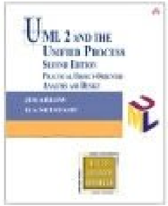
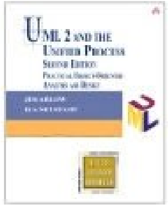
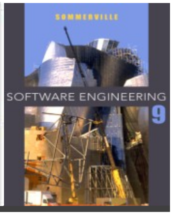


User Interface Design

Lecture 9

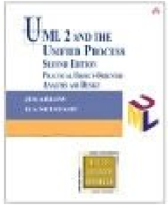
Outline





- History and motivation
- Human limits
- Designing user interface
- Evaluating user interface
- Examples

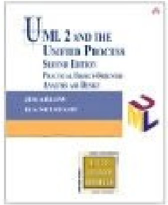
- UML State diagram

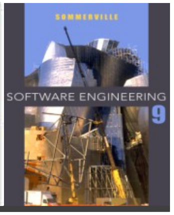


History and Motivation

Lecture 9/Part 1

Importance of user interface



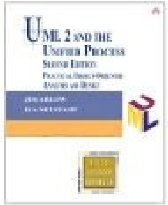


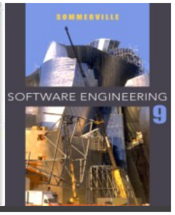
- Computing systems are no longer the province of specialist users.
- Computer rage => aprox. 70% of computer users used violence or offensive language against computers.

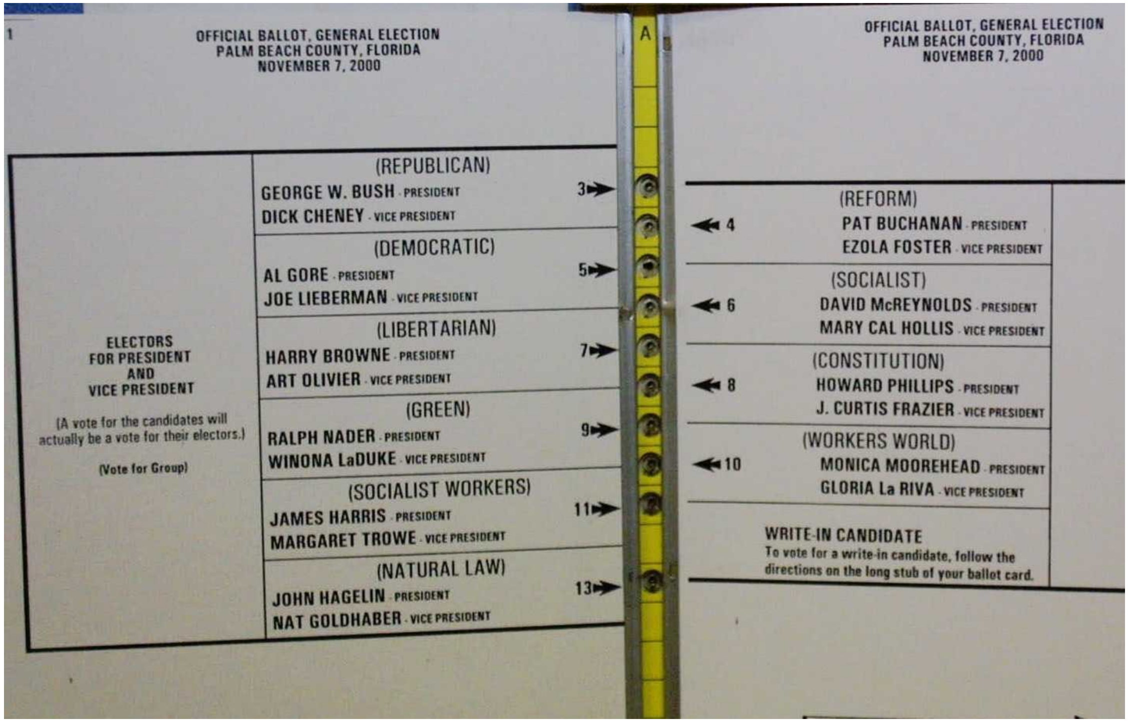
- **Apple iPhone story:**
  - Computer company redefines phone market through one product.

- **The Three Mile Island Nuclear Power Plant Disaster:**
  - Situation misinterpretation (coolant pressure) by the power-plant operators.
  - Oversight of emergency light indicator due to ambiguous control indicators in the power-plant user interface.

US ballot: presidential elections 2000 in Florida







1

OFFICIAL BALLOT, GENERAL ELECTION
PALM BEACH COUNTY, FLORIDA
NOVEMBER 7, 2000

ELECTORS
FOR PRESIDENT
AND
VICE PRESIDENT

(A vote for the candidates will actually be a vote for their electors.)

(Vote for Group)

(REPUBLICAN)
GEORGE W. BUSH - PRESIDENT
DICK CHENEY - VICE PRESIDENT
3→

(DEMOCRATIC)
AL GORE - PRESIDENT
JOE LIEBERMAN - VICE PRESIDENT
5→

(LIBERTARIAN)
HARRY BROWNE - PRESIDENT
ART OLIVIER - VICE PRESIDENT
7→

(GREEN)
RALPH NADER - PRESIDENT
WINONA LaDUKE - VICE PRESIDENT
9→

(SOCIALIST WORKERS)
JAMES HARRIS - PRESIDENT
MARGARET TROWE - VICE PRESIDENT
11→

(NATURAL LAW)
JOHN HAGELIN - PRESIDENT
NAT GOLDHABER - VICE PRESIDENT
13→

A

OFFICIAL BALLOT, GENERAL ELECTION
PALM BEACH COUNTY, FLORIDA
NOVEMBER 7, 2000

← 4
(REFORM)
PAT BUCHANAN - PRESIDENT
EZOLA FOSTER - VICE PRESIDENT

← 6
(SOCIALIST)
DAVID McREYNOLDS - PRESIDENT
MARY CAL HOLLIS - VICE PRESIDENT

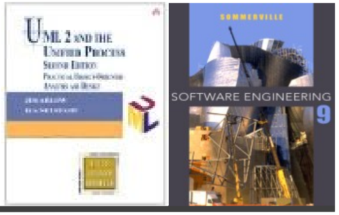
← 8
(CONSTITUTION)
HOWARD PHILLIPS - PRESIDENT
J. CURTIS FRAZIER - VICE PRESIDENT

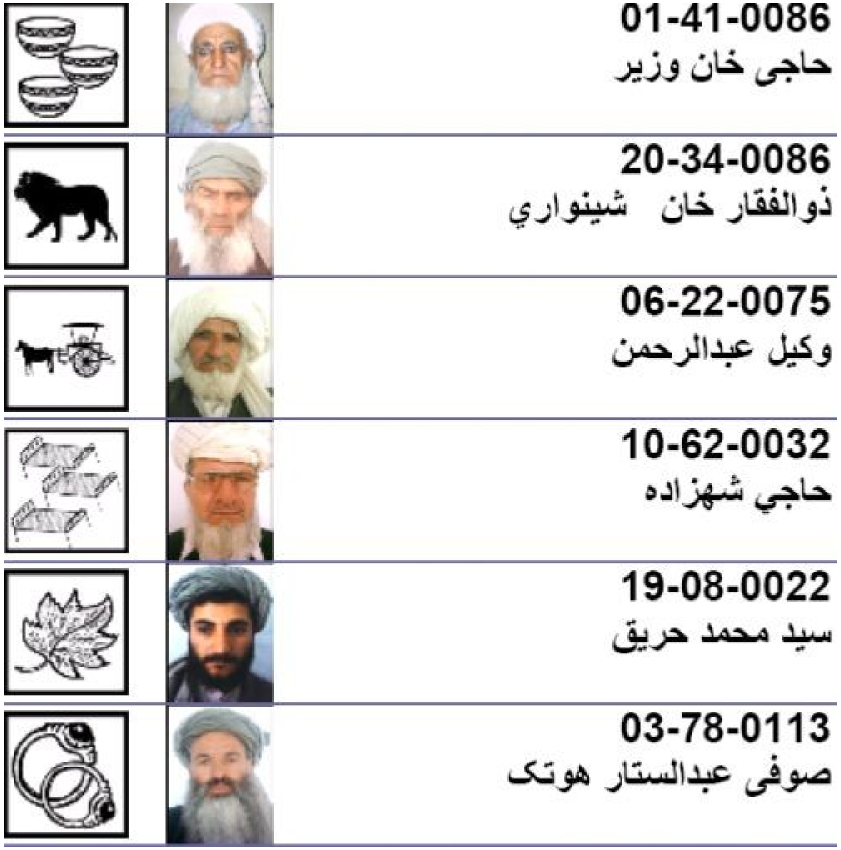
← 10
(WORKERS WORLD)
MONICA MOOREHEAD - PRESIDENT
GLORIA La RIVA - VICE PRESIDENT

WRITE-IN CANDIDATE
To vote for a write-in candidate, follow the directions on the long stub of your ballot card.

- ✧ Ballot misunderstanding suspected to decide the election.
- ✧ Major recount dispute followed, which delayed the outcome for more than a month.

Afghanistan ballot





01-41-0086
حاجی خان وزیر

20-34-0086
ذوالفقار خان   شینواري

06-22-0075
وکیل عبدالرحمن

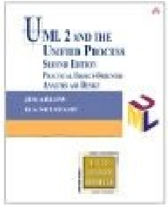
10-62-0032
حاجي شهزاده

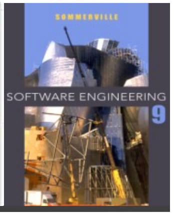
19-08-0022
سید محمد حریق

03-78-0113
صوفی عبدالستار هوتک

✧ So simple that even illiterate person can vote

Human-computer interaction (HCI)



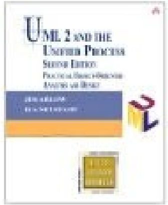


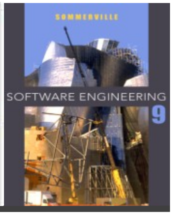
- ✧ HCI is the study of how humans interact with computer systems. It involves both art and science.

- ✧ Many disciplines contribute to HCI, including human factors (ergonomics of human limits), computer science, psychology, ergonomics, engineering, and graphic design.

- ✧ **User interface design** aims at system design with the focus on the user's experience and interaction.

User-centered design and development

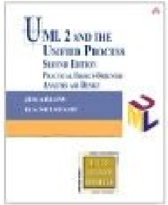


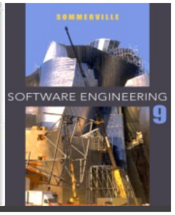


- ✧ The main principles of user-centered design:
  - The active involvement of users
  - An appropriate allocation of function between user and system
  - The iteration of design solutions
  - Multidisciplinary design teams

- ✧ The essential user-centered design activities:
  - Understand and specify the context of use
  - Specify the user and organizational requirements
  - Produce design solutions (prototypes)
  - Evaluate designs with users against requirements

Why developers should’t design user interface





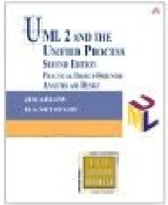
✧ Developers usually focus more on internal product quality than on system usability

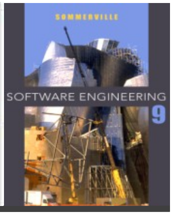
✧ Developers use different mental model than users:

- User’s mental model is based on metaphors and previous experience with similar applications
- Developer’s mental model is based on the knowledge of internal system architecture

✧ User interfaces don't have to conform with domain model

EXAMPLE



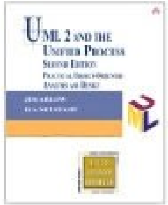


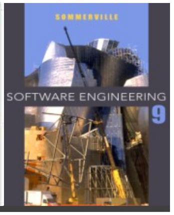
✧ Consider a tablet without hardware brightness-control buttons

✧ Engineers placed software brightness control to POWER MANAGEMENT section. From their point of view it is the right place since brightness influences battery life.

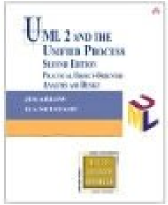
✧ From user point of view, a more proper place for such setting is DISPLAY.

Terms





- **WIMP paradigm (1973):** Windows, Icons, Menus and Pointing device.
- **Usability:** efficient, easy to learn and satisfying to use user interface.
- **User Experience:** feel about using a software.
- **Look & Feel:** induces user experience and product identification. Look can be imitated easily (colors and shapes), but feel (dynamic behavior) cannot.
- **Human Interface Guidelines:** set of platform specific recommendations provided to developers, thus users can carry skill at a standardized interface from one application to another.


UML 2 AND THE UNIFIED PROCESS

SECOND EDITION

SOMMERVILLE

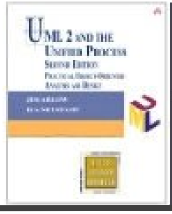
SOFTWARE ENGINEERING

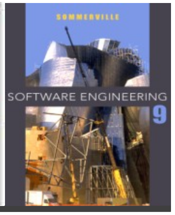
9

Human Limits

Lecture 9/Part 2

Laws of human limits





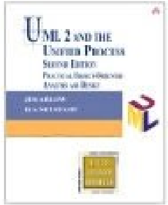
Fitts' law (1954)

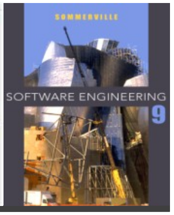
- Model of human movement, predicts the time required to hit a target:
  - physically with a hand or finger,
  - virtually with a pointing device.
- Given by the distance, width of the target and other coefficients.

Hick's law (1953)

- Predicts the time required to select one item from a list.
- Given by the reaction time and entropy of the choices.

Memory

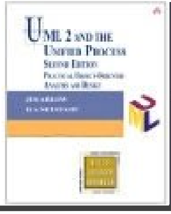




- Short-term memory: 7+2 elements

- Important for example for proper amount of items in the menu.

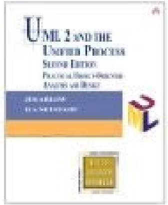
- Long-term memory



Designing Good User Interface

Lecture 9/Part 3

Rules

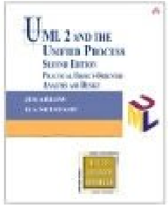


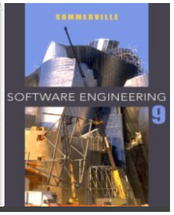
- **Consistence:** similar objects should behave similarly, important factor for predictability

- Always provide proper **feedback** to user:
  - Weak feedback: user may perceive, e.g. tool tip
  - Strong feedback: user must perceive, e.g. dialog box

- Prevention and toleration of users mistakes

Fundamental UI design principles (by Apple)





✧ Metaphors

- Take advantage of people’s knowledge of the world by using metaphors to convey concepts and features of your app.
- E.g. folders to organize documents

✧ Mental model

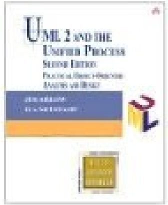
- The user already has a mental model that describes the task your software is enabling. Respect user expectations and strive for familiarity, simplicity, availability and discoverability.
- E.g. the process of sending a letter

✧ Explicit and implied actions

- Explicit actions clearly state the result of manipulating an object.
- Implied actions depend on cues and contexts (drag and drop).
- Keep these two paradigms in mind as you design your UI.

© Apple Inc.

Fundamental UI design principles (by Apple)



✧ Direct manipulation

- Allows users to feel that they are controlling the objects represented by the computer.
- E.g. drag and drop

✧ See and point

- Based on the noun-then-verb paradigm, where the noun (icon) is selected first and then the possible verb list (action menu) browsed.

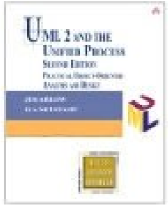
✧ User control

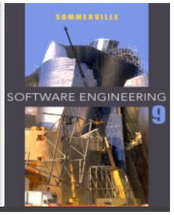
- It should always be the user who controls the situation.

✧ Feedback and communication

© Apple Inc.

Fundamental UI design principles (by Apple)





✧ Consistency

- Visual and behavioral UI consistency with the product itself, with the platform, previous product versions, user expectations.

✧ WYSIWYG

✧ Forgiveness

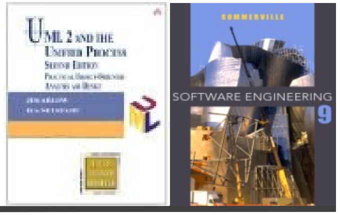
✧ Perceived stability

- The user always feels better in a stable and familiar environment, where e.g. icons do not disappear when inactive.

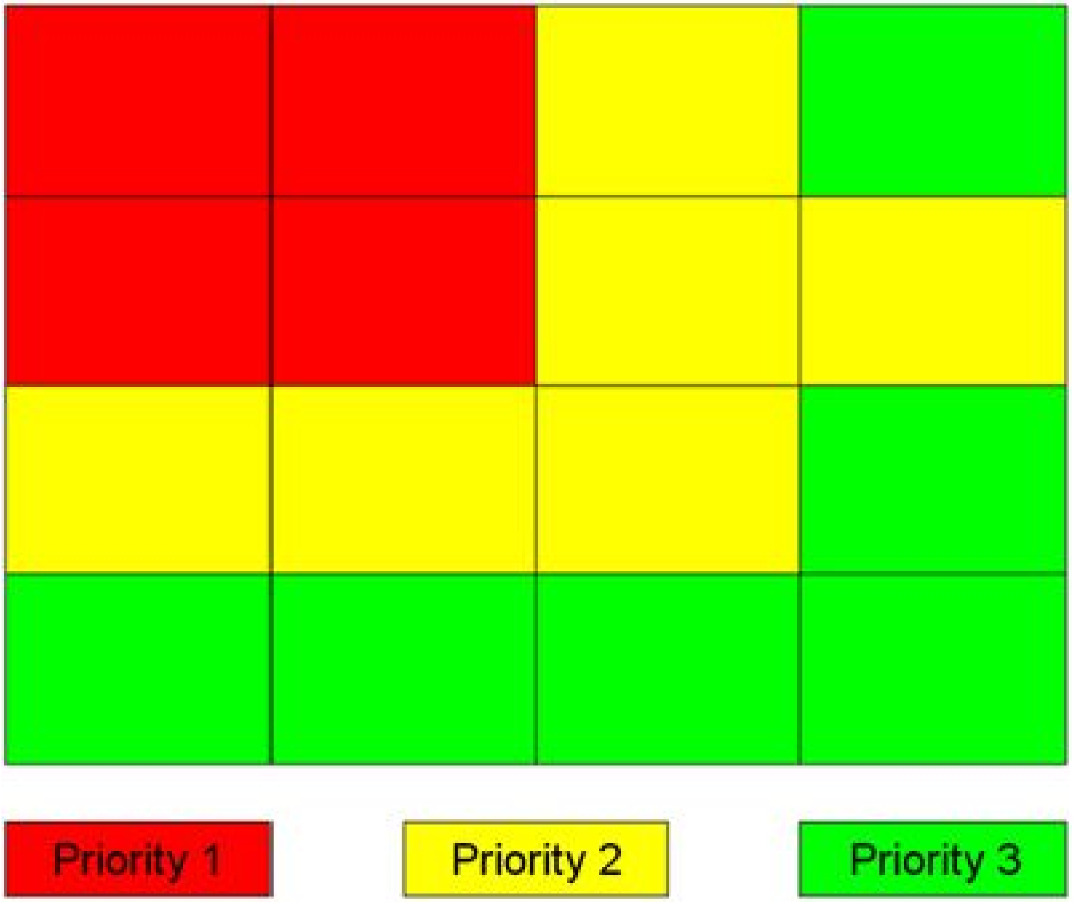
✧ Aesthetic integrity

- Your product should look pleasant on the screen, even when viewed for a long time.

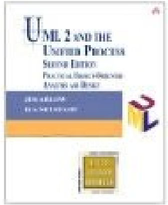
Prominent positions on screen

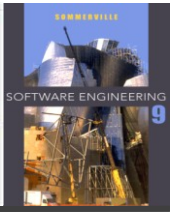


- ✧ Position is preferred over graphical highlight.
- ✧ Observed by EyeTracker device.



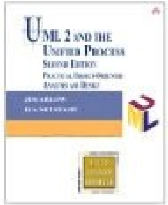
Cross-platform GUI

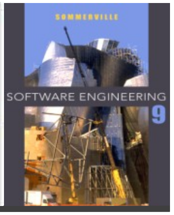




- Do not use (different platform metaphors and Look&Feel)
- Works everywhere => ugly and less usable everywhere
  - It is like designing a house without knowledge where the house will be located (city, village, mountains).
- May adapt Look to particular platform, but Look itself is not Look&Feel

Point and click vs. touch





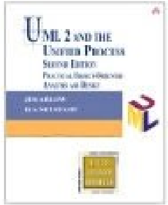
- ✧ Different paradigms, should not be combined.

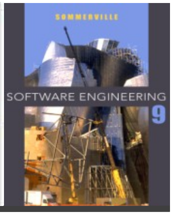
- ✧ Touch supports different model of:
  - Cursors – not only input but also output indicator, e.g. busy cursor
  - Mouse over indication

- ✧ Touch interface should support more direct manipulation and especially the undo operation, to be safe against user inaccuracy.

- ✧ Touch is not suitable for difficult conditions, like turbulence in aircraft.

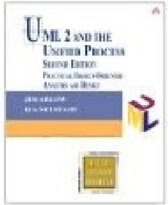
Always follow Human-Interface Guidelines (HIG) if available

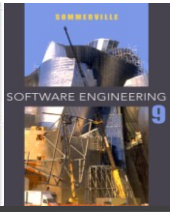




- ✧ HIG describe especially proper use of components, e.g. distances between buttons, labels.
- ✧ Look inside Windows and/or Mac OS X HIG is recommended.
- ✧ Linux user interface guidelines are not competitive to above mentioned ones, thus such applications don’t provide such a standardized user interface interface and Look&Feel.

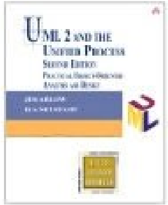
WEB

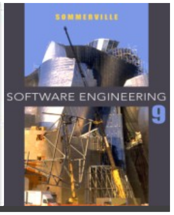




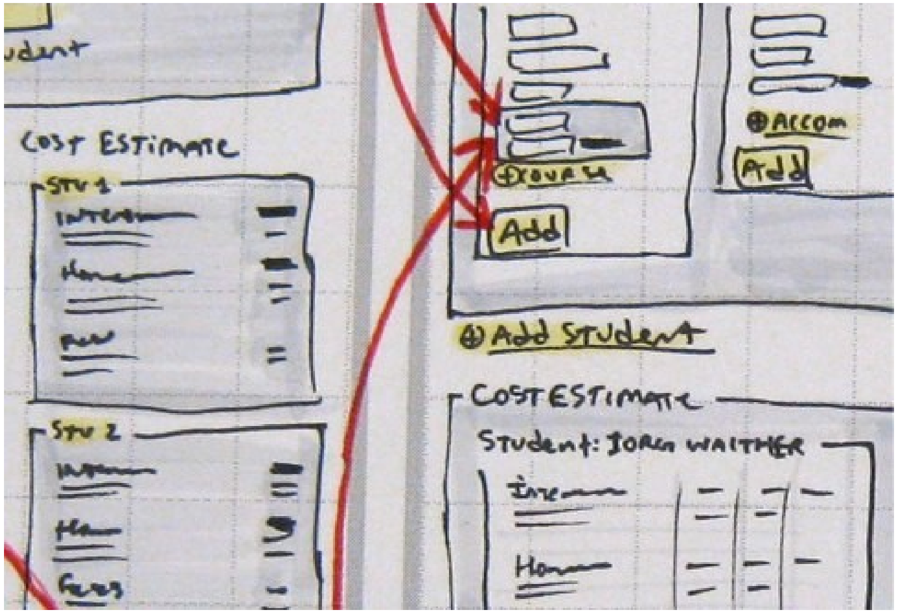
- No strict HIG.
- Designed for content consuming instead of creating.
- Do not try to imitate desktop applications.
- Support browser integrated navigation controls: Next and Previous page.
- Native HTML (HTML5) with CSS is always preferred over non standardized ones like Adobe Flash and Microsoft Silverlight.

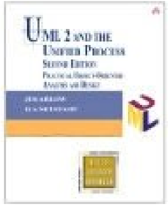
Prototyping





- ✧ Always make a prototype first.
- ✧ We distinguish between:
  - Wireframes – initial sketches
  - Mockups – models of a design used for demonstration or evaluation
  - Prototypes – early (partly-working) samples of the software

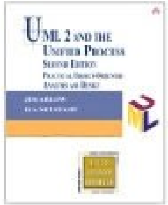


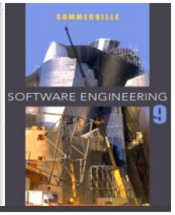


Evaluating User Interface

Lecture 9/Part 4

Evaluation techniques



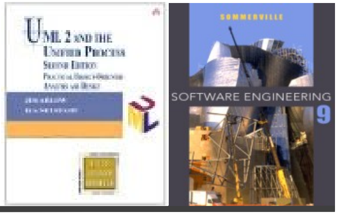


- Interviews (unstructured, semi-structured, structured) and user observation – easy, very useful for beginners.

- Usability Testing (qualitative and quantitative measures):
- Quantitative – time to complete task, error rates
- Qualitative – questionnaires and surveys, subjective

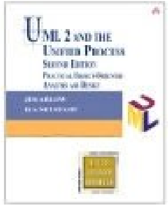
- Field Studies – Complex studies used whenever UI is very critical, time consuming, considers many factors.

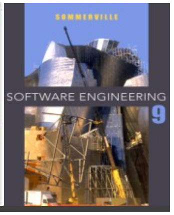
Direct user observation



- Underrated technique
- Useful for beginners in HCI
- Can be combined with qualitative and quantitative measures
- User’s screen and face car be recorder
- Useful for task simulations, may be supplied with e.g. simulated helpdesk

Eye Tracker



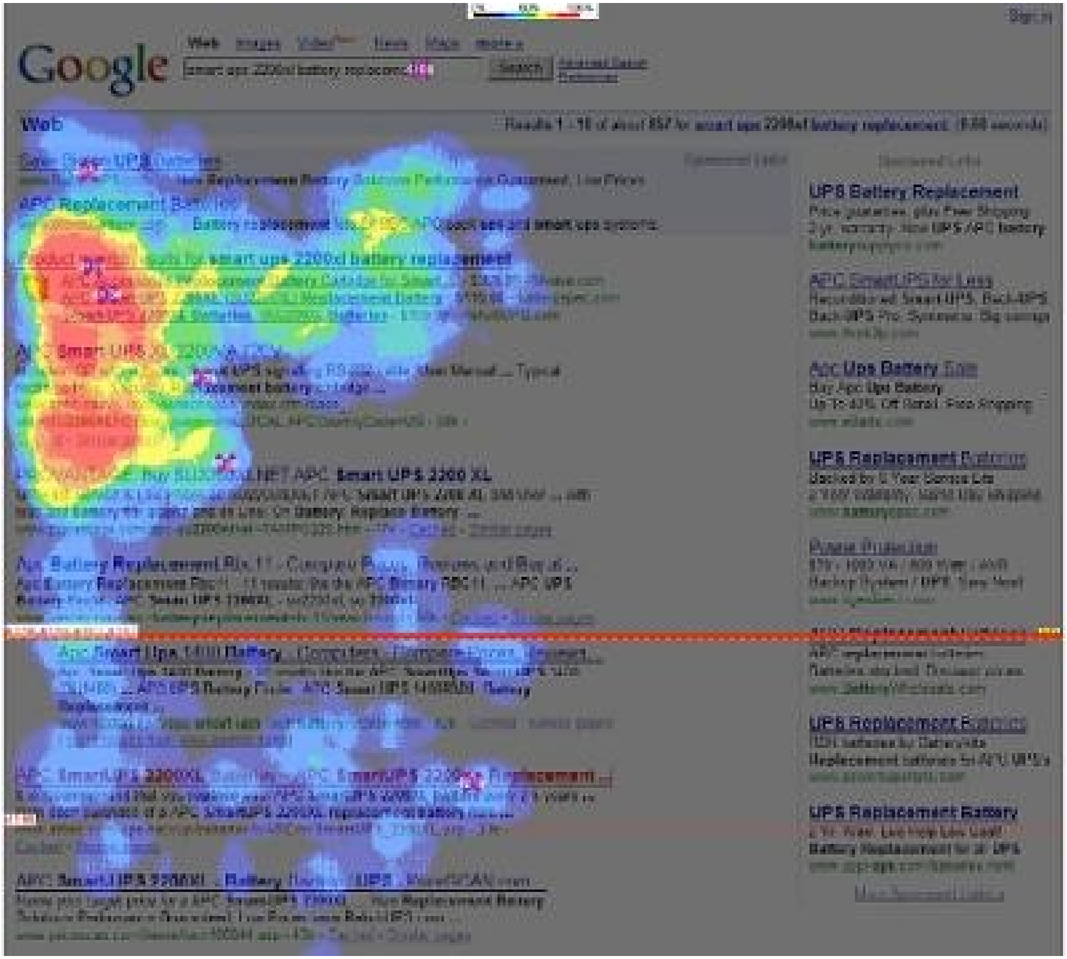


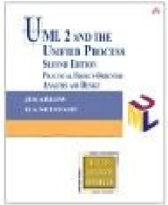
- ✧ Measures the point which the subject is looking at

Output:

- ✧ Heat map
- ✧ Video of a focus point on the interface

- ✧ Useful for marketing

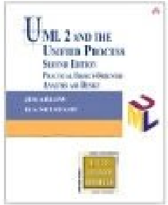


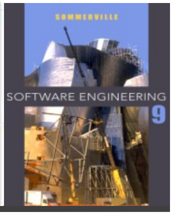


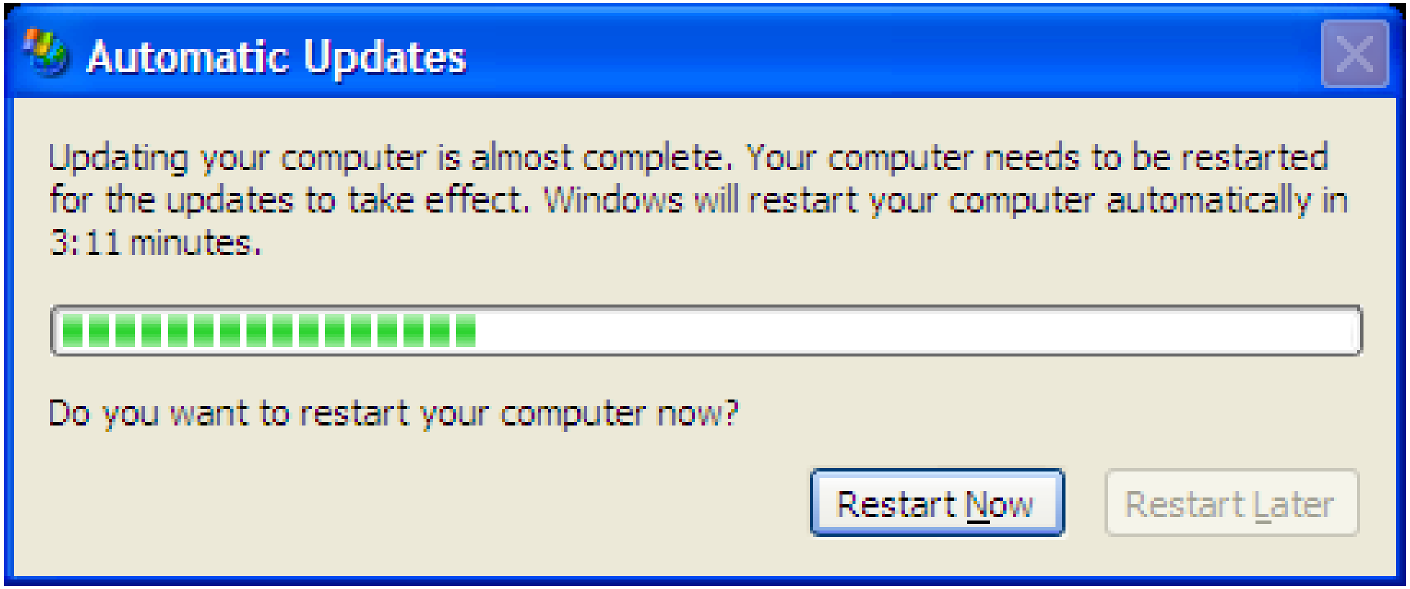
Examples

Lecture 9/Part 5

Who controls the situation?





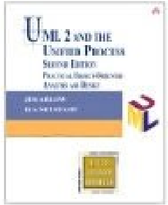


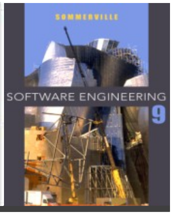
Updating your computer is almost complete. Your computer needs to be restarted for the updates to take effect. Windows will restart your computer automatically in 3:11 minutes.

Do you want to restart your computer now?

✧ "Restart Later" option has been disabled, but is still visible, just to taunt you.

Apple: Ejecting disk through trash can



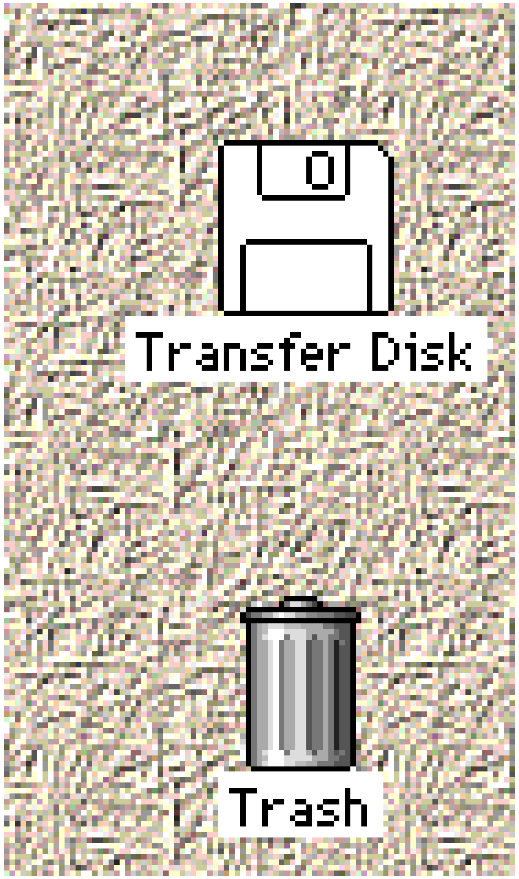


- ✧ What happens when you drag the disk into the trash can?

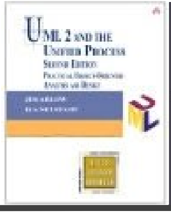
- ✧ Erase the whole disk or eject?

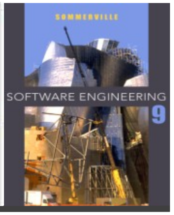
- ✧ Can user be sure without experiment? => Learning through exploration

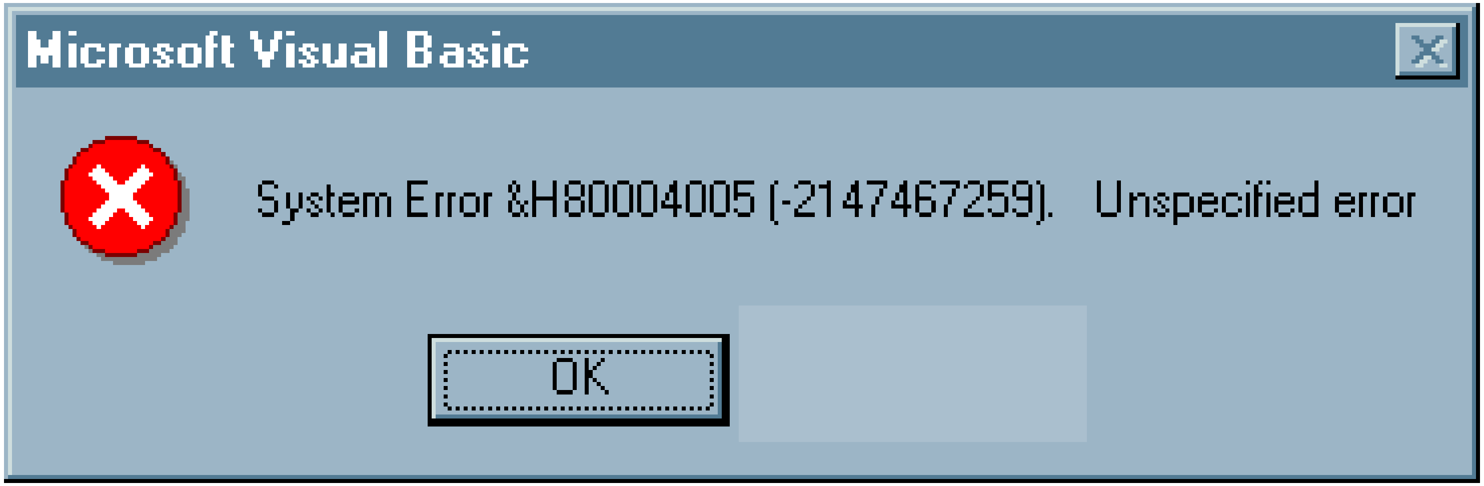
- ✧ Apple later changed the concept.



Error messages



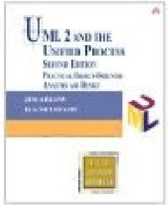


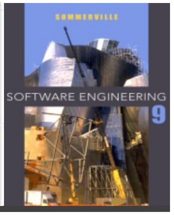


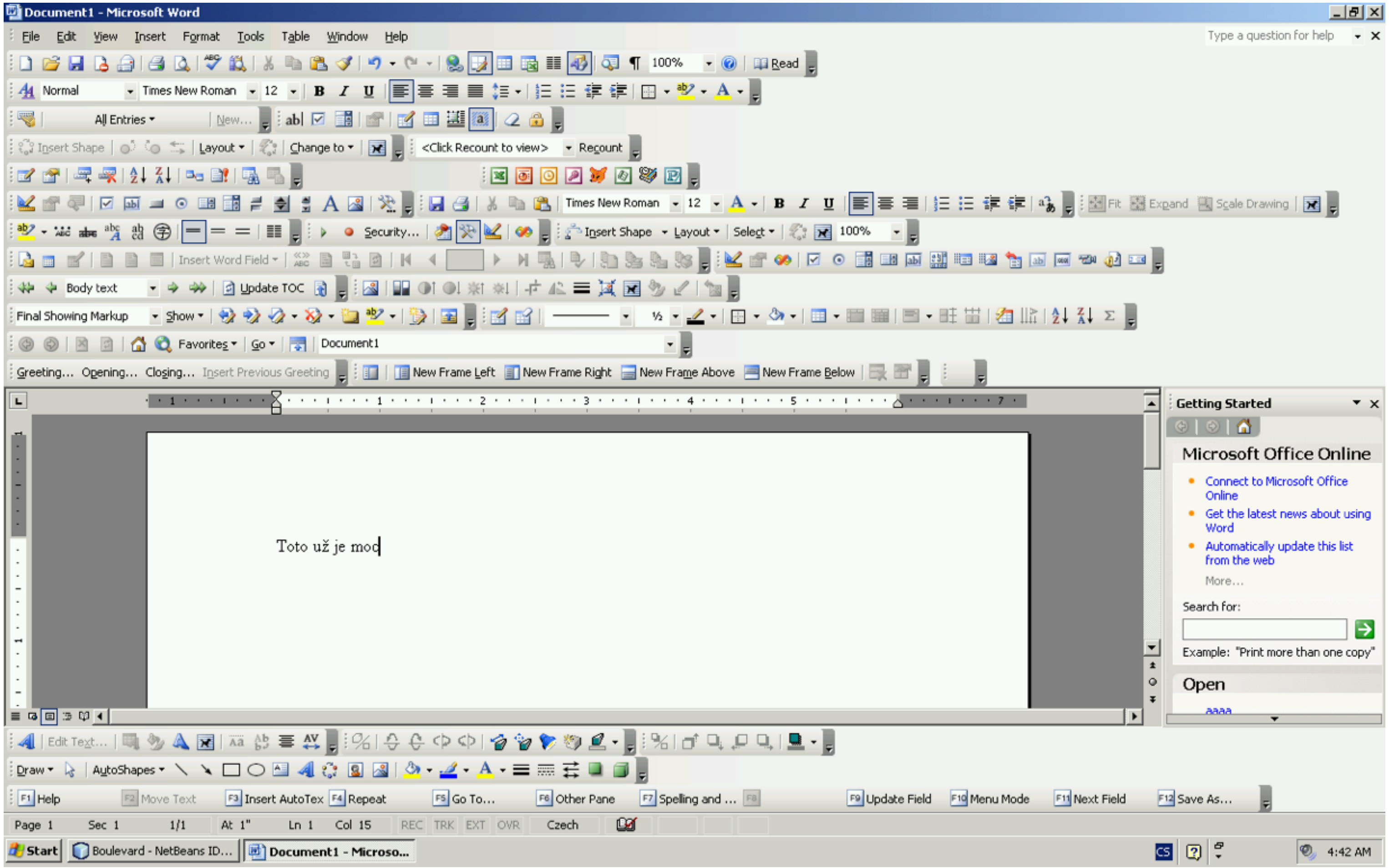
System Error &H80004005 (-2147467259). Unspecified error

- ✧ Does it explain anything to the user?
- ✧ Errors are never “OK”, use “Continue” or “Exit” instead.
- ✧ Always provide feedback in order to help the user with the situation.

Bloatware
(creeping featurism, feature war)

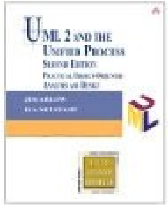
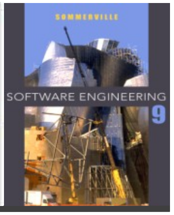


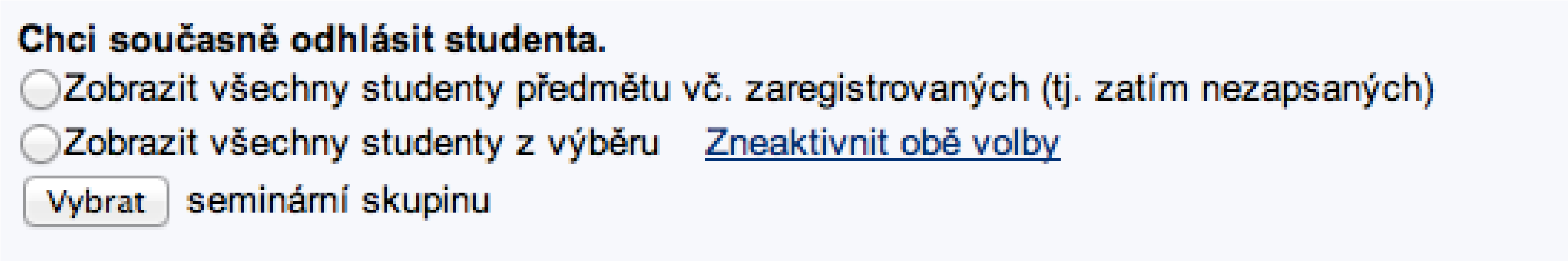




Toto už je moc

IS



Chci současně odhlásit studenta.

○ Zobrazit všechny studenty předmětu vč. zaregistrovaných (tj. zatím nezapsaných)

○ Zobrazit všechny studenty z výběru [Zneaktivnit obě volby]

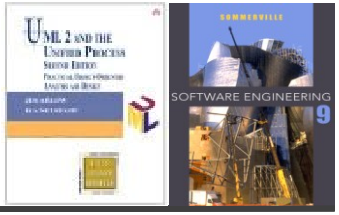
seminární skupinu

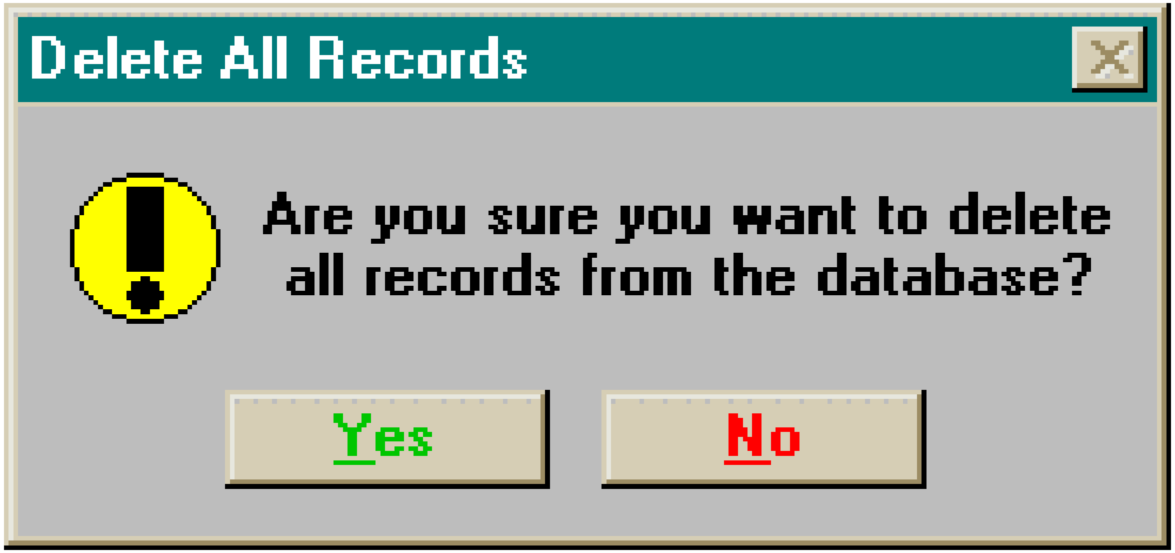
- Correct usage of control features is fundamental for good design!

Main defects:

- Hyperlink “Zneaktivnit obě volby” should be replaced with third radio-button meta-option named “NONE”.
- Label “seminární skupinu”

Colors



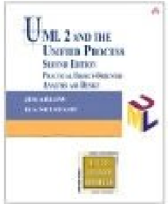


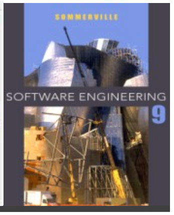
Delete All Records

Are you sure you want to delete all records from the database?

- ✧ Beware of proper color and symbol use

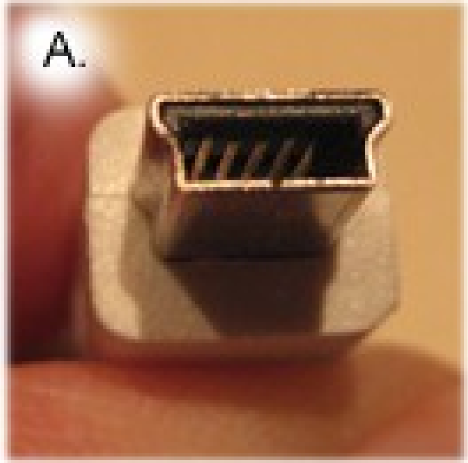
USB

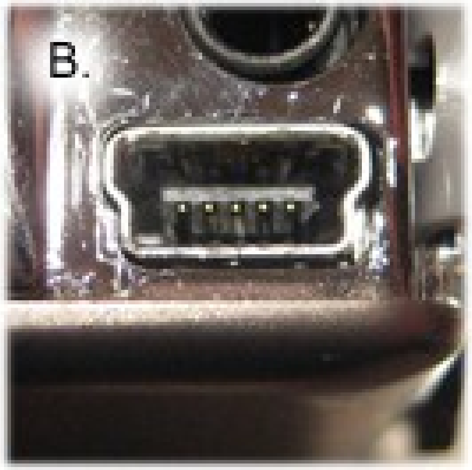


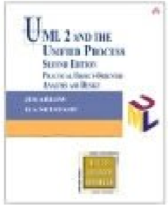


- ✧ Have you ever tried to plug the USB turn the wrong way? Why?

- ✧ Why A and B are better?



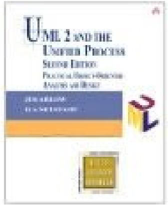




UML State Diagram

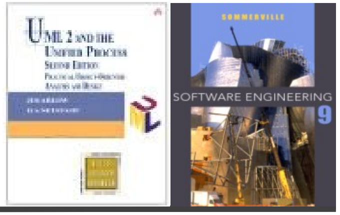
Lecture 9/Part 6

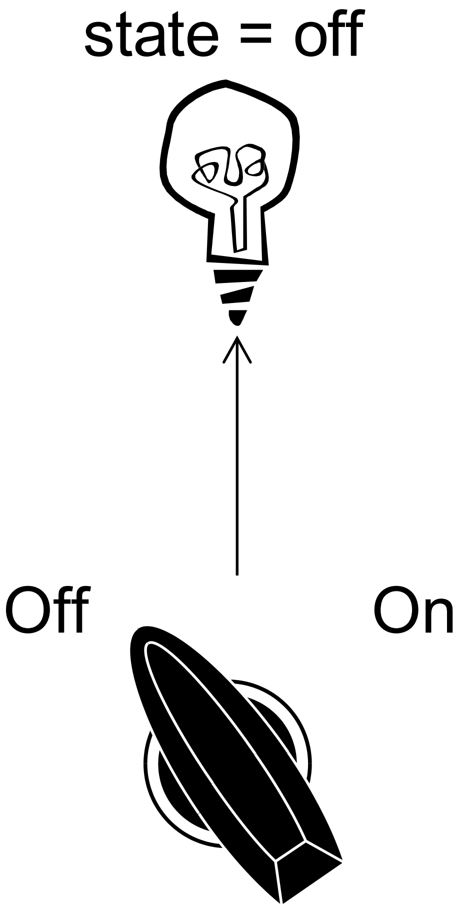
State machines

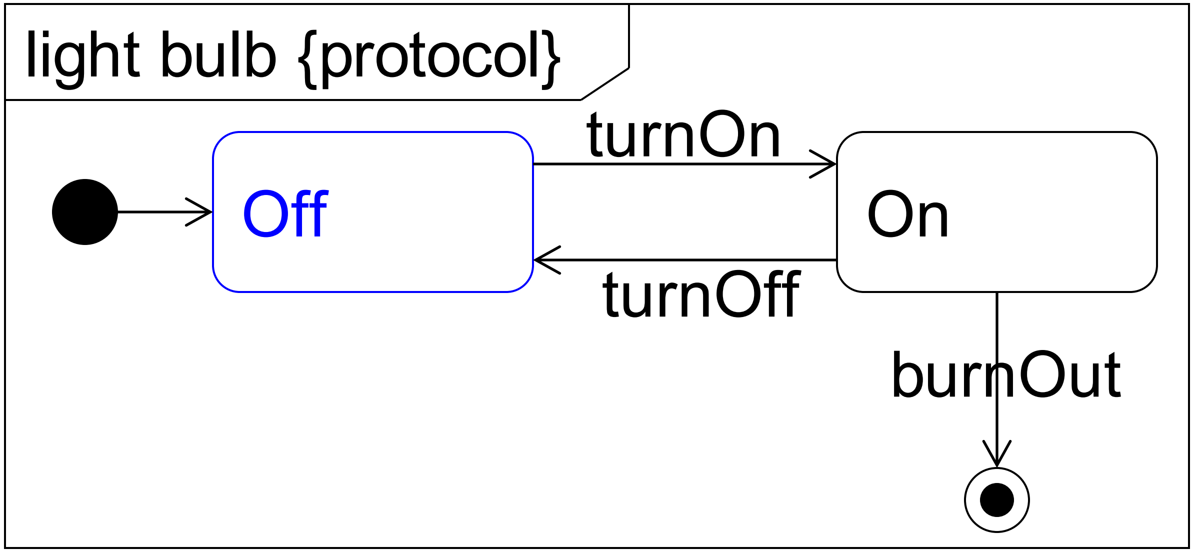


- Some model elements such as classes, use cases and subsystems, can have interesting dynamic behavior - state machines can be used to model this behaviour
- Every state machine exists in the context of a particular model element that:
  - Responds to events dispatched from outside of the element
  - Has a clear life history modelled as a progression of *states, transitions* and *events.* We'll see what these mean in a minute!
  - Its current behaviour depends on its past
- A state machine diagram always contains exactly one state machine for one model element
- There are two types of state machines (see next slide):
  - *Behavioural* state machines - define the behavior of a model element e.g. the behavior of class instances
  - *Protocol* state machines - Model the protocol of a classifier
    - The conditions under which operations of the classifier can be called
    - The ordering and results of operation calls
    - Can model the protocol of classifiers that have no behavior (e.g. interfaces and ports)

State machine diagrams

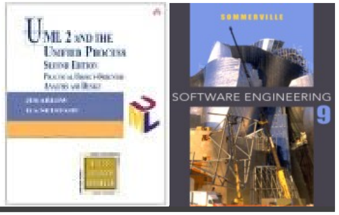


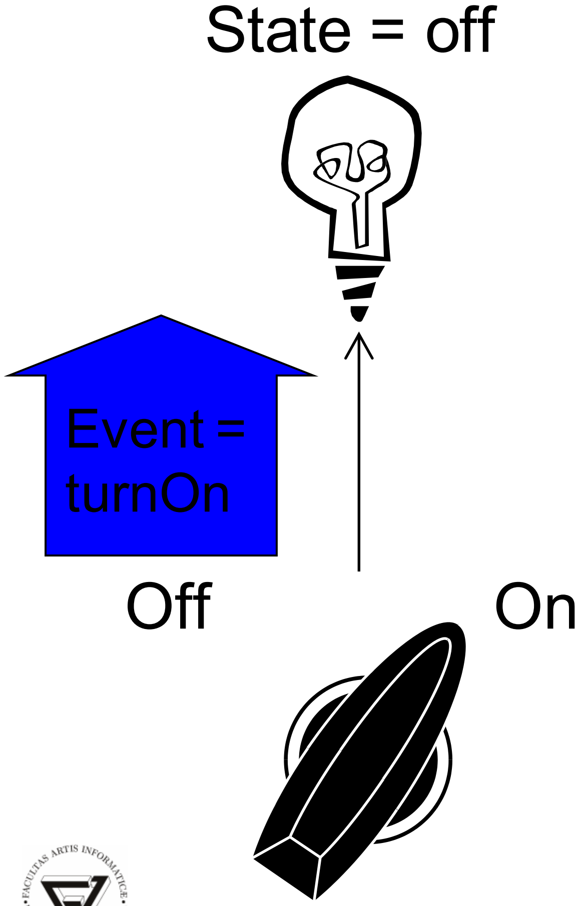


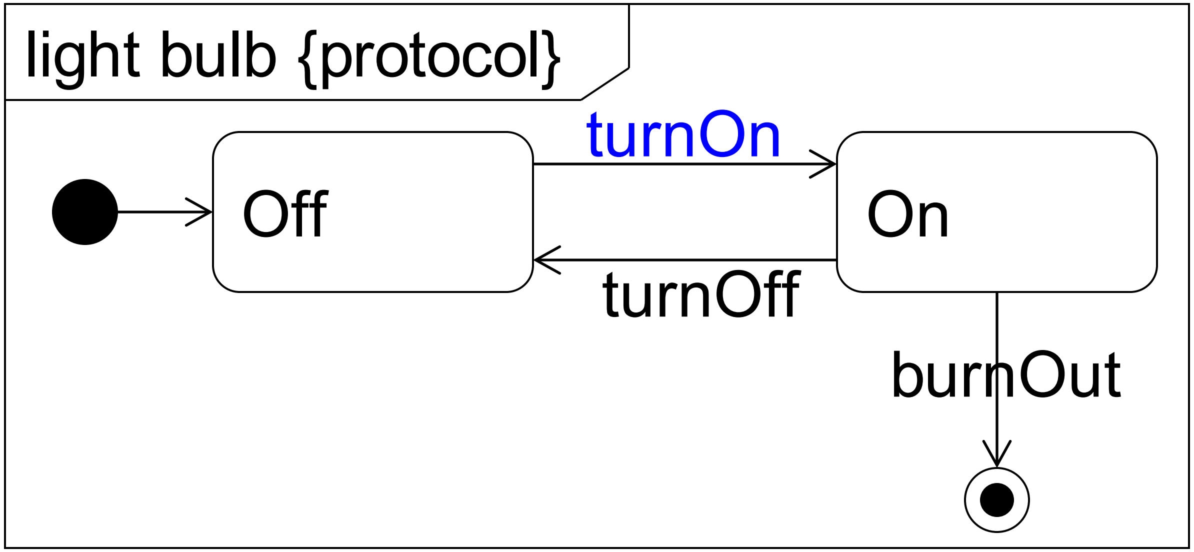


- We begin with the light bulb in the state off

Light bulb turnOn

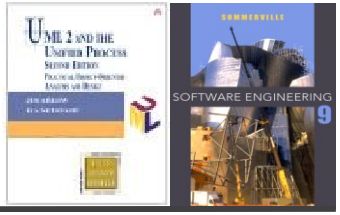


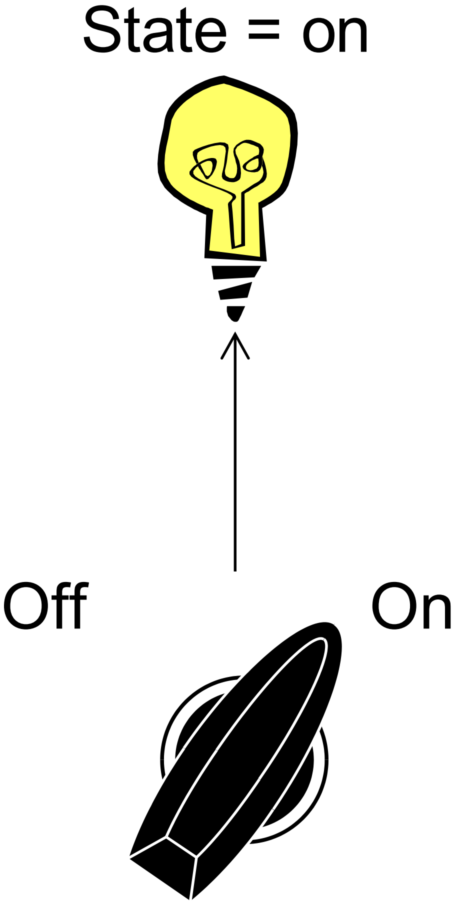


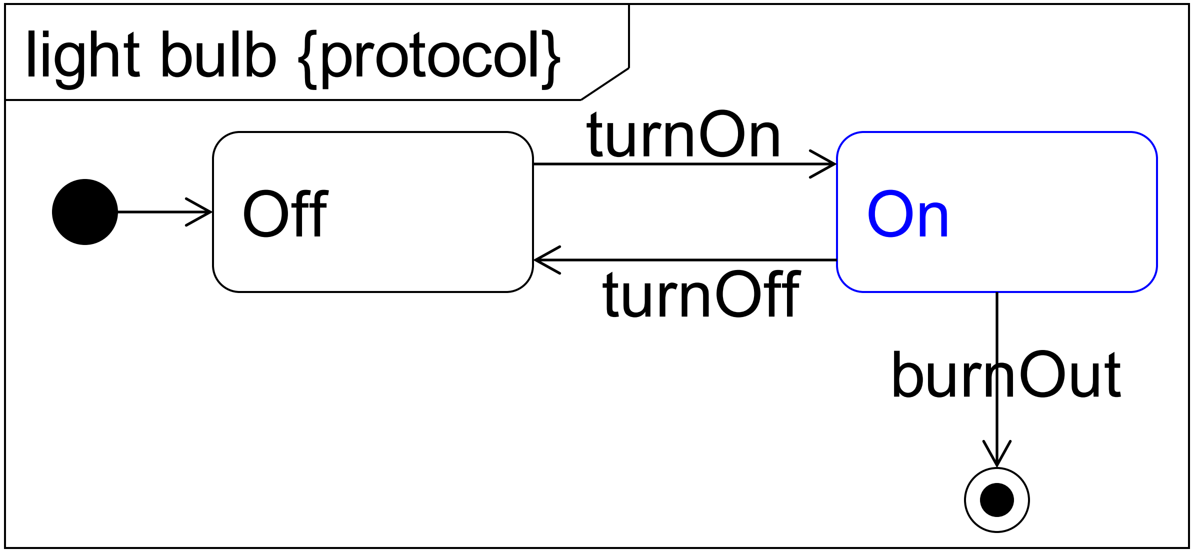


- ✧ We throw the switch to On and the event turnOn is sent to the lightbulb

Light bulb On

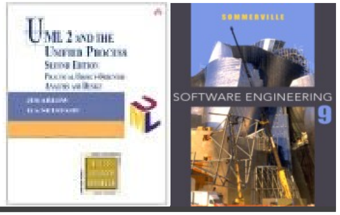


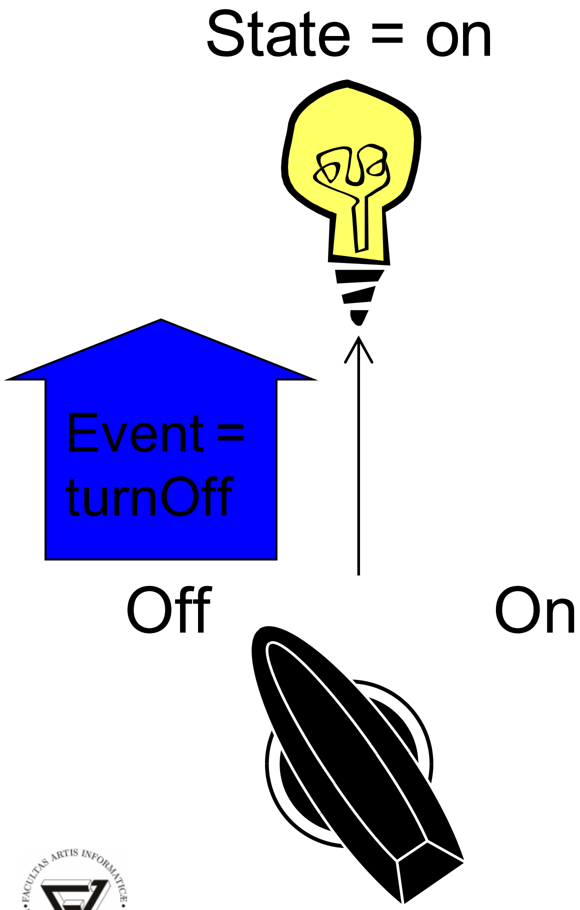


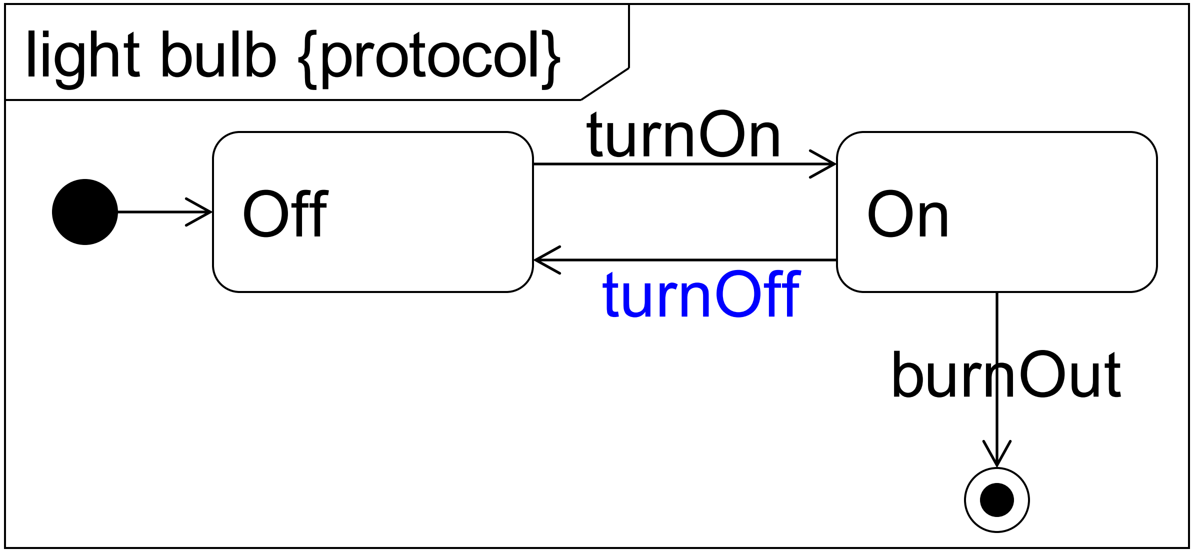


- The light bulb turns on

Light bulb turnOff

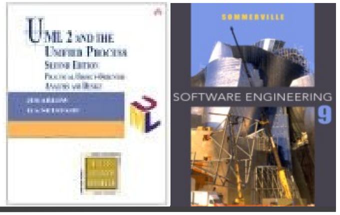


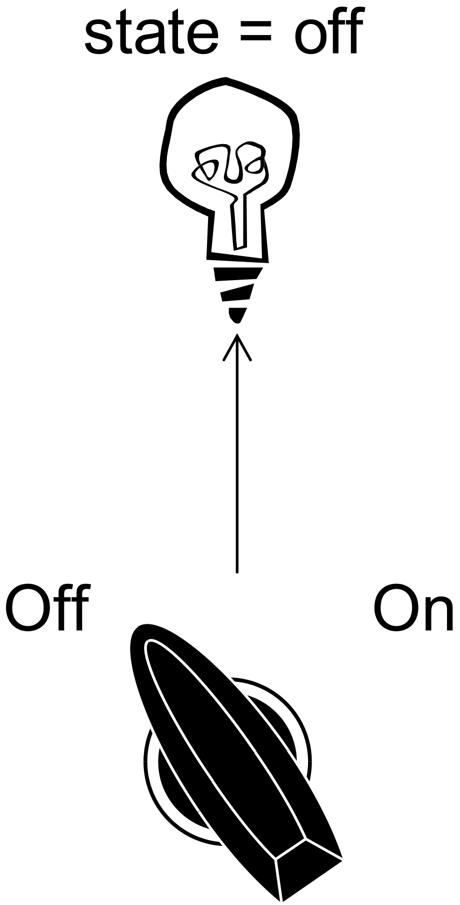


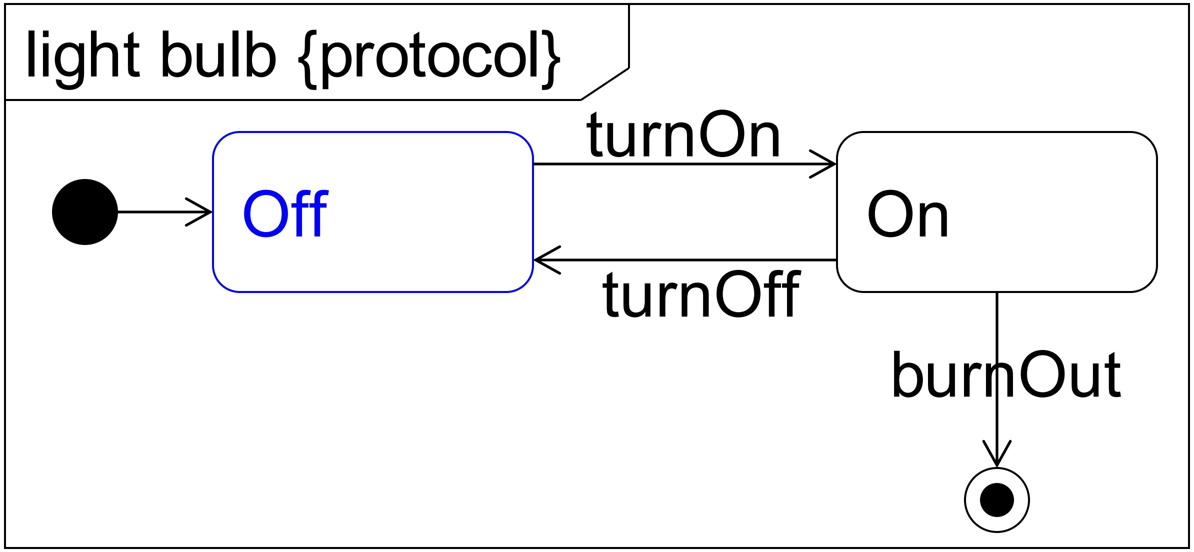


- We turn the switch to Off. The event turnOff is sent to the light bulb

Light bulb Off

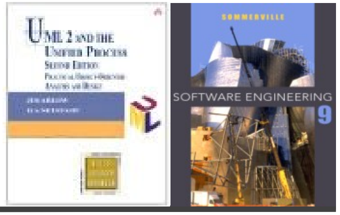


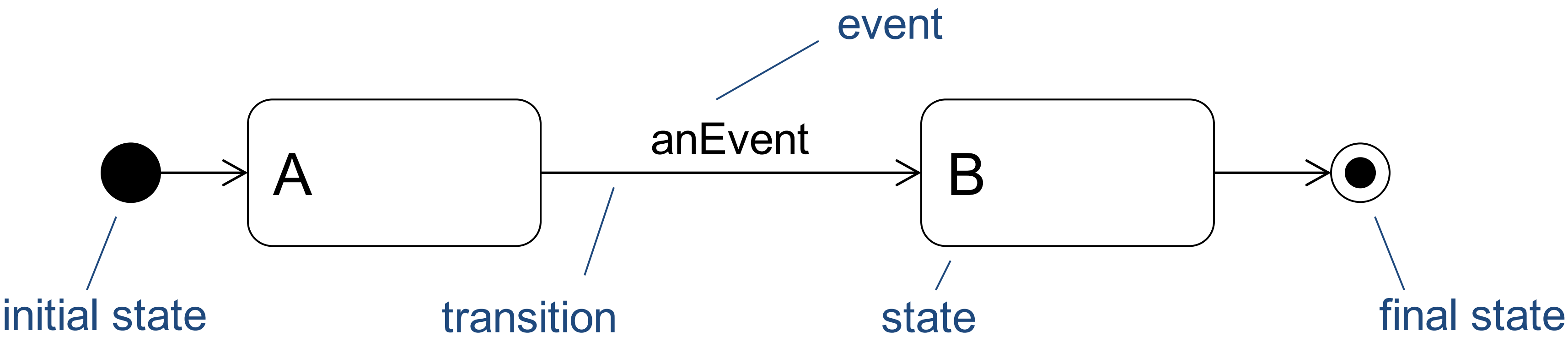




- The light bulb turns off

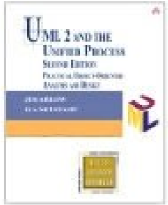
Basic state machine syntax





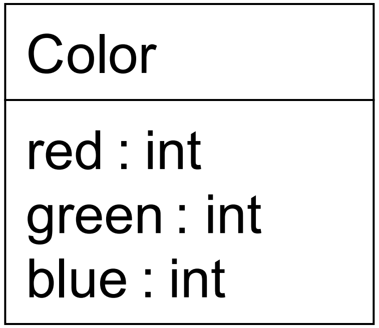
- Every state machine should have a initial state which indicates the first state of the sequence
- Unless the states cycle endlessly, state machines should have a final state which terminates the sequence of transitions
- We'll look at each element of the state machine in detail in the next few slides!

States



- "A condition or situation during the life of an object during which it satisfies some condition, performs some activity or waits for some event"
- The state of an object at any point in time is determined by:
  - The values of its attributes
  - The relationships it has to other objects
  - The activities it is performing

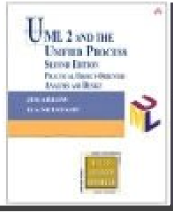
How many states?

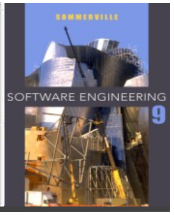


Color

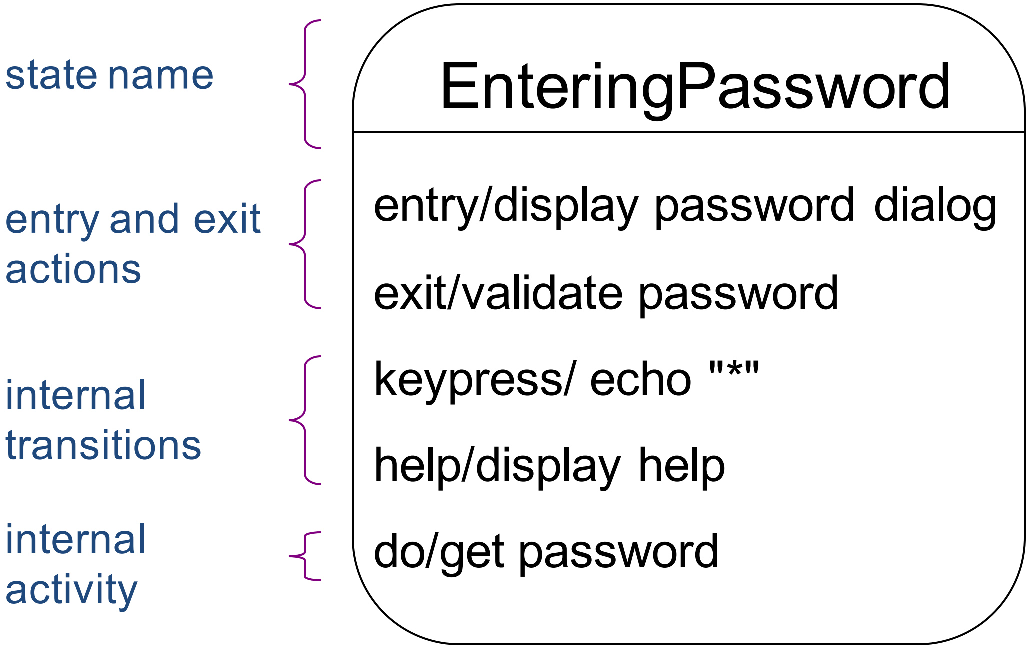
red : int
green : int
blue : int

State syntax





- Actions are *instantaneous* and *uninterruptible*
  - Entry actions occur immediately on entry to the state
  - Exit actions occur immediately on leaving the state
- Internal transitions occur *within* the state. They do *not* transition to a new state
- Activities take a finite amount of time and are interruptible



state name

entry and exit actions

internal transitions

internal activity

EnteringPassword

entry/display password dialog

exit/validate password

keypress/ echo "*"

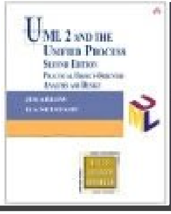
help/display help

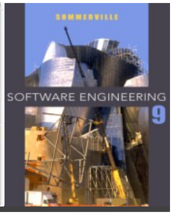
do/get password

Action syntax: eventTrigger / action

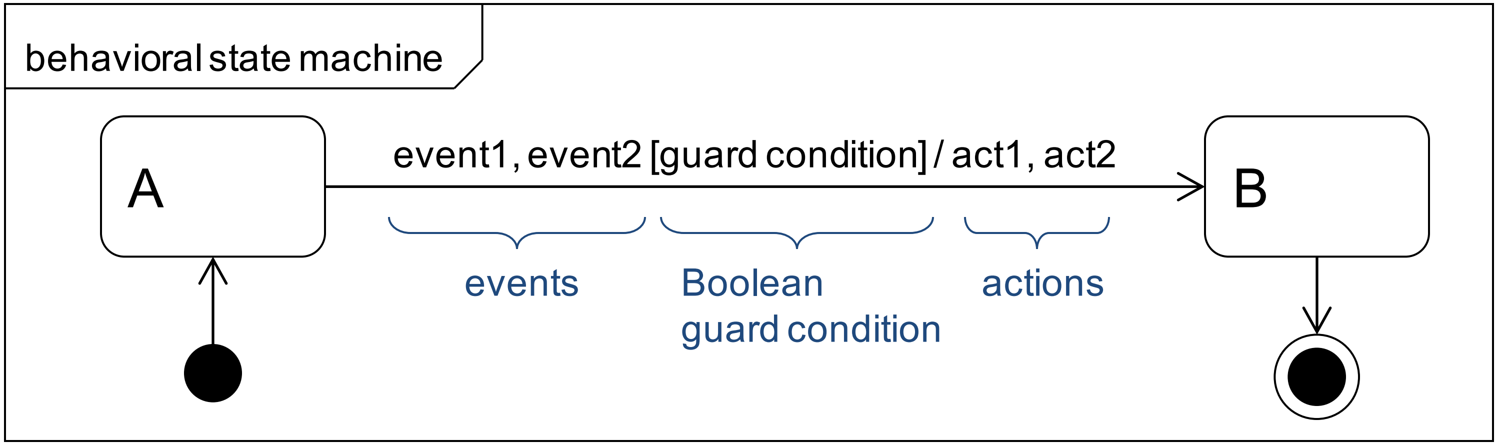
Activity syntax: do / activity

Transitions





behavioral state machine



behavioral state machine

A

event1, event2 [guard condition] / act1, act2

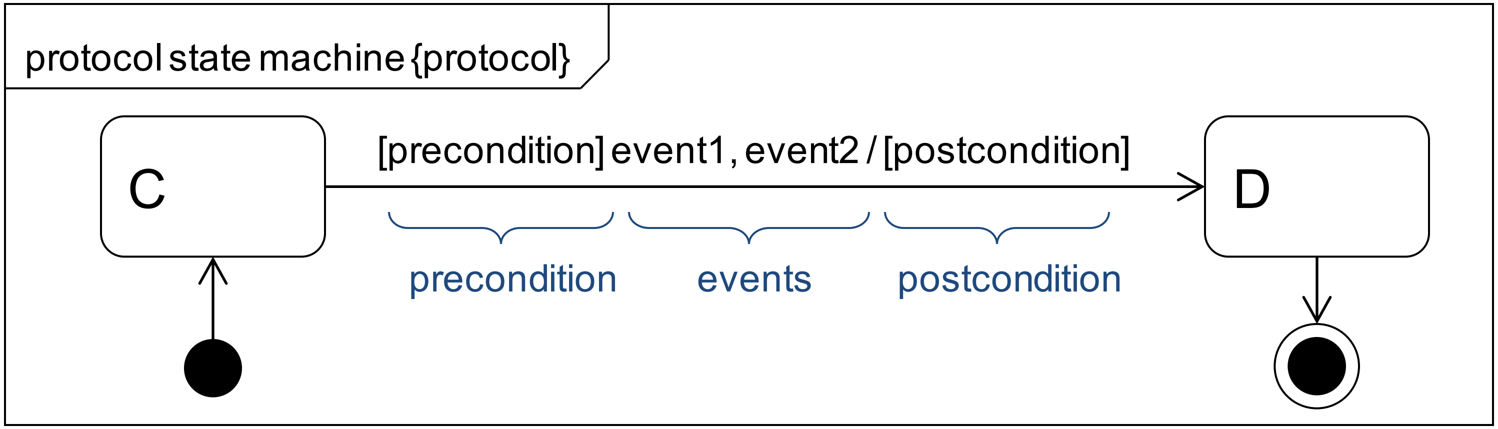
B

events

Boolean guard condition

actions

protocol state machine



protocol state machine {protocol}

C

[precondition] event1, event2 / [postcondition]

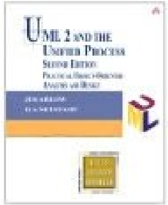
D

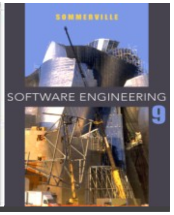
precondition

events

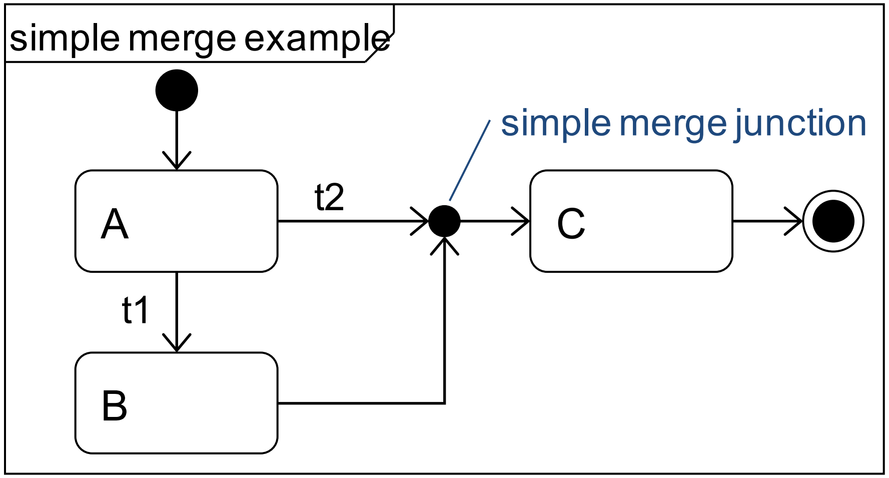
postcondition

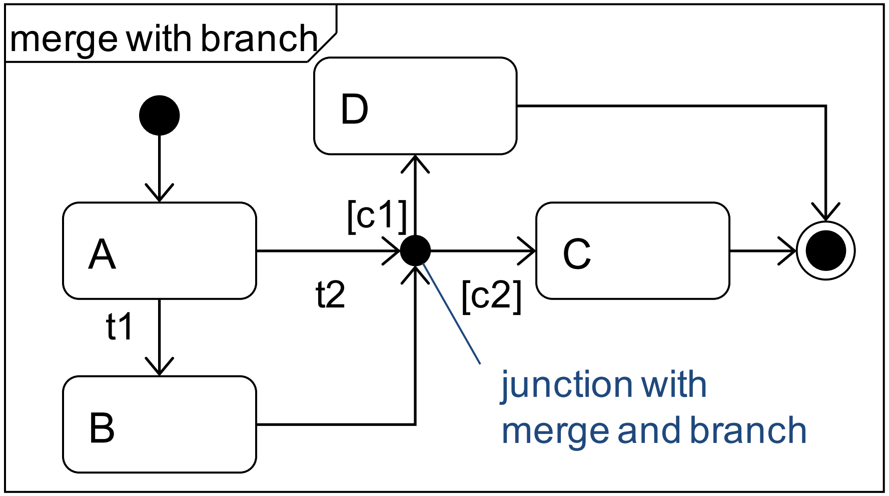
Connecting - the junction pseudo state



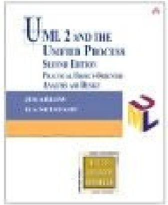


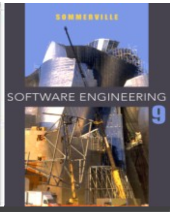
- The junction pseudo state can:
  - connect transitions together (merge)
  - branch transitions
- Each outgoing transition must have a mutually exclusive guard condition



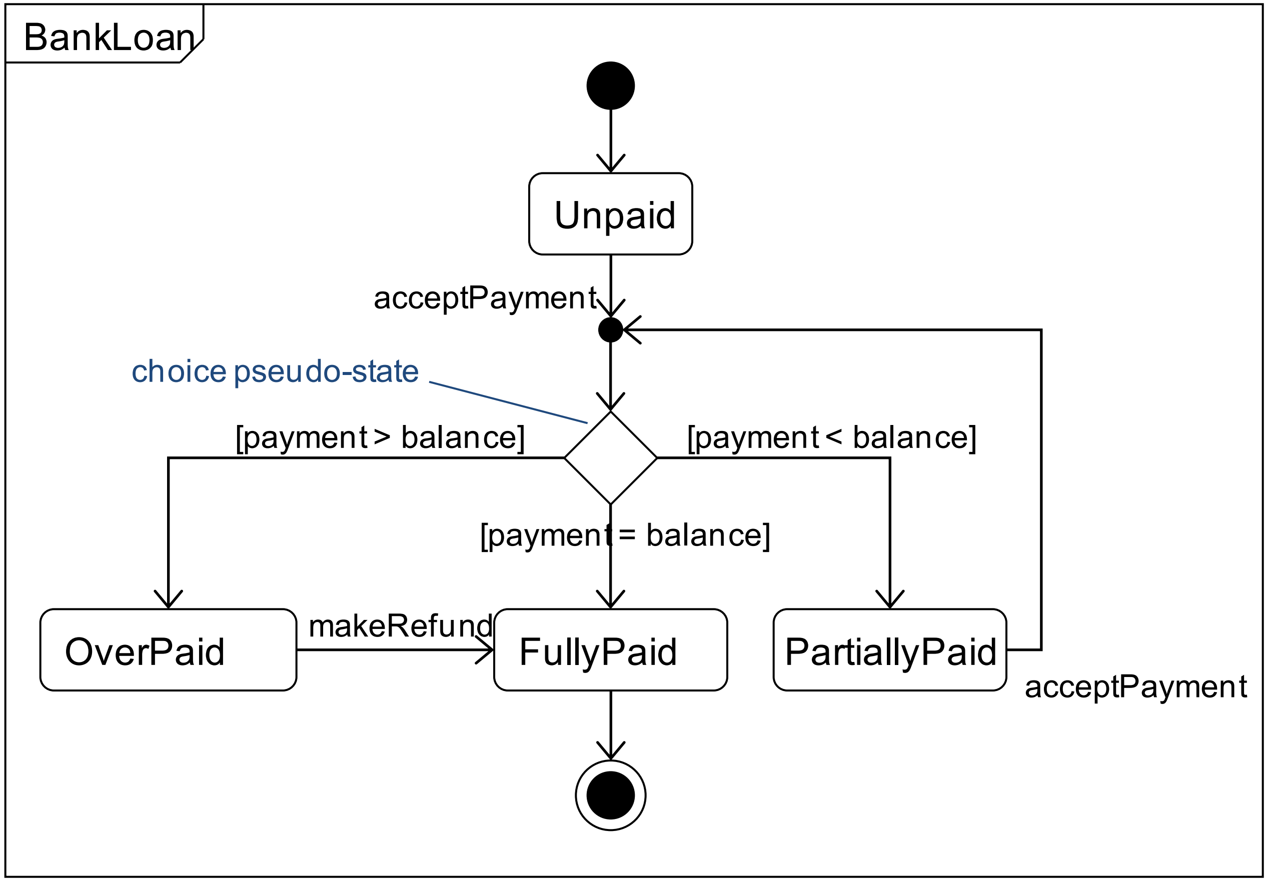


Branching – the choice pseudo state

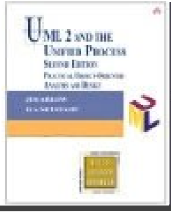


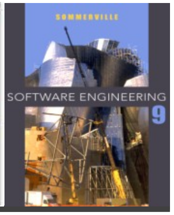


- The choice pseudo state directs its single incoming transition to one of its outgoing transitions
- Each outgoing transition must have a mutually exclusive guard condition

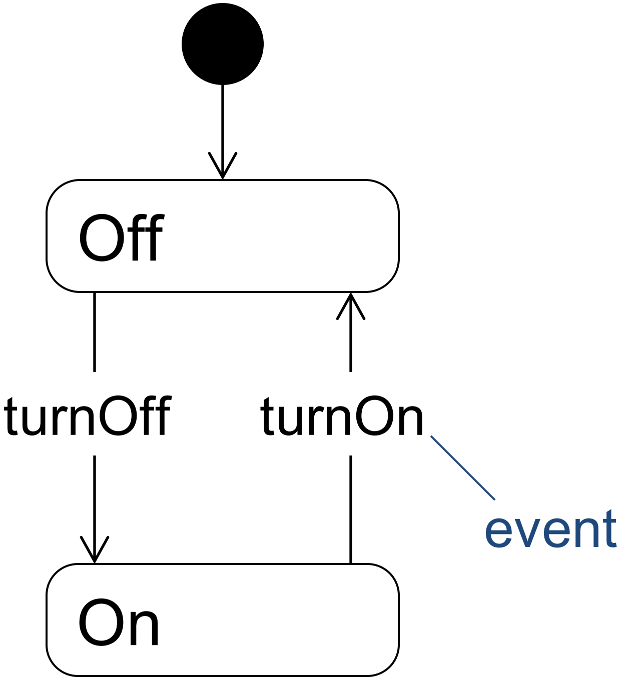


Events





- "The specification of a noteworthy occurrence that has location in time and space"
- Events trigger transitions in state machines
- Events can be shown externally, on transitions, or internally within states (internal transitions)
- There are four types of event:
  - Call event
  - Signal event
  - Change event
  - Time event



Off

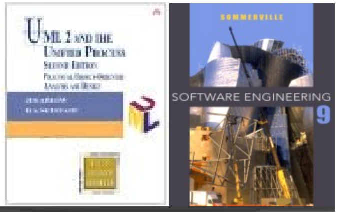
turnOff

turnOn

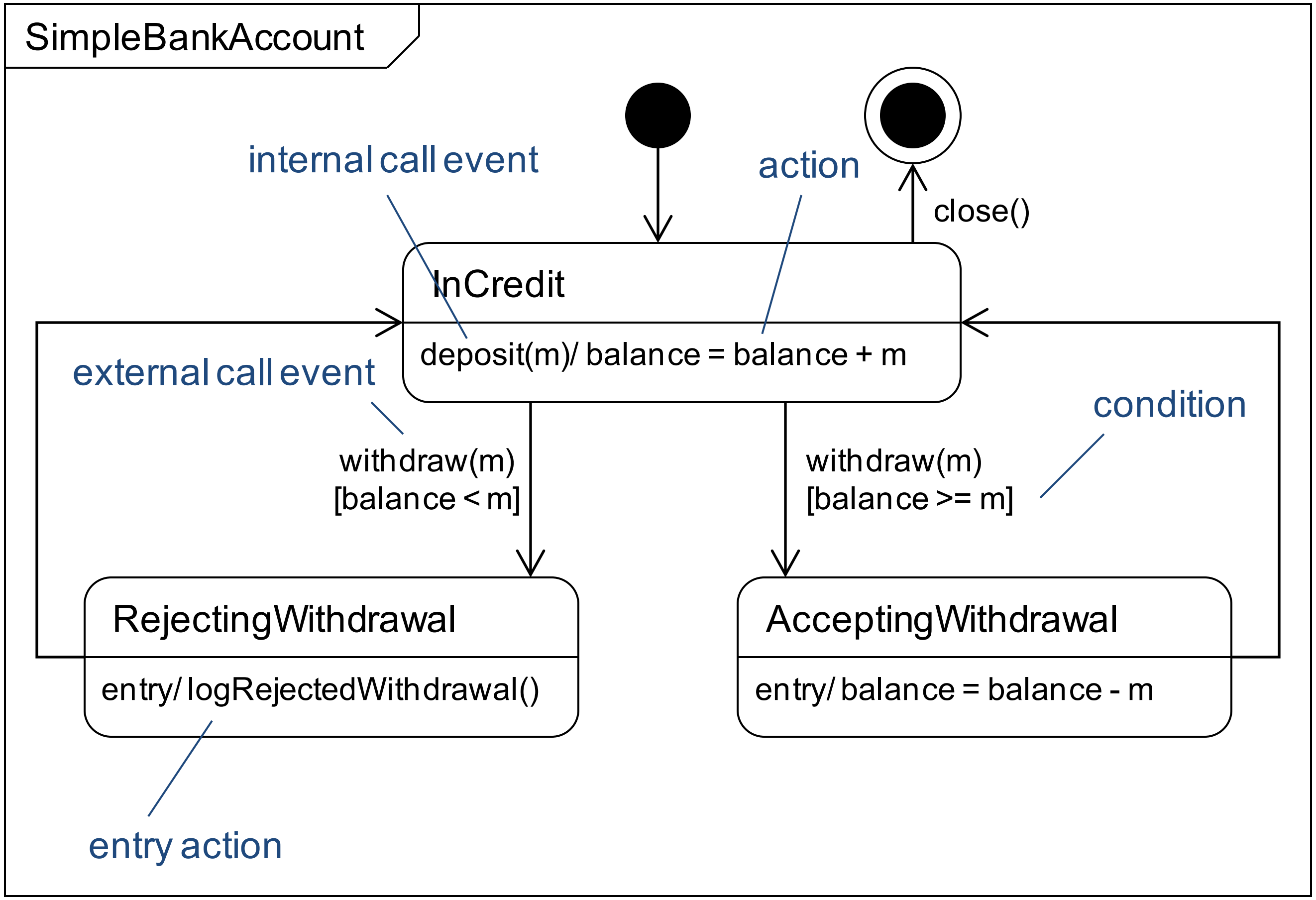
event

On

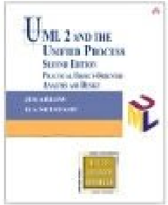
Call event



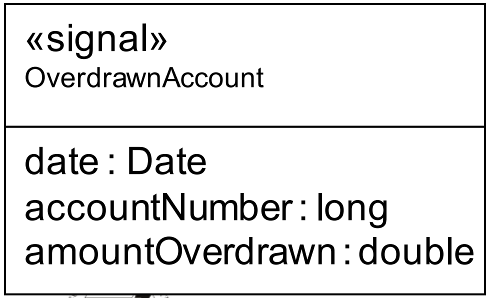
- A call for an operation execution
- The event should have the same signature as an operation of the context class
- A sequence of actions may be specified for a call event - they may use attributes and operations of the context class
- The return value must match the return type of the operation

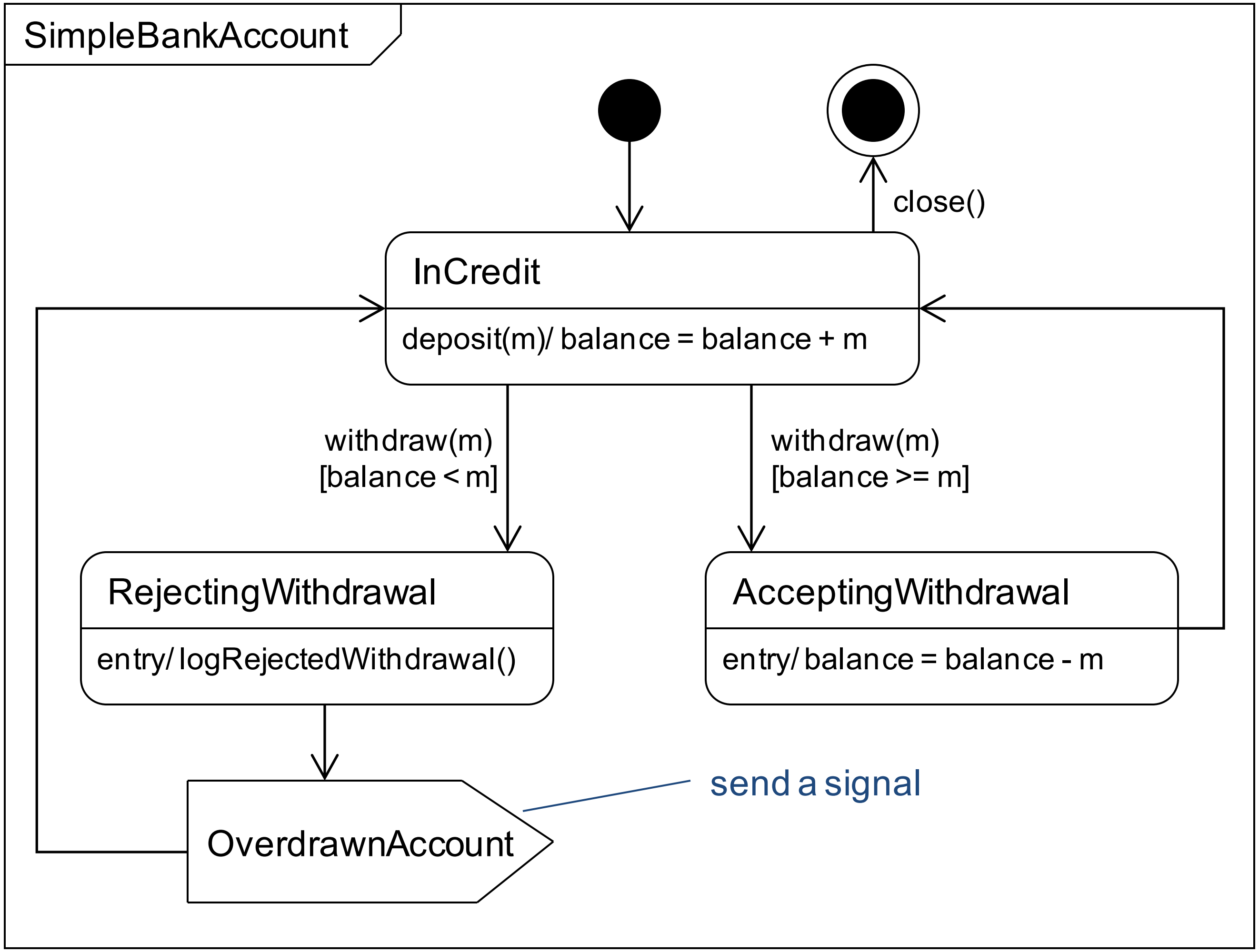


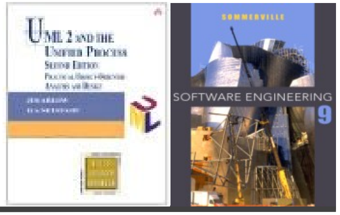
Signal events



- A signal is a package of information that is sent asynchronously between objects
  - the attributes carry the information
  - no operations

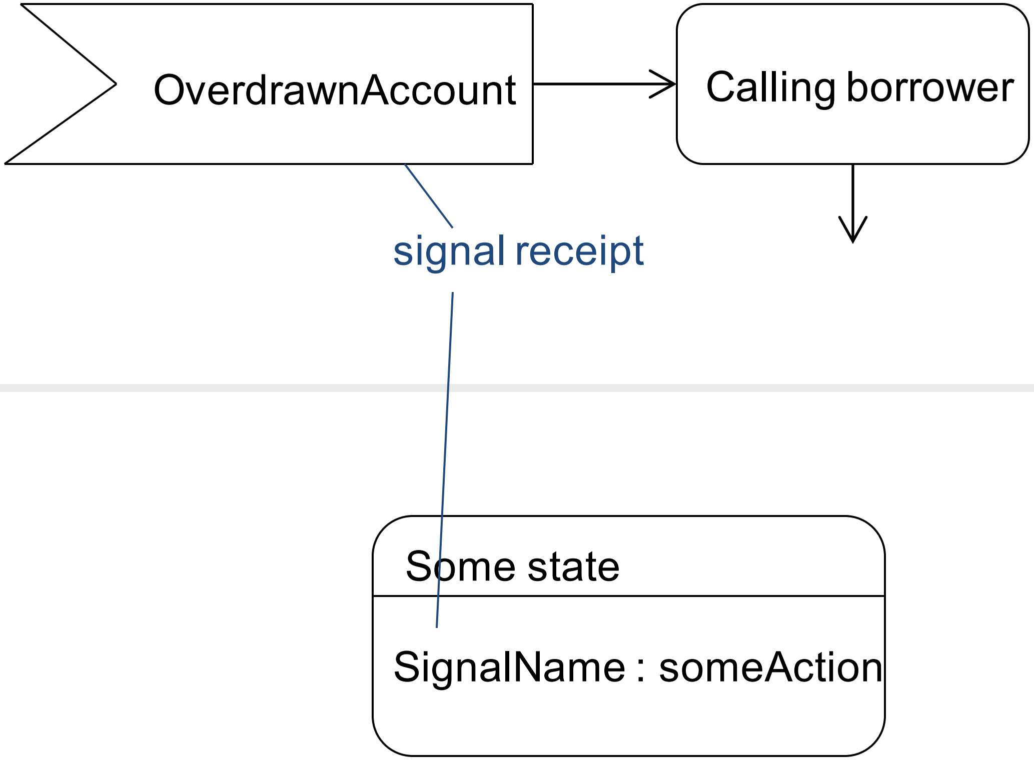






Receiving a signal

- ✧ You may show a signal receipt on a transition using a concave pentagon or as an internal transition state using standard notation



OverdrawnAccount

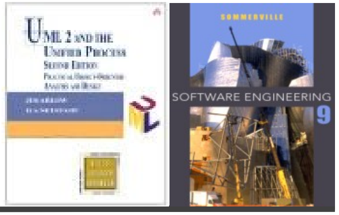
Calling borrower

signal receipt

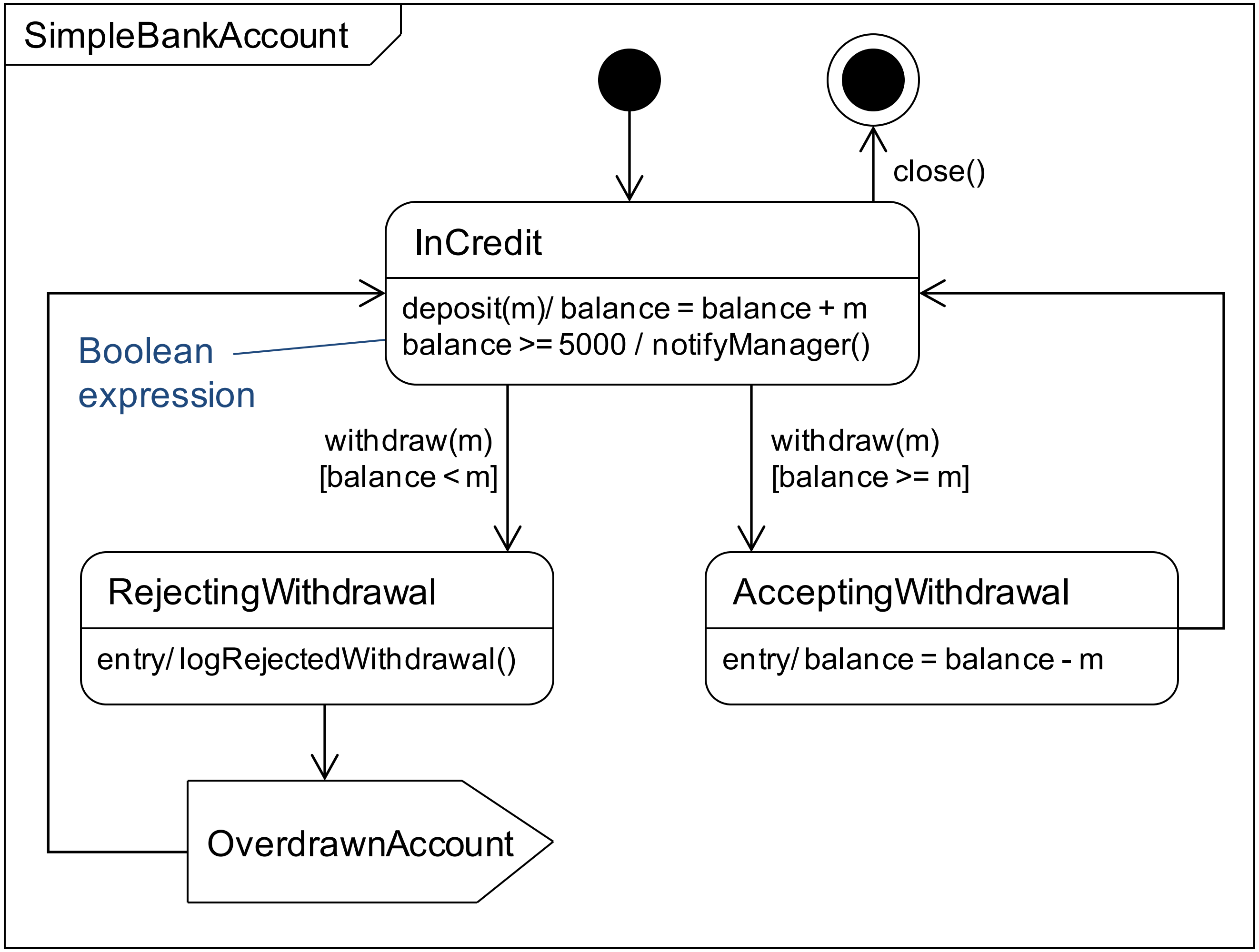
Some state

SignalName : someAction

Change events

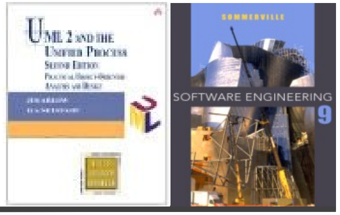


- The action is performed when the Boolean expression transitions from false to true
  - The event is *edge triggered* on a false to true transition
  - The values in the Boolean expression must be constants, globals or attributes of the context class
- A change event implies continually testing the condition whilst in the state

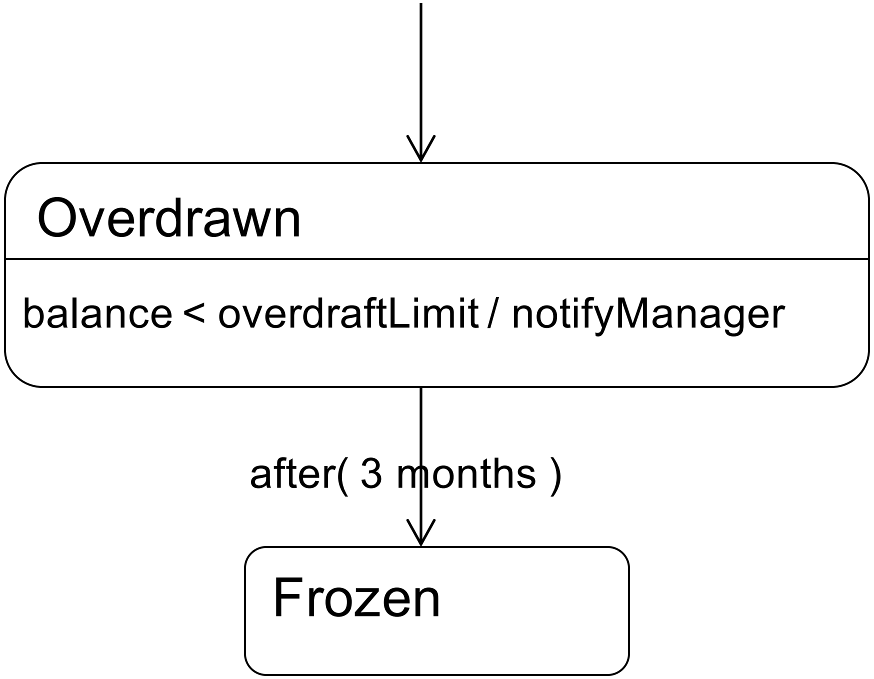


SimpleBankAccount
close()
InCredit
deposit(m)/ balance = balance + m
balance >= 5000 / notifyManager()
Boolean expression
withdraw(m) [balance < m]
withdraw(m) [balance >= m]
RejectingWithdrawal
entry/ logRejectedWithdrawal()
AcceptingWithdrawal
entry/ balance = balance - m
OverdrawnAccount

Time events



- ✧ Time events occur when a time expression becomes true
- ✧ There are two keywords, after and when
- ✧ Elapsed time:
  - after(3 months)
- ✧ Absolute time:
  - when(date =20/3/2000)



Overdrawn

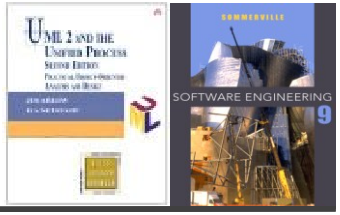
balance < overdraftLimit / notifyManager

after(3 months)

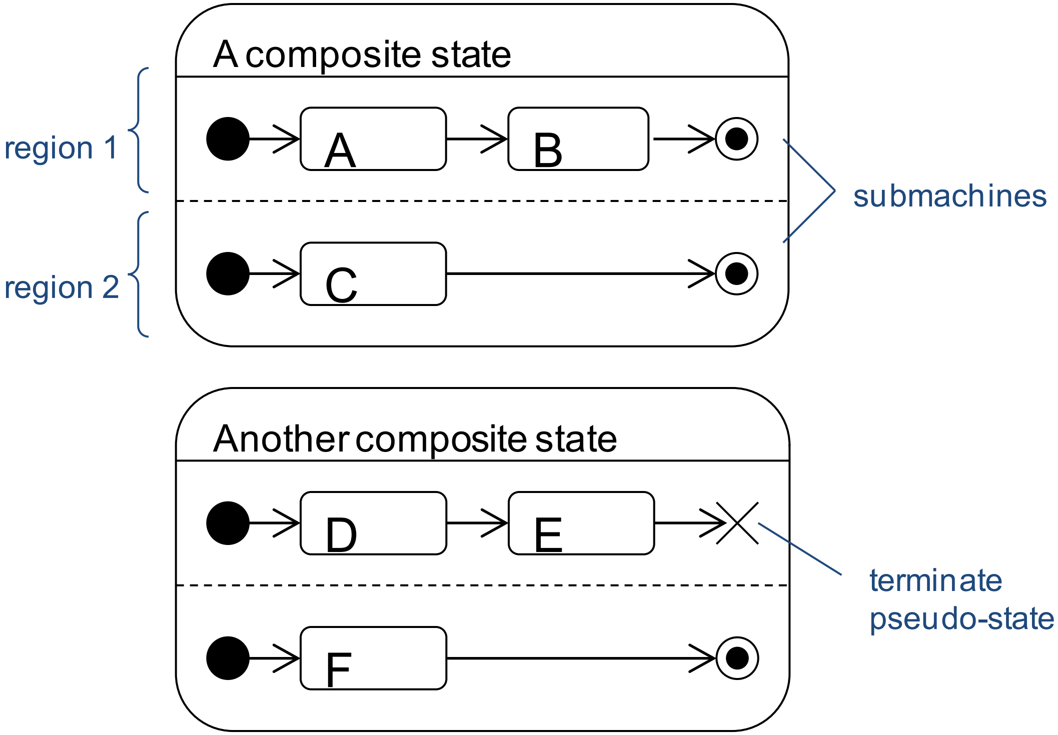
Frozen

Context: CreditAccount class

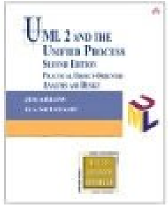
Composite states



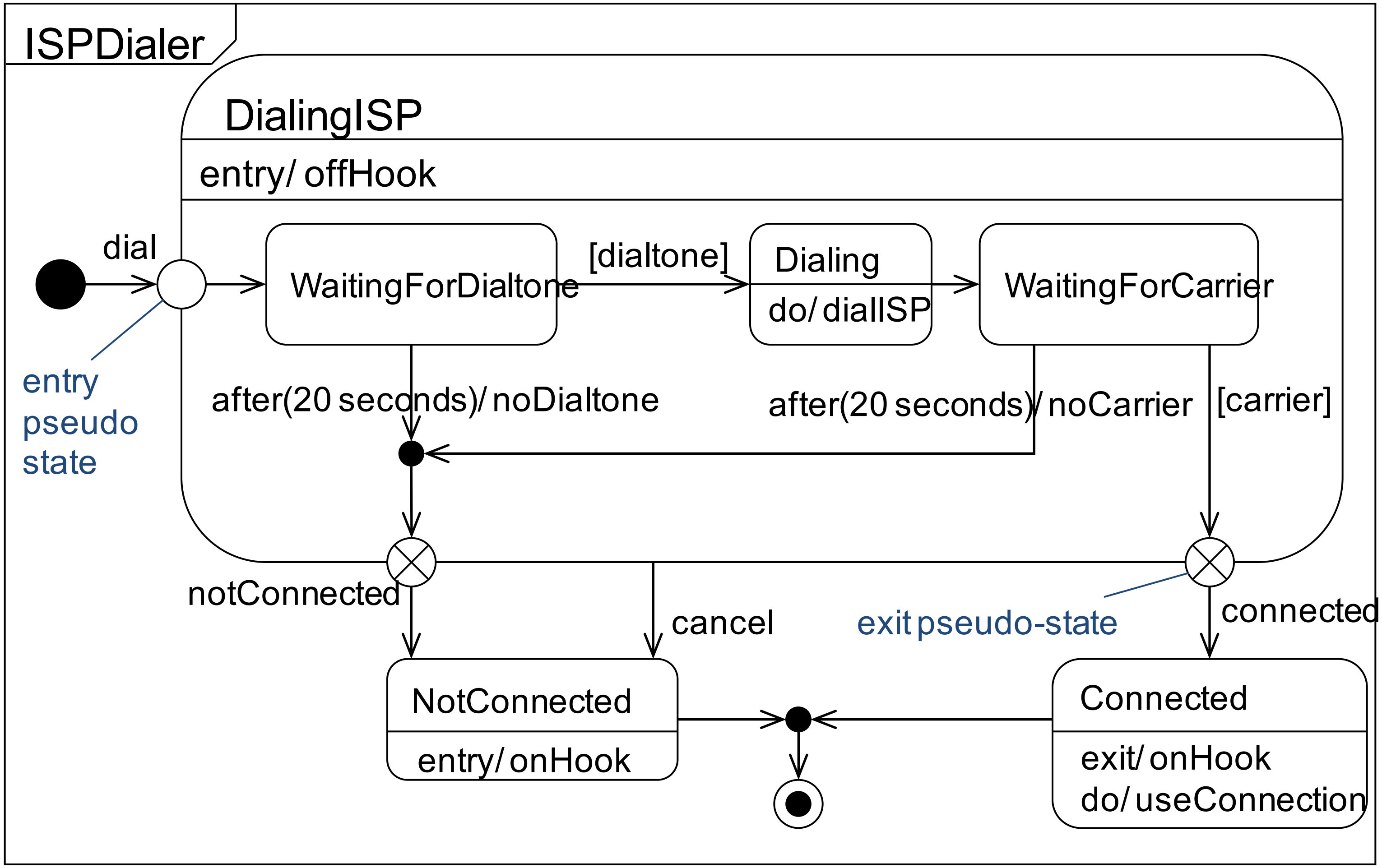
- Have one or more regions that each contain a nested submachine
  - Simple composite state
    - exactly one region
  - Orthogonal composite state
    - two or more regions
- The final state terminates its enclosing region – all other regions continue to execute
- The terminate pseudo-state terminates the whole state machine



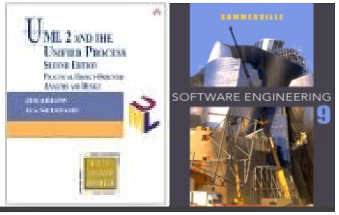
Simple composite states



- Contains a single region
- The nested states inherit the cancel transition from DialingISP

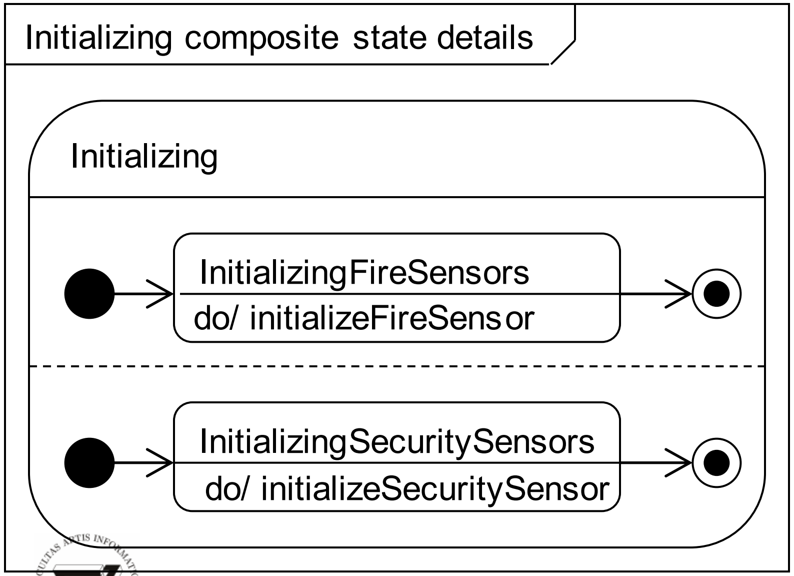


Orthogonal composite states



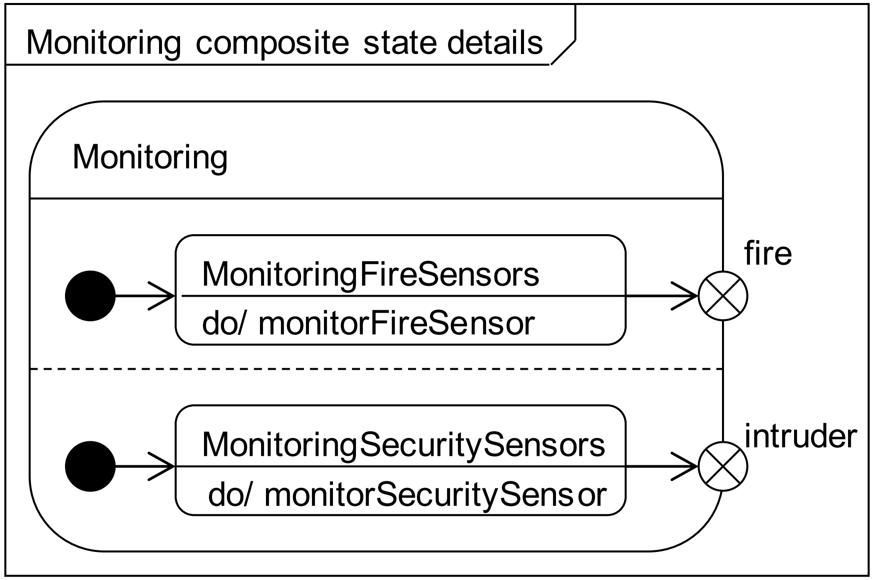
- ✧ Has two or more regions
- ✧ When we enter the superstate, both submachines start executing concurrently - this is an implicit fork

Synchronized exit - exit the superstate when *both* regions have terminated



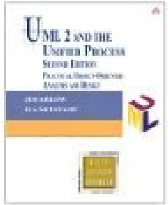
Initializing composite state details
Initializing
InitializingFireSensors
do/ initializeFireSensor
InitializingSecuritySensors
do/ initializeSecuritySensor

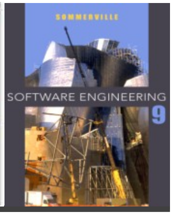
Unsynchronized exit - exit the superstate when *either* region terminates. The other region continues



Monitoring composite state details
Monitoring
MonitoringFireSensors
do/ monitorFireSensor
fire
MonitoringSecuritySensors
do/ monitorSecuritySensor
intruder

Key points





- ✧ Behavioral state machines
- ✧ Protocol state machines
- ✧ States
  - Actions, exit and entry actions, activities
- ✧ Transitions
  - Guard conditions, actions
- ✧ Events
  - Call, signal, change and time
- ✧ Composite states
  - Simple and orthogonal composite states