Canadian Software Requirements Symposium '03

May 26, 2003

The CSRS event is dedicated to all topics related to software requirements. The topics are to be interpreted broadly and inclusively, and in particular cover elicitation, modelling, validation, verification, documentation and requirements based testing.

The aim of this event is to foster a research community in areas related to software requirements in Canada through encouraging communication among researchers. Specific objectives include efforts at integration as well as the transfer of methods between different groups.

Some of the participants are making short presentations (20 minutes). For the academic presentations the aim is to answer the following questions:

1. What research have you done or are you now undertaking to make requirements analysis methods and requirements processes more useful to software developers?
2. How can mathematical methods contribute to tackling the requirements problems you are studying?
3. What research should we be undertaking in the area of software requirements?

For the industrial participants the presentations focus on the following questions:

1. What are the techniques you are using to deal with requirements in your practise and why do you think that they are appropriate for your needs?
2. What are the problems that you are facing with requirements in your practise?
3. Which problem do you think research should focus on in the near future?

Naturally, some people will spend more time on one of these questions than on the others. The presentations will be made by both academic and industrial researchers, with the individuals chosen so as to cover as many different schools of thought as possible.

The intention of the organizers is that this symposium will foster a spirit of collaboration and communication within the software requirements community in Canada. The hope is that the good-will generated at this year's even will encourage a similar event to be organized at another Canadian university next year.

Ridha Khedri
Spencer Smith

Keynote Talk

Goal-Oriented Requirements Engineering: the Way to High Assurance Systems

Axel van Lamsweerde
Département d'Ingénierie Informatique
University of Louvain

High assurance systems must guarantee safety, security, fault tolerance and survivability objectives. It is therefore essential that such objectives be made explicit, refined, interrelated, specified precisely and completely, and analyzed thoroughly.

The talk will argue that goals are an essential abstraction for eliciting, elaborating, modelling, specifying, analyzing, verifying, negotiating and documenting robust and conflict-free requirements for high assurance systems.

Two concrete examples will be used to illustrate the key role of goals while engineering such requirements: a safety-critical system for a nuclear power plant and a security-critical purse system to be run on Palm handhelds.

If time allows, a goal model checker and animator will be demonstrated to suggest the kind of early analysis that can be performed incrementally on partial requirements models.

Contributed Talks

Standardizing Requirements Notation: URN, and What Else?

Daniel Amyot, School of Information Technology and Engineering (SITE), University of Ottawa

Standardizing notations and languages has a positive impact on their acceptance by industries and tool vendors. Yet, very few requirements notation have led to international standards. This talk will briefly introduce the ITU-T User Requirements Notation (URN) and the context in which it is being developed (http://www.UseCaseMaps.org/urn/). The current proposal combines two complementary notations. The Goal-oriented Requirements Language (GRL), which is based on the NFR framework and on i*, enables the description and analysis of business goals and (non-)functional requirements. The Use Case Map (UCM) notation targets operational and functional requirements with causal scenarios whose elements can be allocated to abstract components. UCMs can serve as a basis for a number of analysis tasks, including performance analysis, conflict detection, architectural evaluations, and test generation. Additional standardization efforts within ITU-T, ISO/IEC, and OMG will also be discussed.

Computer-Assisted Assume/Guarantee Reasoning with VeriSoft

Juergen Dingel, School of Computing, Queen's University

We show how the state space exploration tool VeriSoft can be used to analyze parallel C/C++ programs compositionally. VeriSoft is used to check assume/guarantee specifications of parallel processes automatically. The analysis is meant to complement standard assume/guarantee reasoning which is usually carried out solely with "pencil and paper". While a successful analysis does not always imply the general correctness of the specification, it increases the confidence in the verification effort. An unsuccessful analysis always produces a counterexample which can be used to correct the specification or the program. VeriSoft's optimization and visualization techniques make the analysis relatively efficient and effective.

Non-Functional Requirements: From Elicitation to Conceptual Models

Luiz Marcio Cysneiros, Department of Mathematics and Statistics, York University

Non-Functional Requirements (NFRs) have been frequently neglected or forgotten in software design. They have been presented as a second or even third class type of requirement, frequently hidden inside notes. We tackle this problem by treating NFRs as first class requirements. We have developed process to elicit NFRs, analyze their interdependencies, and trace them to functional conceptual models. We focus our attention on conceptual models expressed using UML (Unified Modelling Language). Extensions to UML are proposed

to allow NFRs to be expressed. We integrate NFRs into Class, Sequence and Collaboration Diagrams. We also show how Use Cases and Scenarios can be adapted to deal with NFRs. This work was used in three case studies and their results suggest that by using our proposal we can improve the quality of the resulting conceptual models.

Reasoning with Uncertainty and Inconsistency

Steve Easterbrook, Department of Computer Science, University of Toronto

Eliciting the requirements for a proposed system typically involves different stakeholders with different expertise, responsibilities, and perspectives. This may result in inconsistencies between descriptions provided by the stakeholders. For the past few years, we have been investigating automated techniques that support reasoning over a set of inconsistent and incomplete viewpoints. Much of the work has centred on the use of multi-valued logics, and we have built a model checker, xchek, which generalises classical symbolic model checking to the multi-valued case. We are now exploring applications of these ideas, for supporting negotiation during RE, for composing partial and inconsistent views, for reasoning about relative priority of requirements, and for model exploration through query checking. This talk will give an overview of our work, covering both the theory on which we draw, and some example applications.

Formal Methods, the Very Idea, Some Thoughts

Daniel M. Berry, School of Computer Science, University of Waterloo

The talk defines formal methods (FMs) and describes economic issues involved in their application. From these considerations and the concepts implicit in "No Silver Bullet", it becomes clear that FMs are best applied during requirements engineering. A theory of why formal methods work when they work is offered and it is suggested that FMs help the most when the applier is most ignorant about the problem domain.

The Abstract State Machine Paradigm for Engineering Concurrent Systems: Turning Abstract Requirements into High Level Executable Specifications

Uwe Glässer, School of Computing Science, Simon Fraser University

Based on an abstract operational view of functional and timing requirements for control intensive software, we model dynamic system properties in terms of abstract machine models. Focusing on key system attributes, we propose a gradual formalization of system behaviour with a degree of detail and precision as needed for the elicitation and clarification of requirements. Beyond clear and concise documentations, the resulting models are meant to be executable specifications that support experimental validation of high level requirements in early system design stages.

Typical application examples include communications software and distributed embedded systems in automotive control, industrial automation, and advanced telecommunication

services. Such systems are characterized by their concurrent and reactive behaviour making it difficult to predict dynamic system properties with a sufficient degree of detail and precision under all circumstances. Inevitably, experimental validation of high level requirements facilitates the exploration of undesirable behaviour and hidden side effects, thereby making systems design more reliable.

Promising results from a variety of pilot projects in academia and industry show the practical relevance of the abstract state machine approach, nevertheless there still are many open issues and unexploited potential for future research and development. This talk will outline some lessons learned from recent industrial applications.

Requirements Engineering at MOTOROLA CANADA

Malcolm Goddard, Toronto Design Centre, Motorola Canada Limited

We will describe how we do requirements engineering within Motorola Canada and where we believe there are opportunities for involvement from the universities.

Using Simple Visualization Tools to Improve Early Engineering Design

Filippo Salustri, Mechanical Engineering Department, Ryerson University

One way of dealing with both the importance of early product development and its increased complexity is to find strategies to simplify the methods and tools used by product development teams. The author is developing an integrated strategy to do this through the use of diagramming tools.

The author's approach, called Design Schematics, draws heavily from diagramming methods in computer science and programming, but is also influenced by other sources. Fundamentally, Design Schematics are hypergraphs augmented with semantics to describe the intention of individual nodes and arcs. Some of these include UML, flowcharts, Chapin charts, software architecture diagrams, and constraint graphs. Other sources include IDEF, bond graphs, Modellica, CPM/PERT/Gantt charts, electrical block diagrams, and concept maps. The computer science sources have been the most important so far.

Concept maps form the foundation of Design Schematics, because they are the most flexible and unstructured. All other methods investigated so far impose too much structure and so limit the applicability of the tool in early design settings. The problem with concept maps, however, is that they are too unstructured.

Thus, my general approach is to add features one at a time, drawn from other tools, to structure Design Schematics so that they remain sufficiently flexible and pertinent to design engineering conventions.

Software Requirements Research in SQRL

Alan Wassyng, Spencer Smith, Ridha Khedri and Konstantin Kreyman, Department of Computing and Software, McMaster University

We present an overview of current research efforts underway in SQRL, in the field of Mathematical Software Requirements. We also indicate, briefly, future directions of research in this field that we intend to pursue.

Metro: A Semantics-based Approach for Mapping Specification Notations to Formal Analysis Tools

Jianwei Niu, Joanne Atlee, and Nancy Day, School of Computing Science, University of Waterloo

We propose a semantics-based approach, Metro, to generate analyzers automatically from the description of notations' semantics. In Metro, the semantics of a notation are defined and fed into a model compiler together with a specification in that notation. A transition-relation for the specification is generated automatically, which can be checked using many automatic analysis tools (e.g., model checkers). However, the semantics of many model-based notations (e.g., statecharts and process algebras) are complex and formalizing their semantics can be challenging and error-prone. To ease the effort required to define the semantics, we have developed an operational semantics template that captures the common behaviour of different notations and parameterizes notations' distinct behaviours. We have also identified seven well-used composition operators and defined the semantics of these operators separately, as relations that constrain how components can execute concurrently, and communicate and synchronize with each other by exchanging events and data. By separating a notation's step-semantics from its composition and concurrency operators, we simplify the definitions of both. We have used our template to capture the essential semantics of basic transition systems, CSP, CCS, basic LOTOS, a subset of SDL88, and a variety of statecharts notations.

Back to the Drawing Board: Visualizing Requirements Through Graphical Models

Christopher Sibbald, Telelogic North America Inc.

Writing requirements in a formal, textual way ensure they are well specified and this often forms the basis for a contract between the customer and the supplier. However, there are times when being able to visualize those requirements might help both customer and supplier gain a much quicker understanding of what is intended.

This is made possible through DOORS integration to Telelogic Tau/ Architect, which enables full two-way traceability between textual requirements and graphical models of those requirements expressed in the Unified Modelling Language (UML).

This presentation will feature a live demonstration of the functionality-rich integration between these two innovative tools and show you how to quickly establish traceability be-

tween text and model. You'll learn how Tau/Architect's consistency checking and simulation capabilities can help improve the quality of requirements benefit both systems and software engineering projects.

Detecting Feature Interactions in Web Services

Michael Weiss, Carleton University

Much of the work on web services to date has focused on low-level standards for publishing, discovering, and invoking services. More recent work has looked at how higher-level services can be composed dynamically from lower-level services. However, to date, there has been little research on the problems that can arise from service integration, and how to manage them, a problem known as feature interaction in the telephony domain. In this presentation we will describe our initial results towards an approach for identifying an important type of (undesirable) feature interactions between web services, which are due to the violation of non-functional requirements. Our approach uses the Non-Functional Requirements (NFR) framework.