Lecture 2 - January 12

Introduction

Safety- vs. Mission-Critical Systems
Formal Methods
Industrial Standards
Verification vs. Validation

SCS

→ Auto-pilot / auto-driving

traffic light / train gate

air bag deployment

elevator / escalator

Impulse dector / pacemaker

nuclear power plant / shutdown system

OPG

↳ Ontario Power Gen.

↳ Darlington Shutdown Systems.

Precise — math.

↳ no scope of multiple interpretations.

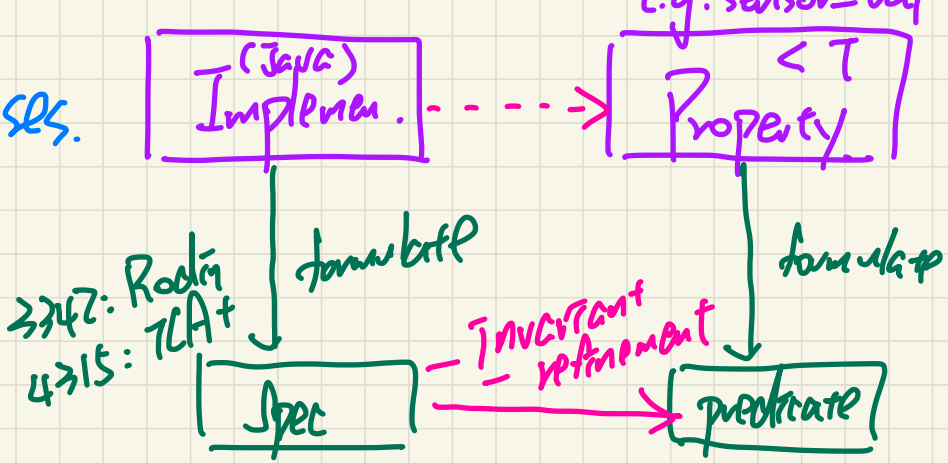
FFS4317

↳ NAT

↳ code: may be precise but too low level!

Complete

↳ no missing cases.



(P_1) System $\leq$ is mission critical

(P_2) System $\leq$ is safety critical

(1) $P_1 \equiv P_2$

(2) $P_1 \Rightarrow P_2$

(3) $P_2 \Rightarrow P_1$ always the case

not true in general

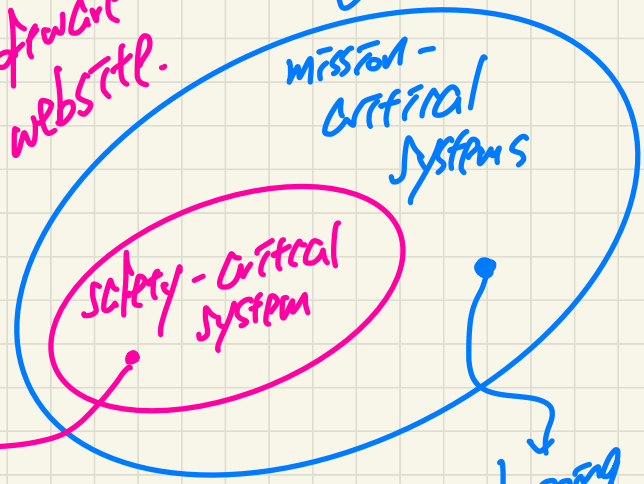
e.g. car gas emission ✓

e.g. manufacture a working software ✓

e.g. financial software website. ✓

shopping

:Pad



Mission-Critical vs. Safety-Critical

Safety critical

When defining safety critical it is beneficial to look at the definition of each word independently. Safety typically refers to being free from danger, injury, or loss. In the commercial and military industries this applies most directly to human life. Critical refers to a task that must be successfully completed to ensure that a larger, more complex operation succeeds. Failure to complete this task compromises the integrity of the entire operation. Therefore a safety-critical application for an RTOS implies that execution failure or faulty execution by the operating system could result in injury or loss of human life.

Safety-critical systems demand software that has been developed using a well-defined, mature software development process focused on producing quality software. For this very reason the DO-178B specification was created. DO-178B defines the guidelines for development of aviation software in the USA. Developed by the Radio Technical Commission for Aeronautics (RTCA), the DO-178B standard is a set of guidelines for the production of software for airborne systems. There are multiple criticality levels for this software (A, B, C, D, and E).

These levels correspond to the consequences of a software failure:

- Level A is catastrophic (most severe)
- Level B is hazardous/severe
- Level C is major
- Level D is minor
- Level E is no effect (least severe)

Safety-critical software is typically DO-178B level A or B. At these higher levels of software criticality the software objectives defined by DO-178B must be reviewed by an independent party and undergo more rigorous testing. Typical safety-critical applications include both military and commercial flight, and engine controls.

Mission critical

A mission refers to an operation or task that is assigned by a higher authority. Therefore a mission-critical application for an RTOS implies that a failure by the operating system will prevent a task or operation from being performed, possibly preventing successful completion of the operation as a whole.

Mission-critical systems must also be developed using well-defined, mature software development processes. Therefore they also are subjected to the rigors of DO-178B. However, unlike safety-critical applications, mission-critical software is typically DO-178B level C or D. Mission-critical systems only need to meet the lower criticality levels set forth by the DO-178B specification.

Generally mission-critical applications include navigation systems, avionics display systems, and mission command and control.

3347 : theorem proving
4315 : model checking.

Source: http://pdf.cloud.opensystemsmedia.com/advancedtca-systems.com/SBS.Jan04.pdf

safety (invariant) property

I ✓

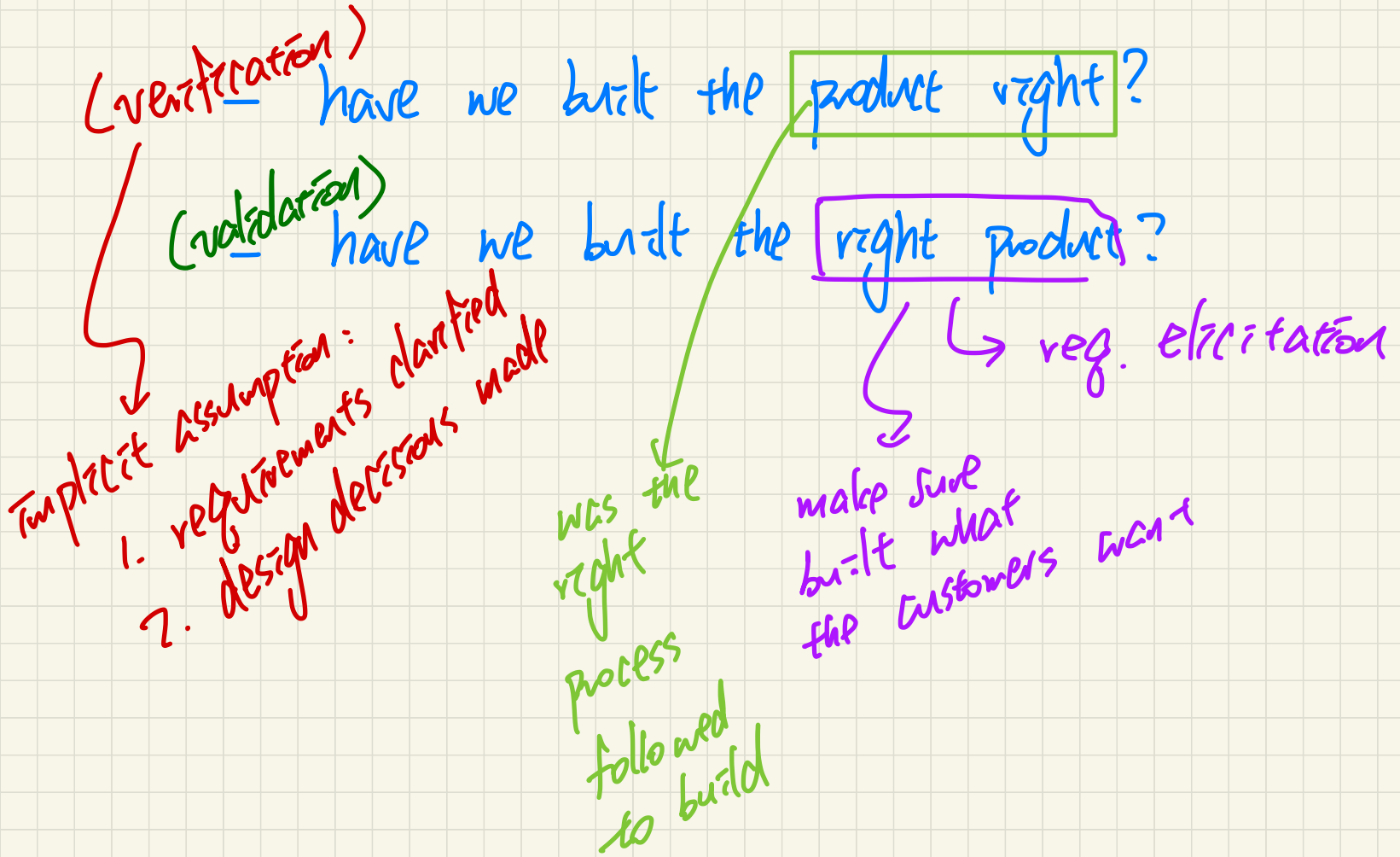
(mathematical induction).

↳ weak

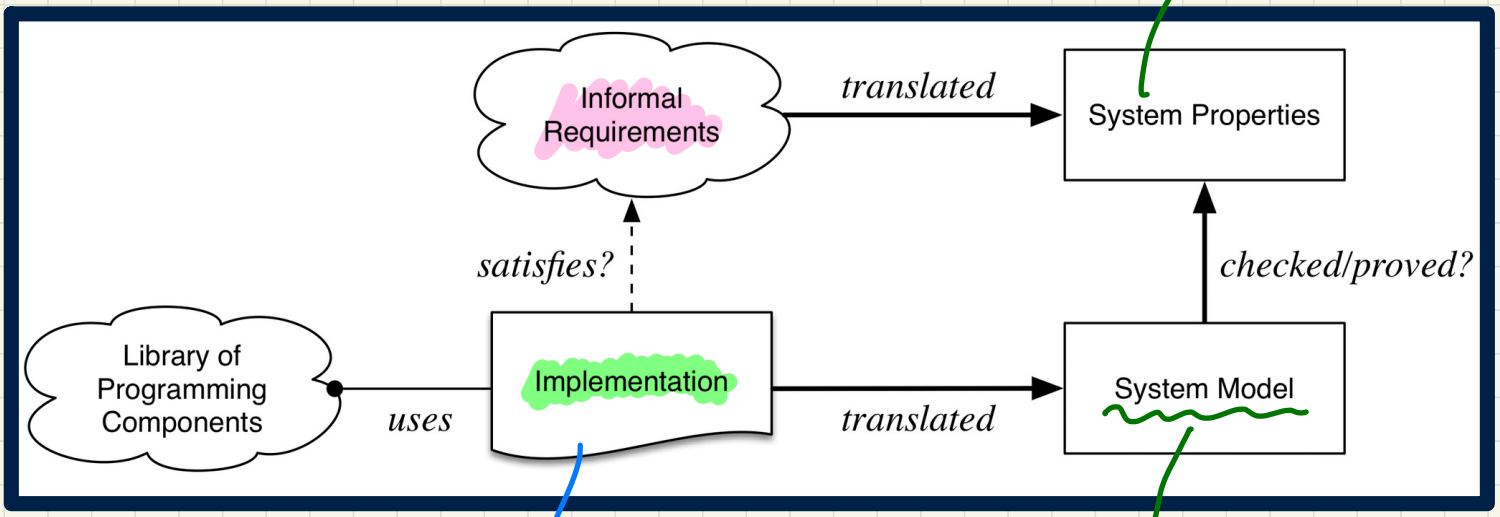
assume I satisfied in S_i

prove I satisfied in S_{i+1}

according to before-after predicate of MLalt.


(verification) have we built the product right?
(validation) have we built the right product?
Implicit assumption:
1. requirements clarified
2. design decisions made
was the right process followed to build
make sure built what the customers want
req. elicitation

Building the product right?



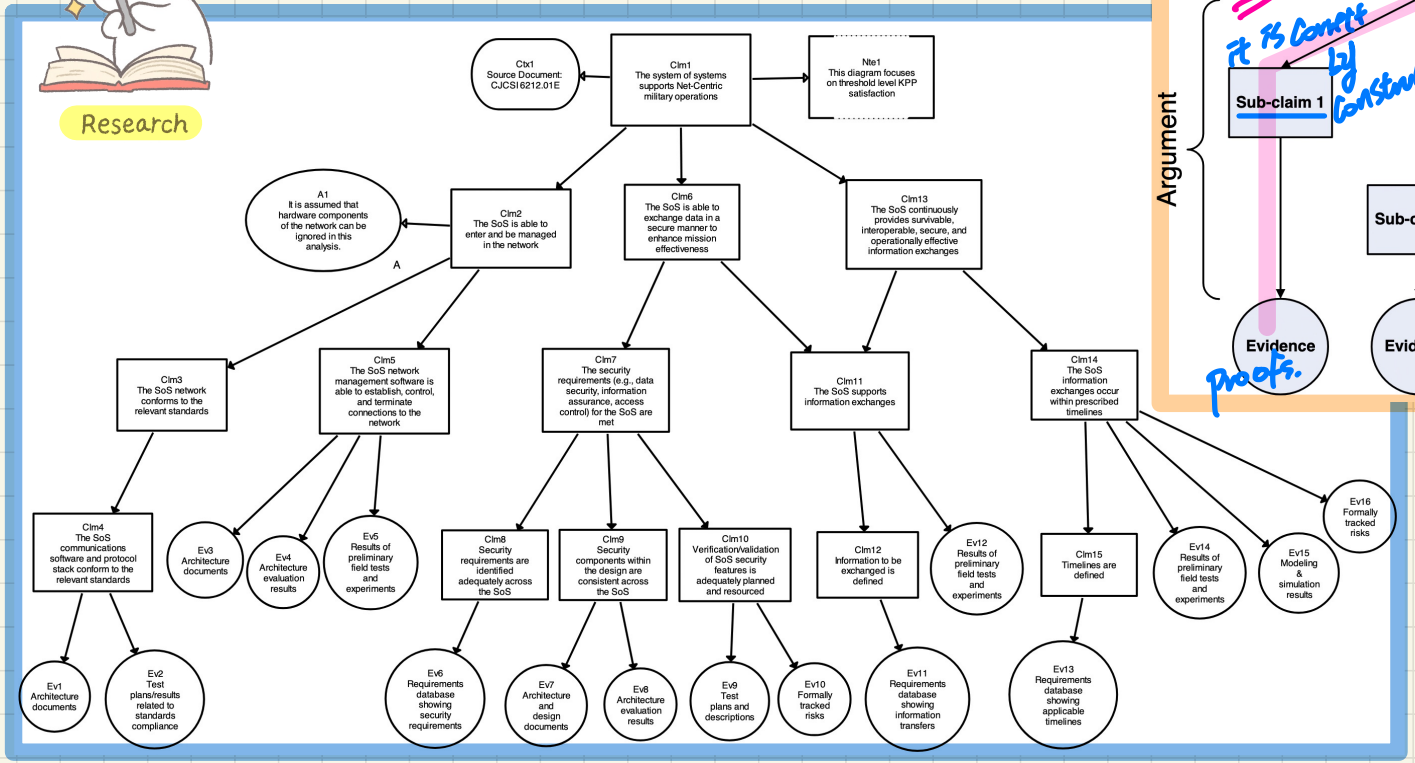
predicates

1. depends on other components
2. those components should be certified!

abstract state machine

Certifying Systems: Assurance Cases

Research on "Assurance Cases" if interested!



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