


# Scope of Software Architectures Canonical Case Studies


1



## Goal of Presentations

- Show how architectures can be used to
  - Specify solution to problem
  - Analyze solution
- Evaluate use of different architecture styles for case studies
  - Key Word In Context (KWIC)
  - Instrumentation Software
  - Mobile Robot Architecture


2



## Review — Definitions

- Perry & Wolf
  - Software Architecture = { Elements, Form, Rationale }
- IEEE Std 1471-2000
  - Fundamental organization of a system embodied in
    - Its components
    - Relationships among the components
    - Relationships to the environment
    - Principles guiding its design & evolution
- Shaw & Garlan (cont.)
  - [A level of design that] involves
    - Description of elements from which systems are built
    - Interactions among those elements
    - Patterns that guide their composition
    - Constraints on these patterns


3



## Review — Key Architectural Concepts

- Component
  - Unit used as a part of some system
  - Locus of computation and state
- Connector
  - Element that models
    - Interactions among components
    - Rules that govern those interactions
- Configuration (Form)
  - Connected graph of components & connectors which describes architectural structure


4

 Center For Software Engineering

## Review — Software Architecture Goals

- **Manage software complexity**
  - Elevate abstraction level
  - Match developers' mental models
- **Explicitly address system's conceptual underpinnings**
  - Act on blueprint of system
  - Increase reuse & component marketplace potential
  - Reduce development costs
  - Shift development approach to component-based philosophy


5

 Center For Software Engineering

## Review — Focus Of Architectures

- **System structure**
  - Correspondence between requirements & implementation
    - Components + rules of composition + rules of behavior
- **Key role in software lifecycle**
  - A framework for satisfying requirements
  - Technical basis for design
  - Managerial basis for cost estimation & process management
  - Effective basis for reuse
  - Basis for consistency & dependency analysis
  - Basis for implementation


6

 Center For Software Engineering

## Review — Focus Of Architectures (cont.)

- **Framework for understanding system-level concerns (properties) & goal**
  - Global rates of flow
  - Communication patterns
  - Execution control structure
  - Scalability
  - Paths of system evolution
  - Capacity
  - Throughput
  - Consistency
  - Component compatibility

7

 Center For Software Engineering

## Review — Focus Of Architectures (cont.)

- **Uses:**
  - Representation
  - Design Process Support
  - Analysis
    - Static
    - Dynamic
  - Evolution
    - Specification-Time
    - Execution-Time
  - Refinement
  - Traceability
  - Simulation/Executability

8

**Center For Software Engineering**

## Scope Of Software Architectures

- ❑ Every system has architecture
- ❑ Architecture is reflection of
  - System requirements &
  - Trade-offs that made to satisfy them
- ❑ Possible decision factors
  - System engineering & building goals
    - Performance
    - Safety
    - Security
    - Fault tolerance
    - Adaptability
  - Compatibility with legacy software
  - Planning for reuse
  - Distribution profile
  - Potential for changes to
    - Processing algorithms
    - Data representation
    - Structure/functionality

9

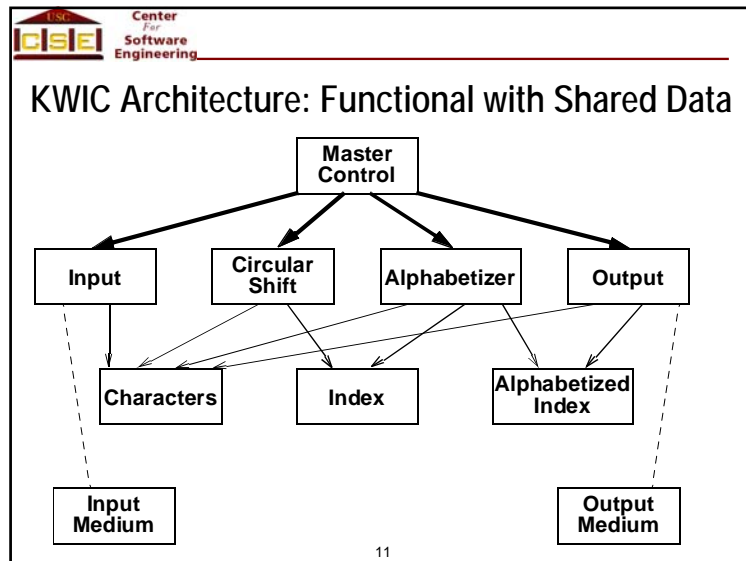
**Center For Software Engineering**

## Case Study: Key Word In Context (KWIC)

The diagram shows a vertical list of lines on the left. A blue box highlights a window that moves across the list. Below the window, the KWIC output is shown as a sequence of lines where each line contains a key word and its surrounding context. For example, 'HELLO WORLD' is followed by 'WORLD HELLO' and 'I AM YODA'.

HELLO WORLD      WORLD HELLO      AM YODA I  
 WORLD HELLO      I AM YODA      HELLO WORLD  
 I AM YODA      AM YODA I      I AM YODA  
 AM YODA I      YODA I AM      WORLD HELLO  
 YODA I AM      YODA I AM      YODA I AM

10

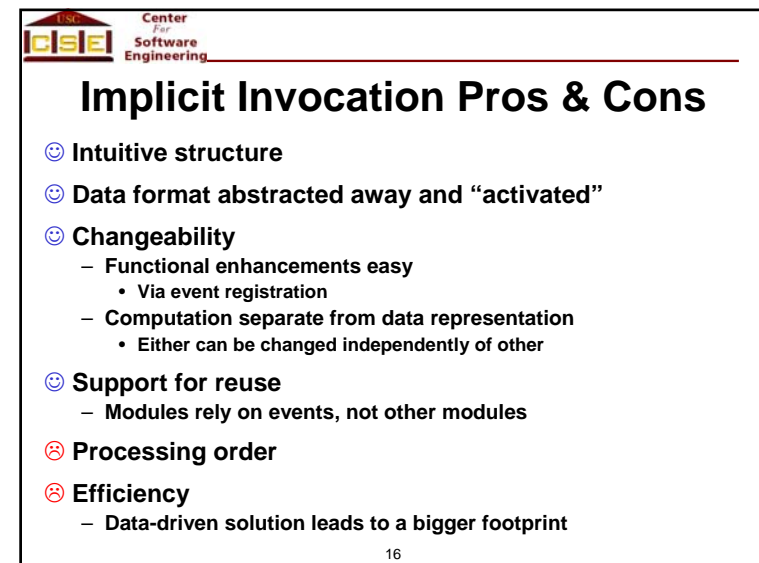
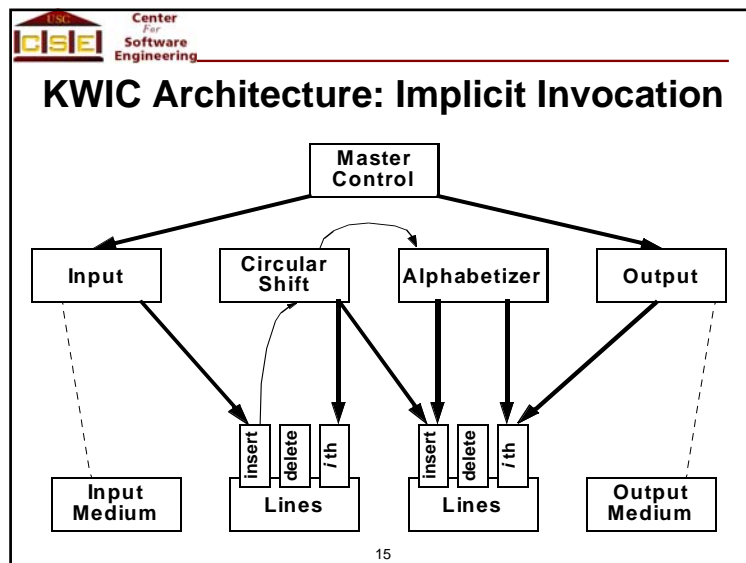
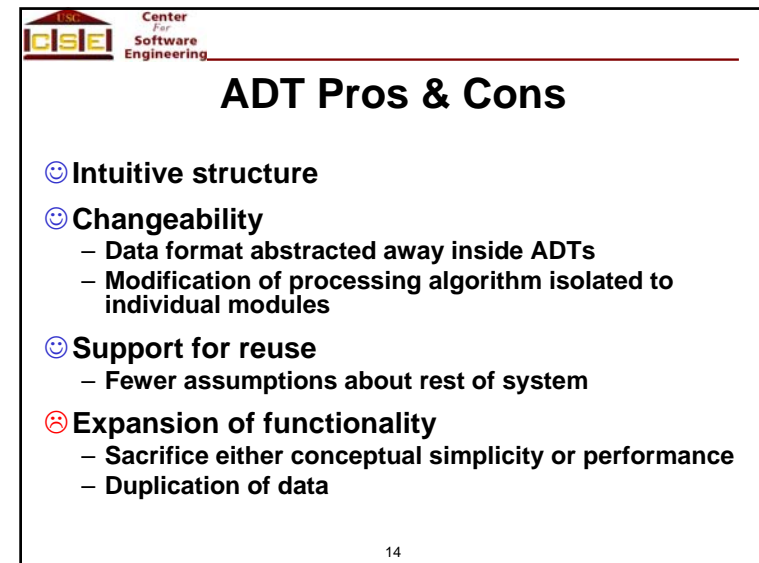
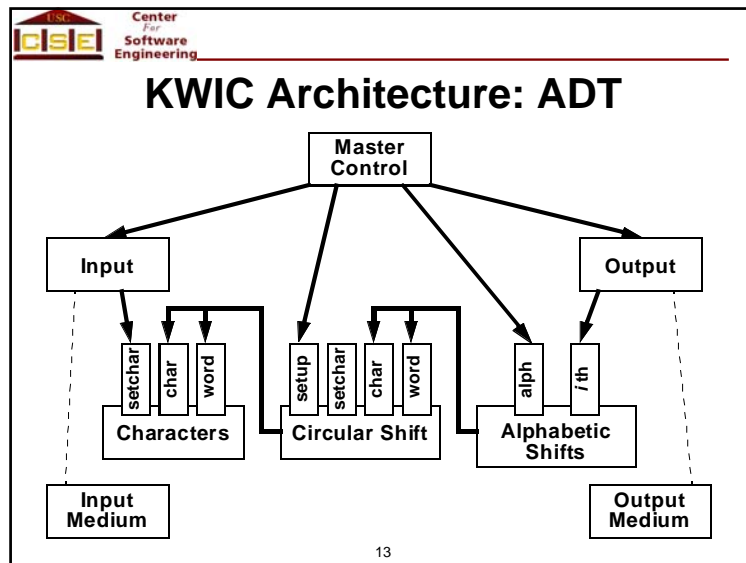


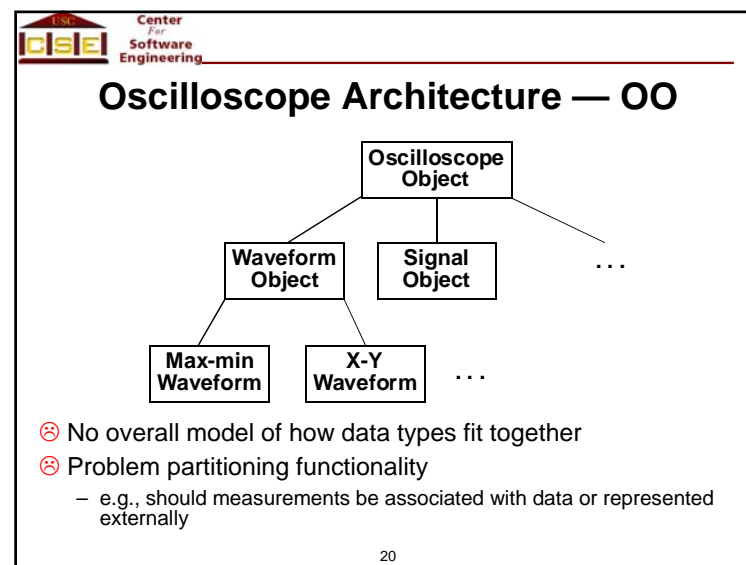
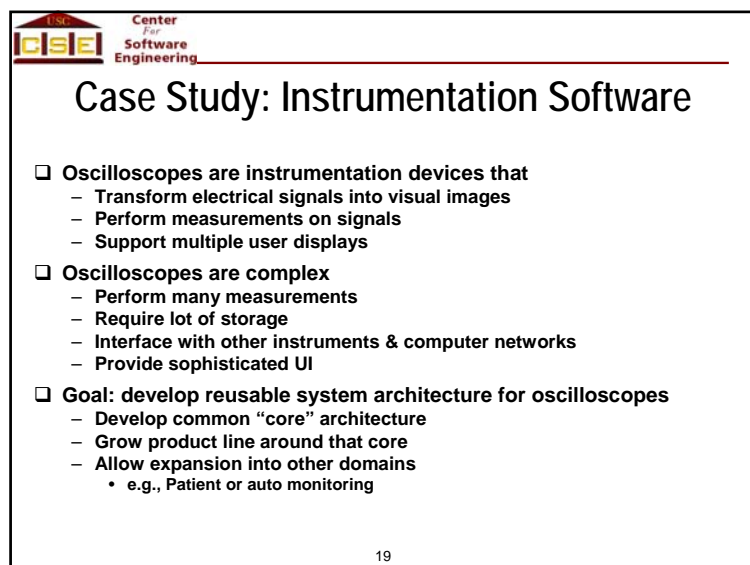
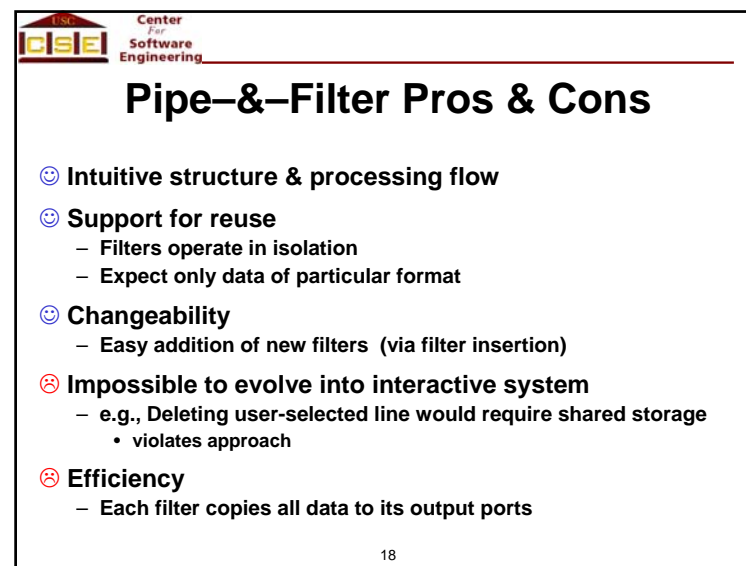
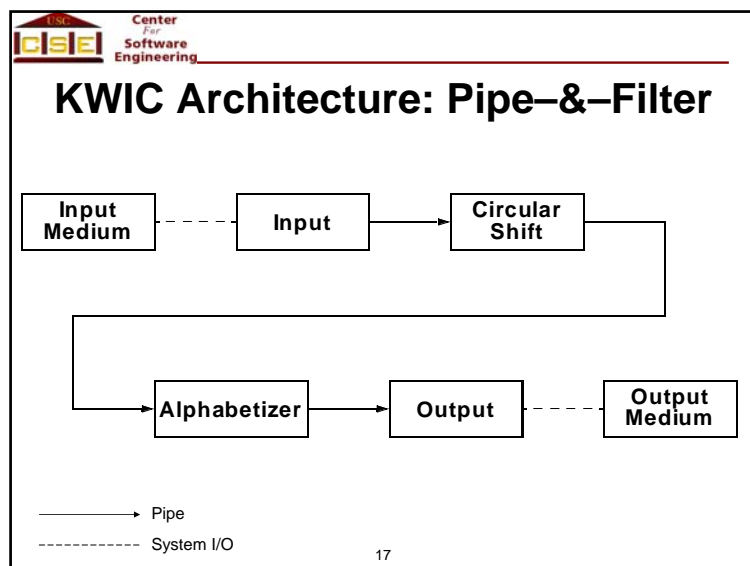
**Center For Software Engineering**

## Shared Data Pros & Cons

- ☺ Efficiency
  - Shared data
  - Efficient data representation
  - Sequential data access
- ☺ Intuitive structure
- ☹ Changeability
  - Data format not abstracted away
  - Functional elements dependent on data representation
- ☹ Support for reuse

12





**Oscilloscope Architecture — Layered**

Core  
Acquisition  
Manipulation  
Visualization  
UI

- ☺ Intuitively appealing
- ☹ Wrong for application domain
  - Actual oscilloscope functions cross layers

21

**Oscilloscope Architecture — Pipe & Filter**

Signal → Couple → Acquire → To-XY → Clip →

Trigger Subsystem → Times → Acquire

Acquire → Waveform → Measure → Measurement

To-XY → Trace → Clip

- ☺ Functions not isolated into separate partitions
- ☺ Data flow nature of signal processing is reflected
- ☺ Allows combination & substitution of software/hardware components
- ☹ Does not enable user to interact with system

22

**Oscilloscope Architecture — Modified Pipe & Filter**

Signal → Couple → Acquire → To-XY → Clip →

Trigger Subsystem → Times → Acquire

Acquire → Waveform → Measure → Measurement

To-XY → Trace → Clip

Coupling → Couple

Kind, Rate → Acquire

Trans → To-XY

Size → Clip

- ☐ **Solution: add control interfaces to filters**
- ☺ Exposes modifiable parts of filter
- ☺ Decouples signal processing functions from UI
- ☹ Poor performance
  - Each filter copies data
  - Slow filters present bottlenecks
  - Alleviated by flexible pipes (connectors)

23

**Case Study: Mobile Robotics**

- ☐ **Manned or partially manned vehicles**
- ☐ **Uses**
  - Space exploration
  - Hazardous waste disposal
  - Underwater exploration
- ☐ **Issues**
  - Interface with external sensors & actuators
  - Real-time response to stimuli
  - Response to obstacles
  - Sensor input fidelity
  - Power failures
  - Mechanical limitations
  - Unpredictable events

24

**Basic Mobile Robot Architectural Requirements**

- ❑ Accomplish goals in face of
  - **Obstacles**
  - **Uncertainty** from incomplete/unreliable information
  - **Dangers** introduced by environment
    - Fault tolerance
    - Safety
    - Performance
- ❑ Exhibit **flexibility**
  - Experimentation (plug and play)
  - Reconfiguration
  - Regular modification

25

**Mobile Robot Architecture — Control Loop**

- ☹ Obstacles
- ☹ Uncertainty
- 😊 Dangers
- 😊 Flexibility

26

**Mobile Robot Architecture — Layered**

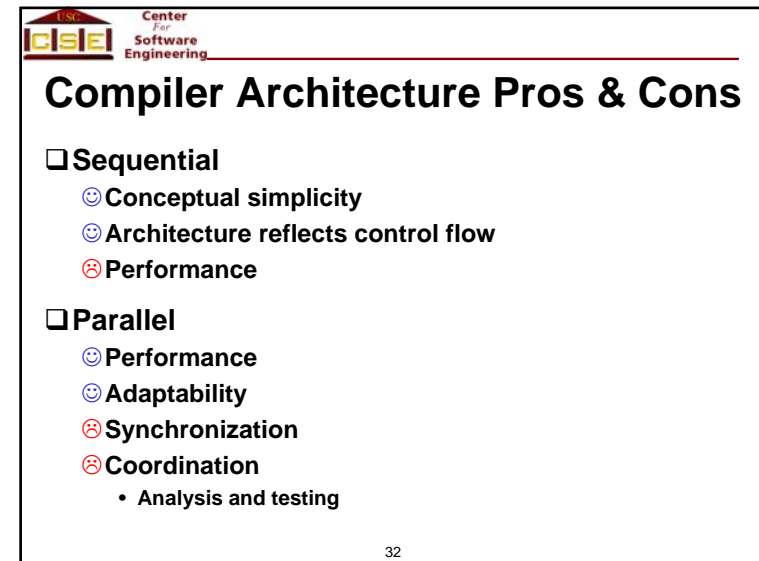
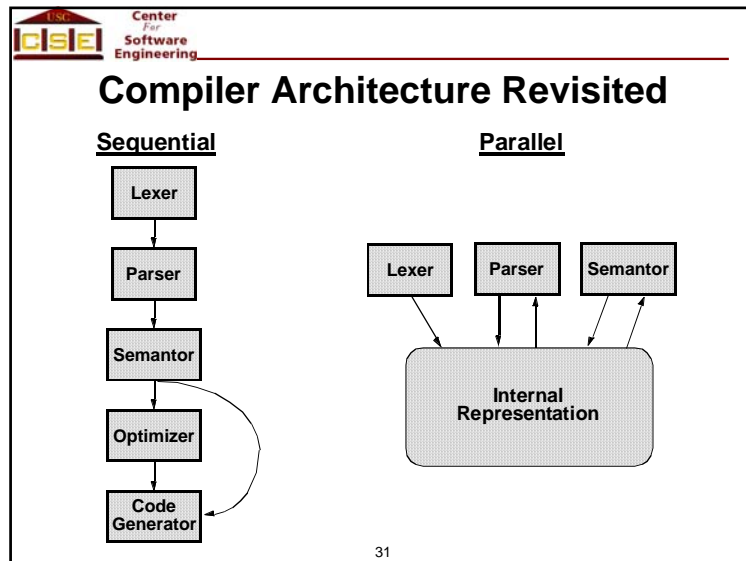
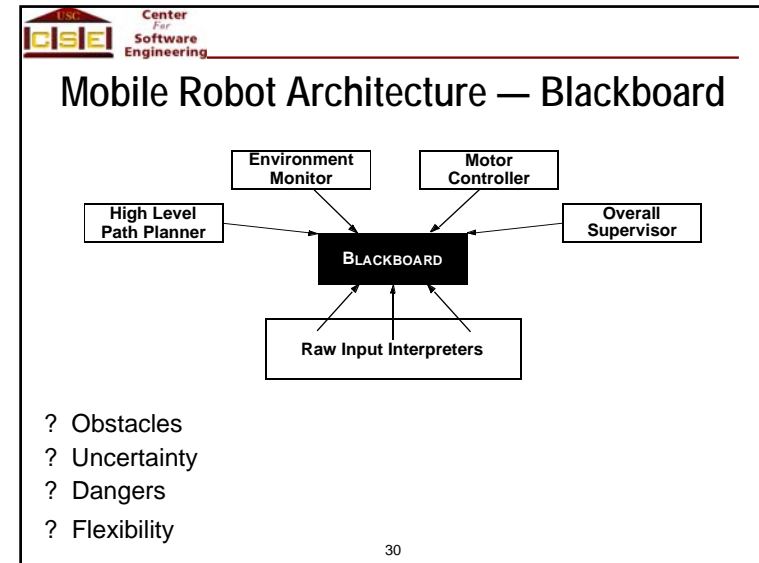
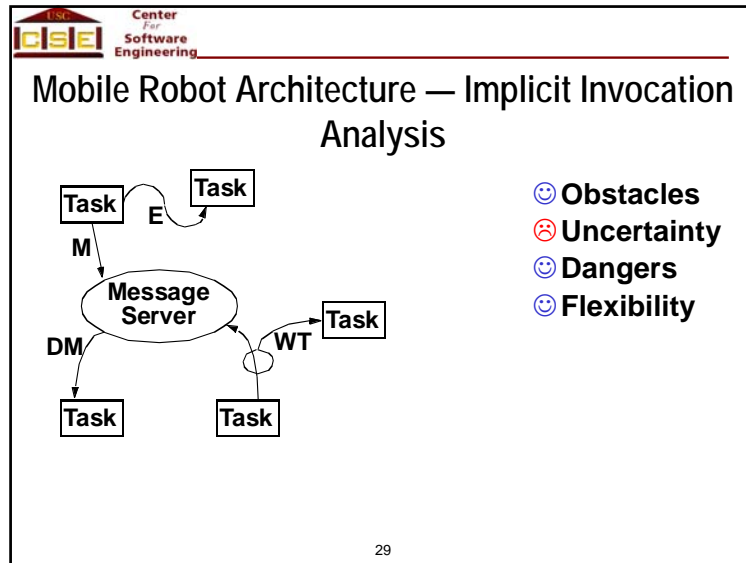
- ☹ Obstacles
- 😊 Uncertainty
- 😊 Dangers
- ☹ Flexibility


27

**Mobile Robot Architecture — Implicit Invocation**

- ❑ Task trees
  - Hierarchies of tasks
  - Tasks temporally interdependent
  - Allows specification of selective concurrency
- ❑ Tasks communicate by multicasting messages
  - Server directs messages to registered tasks

28





## Summary

- **Architectures can be used to**
  - **Specify solution to problem**
    - Designing a system that meets requirements
    - Making trade-offs to made to satisfy requirements
  - **Analyze solution**
    - Understanding system-level concerns (properties) & goal
  - **Manage software complexity**
    - e.g. canonical solutions

33