Chapter 3.8
MUSTER:
A Situational Tool for Requirements Elicitation

Chad Coulin
University of Technology Sydney, Australia & LAAS CNRS, France

Didar Zowghi
University of Technology Sydney, Australia

Abd-El-Kader Sahraoui
LAAS CNRS, France

ABSTRACT

In this chapter they 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.

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

DOI: 10.4018/978-1-61350-456-7.ch3.8

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