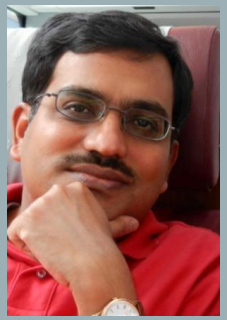
REQUIREMENTS

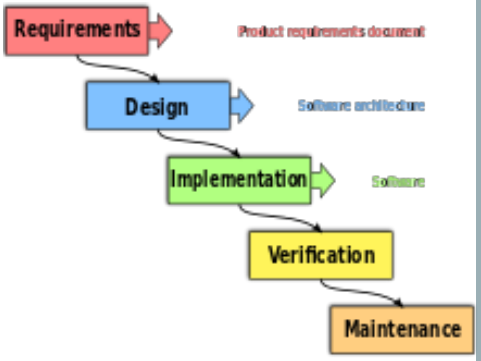
CS3213 FSE

Prof. Abhik Roychoudhury

National University of Singapore

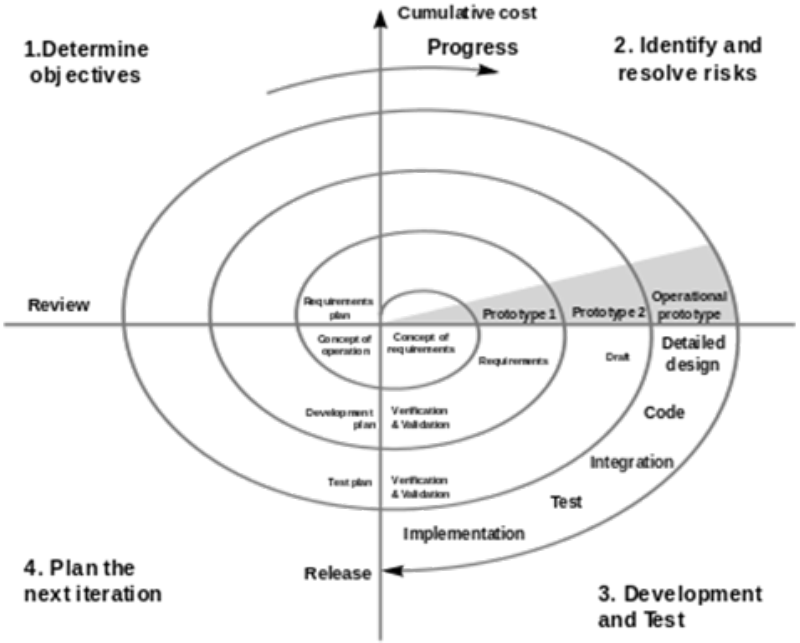


SOFTWARE ENGINEERING



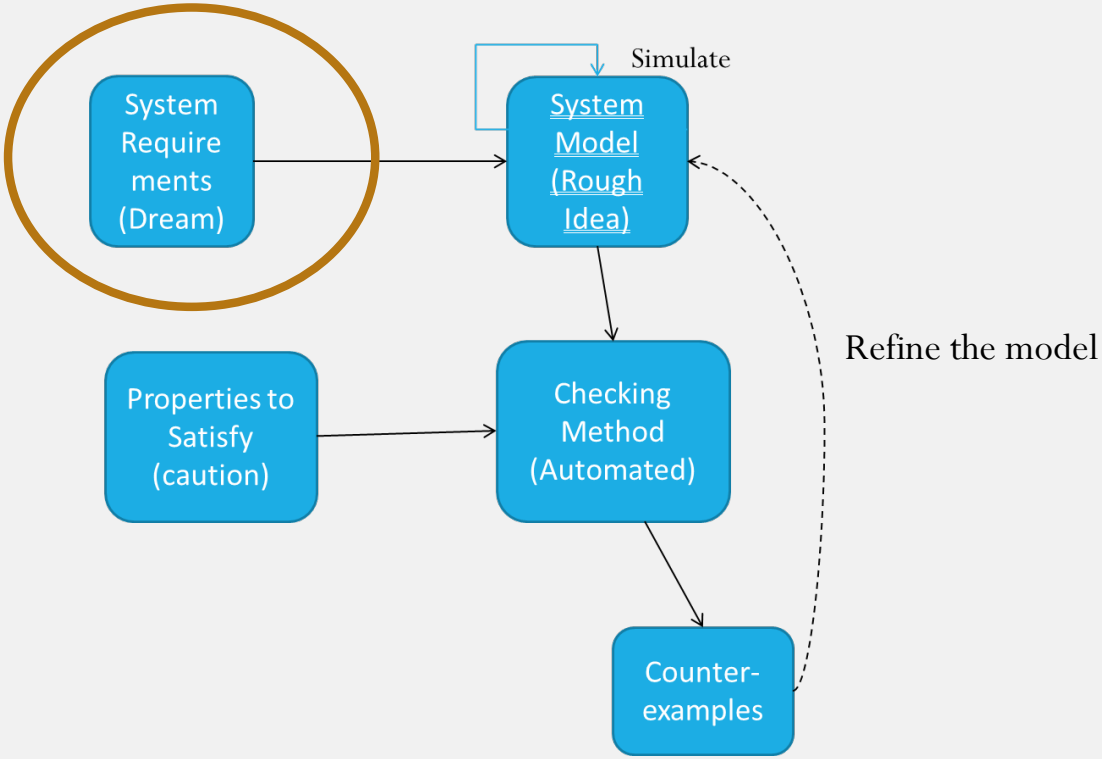
Waterfall Model

SOFTWARE ENGINEERING



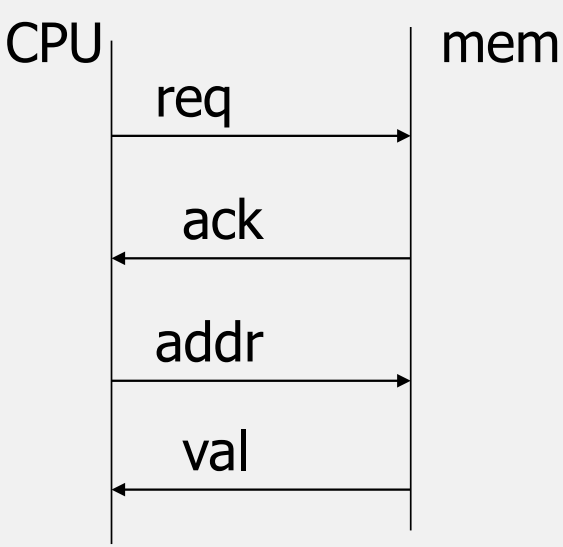
Spiral Model

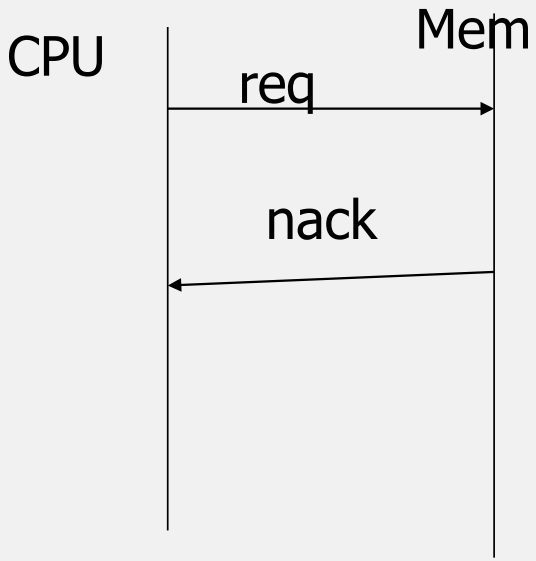
GETTING STARTED: LOOKING INSIDE



Message sequence chart.

MSC - VISUALLY





Used to specify requirements in early stages of system design.
A MSC depicts only a possible scenario of system behavior.

MSC ORDER

Doesn't have to be in the same process.
(It's just one of possible situations).

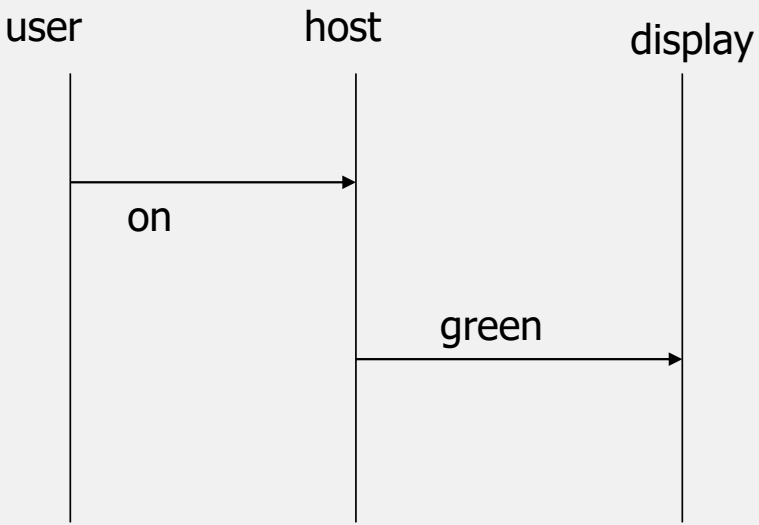
- Intuitively, a "happens-before" relation between events.
- e < e' if the MSC requires e to happen before e' 2 rules.
  - If e, e' occur in the same process and e occurs above e' then e < e'
  - If e, e' are the send and receive of the same message then e < e'
- Think of our partial order as the transitive closure of the above orderings.
  - MSC = labeled partial order over events

DESIGNING BEHAVIORAL REQUIREMENTS

- MSCs - Weak form of requirements
  - System components and their architecture typically known during req. design.
    - This can also be gradually refined.
  - Interaction among components understood better during requirement design.
    - Iterative process often involving the end user.
    - Possible and Illegal system scenarios

uml sequence diagram only describes possible scenarios, not necessarily must happen

POSSIBLE BEHAVIOR

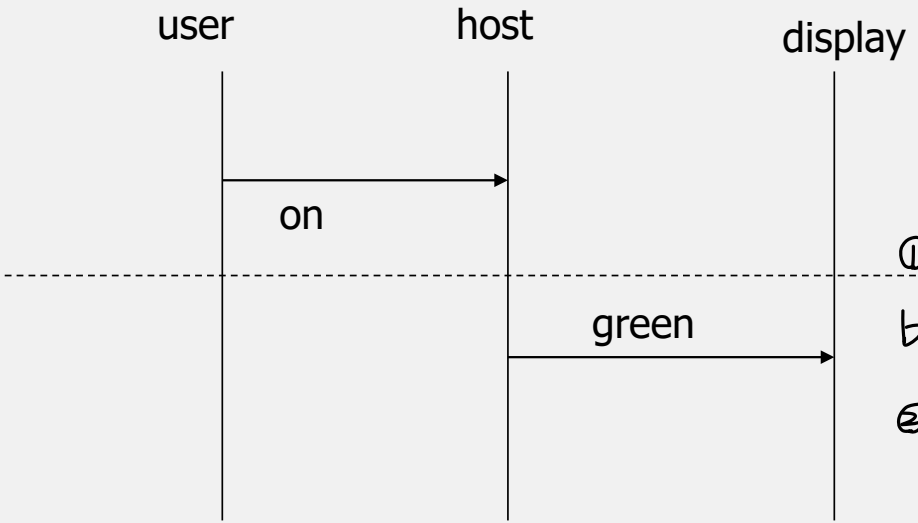


The user may switch on the host following by the host turning the display to green.

possible behavior, not anything illegal.

ILLEGAL BEHAVIOR

How to specify such requirements ?

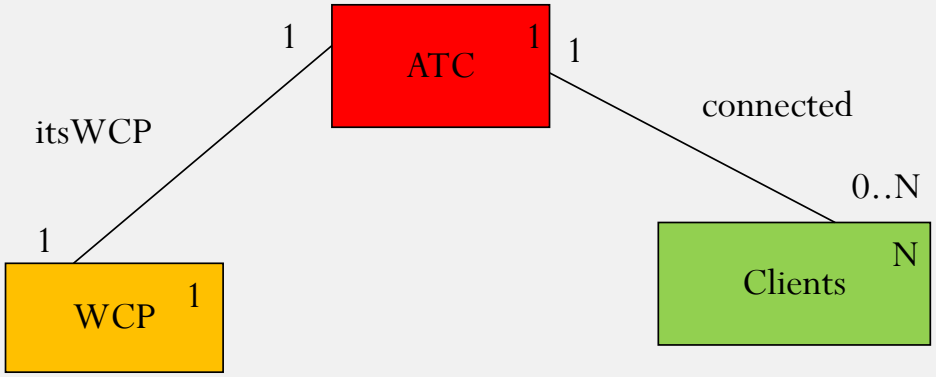


separation (dashed line):
① when top of line happens,
bottom of line must happen eventually.
② Can describe illegal scenarios.

Whenever the user switches on the host , the host must turn the display to green.

illegal not to turn display to green.

RUNNING EXAMPLE - ATC



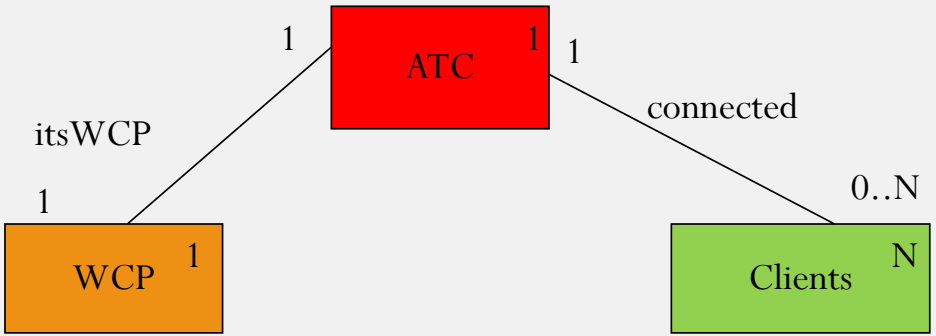
Overall System Structure, Behavior not shown.

ON SYSTEM BEHAVIOR

- Consider a "scenario"
  - Client1 sends "connect" request to ATC
  - Client2 sends "connect" request to ATC
  - ATC sends weather information to Client1, Client2.
- No need to capture "weather info." in model.
- OK to abstract this info. from the requirements while constructing the model, provided
  - No decisions are made in the system based on weather info.
- Model is "complete" at a certain level of abstraction.

ATC – EXAMPLE CONTROL SYS.

- NASA CTAS
  - Automation tools for managing large volume arrival air traffic in large airports.
  - Final Approach Spacing Tool
    - Determine speed and trajectory of incoming aircrafts on their final approach.
    - Master controller updates weather info. to "clients"
      - **controllers using inputs to compute aircraft trajectories.**



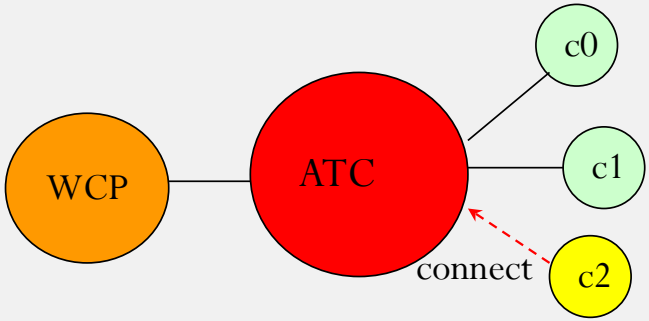
ATC – CONTROL SYS.

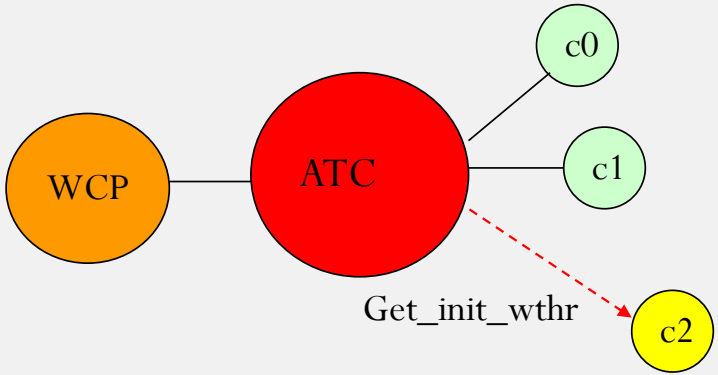
- Part of the *Center TRACON Automation System (CTAS)* by NASA
  - manage high volume of arrival air traffic at large airports
  - **http://ctas.arc.nasa.gov**
- Control weather updating to all weather-aware clients
  - A weather control panel (WCP)
  - Many weather-aware clients
  - A communication manager (CM)

BEHAVIOR OF ATC EXAMPLE

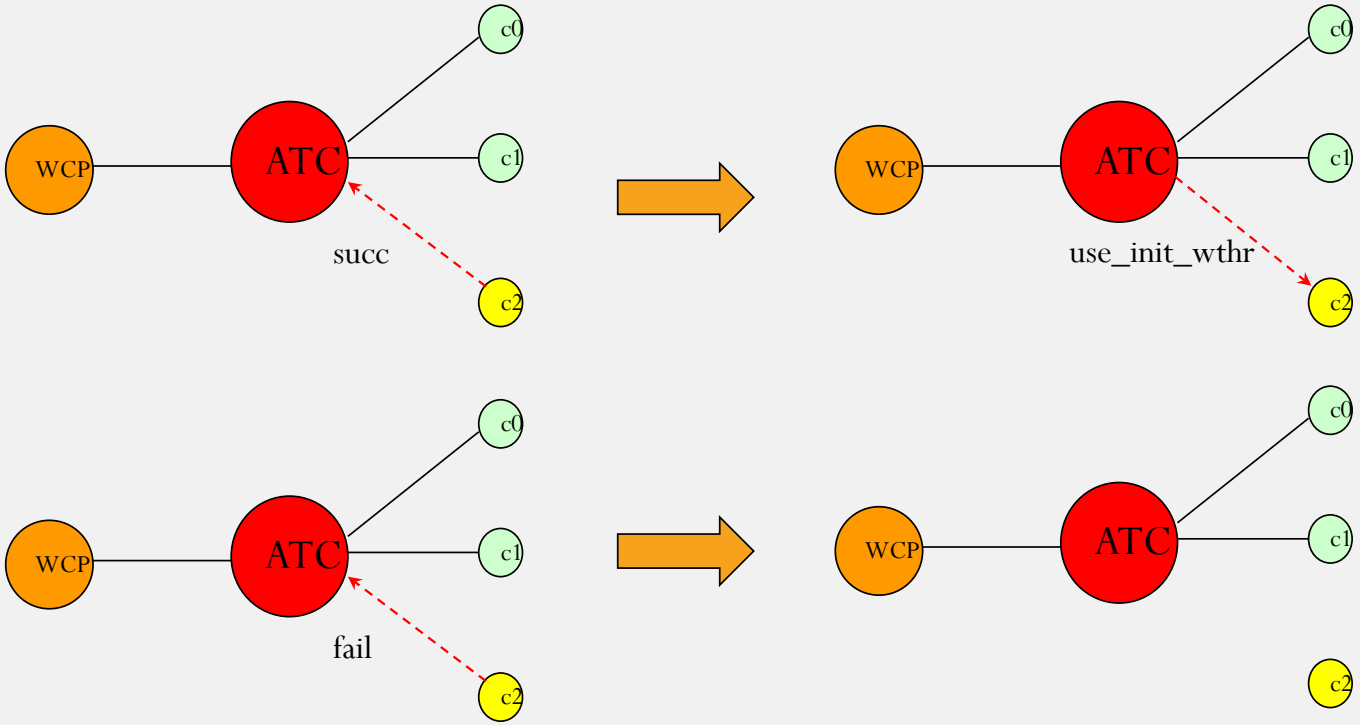
- Two standard behaviors
  - Client initialization
  - Weather update
- Abstracted Information
  - Weather information types
  - Client types
  - Internal computation on weather information

CLIENT INITIALIZATION

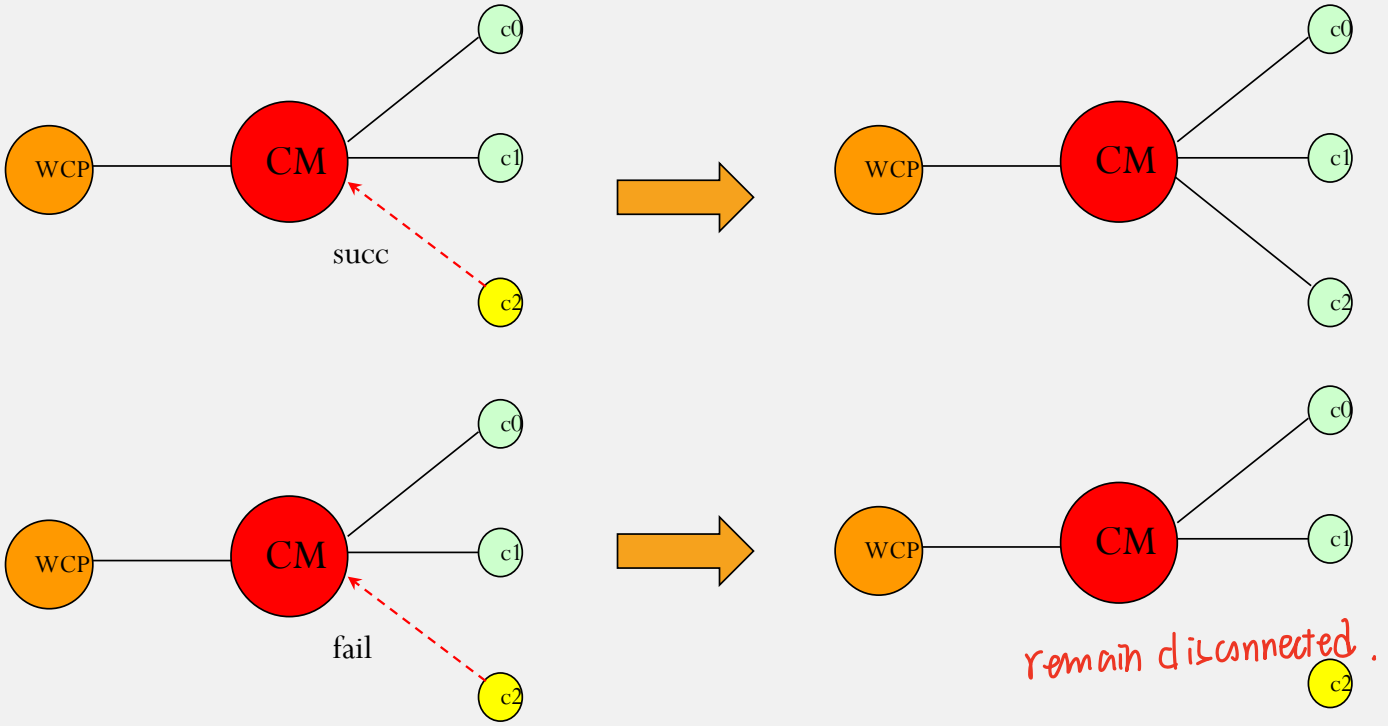




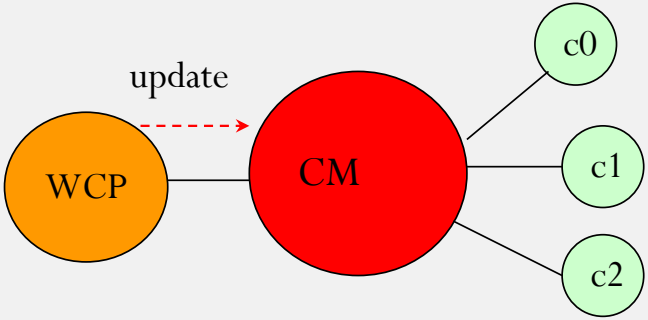
CLIENT - INITIALIZATION

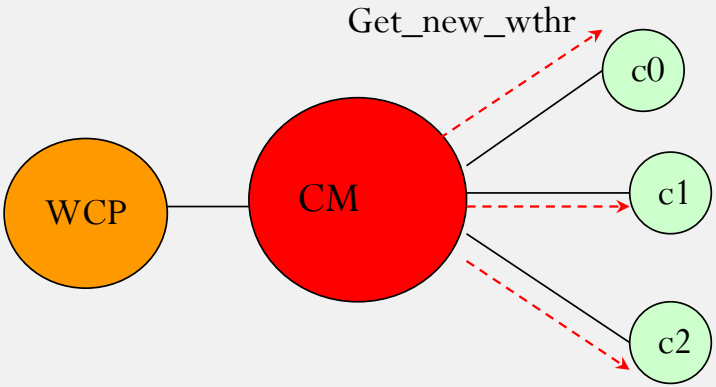


CLIENT - INITIALIZATION

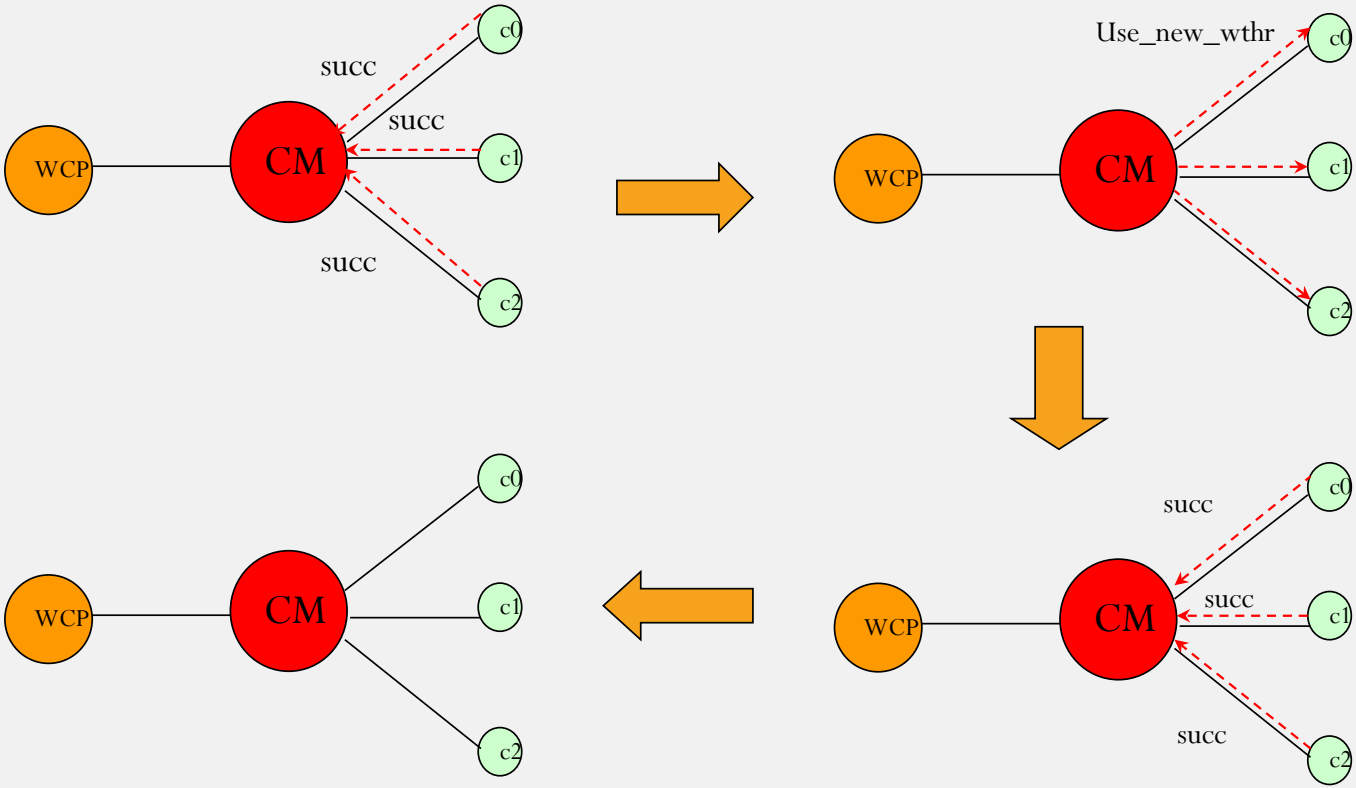


CLIENT – WEATHER UPDATE





CLIENT UPDATE – CASE 1



WHAT DO THE REQUIREMENTS

- … look like ?

A weather update controller consists of a weather control panel (WCP), a number of weather-aware clients, and a communication manager (ATC) which controls the interactions between the WCP and all connected clients. Initially, the WCP is enabled for manually weather updating, the ATC is at its idle status, and all the clients are disconnected. Two standard behaviors of this system are as follows.

-> **see 1 PAGE REQUIREMENTS DOCUMENT IN Luminus**

See other cases from Requirements document in Luminus.

How to translate such structured requirements to **executable descriptions**

-> *Live Sequence Charts*

SAMPLE REQUIREMENTS

- A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.

- *If the ATC's status is idle when the connecting request is received, it will set both its own status and the connecting client's status to pre-initializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization.*
  - Otherwise (ATC's status is not idle), the ATC will send a message to the client to refuse the connection, and the client remains disconnected.

REQUIREMENTS DOCUMENT

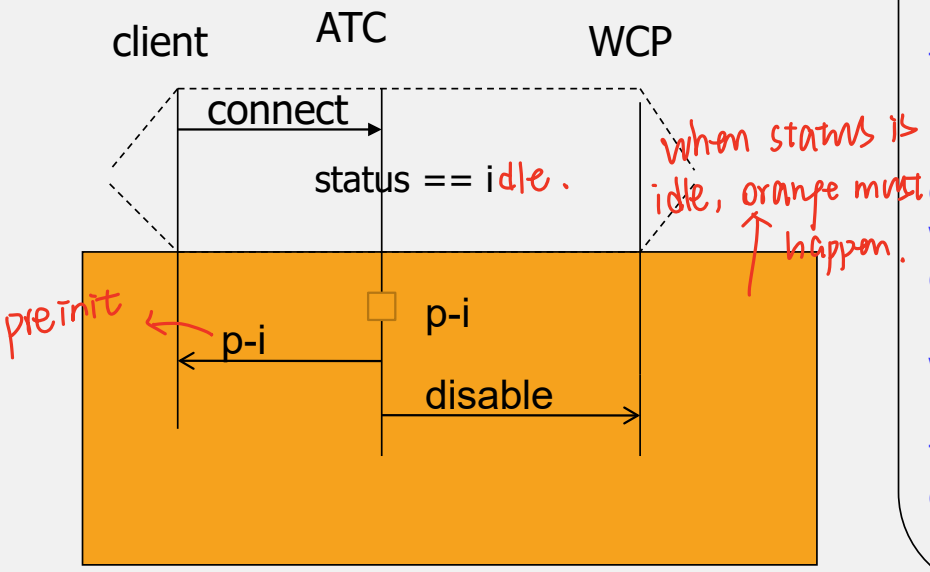
- *If the ATC's status is idle when the connecting request is received, it will set both its own status and the connecting client's status to pre-initializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization.*

- Sample requirement document has been provided in Luminus.
  - Check it out now!

- -> How to translate such structured requirements to models
- -> *Live Sequence Charts*

REQUIREMENTS ENCODING

visualization of requirements.



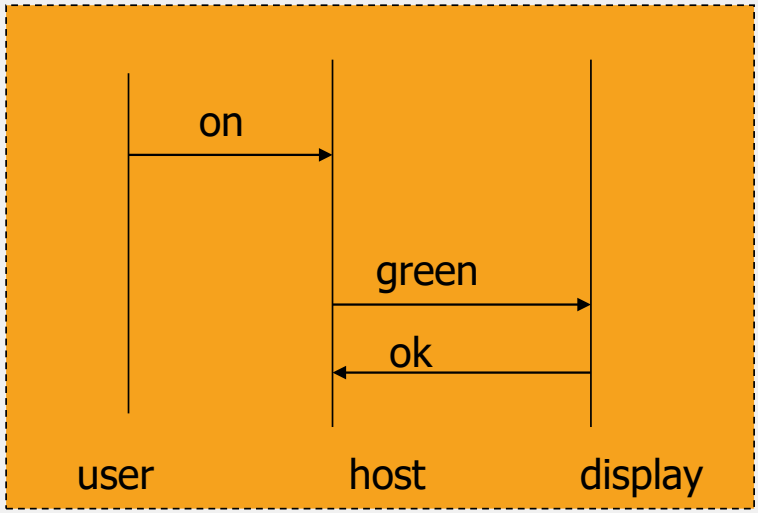
A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.

first recv. req.; second check status

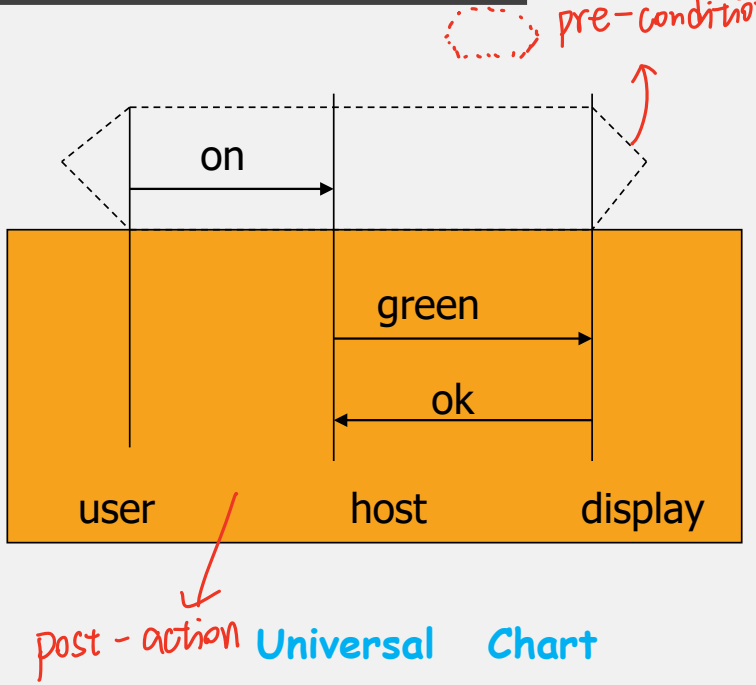
If the ATC's status is idle when the connecting request is received, it will set both its own status and the connecting client's status to preinitializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization.

Can execute these charts to check consistency across different requirement rules in a big requirements document !!

LIVE SEQUENCE CHARTS

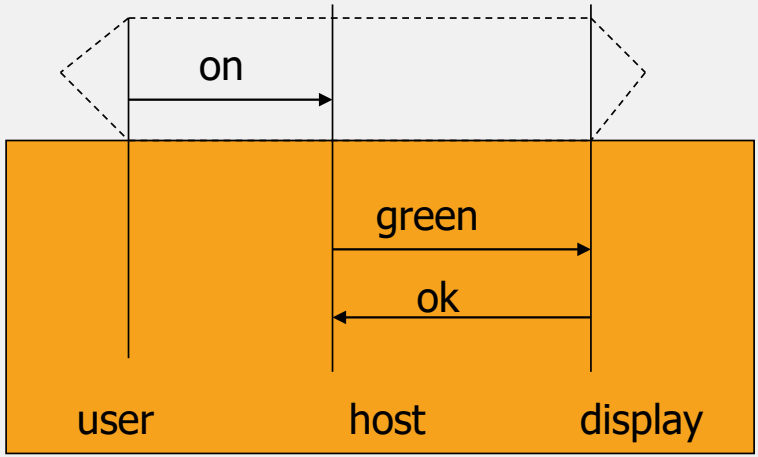


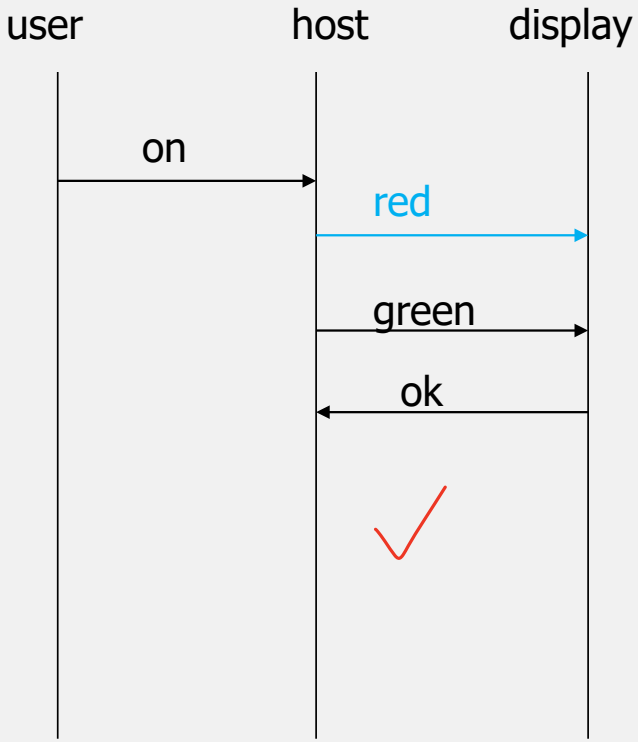
Existential Chart



Universal Chart

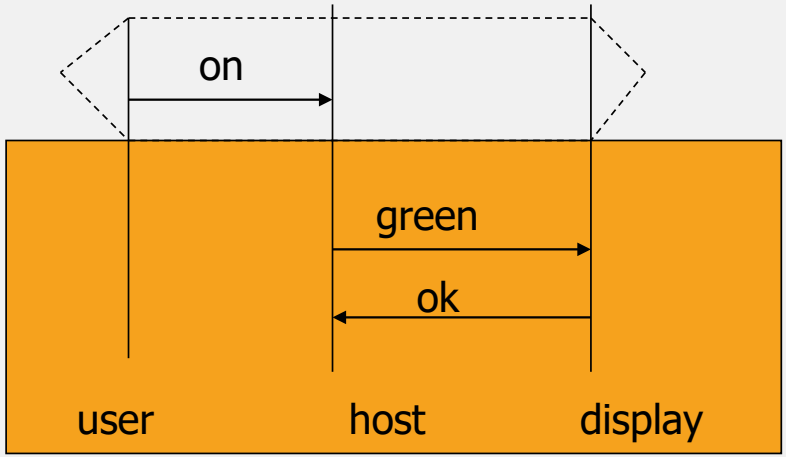
SATISFYING A UNIVERSAL CHART

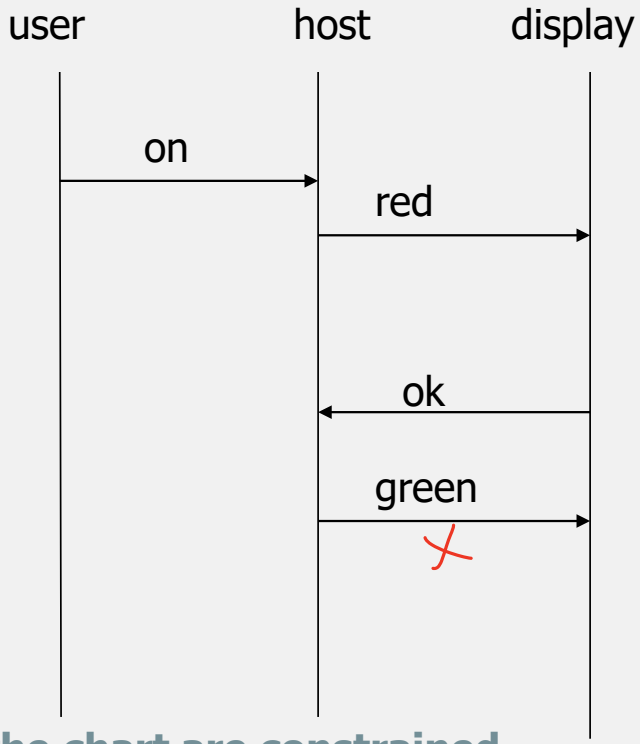




Messages not appearing in the chart are not constrained

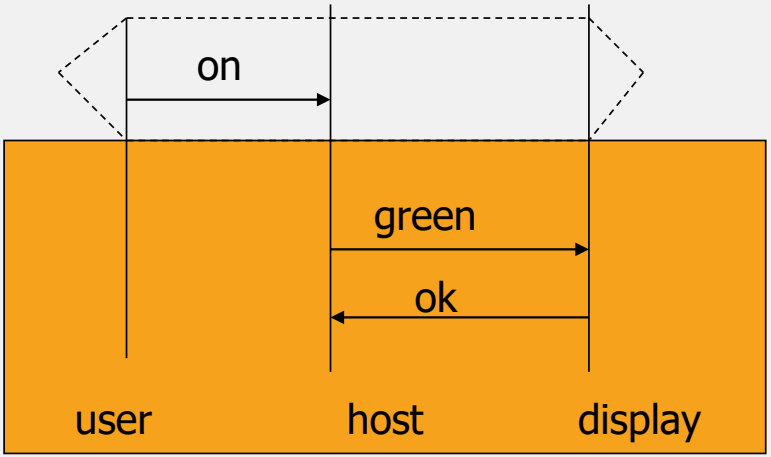
VIOLATING A UNIVERSAL CHART

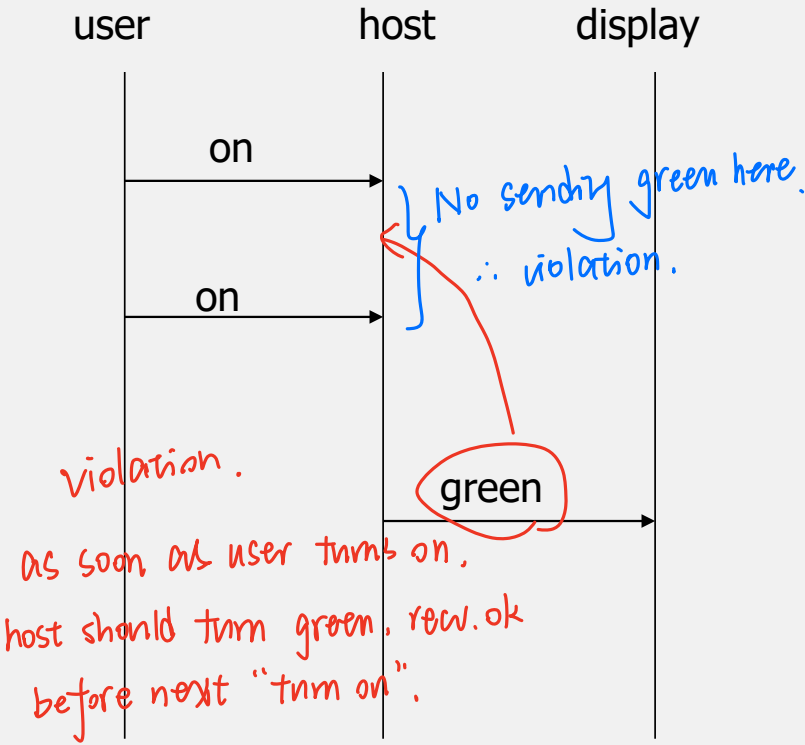




Messages appearing in the chart are constrained

VIOLATING A UNIVERSAL CHART



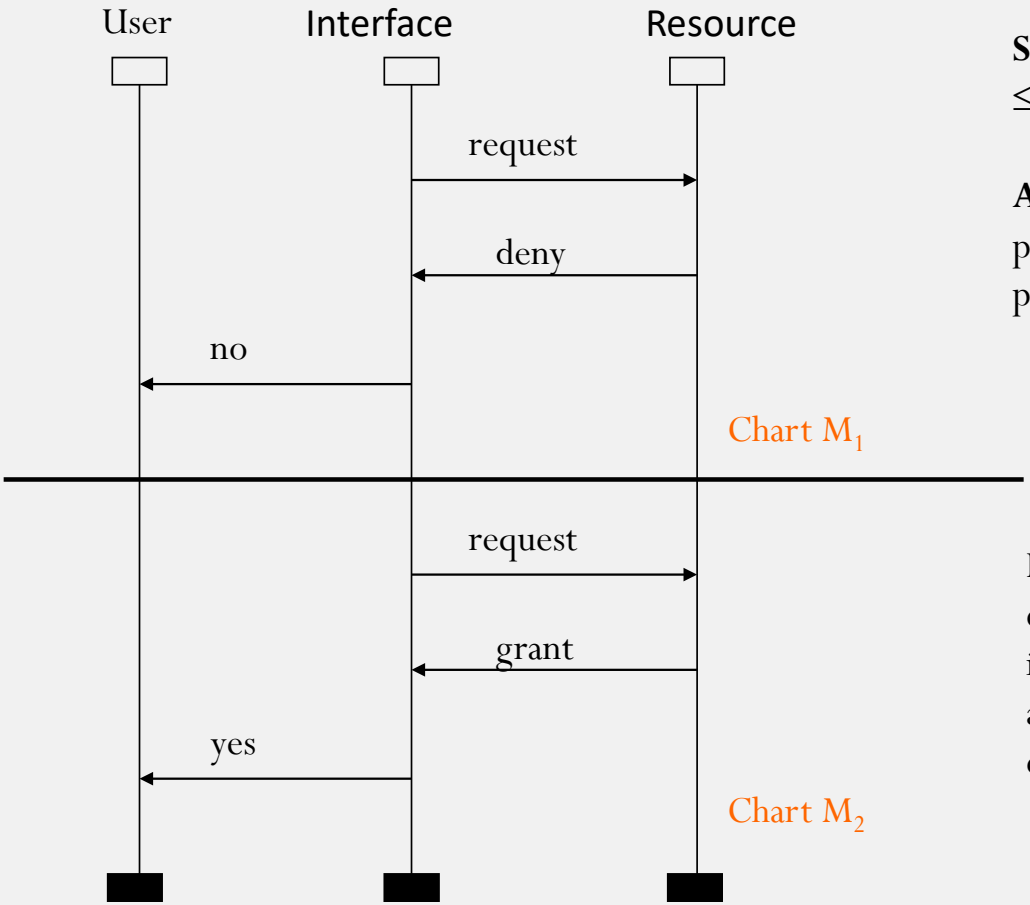


Messages appearing in the chart are constrained

PRELIMINARY ISSUES

- Synchronous or Asynchronous
  - Messages
    - Individual messages in a LSC can be specified as synchronous or asynchronous.
    - We assume asynchronous messages.
  - (Pre-chart, Body-chart)
    - LSC prescribes synchronous concatenation
    - Body chart cannot begin until Pre-chart ends.

MSC CONCATENATION



Synchronous: All events in M1 ≤ All events in M2

Asynchronous: All events in process p of M1 ≤ All events in process p of M2

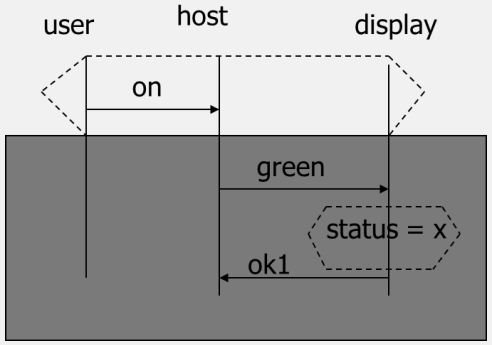
Interface and Resource processes can finish M2 while User process is still in M1 – provided asynchronous concatenation is considered.

COMBINING HOT AND COLD REQ.

- At chart level
  - Hot
    - If pre-chart is completed, body-chart must hold in all system executions
  - Cold
    - An existential chart may hold in some system execution
- … and at lower levels of granularity !

Universal chart:
contains a pre-chart, which specifies the scenario which, if successfully executed, forces the system to satisfy the scenario given in the actual chart body.

Existential chart:
specify sample interactions between the system and its environment, that must be satisfied by at least one system run.



EXECUTING LSCS

- LSC specification
  - A collection of universal charts
  - Existential charts are only monitored.
- Overall Presentation
  - Naïve execution or play-out
  - Smart play-out
    - Finding one violation-free system response
    - Also consider existential charts

EXECUTING LSCS

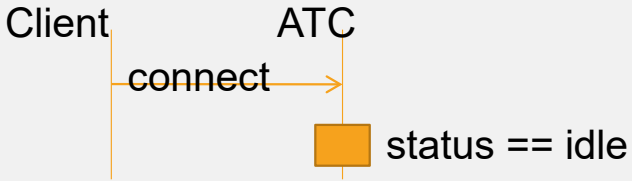
- Why should we execute them ?
  - Find inconsistencies across charts
  - Test out possible system behaviors
    - Which behaviors lead to violation of a chart ?
    - Does there exist any simulation run which avoids violation of the charts ?
- Thus, testing out system behaviors
  - Directly from inter-component requirements.
  - No need to synthesize state-charts from inter-component requirements

SAMPLE INITIALIZATION REQUIREMENTS – CTAS (LUMINUS)

- A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.
- If the ATC's status is idle when the connecting request is received, it will set both its own status and the connecting client's status to preinitializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization.
  - Otherwise (ATC's status is not idle), the ATC will send a message to the client to refuse the connection, and the client remains disconnected.
- Try to construct MSCs to describe these scenarios?

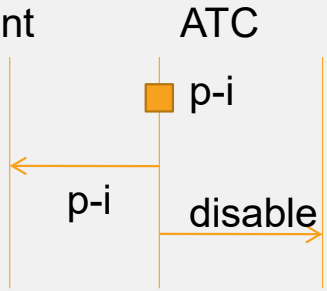
SAMPLE MSCS

Pre-condition: Connect request from Client to ATC, ATC status == idle

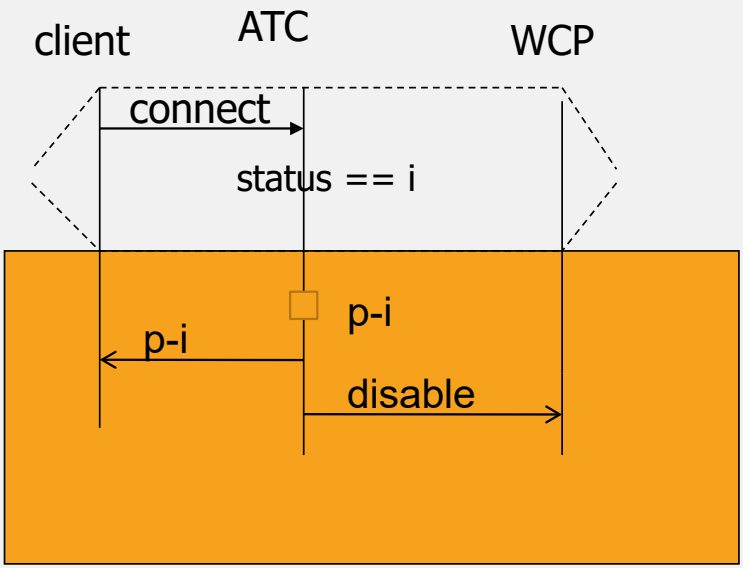


How to represent this check?

Behavior:



LSC ENCODING



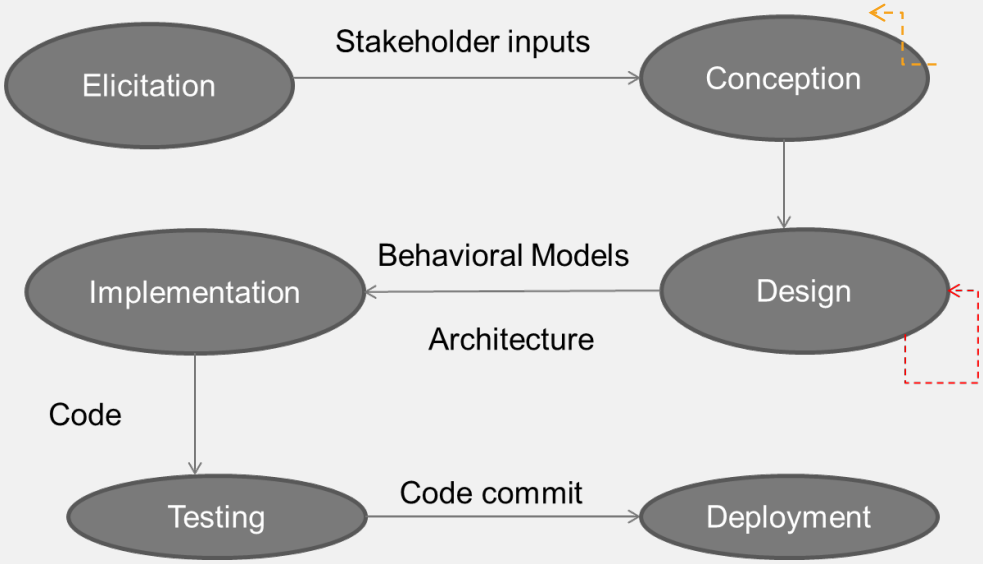
A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.

If the ATC's status is idle when the connecting request is received, it will set both its own status and the connecting client's status to preinitializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization.

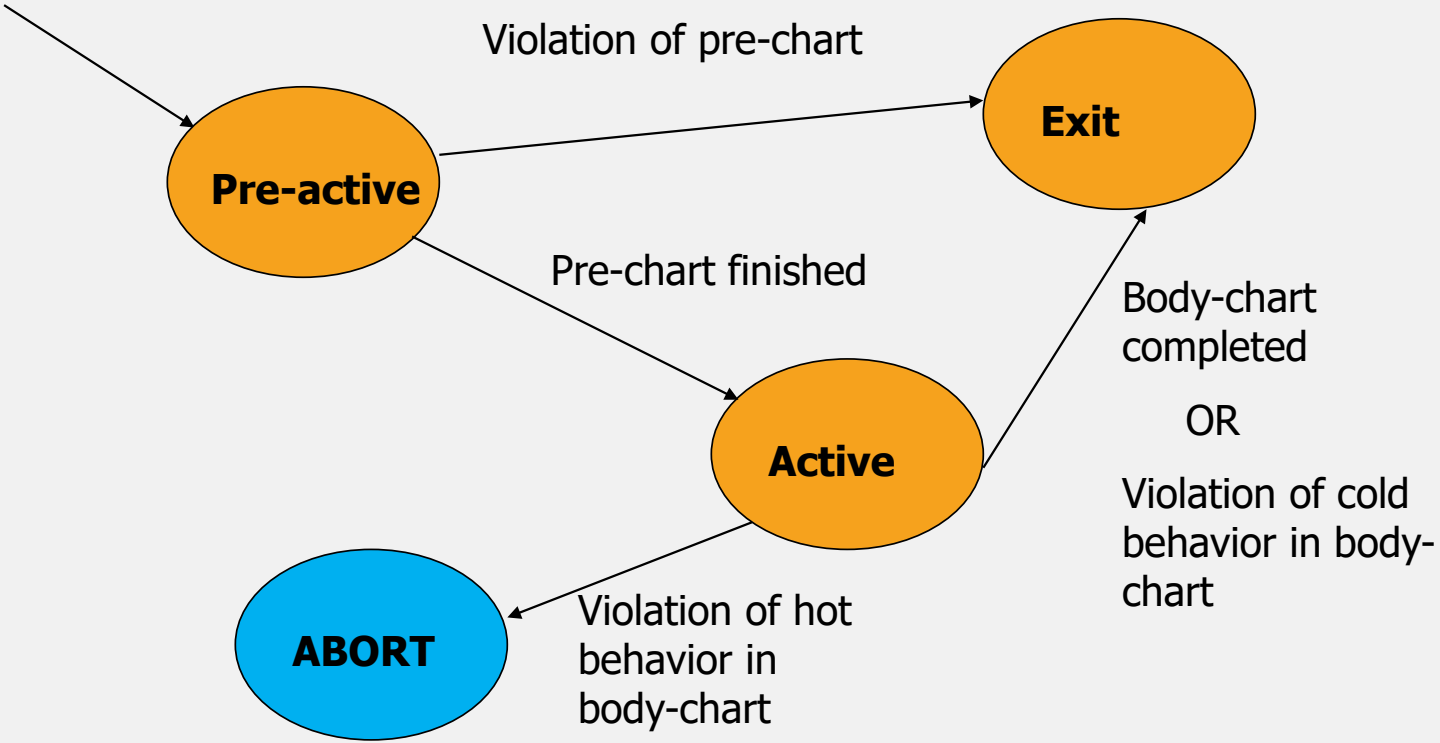
Can execute these LSCs to check consistency across different requirement rules in a big requirements document !!

BIG PICTURE

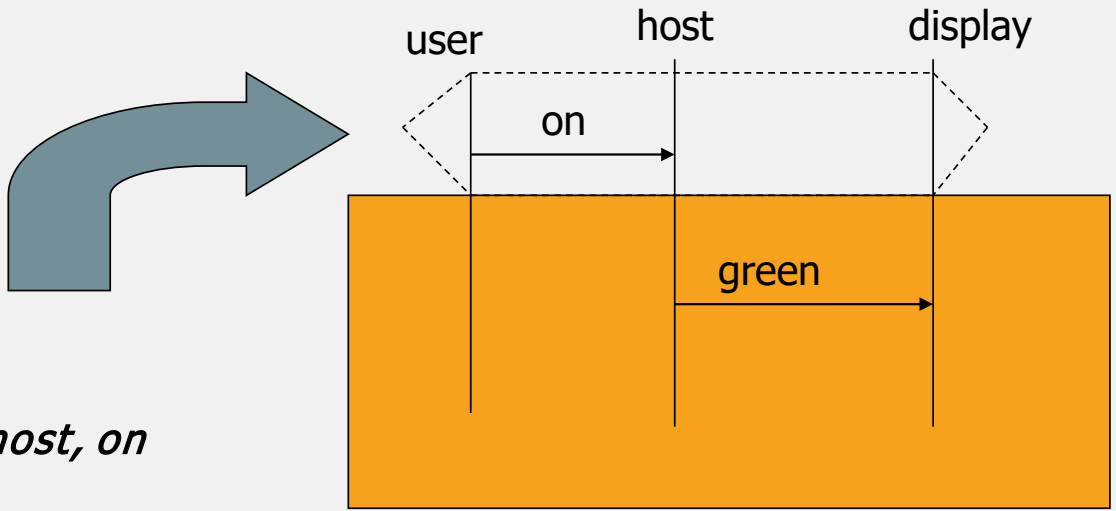
- *Big-picture of how LSCs fit into requirements capture*



LIFE-CYCLE OF A UNIV.CHART COPY



PLAY-OUT: EXAMPLE 1

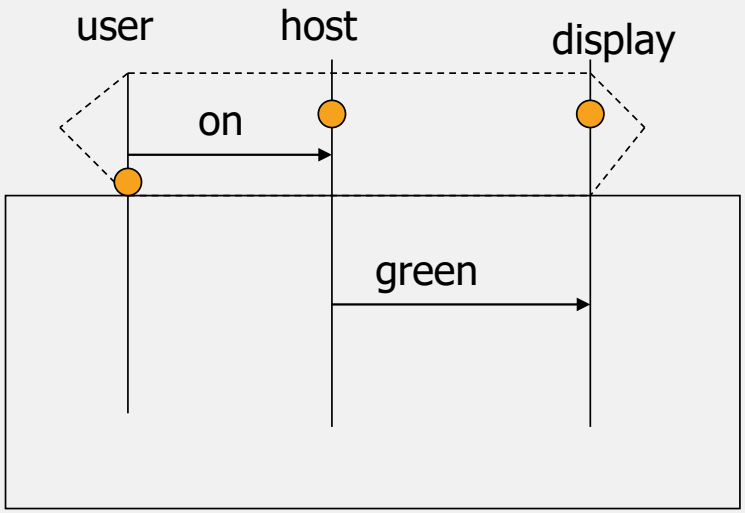


Stimulus

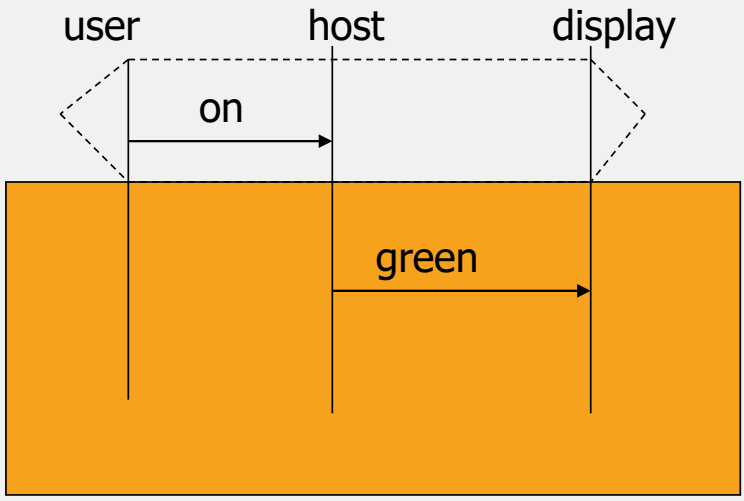
user ! host, on

PLAY-OUT: EXAMPLE I

Pre-active copy

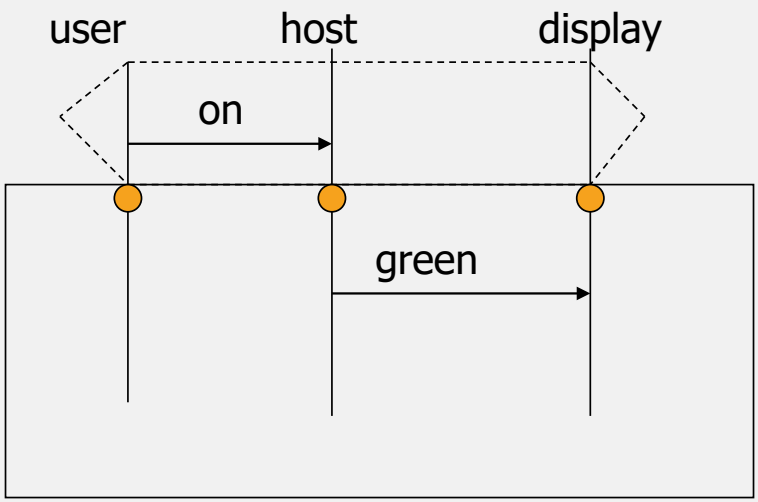


LSC Specification

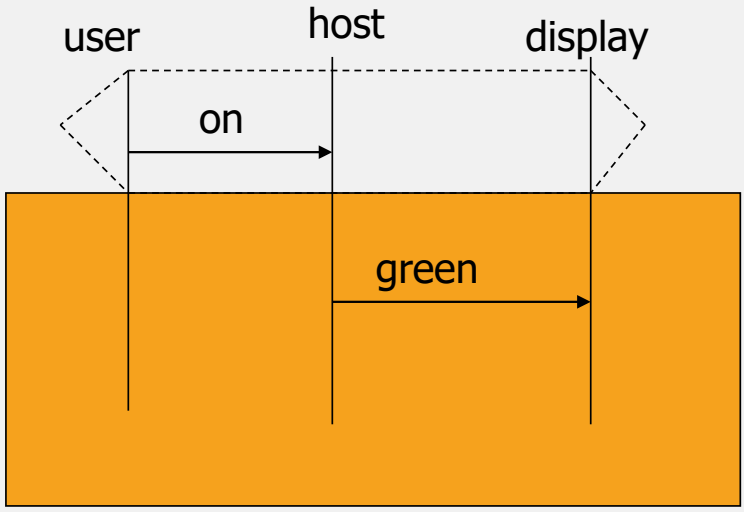


PLAY-OUT: EXAMPLE I

Active copy

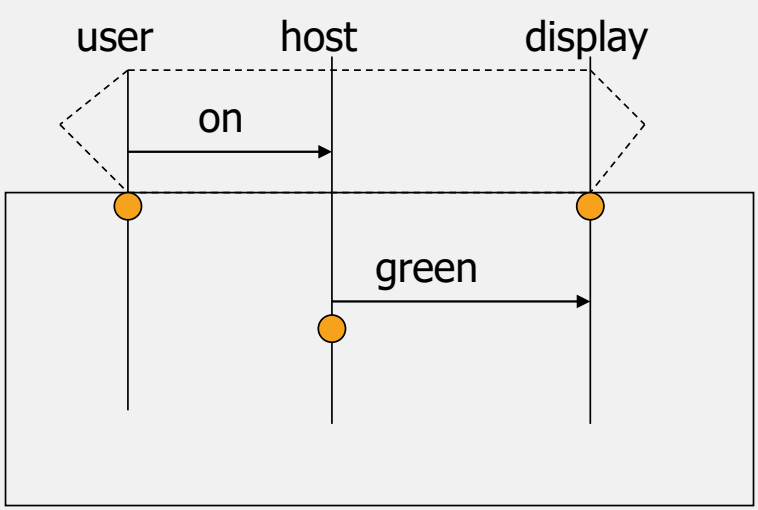


LSC Specification

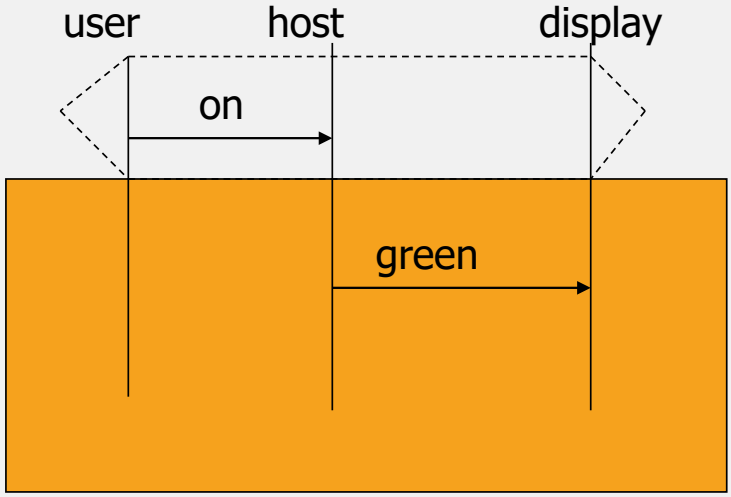


PLAY-OUT: EXAMPLE I

Active copy

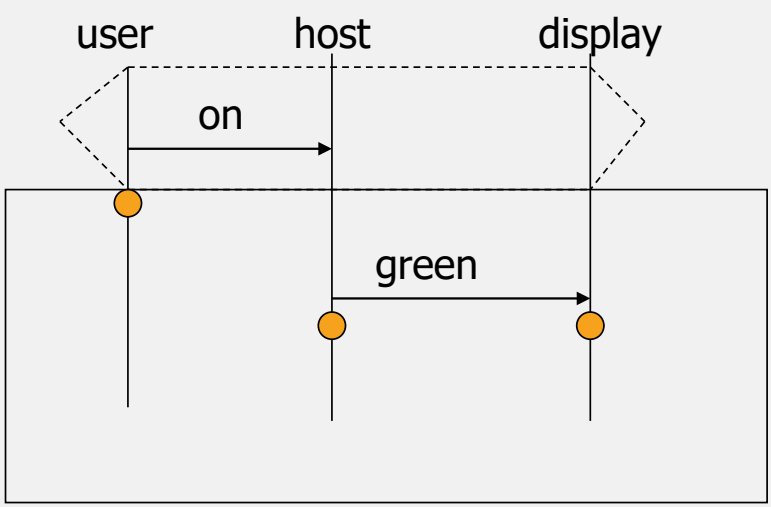


LSC Specification

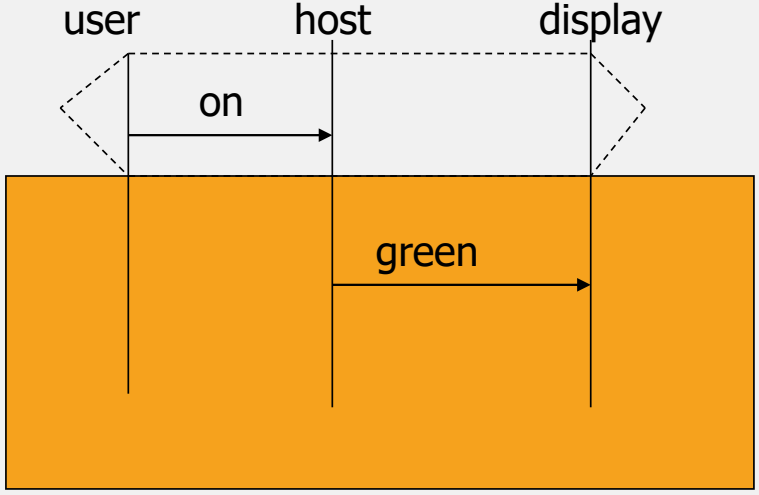


PLAY-OUT: EXAMPLE I

Active copy

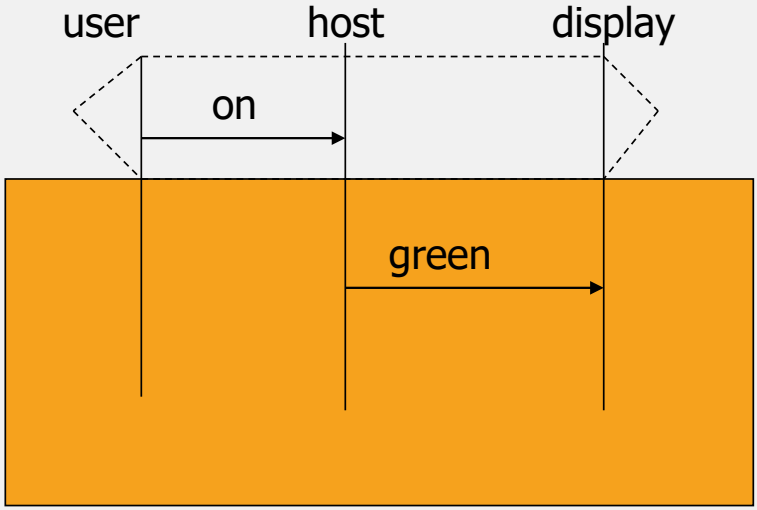


LSC Specification



PLAY-OUT: EXAMPLE I

LSC Specification

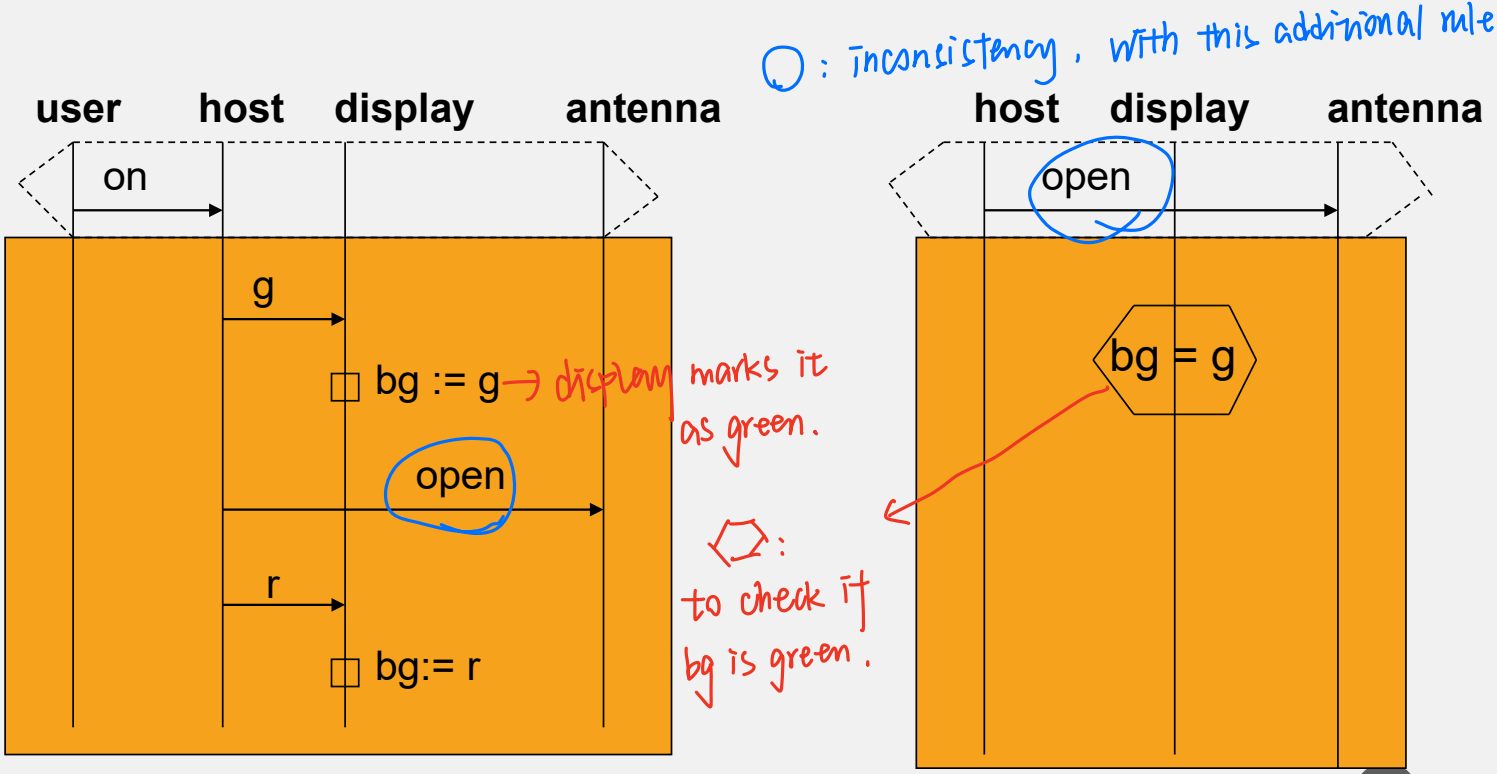


System Response completed.

Simulation run did not lead to violation of LSC specification.

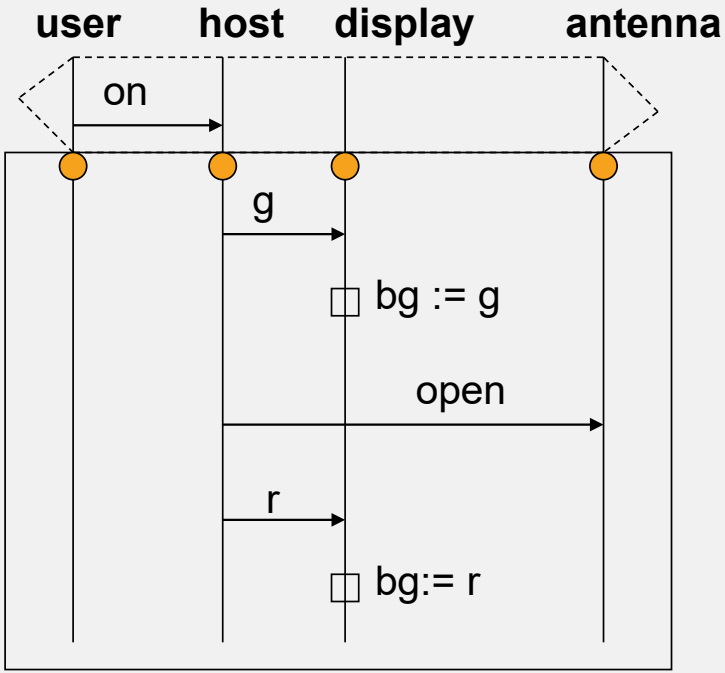
Wait for new stimulus from user.

A LITTLE MORE INVOLVED EXAMPLE



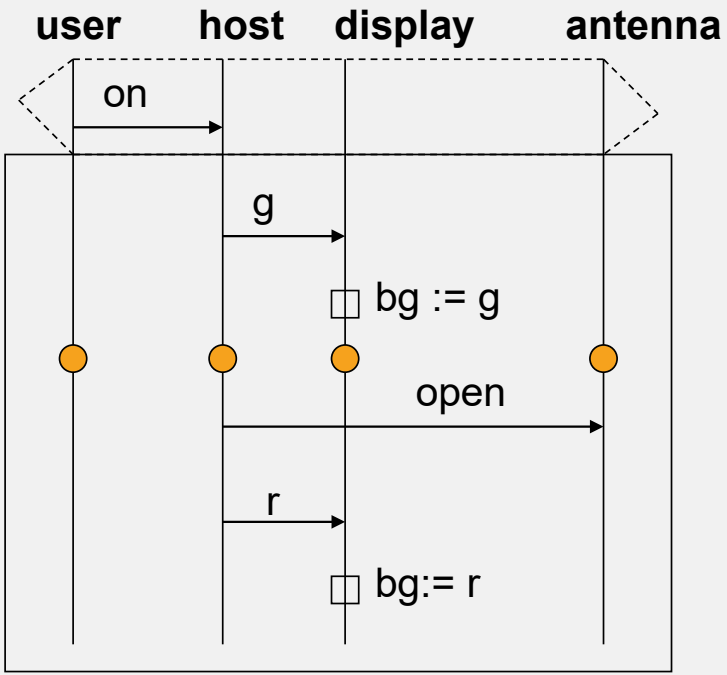
PLAY-OUT

Active copy



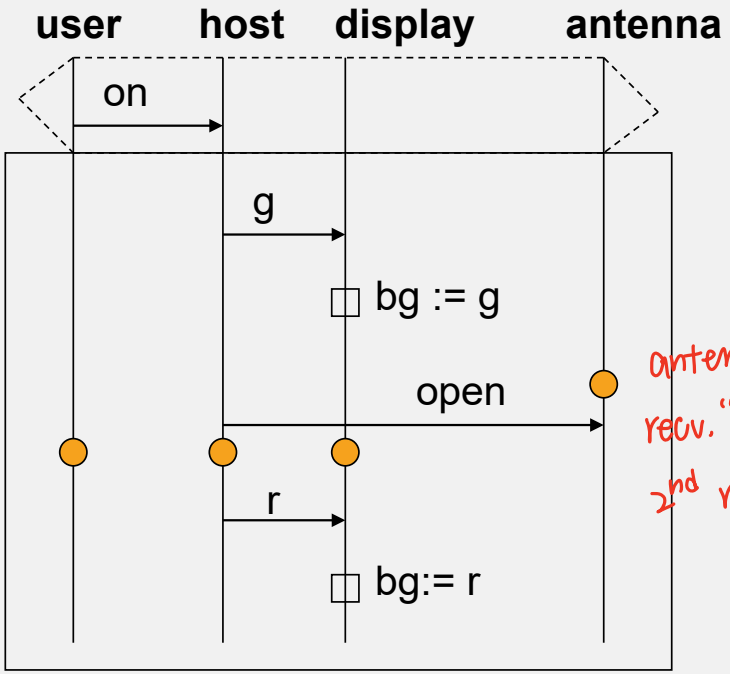
PLAY-OUT

Active copy



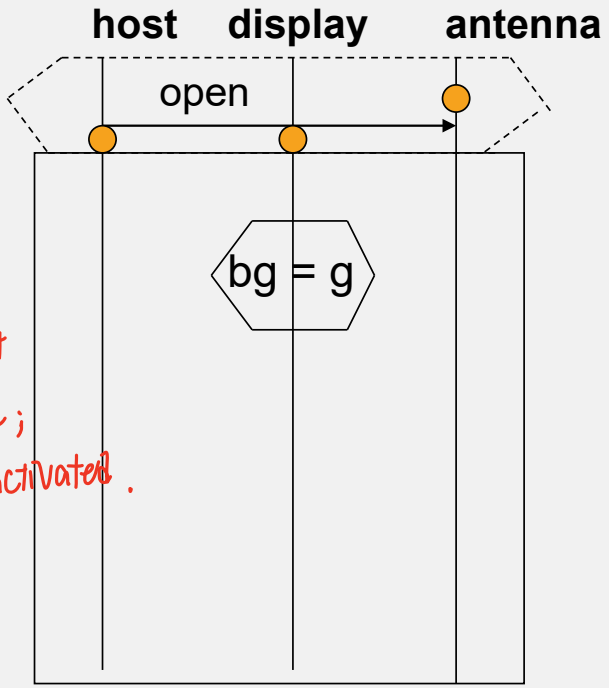
PLAY-OUT

Active copy



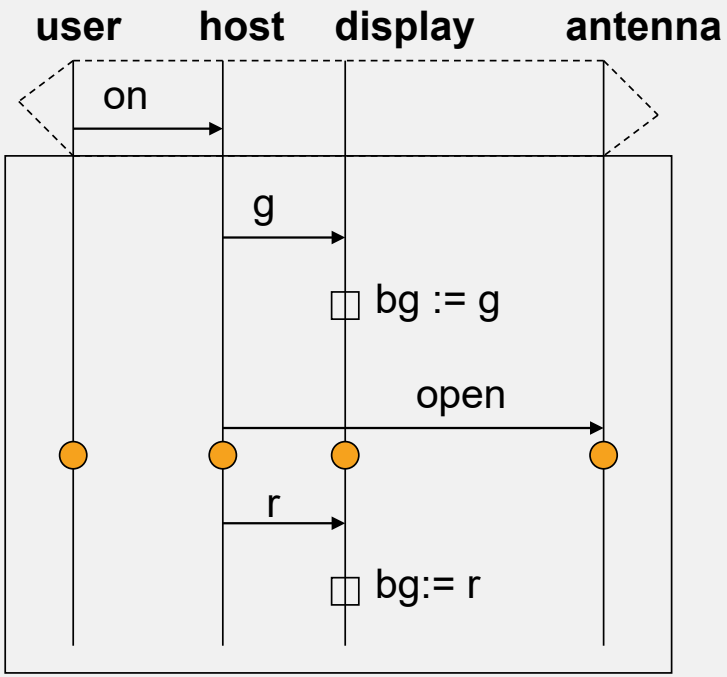
antenna hasn't recv. "open" yet; 2nd rule gets activated.

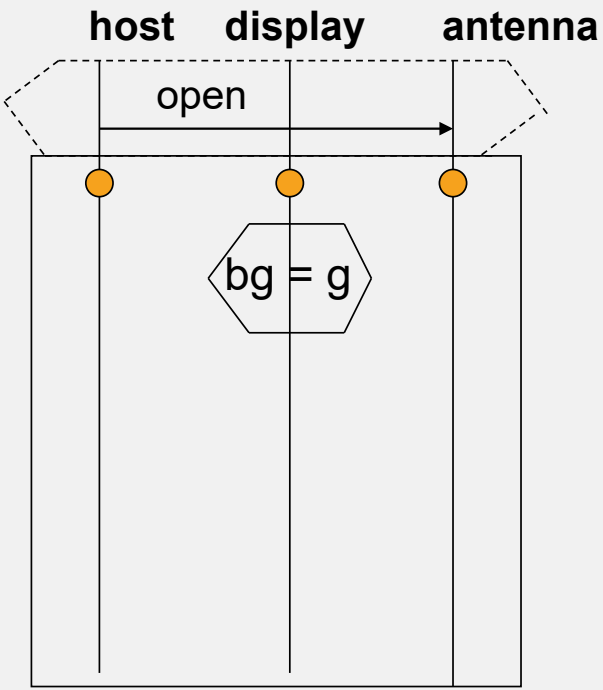
Pre-active copy



How does the LSC execution engine progress the simulation ?

CHOICE 1

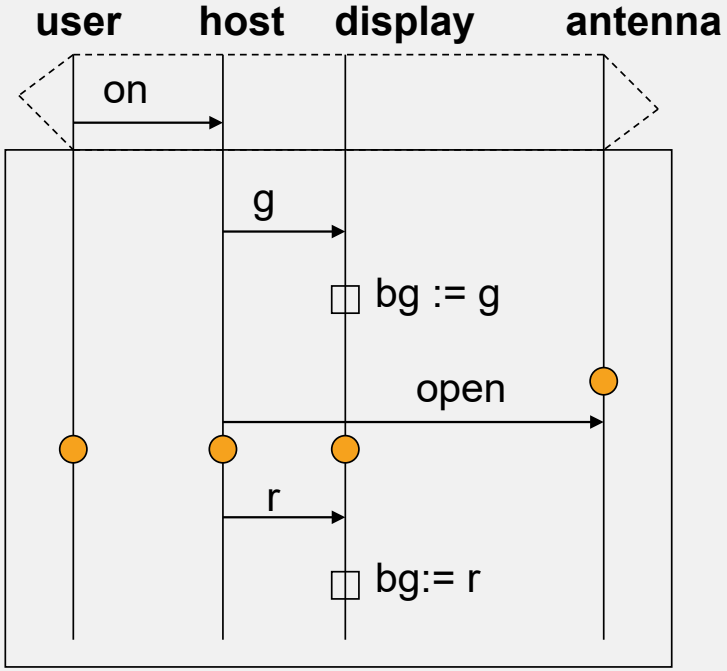




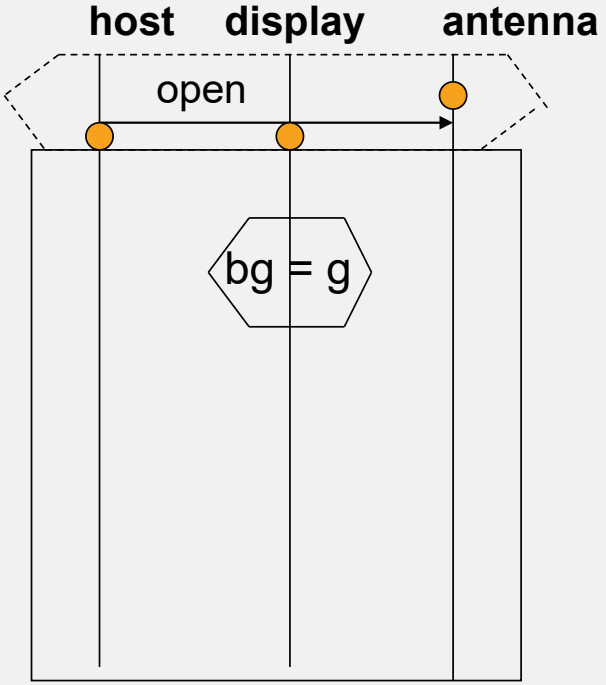
System response can now be completed without violating spec.

PLAY-OUT

Active copy

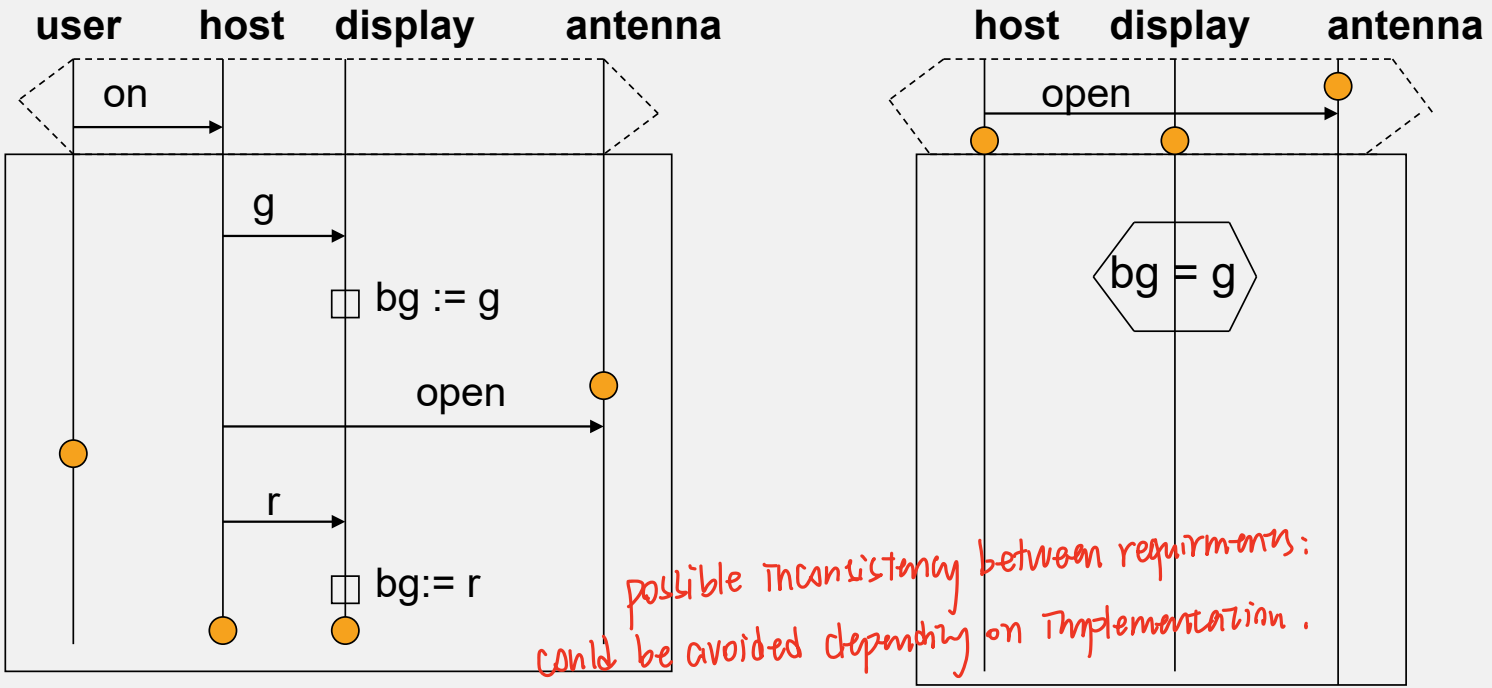


Pre-active copy



How does the LSC execution engine progress the simulation ?

CHOICE 2

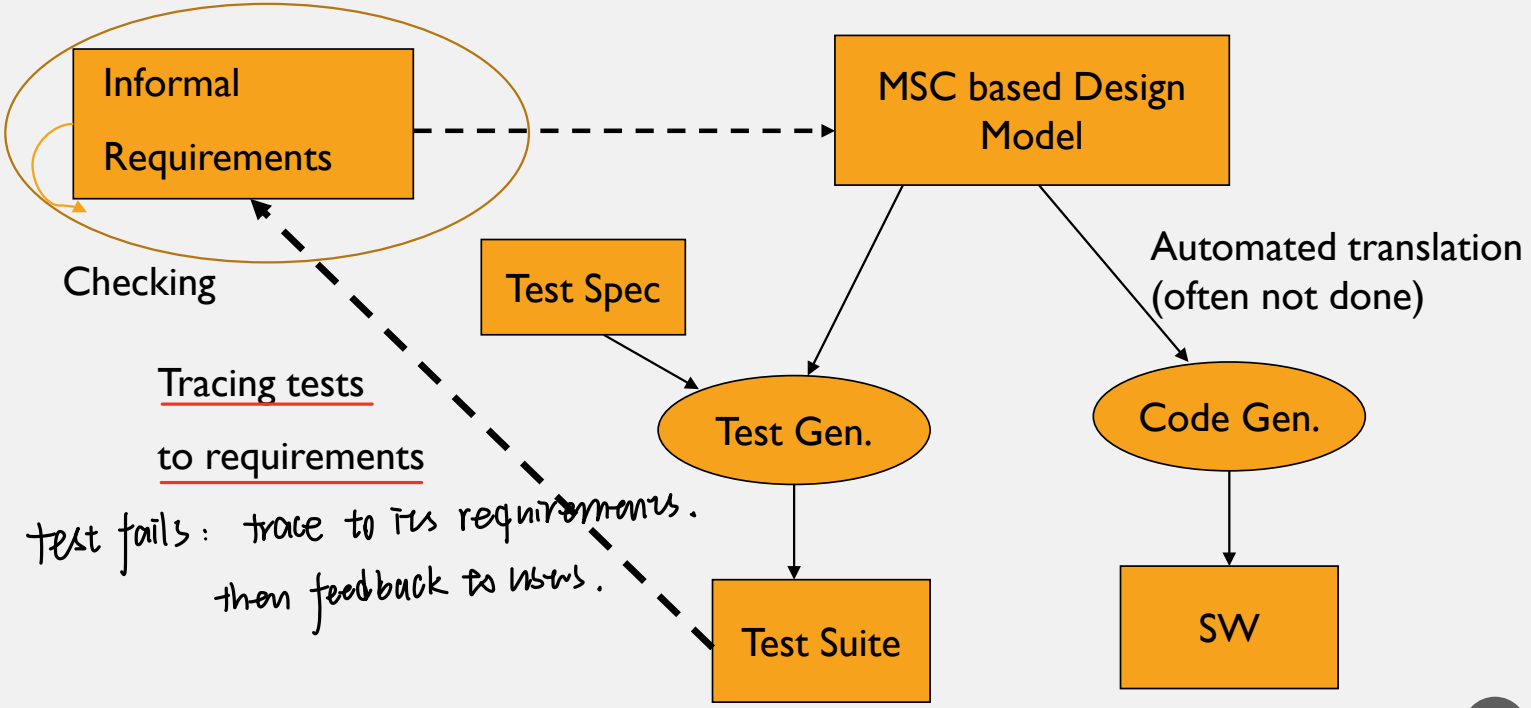


possible inconsistency between requirements: could be avoided depending on implementation.

Progressing the simulation will lead to violation of a univ. chart

eg. ensure when 'open' is recv. before. send r.

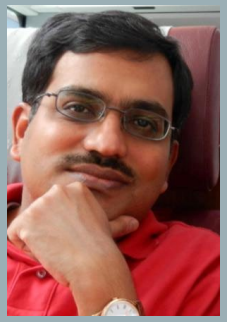
ROUND-UP OF THE TOPIC

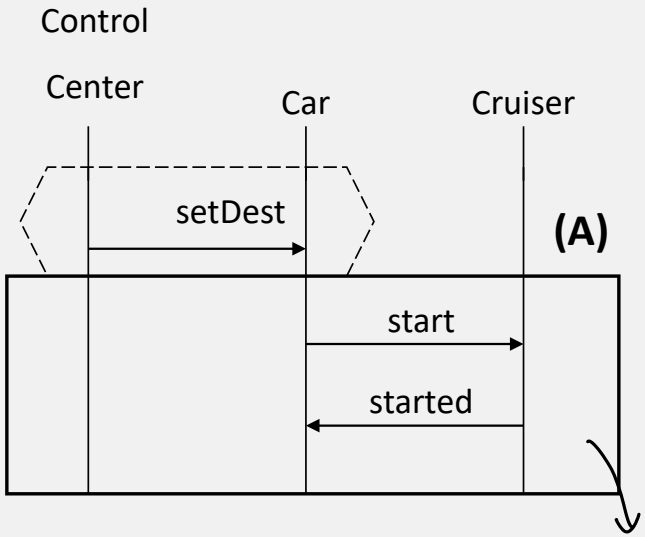


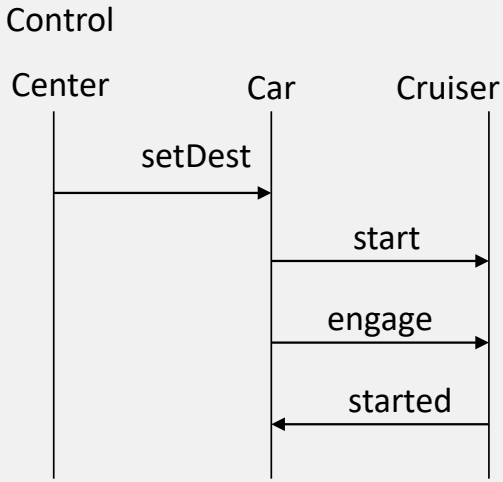
REQUIREMENTS EXERCISES
CS3213 FSE

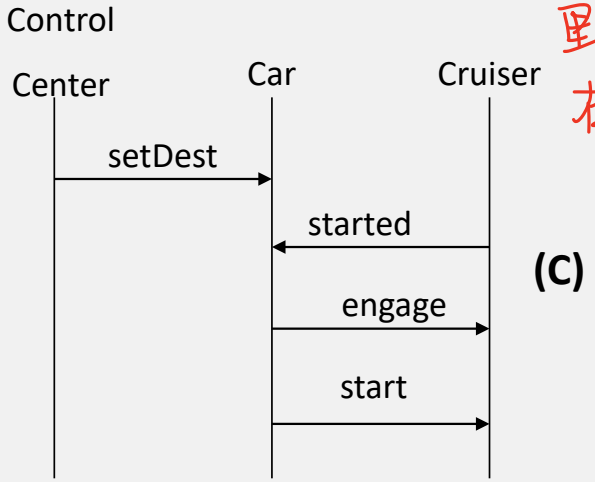
Prof. Abhik Roychoudhury

National University of Singapore









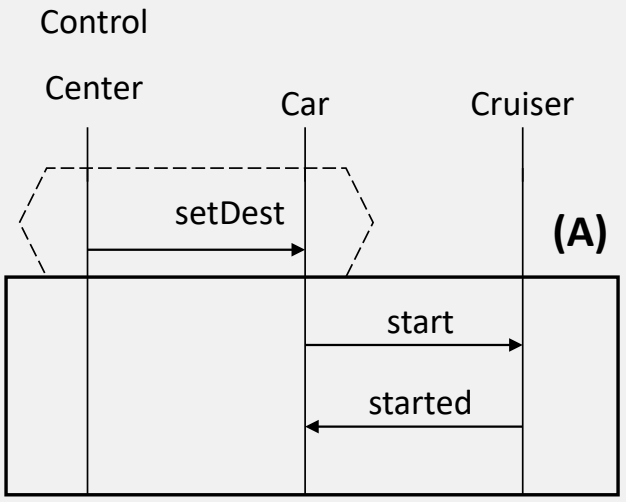
里面没有的 don't care.
有的都对就行.

(A) is the LSC specification

(B) Satisfies (A)? ✓

(C) Satisfies (A)? ✗

SUITABLE FOR DECLARATIVE REQ



"*Whenever the control center sends a setDest message to a car c, the car sends a start message to the cruiser to which the cruiser acknowledges with a started message.*"

Hypothetical requirement, but this is the style in real documents for very safety critical domains

Amenable to even NLP techniques?

Requirements get converted to individual charts, traceability is easy.

QUESTION

- Suppose you had a dream – where in your dream you discovered a perfect technique and tool to convert any informal software requirements in English to a formal model as a finite state machine. Think of all possible ways in which such a tool may be useful for developing software with high quality assurance.
- Answer the same question where the informal requirements got converted magically to Live Sequence Charts.

ANOTHER QUESTION

- Consider a collection of processes consisting of temperature controller, thermostat, air-con and heater.
  - When controller receives low signal from thermostat it sends on signal to heater
  - When controller receives high signal from thermostat, it sends on signal to air-con.
- De-lineate the boundary between "system" and "environment". Which processes are in the "system" in this example?
- Explore the inter-play of the inter-process and intra-process requirements modeling by discussing modeling as
  - Message Sequence Chart
  - Live Sequence Chart
  - State Machines

YET ANOTHER QUESTION

Consider an example situation that occur in a programmed IO environment. The CPU issues a WRITE command to the I/O module which then checks if the peripheral is free. The request may either be accepted or rejected depending on whether the peripheral is free or not. If the request is accepted, the CPU sends the data to the I/O Module to write to the peripheral. If the peripheral is not free, the I/O module makes the CPU wait till the peripheral is free.

Show sample interactions using MSCs.

- How to convert the MSCs into LSCs, which part is the pre-condition?