

Triage Regression Test Guide

Regression Testing is defined as the testing of functions or components that were not intentionally modified to ensure that changes made elsewhere have not caused problems with other code. By some estimates, about 6% of individual code changes made in an application will cause damage elsewhere in the application. Regression testing focuses on finding that damage.

The problem is that as applications grow in size and complexity, it is impractical to retest every function of the application with every new release. Instead, risk management decisions will need to be made to determine where to focus attention. The *Triage Regression Testing Guide* is intended to provide that guidance.

Start by focusing on three key areas:

1. **High Volume Activities** (“Do a Lot”)

High volume activities are defined as those that produce the highest transactional volume for an application. For example, if state specific functions are being tested (e.g., sales tax calculation), start by testing the states with the highest volume of activity.

If you don’t have time to retest every state every time, at least focus on the largest states. Given the “80/20” rule, you’ll get your highest benefit using this approach. Don’t just limit this approach to states. Focus also on the highest selling products, the most popular options, etc.

Information Sources

What if you don’t know what the high volume activities are?

- Your business customers are the place to start. They should know where their users spend most of their time.
- Talk to the Marketing Department. They may not be seen as a traditional source of information for QA, but in most companies they are the keepers of information on what customers are buying the most.

2. **High Defect Potential** (“Break a Lot”)

High defect potential scripts test areas of the system that are considered complex or problematic. These are often not the same as the areas with the highest volume. In fact in some cases, the high volume areas are the least problematic because the problems have worked out. High defect potential areas are often areas of unusual complexity.

Information Sources

The best way to identify these areas is to review your metrics on defects, including the “bug log” and see what is coming up most often.

You can use Microsoft Word’s indexing feature to do this. Select key words in a document, like the names of the states, and MS Word will build an index of every page where they came up (e.g., *Connecticut* or *finance charge*). Look at the words that come up most often in the list. If you find names of certain states, products, screens, etc. coming up frequently, that is an indicator of potential problem areas.

Other areas for information on high defect potential are the Help Desk, as well as the QA and Development Teams

Finally, check out your company’s metrics on defects. If your metrics include Root Cause Analysis, check that information as well.

3. **High Impact** (“Hurt a Lot”)

Scripts that test functions critical to business, where the consequences of errors are very high, fall into this category. These are typically items that don’t occur frequently but when they do occur, the “pain” level is high.

Examples of high impact defects would include:

- a. password or security code failures that allow unauthorized access to confidential information
- b. the potential for loss or corruption of critical data, such as when a server crashes or a backup copy has to be installed
- c. significant pricing errors, such as the miscalculation of an APR on a mortgage

Information Sources

The sources of information on these catastrophic problems are similar to the high defect potential category. It’s easy to decide to forgo testing of these items if no problems have occurred recently, but given the potential impact, these items should get regular testing.

Still Have Time?

These three categories (high activity, high defects and high impact) should be the areas of focus when risk management decisions have to be made. If all three can be covered with resources still available, give attention to the “medium” level items in these categories as well.

Take a look at the new features being introduced in the release being tested. If you see new changes that look like they may be adding new high volume activities, that appear complex or have a high impact if they fail, add them to the list, too.

Since these features are new, they don't fit within the definition of a regression test, but there's no reason to wait until a problem arises. When it comes to testing, it's better to take the approach that a systems change that might be problematic is "guilty until proven innocent!"

Review those new features and determine what changes need to be made to the standard regression test bed. That test bed should be updated after each version, so that it tests your application as thoroughly as possible, within the constraints of your available resources.

Finally, make sure that you're keeping adequate metrics to help you understand the problem areas in your application. Using your metrics, can you determine what types of bugs are coming up most often, and what components of the application are most problematic? If not, it's time to update your metrics. The more you know about the likely trouble spots, the better equipped you are going to be to use your testing resources where they're needed most.

The Triage Approach:

Focus on components that are likely to fall into these categories

