Intrinsic Definition in Software Architecture Evolution

Jeff Magee, Imperial College London, UK

Session Chair: Tom Mens

Incremental change is an integral part of both the initial development and the subsequent evolution and maintenance of large complex software systems. The talk discusses both, the requirements for and the design of, an approach that captures this incremental change in the definition of software architecture. The predominate advantage in making the definition of evolution intrinsic to architecture description is in permitting a principled and manageable way of dealing with unplanned change and extension.

Intrinsic definition facilitates decentralized evolution in which software is extended and evolved by multiple independent developers. The objective is an approach which permits unplanned extensions to be deployed to end users with the same facility that plugin extensions are currently added to systems with planned extension points. The talk advocates a model-driven approach in which architecture definition is used to directly construct both initial implementations and extensions/modification to these implementations.

Existing approaches to dealing with the evolution of software architecture have focused on extending configuration management systems with architectural concepts. The intrinsic definition approach does not replace the need for version control provided by configuration management systems, rather it is intended as a way of capturing architectural change that can both co-exist with and exploit the use of industry standard configuration management systems.

An implementation of intrinsic evolution definition in Backbone is presented. Backbone is a development of the Darwin architectural description language (ADL) and it has both a textual and a UML based graphical representation. The talk uses Backbone and its supporting toolset Evolve to illustrate basic concepts through simple examples and reports experience in applying the language and tools to larger examples.

Short Biography of Keynote Speaker, Jeff Magee

Professor Jeff Magee is Principal of the Faculty of Engineering at Imperial College. He is a graduate in electrical engineering and holds MSc and PhD degrees in Computing Science from Imperial College London. He was appointed Head of the Department of Computing at Imperial in 2004 and Deputy Principal (Research) of the Faculty of Engineering in 2008.

His research is concerned with the software engineering of self-adaptive and distributed systems, including design methods, analysis techniques, languages and program support environments for these systems. His work on software architecture, that resulted in the Darwin architectural description language
(ADL) for distributed systems, was put to commercial use by Phillips in consumer television products as the Koala ADL. He has worked extensively with industry on joint research projects and as a consultant.

He is the author of over 100 refereed research publications in software engineering and has co-authored a book on concurrent programming entitled *Concurrency - State models and Java programs* which is now in its 2nd Edition and has been widely adopted world-wide. He was editor of the *Institute of Electrical Engineering's Proceedings on Software Engineering* and until 2007 was an associate editor of *Transactions on Software Engineering and Methodology*. He is the recipient of the 2005 ACM Special Interest Group on Software Engineering Outstanding Research Award for his work in Distributed Software Engineering. Most recently, he was awarded the SIGSOFT Retrospective Paper Award for his FSE4 paper on “Dynamic Structure in Software Architectures”.

**Reflecting Upon the Useful Life of Software**

Michael Feathers, *DepthFirst*

*Session Chair: Yann-Gaël Guéhéneuc*

Nearly every organization that develops software has maintenance challenges. Implicit in them is the assumption that it is valuable to keep existing software in service and that rewriting it would be cost prohibitive. There certainly is convincing evidence that both of these assumptions are true, but it's worth investigating whether we can finesse the problem of maintenance by treating software lifetime as a variable. In this keynote, Michael Feathers will explore this idea and its ramifications.

**Short Biography of Keynote Speaker, Michael Feathers**

Michael Feathers is a consultant with DepthFirst, prior to that he was Member of the Technical Staff at Groupon. Prior to joining Groupon, Michael was the Chief Scientist of Obtiva, and a Senior Consultant with Object Mentor International. Over the years, Michael has spent a great deal of time helping teams after design over time in code bases.