Principles of Software Construction: Objects, Design, and Concurrency

Toward software engineering in practice

Charlie Garrod       Chris Timperley
Administrivia

- Homework 5c due tonight!
Software Engineering (SE) at CMU

• 17-214: Code-level design
  – Extensibility, reuse, concurrency, functional correctness
• 17-313: Human aspects of software development
  – Requirements, teamwork, scalability, security, scheduling, costs, risks, business models
• 17-413 Practicum, 17-415 Seminar, Internship
• Various courses on requirements, architecture, software analysis, SE for startups, etc.
• SE Minor: http://isri.cmu.edu/education/undergrad
Major topics in 17-313 (Foundations of SE)

- Process considerations for software development
- Requirements elicitation, documentation, and evaluation
- Design for quality attributes
- Strategies for quality assurance
- Empirical methods in software engineering
- Time and team management
- Software engineering meets machine learning
- Economics of software development
Today: Software engineering in practice

• Software engineering for robotics
• Software testing for robotics
• Robot Operating System
Robotic systems are an increasingly important part of our lives
How a Self-Driving Uber Killed a Pedestrian in Arizona

A woman was struck and killed on Sunday night by an autonomous car operated by Uber in Tempe, Ariz. It was believed to be the first pedestrian death associated with self-driving technology.

What We Know About the Accident

Body seen in this area

Elaine Herzberg was struck while walking her bike across the street somewhere in this area.

The self-driving Uber was traveling north at about 40 m.p.h.

An Uber self-driving test vehicle that hit and killed a woman in 2018 had software problems, according to US safety investigators.

Elaine Herzberg, 49, was hit by the car as she was crossing a road in Tempe, Arizona.

The US National Transportation Safety Board (NTSB) found the car failed to identify her properly as a pedestrian.

The detailed findings raised a series of safety issues but did not determine the probable cause of the accident.

A year after the first 737 Max crash, it's unclear when the plane will fly again

Two crashes of Boeing’s 737 Max 8 killed 346 people, and authorities are blaming Boeing's design, a faulty sensor and airline staff. Plus: Everything you need to know about the plane.

How the Boeing 737 Max Disaster Looks to a Software Developer

Design shortcuts meant to make a new plane seem like an old, familiar one are to blame

By Gregory Travis

The views expressed here are solely those of the author and do not represent positions of IEEE Spectrum or the IEEE.

It remains the mystery at the heart of the crisis: how a company renowned for making seemingly basic software that could have prevented deadly crashes. Longtime Boeing employee Mónica Paredes was complicated by a push to outsource to lower-paid temporary workers and contractors.

The Max software -- plagued by bugs and design flaws, and cases of planes grounded months longer than necessary -- was the result of a company that was laying off experienced engineers wasn’t interested in cutting costs.

How would you develop software for a delivery robot?

- What are the requirements of your system?
- Who are your stakeholders?
- What software components might you need?
- How do you safely glue together those components?
- What assumptions are you making?
Robotics software engineering is all about integration.
Metrics of software quality, i.e., *design goals*

<table>
<thead>
<tr>
<th>Functional correctness</th>
<th>Adherence of implementation to the specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustness</td>
<td>Ability to handle anomalous events</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Ability to accommodate changes in specifications</td>
</tr>
<tr>
<td>Reusability</td>
<td>Ability to be reused in another application</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Satisfaction of speed and storage requirements</td>
</tr>
<tr>
<td>Scalability</td>
<td>Ability to serve as the basis of a larger version of the application</td>
</tr>
<tr>
<td>Security</td>
<td>Level of consideration of application security</td>
</tr>
</tbody>
</table>

*Source: Braude, Bernstein, Software Engineering. Wiley 2011*
Today: Software engineering in practice

• Software engineering for robotics
• **Software testing for robotics**
• Robot Operating System
Could we have prevented this bug?

https://www.youtube.com/watch?v=Rjjj6DAylsk
Unit Testing

When is unit testing not enough?
How should we test our robotics software?
Challenges for testing robotics

- It’s really expensive! Requires substantial time and resources
- It’s dangerous!
- Test setup complexity
- Unpredictable corner cases
- The “oracle problem” is even harder
  - How do we know that the robot did the right thing?
  - How do we know that the robot didn’t do a bad thing?
- Cultural and economic issues
  - Lack of incentives
  - Emphasis on results rather than quality
- ...

https://www.youtube.com/watch?v=3hKgEylk8ks
Record-and-Replay Testing

https://wiki.ros.org/rxbag
https://20kh6h3g46l33ivuea3rxuyu-wpengine.netdna-ssl.com/wp-content/uploads/2019/08/1I3NMNgKO9A8W0ww0mgWFBQ.png
Robustness Testing (a.k.a. Stress Testing)

Conceptually similar to fuzzing.

https://edge-case-research.com/
https://www.nrec.ri.cmu.edu/
End-to-End Testing: Field Testing
~$350M software bug reproduced in simulation
Simulation-based testing: Software-in-the-loop

https://i.ytimg.com/vi/FYi8grwEziE/maxresdefault.jpg
https://storage.googleapis.com/groundai-web-prod/media%2Fusers%2Fuser_75855%2Fproject_61467%2Fimages%2Fdrone_depth_materials.png
http://playerstage.sourceforge.net/stage/stage.html

PlayerStage
Simulation-based testing: Hardware-in-the-loop
Simulation-based testing: Hybrid

https://www.orbisprotect.com/empty-warehouse/
~50% of bugs can be detected with low-fidelity simulation

- Only 14% of bugs rely on the presence of physical hardware (e.g., lights and sounds).
- Only 10% of bugs depend upon environmental factors (e.g., human arm).
- 72% of bugs can be triggered using a single form of discrete input.
- Only 5% of bugs require concurrent events in order to be triggered.
- 36% of bugs occur under a particular configuration.
- 89% of bugs occur during normal operating conditions.

Today: Software engineering in practice

• Software engineering for robotics
• Software testing for robotics
• Robot Operating System
Eric Berger and Keenan Wyrobek were PhD students, working on building a platform for personal robotics.
STAIR and the Stanford AI Lab

STAIR: STanford Artificial Intelligence Robot
Artificial Intelligence Laboratory, Computer Science Department, Stanford University

Since its birth in 1956, the AI dream has been to build systems that exhibit broad-spectrum competence and intelligence. In the STAIR (STanford AI Robot) project, we are building a robot that can navigate home and office environments, pick up and interact with objects and tools, and intelligently converse with and help people in these environments.

Our single robot platform will integrate methods drawn from all areas of AI, including machine learning, vision, navigation, manipulation, planning, reasoning, and speech/natural language processing. This is in distinct contrast to the 30-year trend of working on fragmented AI sub-fields, and will be a vehicle for driving research towards true integrated AI.

Over the long term, we envision a single robot that can perform tasks such as:

- Fetch or deliver items around the home or office.
- Tidy up a room, including picking up and throwing away trash, and using the dishwasher.
- Prepare meals using a normal kitchen.
- Use tools to assemble a bookshelf.

A robot capable of these tasks will revolutionize home and office automation, and have important applications ranging from home assistants to elderly care. However, carrying out such tasks will require significant advances in integrating learning, manipulation, perception, spoken dialog, and reasoning.

http://stair.stanford.edu
Robotics had a code reuse problem

A slide from the original pitch!

https://www.theconstructsim.com/history-ros/
Personal Robotics Program and PR1

Mission
Develop platform technology for research and development where robots do mobile manipulation tasks in human environments.

PR1
Prototype mobile manipulation development platform.
- Videos - select Video on right.

PR2
The PR2 Robot is now in production at Willow Garage.

Open Source Robot Operating System (ROS)
ROS code, tutorials and documentation is available at ros.org.

People
Graduate Students
- Eric Berger - CS
- Kwanan Wyrobek - ME
PI
- Prof. Kenneth Salisbury

In Collaboration With
Stanford Artificial Intelligence Robotics Project

The robot is being teleoperated in these videos.
Spinoff: PR2 and Willow Garage

https://robots.ieee.org/robots/pr2/Interactive%20Player/SD-Q3-M360/pr2-int1-01.jpg
http://www.willowgarage.com/
“The Linux of Robotics”: Robot Operating System
Robot Operating System

ROS

- Process management
- Inter-process communication
- Device drivers

- Simulation
- Visualization
- Graphical user interface
- Data logging

- Control
- Planning
- Perception
- Mapping
- Manipulation

- Package organisation
- Software distribution
- Documentation
- Tutorials
Exponential growth in the power of APIs

*Without them, ROS wouldn’t be possible!*

’50s-’60s – Arithmetic.

’70s – malloc, bsearch, qsort, rnd, I/O, system calls, formatting, early databases

’80s – GUls, desktop publishing, relational databases

’90s – Networking, multithreading, 3D graphics

’00s – Data structures, higher-level abstractions, Web APIs: social media, cloud infrastructure

’10s – Machine learning, computer vision, IOT, robotics, pretty much everything
ROS is inherently collaborative
You don’t even need a robot!

https://fetchrobotics.com
ROS in production

https://en.wikipedia.org/wiki/Robonaut
https://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/ROS-Industrial-for-Real-World-Solutions/content_id/7919
Software architecture for ROS
Design Patterns
Architectural styles

Architectural Styles vs. Design Patterns
ROS Graph

/roscore

/map_server

/move_base

/amcl

/robot_state_publisher

...
ROS: Publish-Subscribe Architecture

https://www.researchgate.net/profile/Anis_Koubaa/publication/309668701/figure/fig1/AS:424638028226561@1478253003400/Example-of-a-ROS-Computation-Graph-An-ellipse-represents-a-node-and-a-rectangle.png
https://miro.medium.com/max/814/1*tQs9gRoMS5ePPvQProhbg.png

More info
http://wiki.ros.org/Messages
ROS: Service Calls (Remote Procedure Calls)

Service Name: /example_service
Service Type: roscpp_tutorials/TwoInts
Request Type: roscpp_tutorials/TwoIntsRequest
Response Type: roscpp_tutorials/TwoIntsResponse
ROS: Parameter Server
The evolution of ROS1 to ROS2

### ROS

- Single point of failure (roscore server)
- Designed for researchers
- Assumes excellent network connectivity.
- Hard to build multi-robot systems
- Lack of security
- No built-in real-time control support
- ...

### ROS2

- No need for roscore! Uses dynamic discovery.
- Designed for production
- Operates with degraded network connectivity
- Built for multi-robot systems
- Secure communications over SSL
- Uses new technologies for a smaller implementation (e.g., DDS, Protocol Buffers).
- ...

...
Summary

• Robots are increasingly important to our everyday lives.
• Making sure that robotics software is well designed and tested is essential.
• ROS is an evolving software framework and ecosystem for robotics development.
  – Combination of architectural styles.
  – ROS1 was confined by assumptions, and so ROS2 was born.
• 17-313 jumps from small-to-medium-scale software design to large-scale software development in the wild.
  – Requirements, quality assurance, process, machine learning, large-scale software design, economics