Automated testing of an X-Ray medical device

Bryan Bakker

2nd UCAAT
User Conference on Advanced Automated Testing

September 2014

bryan.bakker@sioux.eu

@Bryan_Bakker
Contents

- Sioux
- Intro – The need for action
- First increment – First success
- Next increment – Logfile interpretation
- ROI
- Results
About Bryan Bakker

- Test Expert
- Certifications: ISTQB, TMap, Prince2
- Member of ISTQB Expert Level on Test Automation
- Tutor of several test related courses
- Domains: medical systems, professional security systems, semi-industry, electron microscopy
- Specialties: test automation, integration testing, design for testability, reliability testing
About Sioux
Intro – The need for action

Medical Surgery Device:
- X-ray exposure + acquisition during surgery activities
- Real-time image chain
- Mobile device (frequently off/on)
- Quality and testing considered important in organization

Reliability was an issue:
- “Frequent” startup failures
- Aborted acquisitions
- Always safe… but not reliable!
Hardware interfaces used to invoke actions on SUT
- Buttons on different keyboards
- Handswitches
- Footswitches
- Different power-switches
LabVIEW generates hardware signals
Test cases defined in LabVIEW
Only logfiles stored, no other verification performed
No software changes needed for this approach
First increment – First success

- Simple, but quick first results
- Multiple reliability issues found
- Work to do for the developers
Logfile scanned during test case execution
Determine pass/fail criteria
Detect system states and act upon:
- Hot generator → extensive acquisition not possible
- Execute other test cases (e.g. power-cycle), until
- Generator has cooled down

Log file analysis after test was still performed
Still no software changes in the SUT, but existing interfaces were available now
Next increment
Logfile interpretation

- System Under Test
- Hardware Abstraction Layer (LabVIEW)
- Test Execution Environment incl. test cases and library (Ruby)
- Test Scheduler (Ruby)
- Test Framework 3rd Increment

Repositories:
- Repository (Test cases + Results)

Input (hardware & software) ➔ Test Execution Environment

Control ➔ Test Scheduler

Result ➔ Repository

Output ➔ System Under Test

© Sioux Embedded Systems | Confidential | 9
>100 reliability hits identified
  - Which ones would have slipped through other tests?
  - Which ones would the customer complain about?

“Independent” analysis of hits:
  - 8 would have been in system test, but not earlier
  - 7 would not have been found, but customer would complain (and fix would be necessary)
ROI:
\[(8 \times X_1) + (7 \times X_2) - \text{costs} > 0\]

Costs (manhours + material) = 200K Euro

\[X_1: \text{costs of defect found in system test: 10K Euro}\]
\[X_2: \text{costs of field defect: 200K Euro}\]

\[80K + 1.4M - 200K \rightarrow 1.2M \text{ Euro saved}\]

More money and time became available…

→ Implementing/executing more tests
→ More projects/products
Results

- Numerous reliability hits identified + solved
- MTBF measured and predicted
- More testing hours on systems
- Customer satisfaction
- More projects wanted this approach
- Only 5 system test cycles remaining (was 15)
This case study is described in detail:

Dorothy Graham & Mark Fewster
Experiences of Test Automation
Case studies of software test automation
ISBN 0321754069