

Professor Steve
MacDonell, SCMS

Director of SERL
(Software Engineering
Research Laboratory)

smacdone@aut.ac.nz

Steve's Software Engineering Research Wish List

Most of the research I do deals with uncertainty, in one way or another. Recognising it, representing it, reducing it. In relation to software systems this is expressed in the form of questions such as: What can we deliver in the next release? How long will it take to develop these new features? How can we cost-effectively decrease the defect rate in our code? Who are 'the users' anyway? And, who owns that software? The research that I do, and that I like to support students in doing, addresses questions such as these.

In terms of research methods, I am relaxed about what people use as long as it enables them to deliver against the objectives of the work in a rational and robust manner. For instance, if a student is building a tool to address a problem then this work would follow a constructive, design science approach. If the project involves the analysis of data to evaluate a new prediction model then a more experimental approach would be used. If the aim was to assess user satisfaction then interviews would be more appropriate.

A. Managing uncertainty in software engineering

The development and management of software systems is inherently uncertain. Software is an abstract entity, representing items or concepts. Software that is yet to be developed or is incomplete is normally described in an abstract way, often requiring extensive reliance on models. Thus when considering software requirements we are considering an abstract representation of an abstract entity that does not yet exist!!

Sample research topics:

- A1. Portfolio risk management for smaller software projects
- A2. Project manager knowledge codification
- A3. Trade-offs between bidding, pricing and costing
- A4. Manager/developer influence on project planning and execution
- A5. Planning/replanning software projects – ongoing release planning
- A6. Building a fuzzy logic toolset for software project management
- A7. Modelling processes and systems using system dynamics and simulation

B. Empirical software engineering

Collecting and analysing measurement data from software products, processes and resources can help individuals, groups and organizations to set and achieve improvement goals – for instance, reducing defect density in code artefacts, or improving cost estimation accuracy by selection more influential features.

Sample research topics:

- B1. Impact of sampling on empirical modelling outcomes
- B2. Optimising estimation accuracy using multiple methods
- B3. Assessing the accuracy and sensitivity of recorded effort data
- B4. Extent of change in metrics data from project inception to project closure

C. ICT systems success and failure

Public attention naturally falls to the failures in systems, and it is quite right that we should try to learn lessons from those failures. However, many software systems are delivered successfully, in the eyes of at least one group of stakeholders. There are opportunities to learn from these experiences in order to identify processes, events and actions that lead to successful outcomes.

Sample research topics:

- C1. Identifying and classifying patterns in successful projects
- C2. Best practice vs. common practice – avoiding mediocrity, maintaining creativity, facilitating innovation
- C3. Negotiated notions of success among stakeholders
- C4. System usability measurement methods
- C5. Process/methodology assessment and refinement in use

D. The software/system boundary, and the evolution of autonomous systems

There is a growing gap between what we know about software systems and how we develop and manage them. We know, for instance, that:

- 'simple' systems become packages; those we will build will be increasingly complex, and will be deployed into contexts that are also increasingly complex
- this complexity is already well beyond any individual's ability to understand, meaning that we will increasingly need systems to understand themselves
- the bulk of this complexity comes not from within but from 'the edges' – associated with interoperability, interactions, interfaces
- today's software engineering methods, that are founded on the premise of providing internal control, are increasingly inadequate in this context
- Today's approaches for managing complex projects, that are also founded on the premise of providing control, are misdirected and outdated.

The scale and complexity of tomorrow's systems, the growing volumes and diversity of sources of data, the ongoing dissatisfaction with development and management methods, and the limited scope of 'software engineering' to the software rather than the solution or service mean that the future usefulness of this paradigm should be questioned and complementary or alternative paradigms sought.

Sample research topics:

- D1. New metaphors suitable for considering the development, deployment and management of future software systems
- D2. Observing systems as they evolve over time
- D3. Treating software systems as continuous rather than discrete
- D4. Viewing systems as autonomous organic beings
- D5. Applying and evaluating agile management practices