Salesforce Test Dependency Analysis

Matt Buland, Lead Platform Developer
Salesforce.com

About Salesforce

Salesforce is the innovative company behind the worlds #1 CRM solution. Our software is cloud-based, so it doesnt require a team of IT experts to set up or manage you just log in and start using it.

The Project

With a large code-base comes complexity in both use-cases and code. Particularly at Salesforce, the number of customizations present a huge surface area of functionality. To ensure trust and reliability in our product, we need to validate changes against expected behaviors (read: test our software). Throughout the years, we've accumulated a test-suite that would take a full week to run in entirety; certainly not something that scales to thousands of changes per day. In theory, we can use intelligent dependency analysis of code & historical issues to build up a tight and representative suite of tests around a particular area.

GRAND VISION

We have a variety of sources of information that we'd like to bridge together:

- Code
- Test Plans (spreadsheet of test-cases)
    - Automated tests (JUnit, Selenium, Aura-cmp, Jest)
    - Manual tests (tracked in spreadsheets)
- Bug / Story

The key question is what the impacts of a change will be, and what should be tested.
Change → Code → Previous Changes → Stories/Bugs with Test Cases → Suite of relevant tests

Previously, I put together a very manual and simplistic form of this, where each code/file is annotated with relevant test-cases, and a manual tree of code dependencies, just for our team, which isn't scalable at all. There's a lot of automation potential that can help drive us towards a complete solution.

CODE ANALYSIS

The biggest gap we have is code-analysis of our Aura (Javascript) code-base
(see: https://trailhead.salesforce.com/en/content/learn/modules/lex_dev_lc_basics) The patterns present in this code-base are inconsistent and fairly disparate, making it challenging to analyze. The first requirement is to be able to (weakly or heuristically) understand relationships between functions / code between Aura components and libraries. This may pique your interest if you want to learn more about compilers, parsers, and graphs.

TEST-CASE LINKING

For a particular piece of code, we need to know which test-cases are relevant, which comes from many sources: unit tests, integration tests, and manual tests, each in completely different domains. The only unifying truth we have across all of these is our agile management tool, which keeps a record of Bugs & Stories, changes made under the work, a description of an issue, and discussion related to the work (where a test plan would be posted). Until code analysis is

done, there won't be a ton of value here, but eventually we'll need to build a map of Code ↔ Bug/Story ↔ Tests. This may pique your interest if you want to learn more about web APIs.

Relevant Technologies

Code analysis: Javascript (solution could be done in JS, Java, or Python)
Test-case linking: SOAP or REST APIs + whichever platform code-analysis is done in

Team Composition

Recommended team size: 3-5 persons
Each team member would be asked to sign a NDA with Salesforce
Work can be performed anywhere, but we will at least give a tour of the Salesforce office in Louisville
Preferred communications are over email, G-Chat, and/or hangout video conferences.