From Model-Driven Testing to Test-Driven Modelling

Dr. Darius Silingas
Principal Consultant @ No Magic
What Is this Session About?

Disclaimer: I am not and have never been a Software Tester

... but I have worked with a lot of software testers
... and I know that a tester can figure out the bugs
... but a developer has to fix them
... and he can prevent them by using test-driven approach

This session is about installing testing approach into software modelling activities for getting better quality models and thus better quality software based on those models
This is an advanced session, which assumes that you have:

- A basic understanding of Model Driven Development
- A basic understanding of Test Driven Development
- A desire to build better quality software 😊
Towards Synergy of Trends

- Improve MDA with Agile practices
- Raise the level of abstraction for Agile practices
Let’s practice example/test-driven approach and learn the principles of Test Driven Modelling by analyzing how to apply it to a small example system MagicTest.

MagicTest provides functionality for a teacher to create test questions, group them into a test and assign it to the class that he is teaching. The students of that classes make assessments of the test that are automatically evaluated.
System Layers

User Interface

Business Logic

Data Layer
How do I know if this data structure is suitable?
A sample shows inconsistency with data structure.
What Can We Do When Sample Violates Data Structure

1. Fix data structure (sample is good, structure was wrong)

2. Fix data sample (sample is wrong, structure was good)

   - Use invalid sample in testing for ensuring that such cases are not allowed and handled appropriately in the system
Decision: Enable multiple supervisors
A sample is now in sync with data structure.
A data designer should prepare it along with data structure

Testers should take over with creating variations
System Layers

User Interface

Business Logic

Data Layer
Identifying business logic services as UML components

Modeling interaction scenarios as UML sequence diagrams

- The operations of business components can be created as needed
- New necessary service components can be created as needed
- JUnit tests or their skeletons can be generated from interaction scenarios

✓ UML tools support these scenarios quite well
Modelling Interaction Scenario at Business Logic Layer

Service operations are discovered based on scenarios

- `TestAssessmentService`
  - `startTestAssessment()`
  - `submitAnswer()`
  - `finishTestAssessment()`
  - `evaluateTestAssessment()`

- `TestManagementService`
  - `startTestAssessment()`
  - `submitAnswer()`
  - `finishTestAssessment()`
  - `evaluateTestAssessment()`
System Layers

User Interface

Business Logic

Data Layer
GUI Modelling in UML

GUI structure can be captured in UML class or composite structure diagrams

- Modern modelling tools typically provide graphical rendering to enable better presentation for users and GUI designers/analysts

1. Identify abstract GUI screens as UML classes

2. Build abstract GUI navigation schema as UML state machine

3. Build a story board GUI sample set as UML instance specifications for validating GUI design with data samples
   ✓ Reuse data for GUI samples from data design (or vice versa)
How do I know it is suitable?

---

GU Navigation Schema

1. **Start**
   - **LoginDialog**
     - exit / submit login info
     - Login
     - [login incorrect] / show error msg
     - [login correct]
       - [password expired] / show change password fields
       - [password not expired]
         - **Authorized**
           - **StudentProfile**
             - EditData
             - OK
             - TakeTest
           - **UserDetails**
             - OK
             - GetTestResults
           - **TestAssessmentWindow**
             - Finish
             - at (timeout)
           - **TestResultsWindow**
             - OK
             - Quit
Validates both navigation schema and screen structures
Validates both navigation schema and screen structures
Validates both navigation schema and screen structures
1. Conceptual mistakes in design models can be found early by validating it with example/test models.

2. Quality engineers need to be involved in collaborative modelling together with developers.
   - Developers should produce essential example models.
   - Testers should take over with creating variations for a better coverage of error-prone situations.

3. Example/test-driven modelling enables achieving a better quality of model-driven software by testing design models early.
Towards Practicing Test Driven Modelling

**Tool aspect:** Modelling tools like MagicDraw already provide partial support for test-driven modelling – they are not a show stopper.

**Human aspect:** Test-driven modelling requires a change in modeller’s mindset for enabling more mature modelling culture.

**Action point:** bring these ideas back to your organisations and see if you can influence a change.
The End

Thank You for the attention!

Any Questions?

Let’s Keep in Touch:

E-mail: darius.silingas@nomagic.com
Skype: darius.silingas
Get connected at LinkedIn social network