## Design notations

Dynamic

- Data flow diagrams (DFDs).
- State transition diagrams (STDs).
- Statecharts.
- Structure diagrams.

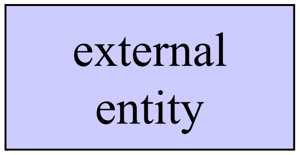
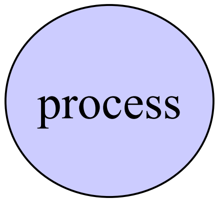
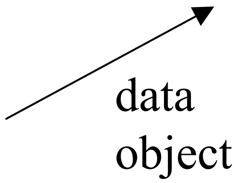
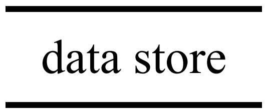
Static

- Entity Relationship Diagrams (ERDs).
- Class diagrams.
- Structure charts.
- Object diagrams.

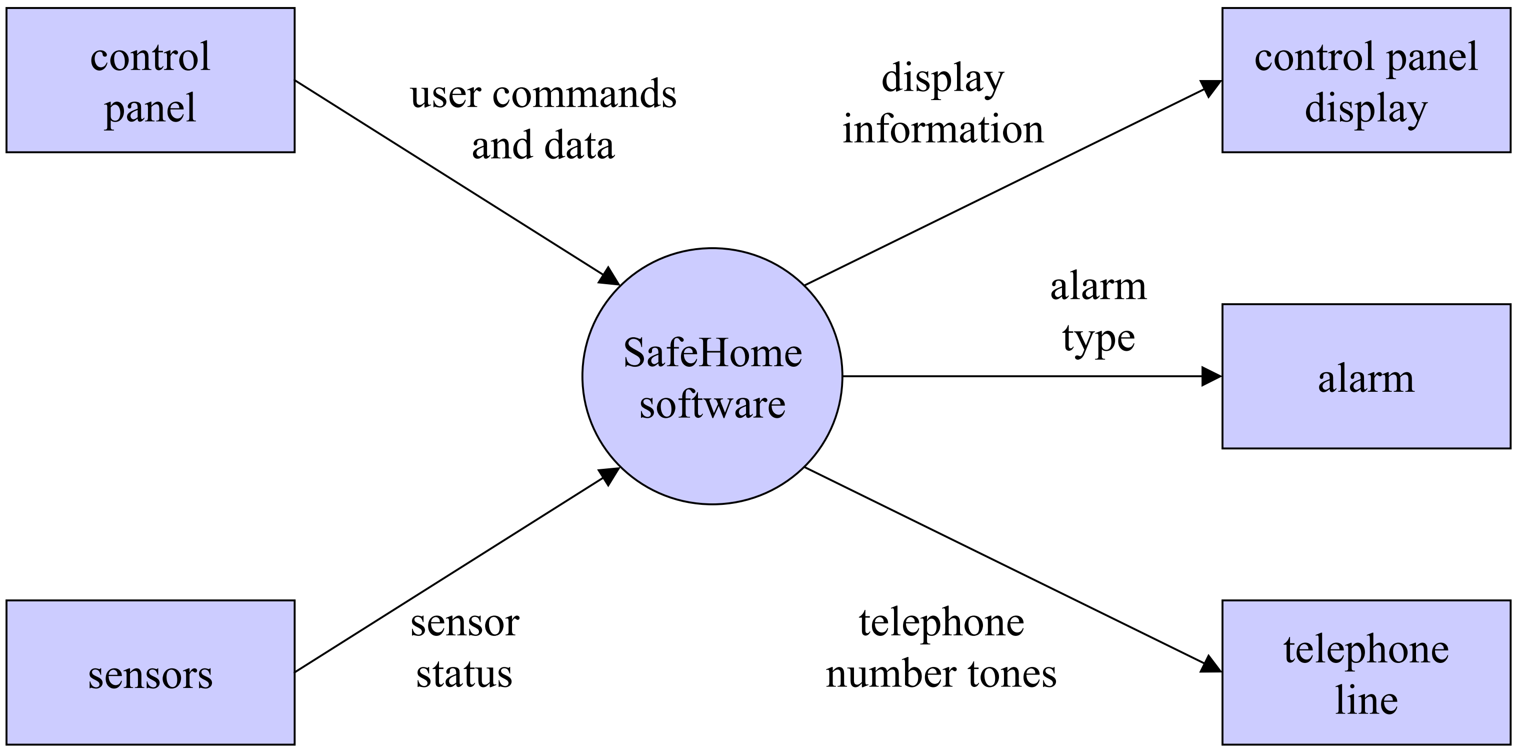
## *Data Flow Diagrams* (DFDs)

- A notation developed in conjunction with *structured systems analysis/structured design* (SSA/SD).
- Used primarily for pipe-and-filter styles of architecture.
- Graph–based diagrammatic notation.
- There are extensions for real-time systems that distinguish *control flow* from data flow.

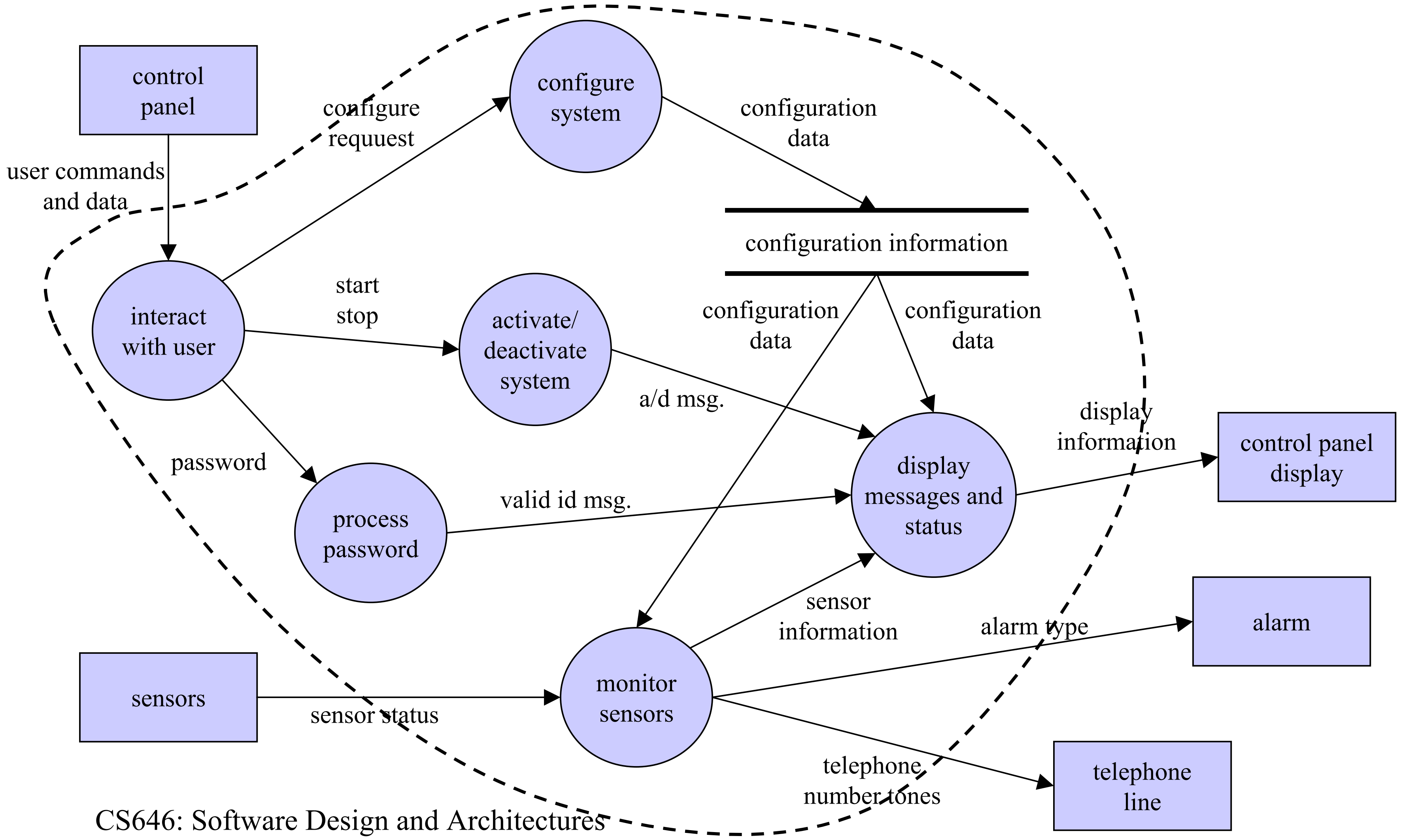
# DFDs: Diagrammatic elements

| | |
|---|---|
|  | A producer or consumer of information that resides outside the bounds of the system to be modeled. |
|  | A transformation of information (a function) that resides within the bounds of the system to be modeled. |
|  | A data object; the arrowhead indicates the direction of data flow. |
|  | A repository of data that is to be stored for use by one or more processes; may be as simple as a buffer or queue or as sophisticated as a relational database. |

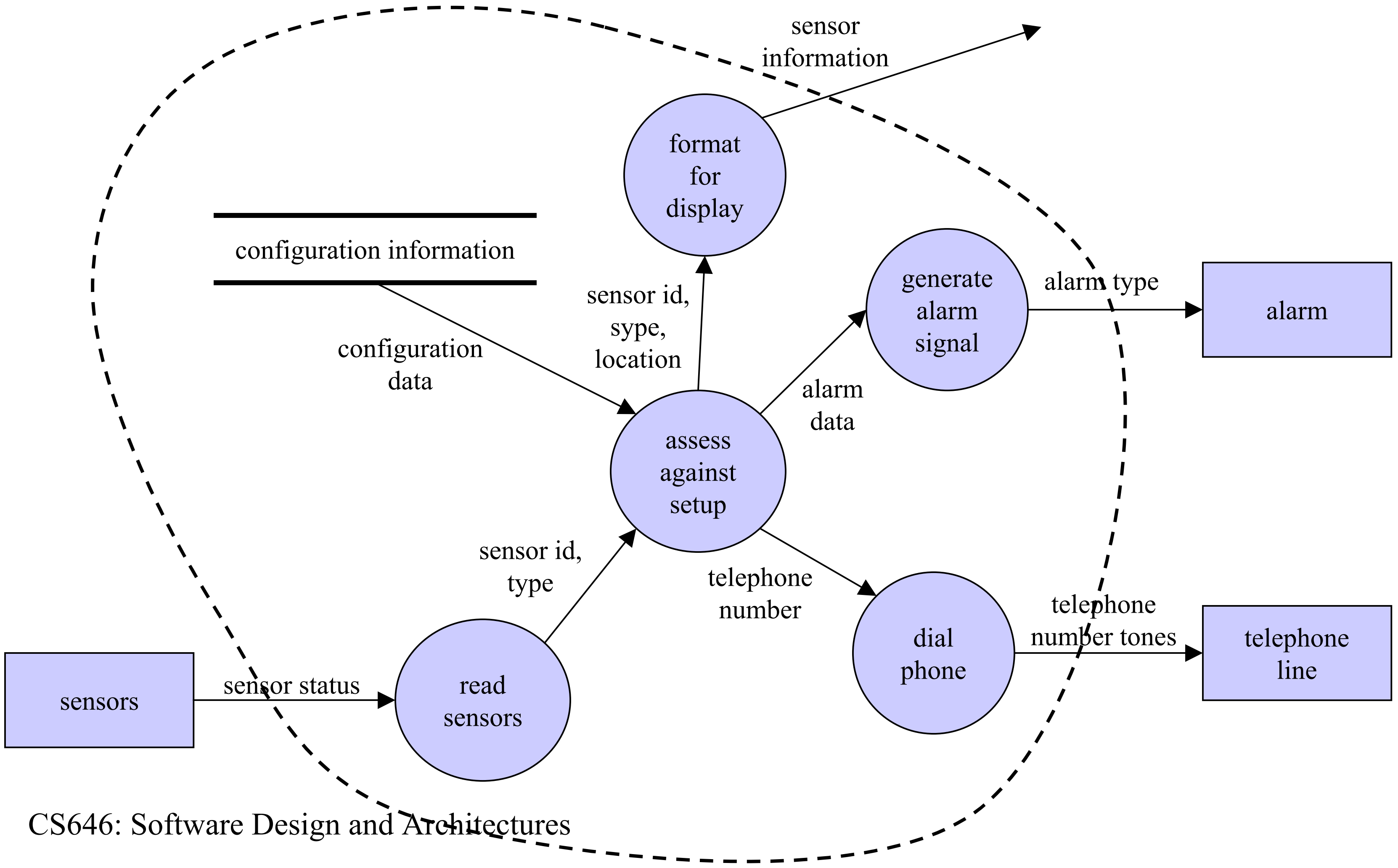
## E.g.: Level 0 for *SafeHome*



# E.g. (cont’): Level 1 (*SafeHome software*)



## E.g. (cont'): Level 2 (*monitor sensors*)



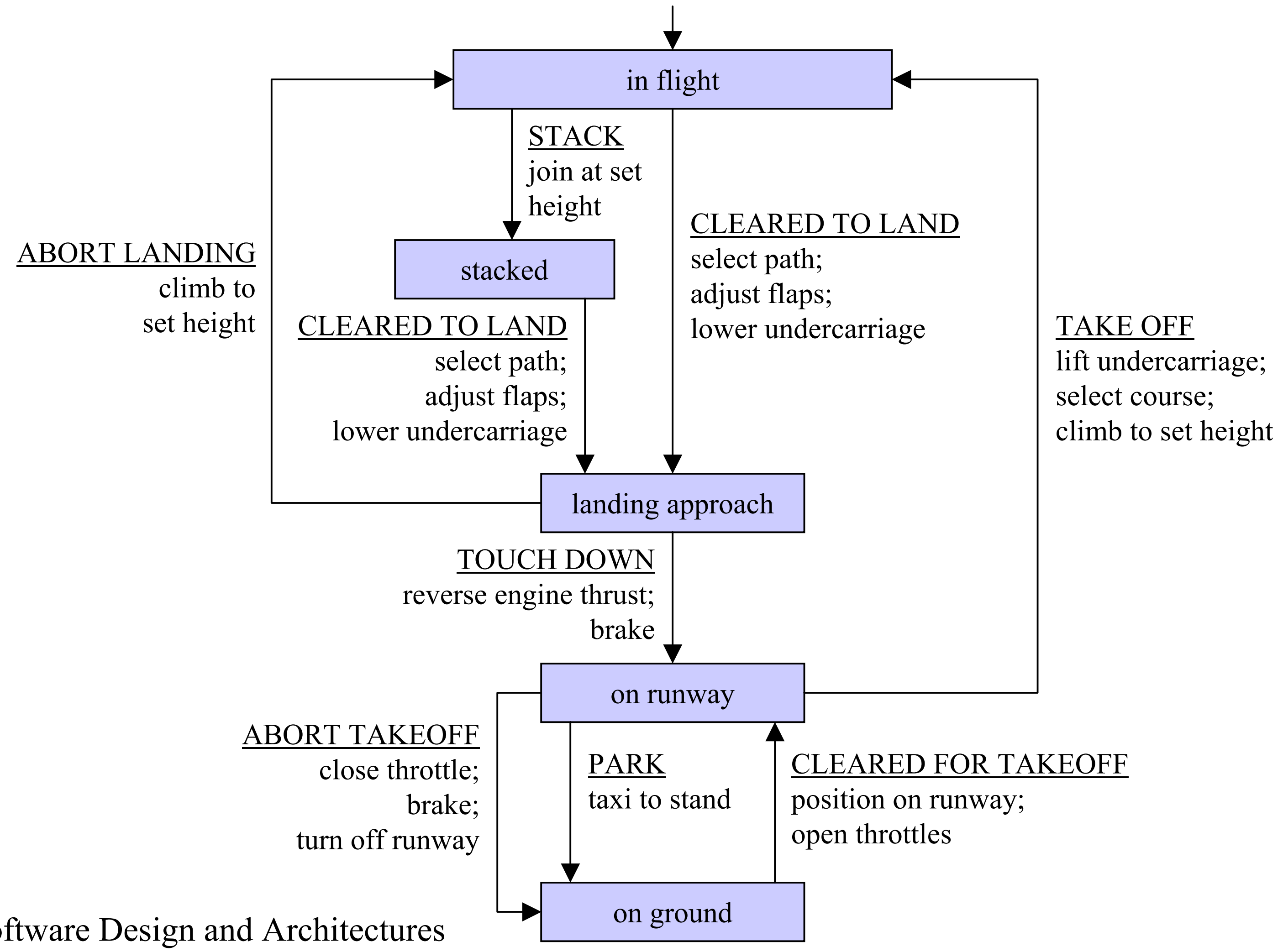
## *State Transition Diagrams* (STDs)

- Used for capturing *state transition behavior* in cases where there is an intuitive finite collections of *states*.

**E.g.:** a telephone call!

- Derives from the notion of a finite state automaton.
- Graph–based diagrammatic notation.
  - Labeled nodes correspond to states.
  - Arcs correspond to transitions.
  - Arcs are labeled with events and actions (actions can cause further events to occur).
- Describes a single underlying process.

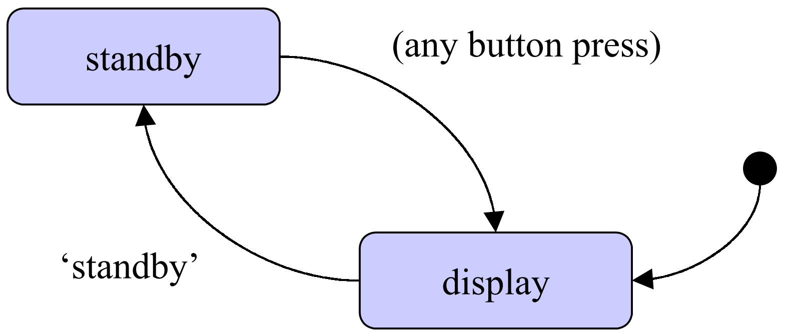
## E.g.: Aircraft landing behavior (from text)



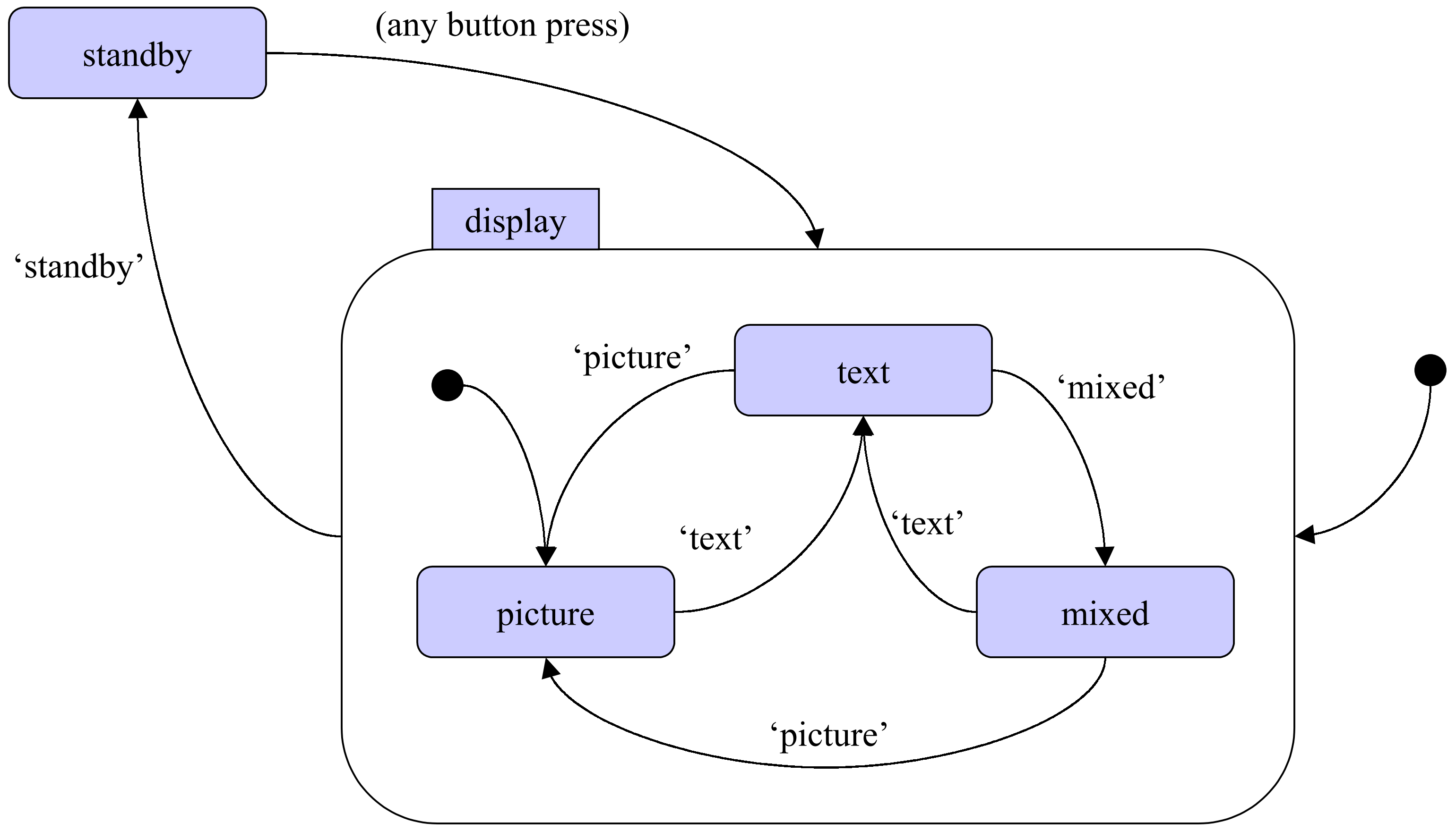
## *Statecharts*

- Developed by David Harel.
- A generalization of STDs: States can have zero, one, two or more STDs contained within.
- Related to Petri nets.
- Higraph–based diagrammatic notation.
  - Labeled nodes correspond to states.
  - Arcs correspond to transitions.
  - Arcs are labeled with events and actions (actions can cause further events to occur).
- Describes one or more underlying processes.

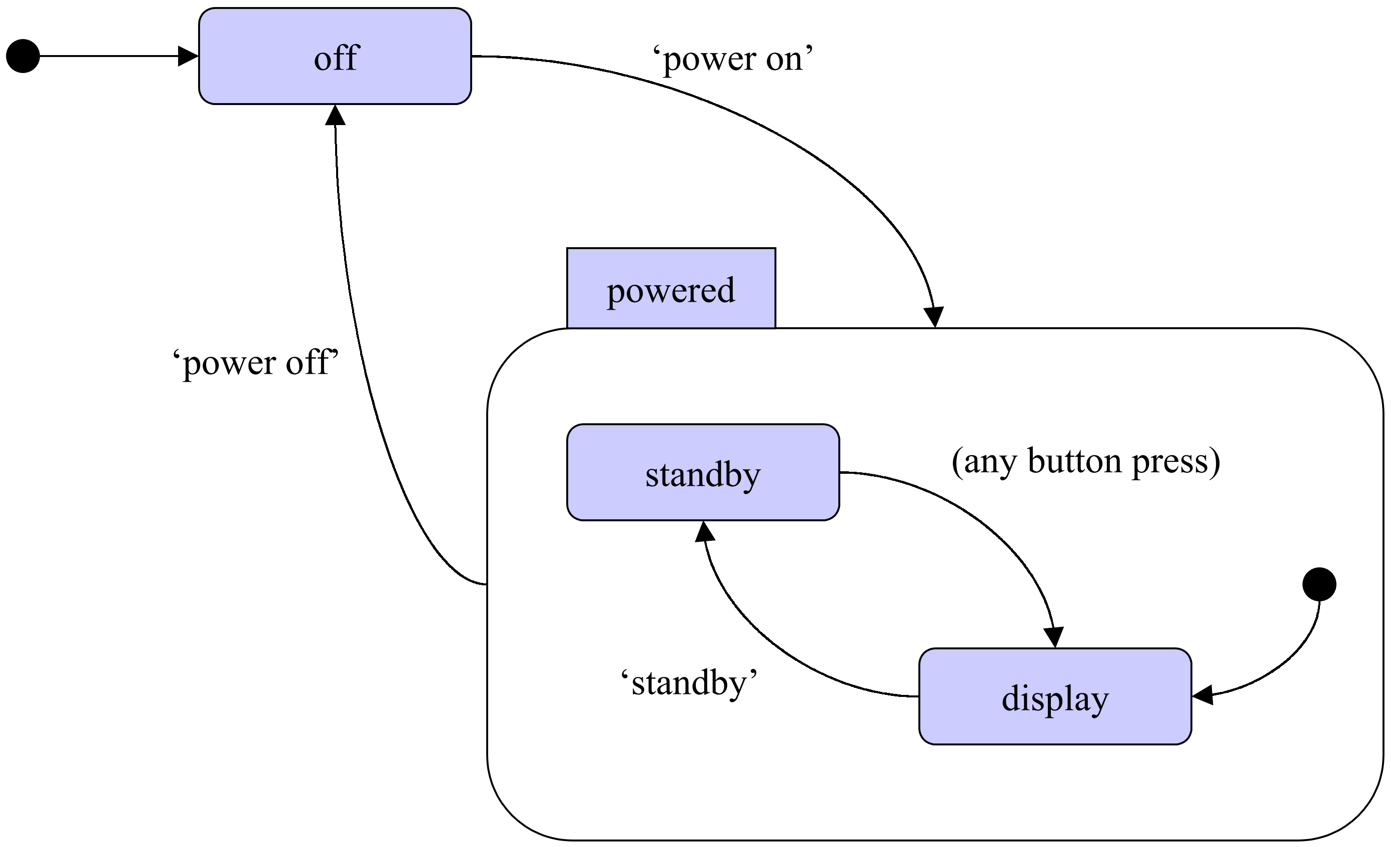
## E.g.: *Teletext television set* (from text)



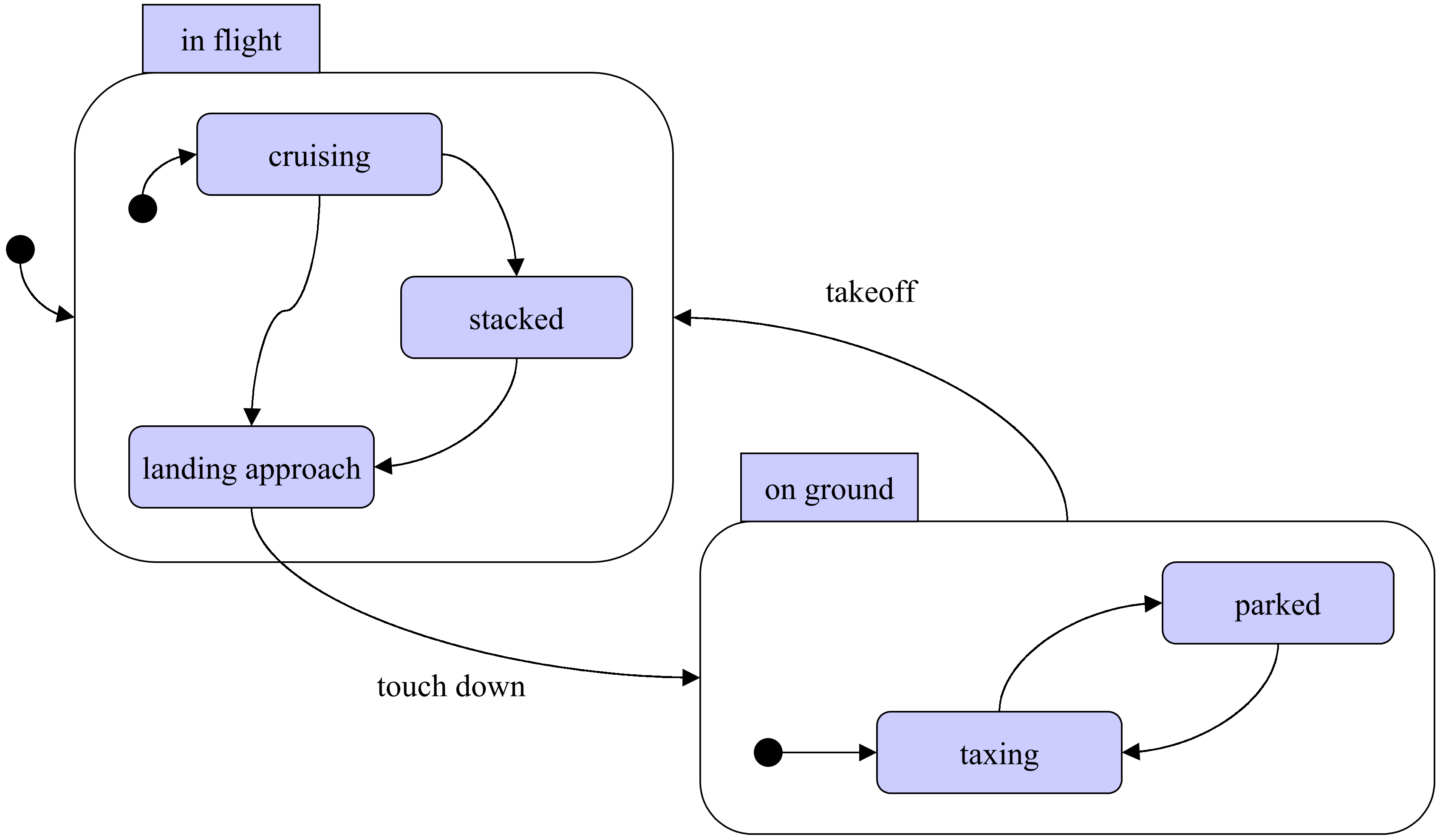
# E.g. (cont’d): *Teletext television set*



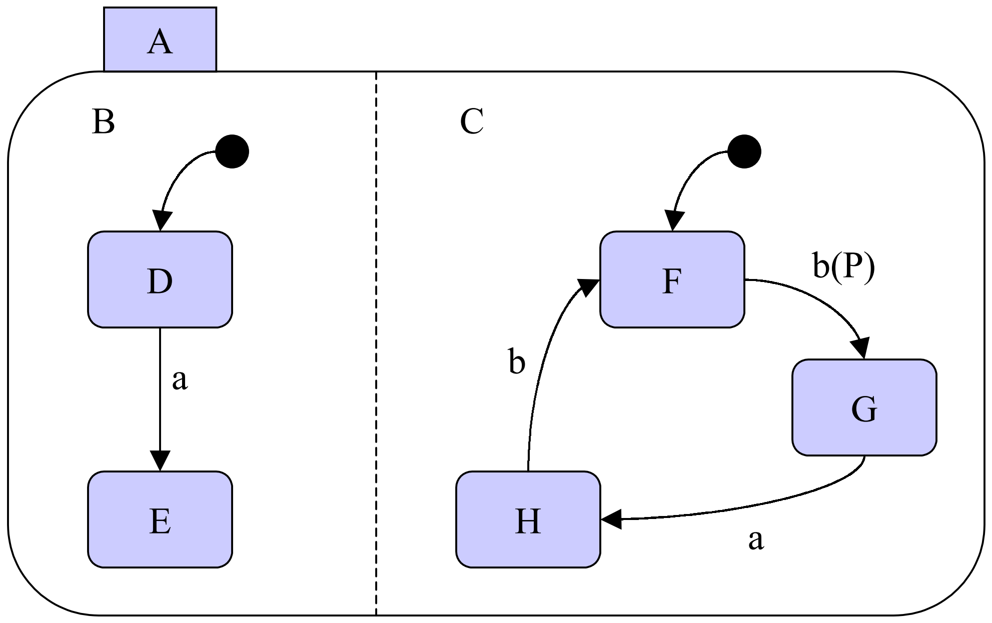
## E.g. (cont’d): *Teletext television set*



## E.g.: *Aircraft landing behavior* (from text)



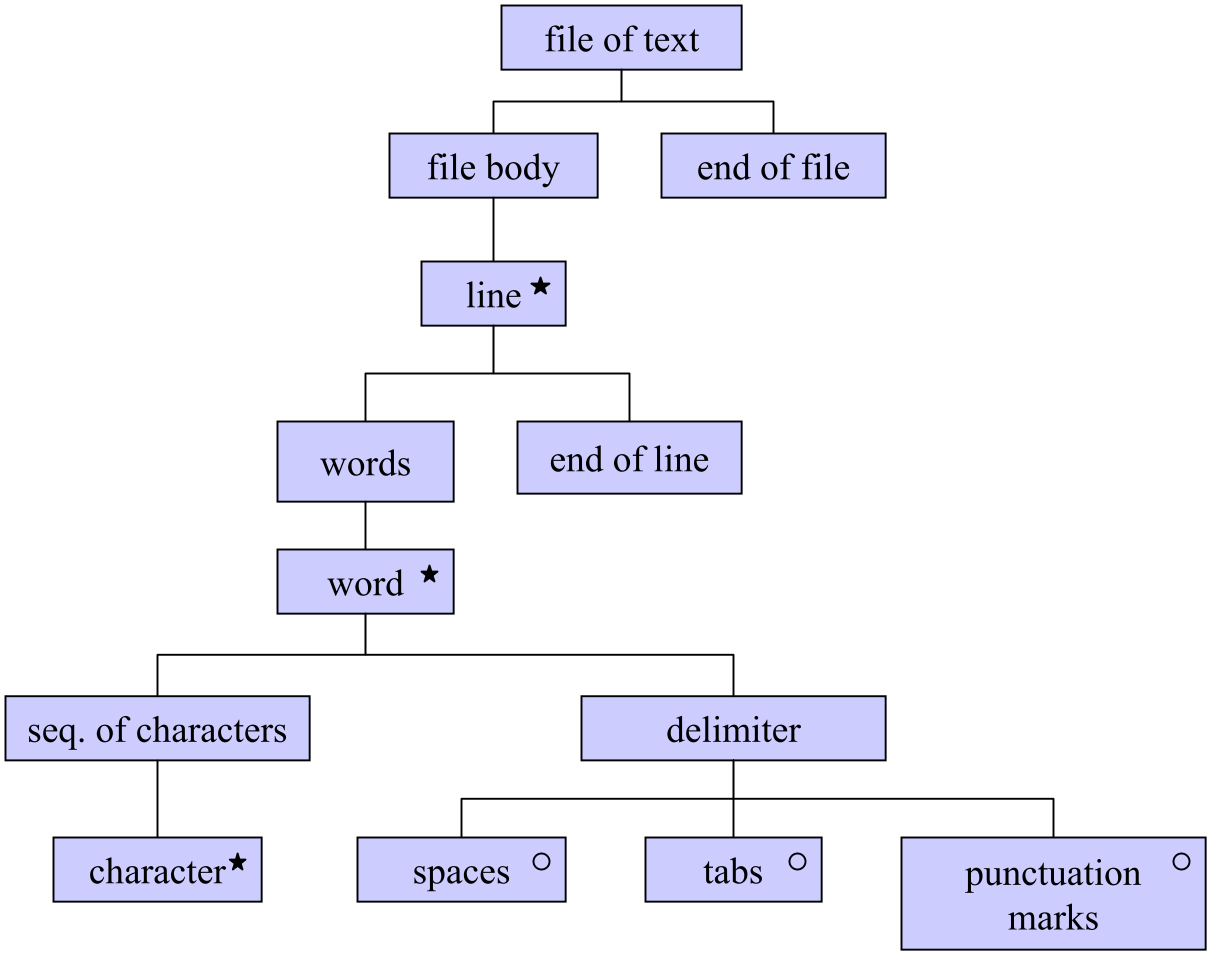
## E.g.: Describing more than one process



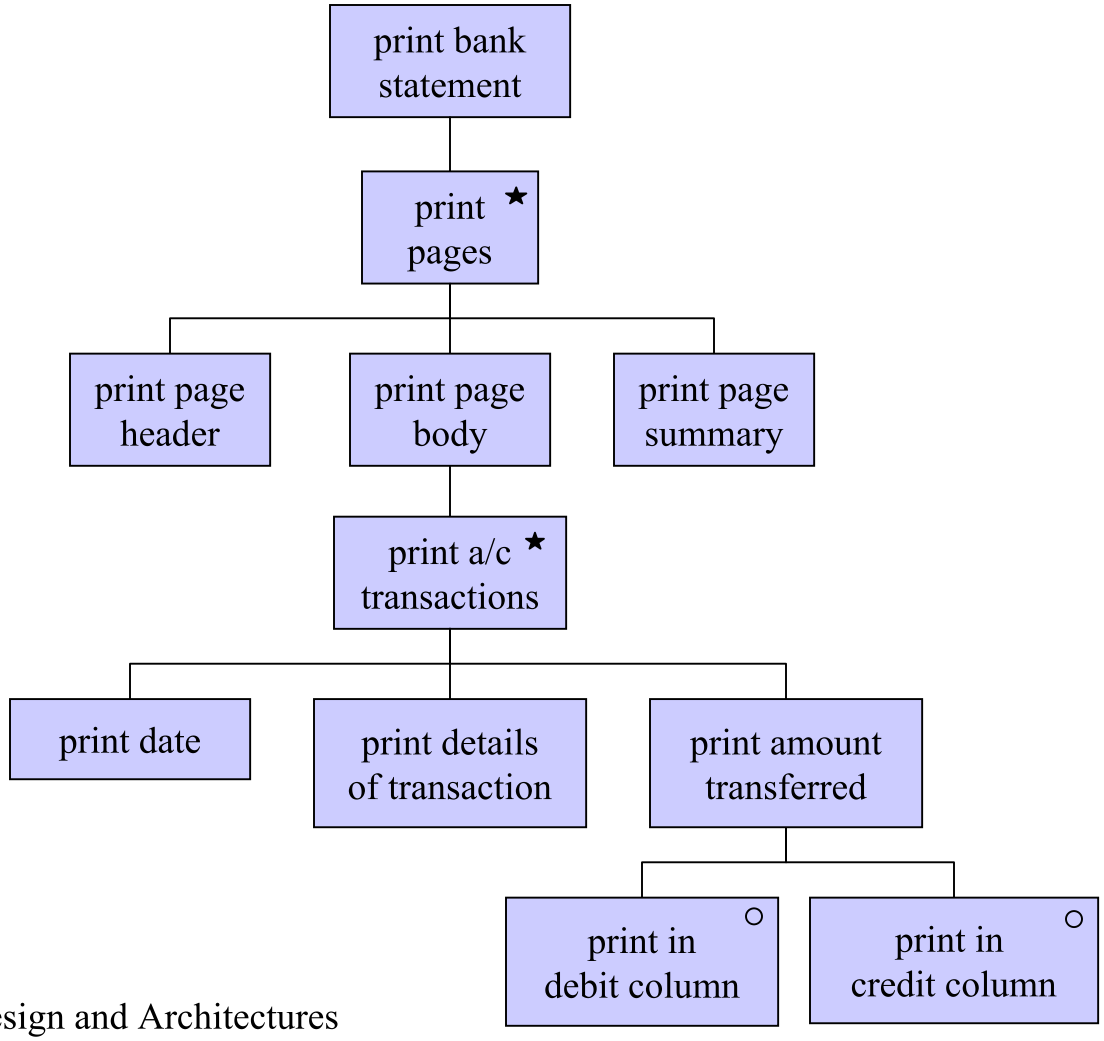
## *Structure Diagrams*

- Used in *Jackson Structured Programming*.
- Used to describe several kinds of things.
  - Ordered hierarchical structure.
  - Sequential processing.
- Based on the idea of regular languages.
  - Sequencing.
  - Selection.
  - Iteration.

## E.g.: Ordered hierarchical structure



## E.g.: Sequential processing



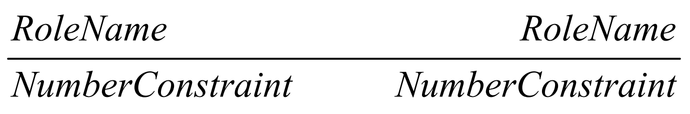
## *Entity Relationship Diagrams* (ERDs)

Slides on this are in a separate file.

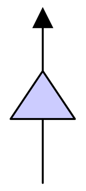
# *Class Diagrams*

- Derived from ERDs.
- Limited to binary relationships.

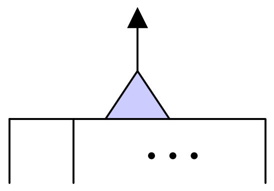
Diagrammatic elements:



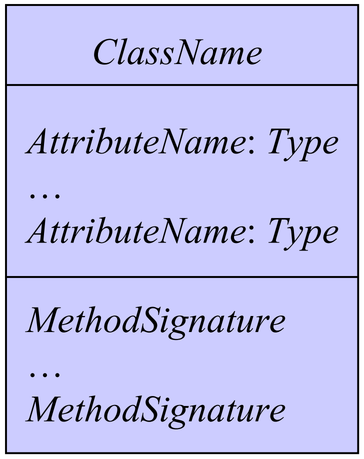
binary relationship



specialization

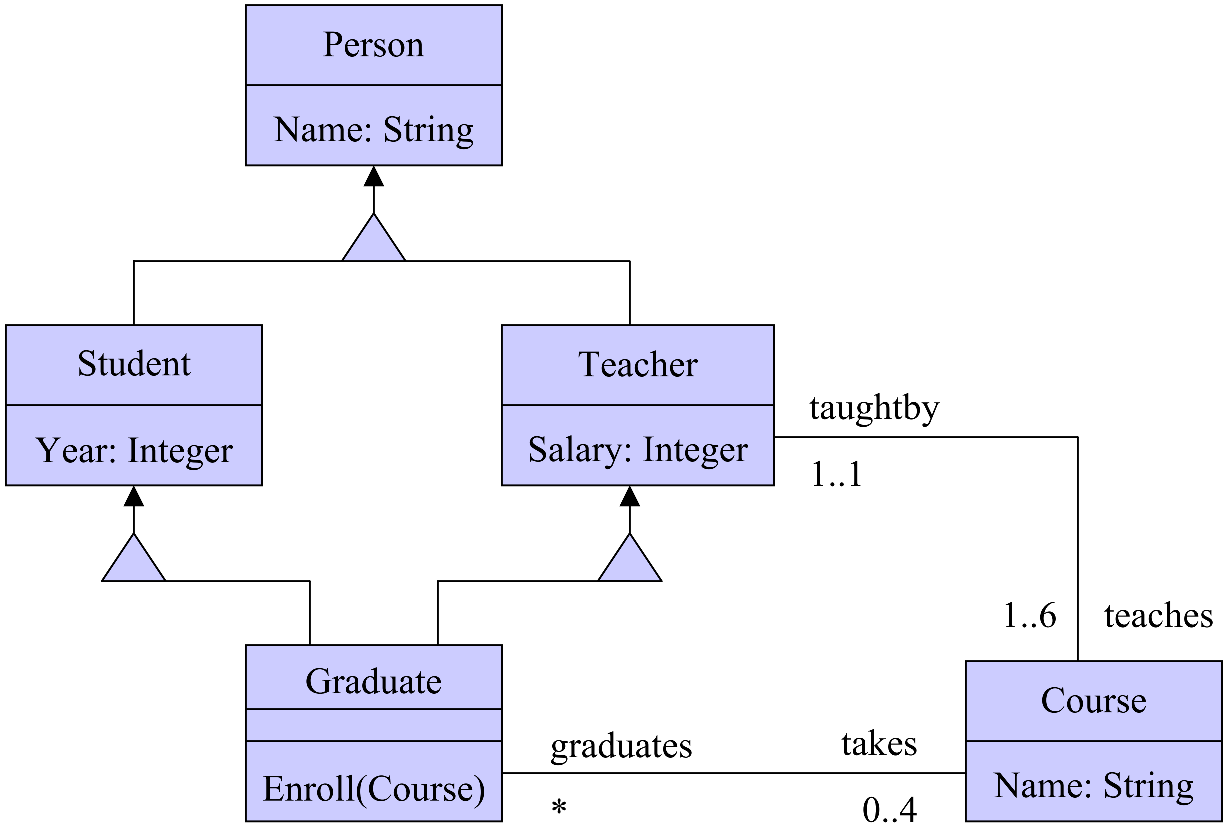


generalization



class definition

# E.g.: University personnel

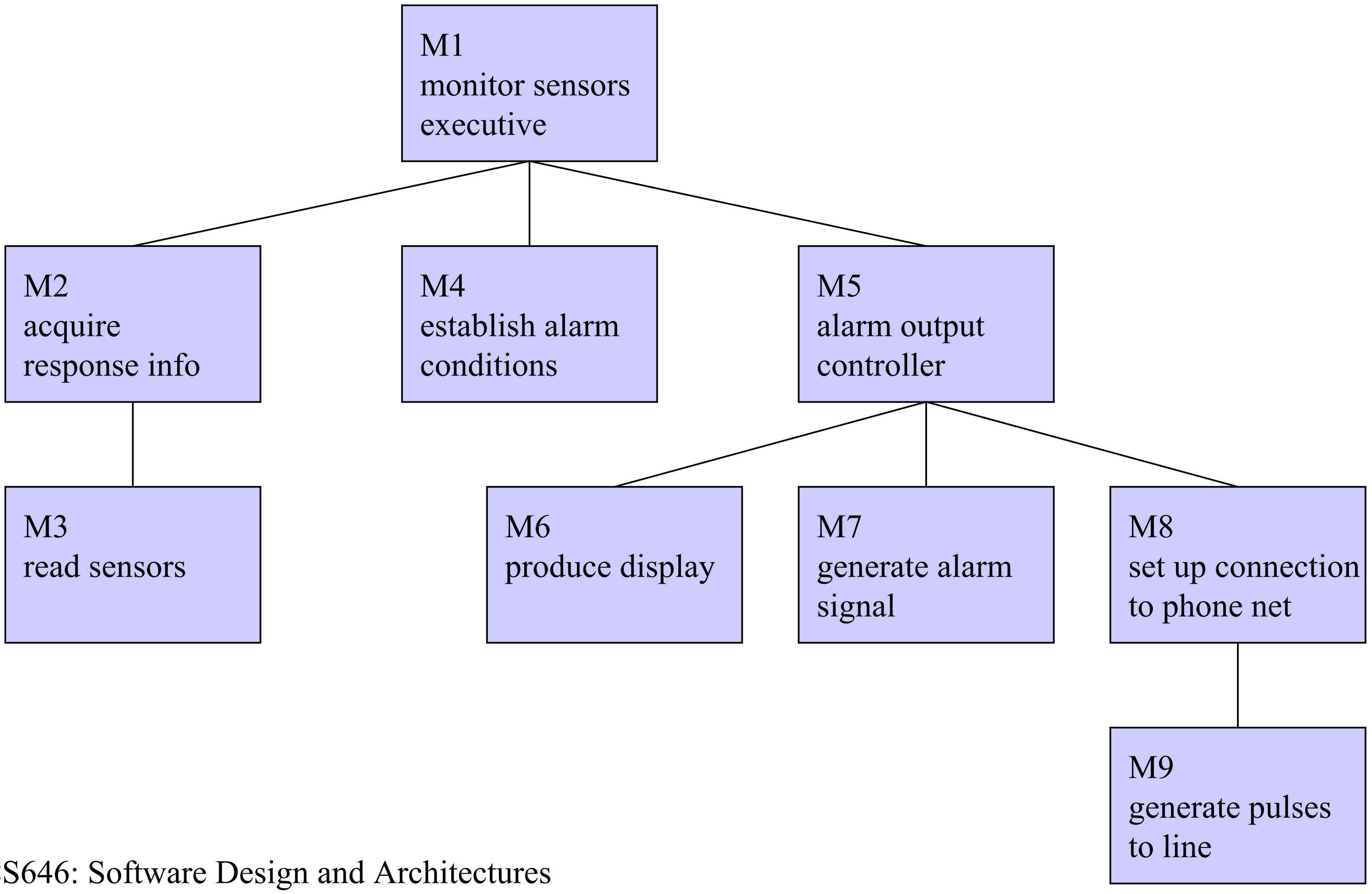


## *Structure Charts*

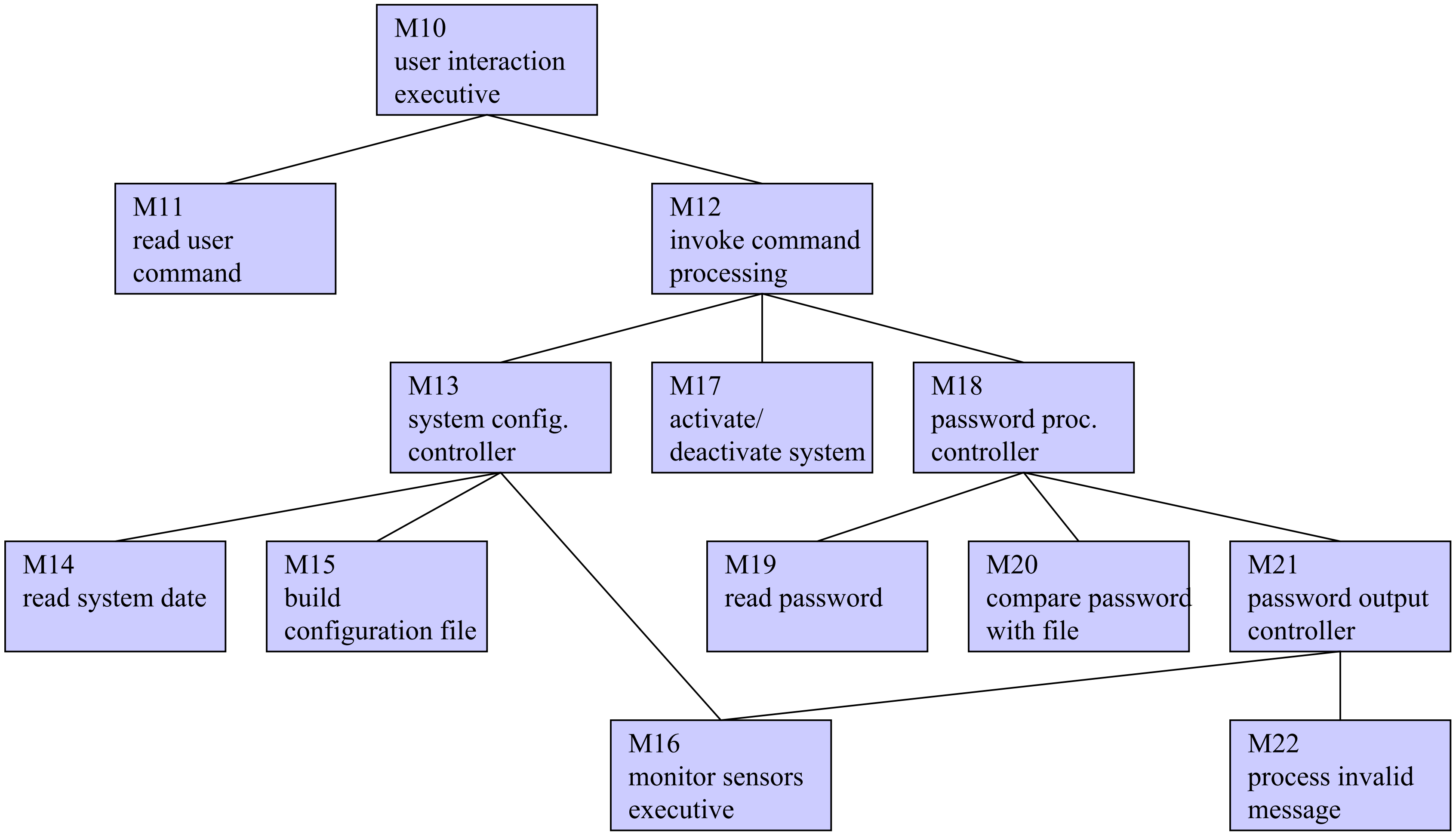
- Based on the fundamental notion of a *module*.
- Used in *structured systems analysis/structured design* (SSA/SD).
- Graph–based diagrammatic notation: a structure chart is a collection of one or more node labeled *rooted directed acyclic graphs*.
  - Each graph is a process.
  - Nodes and modules are synonymous.
  - A directed edge from module M1 to module M2 captures the fact that M1 directly uses in some way the services provided by M2.

**Definitions:** The *fan-in* of a module is the count of the number of arcs directed toward the module. The *fan-out* of a module is the count of the number of arcs outgoing from the module.

## E.g.: *SafeHome* (*monitor sensors*)

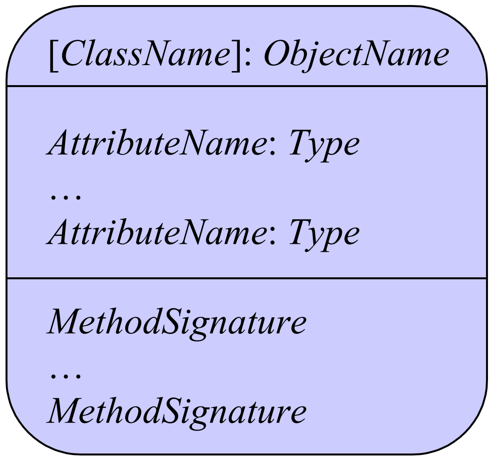


# E.g.: *SafeHome* (*interact with user*)

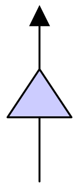

M10
user interaction executive
M11
read user command
M12
invoke command processing
M13
system config. controller
M17
activate/ deactivate system
M18
password proc. controller
M14
read system date
M15
build configuration file
M19
read password
M20
compare password with file
M21
password output controller
M16
monitor sensors executive
M22
process invalid message

# *Object Diagrams*

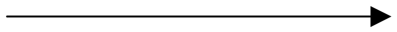
- Derived from structure charts.
- Much in common with class diagrams.



object definition

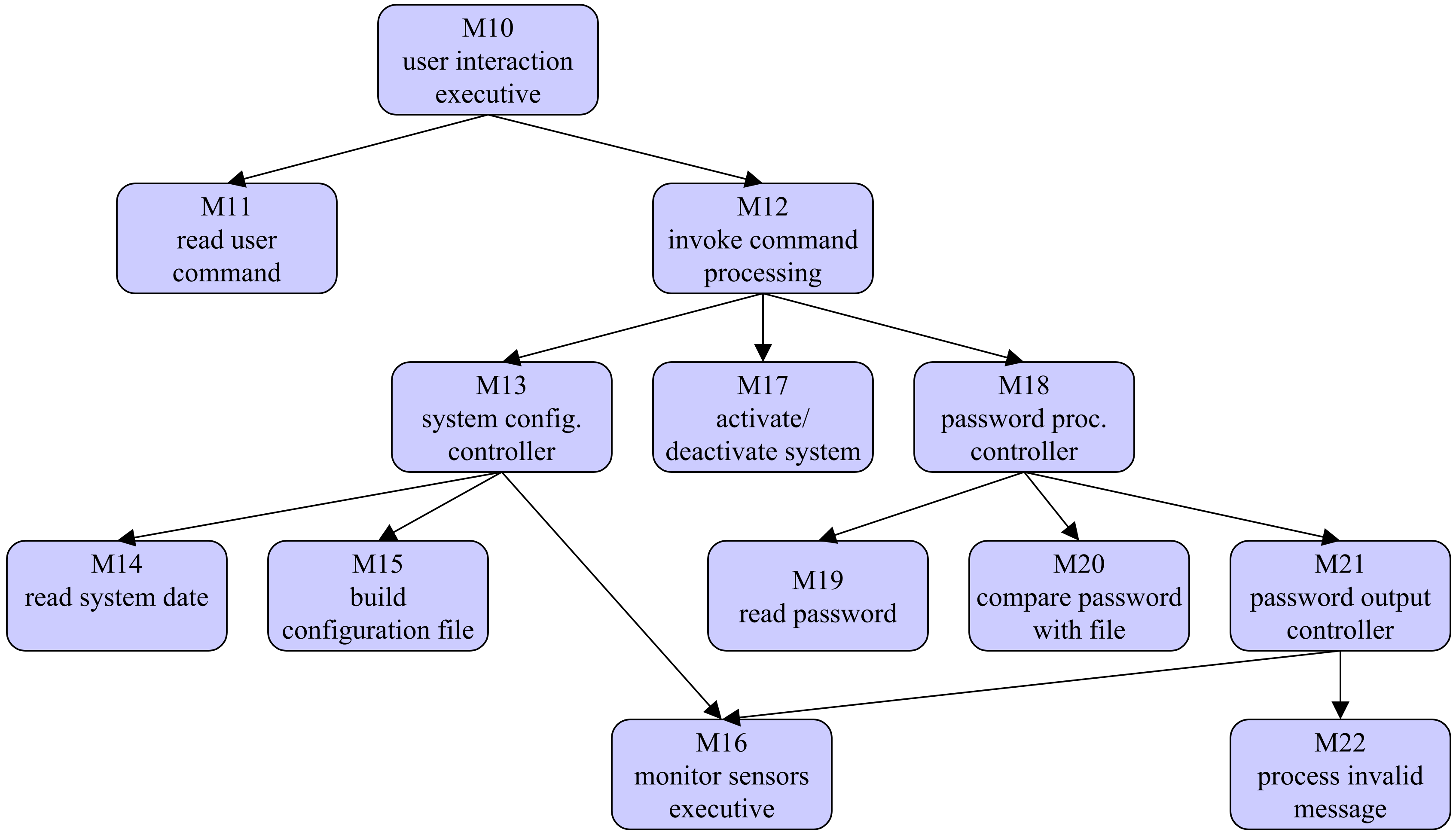


delegation



utilization

# E.g.: *SafeHome* (*interact with user*)


M10
user interaction executive
M11
read user command
M12
invoke command processing
M13
system config. controller
M17
activate/ deactivate system
M18
password proc. controller
M14
read system date
M15
build configuration file
M19
read password
M20
compare password with file
M21
password output controller
M16
monitor sensors executive
M22
process invalid message