INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Fall Quarter, 2007
Michele Rousseau
Set 20

Announcements

- **Homework Due 11/21 – (yep, that'd be today) at 3p**
- **Quiz #4 will be on Wednesday after the holidays 11/28**
- **Happy Thanksgiving!**

Previously in INF 111…

- **UML**
  - Package Diagrams
  - State Transition Diagrams

Review

Review

Today's Lecture

UML

- State Transition Diagrams
- Activity Diagrams

Types of UML Diagrams

Structure	Behavior
(6 types)	(4 types)
Class diagrams	Activity diagram
Object diagram	Use Case diagram
Package diagram	State machine diagram
Composite structure diagram	Interaction diagrams
Component diagram	**Sequence diagram**
Deployment Diagram	**Communication diagram**
	Interaction overview diagram
	Timing diagram

If the appropriate diagram is not part of UML
use it anyways

State Transition Diagrams

- State Transition Diagrams show the *dynamic behavior* of a class instance or of a whole system
- *State*: the duration of time during which an object is doing an activity.
- A ***state diagram*** is a **graph in which**
  - nodes correspond to states and
  - directed arcs correspond to transitions
  - labeled with event names.

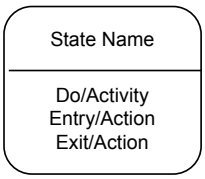
When to use :
Necessary for those objects whose behavior across many use cases needs to be understood

State Transition Diagrams

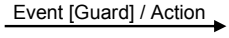
- An ***event*** occurs at a point in time and
  - transmits information from one object to another
- An ***action*** occurs in response to an event and cannot be interrupted
- An ***activity*** is an operation with certain duration that can be interrupted by another event
- A ***guard*** is a logical condition placed before a transition that returns either a true or a false.

State Transition Diagrams: Notation

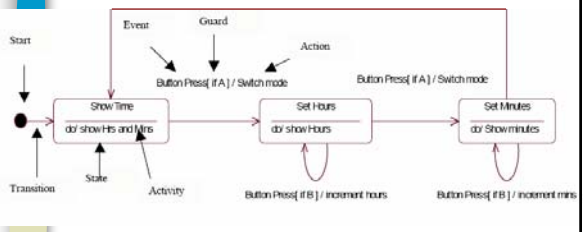
- State symbol:



- Transition Symbol:

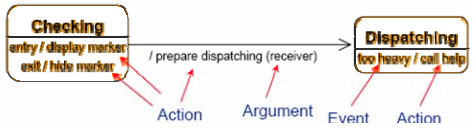


State Transition EX: Digital Watch



Actions

- A short software process that executes immediately.
- A **transition** may **trigger** an action.
- May be triggered on **entry** or **exit** of states (instead of labeling each incoming (entry) and outgoing (exit) transition with these actions).
- An **event** may trigger an action without leaving the state,
  - **i.e., without triggering exit and entry actions as a self-transition would do.**
- An action may trigger events, usually in other objects.
- Actions may take **arguments**.



Activities

- Can take "longer",
  - i.e., they are processes which last as long as an object is in a certain **state**.
- Are **interruptible**,
  - i.e., an event causing a state transition may abort an activity.
- May be constructed from a start and a final action.

Activity Diagrams

- **Describe**
  - Procedural logic
  - Business process
  - Workflow
- **A flow chart with support for parallel behavior**
- **Branches and Merges model the conditional behavior**
- **Branch: has a single incoming transition multiple, conditional, outgoing transitions**
- **Merge: where conditional behavior terminates**

Each branch has a corresponding merge

- **Represented as a Diamond**

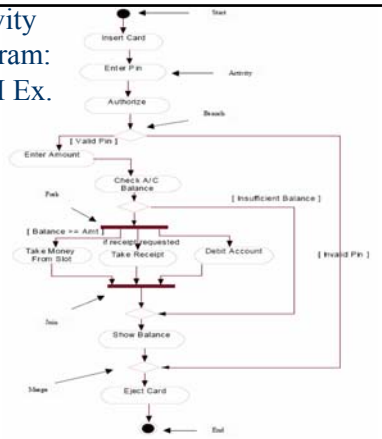
Activity Diagram (2)

- **Forks and Joins model parallel behavior**
- **Fork: has a single incoming transition and multiple outgoing transitions (exhibiting parallel behavior)**
- **Join: synchronizes the parallel behavior**
  - All parallel behaviors complete at the join
- **Represented as a thick line**

Each Fork has generally has a corresponding Join

Activity Diagram: Order Ex

Activity Diagram: ATM Ex.

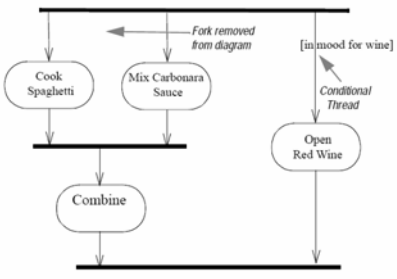


Conditional Thread

There are some exceptions to the each fork having a corresponding join:

- **Conditional Thread: A condition on the thread originating from the fork to create an exception for the join rule.**
  - If the condition is false then that condition is considered to be complete

Conditional Thread: Example

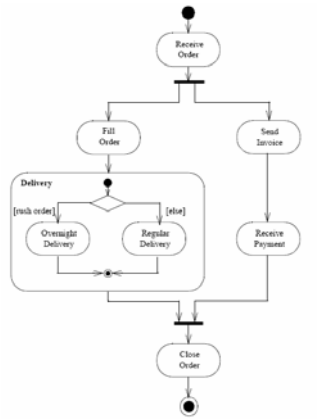


Superstates

- **What if you need to decompose your activity diagram?**
- **Superstates**
  - You can show the superstate with the internal behavior inside or
  - You can show these in a parent diagram
  - You can also use explicit initial and final states

    Adv: you can decouple the parent from the subsidiary and use it in other contexts

Activity Diagram: Superstate



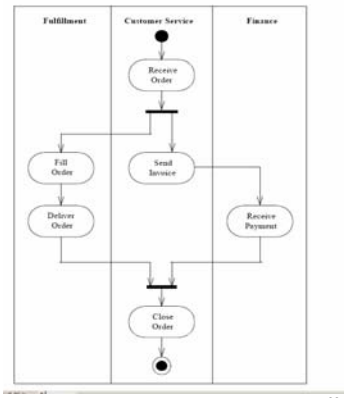
Partitioning an Activity Diagram

Activity diagrams tell you what is happening, but how do you know who does what?

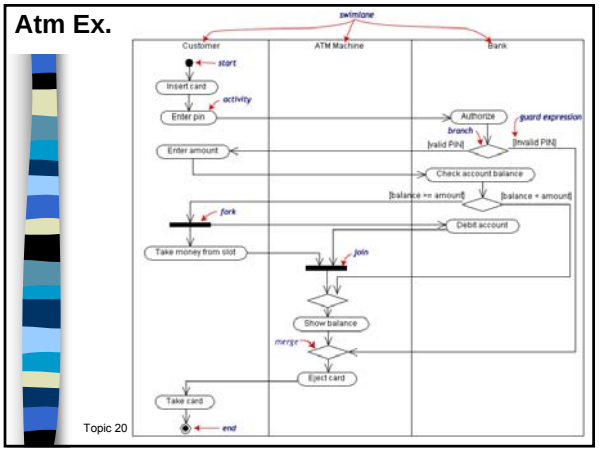
(in programming – which class is responsible for each activity)

- **Swimlanes: group related activities into one column (usually organizationally)**
  - You must arrange your diagram into vertical zones separated by lines.
  - Can be difficult with complex diagrams
    - In this case use non-linear zones – better than nothing

Swimlanes



Atm Ex.



When do you use Activity Diagrams?

- ***Modeling parallel behavior***
- ***Analyzing a use case***
  - Trying to understand what actions need to take place
  - Determine behavioral dependencies
- ***Understanding workflow***
  - Documenting the logic of a business process
- ***Describing a complicated sequential algorithm***
- ***Dealing with multi-threaded applications***

Not so good for

- ***Trying to see how objects collaborate***
  - Use an interaction diagram for that
- ***Trying to see how an object behaves over its lifetime***
  - Use a state diagram for that

Communication Diagrams

- **Used to be known as Collaboration Diagrams (UML 1.x) – but modified for 2.0**
- **Show interactions between run-time elements**
- **Similar to sequence diagrams, but**
  - Focus on objects roles & structure
  - Sequence diagram is better at visualizing processing over time

It is an object diagram that shows message passing relationships

Emphasis on the flow of messages among objects, rather than timing and ordering of messages

- **Sequence Numbers are on arrows rather than vertical order**

Communication Diagrams: Ex