EDITORIAL

A crucial first step in software development is the generation of a description of the artefacts that we seek to build. Among other things, this description must include the requirements that the system must satisfy. It is widely recognised now that these descriptions might be the critical determinants of software quality, given empirical studies showing that errors in generating these descriptions are the most numerous in the software life-cycle and the most expensive and time-consuming to correct. The field of *requirements engineering* considers the problem of engineering such descriptions. Requirements engineering is a relatively mature field of inquiry, but the growing complexity of present-day systems (and their organisational contexts), has focused considerable attention on the area, both in academia and in industry. This special issue is a carefully compiled collection of state-of-the-art papers that seeks to provide readers with a broad perspective on new and interesting developments in the area.

This issue consists of two invited contributions and seven refereed submissions. The two invited contributions are representative of two ends of the research spectrum in requirements engineering: object-oriented approaches and formal methods. The invited position paper by Froehlich, Hoover and Sorenson makes the case for the use of *hooks* in the context of architectures that apply to suites of related products or to entire application domains. It shows how the use of hooks to encapsulate requirements can help localise the effects of requirement changes and focus attention on elements of these generic architectures that require modification for the specific product being developed. The invited paper by Tsai and Li provides an approach to the verification and validation of formal (and executable) requirements specifications via parallel execution. The underlying premise is that executing and reasoning with formal specifications enables us to resolve conflicts amongst the various system stakeholders and verify that certain domain-independent properties (such as liveness or consistency) hold, but is typically too slow to be of practical use. The authors suggest a novel parallel execution model for formal specifications that combines elements of both top-down and bottom-up evaluation strategies to address this problem.

The seven refereed submissions were selected after careful vetting (by at least two reviewers) from a total of twenty submissions and represent the diversity of research themes within the area. Two of these papers present comprehensive requirements engineering methodologies and supporting tools. Barber, Graser, Grisham and Jernigan present the Software Engineering Process Activities (SEPA) methodology that supports the integration new implementations with legacy systems and COTS products in the context of component-based development. Like Froehlich, Hoover and Sorenson, they argue for the separation of requirements that apply across domains or product-lines from those that are application-specific. They describe a set of tools that implement this methodology with additional support for traceability, evolution and re-use. Requirements refinement and re-use is also the focus of the Reuse-Assisted Requirements Engineering (RARE) methodology presented by Cybulski and Reed. Their approach involves the use of faceted classification schemes for classifying requirements and design artefacts, similarity-based retrieval to support artefact re-use and the use of thesauri for mapping problem domain terms (in which requirements are expressed) to solution domain terms (in which design artefacts are described) to support the process of refining requirements specifications to designs.

The difficult problem of requirements evolution is the focus of two separate papers in this issue. A key question in requirements evolution is deciding whether to proceed with a change, given that most changes involve significant cost. Impact analysis provides a means for determining the extent, complexity and cost of proposed changes to a system in order to inform the decision on whether to accept or reject a proposed change. Lock and Kotonya propose a set of requirements-

level techniques to identify how changes to certain artefacts may impact others and how the consequences of these changes propagate through the entire specification. Nuseibeh and Russo address questions that arise after a decision to accept a proposed change has been taken. They propose an ambitious scheme that uses techniques derived from abductive logic programming to identify means for resolving inconsistencies generated by changes to requirements specifications. The formal language in which they propose to represent specifications is in itself quite novel - quasi-classical (QC) logic is a variant of classical logic that permits reasoning with portions of formal specifications that are not directly affected by inconsistencies, in the event that such inconsistencies appear in specifications (typically as a consequence of requirements change).

Formal methods, such as those discussed in the papers by Tsai and Li, and Nuseibeh and Russo, are often criticised for being too inaccessible to practitioners and expensive to implement. Droschl presents a case study that suggests otherwise. His case study involves the application of a semi-automatic theorem prover (PVS) to the analysis of requirements for an access control system, and contains important lessons for future similar efforts.

Extra-functional (or non-functional) requirements are often ignored in requirements engineering frameworks. The paper by Hochmuller makes a strong case for explicitly accounting for extra-functional requirements in requirements process models. Hughes and Wood-Harper make a case for providing more tangible and explicit support for another facet of the requirements engineering process that is typically ignored - the organisational perspective. They support their argument through two case studies that suggest that an explicit sociological/organisational perspective can indeed better inform the exercise of requirements elicitation and analysis.

The problems confronting the requirements engineering community are challenging ones, requiring a complex toolkit of approaches. The nine papers contained in this special issue represent the proverbial tip of the iceberg. This issue, hopefully, provides pointers to the interesting research terrain that remains to be explored.

I would like to specially thank Jacob Cybulski, Frank Moisiadis, Yusuf Pisan, Debbie Richards, Alessandra Russo and Didar Zowghi for their help with reviewing submissions to the special issue.

Aditya Ghose
Department of Information Systems
University of Wollongong