Using requirements specification to speed up STPA-BDD in agile development

Amsterdam, 2018-11-02

Stefan Wagner
You can

(copy, share and change,
film and photograph,
blog, live-blog and tweet)

this presentation given that you attribute it to its author and respect the rights and licences of its parts.

based on slides by @SMEasterbrook und @ethanwhite
Agile Software Development

Product Owner

Stakeholder liaison

Development Team

Sprint Planning

Team forecasts work needed to achieve Sprint Goal

Sprint Backlog

PBI's

Product Backlog

Iterative-Incremental Development & Delivery

Sprint

(max 1 month)

Scrum Master

Daily Scrum

Sprint Review

Sprint Retrospective

Potentially Releasable Increment
Agile Software Development of Safety-Critical Systems?

Safety analysis without an upfront architecture design?

Unstable requirements that can change every few weeks?
There are some approaches – S-Scrum

https://arxiv.org/abs/1703.05375
Focus on communication

We are uncovering better ways of working through doing it and helping others to do it. Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Acceptance-test-driven development

1. Select a user story
2. Identify conditions of satisfaction
3. Implement acceptance test(s)
4. Test
5. Code
6. Refactor
7. Failing acceptance tests
8. Refactor
9. Test
10. Code
11. Refactor
12. Passing acceptance test
13. Refactor the test
14. Customer acceptance

Based on: M. Cohn. Succeeding with Agile. Addison-Wesley, 2010
Behaviour-Driven Development (BDD)

- Developer
- Tester
- Product Owner

Examples → Scenarios → Automated Tests
Feature: Refund item

Scenario: Jeff returns a faulty microwave
  Given Jeff has bought a microwave for $100
  And he has a receipt
  When he returns the microwave
  Then Jeff should be refunded $100
STPA-BDD

<table>
<thead>
<tr>
<th>Input</th>
<th>UCA</th>
<th>Process variables and algorithms</th>
</tr>
</thead>
</table>

- Select UCA *(manually)*
- Write test scenarios *(manually)*
- Generate test cases *(manually)*
- Write test codes *(manually)*
- Run test suites *(automatically)*
Example

Unsafe Scenario from STPA

During auto-parking, the autonomous vehicle does not stop immediately when there is an obstacle up front.

Gherkin Scenario

**Given** the autonomous vehicle is auto-parking

**When** the ultrasonic sensor provides the feedback that the forward distance is smaller or equal to a threshold indicating that there is an obstacle up front

**Then** the autonomous vehicle stops immediately.
Experimental results

But: Communication effectiveness is significantly different!

The developers consider the safety requirements deeply and initiatively. The business analysts are more confident about the test cases. It becomes easier to identify conflicts in business rules and test cases. The business analysts are clear about the status of acceptance testing. The business analysts could spend less time on sprint-end acceptance tests.
## Speeding it up with automation

<table>
<thead>
<tr>
<th>Input</th>
<th>UCA</th>
<th>Process variables and algorithms</th>
</tr>
</thead>
</table>

1. Select UCA *(manually)*
2. Generate test scenarios *(automatically)*
3. Revise test scenarios *(manually)*
4. Generate test cases *(automatically)*
5. Write test codes *(manually)*
6. Run test suites *(automatically)*
### Figure 4: Boxplot for PROD (NIUS), THOR (LC) and FAUL (MSI)

### Table 3: Hypothesis Testing

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Hypothesis</th>
<th>Testing</th>
<th>( p )-value</th>
<th>( Z )</th>
<th>ANOVA ( F )</th>
<th>Effect Size ( d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDD and BDD-R NIUS</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.907</td>
<td>0</td>
<td>72.918</td>
<td>3.817</td>
</tr>
<tr>
<td>BDD and BDD-R LC</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.021</td>
<td>3</td>
<td>10.595</td>
<td>1.459</td>
</tr>
<tr>
<td>BDD and BDD-R MSI</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.021</td>
<td>1</td>
<td>18.619</td>
<td>1.924</td>
</tr>
<tr>
<td>UAT and BDD-R NIUS</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.940</td>
<td>0</td>
<td>70.512</td>
<td>3.761</td>
</tr>
<tr>
<td>UAT and BDD-R LC</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.546</td>
<td>0</td>
<td>29.270</td>
<td>2.422</td>
</tr>
<tr>
<td>UAT and BDD-R MSI</td>
<td>( \mu ) &lt; 0.05</td>
<td>Wilcoxon</td>
<td>-3.415</td>
<td>1.5</td>
<td>19.672</td>
<td>1.971</td>
</tr>
</tbody>
</table>

### Conclusion

First, concerning the violated assumptions of statistical tests, the sample size in our replicated experiment is 33. Thus, the Mann-Whitney \( U \)-test is robust. We use Wilcoxon \( W \) and \( Z \) to enhance the robustness. Yet, under certain conditions, non-parametric rank-based tests can themselves lack robustness [9].

Second, concerning random heterogeneity, we select the new 11 participants, who have the same Java programming knowledge background with the 22 participants in Group A1 and Group B1. We check it by pre-questionnaires. Although they are from two courses, 95.5% of them are in the same major.

### 6.4 External Validity

First, the same as our original experiment, the subjects are still students in our replicated experiment. Although some of them can perform as well as experts, most of them are not professional. This may limit the generalisation of our results. Referring to [6], before running the experiment with professionals, a high sample size should be guaranteed. A long learning cycle of STPA for safety analysis and the new utilisation of BDD for safety verification lead us to believe that using students as subjects is a suitable way to aggregate contributions at this moment. These two articles [3][17] also indicate the possibility of using students to run SCS projects.

Second, the sample system is relatively simple. We consider that the participants are students together with a short time learning cycle. Thus, the difficulty of tasks is in accordance with their capability. This setting can not represent a real-world project.

### 7 DISCUSSION AND CONCLUSION

We conduct this replicated experiment building up on our original experiment, which showed that STPA-BDD is a possible way to perform safety analysis and verification in agile development with a good capability of communication effectiveness. In this article, we aim to speed up BDD for safety verification through developing a semi-automated tool. We evaluate it by replicating our original experiment with 11 new participants (33 subjects overall). We have achieved positive results. By comparing three groups, Group A1 using original BDD, Group B1 using original UAT and Group A1-R using BDD with the semi-automated tool, the productivity, the test thoroughness and the fault detection effectiveness show great improvements. In our original experiment, these three measures showed no statistically significant difference by only comparing BDD and UAT.

In terms of productivity, first, STPA safety analysis results can be automatically generated into test scenarios. Second, developers can construct the hierarchy and name test cases by clicking only one button to trigger the semi-automated tool. Even though the generated test scenarios need human decision makers (3 amigos) to revise and the test code needs human beings (developers) to write. It saves much more time than generating test suites totally manually. More specifically, this semi-automated tool poses a small step of realising continuous test automation [2] when using STPA safety analysis and BDD safety verification in agile development.

In terms of test thoroughness and fault detection effectiveness, the automatically generated test scenarios provide more comprehensive test sets based on the provided UCA (safety requirements), process
Putting a formal basis below it

Input
Context or unsafe scenarios

Generate test scenarios *(automatically)*

Generate test cases *(automatically)*

Write test codes *(manually)*

Run test suites *(automatically)*
Will we lose communication?

Developer  Tester  Product Owner
Additional STPA rather than mathematically creating a formalism of test cases.

In practice, we can see that we are able to join STPA with other tools, not. In this context, we will demonstrate how to support each other.

In diagram 1, we can see that there are two approaches. The first approach is manual and the second approach is automated.

Manual approach:
- Input: Context or unsafe scenarios
- Select UCA (manually)
- Generate test scenarios (manually)
- Generate test cases (manually)
- Write test codes (manually)
- Run test suites (automatically)

Automated approach:
- Input: Context or unsafe scenarios
- Select UCA (manually)
- Generate test scenarios (automatically)
- Revise test scenarios (manually)
- Generate test cases (automatically)
- Write test codes (manually)
- Run test suites (automatically)

In the following, we consider the case of Thomas's process where the model is used to generate a table of test cases.

The number of variables is also shown in the middle of Fig. 1, to demonstrate the execution of the requirements.

In this way, we can see that the tool, which is: a tool, is used by experts during the test execution, which is: a test.

We can also see that the context is: a context.

In conclusion, we can see that the approach, which is: a test, is done for the safety of our systems.
Joint work with Yang Wang (now at Bosch) and John Thomas (MIT)

Prof. Dr. Stefan Wagner

- e-mail: stefan.wagner@informatik.uni-stuttgart.de
- phone: +49 (0) 711 685-88455
- WWW: www.iste.uni-stuttgart.de/se
- Twitter: prof_wagnerst
- ORCID: 0000-0002-5256-8429

These slides are available at www.stefan-wagner.biz
Pictures used in this slide deck

Safety by GotCredit (https://flic.kr/p/qHCmfo, Got Credit)
Scrum framework by Dr ian mitchell under CC BY-SA 4.0 (https://en.wikipedia.org/wiki/Scrum_(software_development)#/media/File:Scrum_Framework.png)
Screenshot from http://agilemanifesto.org by Ward Cunningham