Software Construction by Configuration: Challenges for Software Engineering Research

Ian Sommerville
Lancaster University
Email: is@comp.lancs.ac.uk

Software reuse has been a remarkable success story for software engineering and, over the past few years, this has led to a remarkable paradigm shift in application software development. The vast majority of new business application systems are now developed by reusing and configuring existing systems – ERP systems, vertical application packages, COTS systems, etc. Even when programming languages such as Java are used, much of the application code is concerned with gluing together existing components and the move to service-oriented architectures will accelerate this process.

Construction-by-configuration (C-b-C) involves adopting systems, such as spreadsheets, that have different programming models than conventional application platforms and different processes are used when software is configured rather than programmed. Complete specifications are rare and co-realisation where software is deployed and evolved to fit into a setting of use is a common development approach.

To a large extent, this radical change in software development practice has been ignored by the software engineering research community. By and large, the concerns of that community remain those of the 20th century - methods and techniques for specifying, designing, programming and testing software that is written in a conventional programming language and whose source code is available to the software developers. Software maintenance research has equally focused on conventional programs with little published work on, e.g. the maintenance of ERP systems. The research community is increasingly out of step with industrial practice and, without change, software engineering research will become more and more irrelevant to real-world application software development.

In this talk, I will discuss the problems of construction by configuration and why I believe that this is a challenging area for software engineering researchers. Although widely adopted, it is clearly extremely problematical (e.g. problems with introducing ERP systems are legion) and there is a pressing need for principles and tools that can form a sound scientific basis for this approach to software development.

I believe that the software maintenance research community has a key role to play in this research. Although much maintenance research has been focused on code and code analysis, the community has addressed issues of dealing with software without specifications and where the internal workings are not readily understandable. Some maintenance tools and techniques, such as program visualization, point the way to effective support for construction by configuration.

I will conclude by setting out a research agenda that will allow the research community to play an active part in addressing these 21st century software engineering problems.