Lecture 3 - January 13

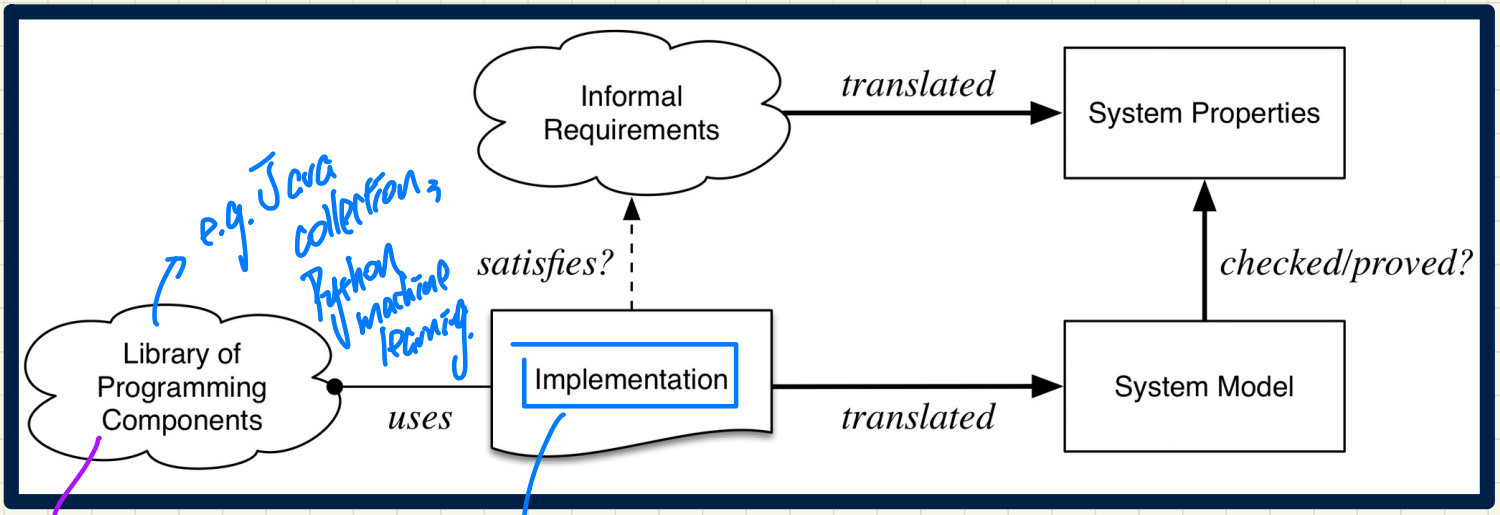
Introduction

Software Development Process
Assurance Cases
Correct by Construction
State Space
Counter Problem: Theorem Proving

Announcements/Reminders

- Lab1 released
- Office Hours: 3pm to 4pm, Mon/Tue/Wed/Thu

Building the product right? Building the right product?



e.g. Java collection, Python machine learning.

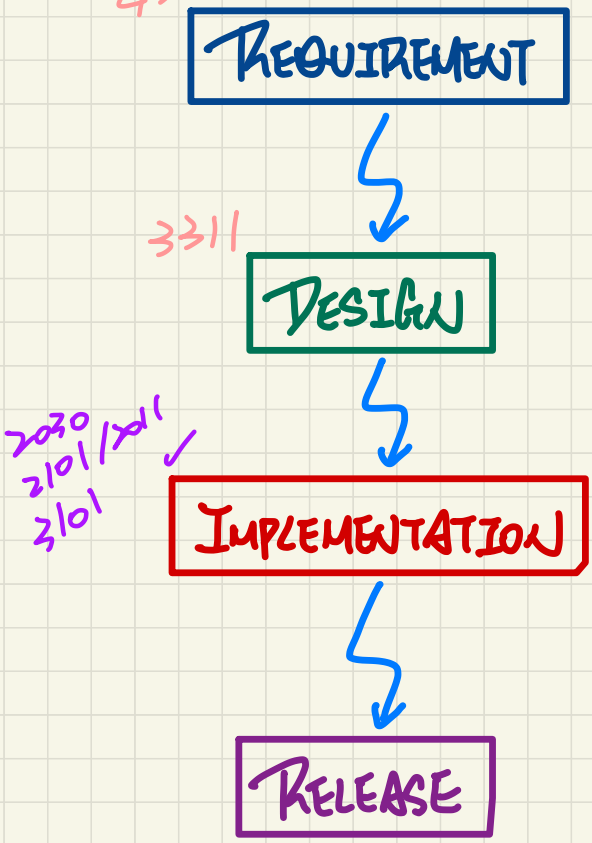
practically, not feasible to verify this component formally.
↳ instead: unit testing

is not necessarily built from scratch
⇒ library classes might be used

("don't re-invent the wheel" unless you really have to)

Software Development Process

the earlier to reveal a defect/bug, the less costly it is



- Natural Language (incomplete, ambiguous, contradicting)
- Requirement Elicitation

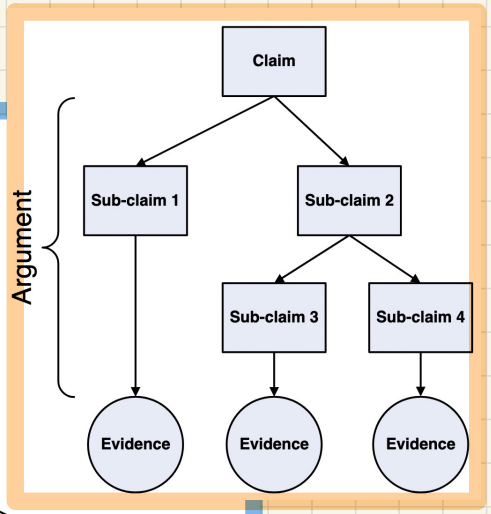
- Blueprints
- Not necessarily executable & testable

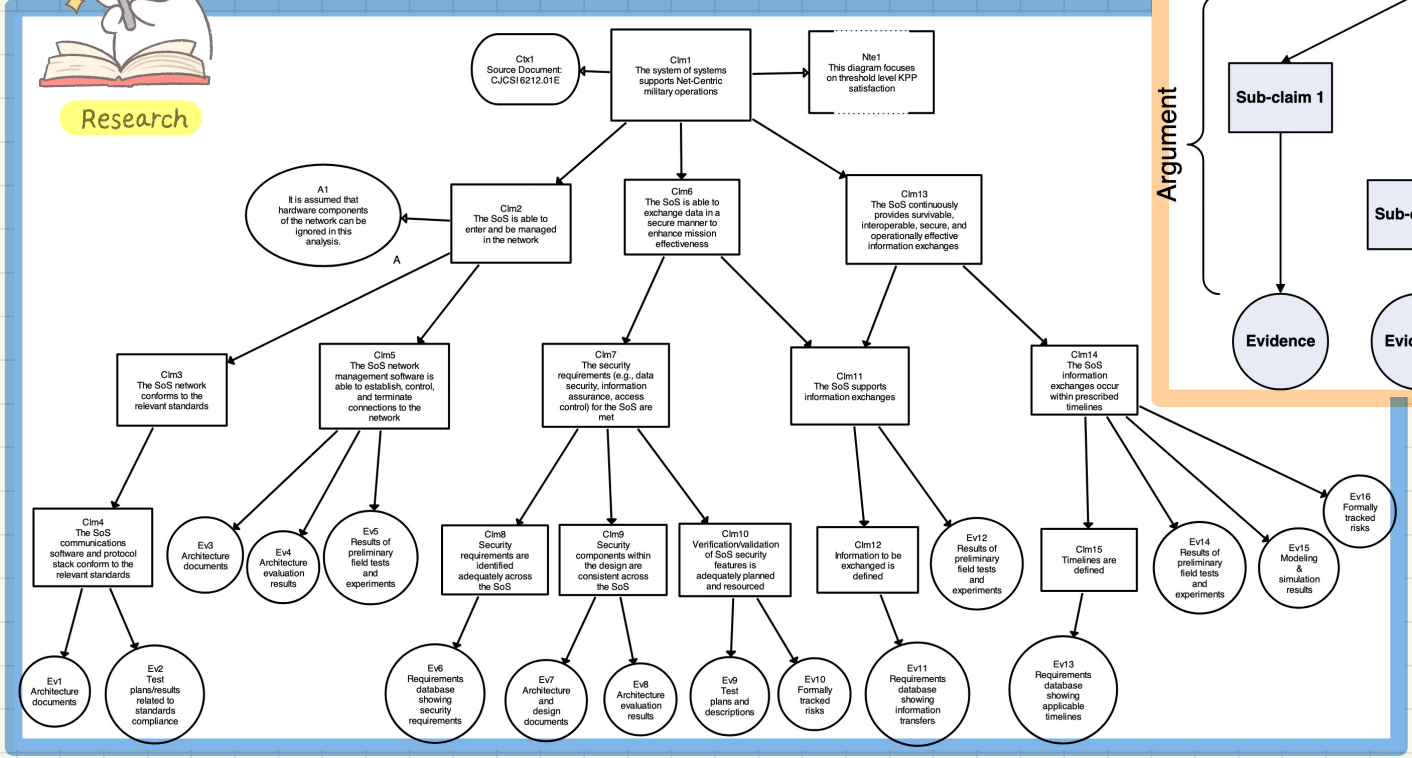
- API Given
- Efficient (data structures & algorithms)
- Unit Tests

- Customer's Acceptance
- Recall?

Certifying Systems: Assurance Cases

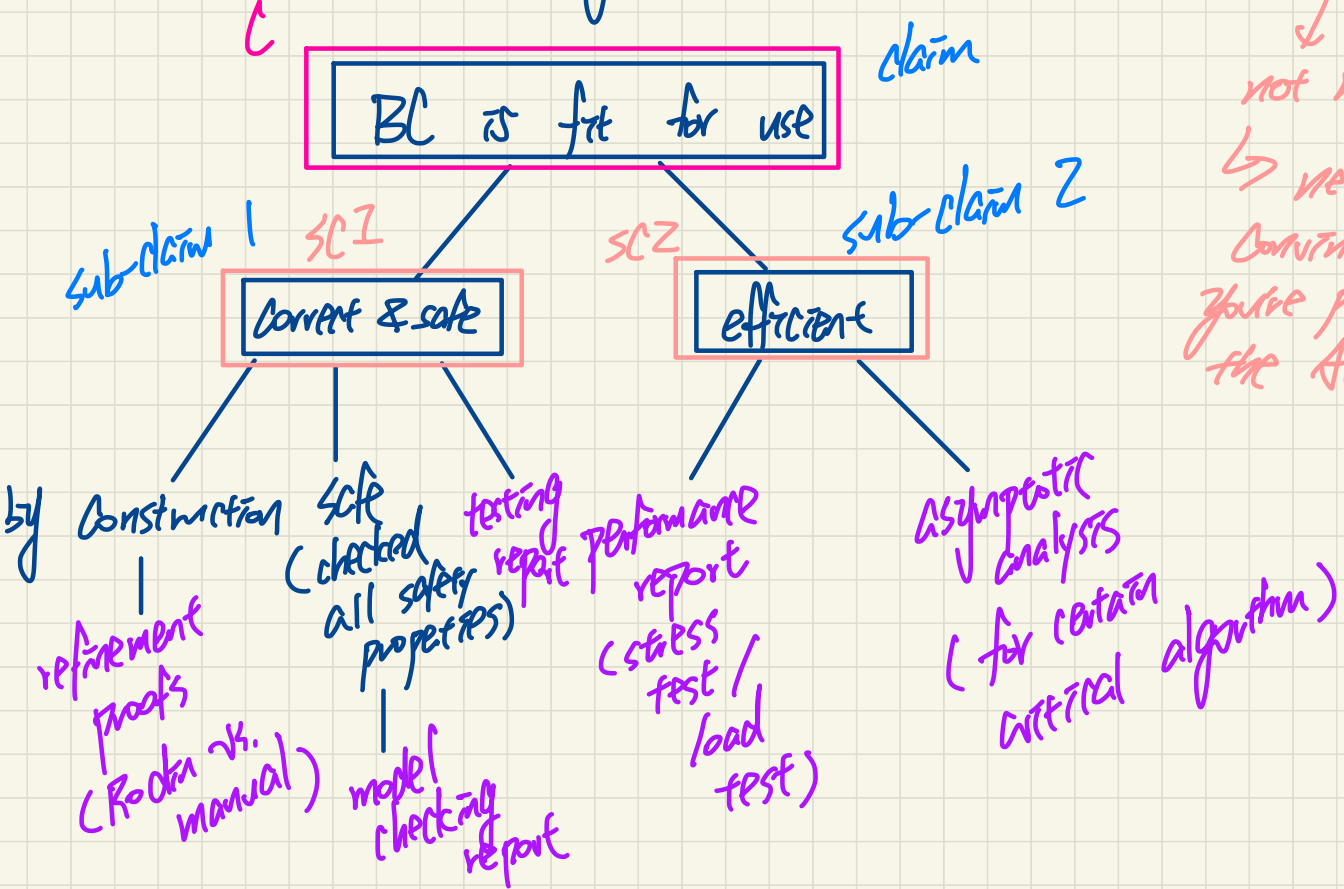
Research on "Assurance Cases" if interested!





Source: https://resources.sei.cmu.edu/asset_files/whitepaper/2009_019_001_29066.pdf

Assurance Case for Bridge Controller



$SC1 \wedge SC2 \Rightarrow C$

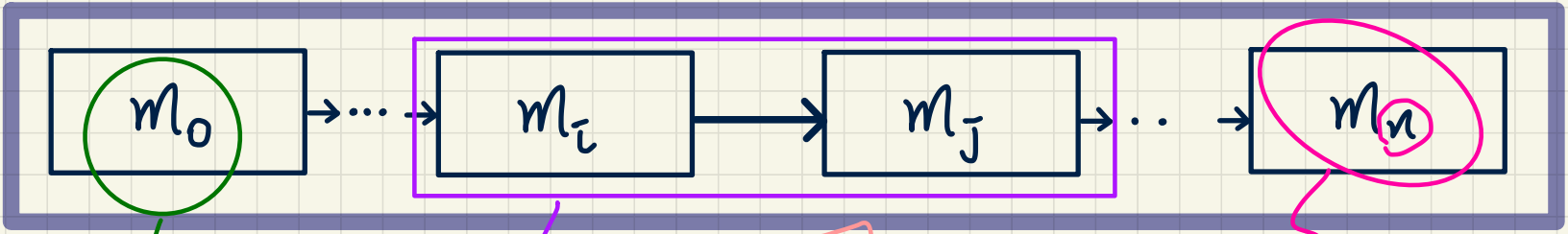
↓

not necessarily

↳ need to convince whoever you're presenting the AC to.

Correct by Construction

No no: a single model containing all details

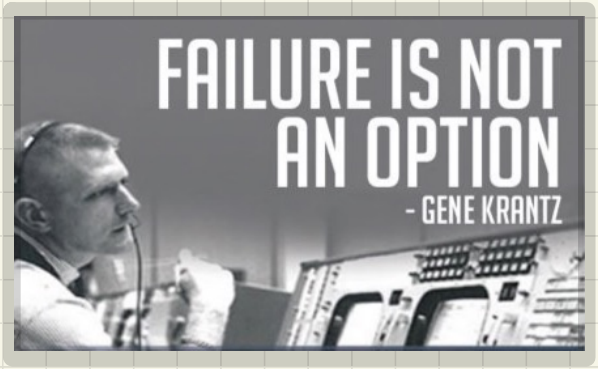


most abstract model

m_i is refined by m_j

↳ added with more details, variables.

most concrete model (closest to final implementation)

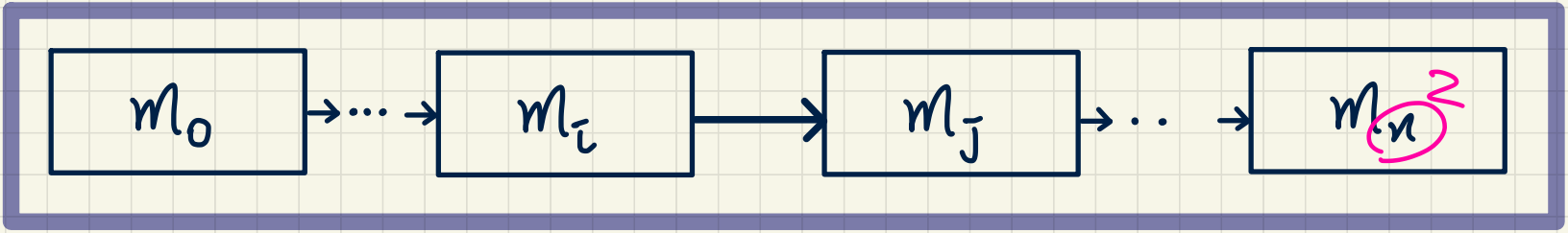


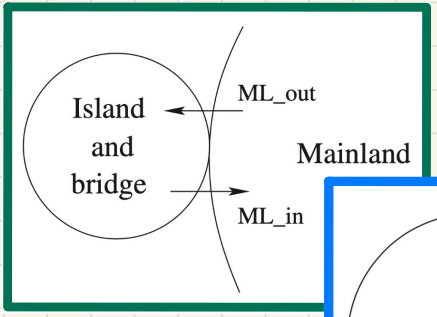
Proof obligations

1. deadlock freedom
2. invariant establishment & maintenance
3. livelock freedom (variant)

Source: https://audiobookstore.com/audiobooks/failure-is-not-an-option-1.aspx

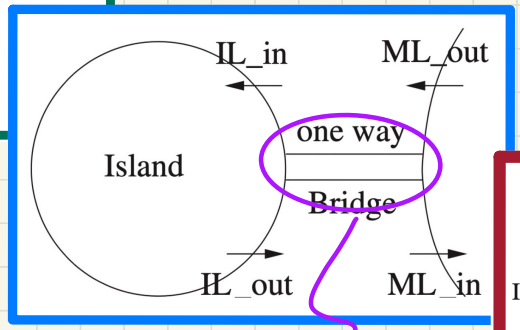
Correct by Construction: Bridge Controller System





m0

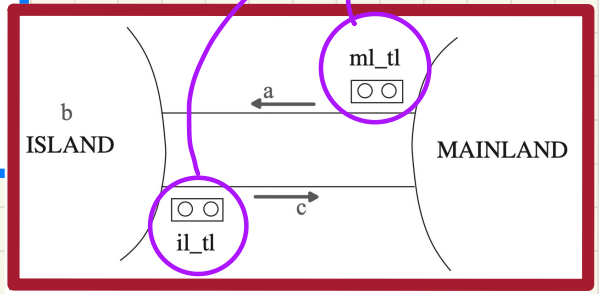
initial abstraction



m1

added concreteness

further added concreteness



m2

state explosion problem.

State Space of a Model

e.g. $C : \{1, 2, 3\}$ — type of constant

[valuations] of $C = |\{1,2,3\}| = 3$

Definition: The state space of a model is the set of all possible valuations of its declared constants and variables, subject to declared constraints.

the more valuations of variables/constants, the more computationally demanding it's to check properties

DFS on graph

upper bound

lower bound

Say an initial model of a bank system with two constants and a variable:

$c \in \mathbb{N}1 \land L \in \mathbb{N}1 \land accounts \in String \pfun \mathbb{Z}$ — set of all possible p.f. /* typing constraint */

$\forall id \bullet id \in \mathrm{dom}(accounts) \Rightarrow -c \leq accounts(id) \leq L$ /* desired property */

Q1. Give some example configurations of this initial model's state space.

$\langle C = 100,\ L = 500,\ accounts = \{\} \rangle$

$\langle C = 5000,\ L = 10000,\ accounts = \{\text{"bill"} \mapsto 6000\} \rangle$

$C = 200$, $L = 600$, owns = {"bill", "steve"}

Q2. How large exactly is this initial model's state space? infinite!

$C \in \mathbb{N}1$ → card is ∞

$L \in \mathbb{N}1$

$accounts \in String \pfun \mathbb{Z}$ — large # valuations, infinite

To be model checkable, need to restrict the state space. balance: interval.

Exercise: Theorem Proving vs. Model Checking

Variable: state space

An integer counter c

Safety Constraints:

MIN_VALUE <= c <= MAX_VALUE

Unconditional Update:

init: initializes c as zero assuming: 0 >= min.

Conditional Updates:

inc: increments c when ?? c < max

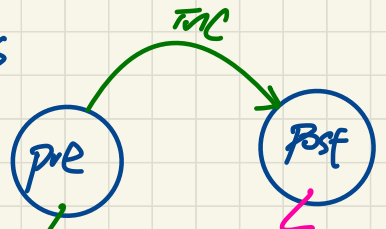
dec: decrements c when ?? c > min

Theorem Proving: Deductive Approach via Inference Rules

1. Formulate the model
2. Discharge proof obligations.

TYC / TYO_1 / INV
TYC / TYO_2 / INV

2. Proof Obligations



machine (dynamic)

1. CONSTANTS

MIN , MAX

AXIOMS

$MIN \in \mathbb{N}$

$MAX \in \mathbb{N}$

$MIN \leq MAX$

context (static)

VARIABLES

c

INVARIANTS

typing constraint

inv0_1 : $c \in \mathbb{N}$

inv0_2 : $MIN \leq c \wedge c \leq MAX$

safety constraint

EVENTS

TYC ✓

when $c < MAX$ — guard / enabling cond.

begin $c := c + 1$ end

dec

when $c > MIN$

begin $c := c - 1$ end