COMP201 – Software Engineering I
Lecture 8 – System Models

Lecturer: T. Carroll

Email: Thomas.carroll2@Liverpool.ac.uk

Office: Ashton G.14

See Vital for all notes

Recap

Recap Lecture 7

- What are system models?
  - Graphical abstractions of the system
  - Show detail from different perspectives
  - Help communication between developers and customers
- Use Case Models
  - Shows how different actors can use the system
- Sequence Diagrams
  - Gives detail of a particular actor's interaction with the system, showing computation and messages passed between objects
- Process Models
  - Shows details of a process in context of the system
- Architectural Models
  - Show how the different subsystems, which comprise the system, are related

Today

Overview – Lecture 8

- Data Flow Diagrams
- UML Quick Overview
- Statechart Diagrams
- Semantic Data Models
- Data Dictionaries

Behavioural Models

Behavioural Models

- **Behavioural models** are used to describe the overall behaviour of a system
- **Data processing models** that show how data is processed as it moves through the system
- **Statechart diagrams** that show how the system should respond to events

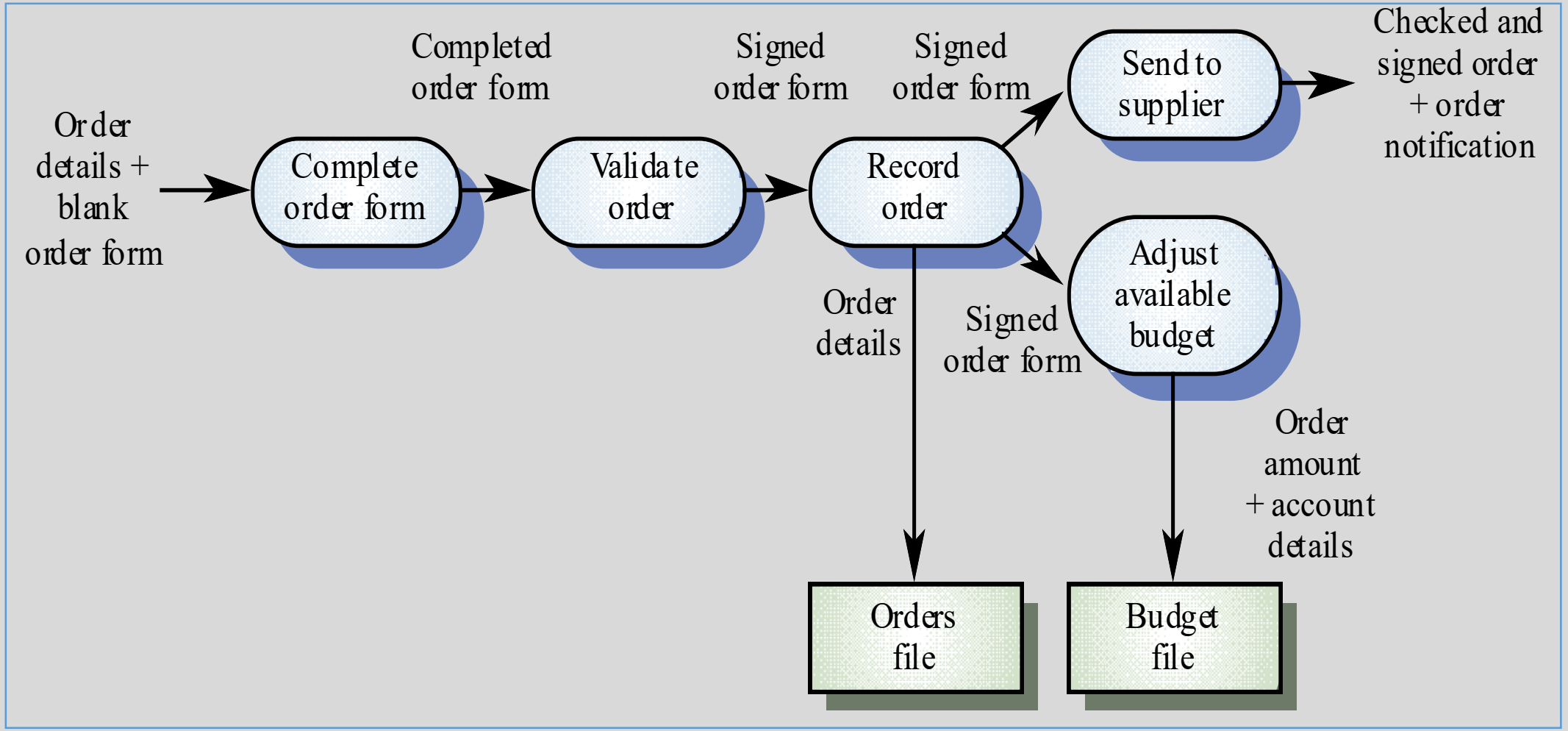
Both are required for a full description of the system's behaviour

Data-Processing Models

- **Data flow diagrams** are used to model the system's data processing
- These show the processing steps as data flows through a system
- They form an **important** part of many analysis methods
- Simple and intuitive notation
- Show end-to-end processing of data

Aknowledgement:http://www.doughlaminator.com/photo/pl13460281-high_automation_donut_production_line_with_modular_dough_sheeting_system.jpg

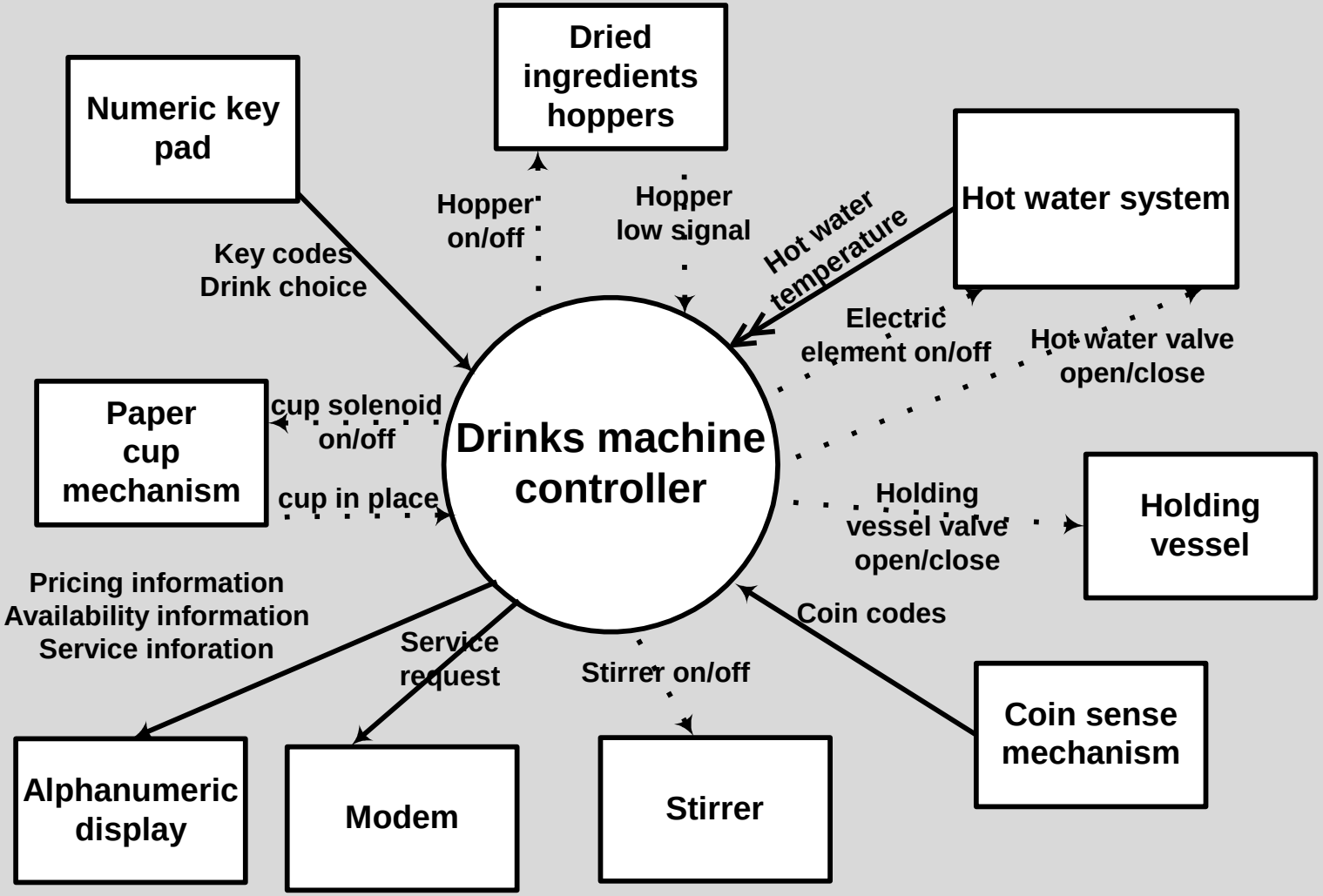
Example - Order Processing Data Flow Diagram



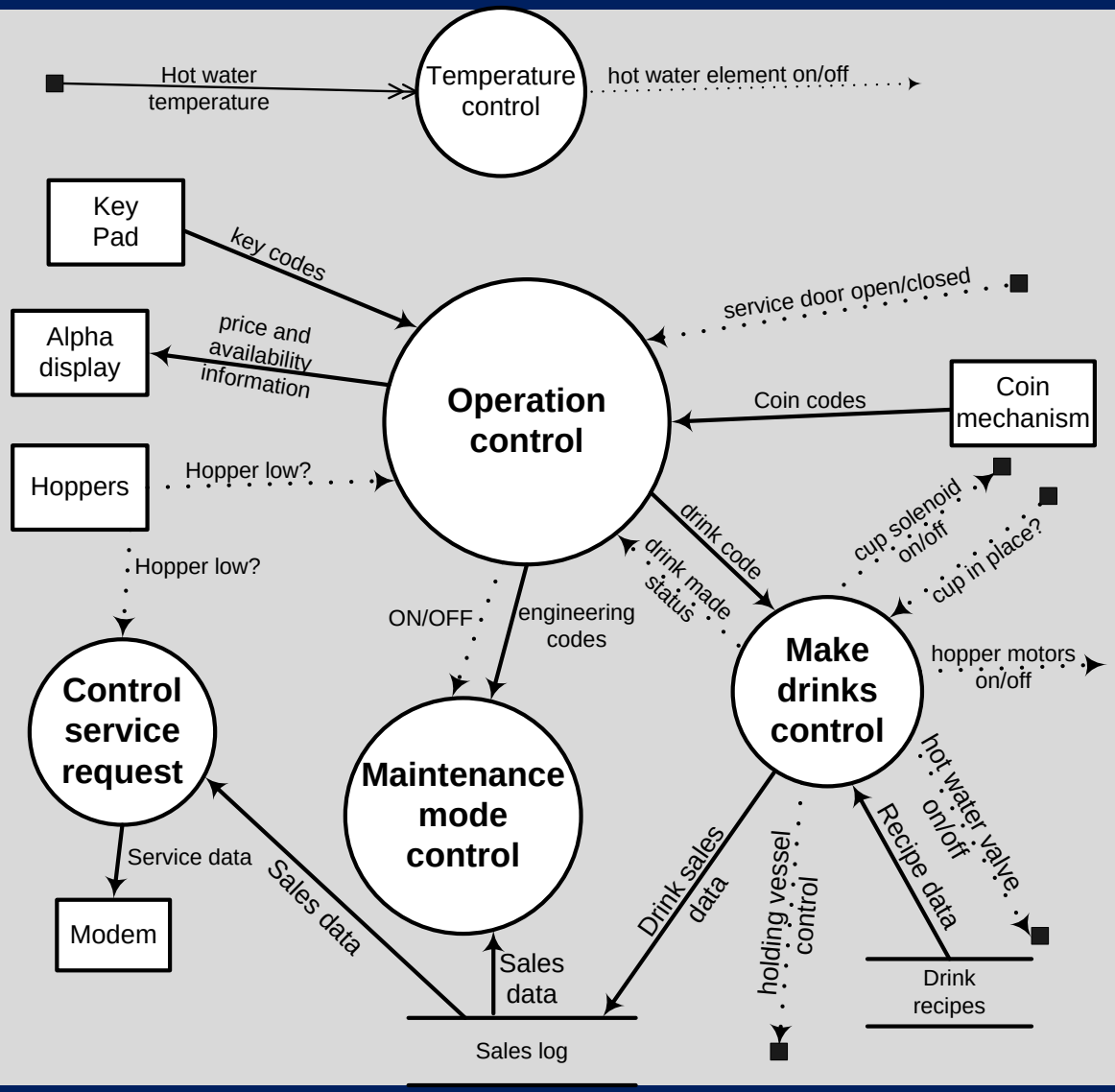
Data Flow Diagrams

- **Data Flow Diagrams** track and document *how the data associated with a process* is helpful
- Data flow diagrams show a **functional perspective** where each transformation represents a single function or process
- Particularly useful during requirements analysis - shows end-to-end processing.
- This helps to develop an overall understanding of the system
- Data Flow Diagrams are **simple and intuitive**
- Can be shown to users who can help in *validating the analysis*
- Data Flow Diagrams can show data exchange between a system and other systems in its environment
- Developing data flow diagrams is usually a *top-down process*
  - We begin by evaluating the overall process we wish to model before considering sub-processes

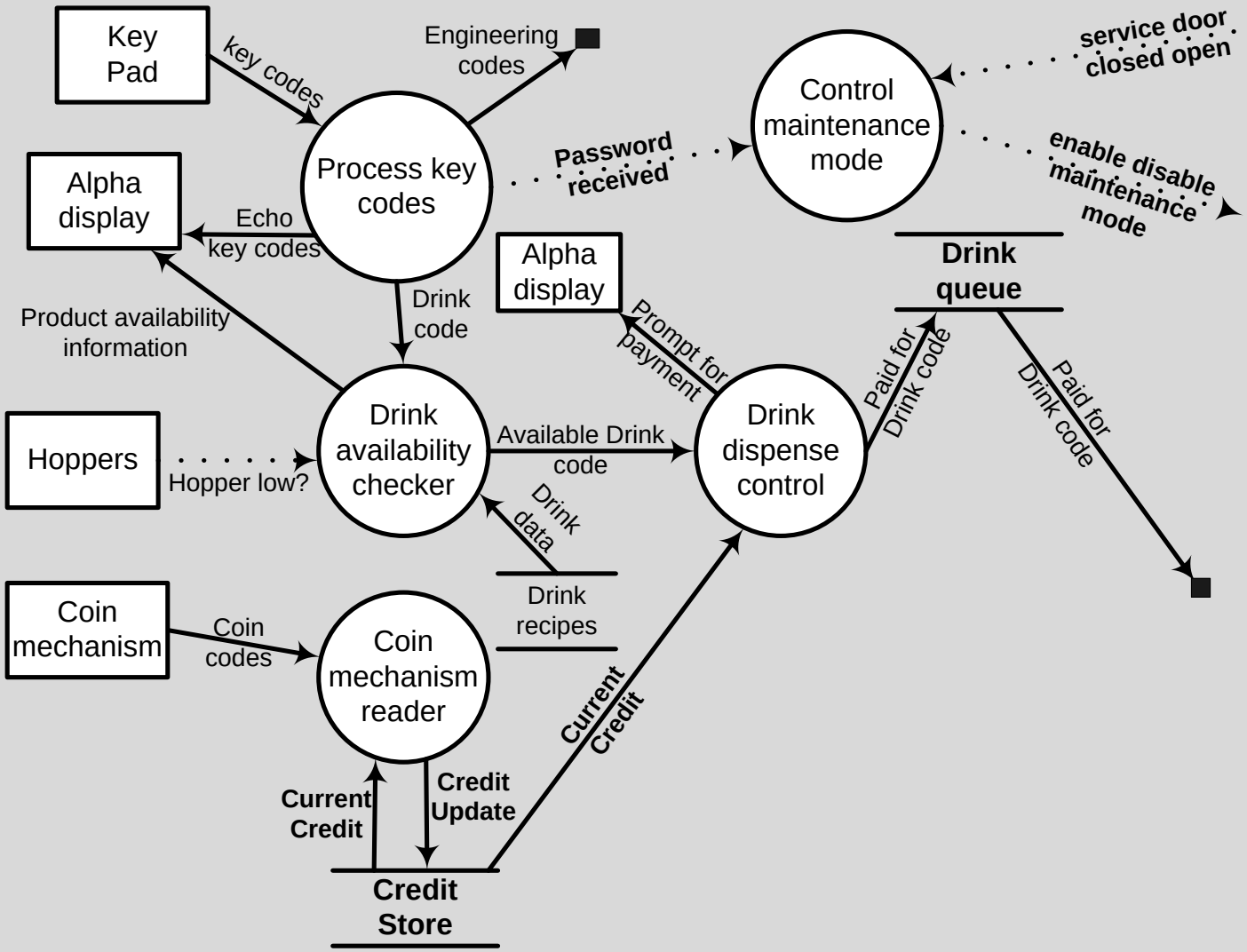
DFD Context diagram



Level 0 DFD


Hot water temperature
Temperature control
hot water element on/off
Key Pad
key codes
Alpha display
price and availability information
Operation control
service door open/closed
Coin codes
Coin mechanism
Hoppers
Hopper low?
Hopper low?
drink code
drink made status
cup solenoid on/off
cup in place?
ON/OFF
engineering codes
Make drinks control
hopper motors on/off
Control service request
Maintenance mode control
hot water valve on/off
Recipe data
Drink sales data
holding vessel control
Service data
Sales data
Modem
Sales data
Drink recipes
Sales log

Level 1 DFD Operation Control



UML – Unified Modelling Language

The Unified Modelling Language (UML)

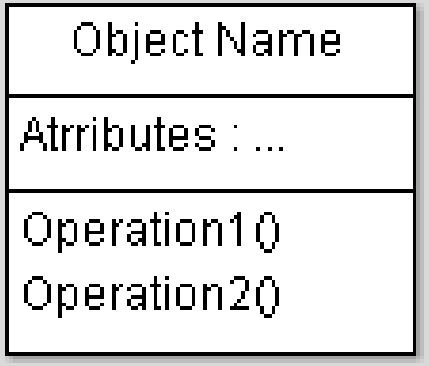
- Devised by the developers of widely used object-oriented analysis and design methods
- Has become an effective standard for **object-oriented modelling**
- UML is:
  - Expressive
  - easy to use
  - Unambiguous
  - well supported by a variety of proprietary and FOSS CASE tools

The Unified Modelling Language

- The unified modelling language contains many different types of diagram, for example:
  - Use-case diagrams (Lecture 7)
  - Sequence diagrams (Lecture 7)
  - Class diagrams (Today)
  - Statechart diagrams (Today)

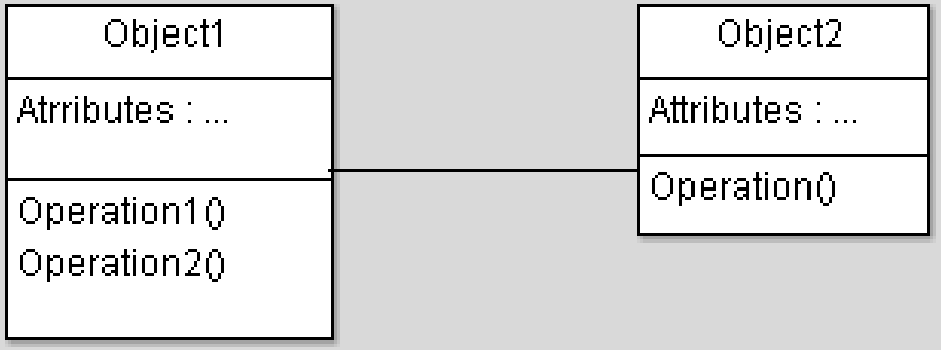
UML Notation

- Object classes are rectangles with the **name** at the top, **attributes** in the middle section and **operations** in the bottom section



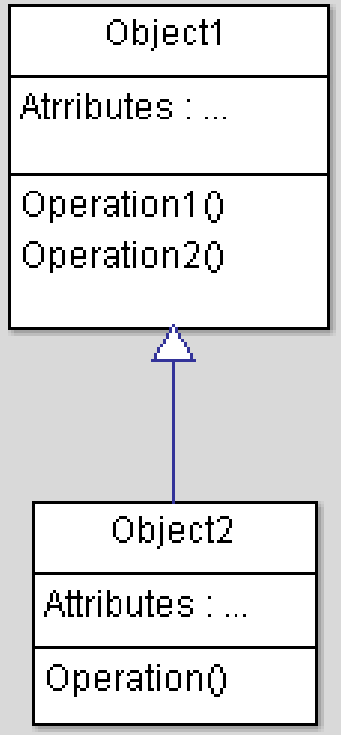
UML Notation

- Relationships between object classes (known as associations) are shown as lines linking objects



UML Notation

- Inheritance is referred to as **generalisation** and is shown ‘upwards’ rather than ‘downwards’ in a hierarchy



UML

- UML is important in modelling the systems and software products that we design
- UML is used (and abused) in many different types of diagrams
- UML diagrams often form a core part of a software specification
- **We study UML in more detail later in the module**

Statechart Diagram

Statechart Diagram

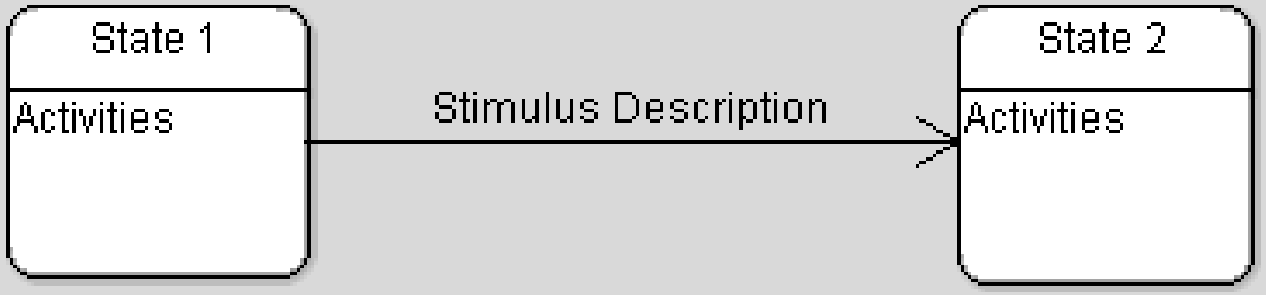
- Statechart Diagrams (or *State machine models*) show the behaviour of the system in response to external and internal events
- They show the system's responses to stimuli (the event-action paradigm) so are often used for modelling **real-time systems**
- We use a level of abstraction so that we can observe the *essential behaviour* of the system we want to model.
- Show **system states** as nodes
- Show **events** as arcs between these nodes.
- When an event occurs, the system moves from one state to another
- Statecharts are an integral part of the **Unified Modeling Language** (UML)

Statechart Diagram Notation

- **States** are denoted by rounded rectangles, containing:
  - the **state name**
  - brief description of the **action performed** in that state (optional: prefix with “Do:”)
  - Optional **Exit action** can be included (Prefix with “Exit:”)

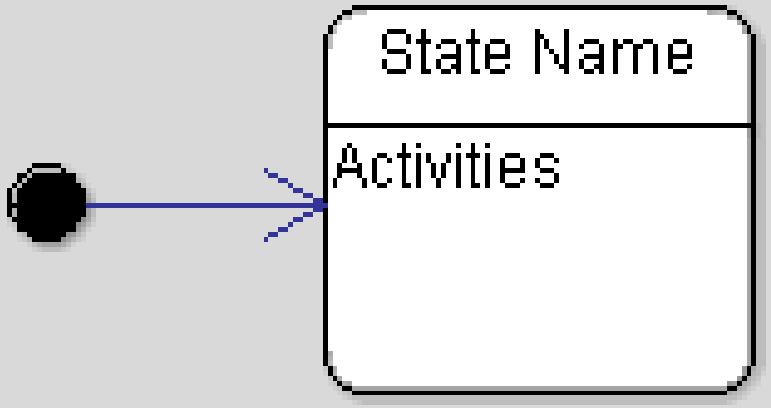
Statechart Diagram Notation

- The system **transitions** between states
  - Transitions are represented as **arcs** (directed arrows) between states
  - The **event** that causes the transition is given as a label
    - Method
    - Stimulus, etc…
  - **Unlabelled** transitions will always happen
  - A supplementary **description table** can be included to describe states, stimuli, events…



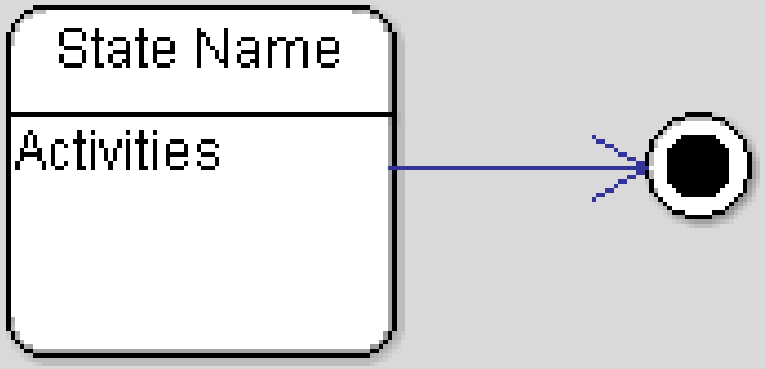
Statechart Diagram Notation

- An (optional) **initial state** is denoted by a solid circle
  - sometimes the system will start in different places and thus the initial state should be omitted.



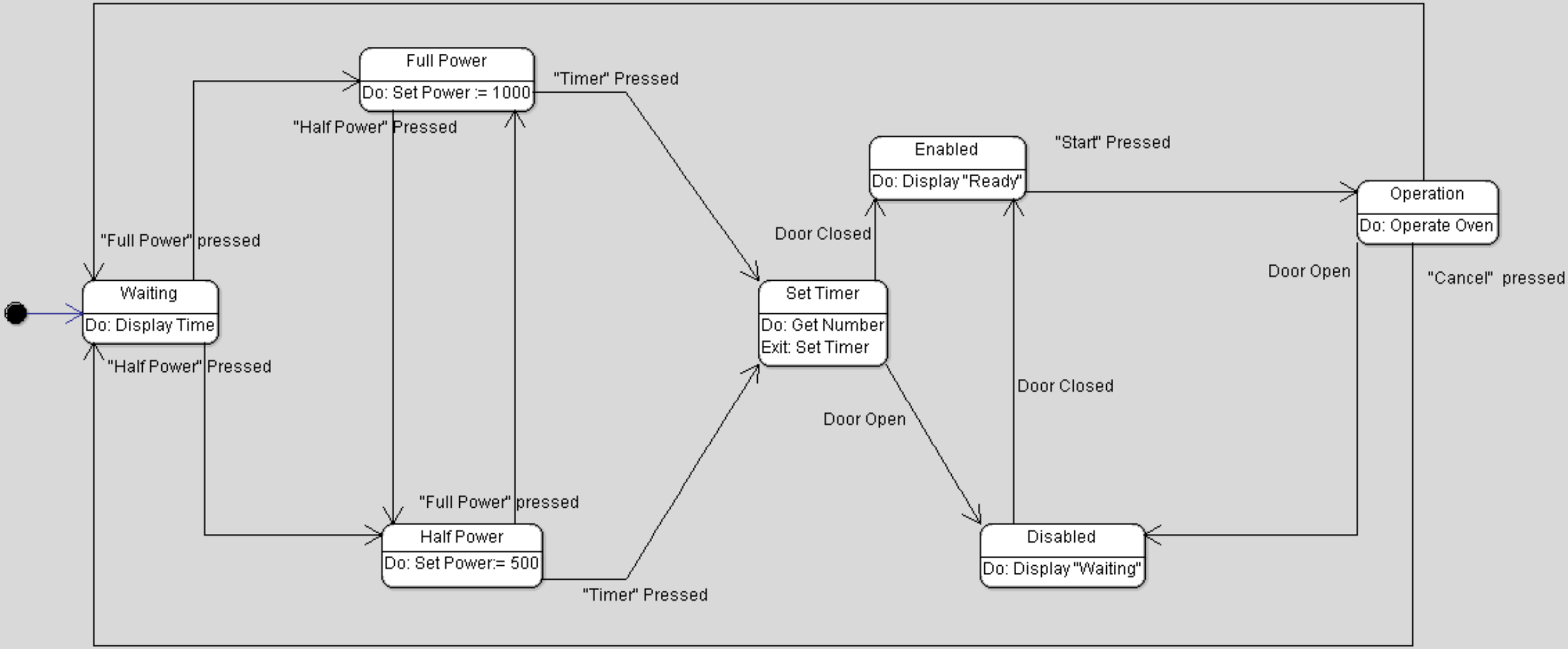
Statechart Diagrams Notation

- An (optional) **final state** is denoted by a solid circle with a ring around it.



Example - Microwave Oven Model

A state machine model does not show flow of data within the system



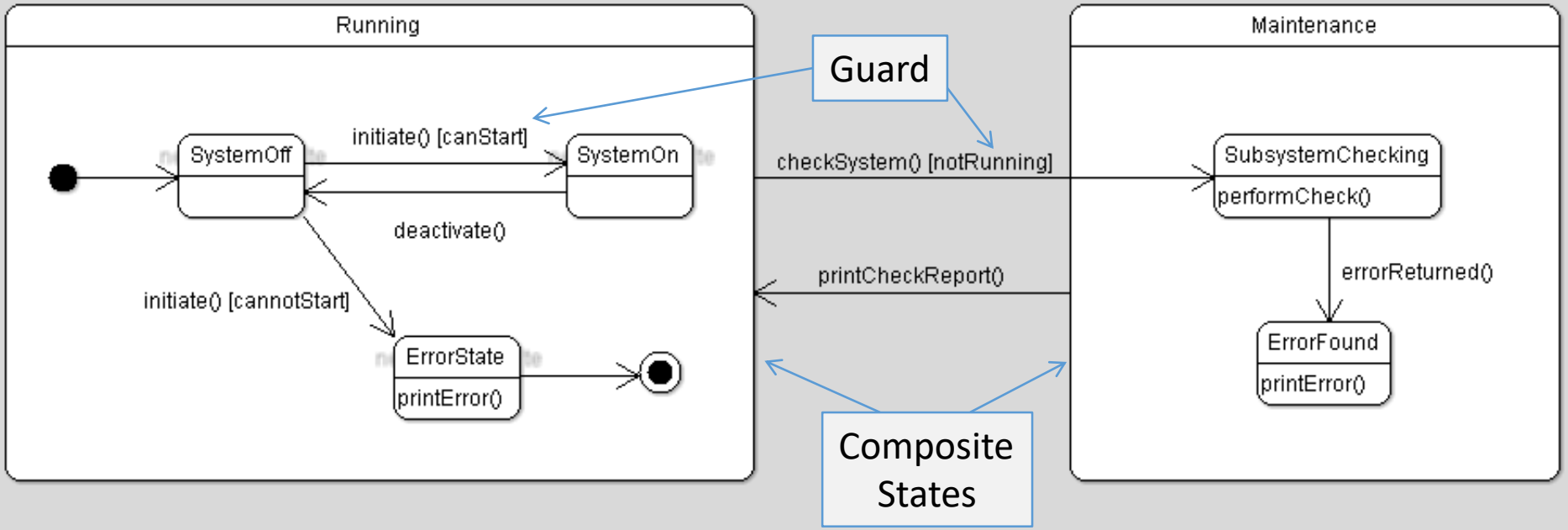
Q: Why is there no final state?

Microwave Oven Stimuli Descriptions

Stimulus	Description
Half power pressed	The user has pressed the half power button
Full power Pressed	The user has pressed the full power button
Timer pressed	The user has pressed one of the timer buttons
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start pressed	The user has pressed the start button
Cancel pressed	The user has pressed the cancel button

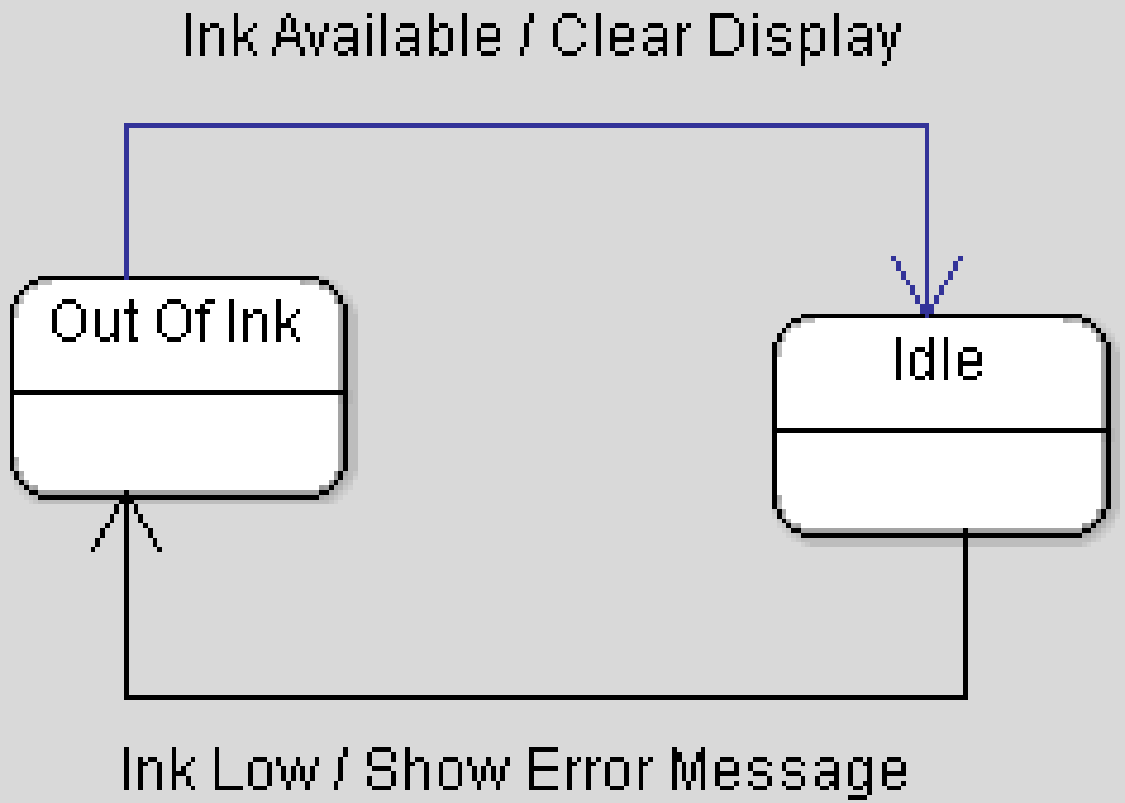
Advanced Features of Statechart Diagrams

- A **guard** ensures that the system only moves from one state to the other if the expression is satisfied.
- A state can contain a **subdiagram/composite state**. Useful when we wish to model a subsystem or substate.



Advanced Features of Statechart Diagrams

- You can put **actions after the event** using a /

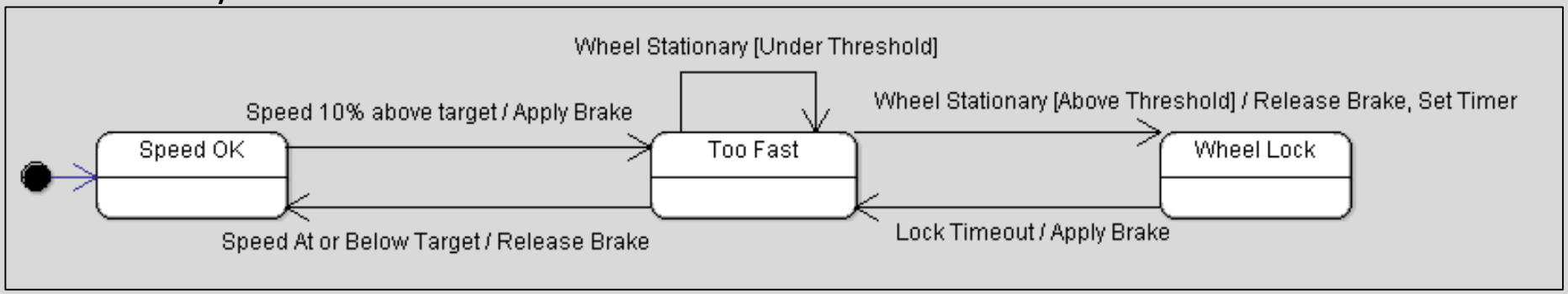


Hints For Statechart Diagrams

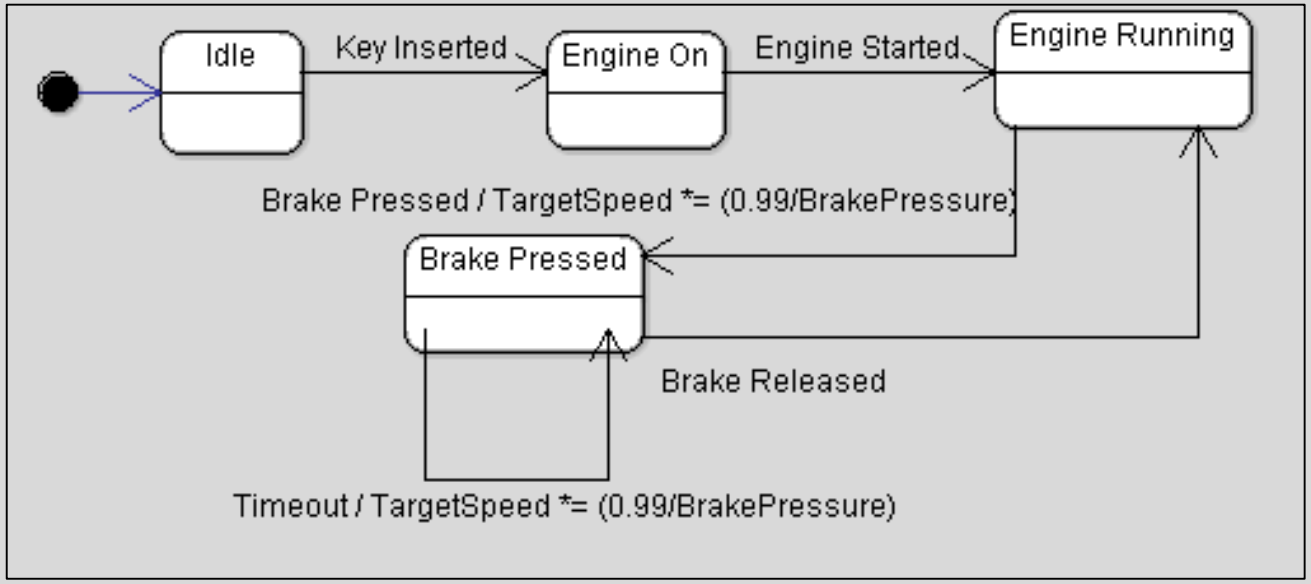
- Often have an Idle state where the process is not active
- All states need some exit (no deadlock, even in error conditions)
- Use multiple state charts to keep the design simple
  - Do NOT need to have a state chart as sub state of other state chart
  - System can be described by multiple state machines running concurrently

Multiple Statechart Diagrams Example

Auto Brake System



Dash Control

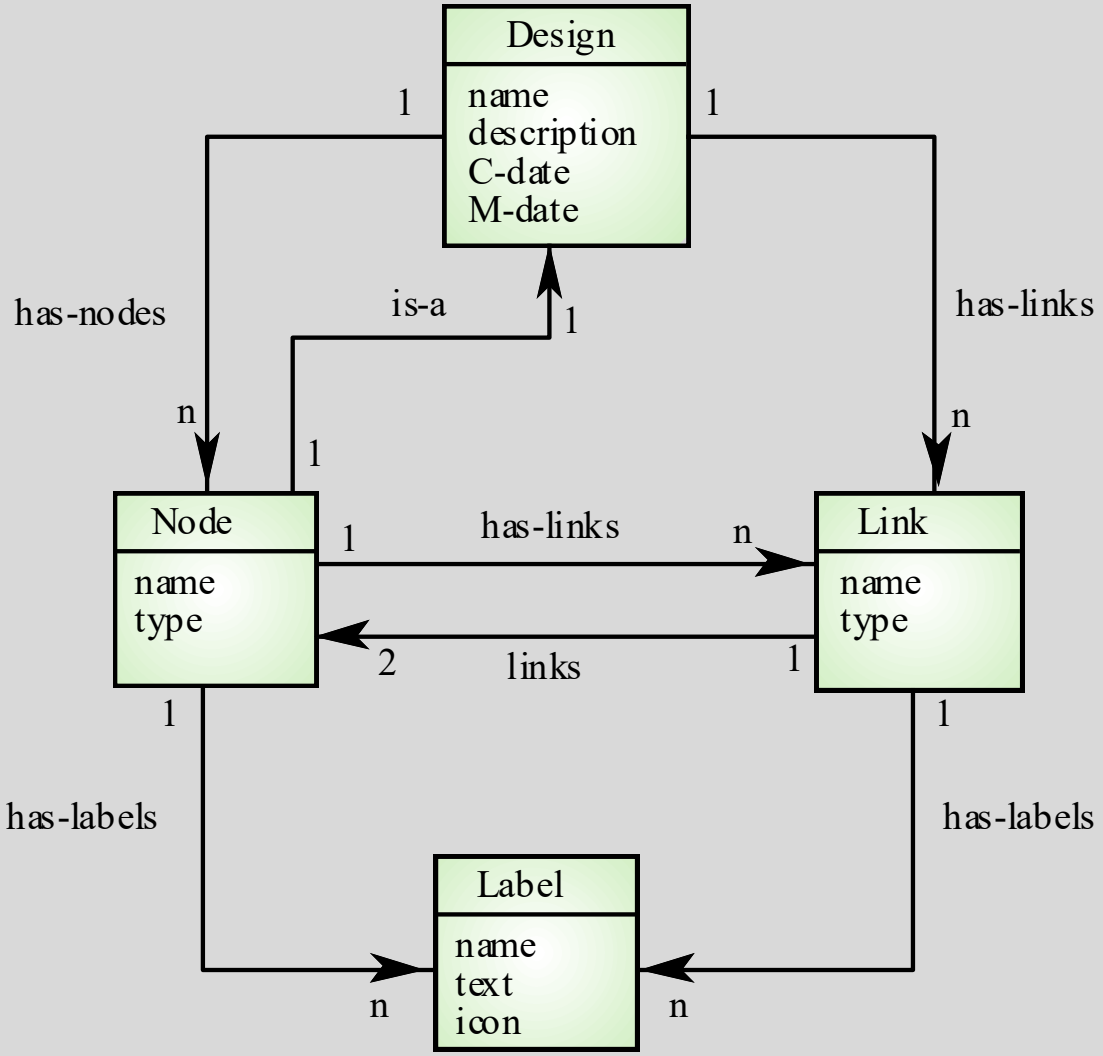


Semantic Data Models

Semantic Data Models

- Used to describe the **logical structure of data** processed by the system
- **Entity-relation-attribute model** sets out the entities in the system, the relationships between these entities, and the entity attributes
- **Widely used in database design.** Can readily be implemented using relational databases
- **No specific notation provided in the UML** but objects and associations can be used
  - Class diagrams show some of the same ideas (More later in the module)

Software Design Semantic Model



Data Dictionary

Data Dictionary

- **A Data dictionary** is a list of all of the names used in the system models.
- Descriptions of the entities, relationships and attributes are also included
- It may be used for **name-management**
  - names used in a system should be consistent, not clashing
- It serves as a central repository of organisational information

Data Dictionary Example

Name	Description	Type	Date
has-labels	1:N relation between entities of type Node or Link and entities of type Label.	Relation	5.10.1998
Label	Holds structured or unstructured information about nodes or links. Labels are represented by an icon (which can be a transparent box) and associated text.	Entity	8.12.1998
Link	A 1:1 relation between design entities represented as nodes. Links are typed and may be named.	Relation	8.12.1998
name (label)	Each label has a name which identifies the type of label. The name must be unique within the set of label types used in a design.	Attribute	8.12.1998
name (node)	Each node has a name which must be unique within a design. The name may be up to 64 characters long.	Attribute	15.11.1998

Recap

Recap

- Data Flow Diagrams
  - Show how data flows through the system
  - Can be shown to customers to help validate the system
  - Developed in a top-down manner
- UML
  - Many diagram types are possible with UML
  - Common notation between diagram types
- Statechart Diagrams
  - A UML diagram type
  - Shows systems behaviour – response to particular stimuli or events
- Semantic Data Models
  - Shows the logical structure of data in system, used in database design
- Data Dictionaries
  - Help data terms to remain consistent – an organisational repository of information – a glossary