SWE 530: Software Design Process

Some Design Representations

Dr. H. Birkan YILMAZ
Department of Computer Engineering
Boğaziçi University

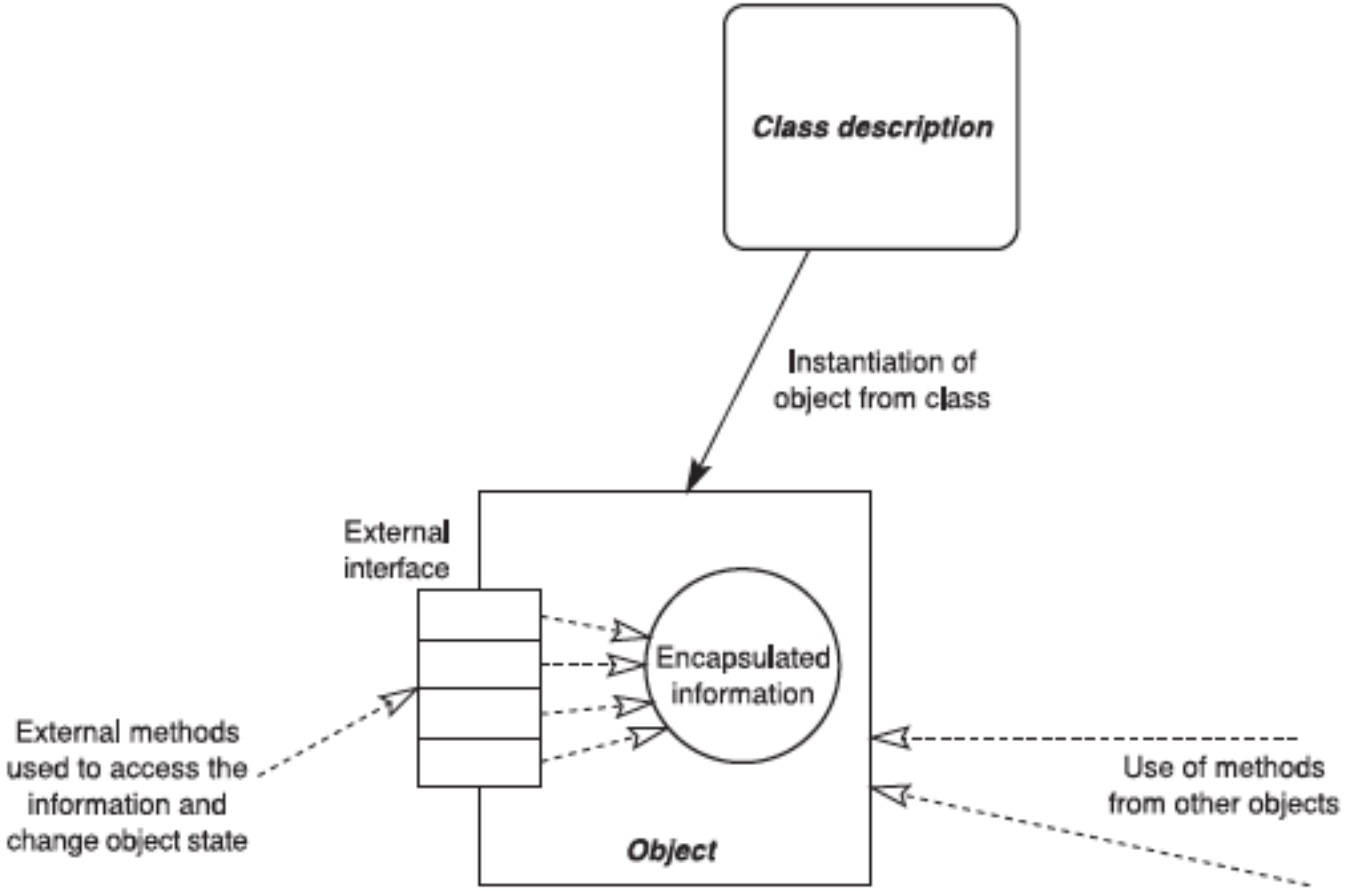
(birkan.yilmaz@bogazici.edu.tr)

Adapted from slides of Dr. Albert Ali Salah & Başak Aydemir

UML Modelling Forms

- Ideas originated from Grady Booch, Ivar Jacobson, and James Rumbaugh
- The rise of OOP
- Object Management Group (OMG) handles UML development since 1996
- Related to the concept of the “object”

What is an Object?



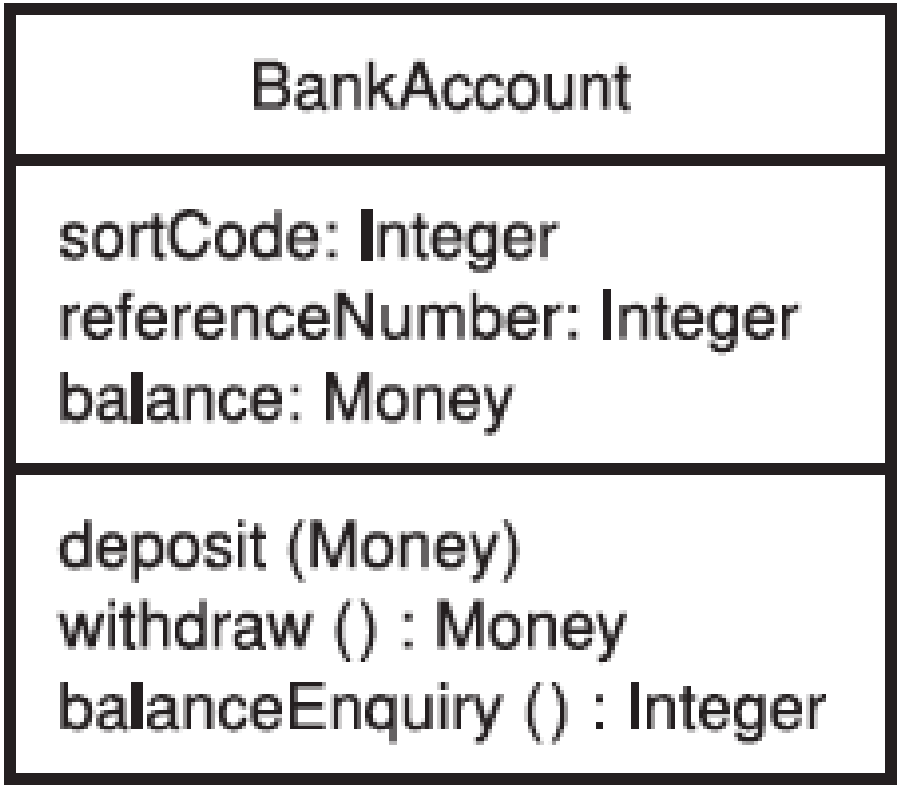
UML Forms

- The UML forms are classified in Rumbaugh et al. (1999) into:
  - structural forms (relating elements of a system);
  - dynamic forms (concerned with describing system behaviour);
  - model management (concerned with the organization of the model itself).
- The first two can be recognized as corresponding to the constructional, functional and behavioural viewpoints

UML Forms

Static Design Diagrams	Dynamic Behavior Diagrams
Class diagram	Use case diagram
Object diagram	Sequence diagram
Component diagram	Collaboration diagram
Deployment diagram	State diagram
	Activity diagram

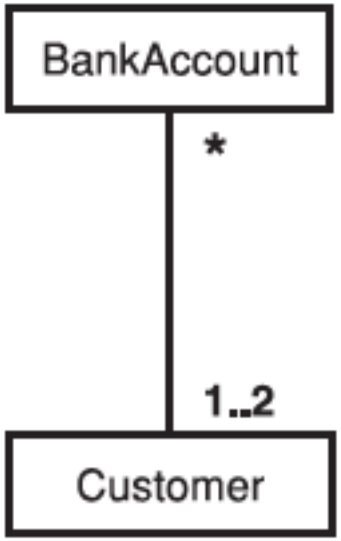
UML: Class Diagram



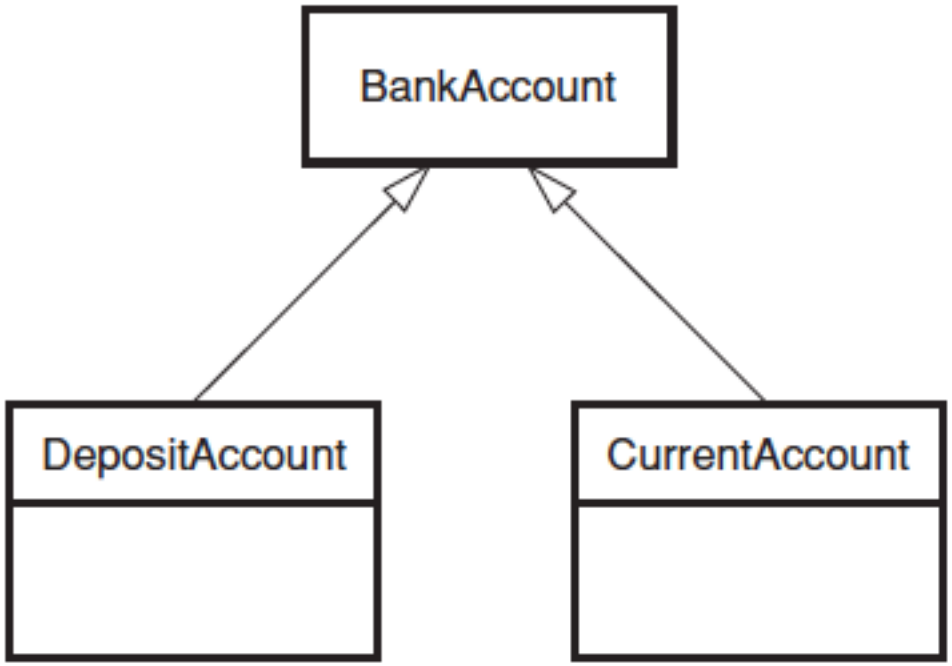
UML: Class Diagram

- Describes classes and their interactions
- Used for a constructional viewpoint
- UML recognizes six forms of relationship (association, dependency, flow, generalization, realization and usage)

UML: Class Diagram



Association



Generalization (via Inheritance)

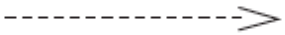
UML: Use Case Diagram

Use case (title indicates the activity covered by this use case)

Actor (not necessarily a human) describing a role in a use case

Indicates **<<communication>>** between actor and use case (only form of link that is employed between these two elements)

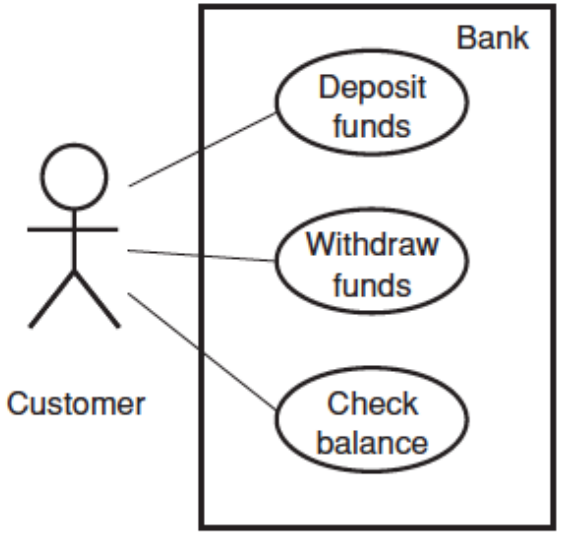
<<generalization>> that links the varieties of a use case with the more general form of these



<<include>> or **<<extend>>** relationships between use cases where one use case makes use of other use cases, labelled to indicate the form that a particular relationship has

UML: Use Case Diagram

(a) The use case diagram



(b) A simple use case specification

USE CASE: Withdraw funds

ACTORS: Customer

PRE-CONDITIONS:
User is a customer of the bank

BODY:

1. User enters card in machine and establishes their identity using their PIN.
2. User enters a request for funds and specifies the amount required.
3. The system checks to ensure that this will not exceed the customer's credit limit.
4. If request is valid, and cash is available in the machine, system directs the machine to deliver the cash and debits customer's account, else request is refused.
5. System directs machine to return card.

POST-CONDITIONS:
User's account is debited by the amount withdrawn

UML: Use Case Diagram

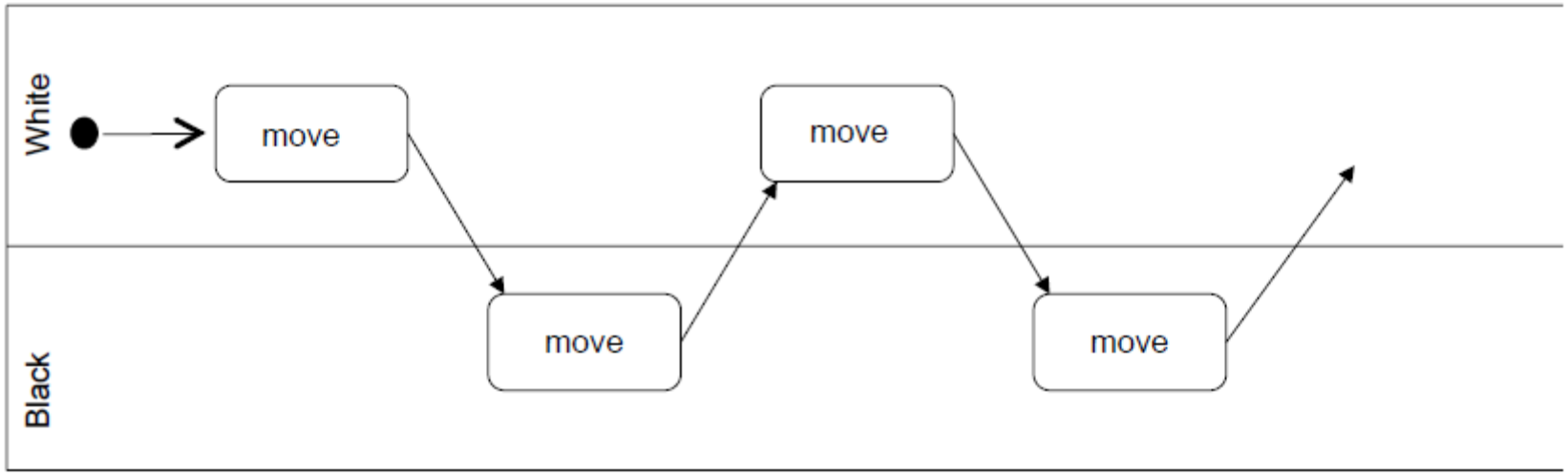
- A **use case** describes a set of possible interactions between a system and other **actors**
- A **scenario** describes a particular sequence of these interactions
- Uses behavioral and functional viewpoints
- “Use case modelling is perhaps one of the most disappointing elements of the UML.” (Budgen, 2003)

UML: Activity Diagram

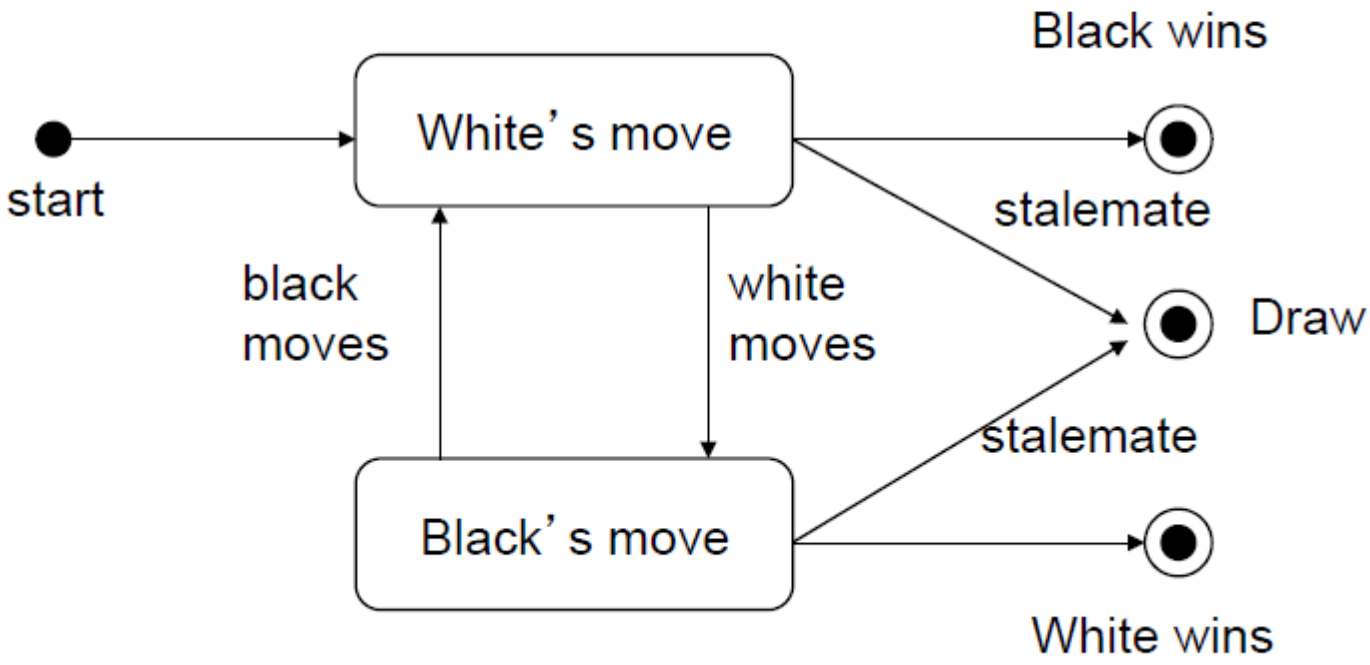
- Models computations and workflows
- Good for synchronization and coordination
- Mainly behavioural (plus some functional) viewpoint
- Notation includes:
  - activity (box with round sides)
  - transition (unlabelled arrow)
  - synchronization bar
  - decision diamond

- activity (box with round sides)
  - Non-interruptable action of objects
- transition (unlabelled arrow)
  - Transition from one action state to another
- synchronization bar
  - Join node
- decision diamond
  - Requires a decision prior to moving on to the next activity

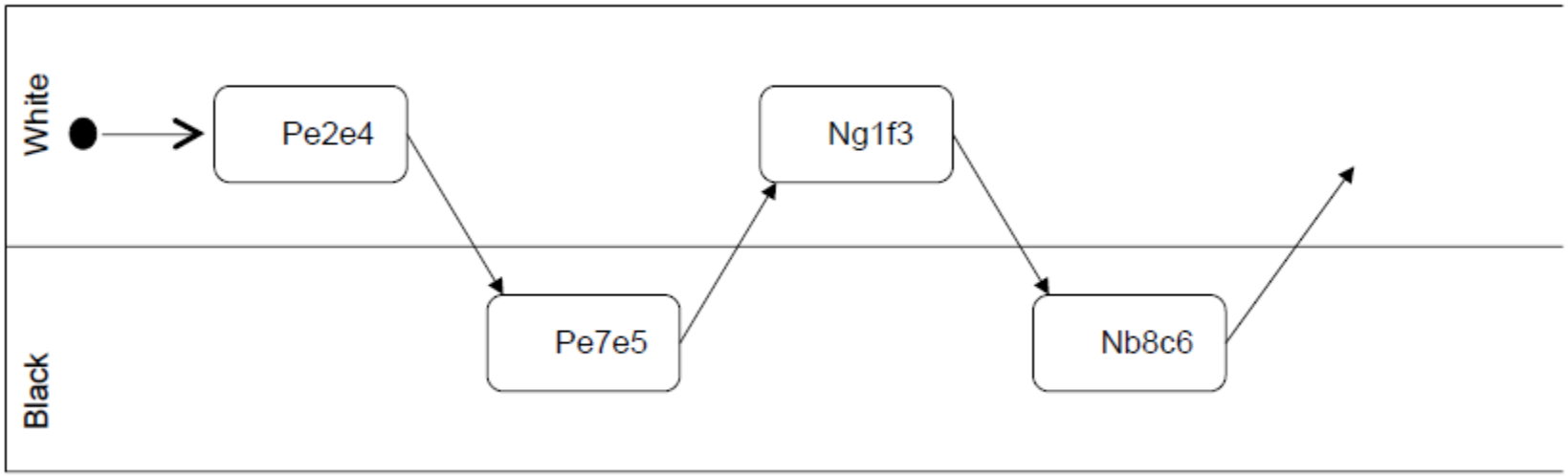
Activity diagram: chess game



State machine: chess game



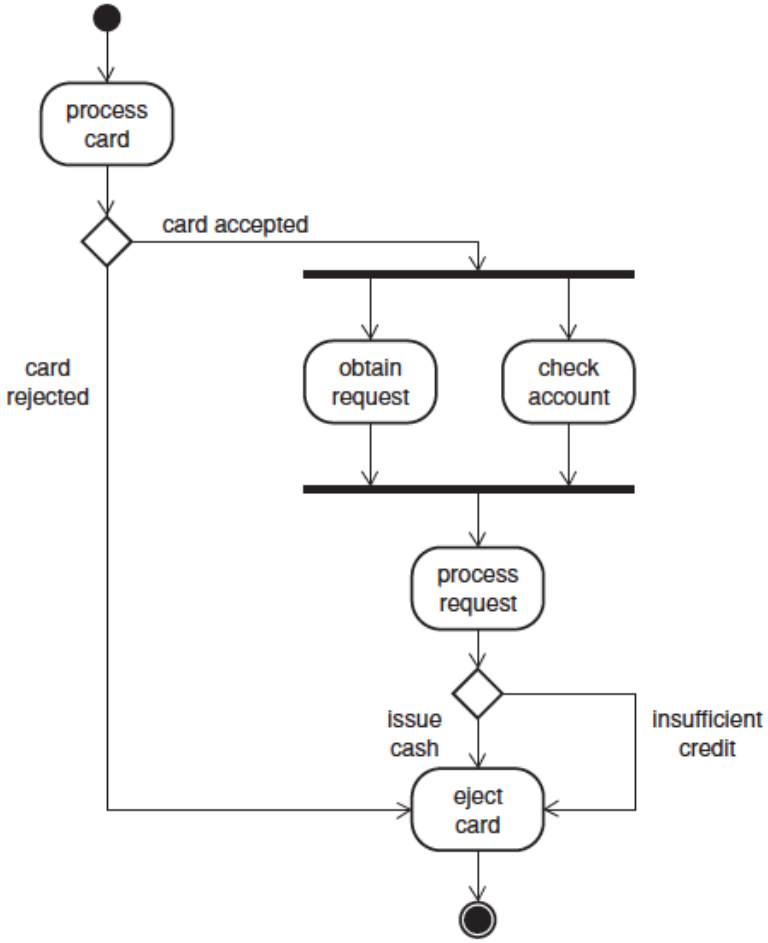

e2e4
e7e5
g1f3
b8c6


White
Black
Pe2e4
Pe7e5
Ng1f3
Nb8c6

What an AD Does Not Show?

- Objects
- States (possible in some cases with indirect ways)
- Messages
- Data passed between steps
- User Interface

UML: Activity Diagram



Selection of black box forms

Representation form	Viewpoints	Design characteristics
Data-Flow Diagram	Functional	Information flow, dependency of operations on other operations, relation with data stores
Entity–Relationship Diagram	Data modelling	Static relationships between design entities
State Transition Diagram	Behavioural	State-machine model of an entity
Statechart	Behavioural	System-wide state model, including parallelism (orthogonality), hierarchy and abstraction
Structure Diagram (Jackson)	Functional, data modelling, behavioural	Form of sequencing adopted (operations, data, actions)
UML: Class Diagram	Constructional	Interactions between classes and objects
UML: Use Case Diagram	Behavioural and functional	Interactions between a system and other 'actors
UML: Activity Diagram	Behavioural and functional	Synchronization and coordination of system activities

Selection of white box forms

Representation form	Viewpoints	Design characteristics
Structure Chart	Functional and constructional	Invocation hierarchy between subprograms, decomposition into subprogram units
Class and Object Diagrams	Constructional	*Uses* relationships between elements, interfaces and dependencies
Sequence Diagrams	Behavioural	Message-passing sequences, interaction protocols
Pseudocode	Functional	Algorithm form

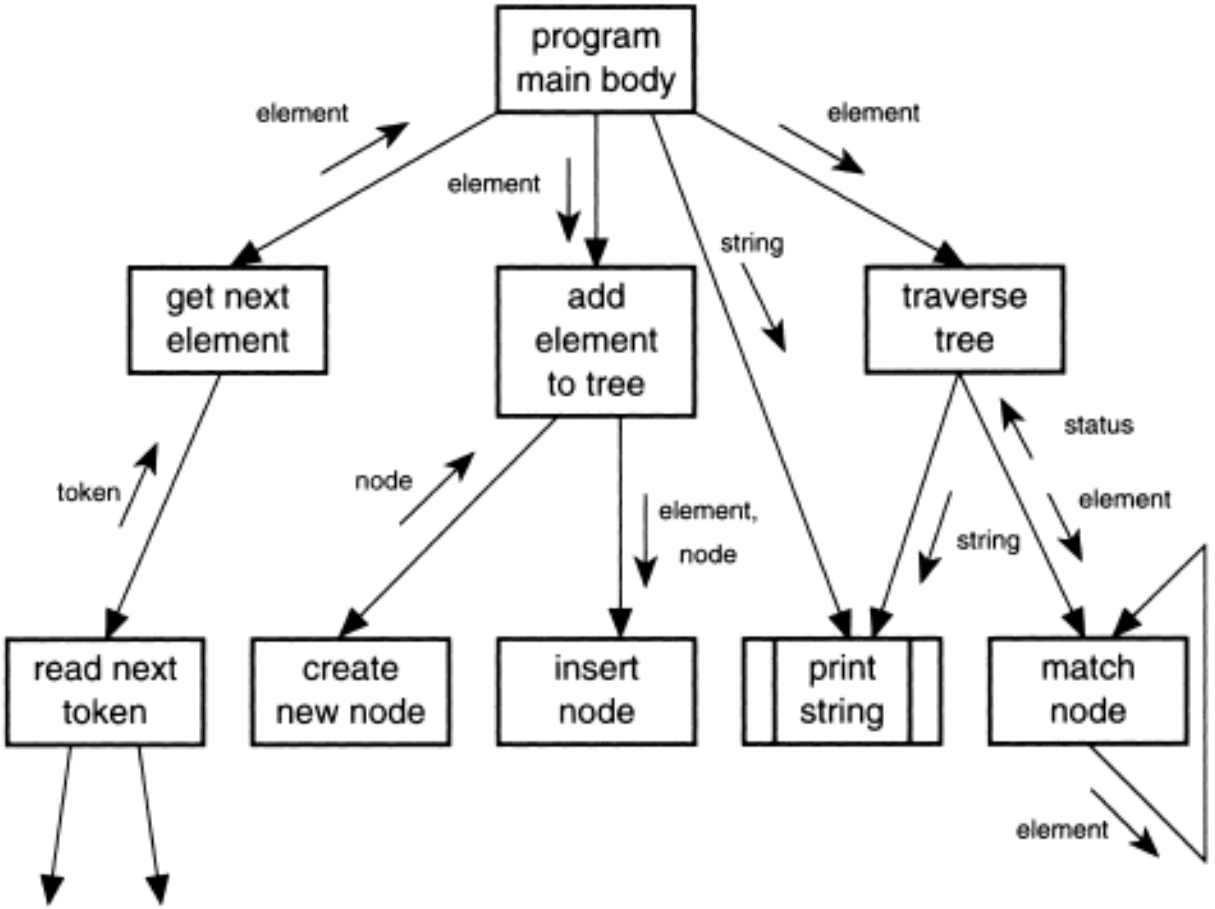
White box forms

- A white box notation is one that is concerned with the **detailed realization** of the design elements.
- Related to constructional and functional viewpoints
- Not a blueprint!
- Structure Chart, Class and Object Diagrams, Sequence Diagram, Pseudocode.

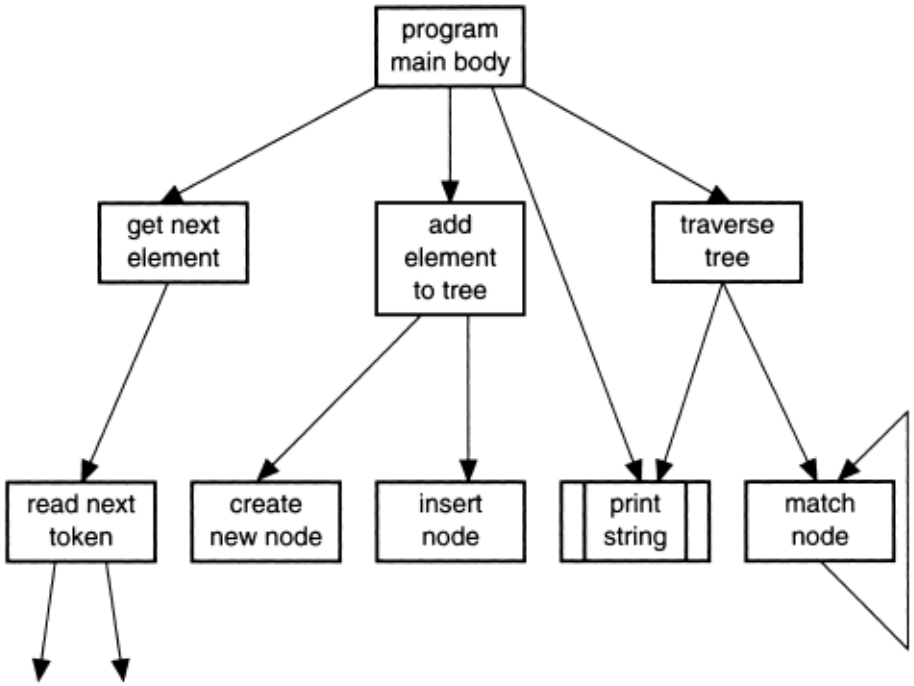
Structure Chart

- A visual index of the hierarchy of procedures within the program
- Origins go to IBM OS/360 in the 1970's
- Uses:
  - boxes (procedures and subprograms)
  - arcs (invocations)
  - side-arrows (information flow, parameters)
- Frequency and sequence of calls not recorded
- Good for programmer, and the maintainer

Structure Chart



Structure Chart

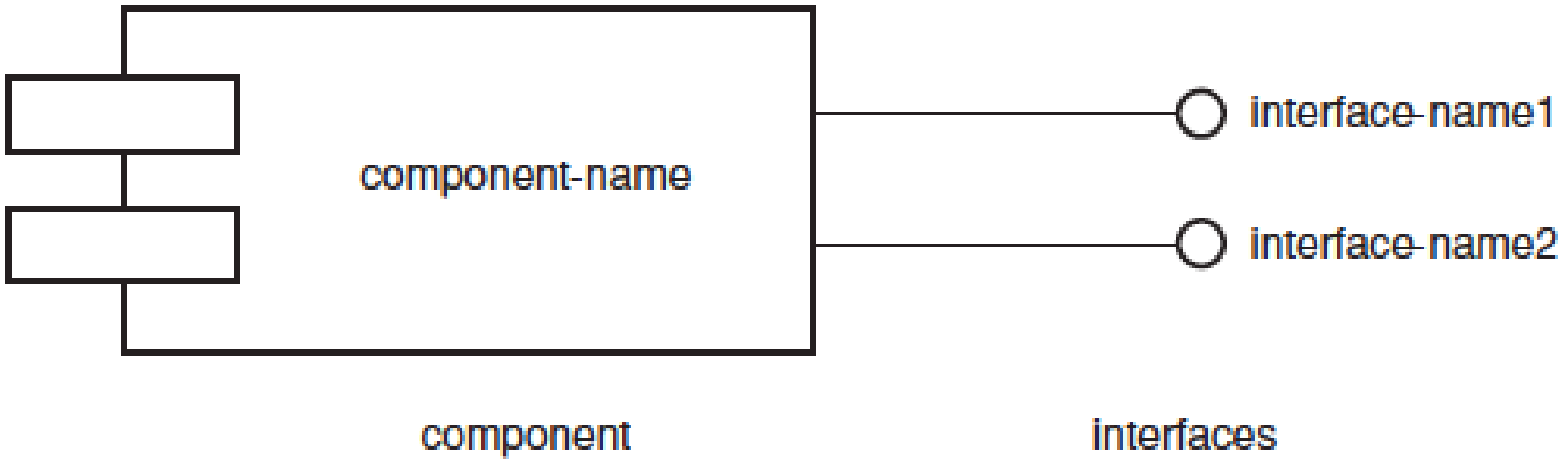


subprogram	in	out
AddElementToTree	e:element	s:status
PrintString	s:string	
MatchNode	n:node	m:match status

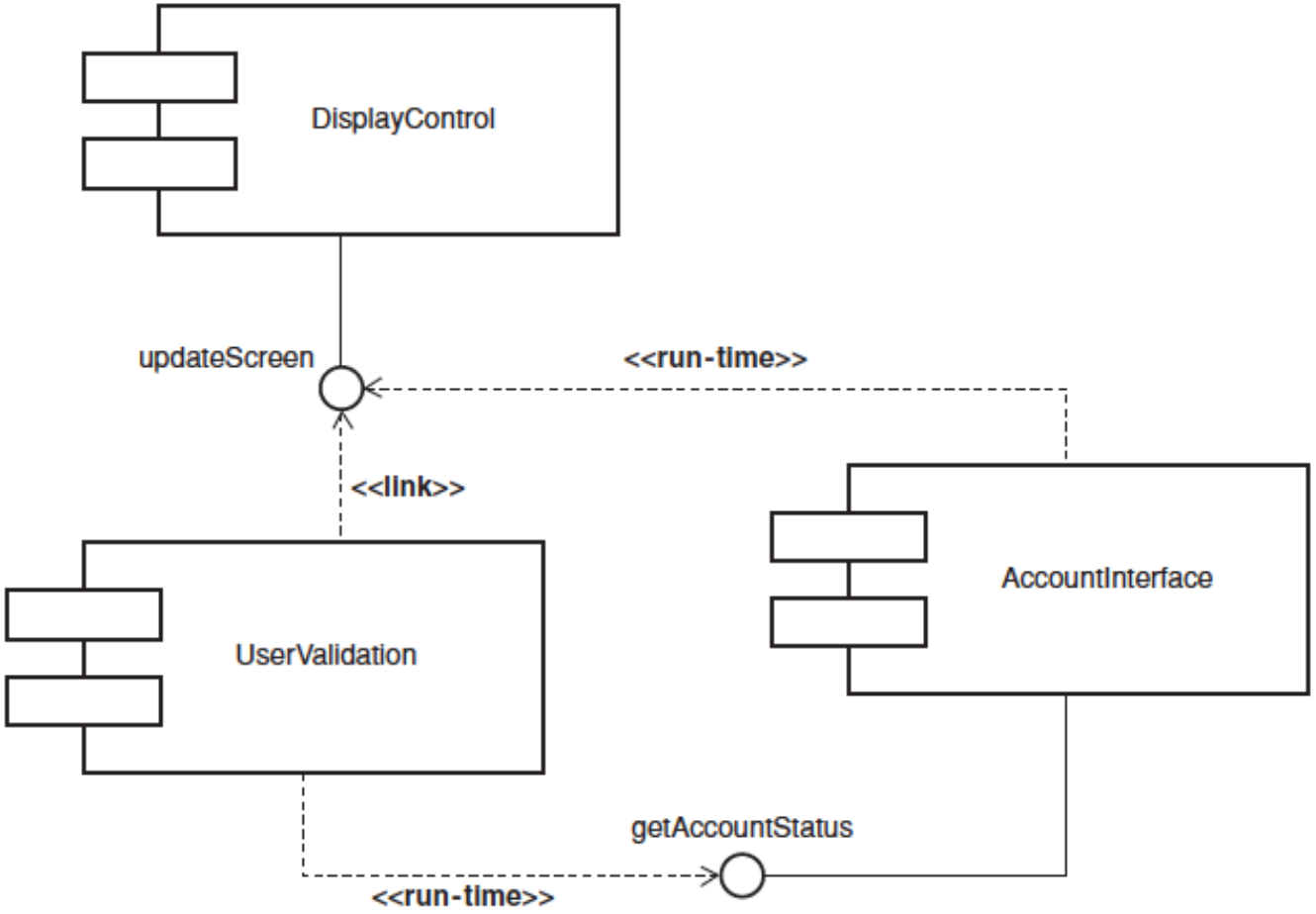
UML: Component Diagram

- UML class and object diagrams can also be used for detailed descriptions
- A UML component is ‘a physical unit of implementation with well-defined interfaces that is intended to be used as a replaceable part of a system’ (Rumbaugh et al., 1999).
- ‘each component embodies the implementation of certain classes from the system design’ (ibid.)

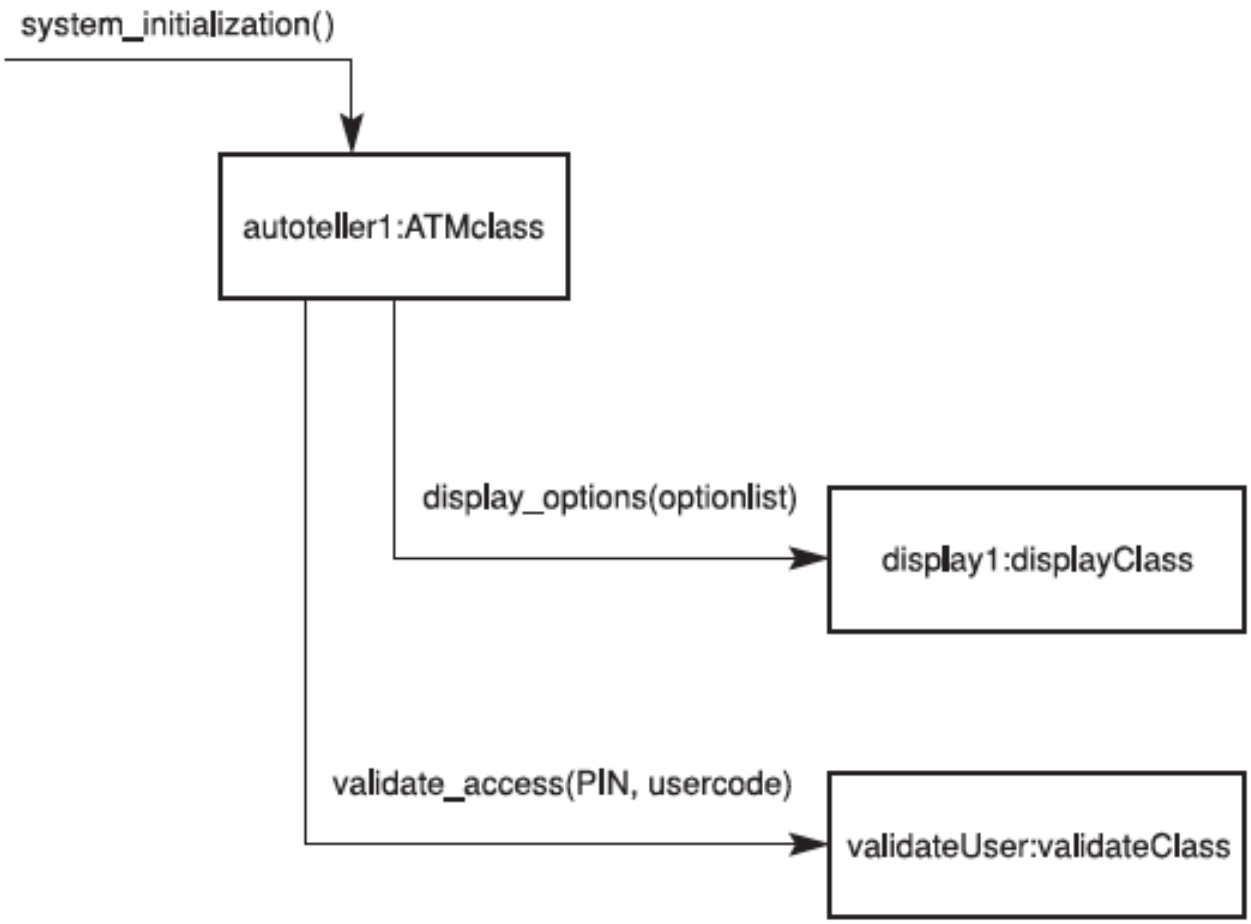
UML: Component Diagram



UML: Component Diagram



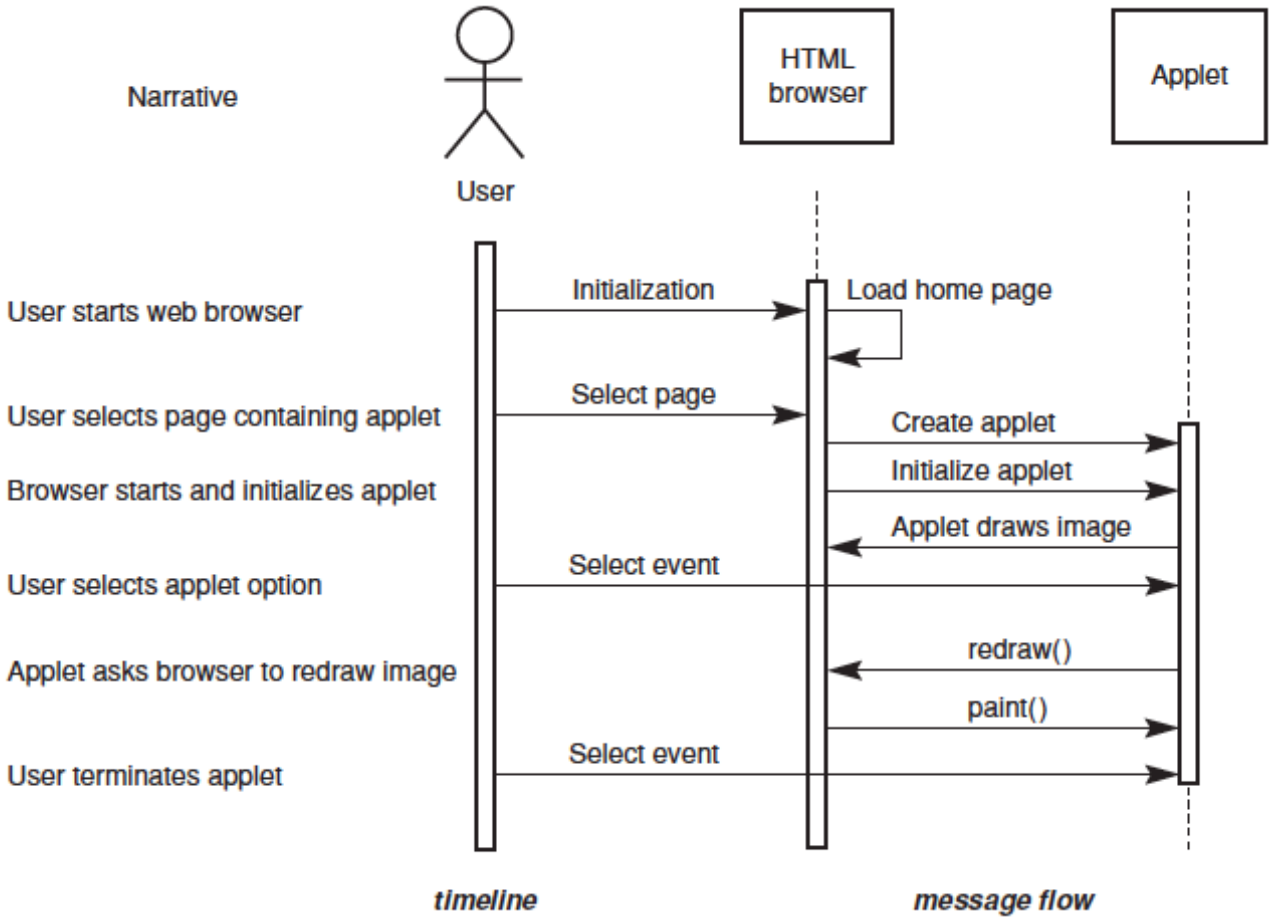
Object interaction graph (Fusion)



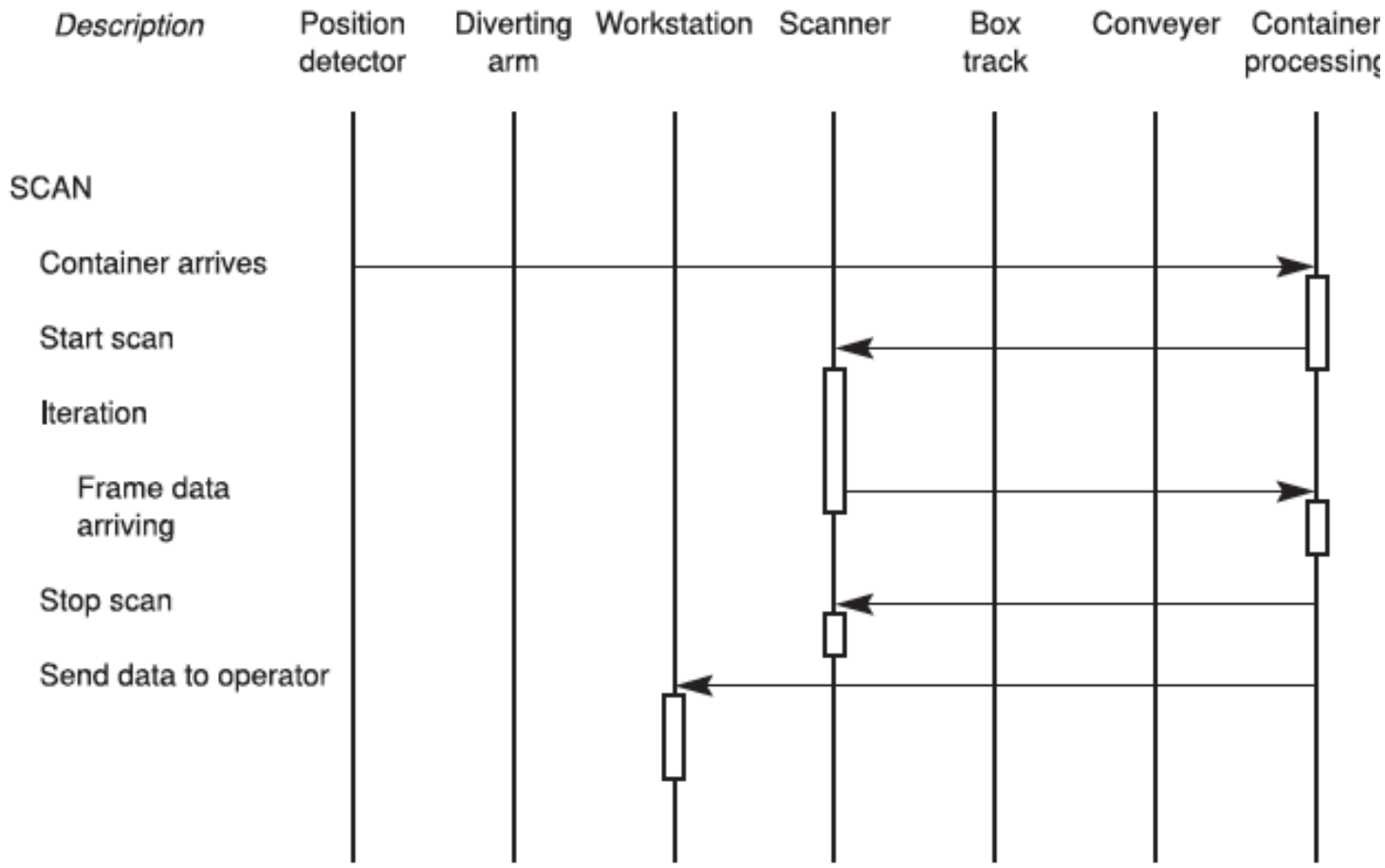
UML: Sequence Diagram

- Takes a behavioural viewpoint of the system
- Intrinsically non-hierarchical
- Some functional information is also provided
- Can be used to describe
  - Use cases
  - Scenarios
  - Protocols

UML: Sequence Diagram



UML: Sequence Diagram



Pseudocode

- Describes a solution in terms of sequences of operations
- Has a functional viewpoint

```
boil water;
pour some water into teapot;
empty teapot;
REPEAT
    place spoonful of tea in pot
UNTIL enough tea for no. of drinkers;
REPEAT
    pour water into pot
UNTIL enough water for no. of drinkers;
```

Pseudocode

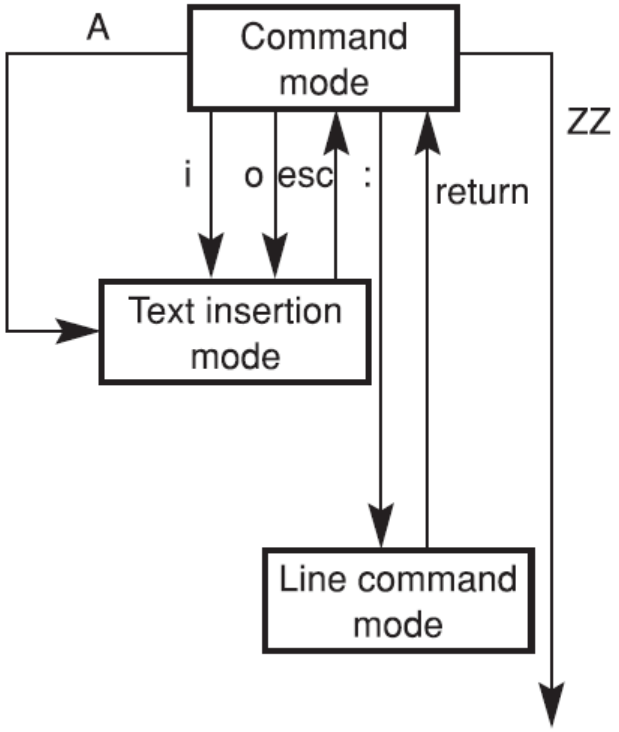
- Rules of thumb:
  - Use indentation
  - Use language keywords (emphasized)
  - Bracket executable blocks (e.g. use LOOP and ENDLOOP)
  - Use clear variable and constant names
  - Should fit on a page, ideally even smaller
- Can complement Structure Charts
- Can be derived from Structure Diagrams

Developing a diagram

- Many notations can be conveniently reduced to a **tabular form** to assist in checking them for completeness and consistency.
- Conversely, starting from a tabular form can assist with generating a diagram.

State Transition Table

(a) The STD form



(b) The STT

	A	i	o	esc	:	return	ZZ
Command mode (C)	T	T	T		L		Exit
Text insertion mode (T)				C			
Line command mode (L)						C	

State Transition Table

	stack	cleared to land	abort landing	touch down	take off	abort take off	park	cleared for take off
in flight	stacked	landing approach						
stacked		landing approach						
landing approach			in flight	on runway				
on runway					in flight	on ground	on ground	
on ground								on runway

Developing a diagram

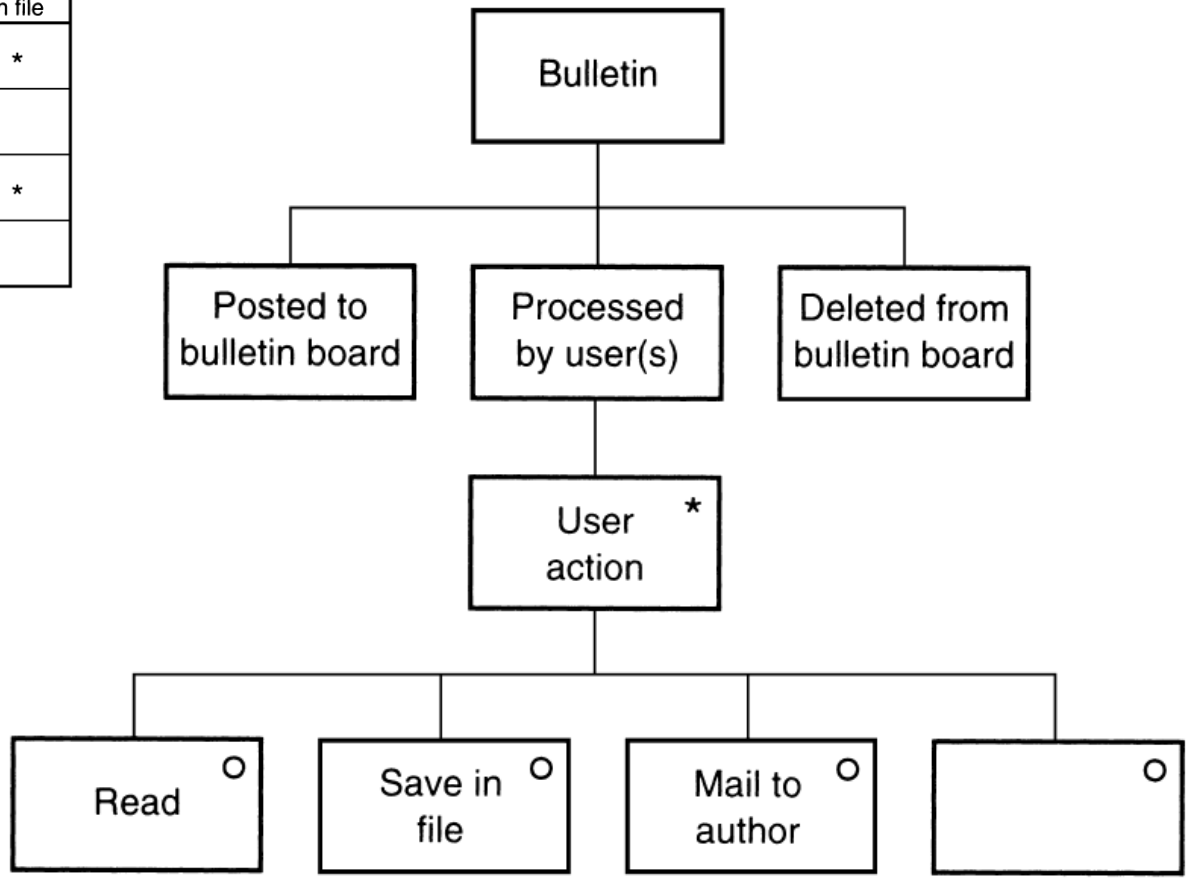
- Example: Entity Life-History Diagram (ELHD)
  - List the entities
  - Identify and list the events, considering:
    - external events (arises outside the system)
    - temporal events (at a particular point in time)
    - internal events (some condition is satisfied)
  - Create Entity Life-History matrix*
    - identify links between events and entities
  - Draw the diagram

Entity Life-History Matrix

ENTITIES	EVENTS					
	Post bulletin	List b-b contents	Read bulletin	Delete expired bulletin	Mail bulletin author	Save bulletin in file
Bulletin	*		*		*	*
Bulletin board	*	*				
User	*	*	*		*	*
Daemon				*		

Entity Life-History Diagram (SSADM)

ENTITIES	EVENTS					
	Post bulletin	List b-b contents	Read bulletin	Delete expired bulletin	Mail bulletin author	Save bulletin in file
Bulletin	*		*		*	*
Bulletin board	*	*				
User	*	*	*		*	*
Daemon				*		



Summary

- A wide range of notations can be used to support and document the development of a design model
- Examples illustrate the forms used for both white box and black box modelling across the set of viewpoints (functional, behavioural, data modelling and constructional)
- The choice of any particular notation depends upon many factors, including the problem domain, architectural form of the eventual system, and design practices being used.

Questions?