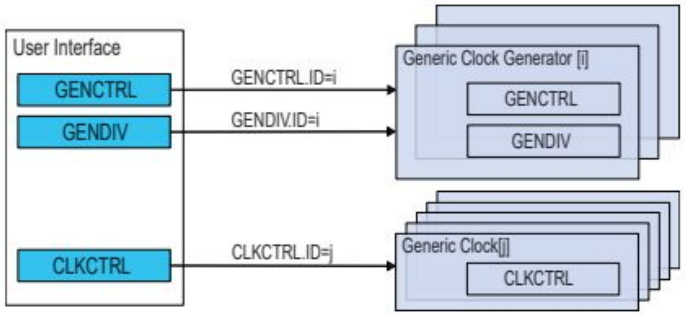
Embedded design and engineering

Assignments

Indirectly addressed registers

Why does SYNCBUSY hang at weird times?

Figure 15-5. GCLK Indirect Access



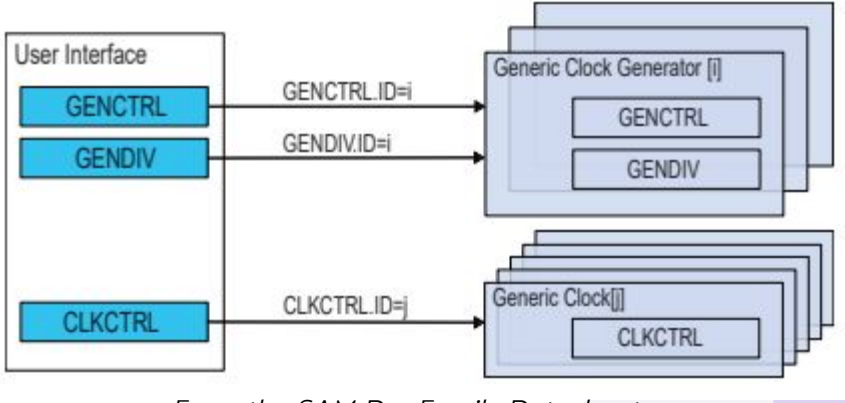
Writing these registers is done by setting the corresponding ID bit group. To read a register, the user must write the ID of the channel, i, in the corresponding register. The value of the register for the corresponding ID is available in the user interface by a read access.

For example, the sequence to read the GENCTRL register of generic clock generator i is:

1. Do an 8-bit write of the i value to GENCTRL.ID
2. Read the value of GENCTRL

From the SAM D21 Family Datasheet

GENCTRL causing hang



From the SAM D21 Family Datasheet

GENCTRL isn't actually a register, it abstracts access to multiple registers

Writing to the ID bits of the register tells the MCU which *actual* GENCTRL register to access

If multiple ID bits are set (i.e. if you wrote to GENCTRL using |= rather than =), the write will hang

- It doesn't know which bits to access!
- SYNCBUSY thinks a write is still happening so never clears

Projects

I will read your project proposals this weekend and give feedback

Each team will get a TA mentor to check in with weekly

If you are capstoning: think of possible individual contributions; watch for Ed post to schedule a meeting with me

Next steps: requirements, design, prototyping

Today

Where we've been

Nitty-gritty of embedded (physics, architecture, memory, clocks)

Where we're going

Bigger picture: software engineering for embedded

What is the difference between software engineering and programming?

Difference

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?

communication

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

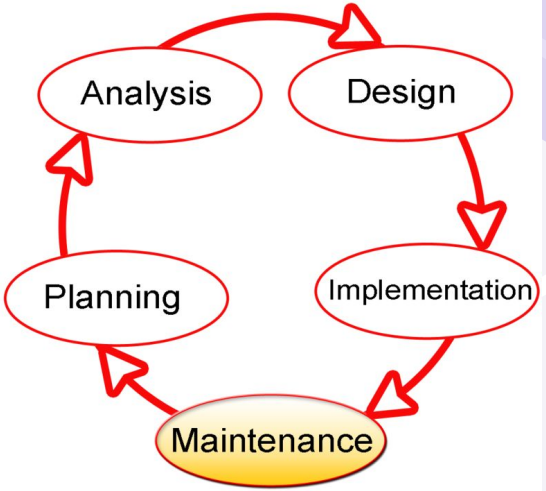


Image source

Planning and design makes up majority of SW process

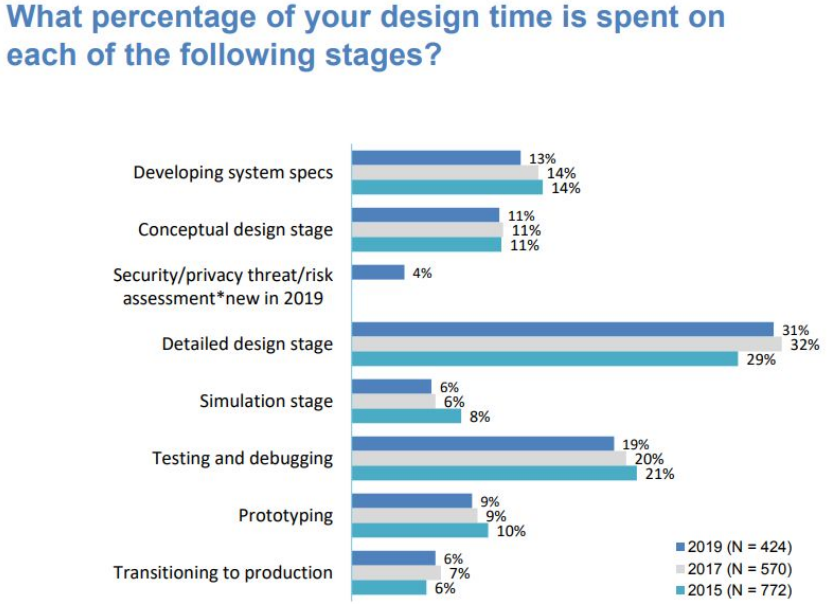


Image source

V model

NEXT WEEK!

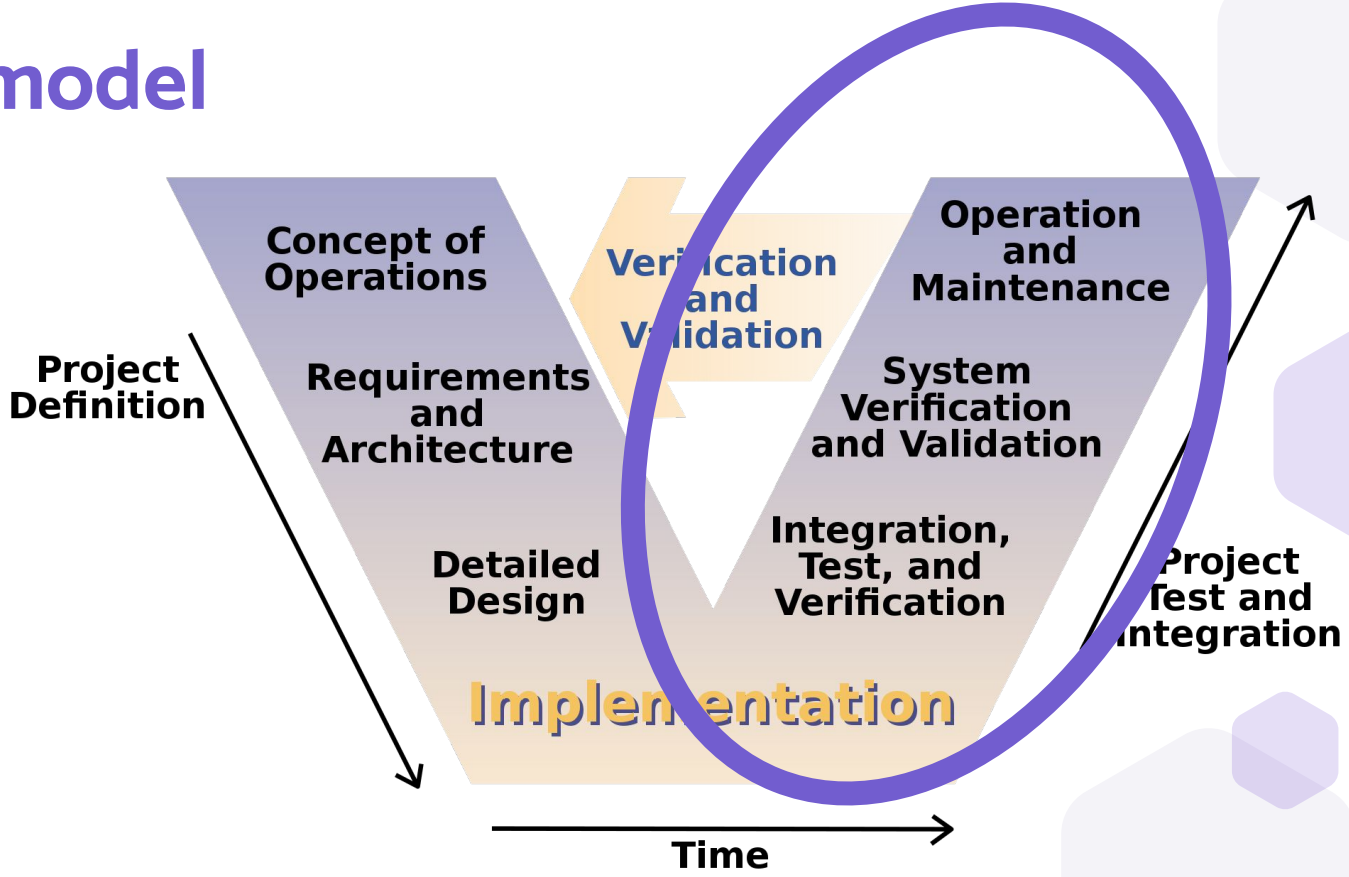
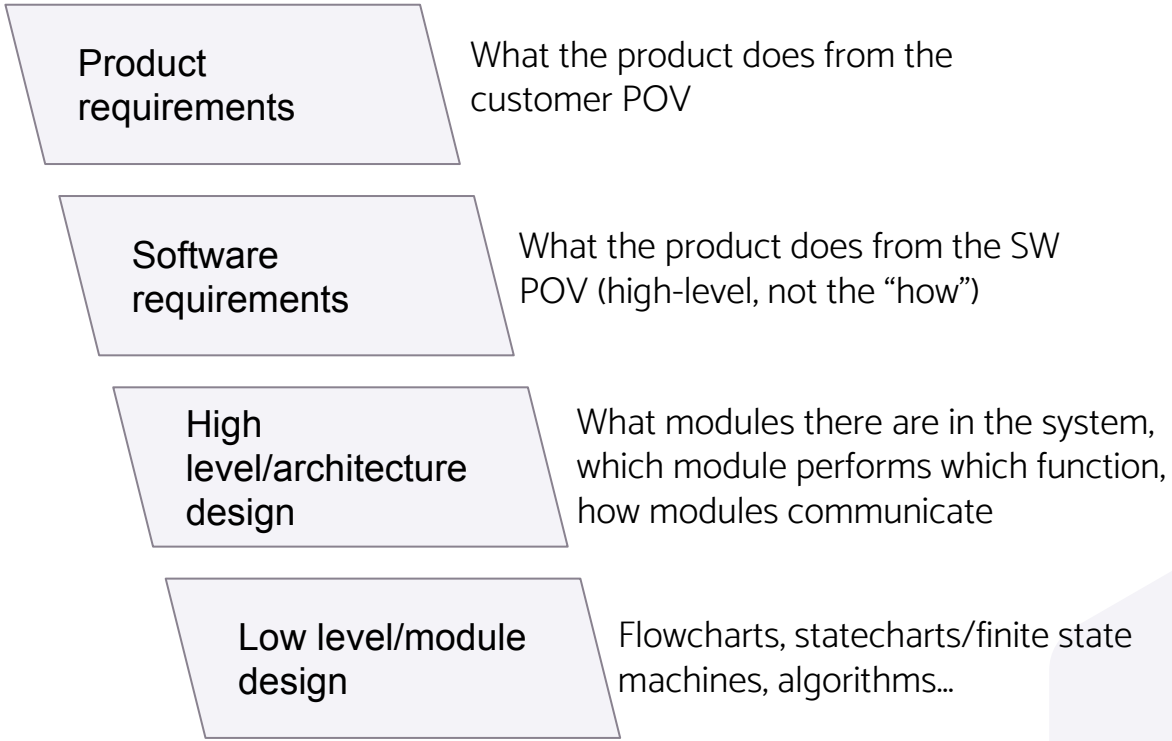


Image source

Left side of V model

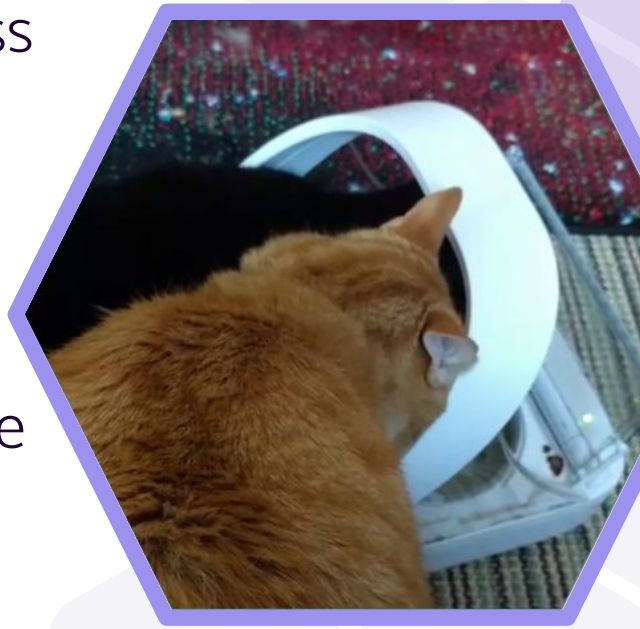


Illustrative example: microchip cat feeder

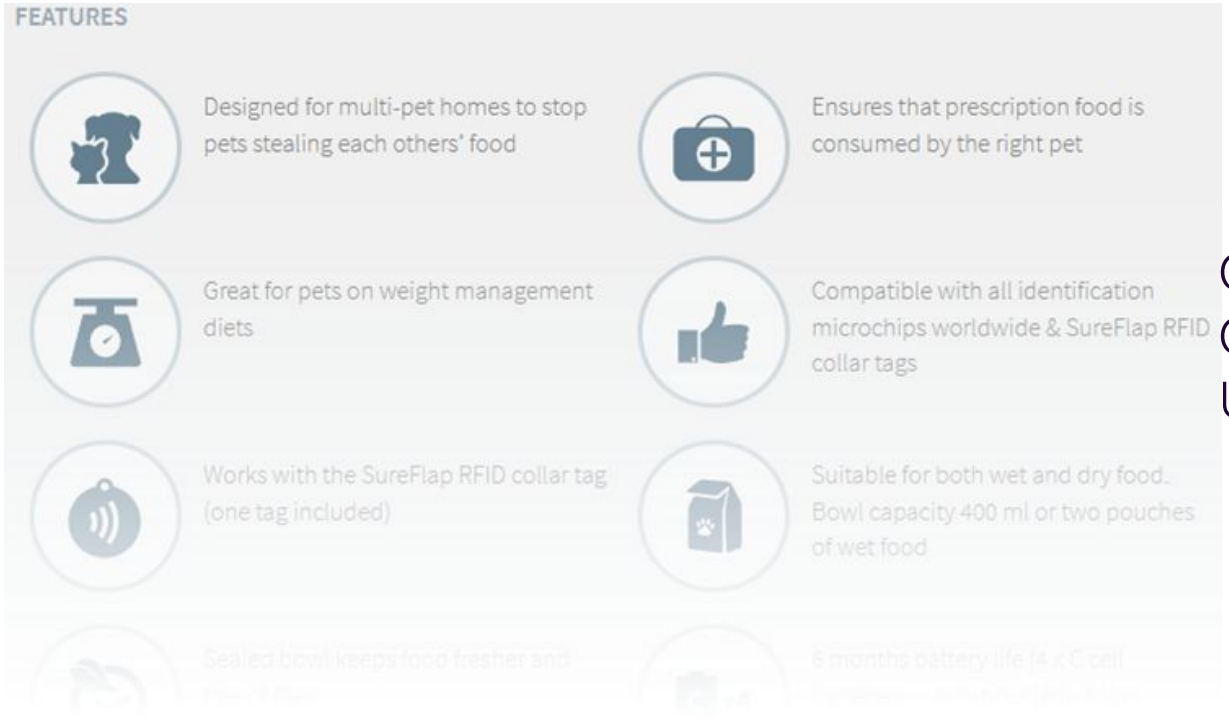
Designed to only give authorized pets access to food

If microchip scanner detects correct cat, lid opens

Also has a “training mode” where lid is in one of several partially open positions

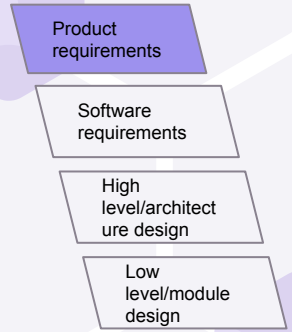


Product requirements



[image source]

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



Break

Software requirements

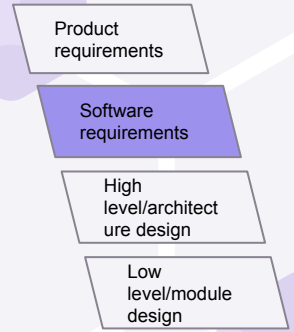
Written with specific wording and format

- “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*



Cat feeder inputs

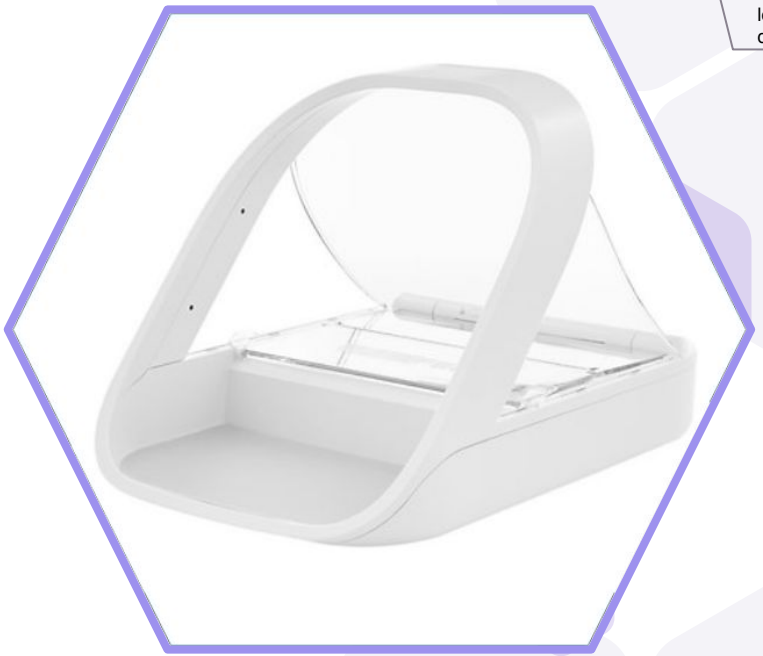
Microchip #

Lid close/open button

Program microchip button

Training mode button

Lid delay switch



Product requirements

Software requirements

High level/architecture design

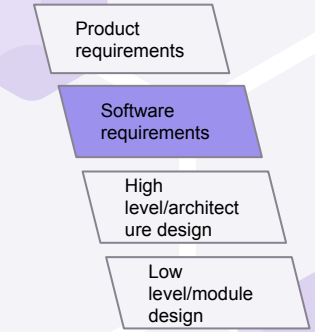
Low level/module design

Image source

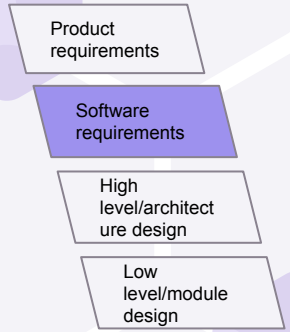
Cat feeder outputs

Lid angle

Status LED



Cat feeder requirements



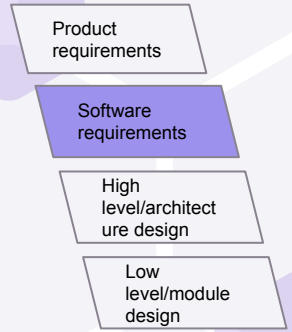
R1: If the lid is closed and the lid open/close button is pressed, the lid shall open

R2: If the lid is open and the lid open/close button is pressed, the lid shall close

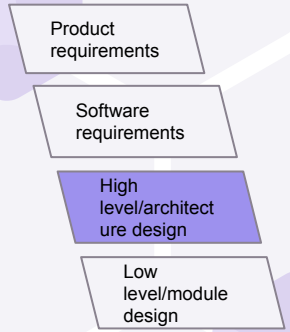
R3: The lid delay seconds shall be 3 if the delay switch is at position 3, 2 if the delay switch is at position 2, and 1 if the delay switch is at position 1

Working with your table group, come up with 4 more requirements for the feeder

More cat feeder requirements



High-level/architecture design

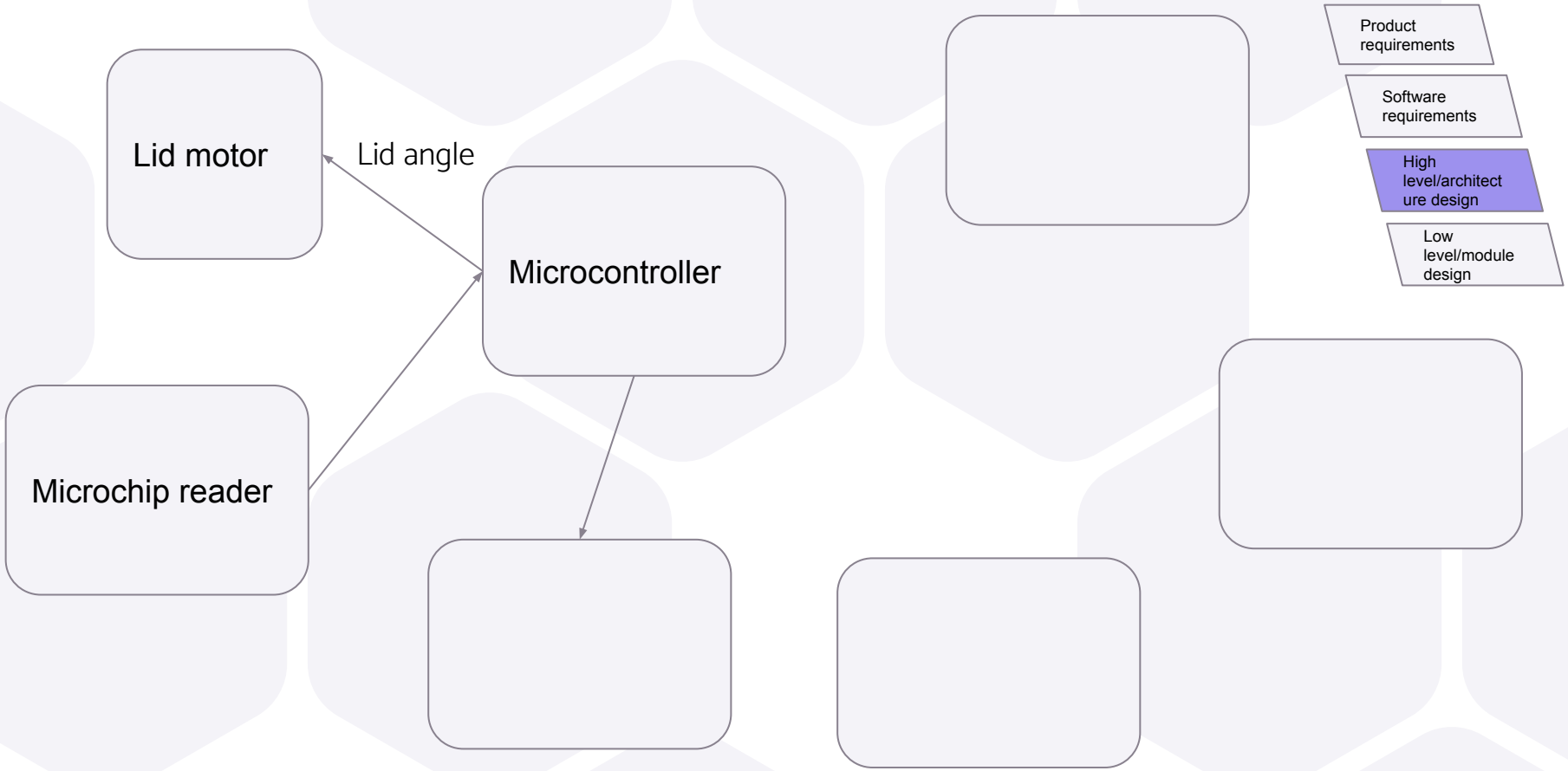


How components fit together and what the interfaces are

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

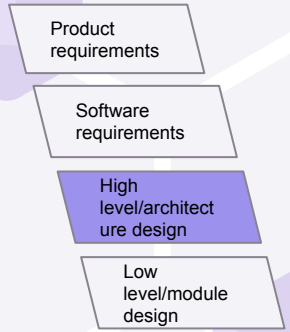
Rule-of-thumb: should fit on one page

- Details of components are left to detailed design



Boxes-and-arrows for cat feeder

Sequence diagrams



Shows interaction between components

Columns: components

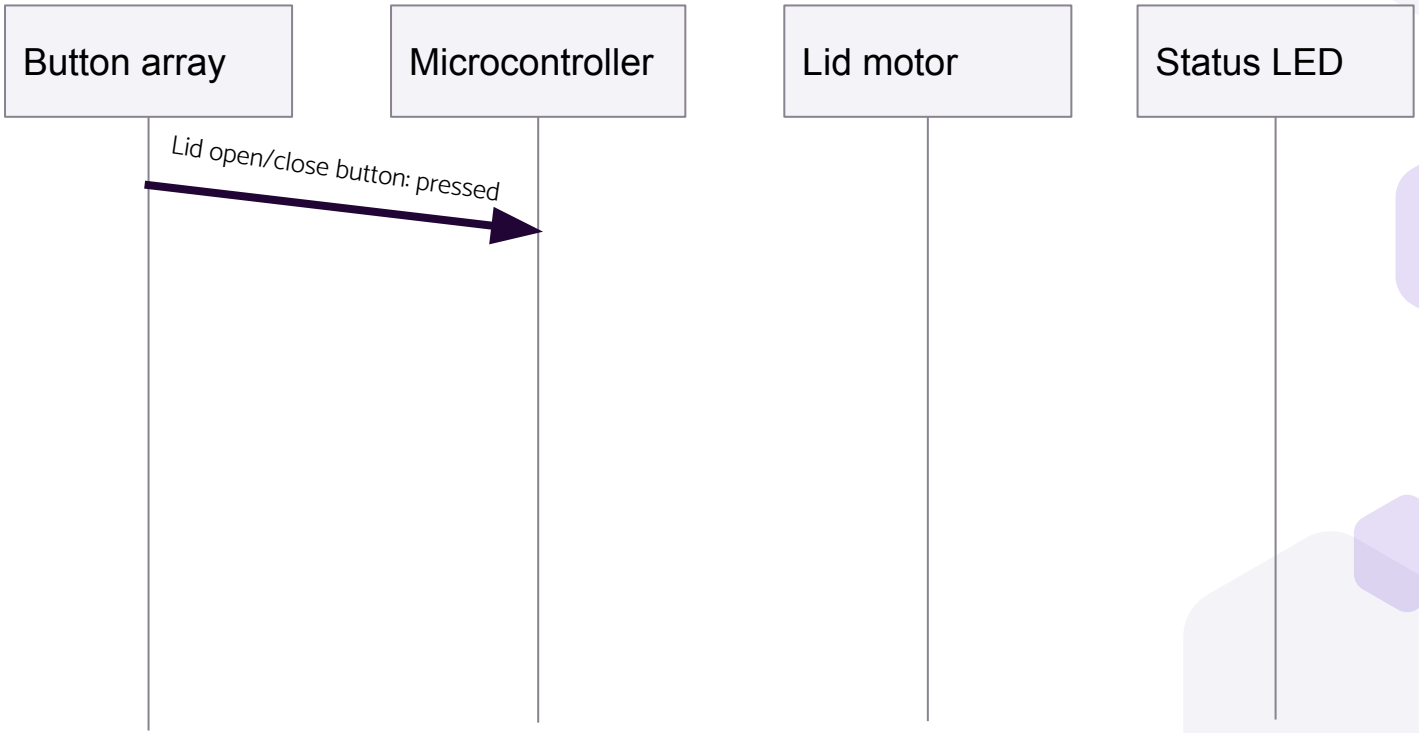
Arrows between columns: data sent across interfaces

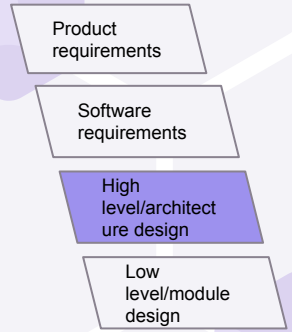
Temporally arranged (lower is later)

Usually one for each customer **scenario**

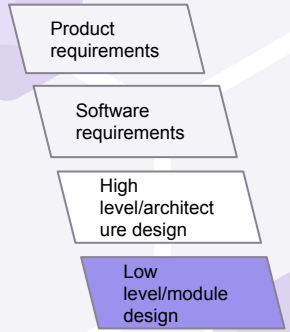
Scenario is variant of a **use case**

Cat feeder sequence diagram example





Break



Finite state machines

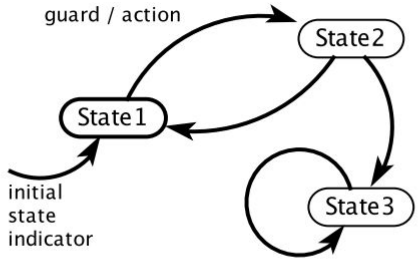
Low-level design for a module

Shows the change in state of a module

- Contrast with **flowchart**, which just shows flow of computation

At basic level, composed of:

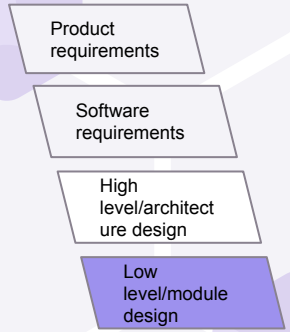
- States (one state is initial state)
- Guards (predicates on inputs)
- Actions (setting outputs)



Lee/Seshia chapter 3

Figure 3.3: Visual notation for a finite state machine.

Variants



Multiple ways to define statecharts/FSMs

- Mealy vs. Moore, deterministic vs non-deterministic, etc
- Extended FSMs: state variables (variables that are **not** inputs or outputs) can appear in guards and actions

We will use **deterministic, extended** FSMs as defined by Lee/Seshia

- Will be useful when we talk about modeling
- Translate well to coding

Cat feeder FSM

Whiteboard demo - concepts discussed: inputs, outputs, variables, guards, actions, stutter

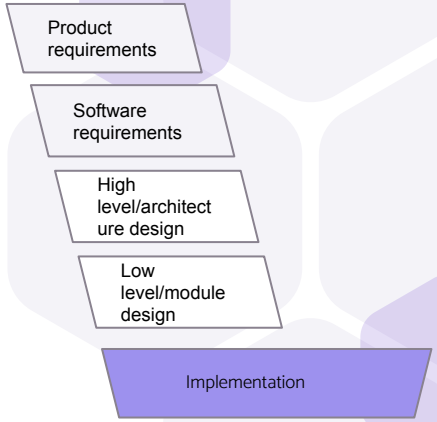
Implementation of FSM

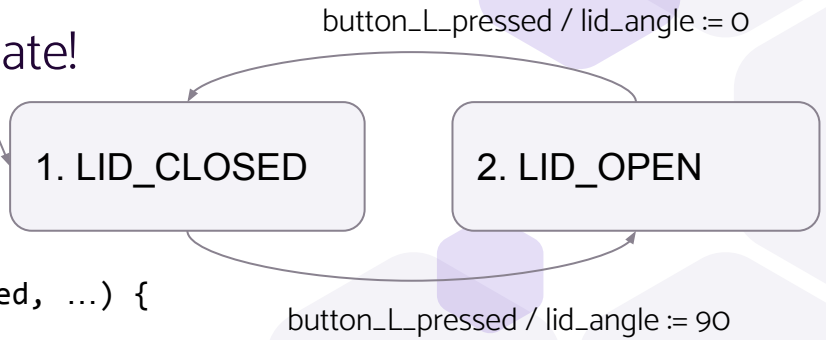
Define states as an enum in code

Switch statement on current state

If/else if on guards, actions in blocks

Stutter - remember to return start_state!





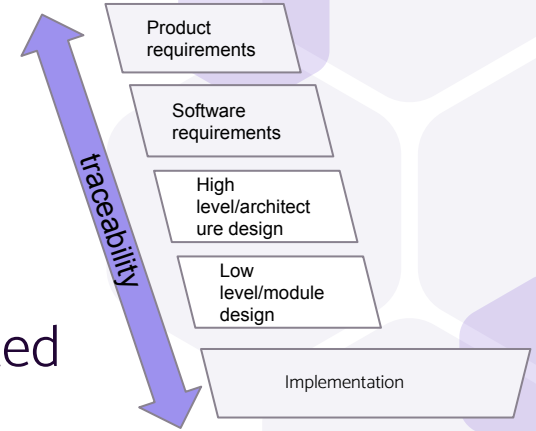

```
typedef enum { LID_CLOSED = 1, LID_OPEN = 2 } state;

state update_fsm(state current_state, bool button_L_pressed, ...) {
  state next_state;

  return next_state;
}
```

How do we know that our design has met our requirements?

Traceability



Ensures that all requirements have been implemented and tested

Often done using a **traceability matrix**

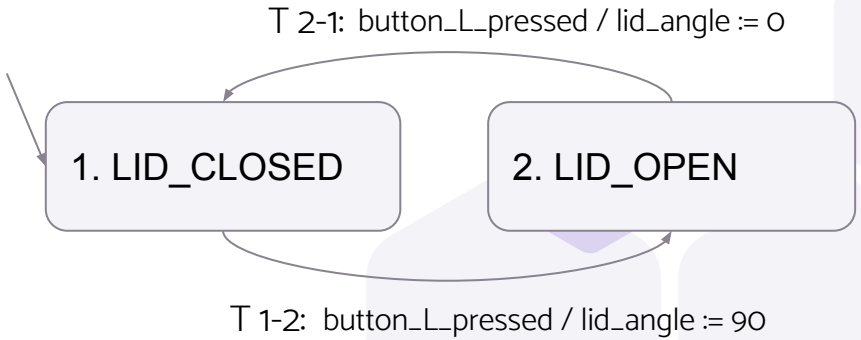
- Example: each column is a requirement; each row is a transition
- “x” in a cell if the transition helps meet the requirement
- If a column has no x’s, means requirement isn’t being met
- If a row has no x’s, means transition is unnecessary (or requirement is missing/wasn’t stated!)

Example: requirements to FSM traceability

R1: If the lid is closed and the lid open/close button is pressed, the lid shall open

R2: If the lid is open and the lid open/close button is pressed, the lid shall close

	R1	R2
T 1-2		
T 2-1		



Summary

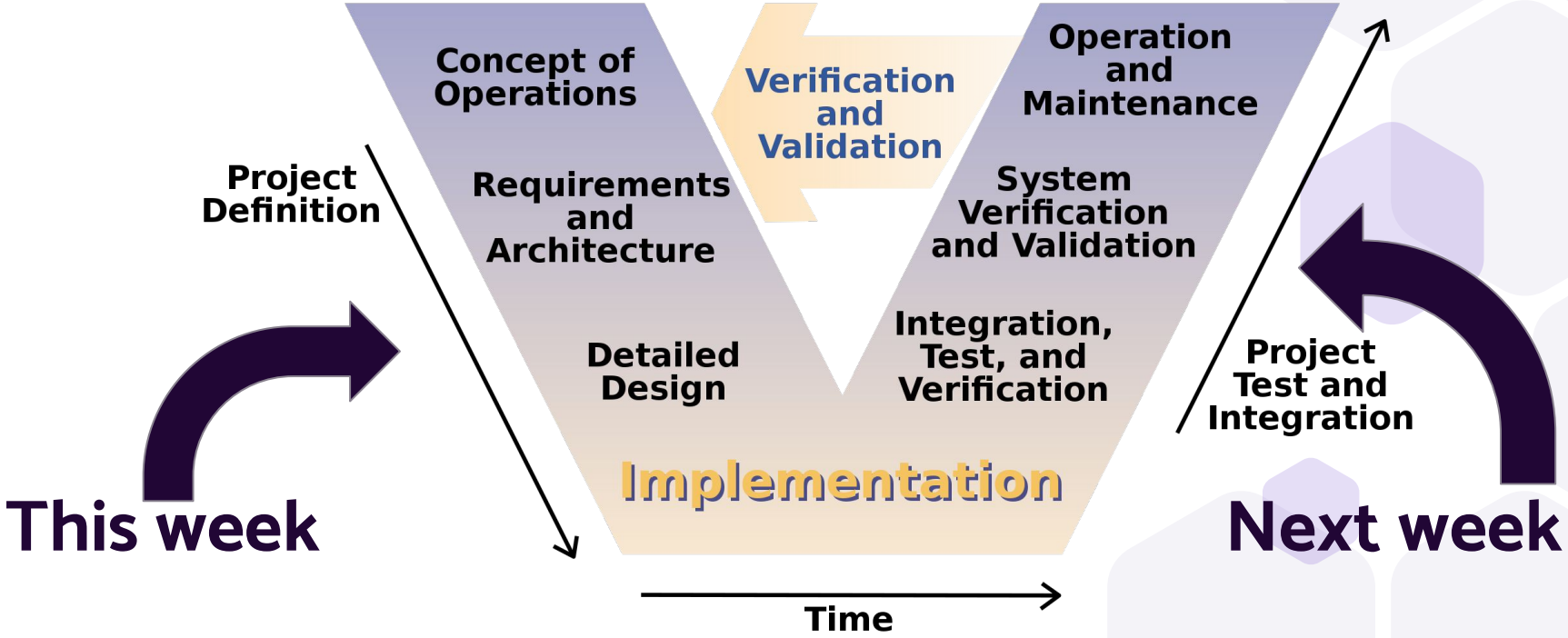


Image source