Chapter 5
Methodologies for Qualitative Spatial and Temporal Reasoning Application Design

Carl Schultz
The University of Auckland, New Zealand

Robert Amor
The University of Auckland, New Zealand

Hans W. Guesgen
Massey University, New Zealand

ABSTRACT
Although a wide range of sophisticated Qualitative Spatial and Temporal Reasoning (QSTR) formalisms have now been developed, there are relatively few applications that apply these commonsense methods. To address this problem, the authors of this chapter developed methodologies that support QSTR application design. They established a theoretical foundation for QSTR applications that includes the roles of application designers and users. The authors adapted formal software requirements that allow a designer to specify the customer’s operational requirements and the functional requirements of a QSTR application. The chapter presents design patterns for organising the components of QSTR applications, and a methodology for defining high-level neighbourhoods that are derived from the system structure. Finally, the authors develop a methodology for QSTR application validation by defining a complexity metric called H-complexity that is used in test coverage analysis for assessing the quality of unit and integration test sets.

DOI: 10.4018/978-1-61692-868-1.ch005
1. INTRODUCTION

Over the last two and a half decades researchers have made significant progress in the theoretical foundations and analysis of Qualitative Spatial and Temporal Reasoning (QSTR) calculi, and a range of commonsense formalisms have now been developed for representing and reasoning about different aspects of space and time (Cohn & Renz, 2008). Moreover, while many QSTR formalisms have been shown to be NP hard, maximal tractable subsets of well known calculi have been identified (Nebel & Bürrckert, 1995; Renz, 1999) and automatic methods for finding tractable subsets have been developed (Renz, 2007), thus informing a user about the classes of problems that are practical to solve. Techniques have also been developed that greatly improve reasoning performance (Westphal & Wölf, 2009; Li, et al., 2009).

Despite this theoretically advanced state of the field, there is a distinct absence of applications that make significant use of QSTR formalisms. There are five critical barriers to QSTR application design that have not yet been addressed.

1. QSTR researchers have not clearly identified the characteristics of the problems that can be uniquely addressed by QSTR applications.
2. In many cases, no pre-existing QSTR formalism will perfectly and completely satisfy the requirements of an application. In most cases, the designer will need to formalise domain knowledge, and design complex, heterogeneous models that build on top of a mix of different existing QSTR formalisms.
3. There is no methodology for developing QSTR applications, and even researchers in the field currently develop QSTR applications in a very ad hoc manner.
4. There are no methodologies for analysing QSTR applications, and therefore no way to make informed design decisions. This contributes to the problem of ad hoc QSTR application development.
5. Making QSTR accessible means having designers from outside the field applying QSTR, that is, the designers will not be experts in QSTR. Design methodologies derived from concepts in software engineering are required to bridge the gap between expert QSTR logicians and application designers from other disciplines.

We address these issues in this chapter with specialised methodologies for QSTR application design, motivated by research in software engineering, knowledge representation, artificial intelligence, and finite model theory. Section 2 reviews unifying frameworks and development tools in related areas and establishes a theoretical foundation for QSTR applications. Section 3 identifies the salient characteristics of QSTR applications that are the focus of our design methodologies. Section 4 characterises the problems that QSTR applications address and enumerates the tasks that they can perform, by adapting formal software requirements from software engineering. Section 5 presents design patterns for organising the components of QSTR applications, and Section 6 presents a methodology for defining high level neighbourhoods that are derived from the system structure. Section 7 presents a methodology that supports designers in QSTR application validation by identifying important classes for unit testing and integration testing based on a novel measure of complexity. Sections 8 and 9 present the future work and conclusions of this chapter.

2. BACKGROUND

A number of unifying QSTR frameworks are now being developed in order to make the field more cohesive and accessible. Three prominent projects are SparQ (Dylla, 2006), GQR (Gantner, 2008), and an investigation into formal algebraic