MUSTER: A Situational Tool for Requirements Elicitation

Chad Coulin, Didar Zowghi, Abd-El-Kader Sahraoui

MUSTER: A Situational Tool for Requirements Elicitation

Chad Coulin[1,2], Didar Zowghi[1] & Abd-El-Kader Sahraoui[2]
[1]University of Technology Sydney (Australia), [2]LAAS CNRS (France)

ABSTRACT

In this chapter we present a collaborative and situational tool called MUSTER, that has been specifically designed and developed for requirements elicitation workshops, and which utilizes, extends, and demonstrates a successful application of intelligent technologies for Computer Aided Software Engineering and Computer Aided Method Engineering. The primary objective of this tool is to improve the effectiveness and efficiency of the requirements elicitation process for software systems development, whilst addressing some of the common issues often encountered in practice through the integration of intelligent technologies. The tool also offers an example of how a group support system, coupled with artificial intelligence, can be applied to very practical activities and situations within the software development process.

KEYWORDS

Requirements, Elicitation, Workshops, Situational, Collaborative, Intelligent, Tool, MUSTER

INTRODUCTION

Requirements elicitation is a fundamental part of the software development process, but often considered a major problem area, and widely regarded as one of the more challenging activities within the scope of Requirements Engineering (RE). Heavily dependent on the experience and expertise of the participating analyst, the elicitation of requirements is often performed badly in practice, as true experts in this area are few and far between. The subsequent effects of poor software requirements elicitation regularly include costly rework, schedule overruns, poor quality systems, stakeholder dissatisfaction, and project failure (Hickey & Davis, 2002). But despite the obvious need for an appropriate level of structure and rigor, this critical, complex, and potentially expensive activity is more commonly performed in an ad-hoc manner, without a defined process or methodology.

Furthermore, many of the current techniques, approaches, and tools for the elicitation of requirements are either unknown or too complex for novices, and a general unwillingness to adopt them by industry, results in a significant gap between requirements elicitation theory and practice (Hickey, 2003). Just as important is the current gap between expert and novice analysts, which can be attributed to a number of factors, not least of which is the extensive skill set and range of experiences that is often required to successfully conduct this difficult yet vital activity (Hickey & Davis, 2003). A lack of systematic methods with situational process guidance, and supporting tools that can easily be applied to real-world situations, are additional reasons for the current state of requirements elicitation in practice.

Subsequently, in this chapter the MUSTER tool is presented, which embodies and enhances the situational OUTSET approach for requirements elicitation (Coulin, Zowghi & Sahraoui, 2006; Coulin, 2007), and is based on the principles of Computer Aided Software Engineering, Computer Aided Method Engineering, Group Support Systems, and Artificial Intelligence. The purpose of this chapter is therefore to present an intelligent tool for software requirements elicitation workshops, which is both useable and useful to practicing analysts. However, the overriding intention of MUSTER is to improve the overall effectiveness and efficiency of the requirements elicitation process specifically for the development of software systems.

BACKGROUND

Computer Aided Software Engineering (CASE) tools support one or more techniques within a software development method (Jarzabek & Huang, 1998). These tools are attractive to use during activities such as design, coding, testing, and validation, mainly because of their potential to provide substantial gains in quality, productivity, management, and communication (Hoffer, George & Valacich, 2002). Furthermore, CASE tools have been found to be efficient in both research and practice for recording, retrieving, and manipulating system specifications (Pohl et al., 1994), partly by automating some aspects of the system development.

Computer Aided Method Engineering (CAME) tools support the construction and management of adaptable methods (Saeki, Tsuchida & Nishiue, 2000). These tools are useful in automating part of the process of engineering a method, to conduct one or more of the various system development activities, by reusing parts of existing methods (Saeki, 2003). In addition, CAME tools have shown to be successful in providing the appropriate amount of process guidance, based on the specific needs of software development problems and projects (Dahanayake, 1998).

A common criticism of CASE tools is that they do not provide appropriate supporting guidance for the development process (Pohl et al., 1994), which can be directly addressed by the integration of a CAME tool. This would result in a process-based environment whereby the users can select, create, and modify method components for specific system development activities, in addition to performing the required system development tasks. The Phedias environment (Wang & Loucopoulos, 1995), referred to as a “CASE shell”, was an early attempt at producing a combined CASE and CAME tool. This tool enabled a method to be modeled at a Meta-level (i.e. a CAME tool), and corresponding CASE tools

designed, developed, and integrated within this model and environment in order to provide support for the various activities (i.e. it was also a Meta-CASE tool (Alderson, 1991)). As a precursor to MUSTER, Phedias is of particular interest because it was specifically targeted towards the development of methods and models of non-functional requirements for software engineering.

Group Support Systems (GSS) (Nunamaker, Briggs & Mittleman, 1996), or groupware, on the other hand, when used within the context of development projects, typically takes the form of a software-based tool focused on supporting communication, coordination, and collaboration within a team working towards common goals, on interconnected workstations, in shared workspaces (Ellis, Gibbs & Rein, 1991). The use of a GSS is particularly appropriate because of the number of key functions often provided by groupware applications that correspond directly to many of the tasks involved in requirements elicitation. These include activities such as information sharing, document authoring, knowledge management, and providing a suitable framework for stakeholder interaction. Furthermore, Group Support Systems have been found to be highly successful in improving group meeting productivity, and outcomes in real world settings (Hickey, Dean & Nunamaker, 1999), as well as enabling larger groups to collaborate faster, particularly when matched with a specific requirements elicitation process (Hannola, Elfvengren & Tuominen, 2005).

Subsequently, the idea of combining a GSS with requirements elicitation has been relatively popular. In fact Hickey et al. state that the challenges of gathering accurate requirements, the inefficiencies of user interviews, and the difficulty of achieving effective group meetings, were early driving forces for GSS research (Hickey, Dean & Nunamaker, 1999). As a result, there has been significant attention in research directed towards integrating groupware and requirements elicitation (den Hengst, van de Kar & Appelman, 2004; Tuunanen, 2003; Venable & Travis, 1999), and of particular note are tools such as GroupSystems (Hickey, Dean & Nunamaker, 1999), which has been used to collaboratively define scenarios, AMORE (Wood, Christel & Stevens, 1994), which utilized advanced multimedia technology, and TeamWave (Herela & Greenberg, 1998), which specifically addressed distributed software development.

In (Liou & Chen, 1993), a GSS, the Joint Application Development (JAD) method, and CASE tools, were integrated to support the requirements specification process. In this work it was identified that support for requirements elicitation, and specifically collaborative requirements elicitation meetings, was a major function missing from CASE research and products. It was also acknowledged that in order for a GSS to be successful in supporting requirements elicitation, it must also be supported with an appropriate methodology for its use, however no such specific process guidance was supplied. Therefore, the additional integration of a CAME tool, would enable not only the development and utilization of a contextual and dynamic method, but also the integration of different techniques to support the overall requirements elicitation process.

As a result, the name ‘MUSTER’ was chosen for the combined CASE / CAME / GSS tool, because it aims to bring together or ‘muster’ the different tasks, data types and techniques of requirements elicitation for software development, into an integrated situational process within a workshop environment, in what is referred to in (Pohl et al., 1994) as a “process-aware CASE tool”. However, unlike most CASE tools, MUSTER is intended to be used by groups rather than individuals, by applying the additional

principles of groupware applications and workshop facilitation, combined with intelligent technologies.

DEVELOPMENT OF THE TOOL

There were a number of important constraints in the development of the MUSTER tool, which had significant impact on its design and construction. Firstly, the system had to be developed by the researchers, and with a timeframe acceptable to the overall schedule of the project. Because there was no budget allocated to the project, it was also necessary for the system to be developed using only available and free technologies. The system needed to implement the OUTSET approach, and support interactive and incremental requirements elicitation workshops. Furthermore, it was decided that the system should be as platform independent as possible, and thereby be able to run on most standard computer hardware platforms and operating systems.

A first prototype of the MUSTER tool was constructed, with a preliminary set of standard features, and a detailed list of requirements elicitation related tasks. This prototype was tested and evaluated at a relatively high-level by numerous people both familiar and unfamiliar with the larger research project, including the research supervisors and fellow researchers. Although the prototype was found to be structured and extensive, it was essentially static with only limited process flexibility. Based on this feedback, a second prototype was developed with considerably more focus on the functionality required to make the tool less constrictive, more dynamic, and offer appropriate situational support. This resulted in several changes to better support the underlying approach, and to provide a suitable foundation for the planned evaluations. The details of this final prototype are described in the following subsections.

High-level Requirements

The detailed list of requirements used for the development of the tool was based on the results of a literature review, a survey of practice, and the need to support the OUTSET approach, in addition to the "wish list" of 70 requirements for Requirements Engineering techniques proposed in (Macaulay, 1996). At a high-level, and in accordance with the goals of the research, the overall objectives of the tool were identified as; 1) improve the process of requirements elicitation in terms of the time and effort required, and 2) directly address some of the common issues and challenges often encountered in practice, such as an incomplete understanding of needs and ill-defined system boundaries. It is therefore important to note that the principle focus of the tool is to improve the process of requirements elicitation, rather than improve the quality of the results, although better quality results are expected from improvements to the elicitation process.

The main functional areas required within the tool were established as 1) visualization, navigation, and administration through the elicitation process, 2) externalization, representation and organization of the elicited information, 3) process guidance, 4) cognitive support, 5) task automation, 6) interaction assistance for the participants, and 7) education of the users on requirements elicitation, primarily by osmosis. Although the

primary usage of the tool is by an analyst to facilitate group workshops, it was determined that the system should also be usable during a traditional one-on-one interview by the analyst with a stakeholder, as well as offline and independently by both the participating analyst and stakeholders. Although this functionality did not affect the design of the tool in any major way, it did provide the tool with an extra and potentially useful element of flexibility.

Architecture and Technologies

The application of artificial intelligence (AI) during requirements elicitation offers the potential to provide the type of help a novice analyst might receive from being mentored by an expert, and stakeholders with the kind of advice and guidance offered by a specialist workshop facilitator. This idea is supported in (Scott & Cook, 2003), where a classical blackboard system with autonomous agents based on a knowledge repository is suggested in order to achieve such goals. Because Requirements Engineering, and especially elicitation, is essentially a cognitive activity, AI presents an appropriate opportunity to address this activity by providing situational cognitive support to both the analyst and stakeholders (Zeroual, 1991). This is confirmed by (Maiden & Sutcliffe, 1993) which states that "requirements engineering is complex, error-prone, and in need of intelligent tool support to assist the capture, modeling and validation of requirements".

Subsequently, two basic architectural orientations were identified as being potentially suitable for the development of the MUSTER tool and its intelligent components, being 1) a Multi-agent system such as JACK and JADE, or 2) an Expert system using Lisp or Prolog for example. It was also determined however that the concept of 'intelligent plug-ins' was not only similar to that of having multiple agents work cooperatively, but was also consistent with the operation of a partitioned expert system. Furthermore, the use of a plug-in architecture would provide many of the advantages of both Multi-agent and expert systems (e.g. the ability to use artificial intelligence), and at the same time enable a much wider choice in the selection of implementation technologies.

Following a thorough evaluation of available options, it was decided that the tool would be an online browser-based application, and the sever-side components would be based on the LAMP platform (Linux operating system, Apache web server, MySQL database system, PHP scripting language) with HTML (Hyper Text Markup Language), JavaScript, VBScript, and CSS (Cascading Style Sheets) incorporated where necessary. The advantages of the specific technologies chosen include the fact that they are easy to learn, use, and therefore maintain, and are entirely open source and completely free of charge, with extensive Internet based support networks. Furthermore, the amount of time required to produce a working prototype of the basic functionally required using PHP with MySQL was anticipated to be less when compared to the other option of a Java with XML based system. Because of the underlying environment, the tool would also be portable, scalable, and most importantly, flexible with respect to the integration of other technologies, tools, and components, necessary for a successful plug-in architecture.

Data Repository

The foundation of the tool is a central Data Repository (DR), which enables the storage and retrieval of large amounts of requirements, and requirements-related data, elicited from the stakeholders during the workshops, as well as configuration information about the tool, projects, and users.

User Interface

The User Interface (UI) of the tool provides the ability to navigate through the required tasks, whilst interacting with other components of the system. As can be seen in Figure 1 below, the 'Home' screen displayed after a successful login has three major areas, as described in detail later, being 1) the Task List, 2) the Main Window, and 3) the Advice Panel. In developing the UI, a number of recognized Internet resources were used, including (Rolston, 2005), to ensure that the overall look and feel of the tool was both simple and consistent.

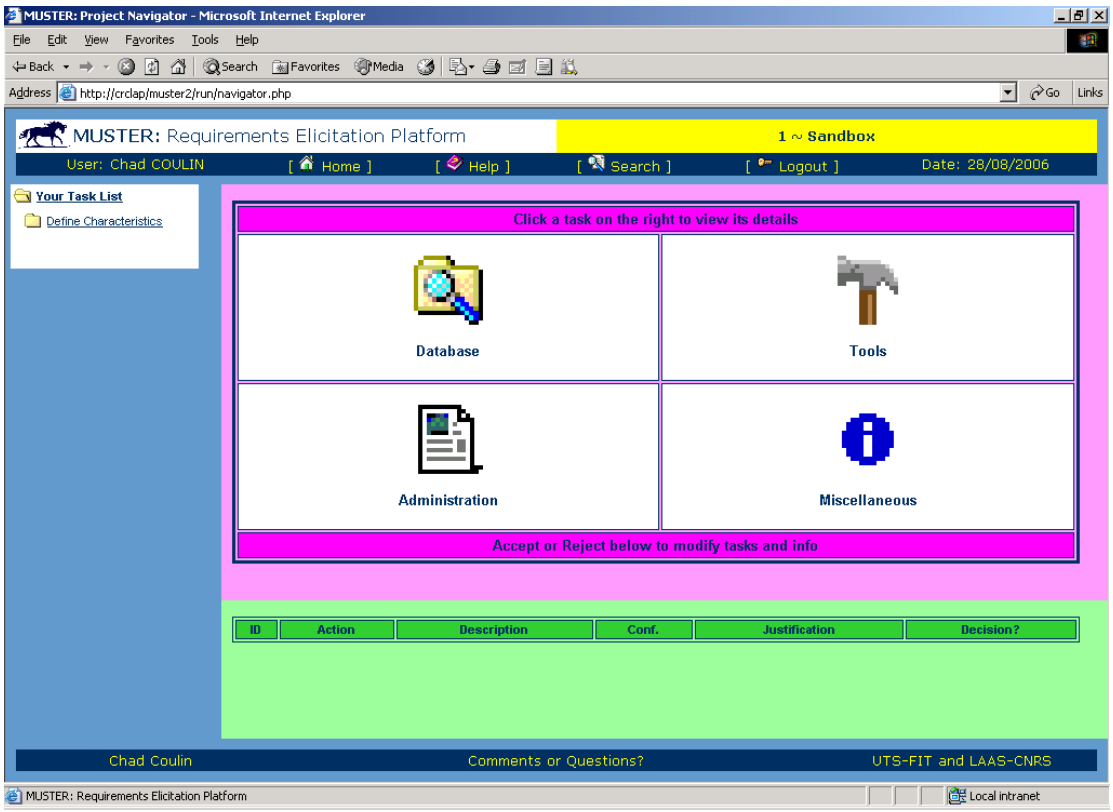

Figure 1. The MUSTER tool 'Home' screen

The Task List (Figure 1, left hand side) provides a dynamically generated list of tasks for requirements elicitation process navigation and execution, which the workshop participants are recommended to perform during the various sessions. This task list is populated by the support plug-ins in accordance with the underlying approach. Each task may be composed of several subtasks, and each task and subtask may have its own corresponding and specific Main Window.

The content of the Main Window (Figure 1, centre right hand side) is dependent on the task currently selected. There is no restriction on what should be displayed for each task,

therefore the screen could be purely informational, or provide an editor for some related part of the Data Repository.

The Advice Panel (Figure 1, bottom right hand side) presents the users with situational advice generated in real-time by the support plug-ins, based on the current state of the data in the repository (see the "Plug-ins for the Tool" section below for more details).

Database Menu

The Database menu, accessible from the 'Home' screen, contains links to Data Repository (DR) maintenance screens for the available data types supported by the tool. By using these editors, which are based on the List-Detail-Post paradigm, users can select any data type in order to directly view, add, change, or delete specific entries within the DR for the current project.

Tools Menu

The Tools menu, also accessible from the 'Home' screen, contains the following features and functionalities:

- **Glossary** – enables the user to maintain a project glossary by being able to add, change, delete, and view definitions of terms, acronyms, and abbreviations.
- **Data Dictionary** – enables the user to maintain a record of data types within the project related to the system under investigation.
- **References** – enables the user to maintain references to other material sources related to the project and/or the target system.
- **Appendixes** – enables the user to maintain a list of the required appendixes to the deliverables that will be generated as a result of the project.
- **Issues** – enables the user to record project related issues that arise during the workshops, as well as their status, who they are assigned to, and their resolution.
- **Actions** – enables the user to record actions that need to be performed during the project, as well as their status, who they are assigned to, and their resolution.
- **Idea Hotpots** – enables all users to record and maintain miscellaneous suggestions and proposals related to the project in order for other users to respond and comment on them anonymously at any time.
- **Reports** – provides a list of onscreen and exportable reports, enabling the user to produce deliverables from the information stored in the Data Repository (DR), for the purpose of reviews, walkthroughs, and inspections.
- **Resources** – provides various additional resources and material for the users, including templates, examples, and checklists, in order to further support the process of requirements elicitation and the workshop participants.

Administration Menu

The Administration menu, also accessible from the 'Home' screen by users with Administrator access (typically the participating analyst only), contains the following features and functionalities:

- **Projects** – allows the user to maintain projects in the system.
- **Users** – allows the user to maintain user accounts in the system.
- **Sessions** – allows the user to record details of the various requirements elicitation sessions performed during a project, including the start time, end time, participants, and location, for generate reports and performance metrics.
- **Tasks** – enables the user to view, add, change, and delete tasks in the dynamically generated Task List for each project and workshop.
- **Plug-ins** – enables the user to view, add, change, and delete information about the support plug-ins of the system, includes being able to install, enable and disable them.
- **Configuration** – enables the user to view information about the various configuration parameters of the MUSTER system, and change their values.
- **Rules** – enables the user to maintain rules and rule sets used by the Processor (see Processor in the Miscellaneous Menu below for more information). The system has default sets of rules however these can also be customized.

Miscellaneous Menu

The Miscellaneous menu, also accessible from the 'Home' screen, contains the following features and functionalities:

- **Messages** – This feature enables the users to record, view, and update messages in the system for other project members, but primarily the participating analysts, the project managers, and the MUSTER system administrators.
- **Logs** – This feature captures and records all events and actions performed by the users and the system. These include logins, logouts, as well as Add, Change, and Delete operations on data in the repository.
- **Categorizor** – This feature endeavors to categorize user entered pieces of miscellaneous textual information into their most appropriate Info Type, based on a small Artificial Neutral Network (ANN) that utilizes a number of key and common word lists. The concept behind this feature is that novice analysts are sometimes unsure as to how to categorize elicited information, especially with respect to goals versus requirements, and functional requirements versus non-functional requirements, for example.
- **Processor** – This feature enables the user at any time during the project to run one or more sets of rules over the information in the Data Repository to check for aspects of quality such as completeness, consistency, etc. For example, this feature could be used to completeness by ensuring that at least one actor has been assigned to each Use Case description, or to check that each feature has one or more individual functional requirements associated to it.
- **Technique Selector** – This feature, which utilizes a simple weighted values criteria approach, provides support for the user in selecting which technique to use for a task prescribed by the process guidance. The Technique Selector takes into account several factors, including the skill level of the participating analyst, the current project situation, and the specific characteristics of the task at hand.
- **Ask REG** – REG (Requirements Elicitation Guide) is a web-enabled pedagogic agent based on the famous and competition winning A.L.I.C.E. chat-bot engine (A.L.I.C.E.

AI Foundation Inc, 2005), and AIML (Artificial Intelligence Markup Language). The intention is that REG acts as an interactive assistant by providing help for all MUSTER users, by responding to entered questions from a knowledge base of general information about requirements elicitation, and more specific information linked to a set of predefined topics and concepts.

PLUG-INS FOR THE TOOL

MUSTER system plug-ins provide the situational process guidance and intelligent cognitive support for the users during the workshop sessions. The primary role of these plug-ins is to add, change, or delete tasks and subtasks in the project Task List, however they can also provide suggestions to the users dynamically, proactively, and reactively, such as introducing tasks, describing relevant background concepts, offering tips and tricks, as well as being able to directly manipulate data in the repository. As a result, plug-ins can provide the users with process guidance, decision support, and knowledge acquisition assistance. Plug-ins may provide generic support, or be based on a specific process or task, as well as a particular method (e.g. SADT, SSM, UML), technique (e.g. Scenarios, Viewpoints, Goals), or system type (e.g. Information, Embedded, Critical).

All advice generated by the installed and configured plug-ins appears in the Advice Panel of the screen, together with a justification and a confidence rating, generated internally by the specific plug-in responsible for that particular piece of advice. The advice provided by the plug-ins can be based on the characteristics of the individual project and workshop, as well as the information already elicited and stored in the data repository. Each piece of advice from the plug-ins presented in the Advice Panel may be rejected or accepted by the users, and an upper and lower threshold for the confidence rating is configured as a project characteristic to determine which advice is automatically accepted, and which is automatically rejected.

Plug-in Architecture

Regardless of the technologies used for its actual implementation, the architecture of a plug-in requires the following four components:

1. Characteristics – these are additional (non standard) situational characteristics which may be used by the conditions of the plug-in to determine which advice should be offered and when. The user is requested by the system to enter values for each of the new characteristics when the plug-in is run for the first time.

2. Conditions – these represent the rules and logic of the plug-in, which is typically based on selected data from the repository, and the values entered for relevant characteristics. The conditions themselves can be implemented in almost any web-based technology, and can range from basic condition statements through to complex intelligent algorithms.

3. Advice – these are the specific pieces of situational and intelligent support that may be offered by the plug-in. These are based on the triggering of or results from the above mentioned conditions, and are presented to the users for action.

4. Action – these are the subsequent and prescribed results of accepting the offered advice. Each piece of advice offered by a plug-in will perform one or more action operations in the system if accepted. These typically take the form of modifications to the task list, or manipulation of the data in the repository.

All the plug-ins are run automatically when any major event in the system occurs, which typically involves an Add, Change, or Delete operation on data in the repository. New advice generated by running the plug-ins is added directly to the bottom of the list presented in the Advice Panel with a status of 'Open'. Once a piece of advice in the list has been either rejected or accepted by the users, and the appropriate actions taken by the system, it is marked with a status of 'Closed', and removed from the Advice Panel list.

Advice can be presented, and appropriate actions performed, using the standard functions available in the Plug-in Function Library (PFL). Operations supported by the PFL include adding a Task or Subtask, adding an Info Type, checking if a particular piece of advice has already been given, and checking if a particular piece of advice was previously accepted or rejected. Depending on the configuration of the plug-in, a piece of advice may be presented only once for each session whether it is accepted or not, or it may be presented multiple times if not previously accepted for that particular project.

Creating Plug-ins

The overall plug-in architecture for the MUSTER system provides the ability for the tool to store and use the knowledge of both methodologists and requirements elicitation experts, within a integrate environment.

Plug-ins can be built using any single or combination of technologies, provided they are executable via the web server through a web page, which includes C++, VB, and Java. As a result, a wide audience is able to design and develop plug-ins for the MUSTER system, since no specific or proprietary technology-based expertise is required. The level of support, in terms of scope and complexity, is also not restricted, and at the higher end of the scale, almost any type of soft computing and machine learning method could be used as the basis for a plug-in, such as Bayesian Conditional Probability, Case Based Reasoning, and Artificial Neural Networks.

The process of installing a new plug-in is as simple as copying the relevant executable web page and associated files into the 'Plug-ins' directory of the MUSTER system. Using the Plug-in Maintenance Utility, details of the specific plug-in are then added to the system, including its status, and the name of the executable file in the directory. The final step is to enter values to the plug-in specific characteristics, which are installed the first time it is run. It is important to note that all plug-ins can be enabled or disabled at any time during a project using this same utility.

Plug-in Example

The example 'Select Info Types' plug-in, as summarized in Table 1 below, provides advice for the users on which Info Types should be elicited for each workshop, and to what level of detail they should be investigated. This plug-in uses a small Artificial Neural Network (ANN) (Young, 2004) to determine the advice offered to the users, developed using a corpus of 15 example but real-world requirements documents from successful industrial projects, and several widely accepted and used requirement document templates. The ANN was trained by running 10 examples through 1000 times each, with the appropriate input values and corresponding output values, and the remaining 5 examples were then used to test the results of the trained ANN.

Table 1. Summary of the 'Select Info Types' plug-in

Name :	**Select Info Types**
Description :	Determines which Info Types should be elicited during the various requirements workshops
Technology :	Artificial Neural Network (ANN)
Characteristics :	Input Nodes for the ANN - 1) Project Size 2) Project Definition 3) Project Domain 4) Project Deliverable 5) Workshop Type 6) Cut-off Level
Conditions :	Output Nodes for the ANN - 1) Goals 2) Assumptions 3) Constraints 4) Environmental 5) Opportunities 6) Challenges 7) Risks 8) Stakeholders 9) Work Processes 10) Functional Aspects 11) Non-functional Aspects 12) Implementation
Advice :	Elicit each Info Type with a value above the cut-off level
Action :	Add Task for each Info Type suggested and accepted

In this case, the characteristics are used by this plug-in as input nodes for the ANN, and include the Project Size, Project Definition, Project Domain, Project Deliverable, and the Workshop Type. Each output node represents a potential Info Type, and the values generated by the ANN for a particular set of input node value, determines what Info Types are recommended for elicitation during the workshop. For each output node (Info Type) with a value above the cut-off level characteristic, an entry is displayed in the Advice Panel, which if accepted, will add a first-level task to the dynamic Task List stating the need to elicit that particular Info Type. Therefore, this plug-in determines which Info Types should be elicited (via the output node values) based on the specified characteristics (from the input node values).

Developed Plug-ins

This subsection contains descriptions of the initial set of plug-ins developed for the MUSTER tool. These plug-ins were developed both as a proof of concept for the designed and developed architecture, and for planned evaluations of the MUSTER tool.

New Requirements Elicitation Project – The 'New Requirements Elicitation Project' plug-in provides initial and core tasks and characteristics for new requirements elicitation projects. This plug-in primarily uses the '3Ds' characteristics from the OUSTET approach (i.e. Definition, Domain, and Deliverable), to determine which tasks should be performed, and was designed to use a basic rules approach, based on a variety of sources from the literature including (Sommerville & Sawyer, 1997) and (Robertson & Robertson, 1999).

Requirements Elicitation Workshop – The 'Requirements Elicitation Workshop' plug-in provides core tasks and additional characteristics for conducting requirements elicitation workshops. This plug-in also uses the '3Ds' characteristics from the OUSTET approach to determine which tasks should be performed, and was designed to use a rules approach based on a variety of sources from the literature such as (Gottesdiener, 2002).

Select Info Types – The 'Select Info Types' plug-in provides guidance on which Info Types should be elicited for the specific project and workshop, based on both project and workshop level characteristics. As described previously, this plug-in was developed using an Artificial Neural Network (ANN) (Young, 2004). A number of sources were used to design this plug-in including 15 example but real-world requirements documents from successful industrial projects, and several requirements specification templates including (IEEE, 1998a), (Atlantic Systems Guild, 2003), (IEEE, 1998b), and (Wiegers, 2003).

Select Goal Subtasks – The 'Select Goal Subtasks' plug-in provides guidance on which subtasks should be performed during the elicitation of both system and project goals. In addition to presenting information to the workshop participants about what a goal is, and how one should be stated, the plug-in also provides instructions on how to brainstorm and prioritize goals. This plug-in was based on separate sources for goal elicitation (Dardenne, van Lamsweerde & Fickas, 1993), brainstorming, and prioritization (Wiegers, 2007).

Goal Investigation – The 'Goal Investigation' plug-in assists users to both decompose goals (by suggesting AND and OR relationships), and elaborate on goals (by proposing 'Why' and 'How' questions), in order to refine them in such a way as to elicit precise goals and related requirements. Several goal-based techniques for requirements elicitation were used as the basis for this plug-ins including (Yu, 1997) and (Dardenne, van Lamsweerde & Fickas, 1993).

Example BIS Constraints – The 'Example BIS Constraints' plug-in provides users with general and example constraints that are common or typical in software development projects for Business Information Systems. The intention of this plug-in is for the workshop participants to use the example constraints presented as the basis for the elicitation of similar constraints specific to the project at hand. The plug-in was designed based on the constraints listed in a relevant subset of 15 requirements documents from successful projects, and a variety of other sources, then implemented using a rules approach.

Use Case Questionnaire – The 'Use Case Questionnaire' plug-in proposes questions and provides suggestions to users on Use Cases that the system under investigation should support. This plug-in was implemented using Bayesian Conditional Probability (Meagher, 2004), and uses Use Cases previously elicited and stored in the data repository, with their corresponding characteristic values, as the basis for questioning the workshop participants about the required Use Cases, and suggesting additional Use Cases to include.

IEEE Functional Headers – The 'IEEE Functional Headers' plug-in uses an Artificial Neural Network (ANN) (Young, 2004) to determine the most appropriate way to group functional requirements (i.e. by mode, by user class, by object, by feature, by stimulus, or by functional hierarchy), based on the options presented in the IEEE Recommended Practice for Software Requirements Specifications (IEEE, 1998a). Coded values for project characteristics are used as the input nodes of the ANN, with the weighted value of the output nodes representing the relative appropriateness for each of the available options for grouping the functional requirements.

Features Questionnaire – The 'Features Questionnaire' plug-in proposes questions and provides suggestions to users on features that the system under investigation should include. This plug-in was implemented using Bayesian Conditional Probability (Meagher, 2004), and uses features previously elicited and stored in the data repository, with their corresponding characteristic values, as the basis for questioning the workshop participants about the required features, and suggesting additional features to include.

Feature Investigation Questionnaire – The 'Feature Investigation Questionnaire' plug-in dynamically generates a project specific questionnaire, to be used by the workshop participants for the elicitation of functional requirements. A list of high-level questions is created by the plug-in from a simple rule set developed from the researcher's experience, which can then be used to investigate and decompose each feature that has been elicited into specific functional requirements.

Non-functional Requirements – The 'Non-functional Requirements' plug-in provides users with additional support information for the elicitation of non-functional requirements. This includes a number of definitions and a list of typical non-functional requirements types gathered from a number of sources in the literature including (Sommerville, 2001), with relevant examples.

Example BIS Non-functional Requirements – The 'Example BIS Non-functional Requirements' plug-in provides users with general and example non-functional requirements that are common or typical in software development projects for Business Information Systems. The intention of this plug-in is for the workshop participants to use the example non-functional requirements presented as the basis for the elicitation of similar non-functional requirements specific to the project at hand. The plug-in was designed based on the non-functional requirements listed in a relevant subset of 15 requirements documents from successful projects, and a variety of other sources, then implemented using a rules approach.

MUSTER TOOL IN ACTION

The following section provides a basic walkthrough of the functionality and usage of the MUSTER system using a typical but simple project for the custom development of a small business information system. Although not all the features and utilities are demonstrated, this example does provide an overview of the general process of requirements elicitation using MUSTER within a workshop environment. In accordance with the process framework prescribed by the OUTSET approach, the following illustration of the system has been divided into three simple stages being 1) Preparation - setup of the new project in MUSTER, 2) Performance - running the workshops using MUSTER, and 3) Presentation - production of the requirements document from MUSTER, as described below.

Preparation

The first step in the preparation of a new MUSTER project is for the participating analyst to log into the maintenance utility of the system using an account with Administrator privileges. From the MUSTER Maintenance Utility menu, the analyst can then add a new Project and new Users to the system, as well as selecting the appropriate plug-ins to use. Each plug-in has a default status (either 'Active' or Disabled'), which provides a standard configuration of active and disabled plug-ins for new projects, that can be then modified by the analyst. As can be seen from Figure 2, only some of the installed and available plug-ins within the system have been enabled by the analyst for this particular project.

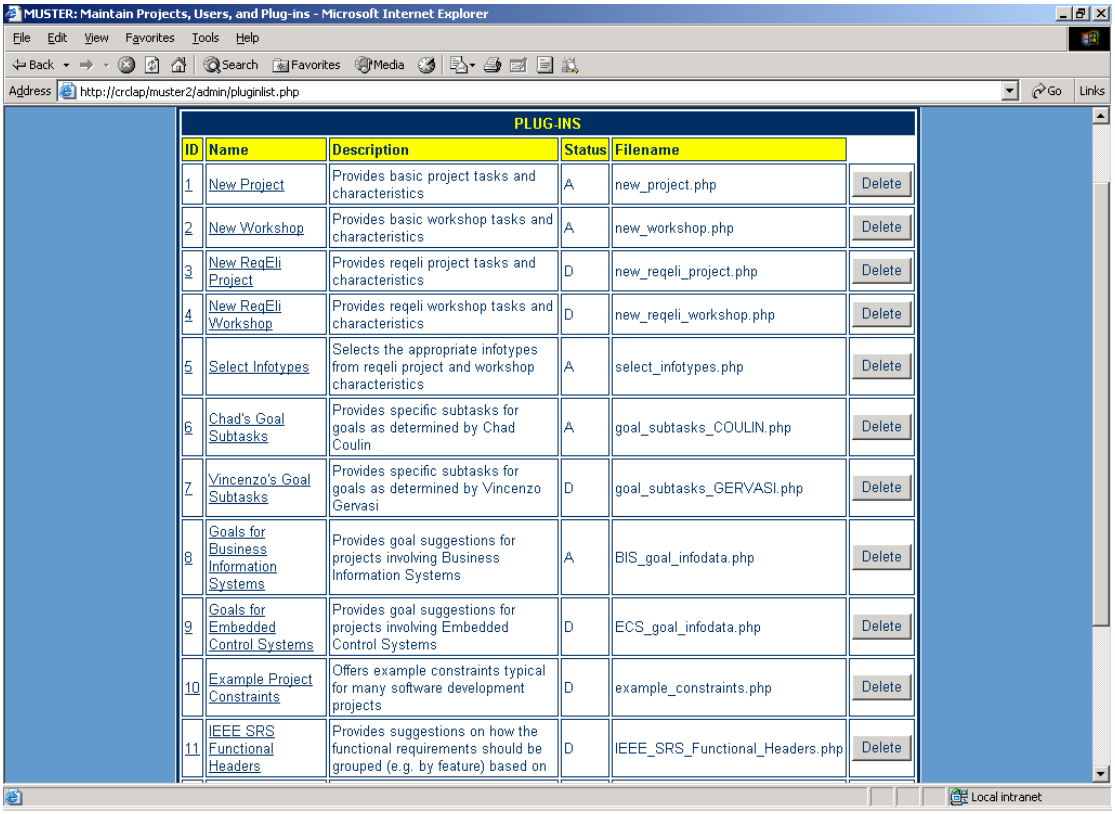

Figure 2. Plug-in maintenance utility of the MUSTER system

The analyst can then immediately log out of the MUSTER system, and log back into the newly created project in order to enter values for the situational project characteristics, required by the select plug-ins, as the first task of the new project (see Figure 3 below). The 'Define Characteristics' task has been added to the situational Advice Panel, and subsequently the dynamic Task List, by the 'New Project' plug-in. The characteristics for each of the enabled plug-ins have been added, and in some cases loaded with default values, automatically the first time they are run, which in this case was triggered by the event which added the 'Define Characteristics' task to the dynamic Task List.

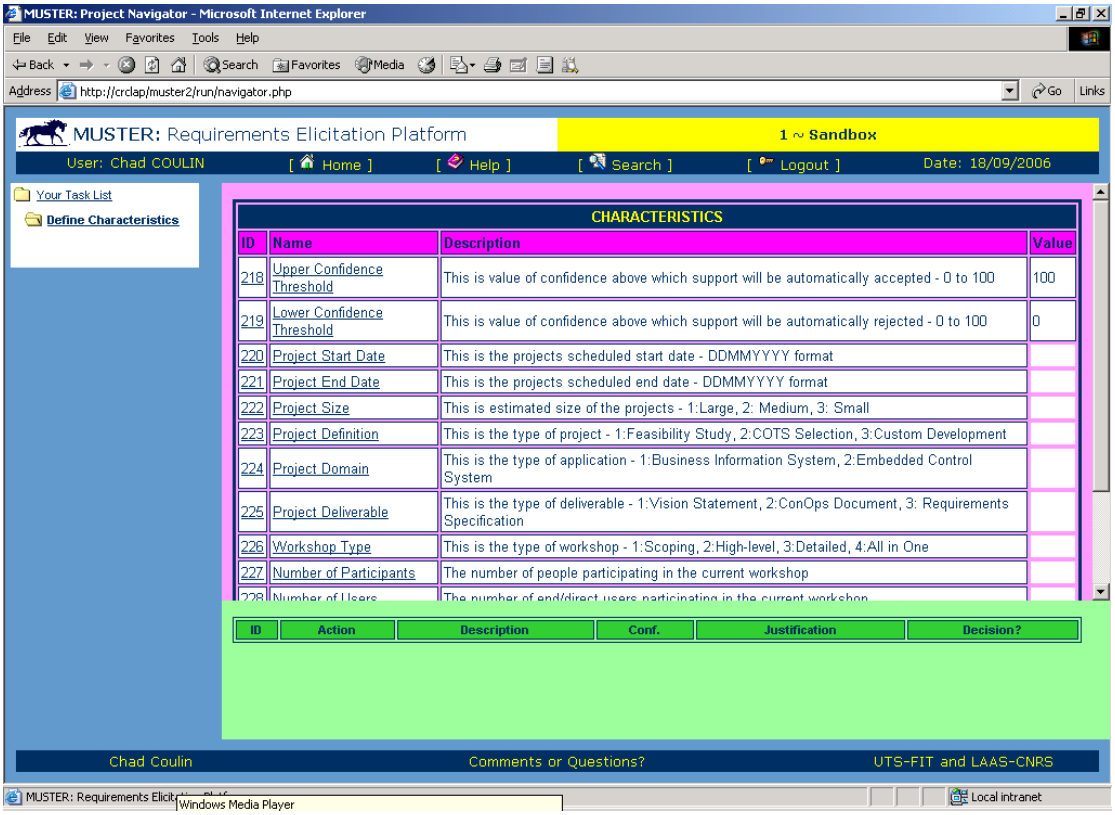

Figure 3. Situational project characteristics maintenance screen

For this example, the current project is small in size, and involves the production of a requirements document for the custom development of a business information system. Furthermore, the analyst's intention is to elicit as much of the information as possible in a simple combined workshop with multiple sessions.

Performance

As values for the characteristics are entered, tasks are added to the list, and data is maintained in the repository, the plug-ins are repeatedly triggered, and are therefore able to start and continue providing situational support to the users via the advice panel. From Figure 4 below, it can be seen that after only some of the characteristics values have been entered, the system has already provided several items of advice via the plug-ins, including the addition of core tasks such as the elicitation of system goals and the elicitation of project constraints.

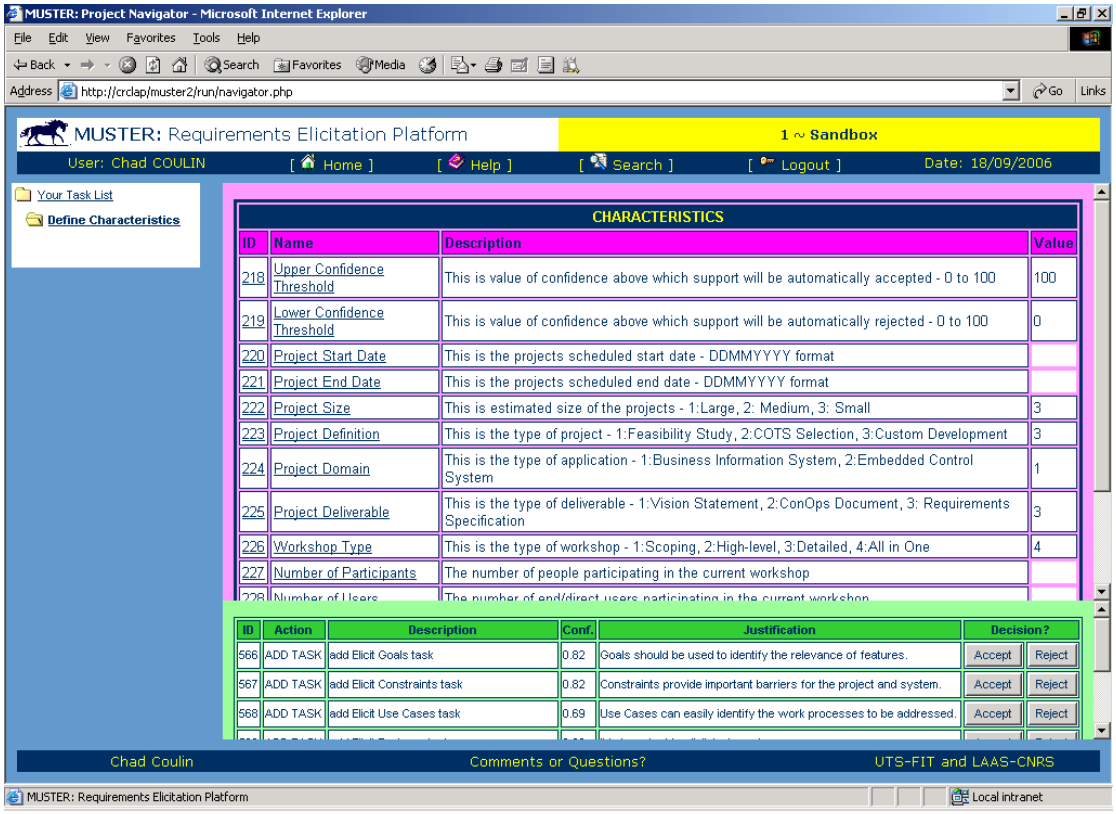

Figure 4. System screen shot after some characteristic values entered

Both the analyst and the stakeholders participating in the requirements elicitation workshop make their way through the task list, adding data to the repository for the recommended and accepted data types, through task specific instructional and maintenance screens. Task specific advice from the plug-ins, including the addition of subtasks, may in some cases only be offered if the dependent first-level task has been added, and once that particular first-level task has been started.

In a screen shot taken towards the end of the workshop (Figure 5), it can be seen that several more tasks and sub-tasks have been added to the list, such as “Brainstorm Goals” and “Elicit Constraints”. Furthermore, the only item remaining in the advice panel relates to the presentation of the data in the repository, by way of quality check and export, in the format of the required deliverable type. In this case, the ‘Presentation’ task has been offered by a plug-in and added to the advice panel only after each of the data types has had at least one entry recorded for it in the MUSTER system.

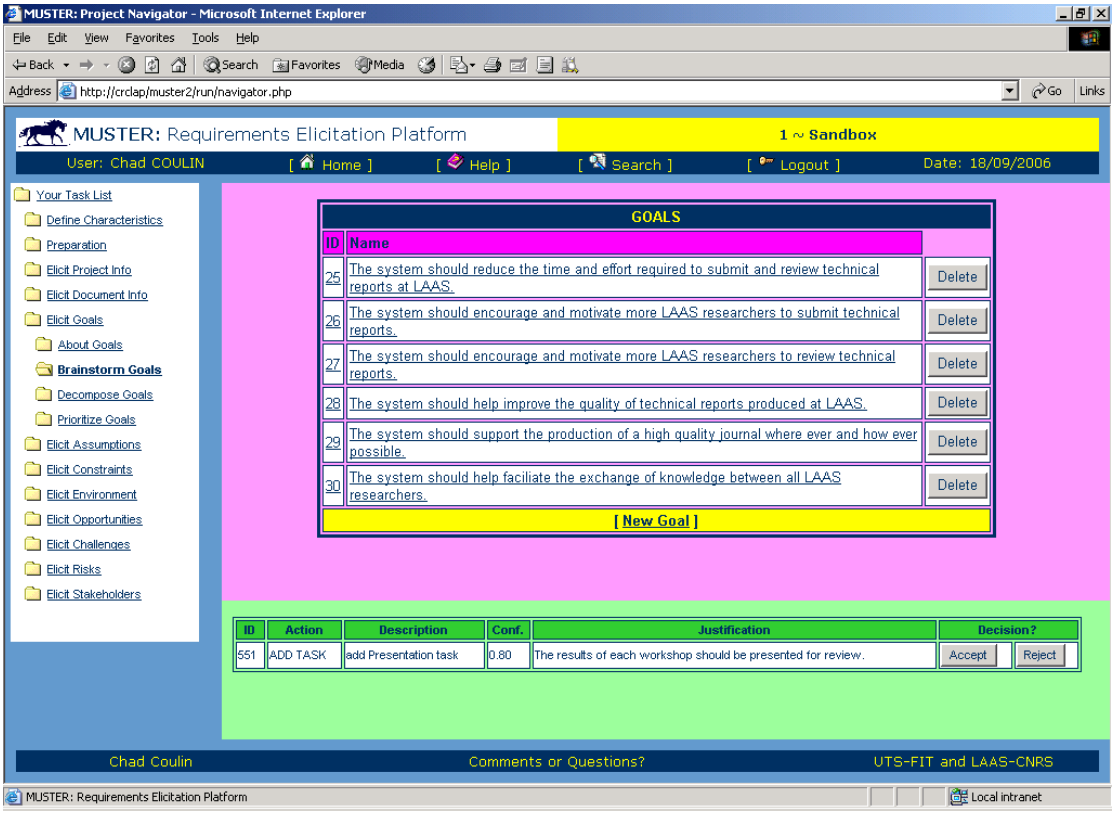

Figure 5. Screen shot of the MUSTER system near the end of the workshop

The workshop may come to a conclusion either 1) when the participants have run out of allocated and available time, or 2) when the participants can no longer think of additional relevant data to enter into the MUSTER system, and the plug-ins can no longer offer meaningful advice to them. At this point the project is ready for presentation as described below.

Presentation

Before exporting the results of the workshop out of the MUSTER system, the elicited information can be reviewed collaboratively, or individually, by the participants. In order for the necessary walkthroughs and inspections to take place, the data from the repository is formatted and exported using one of the available standard reports (see Figure 6 below).

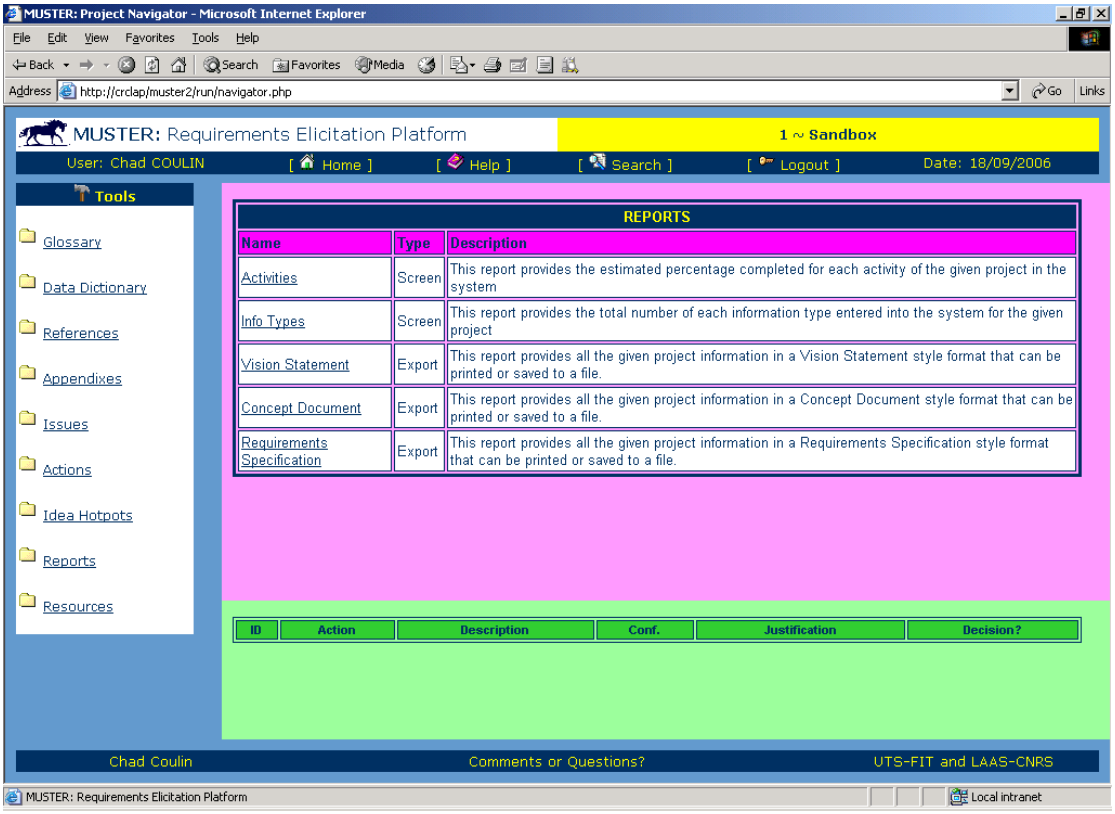

Figure 6. List of available standard reports in the MUSTER system

The resultant deliverable of this process, as produced by MUSTER, is then ready for feedback and approval by the appropriate workshop participants and other stakeholders.

DISCUSSION

A number of additional features were considered but not implemented for MUSTER, including Instant Messaging, Discussion Boards, Reuse Utility, and a Workflow Engine. The reasons for their exclusion were for the most part related to the potential benefit they would provide, compared to their relevance to the overall goals of the research project. Also considered useful, but somewhat out of scope with respect to the focus and objectives of the tool, were the use of visual effects such as zooming user interfaces, mouse over magnification, and wheel or web data representations. In addition, several features such as the Online Help, Categorizor, Technique Selector, and Ask REG, were only partially developed despite their novelty, because of the effort required to implement and evaluate them to a level that would provide substantial enhancement.

The use of a simple and open plug-in architecture has given the tool a number of important advantages, including the ability to utilize different technologies, and offer different types of support to the users. However because these plug-ins can originate from many sources, and may be based on subjective and imprecise reference material (e.g. experience), the potential result of all the advice offered by the combined plug-ins is difficult to predict. Consequently the effectiveness and usefulness of the MUSTER system is heavily dependent on the quality of the advice offered by the plug-ins, and the

way in which expert knowledge is presented within the workshop environment. As a result, it is not possible to claim that the tool is as good or better than having a participating requirements elicitation expert facilitate the workshops, but rather that the support offered will be of benefit to novice analysts in guiding the process of requirements elicitation

Furthermore, in addition to providing an implementation of the situational OUTSET approach, MUSTER addresses some of the issues often experienced in requirements elicitation practice. For example, the tool allows end users of the system to communicate openly, access project information, and be actively involved in both the elicitation process and the development of the target system requirements, thereby encouraging project ownership and stakeholder commitment. MUSTER also overcomes a major limitation of many groupware applications (Alho & Sulonen, 1998), in that it not only supports, but also actively encourages, the use of a dynamically generated process based on contextual factors. In order to achieve this, the MUSTER tool has endeavored to be as flexible and configurable as possible, whilst still providing an appropriately structured and rigorous foundation for requirements elicitation, creative thinking, idea crystallization, and constructivist learning.

CONCLUSION

In a retrospective report on lessons learnt from ten years of Group Support Systems (GSS) research (Nunamaker, Briggs & Mittleman, 1996), it was determined that a GSS can significantly reduce time and effort, but does not replace leadership. Likewise, the MUSTER system is intended to provide practical benefits and support for the analyst and stakeholders during requirements elicitation, rather than replace the role of a workshop facilitator completely. In the same report, it was stated that Group Support Systems should include separate special purpose modules to permit flexible process design, which MUSTER has also endeavored to satisfy through the use of an overriding plug-in architecture for the usage of the tool and generation of intelligent and situational guidance.

In terms of being a virtual workbench for requirements elicitation, and as a special purpose CASE/CAME application, MUSTER provides a number of potential benefits over existing requirements elicitation tools. Through direct interaction with the analyst and stakeholders during the workshops, MUSTER removes the need for costly and time-consuming meeting transcription and report writing, whilst still being collaborative and combinational. The utilization of web-based technologies, and the integration of intelligent technologies, enables contextual support to be provided through an integrated situational approach and environment. Furthermore, the use of only Open Source technologies, and a plug-in architecture as a mechanism to store and transfer expert requirements elicitation knowledge was not only new and innovative, but also allowed the implementation of intelligence into the tool for process guidance and cognitive support.

Three empirical evaluations of the tool have been conducted (Coulin, 2007), showing that MUSTER improved both the overall effectiveness and efficiency of the requirements elicitation process, and provided a system that was both useful and useable to the participating analysts.

REFERENCES

Alderson, A. (1991). Meta-case Technology. *European Symposium on Software Development Environments and CASE Technology*. Konigswinter, Germany, June 17-19.

A.L.I.C.E. AI Foundation Inc. (2005). A.L.I.C.E., Retrieved 2005, from http://www.alicebot.org/

Alho, K. & Sulonen, R. (1998). Supporting Virtual Software Projects on the Web. *Seventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98)*. Stanford, USA, June 17-19.

Atlantic Systems Guild (2003). Volere Requirements Specification Template.

Coulin, C. (2007). A Situational Approach and Intelligent Tool for Collaborative Requirements Elicitation. PhD Thesis. University of Technology Sydney & Paul Sabatier University.

Coulin, C., Zowghi, D. & Sahraoui, A.E.K. (2006). A Situational Method Engineering Approach to Requirements Elicitation Workshops in the Software Development Process. *Software Process: Improvement and Practice*, vol. 11, pp. 451-64.

Dahanayake, A.N.W. (1998). Evaluation of the Strength of Computer Aided Method Engineering for Product Development Process Modeling. *9th International Conference on Database and Expert Systems Applications*. Vienna, Austria, August 24-28.

Dardenne, A., van Lamsweerde, A. & Fickas, S. (1993). Goal-Directed Requirements Acquisition. *Science of Computer Programming*, vol. 20, no. 1-2, pp. 3-50.

den Hengst, M., van de Kar, E. & Appelman, J. (2004). Designing Mobile Information Services: User Requirements Elicitation with GSS Design and Application of a Repeatable Process. *37th Hawaii International Conference on System Sciences*. Big Island, Hawaii, January 5-8.

Ellis, C.A., Gibbs, S.J. & Rein, G.L. (1991). Groupware: Some issues and experiences. *Communications of the ACM*, vol. 34, no. 1, pp. 38-58.

Gottesdiener, E. (2002). *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley, Boston, USA.

Hannola, L., Elfvengren, K. & Tuominen, M. (2005). Improving Requirements Elicitation with GSS in Software Development. *Annual ISPIM Conference, International Society for Professional Innovation Management*. Porto, Portugal, June 19-22.

Hickey, A.M. (2003). Requirements Elicitation Techniques: Analyzing the Gap between Technology Availability and Technology Use. *Comparative Technology Transfer and Society*, vol. 1, no. 3, pp. 279-302.

Hickey, A.M. & Davis, A.M. (2002). The Role of Requirements Elicitation Techniques in Achieving Software Quality. *Eighth International Workshop of Requirements Engineering: Foundation for Software Quality*. Essen, Germany, September 9-10.

Hickey, A.M. & Davis, A.M. (2003). Elicitation Technique Selection: How Do Experts Do It?. *Eleventh IEEE International Requirements Engineering Conference*. Monterey Bay, USA, September 8-12.

Hickey, A.M., Dean, D.L. & Nunamaker, J.F. (1999). Establishing a Foundation for Collaborative Scenario Elicitation. *The DATA BASE for Advances in Information Systems*, vol. 30, no. 3-4, pp. 92-110.

Herela, D. & Greenberg, S. (1998). Using a Groupware Space for Distributed Requirements Engineering. *Seventh Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Stanford, USA, June 17-19.

Hoffer, J.A., George, J.F. & Valacich, J.S. (2002). *Modern Systems Analysis and Design*, Third edn, Prentice Hall, USA.

IEEE (1998a). *IEEE Std 830-1998*. Recommended Practice for Software Requirements Specifications.

IEEE (1998b). *IEEE Std 1362*. System Definition - Concept of Operations (ConOps) Document.

Jarzabek, S. & Huang, R. (1998). The Case for User-Centered CASE Tools. *Communications of the ACM*, vol. 41, no. 8, pp. 93-9.

Liou, Y.I. & Chen, M. (1993). Integrating Group Support Systems, Joint Application Development, and Computer-Aided Software Engineering for Requirements Specification. *26th Annual Hawaii International Conference on System Sciences*. Wailea, Hawaii, January 5-8.

Macaulay, L. (1996). Requirements for Requirements Engineering Techniques. *International Conference on Requirements Engineering*. Colorado Springs, USA, April 15-18.

Maiden, N.A.M. & Sutcliffe, A.G. (1993). Requirements Engineering by Example: an Empirical Study. *IEEE International Symposium on Requirements Engineering*. San Diego, USA, January 4-6.

Meagher, P. (2004). Implement Bayesian inference using PHP, Part 1: Build intelligent Web applications through conditional probability. Retrieved 2005, from http://www-106.ibm.com/developerworks/web/library/wa-bayes1/

Nunamaker, J.F., Briggs, R.O. & Mittleman, D.D. (1996). Lessons from a Decade of Group Support Systems Research. *29th Annual Hawaii International Conference on System Sciences*. Maui, Hawaii, January 3-6.

Pohl, K., Assenova, P., Doemges, R., Johannesson, P., Maiden, N., Plihon, V., Schmitt, J.-R. & Spanoudakis, G. (1994). Applying AI Techniques to Requirements Engineering: The NATURE Prototype. *International Conference on Software Engineering*. Edinburgh, UK, May 23-28.

Robertson, S. & Robertson, J. (1999). *Mastering the Requirements Process*. Addison-Wesley, Great Britain.

Rolston, D. (2005). LAMP, MySQL/PHP Database Driven Websites - Parts I, II, and III, Retrieved 2005, from http://www.phpfreaks.com/tutorials/

Saeki, M. (2003). CAME: The First Step to Automated Method Engineering. *Workshop on Process Engineering for Object-Oriented and Component-Based Development*. Anaheim, USA, October 26-30.

Saeki, M., Tsuchida, M. & Nishiue, K. (2000). Supporting tool for assembling software specification and design methods. *24th Annual International Computer Software and Applications Conference*. Taipei, Taiwan, October 25-27.

Scott, W. & Cook, S.C. (2003). An Architecture for an Intelligent Requirements Elicitation and Assessment Assistant. *13th Annual International Symposium - INCOSE 2003*. Crystal City, USA, July 1-3.

Sommerville, I. (2001). *Software Engineering*. 6th edn, Addison-Wesley, USA.

Sommerville, I. & Sawyer, P. (1997). *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Great Britain.

Tuunanen, T. (2003). A New Perspective on Requirements Elicitation Methods. *Journal of Information Technology Theory and Application*, vol. 5, no. 3, pp. 45-62.

Venable, J.R. & Travis, J. (1999). Using a Group Support System for the Distributed Application of Soft Systems Methodology. 10th Australasian Conference on Information Systems. Wellington, New Zealand, December 1-3.

Wang, X. & Loucopoulos, P. (1995). The Development of Phedias: a CASE Shell. *Seventh International Workshop on Computer-Aided Software Engineering*. Toronto, Canada, July 10-14.

Wiegers, K.E. (2003). *Software Requirements*. Second edn, Microsoft Press, USA.

Wiegers, K.E. (2007). Process Impact. Retrieved 2007, from http://www.processimpact.com/

Wood, D.P., Christel, M.G. & Stevens, S.M. (1994). A Multimedia Approach to Requirements Capture and Modeling. *First International Conference on Requirements Engineering*. Colorado Springs, USA, April 18-22.

Young, E. (2004). Artificial Neural Network in PHP. Retrieved 2005, from http://coding.mu/archives/2004/03/19/artificial_neural_network_in_php/

Yu, E.S.K. (1997). Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. *Third IEEE International Symposium on Requirements Engineering*. Washington D.C., USA, January 5-8.

Zeroual, K. (1991). A Knowledge-based Requirements Acquisition System. *6th Annual Knowledge-Based Software Engineering Conference*. Syracuse, USA, September 22-25.