Applying Model Based Testing to the Windows Web Services Software Process

MBT User Conference 2011

Kunal Chopra
Rafael Forsbach
Huiyong Xiao
Why MBT?

**MBT advantages**
- Better understanding of features
- Risk frontloading
- Same language for Dev, Test and PM discussion
- Natural support for scenario test
- Better documentation

**Product features**
- Databases
- Event logging
- Stateful client
- Protocol
- Work flows with complex internal states
- Service APIs with complex internal states
The 3 Step Process
Plan – Pilot – Implement

Step 1 – Plan
1. Review Existing Solutions
2. Consolidating Educational Resources
3. Management Approval

Step 3 – Implement
1. Feedback Incorporated
2. MBT implemented as a Test Methodology
3. It is another tool in the tester’s toolbox

Step 2 – Pilot
1. Establish Pilot Group
2. Apply MBT to Windows Web Services Features
How did we “Fit” in?

Traditional Software Lifecycle in the Windows Web Services Product Group

The dates shown are for illustrative purposes only
Product Cycle Phase I
Pre specification

Pilot Results

Reviewed Processes

Commit to it

Best MBT Candidates

Feature A

Feature B

Feature C
Product Cycle Phase II
Specification & Design

Start Modeling

Model 1

Model 2

Model 3

Find design flaws
Functional/Development Specifications

Test Design Specification
Standardize MBT specs
Review with stakeholders

10M states! -> Slice, refactor, rethink model
Support group -> help new MBT testers
Product Cycle Phase III Development

- Complete Models
- Model Reviews
- Test Case Generation
- Instrument Tests
- Submit Tests

Support group

Buried in failures! Now what?

Test driven development

Test Case Manager Integration
Product Cycle Phase IV Testing & Stabilization

1. Test harness integration
2. Submit Tests Again
3. Fix the Models
4. Generate new Tests
5. Adapt to Changes
6. Update support materials
7. Handoff Process
MBT Adoption Results

**MBT Tester Adoption**

- People Adoption
  - 0% at 0 months
  - 10% at 6 months
  - 30% at 12 months

**MBT Bug Types**

- Code Bug: 74%
- Test Bug: 15%
- Spec Bug: 10%
- Doc Bug: 1%

**MBT Bug Resolutions**

- Fixed: 93%
- Won't Fix: 1%
- By Design: 1%
- Duplicate: 0%
- External: 1%
- Not Repro: 3%

**MBT Fixed Bug Priorities**

- 1: 31%
- 2: 55%
- 3: 14%

**MBT Tester Adoption**

- People Adoption
  - 0% adoption at 0 months
  - 6% adoption at 6 months
  - 10% adoption at 12 months
  - 3% adoption at 24 months
  - 2% adoption at 30 months
  - 1% adoption at 36 months
  - 0% adoption at 42 months

**MBT Fixed Bug Priorities**

- Priority 1: 31%
- Priority 2: 55%
- Priority 3: 14%
MBT Survey and Takeaways

- 64% Positive
- 36% Negative

- Modeling really helped us review our feature specs with the feature crew using a diagrammatic representation.
- There was a lot of enthusiasm from the people involved in modeling.
- Modeling helped me find additional bugs around the UI as a result of generating models.
- There were a huge number of spec issues found when modeling.
- The TDS was very detailed because of modeling and very understandable by the concerned parties.
- The model blew up because of capturing all the functionality in one model.
- Needed to redo the model a couple of times, so spent some time modeling.
- Once you have too many states, it's hard to keep track of them all in your actions.
- It would be great to validate the model quickly e.g. at every level you can find how many nodes are generated.
- If you don't build your model piece by piece there is no assurance that you did it correctly.
- Maintaining the model takes quite a bit of work.
- Would like to see a good process to track code coverage for model test cases.
- Would like to see more efforts to see MBT integrated.

Think small, think scenarios, break down the model to small user interaction pieces.
Enhancing data sets with real world data to populate strings.
Questions
More Resources

- Windows 8 Engineering Blog
  - http://blogs.msdn.com/b/b8

- Information on the //BUILD conference
  - http://www.buildwindows.com