14: Embedded SW Engineering
Switching to SW engineering

Where we’ve been

Low-level details of embedded (IO, architecture, clocks, some OS/scheduling)

Where we’re going

Bigger picture: software engineering for embedded
What is the difference between software engineering and programming?
People skills

Software engineering involves working with people, to make products that will be used by people.

We are not flawless, nor are we machines.

We have biases, bad days, grudges, weaknesses, but also empathy, collaboration, and diverse viewpoints.
What are some ways that sloppy communication or poor management can make for bad code?
System development life cycle

5-10+ stages, may include

- Idea or solicitation by customer
- Marketing
- Planning
- Requirements/analysis
- Design
- Implementation/development
- Testing
- Verification and validation
- Operation/maintenance
Planning and design makes up majority of SW process

What percentage of your design time is spent on each of the following stages?

- Developing system specs: 13% (2019), 14% (2017), 14% (2015)
- Conceptual design stage: 11% (2019), 13% (2017), 13% (2015)
- Detailed design stage: 31% (2019), 32% (2017)
- Simulation stage: 6% (2019), 6% (2017), 8% (2015)
- Prototyping: 9% (2019), 9% (2017), 10% (2015)
- Transitioning to production: 6% (2019), 7% (2017), 6% (2015)
Illustrative example: standing desk
What would it make more sense to do first:

1. Write standing desk controller requirements
2. Unit test standing desk controller
3. Write standing desk user requirements
4. System test standing desk
V model

Week after next!

Concept of Operations
Requirements and Architecture
Detailed Design

Verification and Validation
System Verification and Validation
Integration, Test, and Verification

Operation and Maintenance
Project Test and Integration

Project Definition

Time
Left side of V model

Product requirements
What the product does from the customer POV

Software requirements
What the product does from the SW POV (high-level, not the “how”)

High level/architecture design
What modules there are in the system, which module performs which function, how modules communicate

Low level/module design
Flowcharts, statecharts/finite state machines, algorithms...
Product requirements

Our electric height-adjustable table allows you to easily and effortlessly change from sitting to standing positions throughout your day. Raising and lowering the table is simple, using its ultra quiet, feature-rich electric mechanism. It's an essential tool to any modern workspace.

Details:

- Changing your posture often keeps you more engaged and more comfortable
- Meetings are significantly shorter when standing vs. sitting
- Height-adjustable tables are essential to modern workspaces and prized by office workers everywhere
- Push-button activation with height display readout
- 3 memory positions

Customer-facing
Can be a list of features
Used in marketing

image source
Software requirements

Written with specific wording and format

“Shall” - the software must do this

“Should” - the software has this goal

Labeled or numbered (RS-1, RS-2, RS-2.a…)

Precise and measurable

Quantitative over qualitative

Can be tested

What the software does, not how
Standing desk inputs

Current height*

Buttons: 1, 2, 3, up, down, M
Standing desk outputs

Motor command (stopped, up, down)

Display
Standing desk requirements

R1: If the desk is not at its maximum height, and the up button is held, the motor shall be commanded UP

R2: If the M button is pressed and released, and one of the numbered buttons [1, 2, 3] is pressed and released within 10 seconds, then the current height shall be stored as a preset for the corresponding numbered button

R3: If one of the numbered buttons [1, 2, 3] is held, the motor should be commanded such that the desk height moves to the corresponding preset height
Come up with additional requirement(s) that refine the preset behavior

R3: If one of the numbered buttons [1, 2, 3] is held, the motor should be commanded such that the desk height moves to the corresponding preset height
Refined requirements
High-level/architecture design

How components fit together and what the interfaces are

Boxes-and-arrows diagram: **boxes** are components, **arrows** are interfaces

General rule: should fit on one page

Details of components are left to detailed design
Motor controller*

Motor command (up, down, or stopped)

Current height of desk

Microcontroller

Button [1, 2, 3, M, Up, Down] pressed or held down

Height to display

LED Display

Button array

Boxes-and-arrows for standing desk

*Here we make the assumption that the motor controller is able to output the current height of the desk (for example, based on initial calibration and on how long the motor is commanded on). This may not actually be how the controller receives the height on the product, but it’s an assumption of how there might be bidirectional communication between two components in a product architecture.
Sequence diagrams

Shows interaction between components

Columns: components

Arrows between columns: data sent across interfaces

Temporally arranged (lower is later)

Usually one for each customer **scenario**

Scenario is variant of a **use case**
Scenario: user wants to raise desk, presses up button and desk rises

Product requirements

Software requirements

High level/architect design

Low level/module design

Button array

Microcontroller

LED Display

Motor Controller

button UP pressed

get current height

receive current height

display current height

command UP

button UP released

command STOPPED
Scenario: store current height as preset 2

- Button array
  - M pressed, M released
- Microcontroller
  - Get current height
  - Current height
  - Display current height
  - Start 10s timer
  - Display nothing, display current height, display nothing
- LED Display
- Motor Controller
  - Store current height in preset 2
  - Display nothing