Lesson learnt from integrating MBT for Messaging App

By Chan Chaiyochlarb

Why MBT?

Motivations for adopting MBT

Find bugs earlier

Exploration of missed paths

Increase validation

Easy to adjust to specification changes

Real world scenarios

Help find the last 20% of bugs

Reduce test cost

MBT For Messaging App

Microsoft

MBT For Messaging App

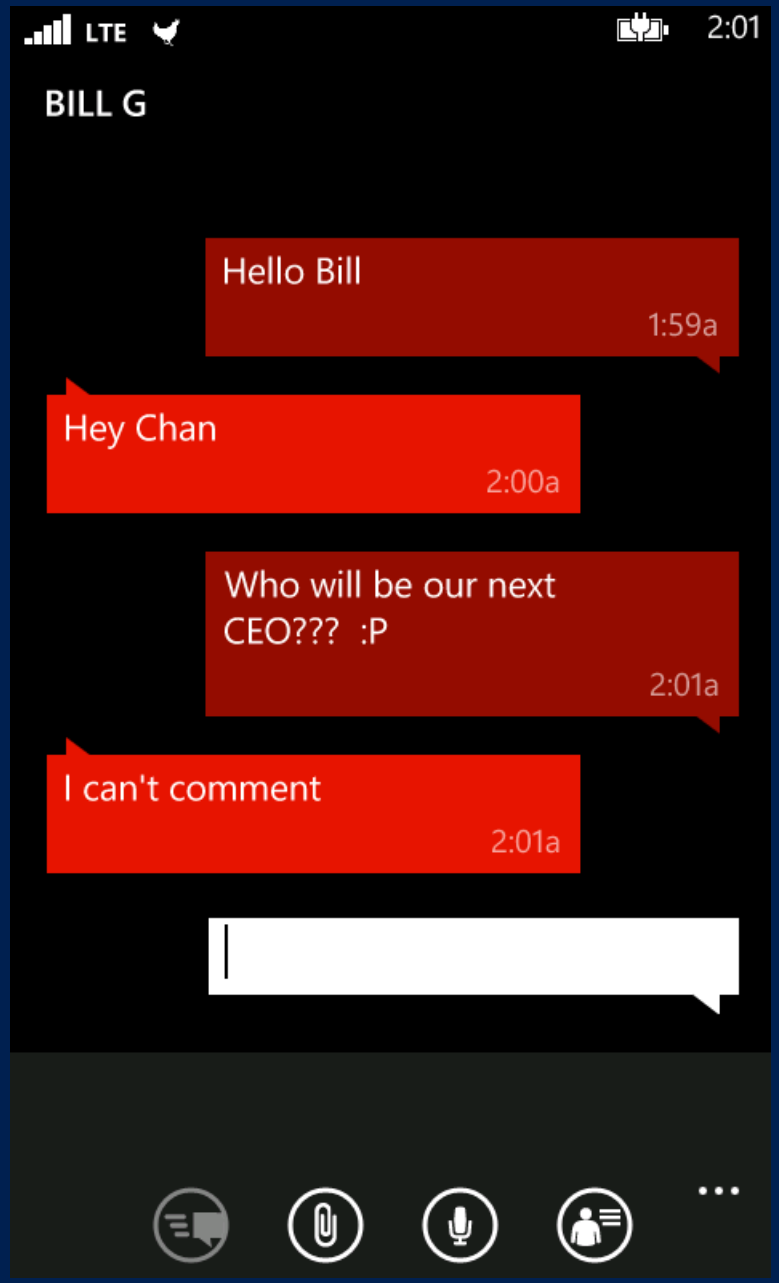
90% of test automations are MBT generated

Real user End-to-End scenarios

Permutations of actions yielding high coverage

Error scenarios

UI and backend verifications



Microsoft

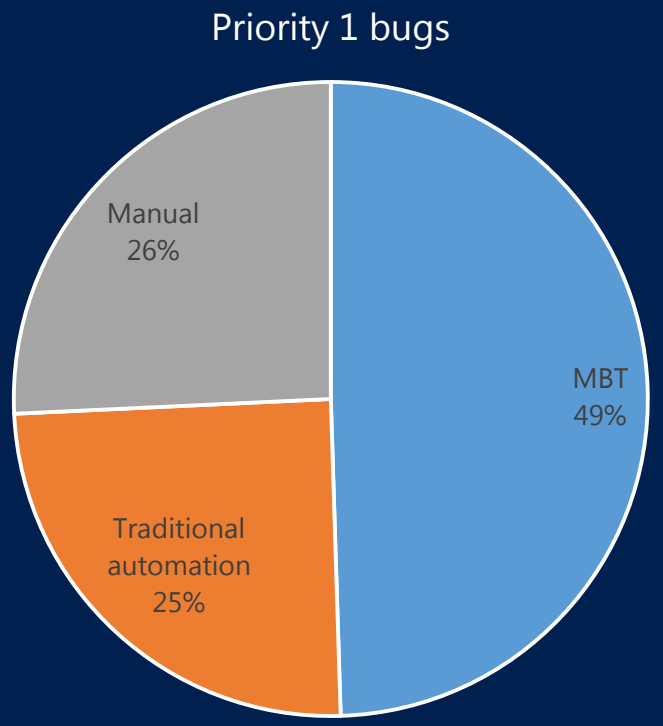
Analysis: The Good

MBT finds the most high priority bugs

Uncover a lot of functional specification bugs

Catch a lot of regressions

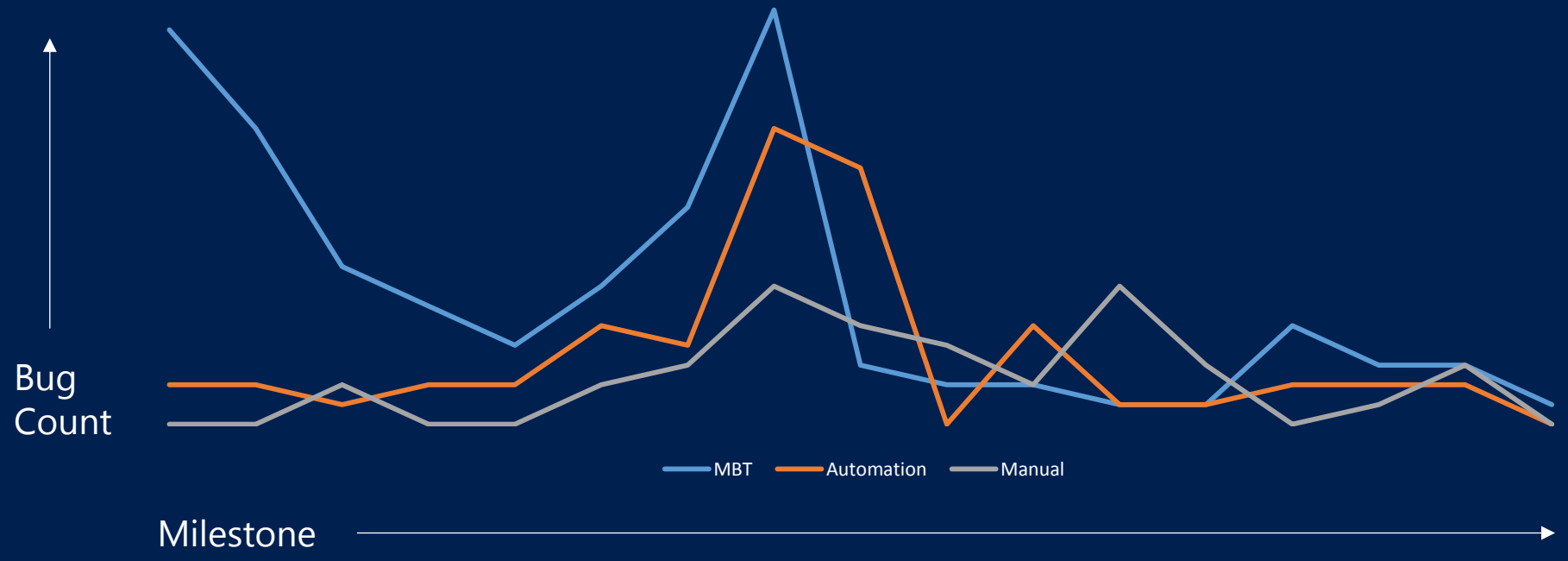
Lots of validations



Find bugs early

Test the specification

Model development in parallel with product



Microsoft

Agility

Easily react to new feature changes

Reusability of test semantics

Early test engagement

Drive quality upstream

Analysis: The Bad

MBT is not easy

Different mind shift from traditional testing

Steep learning curve and high ramp up cost

Need to pick the right tool set

Difficult to explain test coverage

Complex Design

Single model which represents the whole Messaging Application

Model is nearly as complex as the product

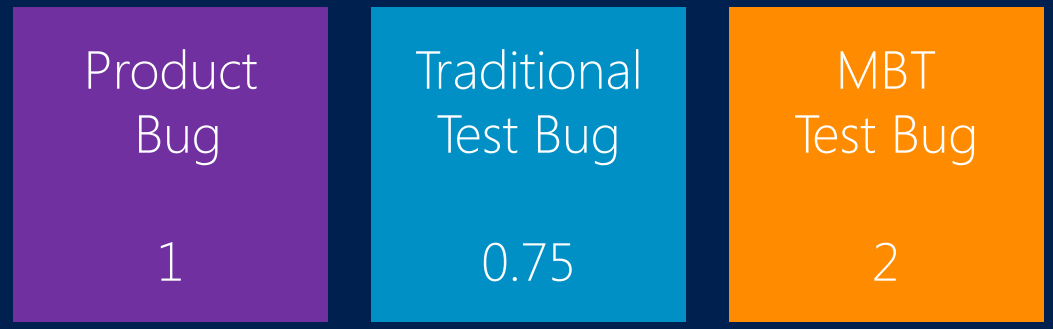
Bug in model is difficult to find

State tracking and other book keeping (for validation) make things even worse

Every behavioral change has large impact to existing scenarios

Maintenance Costs

Complexity kills
Bug ratio



More test code means higher maintenance costs
Bug turn around time nearly double developers

Reflections

Back to our original motivations

Find bugs earlier	Yes
Exploration of missed paths	Yes
Increase validation	Yes
Easy to adjust to specification changes	Yes
Real world scenarios	Maybe
Help find the last 20% of bugs	Maybe
Reduce test cost	No

Moral of the story

What did we learn?

MBT is different

Model Design is important

Smaller model is okay

It's okay to have multiple models for different feature set

"Use MBT to generate a lot of test cases" paradigm is misleading

Resist the temptation to use MBT for everything

Knowing MBT strengths and weaknesses

MBT is highly effective for stateful system, or with systems lots of input/output combinations

For stateless system with simple inputs/outputs, it might be more effective using data-driven approach instead

For undeterministic behavior, MBT might not be a good fit

Example: Image resizing algorithm, data decompressor, etc.

Thank you

Microsoft