Large-Scale Integration Testing at Microsoft

Jean Hartmann
Test Architect
Developer Division Engineering Systems
Microsoft Corp.

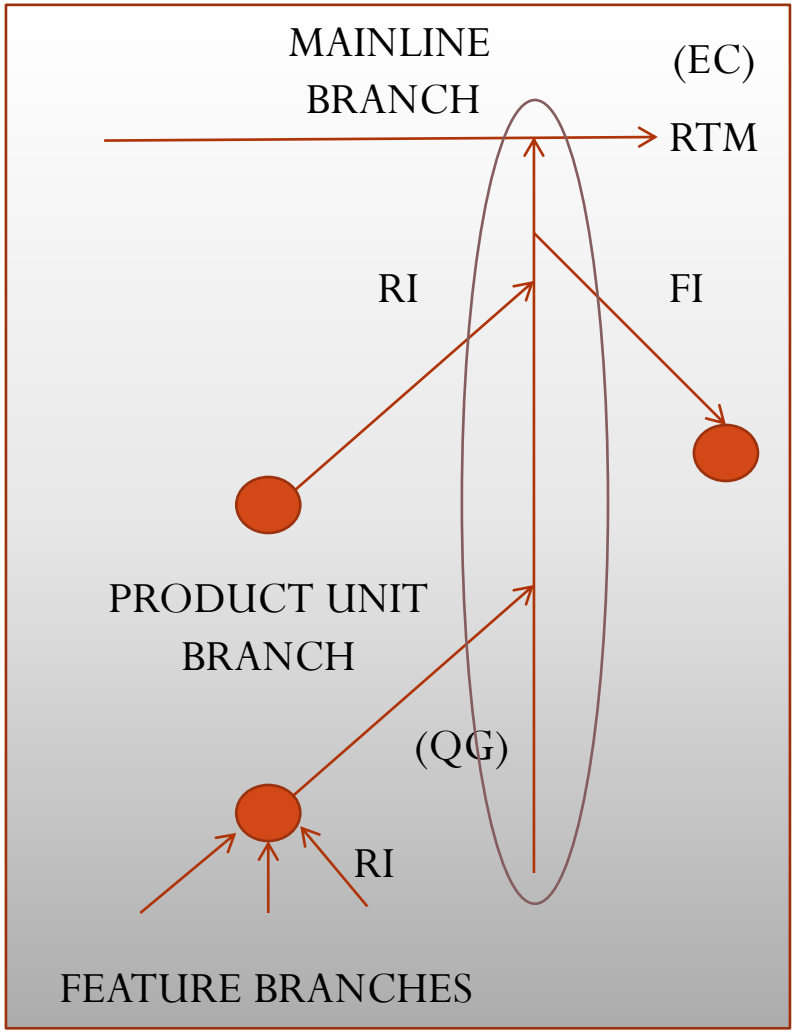
jeanhar@microsoft.com

Agenda

- Integration Testing @ Microsoft
- Definition thru Deployment
- Enhancing Tools and Infrastructure
- Benefits and Challenges
- Conclusions and Future Work

Integration Testing : Terms

- Builds
  - Feature Branches (FB's)
  - Product Unit (PU) Branches
  - Mainline Branch (Visual Studio)
- Integrations
  - Reverse Integrations (RI's)
  - Forward Integrations (FI's)
- Testing
  - Quality Gates (QG)
  - Exit Criteria (EC)
- Feature Crews



Integration Testing : Code Flow

- Large number of PU branches with complex dependencies
- *Breaking changes* quickly propagate across build tree
- Debugging becomes a nightmare, randomizing teams
- Often cripples PU and mainline builds for days
- Teams become grid-locked and begin to isolate themselves
- Huge effort required to unblock and produce stable mainline build for major milestones, e.g. CTPs, Betas and RTM
- Want to get to a "**fearless FI"** scenario!

Integration Testing: Test Perspective

- PU branches often not at *self-host* quality before RI
- Only informal (*non-blocking*) testing at RI time
- Central validation suite available, but:
  - Not focused, too large and included legacy tests
  - Run too late (against mainline branch after RI's)
  - Tests written against various automation frameworks
  - Tests required difficult-to-reproduce installs/set-ups
  - Most tests were UI-based, often unstable
  - No standardized logging for easy debugging of test issues

Defining a New Suite: Criteria

- Strike balance between *efficiency* and *effectiveness*
- Apply more stringent test selection criteria from the onset
- Well-defined update process to evolve it alongside product
- Efficiency criteria
  - Tests execute in a two hour timeframe (excl. setup)
  - Tests are reliable, consistent and easy to debug
- Effectiveness criteria
  - Tests exercise key product integration points
  - Tests reflect customer success scenarios ('happy paths')
- Result- Two-tiered test set (RI Tests)

Defining a New Suite: RI Tests

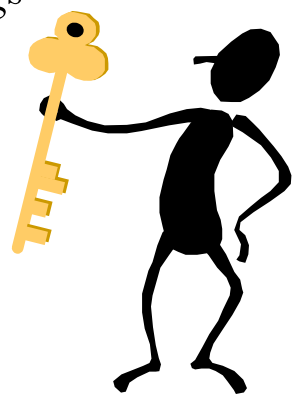
- Primary – Developer Division RITs (DDRITs)
  - Testing major integration points coinciding with top customer success scenarios
  - Failure of this suite during RI means blocking bugs (P0)
  - Quick turnaround for fixes needed
  - Currently, tens of test cases
- Secondary – Product Unit-specific RITs (PURITs)
  - Testing unique integration points between specific PU's
  - Failure of this suite during RI does not block integration (P1)
  - Turnaround time dictated by SLA (service level agreement)
  - Currently, tens of test cases

Defining a New Suite: Review

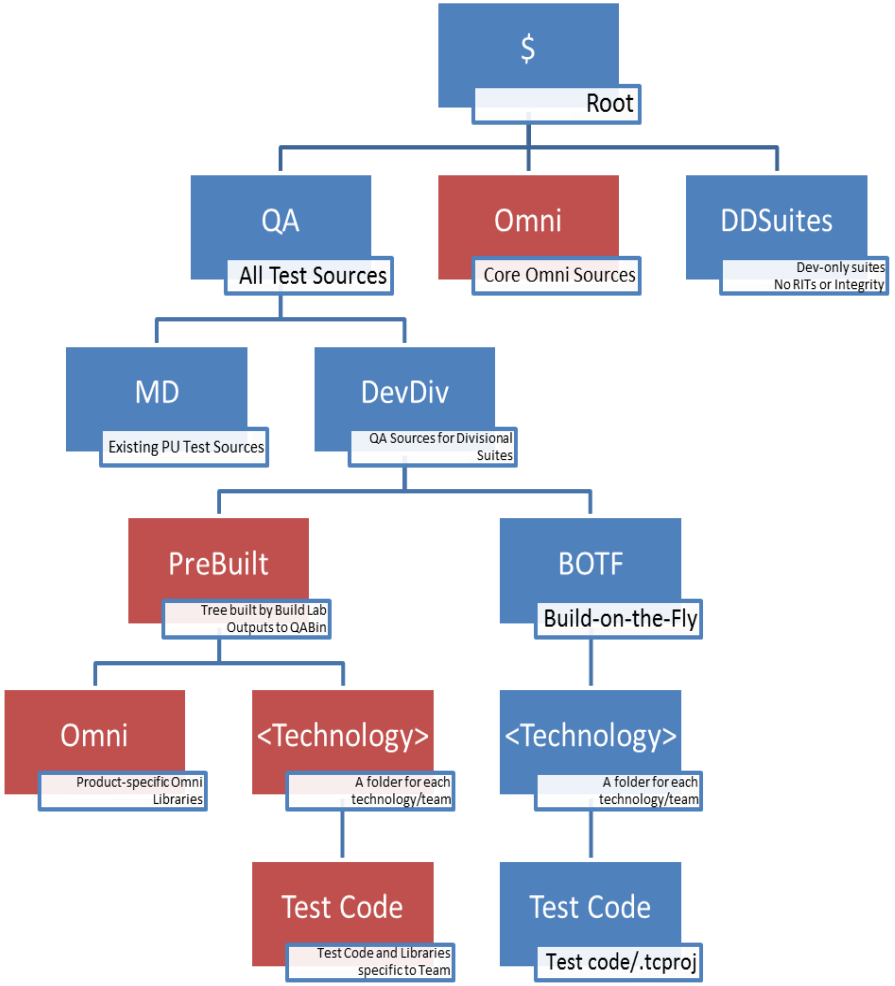
- DDRITs
  - *Divisional* review committee established (development and test managers, senior technical staff)
  - Review and approve proposed tests, ensure strict adherence to selection criteria
  - Considered functional and key non-functional (performance) tests (within time constraints)
- PURITs
  - Review committee comprised of staff from the partnering, dependent teams only
  - Test purpose and content were captured in SLAs
- Reviews form integral part of new integration testing process
- Critical in helping teams define new RITs, update existing ones and deprecating old tests

Defining a New Suite: Resolution

- Goal - minimize randomization during issue resolution
- Requires good inter- and intra-team communications
- Resolution task needs to have an owner per team – *PU RI Rep*
- Job description
  - Initial triaging (investigation) and assignment of resulting bugs
  - Collaborating and liaising with:
    - Developers and testers within own team,
    - Dependent partner and central tooling teams to resolve bugs
  - Updating of SLA agreements, if applicable
- Later on, I'll show the workflow…



Implementing the New Test Suite



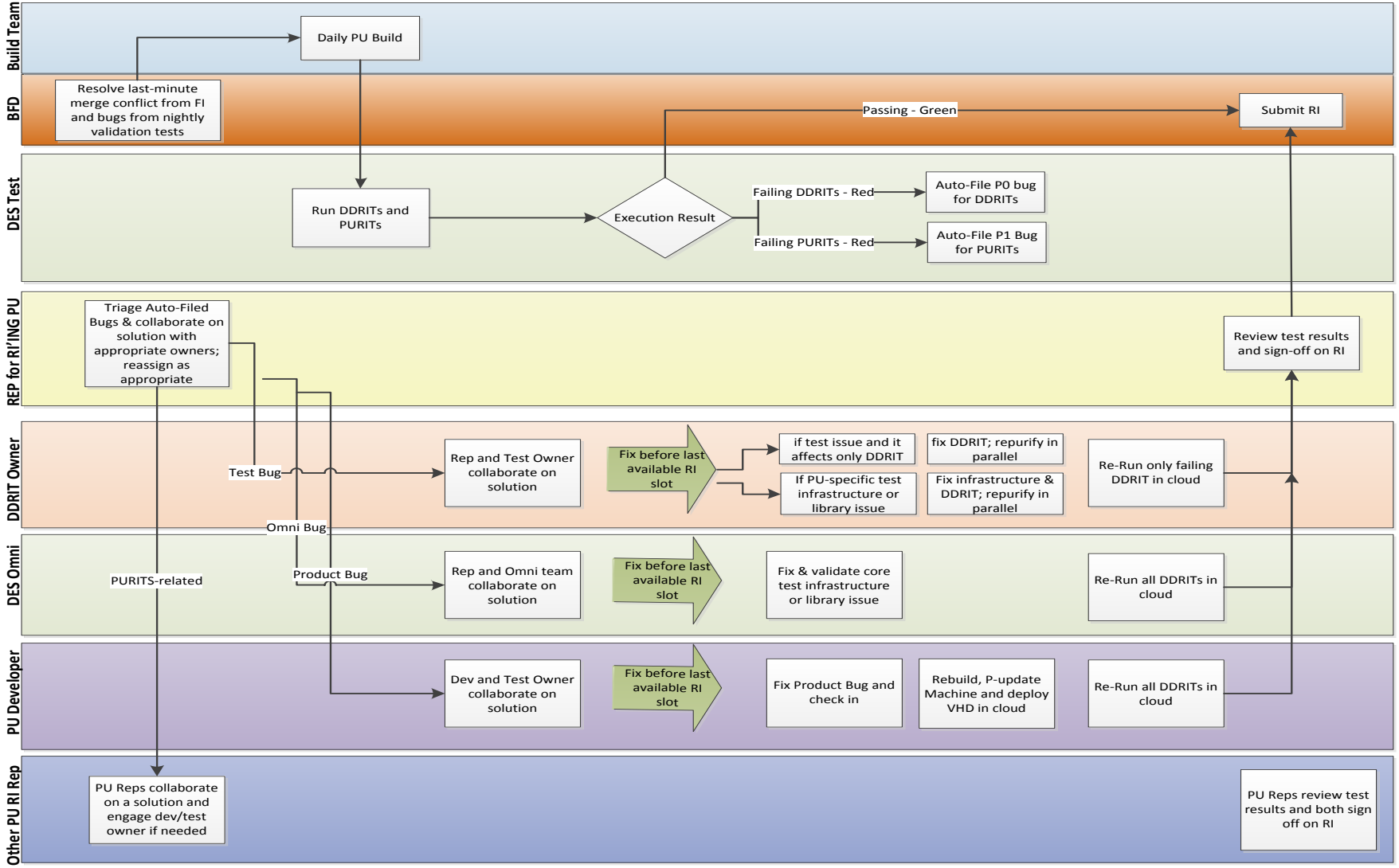
- Implementing with efficiency and effectiveness in mind
- Balanced portfolio of UI- and API-based tests
- Standardized on two automation frameworks (UI and API)
- Tests stored in central location (build tree)
- Core tooling and product-specific tooling libraries logically separated
- Tests were ‘purified’
- Ensuring tests were TRA-compliant (Test Run By Anyone)

Deploying the Test Suite

- Consolidating our existing, team-specific testing lab resources
- Moving to new cloud infrastructure at a geographically remote site
- RIT are executed using Virtual Machines (VMs), enabling quick re-imaging of machines using canned VM images (VHDs)
- Test case management system streamlines test run queue by holding VMs for similar test configurations
- As failures occur, images are reserved for later remote debugging
- RITs executed by our central engineering team (DES)

Results Reporting for New Test Suite

Deploying the Test Suite: Workflow



Enhancing Tools and Infrastructure

- New API-based test automation framework
  - Evaluated and consolidated existing frameworks
  - Implemented common logging framework
  - Adapted product interfaces to improve testability
  - Part of Omni (UI-/API-based) automation infrastructure
- Test purification and portability (TRA)
  - Providing tool support to ensure test reliability and stability
  - Enhancing test cases (and management system) to capture test case metadata
- Dependency analysis, code coverage and churn tools
  - Providing better intelligence on RIT coverage, public API churn and integration order during RI's
  - Goal is to evolve the integration test suites alongside the product, ensuring effectiveness

Benefits and Challenges

- Increased code velocity throughout the build tree (RI/FI)
- Earlier defect detection to prevent major breaking changes
- Faster and more reliable test execution, easier failure analysis
- Reduced effort to meet quality goals for major milestones

- Defining, socializing and executing on this concept
- Introducing the API-based testing concept/product changes
- Establishing new cloud-based infrastructure to execute tests
- Restructuring the build tree to ease test case management, execution and maintenance

Conclusions and Future Work

- Gave insight into existing integration test issues @ Microsoft
- Discussed our new integration test strategy to ensure more rapid code flow
- Highlighted our process and tool improvements, deliverables and deployment issues

- Comparing fault detection (between old/new test suites)
- Refining processes and tools, e.g. contention for same machine for debugging test failures
- Investing in tools to track test suite adequacy (with results being fed back into process)