Real World Software Testing at a Silicon Valley High-Tech Software Company

By:
Giora Ben-Yaacov and Lee Gazlay, Synopsys Inc.

Contact: Giora Ben-Yaacov, Synopsys Inc., (650)584-1410, giora@synopsys.com

==========

Abstract

Silicon Valley high-tech software product teams face a troubling paradox on a daily basis -- How to introduce new technology and features faster than ever, while simultaneously improving product quality and responsiveness to customer quality issues.

This paper describes a methodology for allocating priority levels and resources to software testing and other quality activities to achieve "customer satisfaction". This methodology is based on understanding of what the market and the target users require at any point in time during the product technology adoption life-cycle.

The paper also describes the deployment by a leading market-driven company of effective software testing processes and methods that represent real-world customer issues.

Silicon Valley Culture for Software Testing

QA and testing professionals that are trained in the traditional software testing and QA methods often have a hard time understanding the reality of the Silicon Valley "good-enough testing" approaches. The "good-enough testing" approach is illustrated in Figure 1.

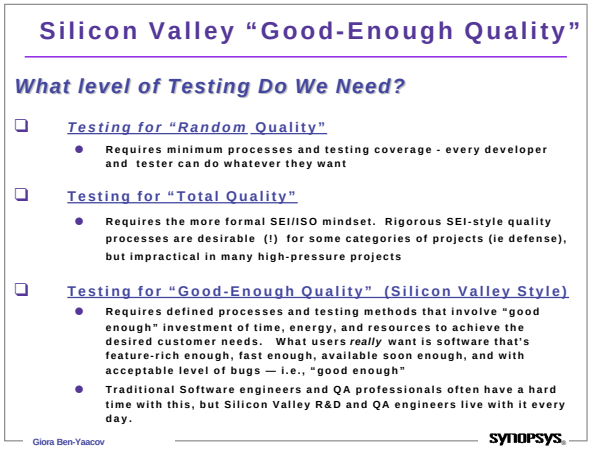

Figure 1: Silicon Valley "Good-enough testing" culture

Today, it is very hard to get highly trained software/electrical engineers. With this constraint, product development teams at high tech companies are constantly striving to balance competing Goals:

- Minimize time-to-market by delivering new technology as soon as possible
- Maximize customer satisfaction by delivering a specific set of features
- Minimize number of known defect in the shipped product releases (“built-in quality”)
- Maximize level of support to customer quality issues

There are 3 types of customer satisfaction criteria which drive the development/QA strategies:

- The ultra-rapidly evolving technology market. This is characterized by the fast-paced introduction of fundamental technologies which are otherwise not available to customers. It is further characterized by the introduction of successive revisions > very quickly. Customers are willing to put up with low quality to obtain this technology
- The high-end technology market. Quality is more important but not as much as new, robust features.
- The main-stream technology market. Quality and stability are of paramount importance

Before deciding to focus our attention on a particular quality or business priority, it is generally useful to ask what's important to customers. We need to know:

- What would customers value most at this specific stage of the product life cycle?
- Are there features that would influence customers’ decisions to buy or not buy the product?
- What aspects of the product do customers perceive as drivers of their success?

If you ask customers what they look for in a quality product, you'll hear comments such as these: "One that uses the latest and most appropriate technology", "Features that do special functions I need in my work", "One that regularly works the way it's supposed to", and "Decent support when I need help". It all distills to four key measurements of quality:

- **Technology**
- **Features**
- **Freedom from bugs (QA and testing), and**
- **Responsive support**

A good method for assessing our customers' priorities is the one that is incorporated in Geoffrey Moore's technology adoption life cycle model (Reference: Geoffrey Moore, “Crossing the Chasm” and “Inside the Tornado”, Harper Business Press, 1995). This model provides an excellent baseline understanding of what the market and the target users require at any point in time during the product life cycle. Current and potential classes of users each have different perceptions and priorities over the product's lifetime.

Moore's technology adoption life-cycle model

In his model, Moore describes the "technology adoption life cycle" in which he observes that just as products progress through a life cycle, customers' relationships with a given product change in a cyclical manner. Our awareness and sensitivity to these changes allows us to present our view of quality of our product in terms that meet the customer's phase needs and expectations. Figure 2 illustrates how Moore model segmented the entire potential market for a product into five parts, which make up a bell curve.

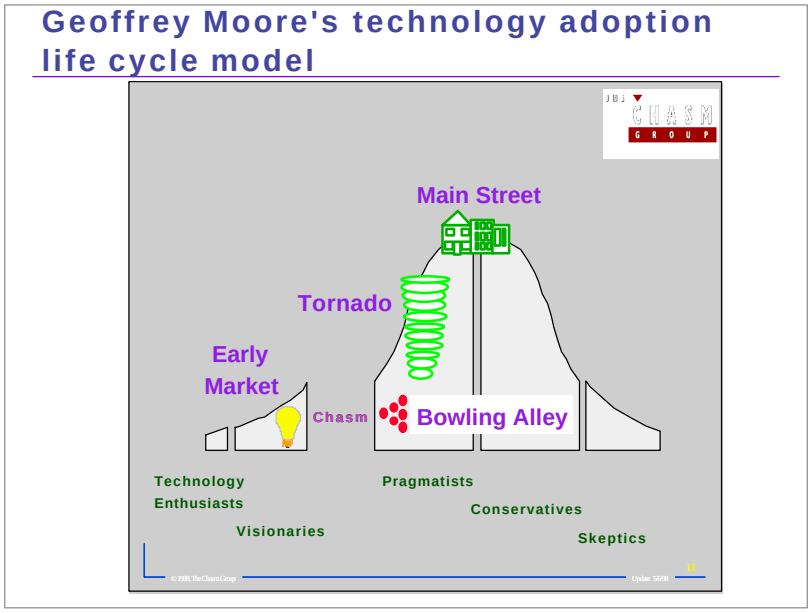

Figure 2: Geoffrey Moore Technology Adoption Life Cycle model

The Early Market – The "Early Market" is made up of two types of customers, technology enthusiasts and visionaries. The technology enthusiasts are very tolerant of bugs and usually want to be their own support. They're fascinated with the emerging technology and are eager to adopt the product simply to see how it works. To appeal to them, the technology must be very new, but it doesn't have to promise to be the next greatest thing.

Following their experimental lead, when the technology begins to sort itself out and become talked about as the next great technology, the visionaries take the leap and are willing to accept it. The focus is still on the technology, but this group of adopters trusts that there's a benefit to their business position if they adopt the product. They want it to succeed and are highly swayed by suites of features that will give them an

edge against their competitors. They're willing to put up with some problems with the product, so long as support is readily available. Note that the type of support the visionary wants is leading edge expertise on call, not ordinary customer support. He expects quality measured by traditional metrics to be awful, because it is so early, and wants experts who can “fix it on the fly.” He doesn’t care where the breakdown is - in the product or somewhere else in the value chain - he just wants it fixed.

There is often a period of calm following these early adopters. Moore calls this the Chasm, a period when it may seem all interest has evaporated. But, never fear. The visionaries are demonstrating success with the innovation and before long, one or two pragmatists who are on the visionary end of pragmatism will begin to show interest. Often, these early pragmatists come from specialized industry niches which are particularly suited to the innovation. It is important to recognize that pragmatists have diametrically opposed motives from visionaries - the former voluntarily seek disruption to get ahead, the latter only under duress to get out of being behind. The early pragmatist adopters are “pragmatists in pain” who are so far behind on a key success factor that they are willing to undergo technology adoption pain to fix it. But pragmatists *must* move as a herd so this tends to be a market segment in pain, not just a single company.

The Bowling Alley – This is a period of niche-based adoption in advance of the general marketplace. The key customers for the niche specific whole products are the “pragmatists”. Each individual niche pragmatist looks to the others to observe whether it's "safe" or appropriate to take on the new technology. So the start-up to infiltrate this market segment is likened to a "Bowling Alley" where one customer's acceptance or one specific niche-based adoption acts like the first pin in the bowling pin lineup.

The “whole product alignment” is the number one quality issue for this part of the value chain. It relates to end-to-end or systems-level testing as opposed to product or feature level testing. Maybe we can divide “bugs” into two sets - intra-product bugs, which can be lived with, and inter-operability bugs which cannot be lived with. To fix the latter does require a very high level of support, but it does not require general purpose expertise so much as domain- and niche-specific expertise. Thus a VAR (Value Added Resaler) is a better partner here, whereas a systems integrator is a better partner in the early market. Therefore, the key “quality goals” for niche-based adoption of the new product are the specific features for the niche market and the level of support in a form of business alliances, partnerships, outsourcing and collaborations with other suppliers (and with customers). These ensure that the new product is an integral part of the entire “value chain” for the specific niche market, and will allow the new product to become the niche-market dominating choice. (References: “Intelligent Business Alliances” by Larraine Segal, 1996, and ”Living on the Fault Line” by Geoffrey Moore, 2000).

The Tornado – Once the new product becomes the dominating choice a few of these early pragmatists jump on board, the community of trust gradually grows; one leads to two, two lead to four, and on and on until the rush for your product seems like a tornado.
When the tornado phase does come, it's best to be ready with plenty of product, of course, and support teams who are well prepared for rapid turn-around of bugs, staff or documentation for installation support, training for users and accurate, helpful, and easily accessed documentation. In the Tornado the number one issue is bugs because they swamp the customer support systems. Support itself should be high only in the sense that it must answer the call, not in the sense that it should be expert.

Main Street – When the product is perceived to be stable and the "new" technology has proven it's place in the market, the remaining holdouts eventually turn to your product. The true conservatives are low risk takers. Sometimes they are driven to adopt your technology simply because they can no longer get replacement parts, experienced staff, or fixes for their antiquated current product. They're not fascinated with the flashy attraction of new technology or features beyond the truly necessary. They don't have much patience for working out your quality issues for you. And since the bugs have been mostly worked out by the earlier adopters, this group doesn't rely heavily on your support resources. In Moore's model, this is the mainstream marketplace, where your product can live happily ever after, sometimes for decades, until the next greatest technology comes along to replace it. The key to Main Street is converting customer support from a cost center to a revenue center as it sells in aftermarket products and services, including eventually a major outsourcing offer as the product moves from core to context. Quality becomes a function of the *customer experience around the product* more than an attribute of the product itself.

Each phase and each user category has its own mix of priority levels for the four quality areas. These are summarized in the Table below.

Product Phase	**Early Marke**	**Bowli ng**	**Tornado**	**Main s :ream**
Dominant User → Category	**Technolog y Enthusiasts & Visionaries**	**Visionaries & Early Pragmatists**	**Pragmatists & Early Conservatives**	**Pragnatists, Conservatives & Skeptics**
Qty 1: Technology	**High +**	**Med**	**Low**	**Low**
Qty 2: Features	**Med**	**High +**	**High**	**Med**
Qty 3:QA & Testing	**Low**	**Med**	**High**	**High +**
Qty 4: Support	**Low**	**High +**	**High**	**High**

Roadmap for QA and Testing Improvements

The key goal in improving the process of quality assurance and testing activities during the software development life cycle is to increase the rate of finding and removing of software defects so that the number of defects at the product release time is getting lower and lower. This process is illustrated in Figure 3.

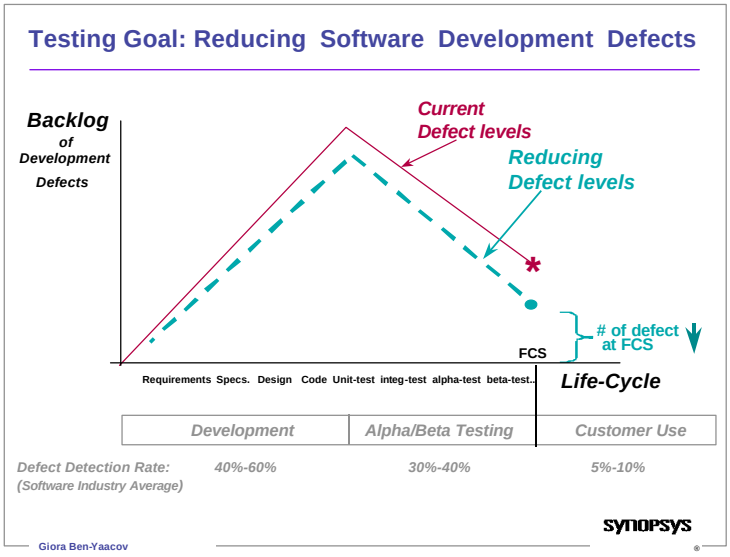

Figure 3: Key testing goal – reducing software development defects

With our current processes, we introduce defects at every development phase. As we progress with the planning, the specifications, the design and the coding phases, we continue to introduce new defects and the defect backlog continues to rise -- the solid line with up-trend in Figure 3. Similarly, with our current processes, we find (and remove) defects as we progress with the various phases of testing (unit test, system test, alpha, beta, etc.) – the solid line of backlog of defects with down-trend in Figure 3. Now, with improved QA processes (such as specs and code reviews or automated code checking tools) and with improved testing processes and techniques, we can find and remove more defects – this is illustrated by the dash line in Figure 3. The end result from improving the QA and testing processes is that the number of defect at the release time is getting lower and lower.

At our company, we adopted a roadmap for quality improvements that includes three branches: Quality build-in, Quality testing, and Quality defect management. This is illustrated in Figure 4.

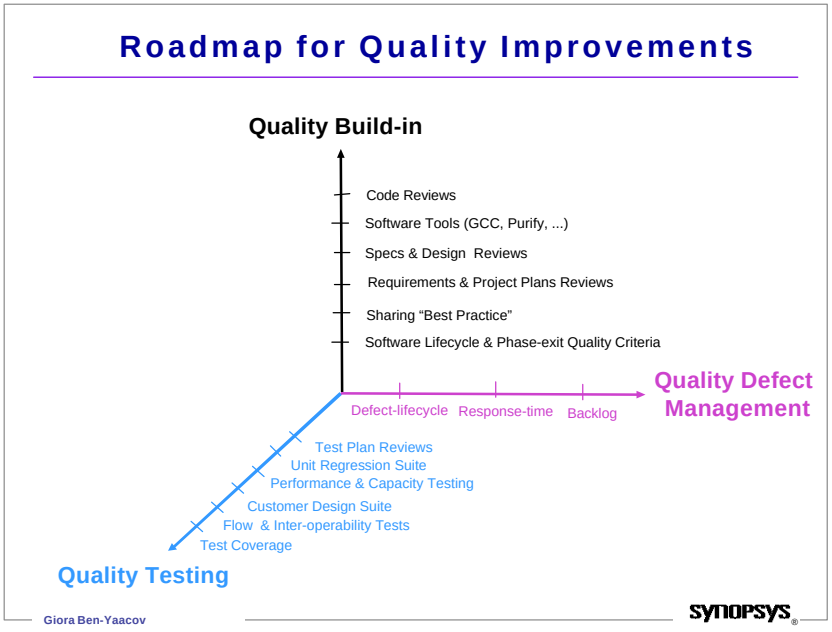

Figure 4: Roadmap for Quality Improvements

For the "Quality build-in" process, we prepared a guide " Five-Step Process for Delivering Defect-Free code":

- **Step 1**: Specifications and Design for Simplicity and Reliability
  A clean and well structured functional and design specifications simplifies the development of reliable code. Specifications reviews are important to ensure correctness, completeness and simplicity.

- **Step 2:** Use Automated Tools to Check the Code
  Use good coding style guide during coding. Use Automated checking tools (such as Purify, Lint and GCC) to find coding and memory management bugs (and warnings) that would otherwise take a lot of testing time and effort to find and fix.

- **Step 3**: Review Code (typically 2 engineers: developer+senior)
  Code reviews have consistently been shown to be cost-effective way of removing bugs from code. The process of showing and explaining a new section of code to another engineer has several positive impacts:
  -- confirms that the design is functioning as intended, exposes inefficient code, ...
  -- forces the engineer to articulate assumptions;
  -- encourages cross-training and sharing of techniques.

- **Step 4**: Create Regression Test Suites
  The most effective testing for delivering "defect-free" code is to create a fully automated regression test suite that are run after each build of the software. The tests should be designed to exercise every part of the software and produce a success/failure report.

- **Step 5** : Build and Test Daily
  Daily builds and running the regression test suite after every build give developers and integration engineers quick feedback about the changes they are making.

For the "Quality testing" process, we have deployed a test process that covers all testing activities during the software development life cycle. This test process is illustrated in Figure 5.

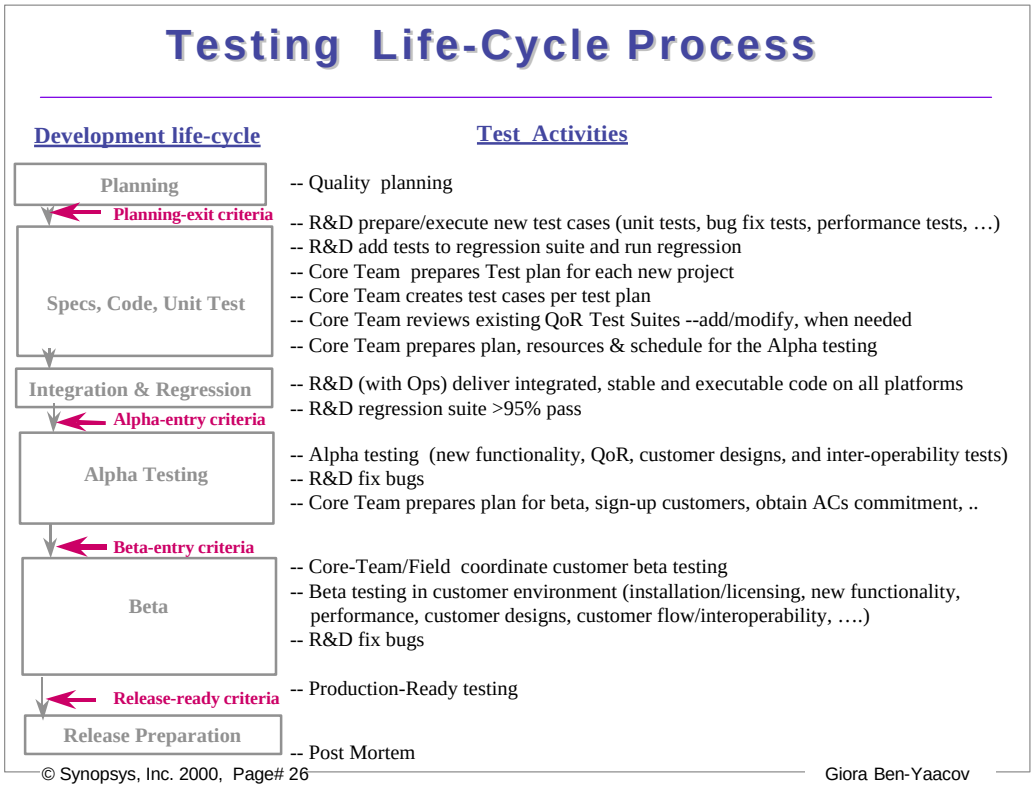

Figure 5: Testing life-cycle process

The Table below shows few of the key testing categories that are performed at our company:

Action	Description
Unit Testing	Unit testing is usually at the code level, where the developer wants to test the C-function(s) created. Some of these tests may be added to the regression.
Feature Testing	Also referred as function testing, this generally focuses on the product features. Ideally, a test should be developed for every feature in the product. However, in practice, a number of small features will undergo a single test. The development teams are expected to do all feature-level testing. The R&D team can negotiate with the testing team to do some or all feature testing. These tests are typically added to a feature regression suite. One of the criteria for product release is to ensure that 100% of feature regression passes.

Action	Description
Extended Feature	Similar to feature testing, but a few features are used together in logical order, representing a typical customer task. If, for example, a simulator is being checked, the feature testing may look at the ability to set a breakpoint, while the extended feature testing might use the ability to set breakpoints while debugging. All extended feature tests are automated and added to the regression.
Flow Testing	Flow testing addresses product-to-product interactions and is based on information from customer use models and flow documents. Flow testing tries to verify that the customer can pass data from one tool to another in the given use model. It would be ideal to automate these tests; however, operational/tool issues usually prevent this. As a result, flow tests remain mostly manual in the regression.
Regression Testing	As new test cases are created, they are automated and added to regression so that they can be executed for every release. This ensures that adding features or fixing bugs has not interfered with the rest of the product. The regressions can include any type of testing.
Performance Testing	The product performance is compared against the previous versions and against the stated goals of performance improvements. Performance measurements include speed in completing the task, memory consumption (swap), and accuracy of results.
Stress Testing	Stress testing (also known as capacity testing), subjects the product against known boundary conditions or large customer designs with the objective of characterizing the product stress levels.

CONCLUDING REMARKS

This paper describes a methodology for allocating priority levels and resources to software testing and other quality activities which are based on understanding of what the market and the target users require at any point in time during the product technology adoption life-cycle model.

The paper also describes the deployment by a leading market-driven company of effective software testing processes and methods that represent real-world customer issues.

ABOUT THE AUTHORS

Synopsys, Inc. provides comprehensive software tools and technology for the product development requirements of the world's leading electronics companies. Synopsys is the largest supplier of software tools used to accelerate and manage the design of semiconductors, computer systems, networking and telecommunications equipment, consumer electronics, and a variety of other electronic-based products. With nearly 3,000 employees and 1999 annual sales of over $800 million, Synopsys has software development and research facilities around the world. The company is headquartered in Mountain view, California

Giora Ben-Yaacov
Giora's 30 years of experience in software quality improvement programs, software development and testing, program management, and customer support include a spectrum of results ranging from that of individual contributor, through project leader, program manager, instructor, consultant, and, since August 1998, a Quality Architect with Synopsys Inc.. Giora is a senior member of IEEE, a member of the American Society for Quality, and has served for 6 years on the editorial board of the IEEE CAP journal. He has authored and presented more than 35 technical papers.

Lee Gazlay
Lee has 25 years of R&D development and management experience in the EDA industry. At Synopsys, Lee is in charge of the Software Engineering organization which is responsible for building, testing, porting and releasing of Synopsys products.

Giora Ben-Yaacov

Giora's 30 years of experience in software quality improvement programs, software development and testing, program management, and customer support include a spectrum of results ranging from that of individual contributor, through project leader, program manager, instructor, consultant, and, since August 1998, a Quality Architect with Synopsys Inc.. Giora is a senior member of IEEE, a member of the American Society for Quality, and has served for 6 years on the editorial board of the IEEE CAP journal. He has authored and presented more than 35 technical papers.

Lee Gazlay

Lee has 25 years of R&D development and management experience in the EDA industry. At Synopsys, Lee is in charge of the Software Engineering organization which is responsible for building, testing, porting and releasing of Synopsys products.