Model-Based Testing for Enterprise Software Solutions
Ashish Jain
Telcordia, One Telcordia Drive, Piscataway, NJ, USA 08854
jain@research.telcordia.com

Abstract
Trends in model-based development which are enabling model-based testing of complex enterprise software solutions are outlined. However, a fundamental factor contributing to lifecycle costs continues to be the inability to effectively handle planned or unplanned changes. With respect to lifecycle costs, the current model-based approaches are off-target, and an integrated model-based development and testing approach to get us back on track is outlined.

1. Introduction
Agility stems from the ability to rapidly change software components. However, enterprise software solutions tend to be a complex web of system of systems. Thus, besides the flexibility of individual component, another factor which is a fundamental to agility is the ability to do rapid, even automated, change impact analysis in this complex web of systems. I will present modeling trends which are enabling creation of flexible components and thus fostering model-based testing. I will also discuss the gaps in the current efforts that are preventing rapid change-impact or synchronization analysis.

2. Enabling Model-Driven Testing
Meta-data driven components make IT infrastructure flexible. Modern day system components are more like model-interpreters and less like a prescriptive sequence of program instructions. Models or meta-data that these programs interpret are becoming more formal like XML Schemas, Workflow specifications, and Business Rules. Modelling of data, workflows, and business rules in a platform independent way using open standards is now common. Declarative specifications are a perquisite for model-based testing, and such specifications are finally becoming readily available. Further, testers and testing tools now have the ability to directly reuse the models created by business or system analysts. However, currency of the information is vital because of the dependency chain. For example, once the tester has annotated the system level models for testing, keeping the two models synchronized becomes critical. For the remainder of the paper, I will discuss gaps in the present-day modelling approaches for keeping information synchronized, and I will also outline an approach for addressing this gap.

3. Managing Lifecycle Changes
A process-heavy and unstructured document-centric information management approach is not cost-effective. Artifact management using structured models; however, has considerable promise. Models for all the software artifacts and formalisms to capture the inter-model dependency are a critical gap. Example of the artifacts includes business requirements, system requirements, system designs, and test plans. An integrated environment that allows role-based views and consistent editing of models for analysis, design, and testing is the critical second gap.

Current efforts, including OMG’S MDA, are focused on mapping formalisms to go from Platform Independent to Platform Specific Models. While mapping to platform specific models or code helps, and automation helps a bit more, major costs are not in the final development of the application code (!). Thus, efforts in transforming detailed models to foster automation not only takes away from the value of modeling (modeling is about precision and abstraction, not details), but also solves a wrong problem.

3. An Integrated Framework
At Telcordia we are putting together an integrated development and testing framework. Using the framework, engineers are developing interface models for the components. Interface models consist of message models and use-case specific business rules. Business analysts create the business use-case models. Use-case models are formally linked to the interface models. All models are stored in a single logical repository. Testers reuse the use case models and the interface models to create testing use-cases (scenarios). Though we do model-analysis to automatically generate test cases for wider coverage of use-cases, it has been our experience that the biggest value of the framework is not automation, but the ability to manage changes in the test plan in face of lifecycle changes.

1 Business rules are constraints or production rules.