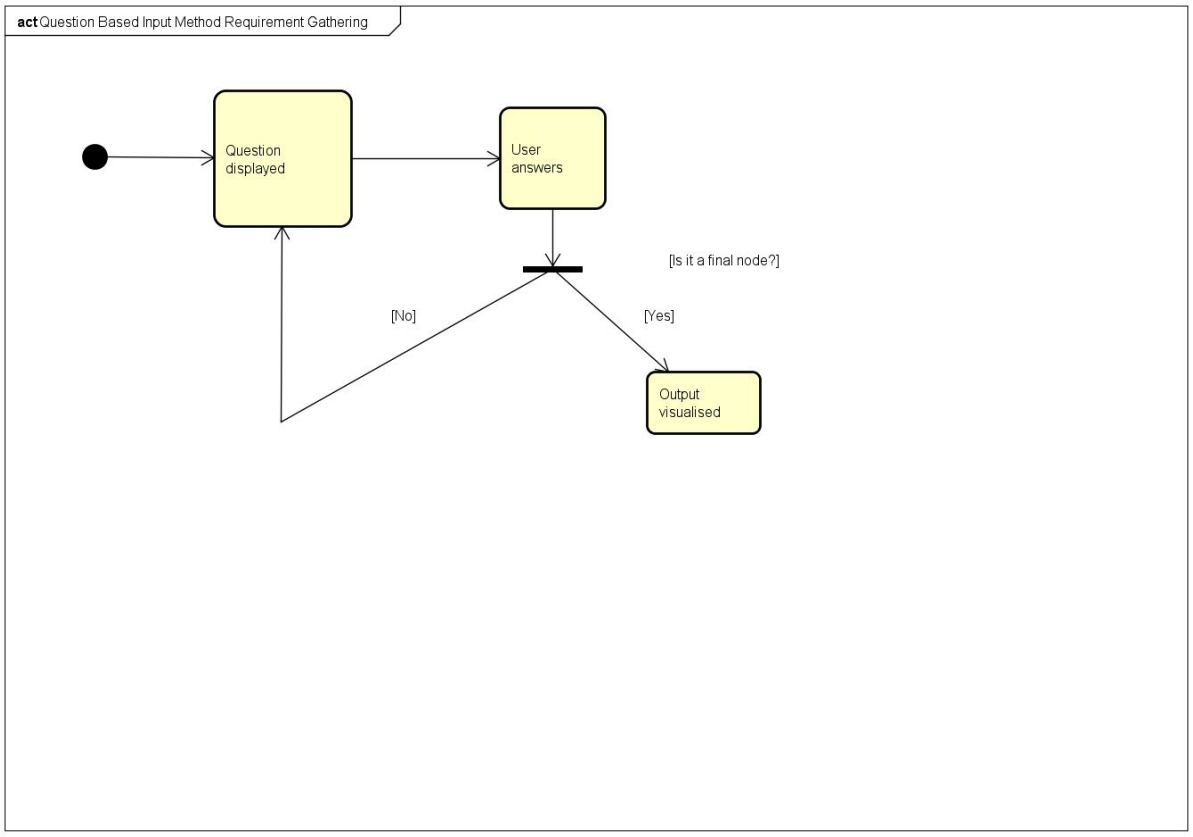


Diagram 5: Activity Diagram for the question-based input method.

Appendix B

There exists several properties or quality attributes which describe different facets of computer systems. These are categorised into four quality areas; design quality, run-time quality, system quality and user quality. Inside each category we can find the different quality attributes[28].

We identified several generic architectural components and design patterns (shown in table 1) that influenced the following quality attributes: security[29], usability[30], efficiency[31] and scalability[32]. These four attributes were chosen since they have a greater role in system architecture and direct value for the customer than, for example, testability.

Security [24]	Usability [25]	Efficiency	Scalability [26][27]
Policy pattern	MVC, PAC, Seeheim	Cache-Aside	Divide and Conquer
Authenticator	Cancellation Manager	CQRS	Caching
Authorizer	Prior State Manager	Event Sourcing	Co-Location
	View	Index Table	
	Active Command	Priority Queue	

Table 1: Architectural components that influence security, usability, efficiency and scalability quality attributes.

[28] "Chapter 16: Quality Attributes - MSDN - Microsoft." https://msdn.microsoft.com/en-us/library/ee658094.aspx. Accessed 5 Aug. 2017.
[29] "Security Architecture Model Component Overview - SANS Institute." 13 Aug. 2001, https://www.sans.org/reading-room/whitepapers/basics/security-architecture-model-component-overview-526. Accessed 5 Aug. 2017.
[30] "Usability-Supporting Architectural Patterns - Carnegie Mellon School ...." 19 Feb. 2004, http://www.cs.cmu.edu/~bej/usa/publications/ICSEtutorial04-final.pdf. Accessed 5 Aug. 2017.
[31] "Cloud Design Patterns | Microsoft Docs." 26 Jun. 2017, https://docs.microsoft.com/en-us/azure/architecture/patterns/. Accessed 5 Aug. 2017.
[32] "Hello, Startup: A Programmer's Guide to Building Products ...." http://www.hello-startup.net/. Accessed 5 Aug. 2017.

Security components are those ones that manage data accesses and log how the system is being used. The usability components are more focused to the use of graphical interfaces and user interactions, for example the cancellation manager and prior state manager would allow the user to cancel a previous command and return to the un-altered previous action. The components that would make the system efficient, in the most part, have to do with organization of data and storage management. For example making use of caches or index tables to store data and then using a priority queue to organize inputs and outputs. Scalable patterns such as co-location make it easier for the system to increase in size with not so many dependencies.

Appendix C

Examples of expert architecture questions:

Do you want to build a new system?
Why do you want to build a system?
How much time and capital are you willing to invest to maintain the system?
If you are thinking of extending the new system, what functions or activities are you thinking to host?
How many of those functions do you think you need?
What do you think the extension/renovation/new system should look like?
What do you envisage in your new system that your present one lacks?
How much can you realisitcally afford to spend?
How soon would you like to have the new system?
If you are thinking of building a system, do you have a deployment site selected?
What are your design preferences?
Any specific availability features you are willing to implement?
Does the system ned to be flexible by increasing its complexity?

Appendix D

List of requirements from RI Engineer/Scientist participants

Metadata (data, sensors, environment, protocol) acquisition to IS workflow provenance Standardized automated procedures gap filling standards, semantics provenance Data policy IP, legal/ethical, license DOI cataloguing **Accessible data repository** standard file formats semantics **interoperability** VRE computing e-resources	(raw) data identification data citation data product generation automatic data quality check data versioning data storage citation data acquisition information data annotation metadata harvesting Resource registration data conversion semantic harmonisation data discovery and access data analysis modelling and simulation support data mining API to be connected with external services provided by third parties
Data Collector PID Generator Conceptual model Metadata Harvester PID Generator Persistent data Data Store Conceptual model Metadata Harvester Persistent data Data Store Conceptual model Metadata Harvester Catalogue service Service Provider Data Mining Data Store Conceptual model Metadata Harvester Catalogue service Data broker Service Provider Data Mining	Data Collector Metadata Harvester Conceptual model PID Generator Data Store Metadata Harvester Conceptual model Catalogue service Catalogue service Data Mining Metadata Harvester Data Store Data broker Data Mining Service Provider Conceptual model

organising the activity sensors ensure real-time data acquisition **improve data accuracy** support curation process impose a standard/consistency for metadata (e.g. metadata format) **ensure data protection** **accessibility of data** improve data visualisation flexibility in the location of data processing flexibility for exploring various algorithms/software to process data

Table 3: RI Engineers requirements

References to viewpoints by RI Engineer/ Scientists

Science	**Information**	**Computation**
Data storage x6 Data citation x2 Metadata harvester x6 Service provider x4 Data mining x5 Data quality checking Data product generation Semantic harmonisation Data discovery and access Data analysis Modelling and simulation Semantic interoperability	Conceptual model x6 Data versioning Data acquisition Information Metadata format Data mining x5 Data visualisation Conversion Semantic harmonisation Provenance x2 Semantic interoperability	Raw data identification Data citation x2 PID x3 Persistent data x2 Data storage x6 Catalogue service x4 Data collector x2 Sensors x2 Data annotation Data broker x2 Cataloguing

Table 4: RI Engineers viewpoints references

List of requirements from CS Engineer/Scientist participants

Communication with data providers Data filtering Assignment of provenance Implementation of storing scripts **Verification of reusability of data (reuse Already processed data)** **Ensure that data are accessible from the platform** **Ensure that data are accessible from outside the platform** Provision of services for harvesting, compressing and packaging data Provision of query and search tools. Provision of tools for authorisation and authentication policies Provision of services for analysis, data mining and visualization Provision of services for modeling and simulation	Good tools supporting provenance tracking Good implemented QA methodologies at the sensors level Clear metadata description about the Investigation design Good services for Quality Control PID Service Data Annotation Service supported by semantic resources Semantic Harmonisation if needed Query about provenance Faceted search about measurement context Virtual laboratory
Metadata from Sensors and Sensor Networks Metadata from Observer/Experimenter Cataloguing services Identification services Provenance Tracking Facilitate Data Citation/Referencing (AAAI) Access management User interfaces (VRE/eLaboratory) [Scicentific] Workflow management Provenance Tracking Submitting jobs to suitable processing centres **Optimise data transfer for staging** **Optimise data transfer for results**	To be able to record real-time data from multiple instruments. To record the provenance of all data acquired. To be able to attach metadata to any data collection. **To have a long-term preservation policy for sustainability.** To be able to search data collections by a range of different criteria. To be able to identify datasets globally via persistent identifiers. To provide access to a preconfigured processing platform. To record the workflow executed to process data for provenance purposes.
Sensor network or another research infrastructure Agreed data syntax and semantics between system serving data and the RI	Need sensors, observation or measurer Need metadata Need PID Need quality control Need preservation

Agreed data transfer protocol between system service data and the RI Storage system capable of managing the data heterogeneity and volume characteristics in the RI Mature workflows for data quality control Workable approach for data identification Data access interfaces that meet common standards Data representation (syntax and semantics) that meet common ontologies Data access via persistent identifiers. Fast and intuitive search functionality (metadata and possibly data). A flexible approach (probably some kind of virtual research environment) were researchers can process data. Well-defined and well-implemented workflows for data processing within the RI Support for provenance	Need storage Need metadata Need curation manager Need PID Need catalogue Need publisher Need data management plan Need workflow management Need provenance Need workflow coordination Need computing infrastructure

Table 5: CS Engineers requirements

References to viewpoints by CS Engineer/ Scientists

Science	**Information**	**Computation**
data providers Observer Measurer Semantic harmonisation x2 Data collection Storage system Storage Data citation Data collections Publisher Data mining Modelling and simulation Scientific workflow management Researchers Workflow management	Data filtering Assignment of provenance Provenance tracking x3 QA x2 Investigation design Record provenance of data Semantics Need metadata Data accessible Quality control x2 Data annotation service Semantic harmonisation Identification services Long term preservation Data identification Preservation Metadata	Sensors x2 Sensor networks x2 Record real-time data from multiple instruments Data transfer protocol PID x3 Cataloguing x2 Curation manager Query and search tools Authentication tools AAAI User interfaces Persistent identifiers x2 Interfaces Virtual Laboratory Data transfer x2

	Harvesting Query provenance Measurement context Syntax and semantic metadata search Data management Data mining Record provenance Provenance x2	Processing platform Coordination Computing infrastructure

Table 6: CS Engineers viewpoints references

Appendix E

These are the results for the anchoring effect test in extension 2 from section one to section two.

Keywords for Science Viewpoint	No. of Appearances RI	No. of Appearances CS
Data Collector	0	1
PID Generator	0	0
Metadata Harvester	6	0
Service Provider	4	0
Citizen Scientist	0	0

Table 7: Anchoring effect for the science viewpoint keywords

Keywords for information Viewpoint	No. of Appearances RI	No. of Appearances CS
Persistent Data	0	0
Conceptual Model	6	0
Semantic Harmonisation	1	1
Provenance Tracking	0	3
Data Mining	5	1

Table 8: Anchoring effect for the informational viewpoint keywords

Keywords for Computational Viewpoint	No. of Appearances RI	No. of Appearances CS
Virtual Laboratory	0	1
Data Broker	2	0
Catalogue Service	4	0
Data Store	6	0
Sensor Network	2	0

Table 9: Anchoring effect for the computational viewpoint keywords