CS4272: Hardware Software Codesign

Statecharts

Abhik Roychoudhury
School of Computing
National University of Singapore

What we did last week

- Co-design Methodology
  - Modeling
  - Partitioning
  - Allocation/Scheduling
  - Software Analysis & Compilation
  - ...
- UML as modeling notation
  - System Requirements: Use-cases, Scenarios
  - System structure: Class diagrams
  - System behavior: State diagrams (today!!)

Organization

- Finite state machines
  - Other variants
- Reactive and transformational systems
- Statecharts
  - Depth (OR-states)
  - Orthogonality (AND-states)
  - Broadcast communication

Readings

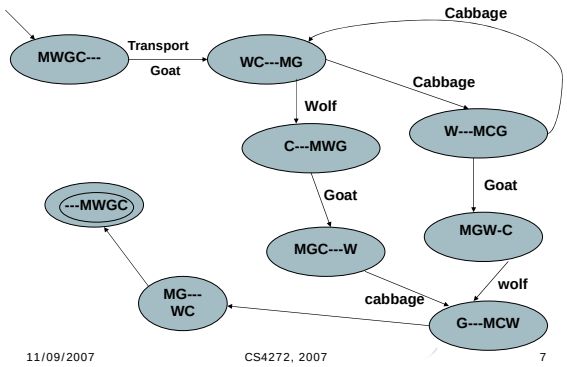
- Statecharts: A visual formalism for complex systems, by David Harel, Science of Computer Programming, 1987
- Executable object modeling with statecharts, by David Harel and Eran Gery, IEEE Computer, 1997
- Basic understanding of states/transitions is introduced first.

Introducing FSMs --- a puzzle

- A man with a goat, a wolf and a cabbage wants to cross a river.
- A boat can carry only 2 of the 4 entities.
- Wolf wants to eat the goat.
- Goat wants to eat the cabbage.
  - How to transport all the 4 entities ?
- Think of modeling the local state of each entity – on which side of the river?
  - A global state is a composition of these local states --- transitions of global states form FSM

State change

State change



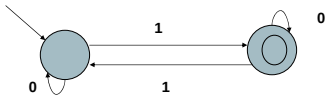
Modeling using FSMs

- A solution to our problem is a path from the initial state to a state where all 4 entities are on other side of river.
  - Notion of "termination" of the problem.
  - Shown as accepting states of FSMs
- Minor note:
  - Not all cycles in the FSM for this problem have been shown.

FSM --- Definition

- M = (S, S0, Σ, $\rightarrow$, F)
  - S is a set of states
  - S0 $\subseteq$ S is the set of initial states
  - $\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation
  - F $\subseteq$ S is the set of final or accepting states
- The set of strings accepted by M or the language of M
  - L(M) = all strings which have a path from an initial state to an accepting state.
  - Using finite state machines for recognizing or distinguishing (infinite) set of (finite) strings.

FSM --- Example



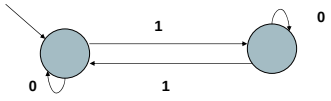
Accepts all binary strings with odd number of 1s

An infinite collection of finite strings

Transition Systems

- FSMs can accept infinite strings too, change accepting condition
  - An infinite string is accepted iff it visits at least one final state infinitely often.
- Transition systems go one step further where all states are accepting.
  - TS = (S, S0, Σ, $\rightarrow$)
    - No notion of terminating or accepting states
    - The alphabet Σ labeling the transitions is also optional.
    - The traces captured by a transition system are obtained by unrolling the graph from the initial state(s).

TS - Example



Traces captured by this transition system are

(0* 1)* 0ω

(0*1)ω

Transformational Systems

- Conventional notion of a terminating program.
  - Takes in input.
  - Performs computation step.
  - Terminates after producing output.
- System behavior
  - Can be described as a transformation function over the input.
- What about controllers ?
  - In continuous interaction with the environment.

Reactive Systems

- Continuously interacts with its environment.
  - No notion of system termination.
- Interaction with environment is typically asynchronous.
- Often consists of a concurrent composition of processes.
  - Often, its response to environment needs to obey time constraints.

Reactive system behavior

- (Infinite) collection of infinite traces.
- Traces denote ongoing interaction with environment.
- Use state transition systems to describe behavior of a reactive system
  - Too much complexity
  - Many processes --- concurrency
  - Each process has many states --- hierarchy
  - What kind of inter-process communication?
- The language of Statecharts addresses these practical issues !!

Visual Formalisms

- Important/imperative at initial design stages.
- Vital for communication.
- Formal visual languages can help in:
  - Documentation
  - Initial analysis.
  - Developing correct-by-construction translation to more detailed (non-visual) descriptions.

Statecharts

- Statecharts =
  - FSMs +
  - Depth +
  - Orthogonality +
  - Structured transitions +
  - Broadcast communication
- Used in the Rhapsody tool.
- Included in UML 2.0 as state diagrams.

Statecharts

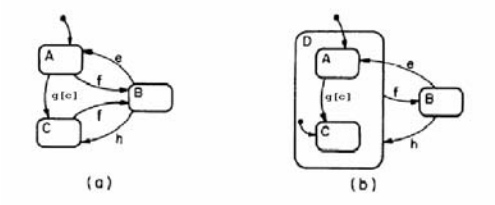
Depth:
States can have internal structure.
OR type states

Orthogonality
Independent states
Concurrency
AND type states

Structured transitions
Succinct descriptions of transition families.

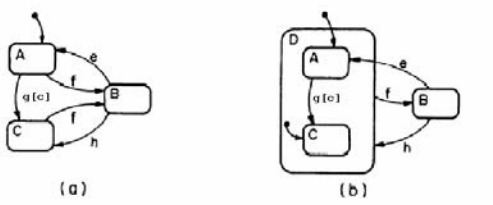
Broadcast communication
Succinct descriptions of synchronizations

Depth : OR States



(b) is the statechart representation of the FSM (a).

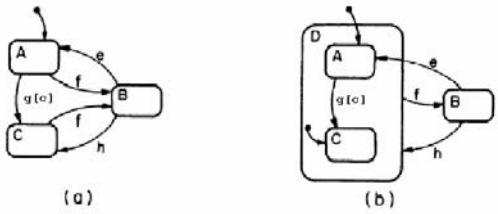
Depth : OR States



A and C are clustered into a superstate D

A and C are the internal exclusive-or components of the D state.

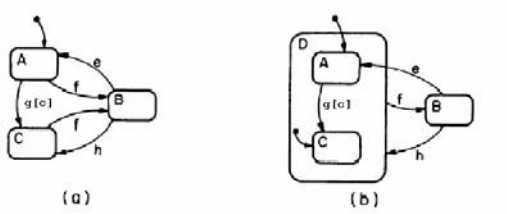
Depth : OR States



e, f : are trigger (external) events.

g [c]: g, a trigger event and c a condition

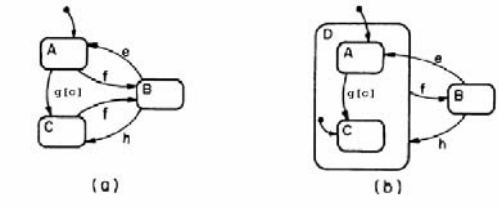
Depth : OR States



f is a transition from D to B.

From any D-state (A or C) there is an f-move to B

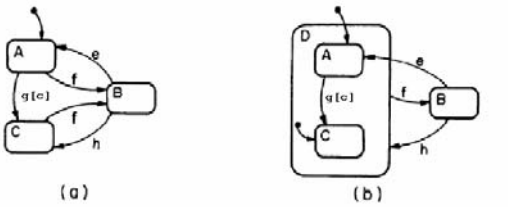
Depth : OR States



h is transition from B to D (A or C).

The actual state entered is the default entry state; the state C

Depth : OR States

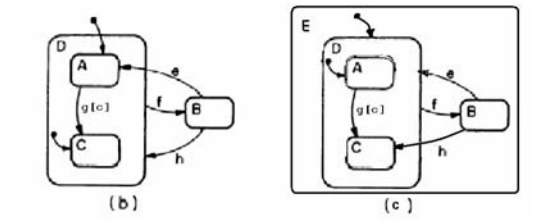


D is the initial state.

The actual initial state within D is not the default state C.

Instead, it is A.

Depth: OR States

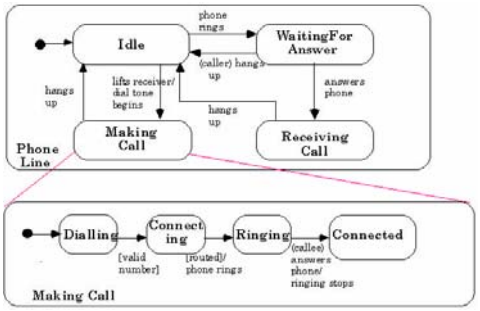


Which state will transition e yield in (b) and (c)?

Which state will transition h yield in (b) and (c)?

What's the default state for the superstate E in (c)? Hierarchically!

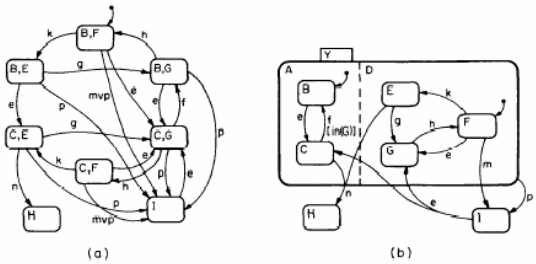
A Concrete Example: OR-chart



OR-State: in a nutshell

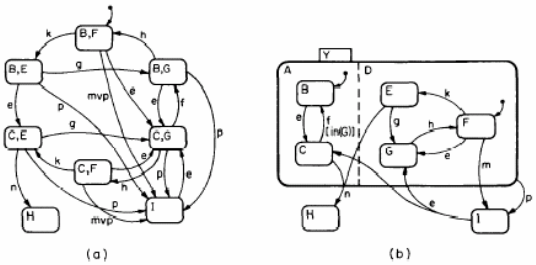
- An OR-state can contain other states as its internal substates (hierarchical internal structure);
- A super OR-state is active, if and only if one of its immediate substates is active (exclusive or);
- When the control enters a (super) OR-state, its default substate is entered and becomes active;
- When the control leaves a (super) OR-state, all its substates become inactive!
- More issues: history, priority, ...

Orthogonality: AND States



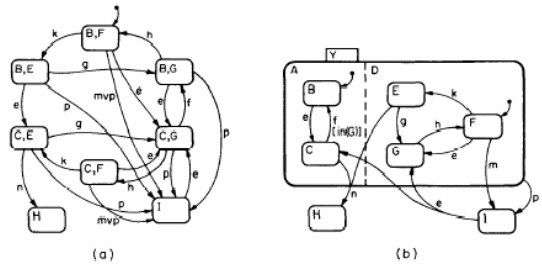
(b) is the statechart representation of the FSM (a).

Orthogonality: AND States



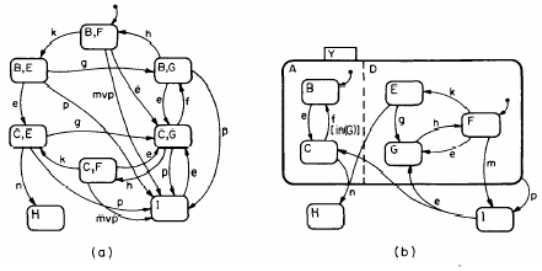
Y is an AND state.
It has two orthogonal components A and D.
A is an OR state with components B and C.
D is an OR state with components E, F and G.

Orthogonality: AND States



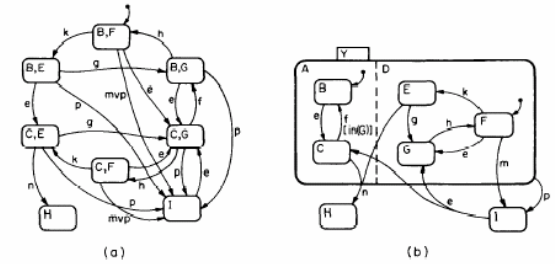
Y is an AND state.
It has two orthogonal components A and D.
a state of Y is composed of a state of A and a state of D
What is the default initial state of Y?

Orthogonality: AND States



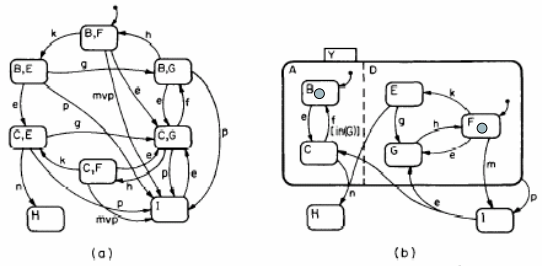
Y is an AND state.
It has two orthogonal components A and D.
a state of Y is composed of a state of A and a state of D
What is the default initial state of Y? (B,F)

Orthogonality: AND States



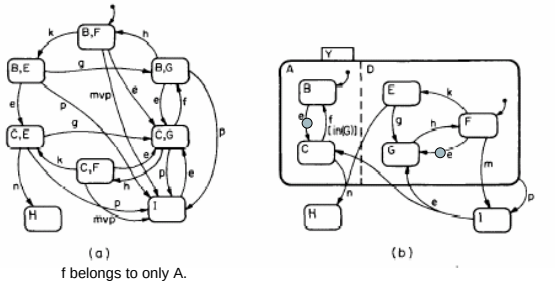
f belongs to only A.
e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

Orthogonality: AND States



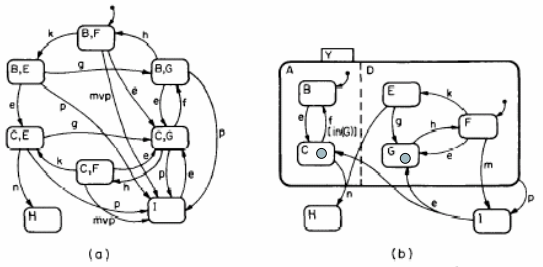
f belongs to only A.
e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

Orthogonality: AND States



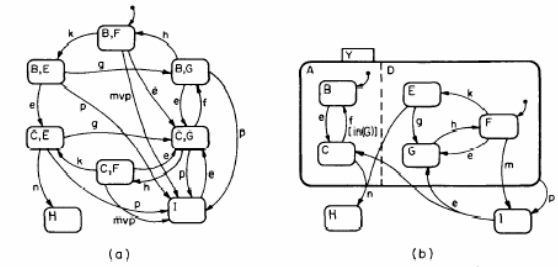
f belongs to only A.
e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

Orthogonality: AND States



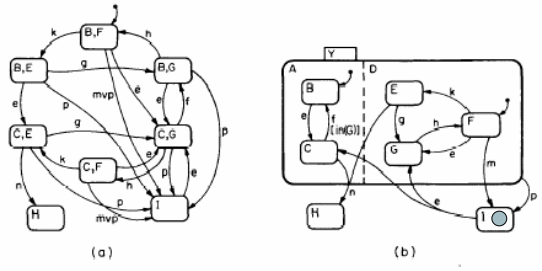
f belongs to only A.
e belongs to both A and D.
From (B,F) there is a simultaneous e-move to (C,G)

Orthogonality: AND States



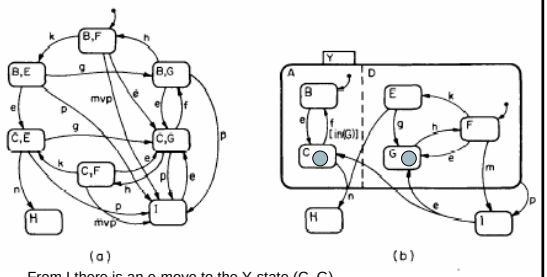
From every Y state (how many?) there is a p-move to I

Orthogonality: AND States



From every Y state (6!) there is a p-move to I
From I there is an e-move to the Y-state (?, ?)

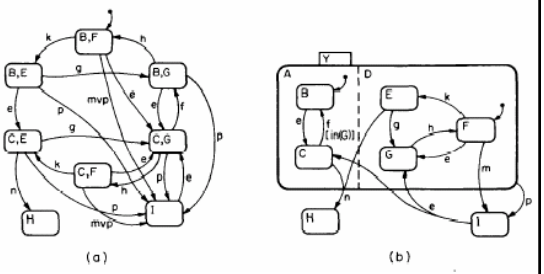
Orthogonality: AND States



From I there is an e-move to the Y-state (C, G)

What if there is an e-arrow from I to just the surface of Y

Orthogonality: AND States

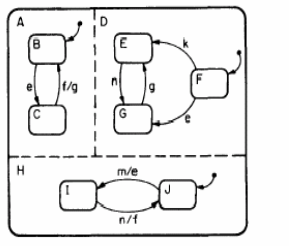


For each (?, F) state there is an m-move to I
Note the [in G] condition attached to the f-move from C (state reference!).

AND-state: in a nutshell

- An AND-state is composed of several independent (OR-)states that run in parallel (concurrency);
- An *active state* of an AND-state comprises a state of each concurrent component, i.e., $(S_1,S_2,\ldots,S_n)$;
- When the control enters (leaves) an AND-state, it simultaneously enters (leaves) all its components;
- An AND-state can even occur inside an OR-state (different from conventional programming languages)

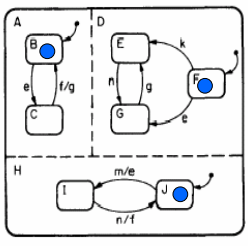
Broadcast Communication



A transition has a trigger and an action (output!)

But the output of a transition can be inputs for other orthogonal components!

Broadcast Communication

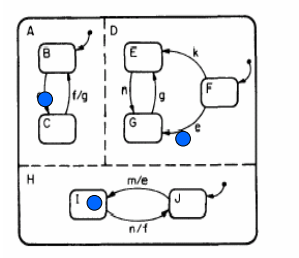


Start configuration (B, F, J)

m/e: m is the trigger event, while e is the action (output!)

Suppose m (external event) occurs.

Broadcast Communication

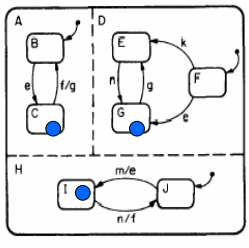


Start configuration (B, F, J)

Suppose m (external event) occurs.

H goes to I from J ; e-moves are enabled in A and D

Broadcast Communication

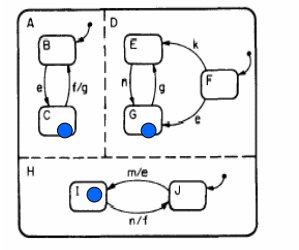


Start configuration (B, F, J)

m occurs

Final configuration (C, G, I)

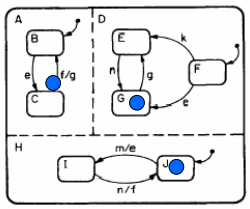
Broadcast Communication



Suppose event n comes,

What happen now?

Broadcast Communication

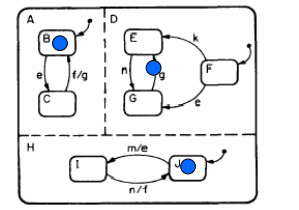


Now suppose event n comes,

What happen? Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$

Broadcast Communication

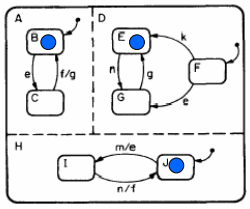


Now suppose event n comes,

What happen? Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$, which again fires $G \xrightarrow{g} E$

Broadcast Communication



Now suppose event n comes,

Transition $I \xrightarrow{n/f} J$ is fired, f is generated,

which fires transition $C \xrightarrow{f/g} B$, which again fires $G \xrightarrow{g} E$

Finally yielding (B,E,J)

What are the triggers/actions

- Method call
  - Method_name(parameters)
- Or, Event
  - Event_name(parameters)
- Is there a difference?
  - Lots, in terms of semantics
  - A method call involves a transfer of control
    - If there are nested method calls, they can cause further transfer of control
  - An event will be lodged in a system queue
    - It will be removed by the recipient later.

Events and Method calls

- Event based communication
  - The one we usually adopt
  - Inherently asynchronous
    - Designer does not worry about controlling all interaction sequences (this is taken care of by the system queue)
- Method call based communication
  - Synchronous, involving transfer of control
  - Involves close control by the designer over interaction sequences ---
    - getting closer to code level

Most General form of ...

- ... annotation for a transition
  - Trigger[condition]/Action
- Trigger is event expression or method invocation
- Condition is like a branch condition on data variables
- Action is a program
  - Sequence of event generation or method invocation or even code in a programming language.

Summary

- Practical Use of Statecharts in Modeling Object-based systems
  - Use statecharts to describe behavior of classes (of active objects)
  - Class Associations given by class diagrams.
  - Contains code in the actions for realistic designs
- A realistic approach for modeling (distributed) embedded controllers.

Example Modeling via StateCharts

Abhik Roychoudhury
National University of Singapore

The Example Control System

- 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.**
    - **Modeled and simulated the Weather update subsystem from Requirements Document.**

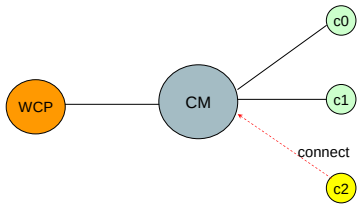
Weather Update Controller

- 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)

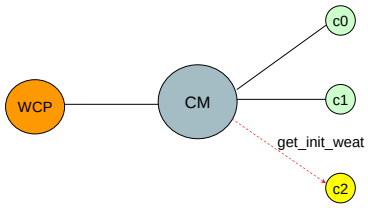
Weather Update Controller

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

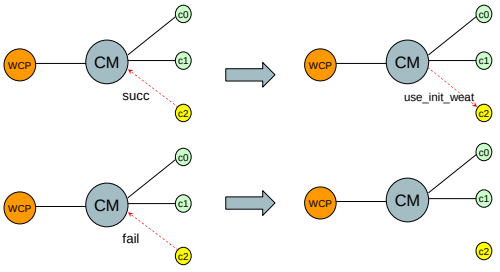
Client Initialization



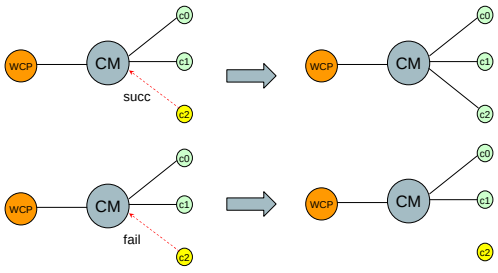
Client Initialization



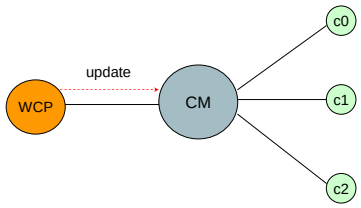
Client Initialization



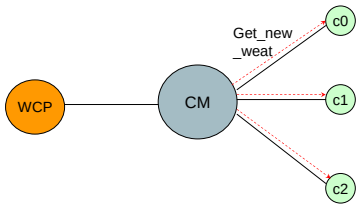
Client Initialization



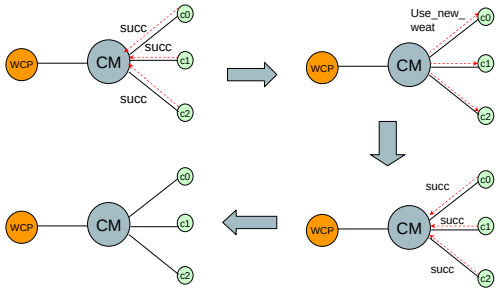
Weather Update



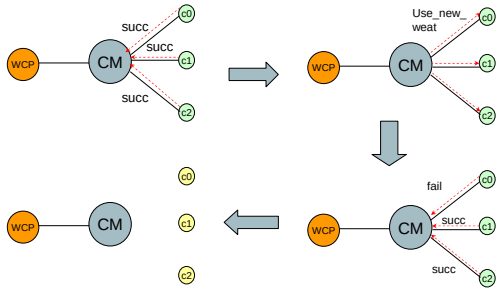
Weather Update



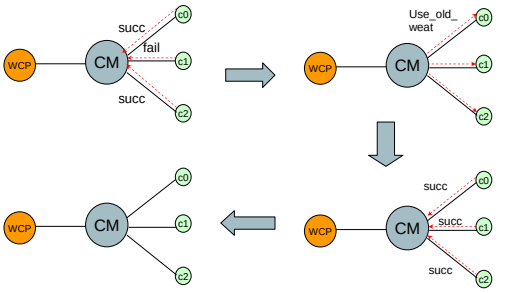
Client Update – Case 1



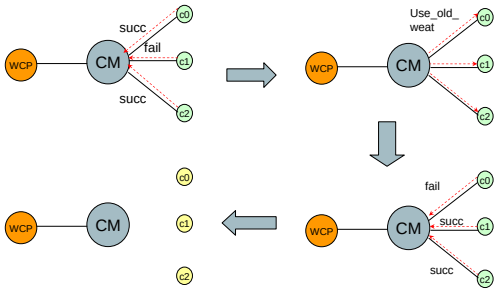
Client Update – Case 2



Client Update – Case 3



Client Update – Case 4



What do the Requirements …

- … look like ?

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

Client Initialization Requirements

- A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.
- If the CM'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 (CM's status is not idle), the CM will send a message to the client to refuse the connection, and the client remains disconnected.

Client Initialization Requirements

- When the CM is pre-initializing, it will send a message to instruct the newly connected client to get the new weather information, and then set both its own status and the client's status to initializing.
- If the client reports success for getting the new weather, the CM will send another message to inform the client to use the weather information, and then set both its own status and the client's status to post-initializing.
  - Otherwise, if getting new weather fails, the CM will disconnect the client and set its own status back to idle.

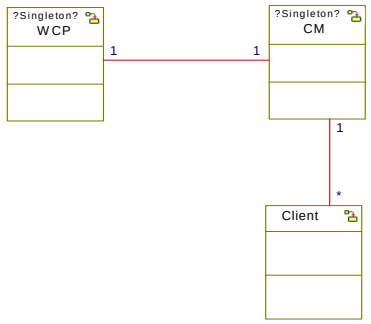
Client Initialization Requirements

- If the client reports success for using the new weather, this initialization process is completed. the CM will set both its own status and the client's status to idle, and re-enable the WCP so that manual weather update is allowed again.
  - Otherwise, if using new weather fails, the CM will disconnect the client, re-enable the WCP, and set its own status back to idle.

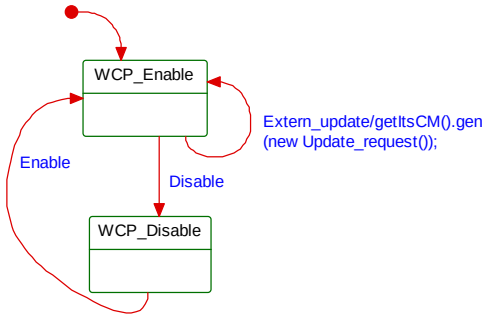
What does the …

- … actual system modeling look like?
  - The Class Diagram
  - State Diagrams
    - **Fragments showing the initialization part of the Client and CM**
  - Sample Sequence Diagram
    - (a) c1 connects to the CM successfully.
    - (b) c2 connects to the CM successfully.
    - (c) After that, a weather update request is sent by the WCP.
    - (d) Both c1 and c2 report success in getting new weather information.
    - (e) In the meantime, c3 tries to connect to CM. The connection request fails since the weather update is in progress.
    - (f) c1 successfully uses the new weather information, but c2 fails, which causes both of them disconnected from the CM.

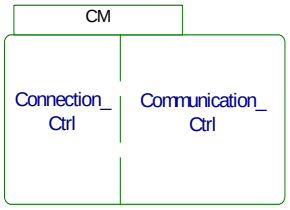
Class diagram



Statechart for WCP



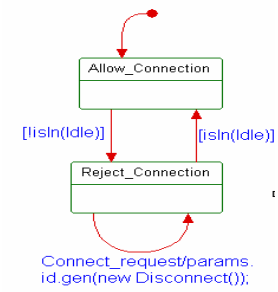
StateChart for CM



Connection_Ctrl: make sure a client can get initialized only when CM is neither initializing any other client nor performing an update

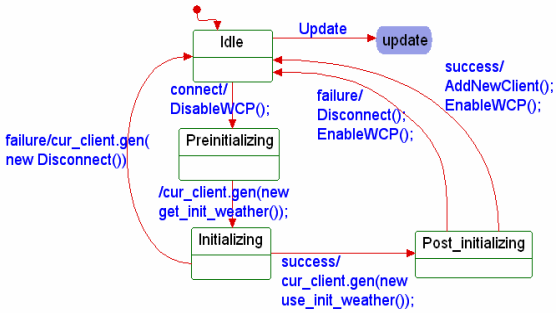
Communication_Ctrl: control initialization and update process

Connection_Ctrl

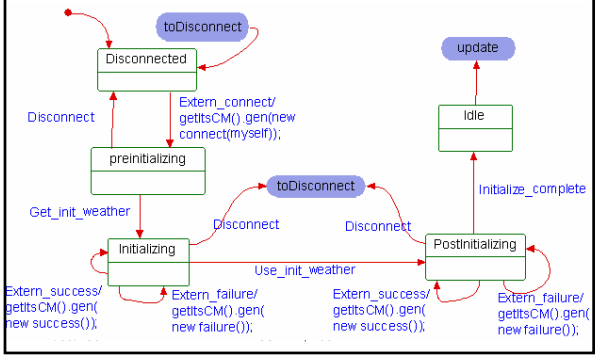


isIn(s): test whether the object (CM in this example) is currently in state s.

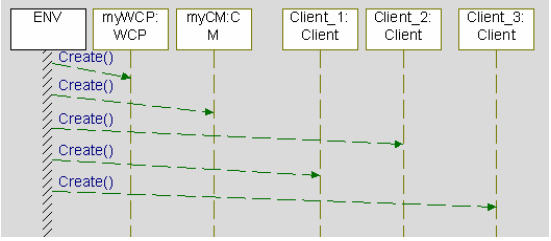
Communication_Ctrl (initialization)



Client (initialization)

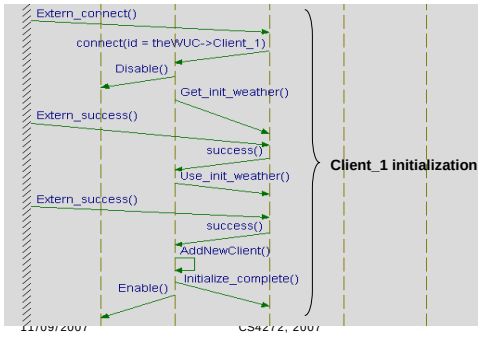


MSC example

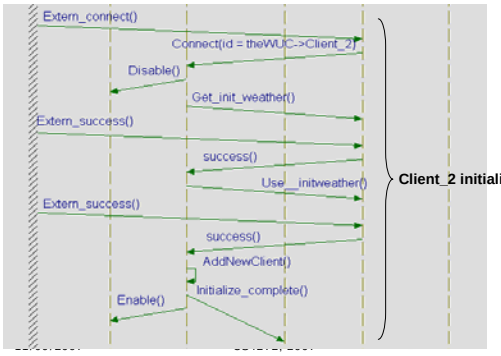


Rhapsody animation starts

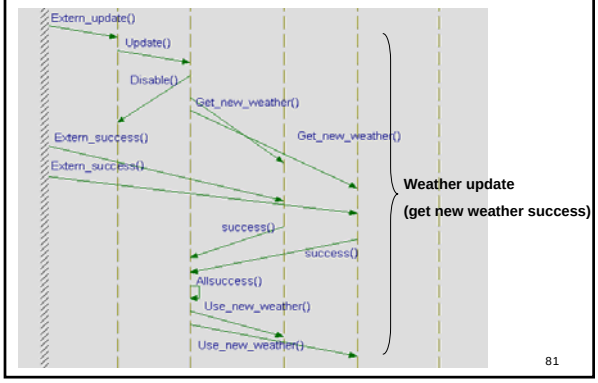
MSC example cont…



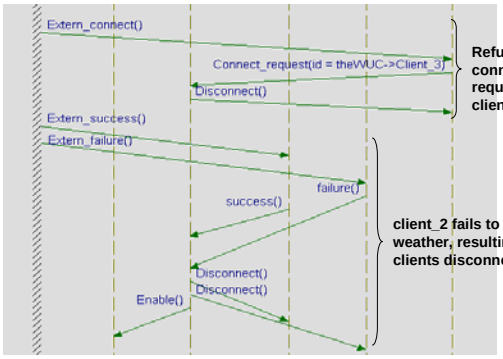
MSC example cont…



MSC example cont…



MSC example cont…



Wrapping up …

- Discussion of the first assignment
  - Due in 3 weeks time (20 September)