Development-Driven Testing
Ensuring Testing Meets the Needs of Software Developers

tim.farley@gylity.com
Anybody here work in SQA? Testing? Who’s your customer? Both? How much time do you spend on each?
End user? Focus on final validation of product requirements. Clearly, he’s counting on your product to work!
Developers? Focus on sustaining the development effort.
You’re probably concerned with both. Probably more focused on the end user.
This talk is about focusing on your internal customers, in particular the software developers.
This is what I’m calling Development-Driven Testing. A testing strategy that is focused on the needs of developers rather than the end user.
The way you do this is by understanding the constraints on the project and the critical needs of the developers.
Constraints on the testing strategy.
Requirements Developers have in order to do their job effectively.
Read the paper to understand how to determine developer needs and assess your SQA capabilities.
This part is the print engine. It’s the part that actually puts the image on the page. There are other components.
This was a large project that lasted a long time. Iterative means they developed a prototype, wrote enough software to make it work, and learned enough from it to build the next prototype. Old code was discarded as new code was written with each prototype.
Large team can’t ever sit idle. Burn rate was one million a day for the project. About 100k a day for the engine. Prototypes were 100k each. Needed to make that investment last. Also, SQA and Development wouldn’t be getting many or any. Integration site had no overlapping hours with engine development site. And all other components were new too.
Huge team needs to sustain development progress. Immediate feedback needed to ensure there’s always a working build and no regression failures. Need to extend the life of expensive prototypes by supporting multiple builds for multiple prototypes. Can’t have integration problems due to the engine. Everything else depends on a working engine to test. Would delay engine development too.
Not clear whether SQA organization could actually do any of this.
Very little of this is about running tests. Your ability to deliver immediate feedback depends on many other things besides just running the tests.
Still need to demonstrate test coverage against whatever has been implemented.
2. Test Execution Assessment

- **Current SQA Practice**
  - Testing cycles from 1 week (targeted) to 2 months (full regression)
  - Test on pre-production prototypes
  - Mix of manual and automated tests

- **Critical Development Needs & Project Constraints**
  - Full regression test needed every day
  - No budget for prototypes for daily full regression test
  - No budget for manual testers for daily full regression test

- **Assessment**
  - SQA needs a new way of executing tests
Summary reported coverage of requirements by feature.
Multiple Builds Assessment

- **Current SQA Practice**
  - Only latest build tested
  - Latest build tested on latest customer-intent prototype
  - Testing targeted at validating HW/SW for external customer use

- **Critical Development Needs & Project Constraints**
  - Testing to sustain development effort and extend life of prototypes
  - Multiple, simultaneous builds need to be tested
  - One build for each prototype revision in use

- **Assessment**
  - SQA needs to test multiple, simultaneous builds
Component Qualification Assessment

- Current SQA Practice
  - SQA tests integrated systems only
  - Tests run on customer-intent hardware
  - Human testers interact with device the same way end users would

- Critical Development Needs & Project Constraints
  - Test print engine as stand-alone component
  - No budget for prototypes for software developers and SQA
  - No way to interact with device as an end user would

- Assessment
  - SQA needs to qualify component without prototype hardware
Pretty much everything we were doing had to change to meet the needs of the developers.
This was a given based on the requirements and constraints. We couldn’t do anything without it.
Simulation environment allowed us to test without prototypes but with high confidence.
Automation let us test as often as we wanted without people.
Infrastructure allowed us to test as many different products as we wanted simultaneously.
TDD approach for test development. Some XP. Co-located.
Tests could have been run more often than daily. This is as much as Development could support.
Guidelines for automated test development to ensure parallelizable automated tests. Simulators were real controllers with simulated HW interfaces so that all real time requirements would be tested. Same as on a real device.
Daily test includes checking build, flashing devices, qualifying features before running full test, running tests, reporting results.
Gives developers confidence to keep developing.
This is not unit testing. This is testing the integrated component code on the real controller.
TRS – REALLY IMPORTANT
SQA INVESTIGATION – REALLY IMPORTANT
This sustained development momentum but also extended the life of old prototypes. You don’t have to extend the life of too many over 5 years to save a lot of money.
Component Qualification Adaptations

- Daily fully automated test on simulators
  - Full regression test (Functional, Performance, Stress, ...)
  - System communications protocol tested
  - Fully automated build checking, device upgrading, ...

- Weekly 4-hour test on integrated system with latest prototype hardware

- Complete regression test on integrated system with latest prototype hardware prior to program milestones
  - Automated tests run on simulators and real devices
  - Write the test once and use in both environments

It’s not enough to just make sure your part works alone in an unrealistic environment. There was a weekly integration release and we made sure the engine would work correctly prior to the release.
This was an $800M project with 450kloc in the final version – not counting what was written and thrown away – with 8 prototype iterations. The tens of millions is just direct costs. It’s hard to calculate all the savings. Prolonging the life of old prototypes. Preventing problems. Keeping people productive. It’s huge. The consumables cost saving by using simulators paid for the entire SQA engineering effort. Extending life of prototypes to avoid the cost of 100 over 5 years is another $10M. Could be more! Burn rate for engine was $100k/day. How many days do you have to save over 5 years to add up to real money? Missing market window was $1M/day. Same with that. Next clean sheet took 12 hours to bring up prototype that would have taken 6 weeks before.
Tell the “All testing is waste” story.