

Statechart Diagrams

- describe changes between states of instances
- similar to activity diagram (work flow)
- behaviour of dynamic elements
- generally, behaviour of classes
- single entity viewpoint
- transitions triggered by external/internal events

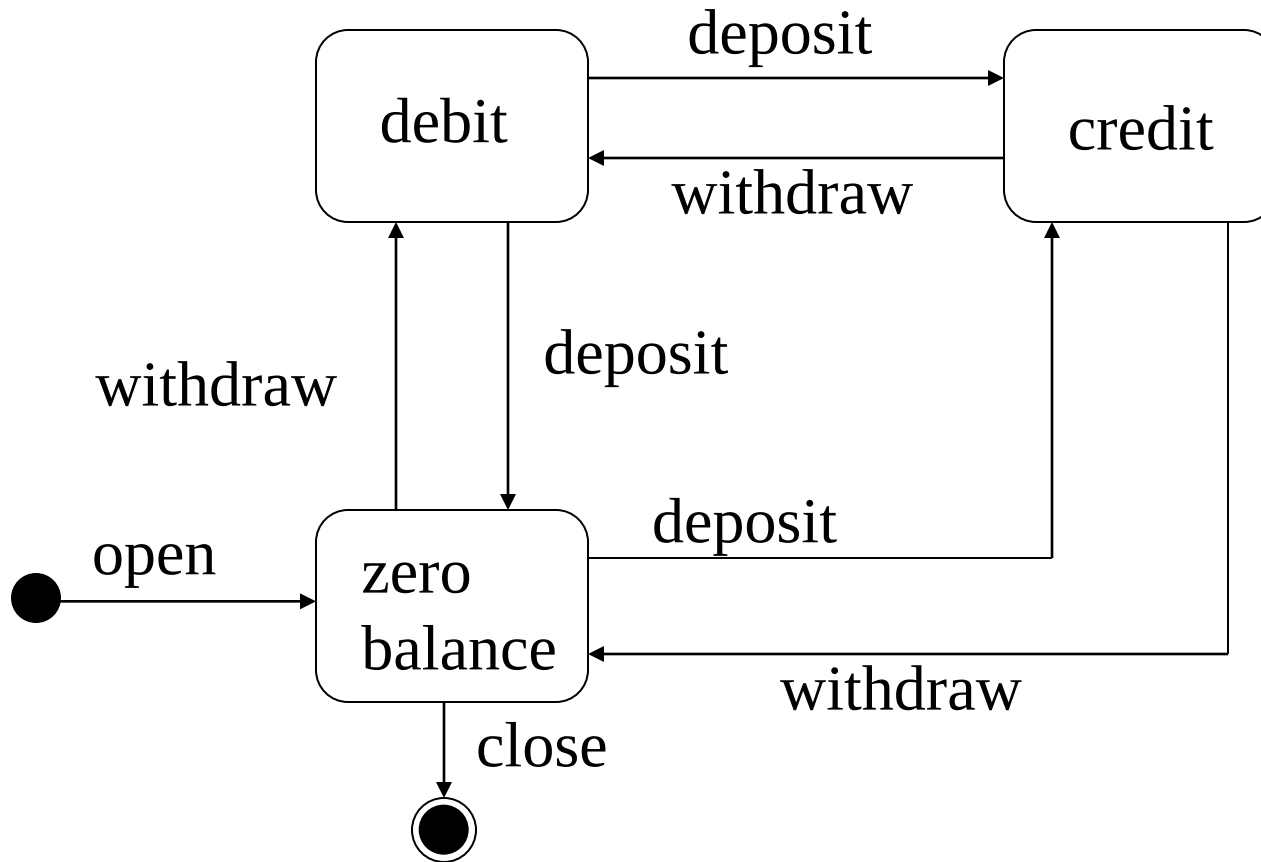
Statechart Diagrams

- state - action being performed, event awaited
 - rectangle with rounded corners
 - start and end states (activity diagrams)
 - states trigger actions
 - on entry, do, on event, on exit, include (calls another statechart diagram)
 - action label / action
 - event(parameter) [guard condition] / action

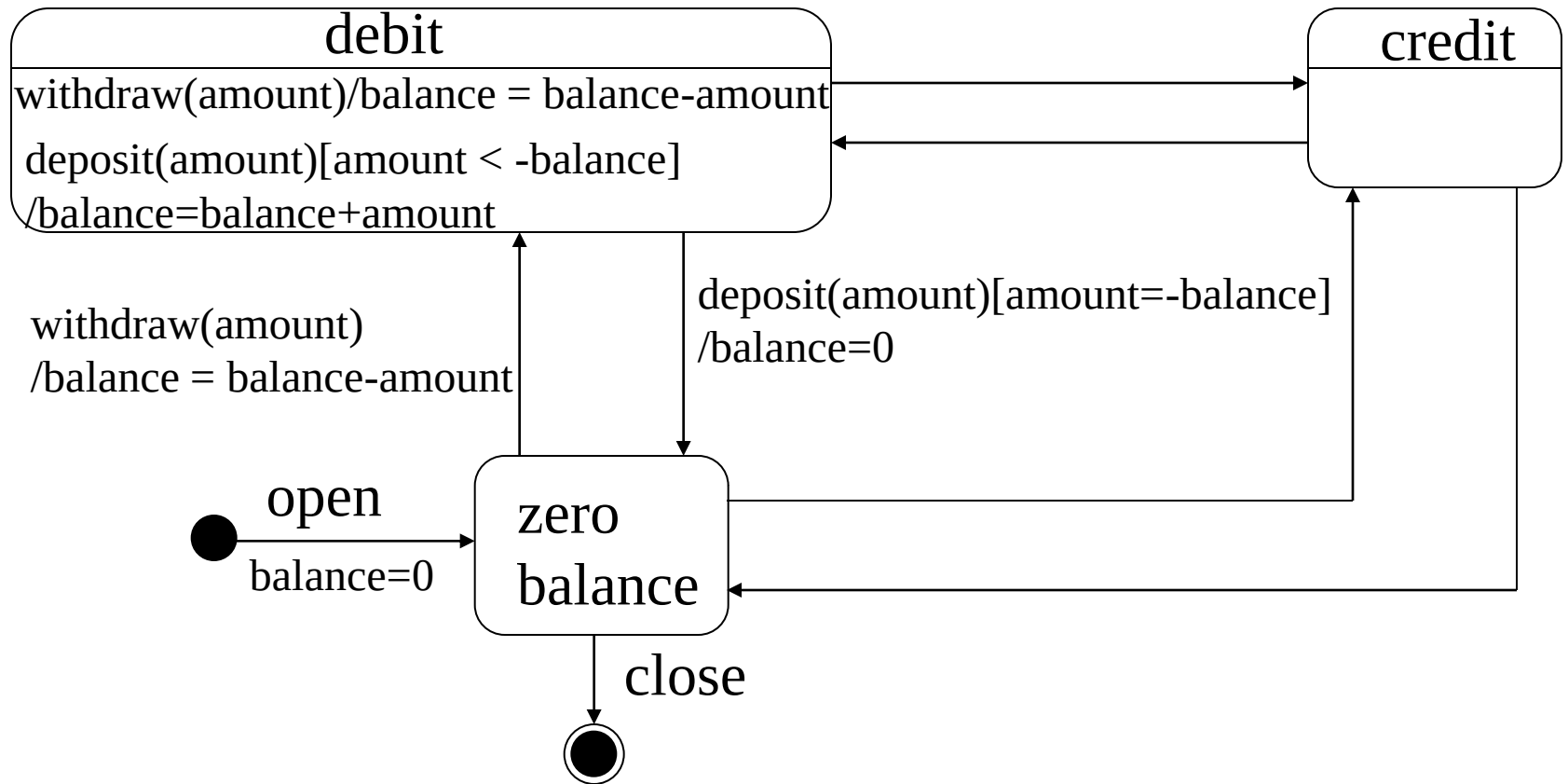
Statechart Diagrams

- transition - movement between states
 - labelled, arrowed line
- composite diagrams sometimes used
 - nested state diagram

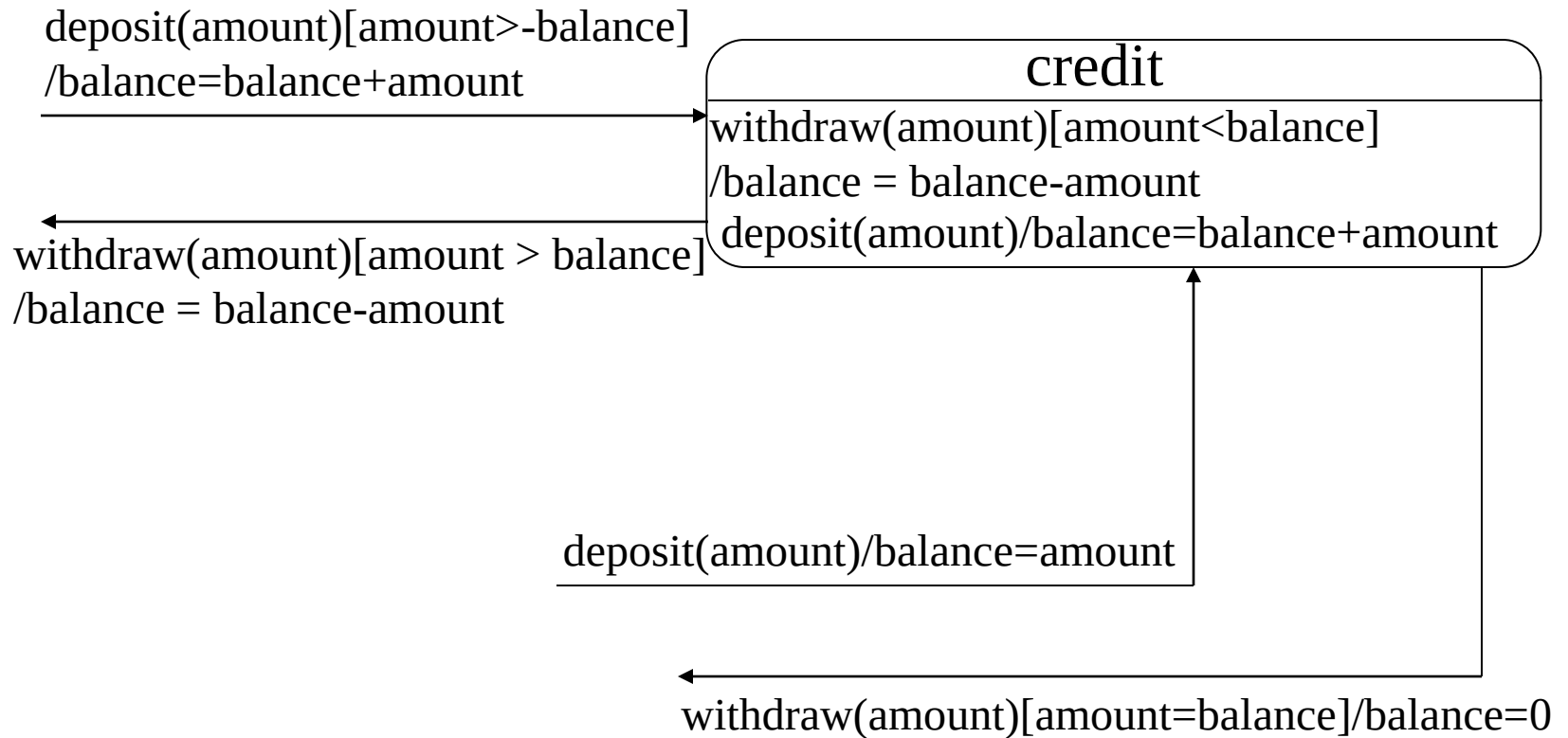
Statechart Diagrams



Statechart Diagrams



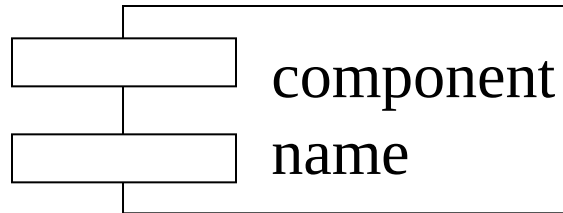
Statechart Diagrams



Implementation Diagrams

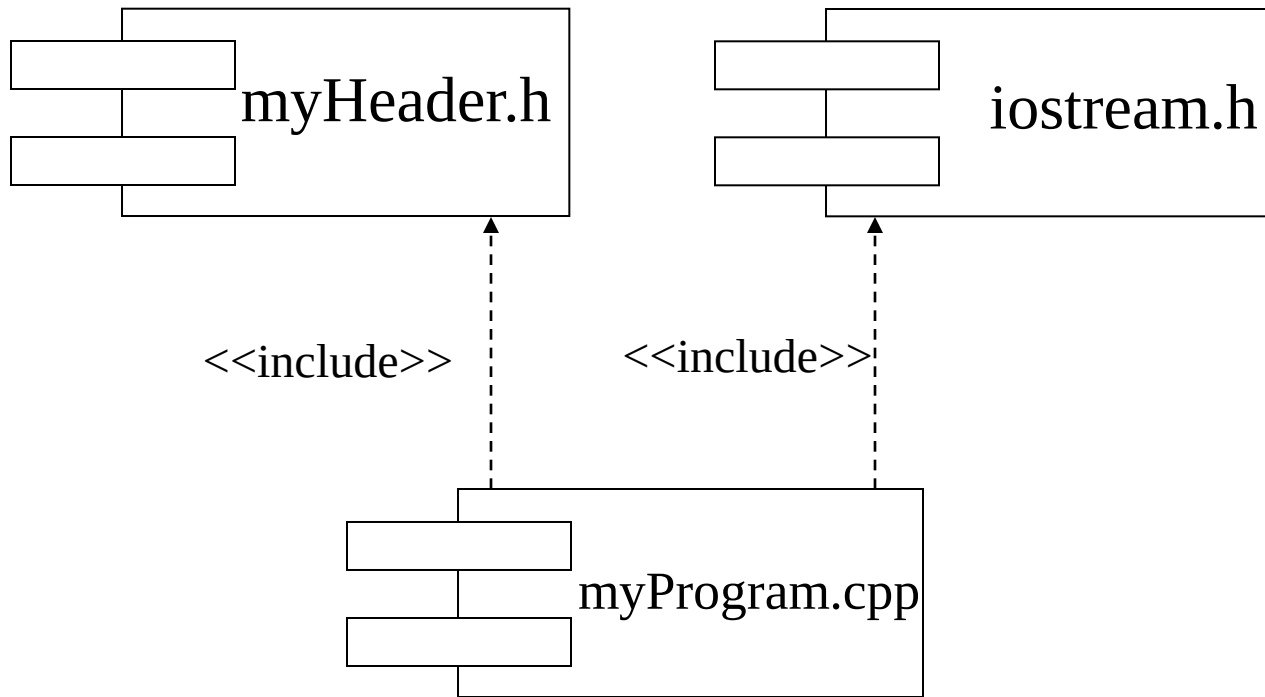
- component diagram
 - relationship between software components
- deployment diagram
 - model hardware on which system is implemented
 - embedded processors
 - sensors, peripherals

Component Diagrams

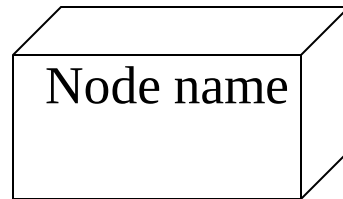


- physical file, sub-system
- dependency
 - dashed arrow, from dependent
 - <<include>>, <<derive>>, <<friend>>, <<import>>
- classes contained can be listed
- types not instances

Component Diagrams

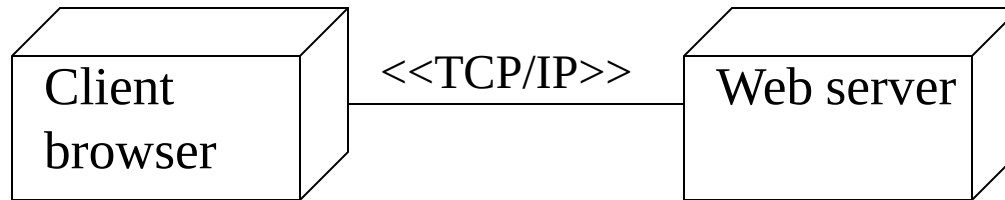


Deployment Diagrams



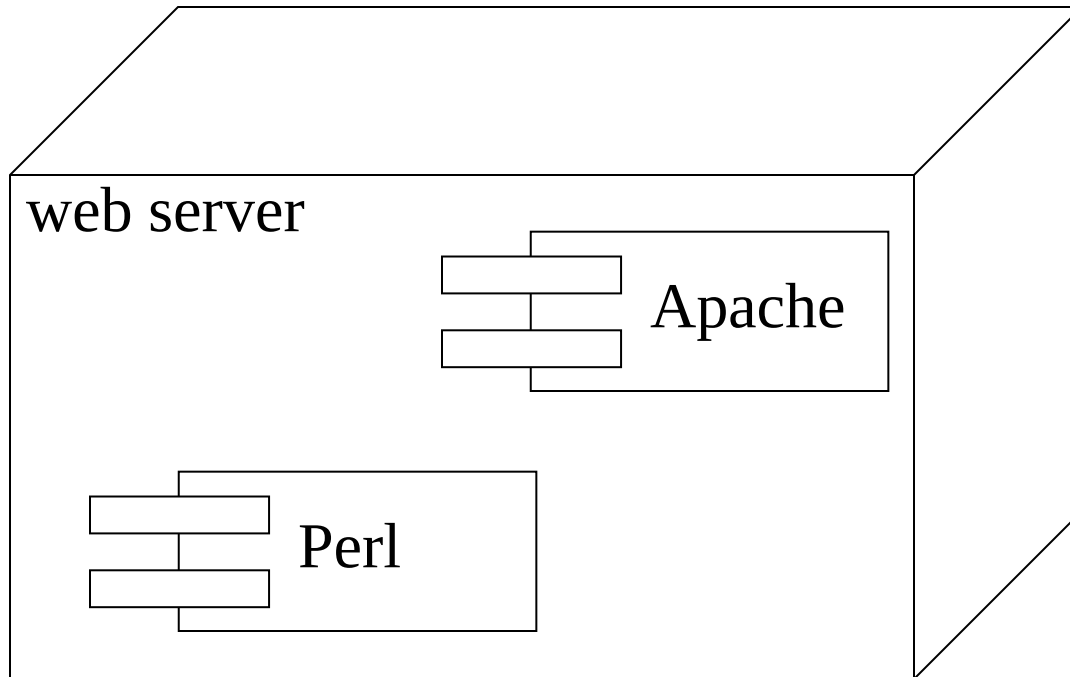
- processing resources, computers
- associations
 - connections, solid line
 - labelled to show nature of connection

Deployment Diagrams



- show types or instances
- component instances can be shown
 - executables only

Deployment Diagrams



Object Constraint Language

- rigorous expression of constraints/rules
- precise notation for definition of behaviour
- clear, unambiguous language
- Z, VDM (formal languages)
 - set theory
 - predicate calculus
- less mathematical

Object Constraint Language

- design by contract
 - Bertrand Meyer, Design by Contract, Technical Report TR-EI-12/CO, ISE Inc., 1987
 - relationship between class and client, formal agreement
 - reliability (correctness and robustness)
 - obligations of class and client
 - data supplied
 - results produced
 - contract violations (exception handling)

Object Constraint Language

- invariant
 - global properties
 - conditions to be preserved by all routines
 - consistency constraints
- pre-condition
 - client obligation/class(supplier) benefit
 - legitimate call
 - various assumptions satisfied

Object Constraint Language

- pre-condition
 - constraints under which routine will function correctly
 - require
- post-condition
 - client benefit/class obligation
 - various properties satisfied after call
 - ensure

Object Constraint Language

- notation
 - context
 - inv
 - pre
 - post
 - operations (typical programming language)
 - arithmetic
 - relational
 - logical
 - dot notation

Object Constraint Language

- data types

- Boolean
- integer
- real
- string

- collections

- set - unique instances
- bag - repetitions, unordered
- sequence - repetitions, ordered

Object Constraint Language

– select operation

- arrow notation, $\rightarrow$
- extract subset satisfying particular condition
- $a \rightarrow \text{select}(x \mid x > 100)$
- union, intersection, including, ...

– reject operation

- arrow notation, $\rightarrow$
- extract subset failing particular condition

Object Constraint Language

- collect operation
 - arrow notation, $\rightarrow$
 - apply expression to each element of collection
 - $a \rightarrow \text{collect}(x \mid x * x)$

Stack

context Stack inv :

Stack.size ≥ 0 and

Stack.size $\leq$ MAX_SIZE

context Stack::push(value:integer)

pre: Stack.size $<$ MAX_SIZE

post: Stack = Stack + value

size = size + 1

Stack.empty = false

Stack

context Stack::pop():integer

pre: Stack.size > 0

post: Stack = Stack - tos

size = size - 1

Stack.full = false

return (tos)

Stack

context Stack::full():Boolean

pre: true

post: return (size=MAX_SIZE)