Build Quality In – Don't Test It In

By Kathy Iberle

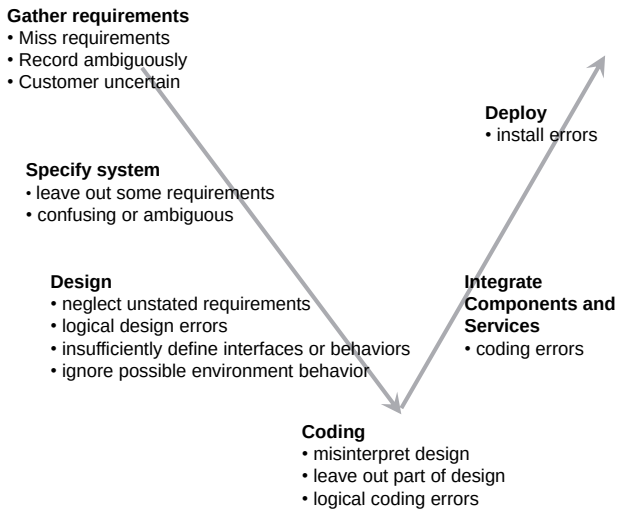
Key Points

- **Defects are created at many points during the development process.**
- **Quality can be improved permanently by blocking the most common sources of defects.**
- **Blocking the sources which are most common in *your* organization is less expensive and more effective than following a one-size-fits-all quality strategy.**

Where Do Defects Come From?

Whether you're doing agile, waterfall, or something in between, you still perform the same key activities during development:

- Gather user requirements (determine what the customer wishes the product would do)
- Specify the product (decide what the product shall do)
- Design the software (figure out how the product will do those things)
- Implement the design (write the code)
- Integrate the various pieces together.
- Deploy the software (install and/or configure onto its final host).



An agile team may push a few user stories through this sequence in just a week or two with minimal documentation, while a waterfall organization might move a large set of functionality through in several months, but it's still the same sequence.

An individual defect is created when something is left out or done incorrectly during one of these activities. Each activity is prone to characteristic mistakes, as shown in the diagram. For instance, failure to initialize variables is a mistake made during coding, whereas misunderstanding the user's needs is a mistake made during requirements gathering. The design stage is especially prone to errors of omission, such as failing to check for external conditions.

All organizations make most of these mistakes, ***but not in the same proportions***. Thus, the most efficient methods of quality control are those which address the patterns of errors characteristic to ***your*** organization.

Identify Your Most Expensive Defects

If you follow someone else's laundry list of quality-control practices, you won't be focusing on your own unique problems. You may find that you're investing a lot of time but not getting a great deal of improvement.

It's much more satisfying to go after the defects which have the most effect on your organization. That's the failures which cost you the most – in development time, in support costs, in reputation. You might consider starting with a set of problems that have an obvious cost, such as:

- Failures which required maintenance releases or patches.
- Failures which forced a project to miss its planned release date.
- Failures which routinely eat up a lot of development time.

Track Your Expensive Defects Back to Their Source

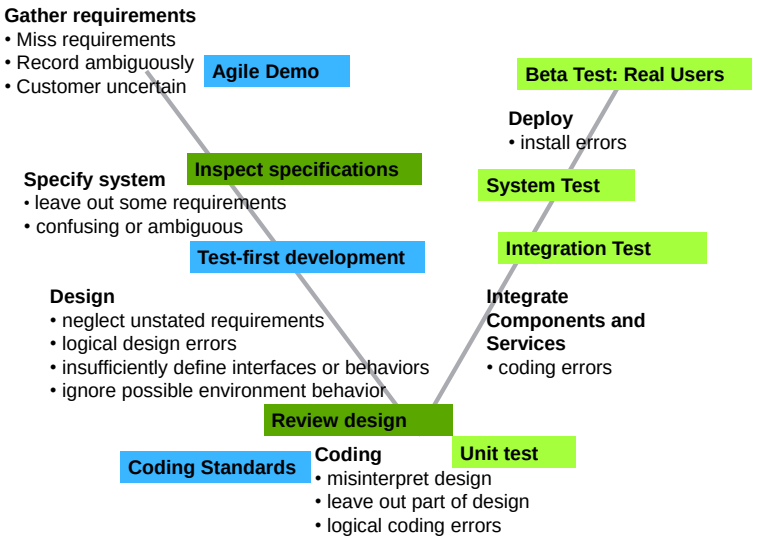
Next, figure out why that class of defects is appearing. During which activity were they inserted? This is a development activity, not a test activity – the testers cannot read the developers' minds to understand what they were thinking at each stage. A developer's initial gut-level response to the failure report often provides a clue:

- "It shouldn't be doing *that*." - The code is not doing what the developer intended, suggesting an error in the code or the design.
- "It works as designed." The test team and the developer don't agree on what the user's requirements really are.
- "But users won't do *that*." Another disagreement – the test case may be wrong.
- "I'm not sure if that's ok." – the specification may be ambiguous
- "I didn't know anyone wanted that." – a missing requirement

Think of the defects as leaking into your product through holes in these development activities. Your defects will probably be leaking through in several different spots. You'll want to address the holes which are letting in the most issues.

Plug the Hole in Your Process

Now that you know where defects are getting into your product, you'll want to block those holes. Knowing which activity is leaking defects will help you choose an appropriate prevention or detection method. All quality-control methods aren't created equal: each one is most effective against defects spawned during a particular activity, as shown to the right. (Detection methods are shown in green and prevention methods in blue.) There's a longer list of these methods in my paper "Building Quality In" (see references).



Prevention methods generally require the most thought to set up, but they're often the most effective and the least expensive in the long run. The starting point for prevention is to ask "Why?" Why is this error made in the first place? How would it be possible to prevent it? This is similar to "mistake-proofing" in Lean Manufacturing. Techniques can be borrowed from that field – checklists, work procedures, and appropriate tools come to mind. Additional training is often effective.

Rinse and Repeat

No process is ever perfect. Defects will continue to occur, but they can be driven to amazingly low levels by systematically and repeatedly applying the sequence of activities I've just described. Focusing on your unique process within your organization allows you to identify your characteristic leaks and plug them with the most effective and least expensive methods.

Learn More

- ❑ *Corrective Action for the Software Industry*; Johanna Rothman; 2011.
- ❑ "Building Quality In"; Kathy Iberle; Proceedings of the Pacific Northwest Software Quality Conference 2007; on www.kiberle.com. This includes a list of various prevention and detection methods.