Autom Softw Eng (2011) 18: 115–116
DOI 10.1007/s10515-011-0081-4

Editorial: Controlling change

Robert J. Hall

Received: 1 March 2011 / Accepted: 1 March 2011 / Published online: 11 March 2011

Change is a source of complexity in software engineering. Requirements change during or after development; the runtime environment and critical resources can change during execution; even our viewpoint must often change to support different types of reasoning and design activities during development. The risks of inadequately controlling change are well documented by Brooks and others; such risks range from correctness and performance problems in deployed software to outright failure of development projects. This issue's articles share a common high level theme of controlling various types of change. Each addresses controlling change at some stage of the software lifecycle.

– In a service oriented architecture, user level applications are assembled by orchestrating connections and relationships among more primitive networked services. But what happens, or should happen, when a service on which one's application depends changes its behavior, its context, or becomes overloaded or unresponsive? In "A monitoring approach for runtime service discovery", Mahbub, Spanoudakis, and Zisman describe an approach that allows monitoring and timely recognition of such changes, as well as discovery of potential alternatives that satisfy the need of the application.
– In a feature oriented software product line, one configures each specific instance of the line to the requirements of a specific customer by specifying which features are to be included. Features can be compiled in, for maximum efficiency, or bound dynamically at deployment for maximum flexibility. Each of these approaches is useful in different situations. Rosenmüller, Siegmund, Apel, and Saake give us a new way to allow for both of these types of feature assembly, as well as support for

R.J. Hall (✉)
AT&T Labs Research, Florham Park, NJ 07932, USA
e-mail: Bob.ASEJ@gmail.com

ensuring that product line instances obey consistency constraints, in their article, "Flexible feature binding in software product lines".
– In model driven engineering, models abstract a target system in various ways, and it is important to be able to transform a model from one model formalism to another in a way that preserves expected properties. Such transformations must be validated, typically through testing, before engineers can rely on their outputs. In "Example-based model-transformation testing", Kessentini, Sahraoui, and Boukadoum describe a technique for generating test oracles for model transformation testing. Their approach is based on grading candidate oracles' risk of incorrectness based upon the degree of variation from known-good examples.

Please enjoy these contributions to our understanding of the control of change during the software lifecycle, and feel free to email the authors or me with any thoughts.