 Embedding STAMP in Teaching Software Testing

Natalia Silvis-Cividjian, Tanjina Islam, Miguel Morales-Exposito, Abel Pelser, Abinav Shankar and Robert Zondervan
Outline

• Rationale
• Using CAST
• Using STPA
• Conclusions and future work
Ideally

A software product

=  

Vision/idea/needs/expectations

However...
Bizar softwareprobleem op hogesnelheidslijn tussen Amsterdam en Antwerpen: trein stopt soms doodleuk tijdens de rit

ADN  |  31 oktober 2018  |  17u54  |  Bron: Belga

Deel  |  5  |  Reacties  |  10

European STAMP Workshop 2018
Amsterdam
Toyota's killer firmware: Bad design and its consequences

Michael Dunn - October 28, 2013

How insulin pumps could give a fatal overdose to diabetics: The 'foolproof' alternative to daily injections

- Shayla Walmsley, 44, is suspected to have died from an insulin overdose
- She was using an insulin pump which her family believe was malfunctioning

By LYNNE WALLIS
PUBLISHED: 22:00 GMT, 14 December 2013 | UPDATED: 22:00 GMT, 14 December 2013
• Our society relies more and more on software
• Therefore, software testing (ST) is of paramount importance
• However, standalone courses on ST are rare.
Software testing compares **what it is** with **what it ought to be**.
Traditional courses cover only unit testing and functional testing.
A software testing course @VU

• Graduate Computer Science students
• Core module: 50 students
• Systems testing project: 20 students

What makes it different?

• Standalone course
• Extends to system testing
• Uses physical models of critical systems
• Analyzes software related accidents
• Extends to testing of non-functional requirements, such as safety
A robotic vehicle –
(mimics an autonomous car)
A water salinity & temperature controlled plant (mimics an insulin pump)
Functional testing

• FUNC_REQ: When salinity is higher than 5ppt, the fresh water pump should start working.

• FUNC_TEST: Start from a solution of 3ppt. Add salt until the sensor reads 5 ppt and check if the fresh water pump starts working.
A model railroad (mimics a train management system)
Rationale

• STAMP is a rising star that promises to better understand accidents and find new hazards
• All this makes it suitable for our course
• We decided to adopt STAMP along with traditional methods such as event chain causality model, FTA and FMEA.
Outline

• Rationale
• **Using CAST**
• Using STPA
• Conclusions and future work
CAST-assignment

• Assignment: First steps in a CAST of famous software related failures.
• Task description. Draw the high-level control structure. Identify the constraint that has been violated/missing.
• Rationale: To make testers aware of the risk of not properly tested software. To learn about possible hazards.
• Work load: 12 hours
AT&T Says Failure Was Software Flaw

By BLOOMBERG NEWS  APRIL 23, 1998

Mars climate Orbiter loss

Patriot Missile failure

Therac-25 accidents

Ariane 5 explosion 1996
Control structure for Mars Orbiter

Koen Kahlman  Jasper Veltman  Glenn Visser (VU)
Control structure for the Patriot missile accident

5 controllers

Hasine Efetürk - Yasmina Kada - Kim van Putten - Sesegma Sanzhieva (VU)
Control structure for Intel FDIV error
Evaluation of CAST

• Easy to analyze at high-level
• Students can easily find interesting problems in mental model, communication, missing feedback.
• Students have difficulties in drawing a correct control structure. They easily draw a wrong structure as a flow diagram.
• They do not know at what level of detail to start.
Outline

• Rationale
• Using CAST
• Using STPA
• Conclusions and future work
• SAFE_REQ: If the salinity sensor reads zero for 10 seconds, then the system should issue an alarm
• SAFE_TEST: Unplug the sensor and check that the system issues an alarm
STPA-assignment

• We use STPA for the train management system and for the water plant as an alternative method along with FMEA to perform a hazard analysis.

• Hazard analysis generates safety related requirements
Accident
head-on collision
Testing for safety

• **Hazard.** Switch does not move when the software command is given. Control action not executed.

• **Test scenario.** Manually change the switch in the wrong position and produce a dangerous situation (imminent collision). Check whether the system shuts down.
Demo test scenario

Authors: Lars Cordewener, Ramon de Winder and Roy Overbeek (VU)
STPA-Experiment

- STPA with 5 students and a teacher performed on the model train management system.
- We evaluated each step.
- We compared the results with the FMEA performed by almost the same students 2 months before.
Level D2

- 1.1 Turn on
- 1.2 Turn off

Drives
- 2.1a Drive
- 2.1b Reversed
- 2.2 Stop

Control Unit
- Sends Commands
- Sends Move Commands
- Move manually
- Sends Position Data
- Visual status
- Changes

Human
- Operates panel
- Status info

Actuators

Sensors

Trains

Rail Track Layout

Influences

Switch decoder

European STAMP Workshop 2018
Amsterdam
Step 0. Building the control structures

• Step 0 is difficult. We had to redraw the structure a few times. Who is a controller and who is controlled process? For example train drives on the rails. What is a control action, what is a feedback channel?

• Which level of detail is good for a start? Both trains in one controller? When to split them?

• Difficult to find good verbs to describe a control action. Words don’t come easy. Controls or sends commands or operates panel

• Easy to find missing feedback
Step 1. UCAs

- The table with keywords is helpful.
- We were still uncertain on how to model the system with control structures. We went back to Step 0 a few times. Example. First we had drive – reverse - speed up - stop then changed to drive and break.
- Our conclusion: If you don’t find any UCA for a certain control action then this control action is useless.
- We could not get enough support from STAMP for the UCA of type wrong control action. We learned from the project. Domain knowledge helps to find UCAs. Mr. Bean approach helps.
Step2. Causal scenarios and test cases

• Interesting HCI problems related to the labeling of the front panel
• We find many component interaction type of hazards involving software. Also out-of-sequence type of hazards.
• Many hazards are duplicated more than expected
• Many hazards have the same scenario
• Some hazards have so many scenarios, how do we know that we are exhaustive?
• The results depend on the domain knowledge.
Metrics

- STEP0. Accidents + control structures: 1.5 h
- STEP1. UCAs. Found 34 UCAs. : 1 h
- STEP2. Causal scenarios, constraints and test cases: 2h

- Group 1 FMEA found 22 hazards
- Group 2 FMEA found 30 hazards
- Both groups took ca 4-5 h
STPA / FMEA results

• Quite similar qua workload
• FMEA is more easy and straightforward.
• STPA is good for detecting hazards based on component interaction and wrong time sequence, whereas FMEA worked well for hazards resulting from individual component failures.
• STAMP is better when the implementation is not known or not ready.
• Both methods are valuable.
Conclusion

• STAMP can be used in many places while teaching software testing
• On small software intensive systems, the method performs similar to FMEA qua workload, but finds different types of hazards.
Future plans

• We will continue to train students in STAMP along with FMEA
• New comparison experiments
• Human engineering for accidents analysis
• Use STAMP for security, ethics
• Promote STAMP in testing communities
Acknowledgements

- Jaap van Ekris Delta Pi
- Mechanical Engineering and Electrical Systems Engineering groups at Vrije Universiteit
- Students and TAs
Thank you!