



EFFECTIVE TESTING TECHNIQUES FOR UNTOLD STORIES IN STORY-DRIVEN DEVELOPMENT

Erbil YILMAZ, Senior SDET
Visual Studio Ultimate, Microsoft
Erbil.Yilmaz@microsoft.com

AGILE DEVELOPMENT

- In each sprint, teams take a few stories and implement
- Demonstrate and complete the customer experience along the stories designed
- Proving the software being built can deliver the customer experience



TESTING CHALLENGES WITH AGILE

- Usually, different parts of the software are developed by different sub-teams
- As a result, at the end of each sprint, the software product consists of:
 - a. Features that are developed for the told stories
 - b. Features/defects that emerge from untold stories
- Focus on testing the story within the iteration accumulates test debt for integration points
- More defects will be discovered around the integration points (untold stories) rather than coming from existing test coverage

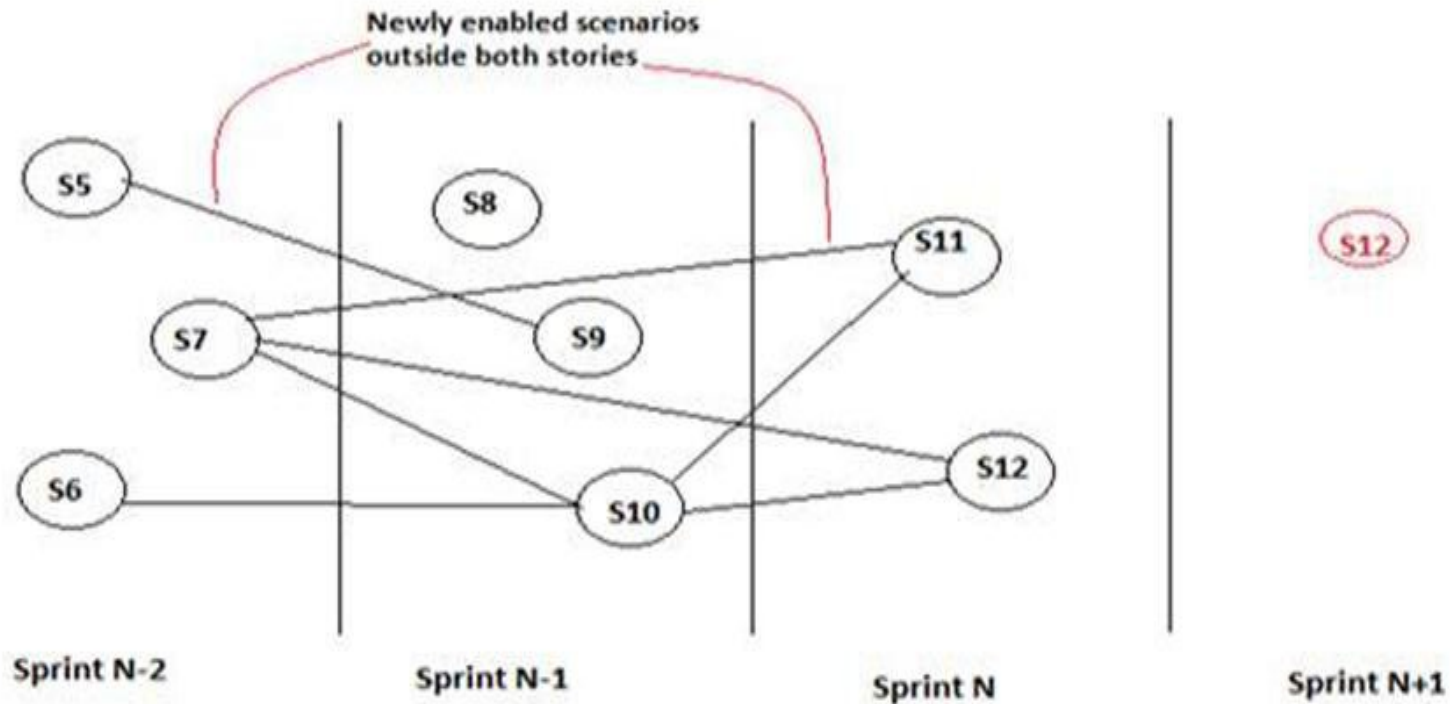


PROBLEM STATEMENT

- 100 % test automation goal
- Majority of automation discovered bugs fixed
- 95+% test pass rate
- Over 1000 triaged bugs after Code-Complete
- Mostly found on integration scenarios
- Found by manual testing
 - Bug bashes
 - End-to-end scenario testing
 - PUM walkthroughs
 - Demo preparations
 - Customer feedback



UNTOLD STORIES (LINKS AMONG STORIES)

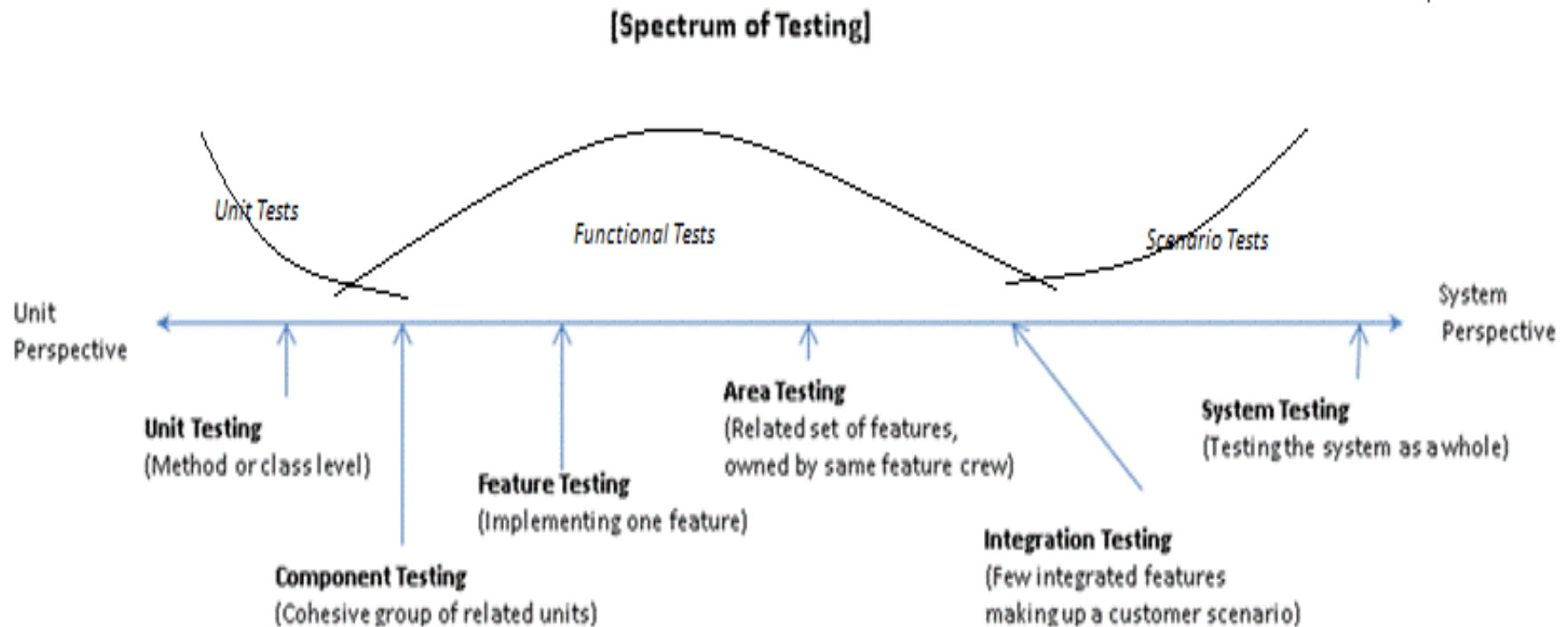


TECHNIQUES FOR MANAGING UNTOLD STORIES

- Using Test Spectrum
- Test Meta-Data
- Using Architecture Diagrams to Relate Stories to Components
- Story Links over Architecture Diagrams



TEST SPECTRUM



TEST PLANNING GUIDELINES WITH TEST SPECTRUM

- Multiple levels of coverage are required focusing on different aspects of the implemented code
- Optimal coverage could be achieved only by a combination of the various elements in the test spectrum
- Choose a good set that is a combination of these types
- Do not rely only on unit and feature tests
- Test plan review checklist
 - Call out integration points with other components/features/areas that are implemented or planned
 - Either add tests for these to the current test plan or add a task to the backlog to identify or implement these later
 - Feedback about missing integration pieces in the coverage



TEST META-DATA

At a minimum, each test case should have meta-data that describes:

- Components to be tested with their scenarios,
- Test types according to the test spectrum
- Stories that the tests are covering



META-DATA WITH VISUAL STUDIO

- Simple:
 - [TestProperty("PropertyName", "Value")]
 - [TestProperty("IntegrationTest", "ToComponent")]
- Advance:
 - Custom implementation of TestAttributes
 - [IntegrationTest("Component1", "Component2")]
 - [ComponentTest("Component1")]
 - [StoriesCovered("Story1")]

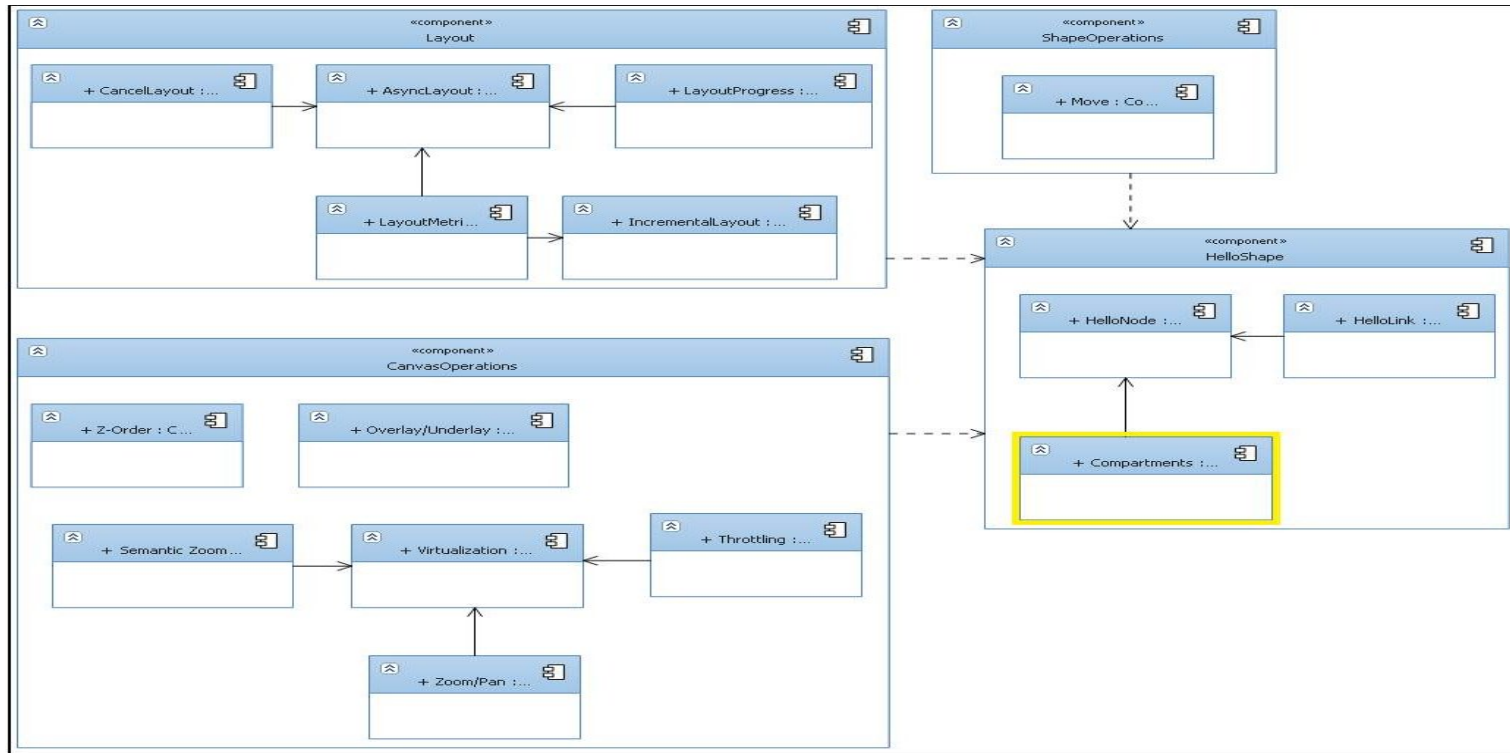


USING ARCHITECTURE DIAGRAMS

- Each test plan had one or more UML architecture diagrams
- Using these diagrams in test plans helped relate test areas to product architecture
- Help question the test plan in terms of links between components/layers in the product architecture
- Link between the components being tested and other components means there might be a code execution path that enables a story (possibly an untold one)
- This will translate as requirement to design a set of test cases in the test plan to cover the interaction



SAMPLE DIAGRAM



STORY LINKS OVER ARCHITECTURE DIAGRAMS

- In these diagrams, each story that has been implemented thus far overlays architectural components that implement or enable that story
- As more stories are implemented, these component diagrams encapsulate all stories overlaid on the component diagrams that represent the current architecture
- If there is a link between two architectural components, it implies a link between the stories that overlay those architectural components
- Each link at either level deserves an investigation on how to design test cases to cover the interaction
- If the link is at component level, the corresponding test could range from component test to integration test
- If the link is at the story level, then the corresponding test could range from feature testing to system testing



CASE STUDY: CLOSING INTEGRATION TEST GAP

Problem Statement:

In Sprint 5, team discovers that existing test coverage is sub optimal and needs to be improved for integration points between already implemented stories



CASE STUDY: CLOSING INTEGRATION TEST GAP

Two Step Process:

- Investigate and identify the low-level testing gaps for all areas
 - Use area tests only for code-coverage runs and, without integration/system level tests
 - Figure out how much coverage gap they have with existing tests
 - All areas should have good low-level test coverage by only area test
- Go over all the previously implemented stories and looked for possible integration links or scenarios across the areas



CASE STUDY: CLOSING INTEGRATION TEST GAP DISCOVERING MISSING LINKS

Context: 25 stories implemented in five sprints

Example: when we implemented the Cancel story,
the Update story did not exist

- QA team members had brain-storming meetings to enumerate integration scenarios across components or areas as well as stories
- They updated the test plans as they found these integration scenarios
- Easy at this stage as the team had already started using architecture diagrams in their test plans



CASE STUDY: CLOSING INTEGRATION TEST GAP DISCOVERING MISSING LINKS

- Going forward, all test plans will have sections for integration scenarios
- The team gets better at questioning the integration points between stories
- Features are more immune to impactful bugs that fall into the cracks between stories or components



LEARNING SUMMARY (1)

- Ensure that you have a good set of test cases across the test spectrum for a given story
- Do not focus only on unit tests or acceptance tests, but also carefully consider integration points for already implemented stories
- Mark your test cases with meta-data associated with test coverage intention, test type, and the story or feature that is related to the test
- This will provide traceability and a test bed that you can query.



LEARNING SUMMARY (2)

- Use internal product knowledge, especially architecture diagrams, to map stories to actual code
- Further, design test cases over this mapping with both links to stories and product code.
- If you fail to identify integration points within the iteration, add high priority test coverage work items to the backlog as you discover them
- Do not accumulate debt as it carries significant risk closer to the release date



WHAT IS NEXT?

- Defining a schema over meta-data
- Integrating manual tests (exploratory charters)
- Team culture: use incoming integration bugs for learning opportunity
- Track metrics over a full-release



QUESTIONS?

Now: Feel free to ask

Later in PNSQC: Erbil Yilmaz

After PNSQC: Erbil.Yilmaz@Microsoft.com

