Timing and Activity Models
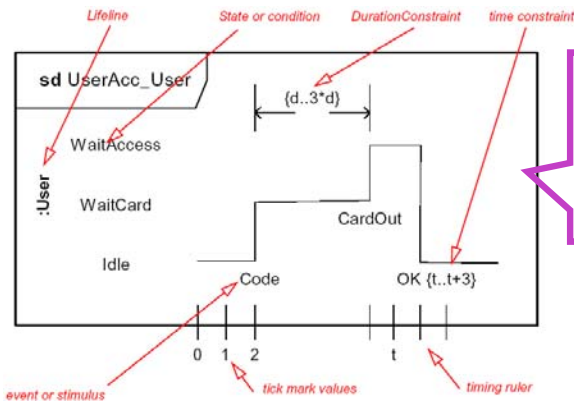
Prof. T.H. Tse

Department of Computer Science
Email: thtse@cs.hku.hk
Web: hku.hk/thtse .

Timing Diagrams

- ◆ A new feature since UML 2.0
- ◆ Shows
  - ▪ time constraints
  - ▪ how time constraints affect interactions among lifelines
- ◆ Very useful when timing is important, such as in engineering applications .

Timing Diagrams
Example 1

Like sequence diagram rotated left by 90% .
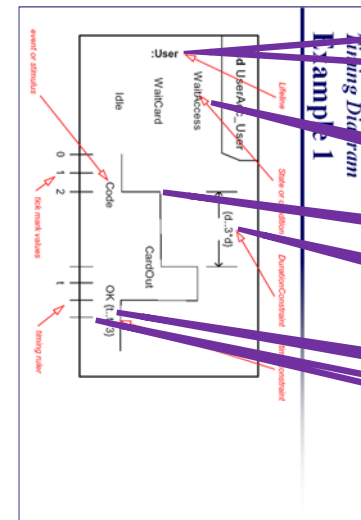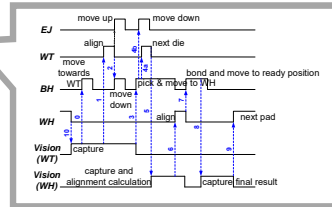
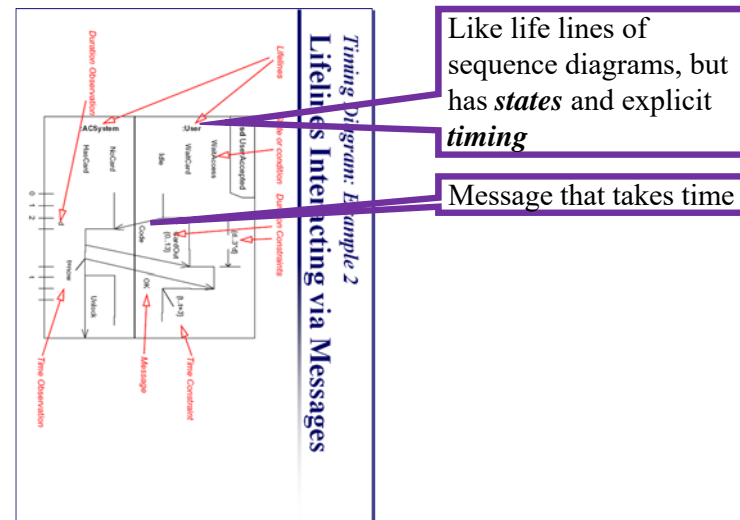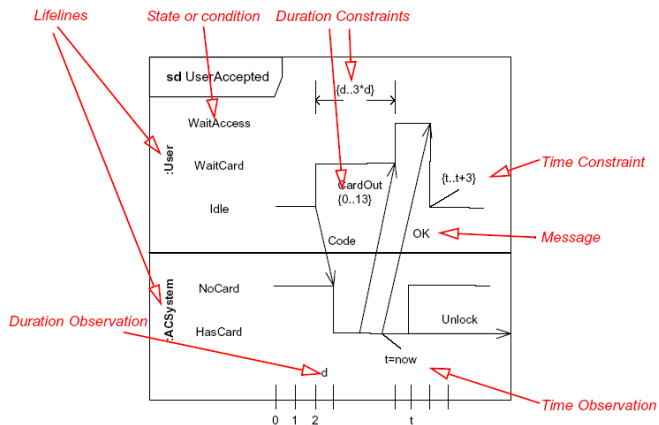Like life lines of sequence diagrams, but has *states* and explicit *timing*

States

Event between states

Duration constraint

Time constraint

Timing ruler .

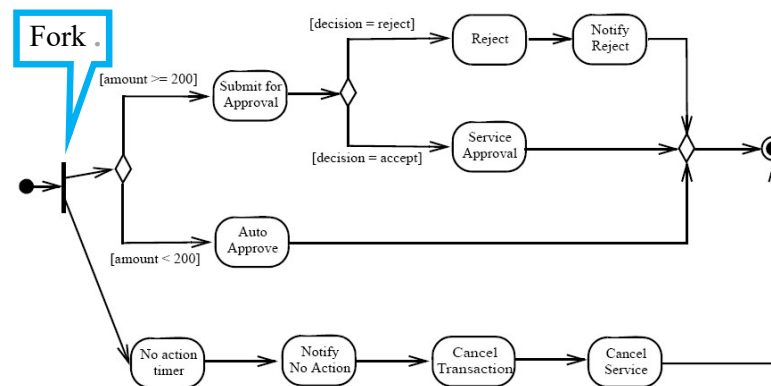## Timing Diagrams: Example 2
Lifelines Interacting via Messages

Lifelines

State or condition   Duration Constraints

sd UserAccepted

WaitAccess

WaitCard

Idle

CardOut {0..13}

Code

{d..3*d}

{t..t+3}   *Time Constraint*

OK   *Message*

:User

:ACSystem

Duration Observation

NoCard

HasCard

Unlock

t=now

d

Time Observation

0 1 2   t

Like life lines of sequence diagrams, but has *states* and explicit *timing*

Message that takes time

Activity Diagram

- Specifies the detailed operations and conditions for *lower-level* behaviour within an object, rather than interactions with actors or other objects
- Resembles traditional *flowcharts*
- Mainly focuses on the *flow of control*
- Activity diagrams are *optional* in UML
  - It is *not* the most important diagram in UML
  - *Pseudocode* may be used instead

Activity Diagram
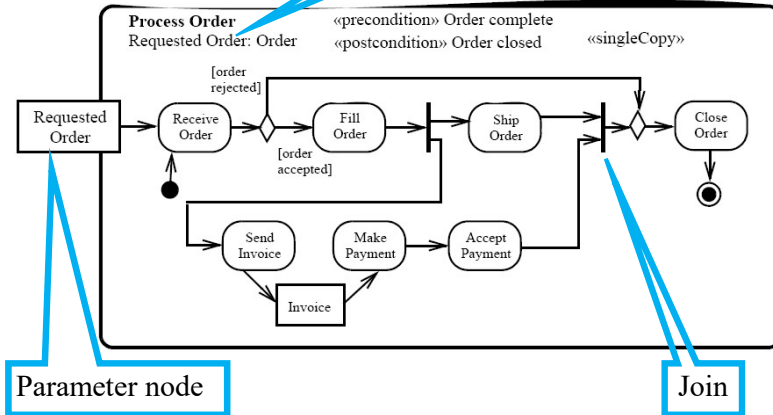# Example 1

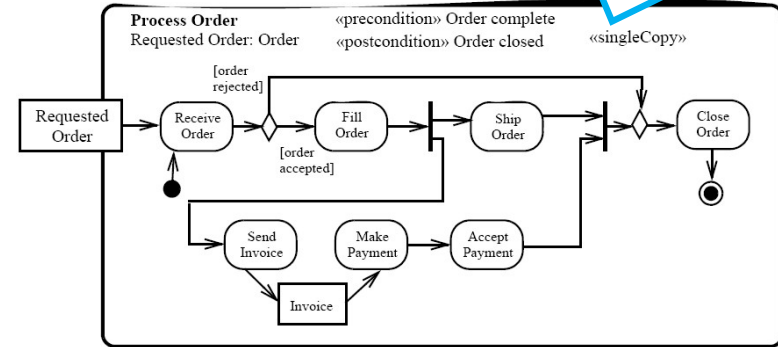Fork

[amount >= 200]

Submit for Approval

[decision = reject]   Reject   Notify Reject

[decision = accept]   Service Approval

Auto Approve

[amount < 200]

No action timer   Notify No Action   Cancel Transaction   Cancel Service

Example 2

Parameter name: type .

Parameter node

Join

«precondition» Order complete
«postcondition» Order closed
«singleCopy»

Process Order
Requested Order: Order

[order rejected]
[order accepted]

Requested Order → Receive Order → Fill Order → Ship Order → Close Order
Send Invoice → Make Payment → Accept Payment
Invoice

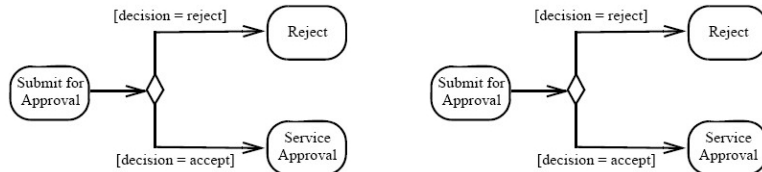
Example 2

- ◆ <<singleCopy>> means the same activity handles all invocations
- ◆ Otherwise, a separate activity for each invocation .

«precondition» Order complete
«postcondition» Order closed
«singleCopy»

Process Order
Requested Order: Order

[order rejected]
[order accepted]

Requested Order → Receive Order → Fill Order → Ship Order → Close Order
Send Invoice → Make Payment → Accept Payment
Invoice

We Learn from Mistakes

[decision = reject] → Reject
Submit for Approval →
[decision = accept] → Service Approval

Activity Diagram

[decision = reject] → Reject
Submit for Approval →
[decision = accept] → Service Approval

State Machine

State machines **are not** *activity diagrams* **and not** *flowcharts* .

11

Activity Diagram with Swim Lanes

Too high level as activity diagram .

«attribute» performingDept: Department

Order Department

Receive Order → Fill Order → Ship Order → Close Order
[order accepted]

Acctg Department

Send Invoice → Invoice
Accept Payment

«external» **Customer**

Make Payment
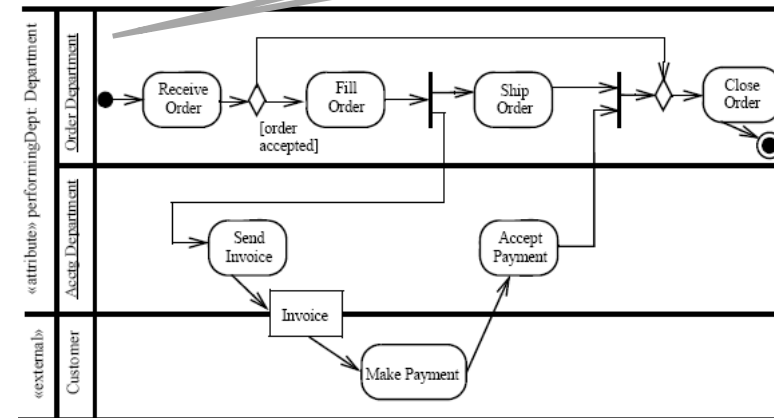
Interaction Overview Diagram

- ◆ *Too high level as activity diagram*
- ◆ *Nodes may be sequence diagrams*
- ◆ *Bad UML feature .*

sd Add a new advert to a campaign if within budget

sd List Campaigns for Client

:CampaignManager

:Client

:Campaign

getName

listCampaigns

loop [For all client's campaigns]
getCampaignDetails

ref Get campaign budget

sd Add costed advert

:CampaignManager

:Campaign

addCostedAdvert

[totalCost <= budget]   [totalCost > budget]

ref Create advert

ref Create request