L10 - Requirements analysis dynamic model

CS3028 - Principles of Software Engineering

Ernesto Compatangelo

Department of Computing Science

UNIVERSITY OF ABERDEEN

10.1 Reminding past issues and mapping them to current topics

Where are we now?

Software development paradigms

⇒ The Unified Process (UP) paradigm

⇒ UP phases and UP disciplines (activities) within each phase

⇒ Inception (first UP phase)

⇒ Elaboration (second UP phase)

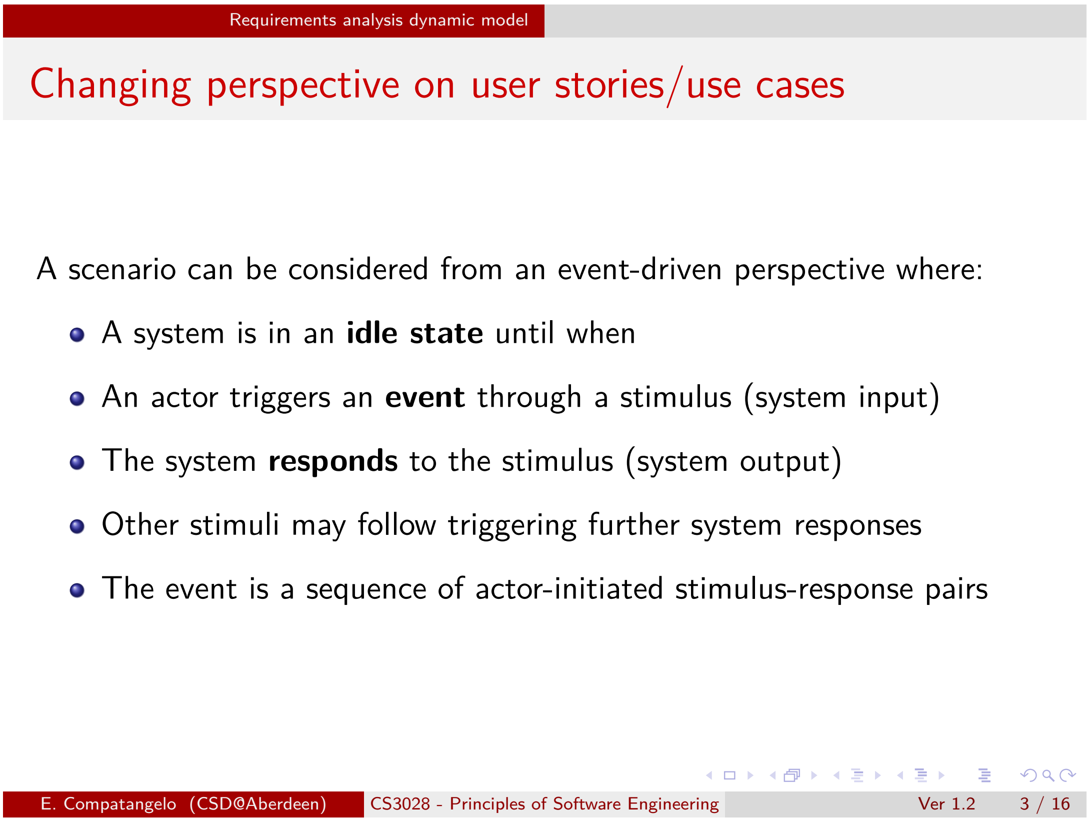
⇒ Moving from inception to elaboration

⇒ Functional model, structural model

⇒ Dynamic models

⇒ · · · · · ·

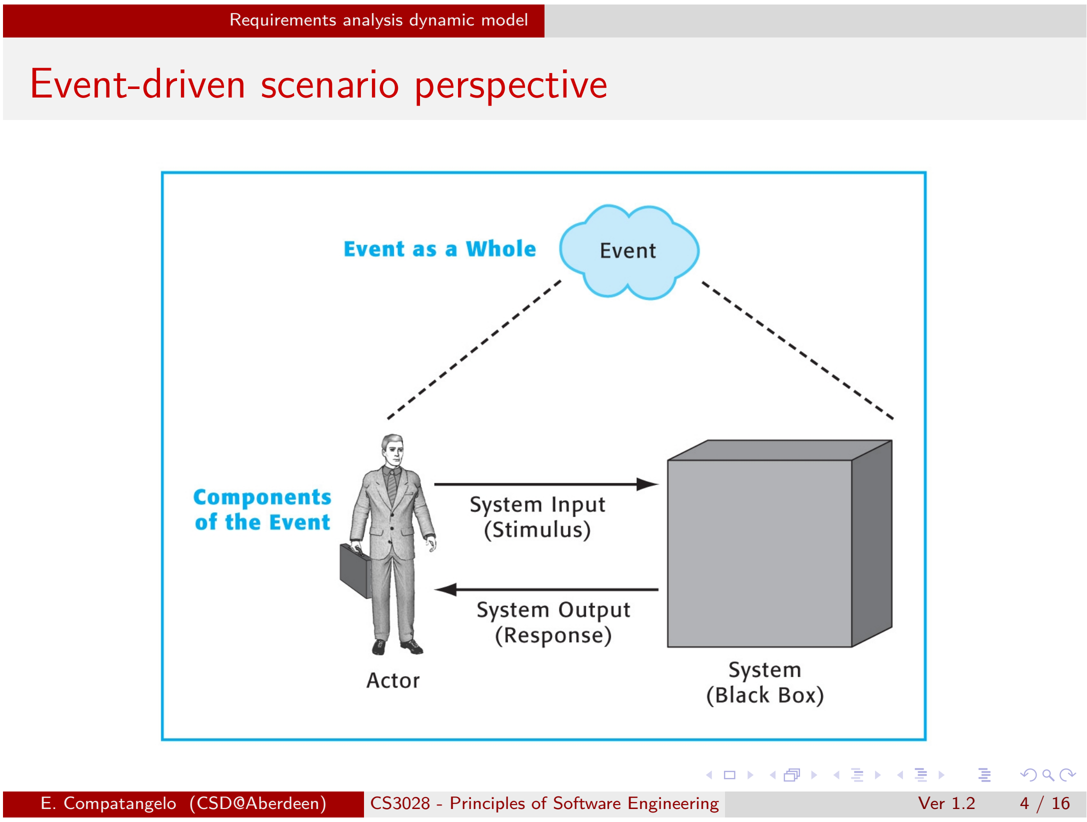
10.2 Event-based requirements modelling



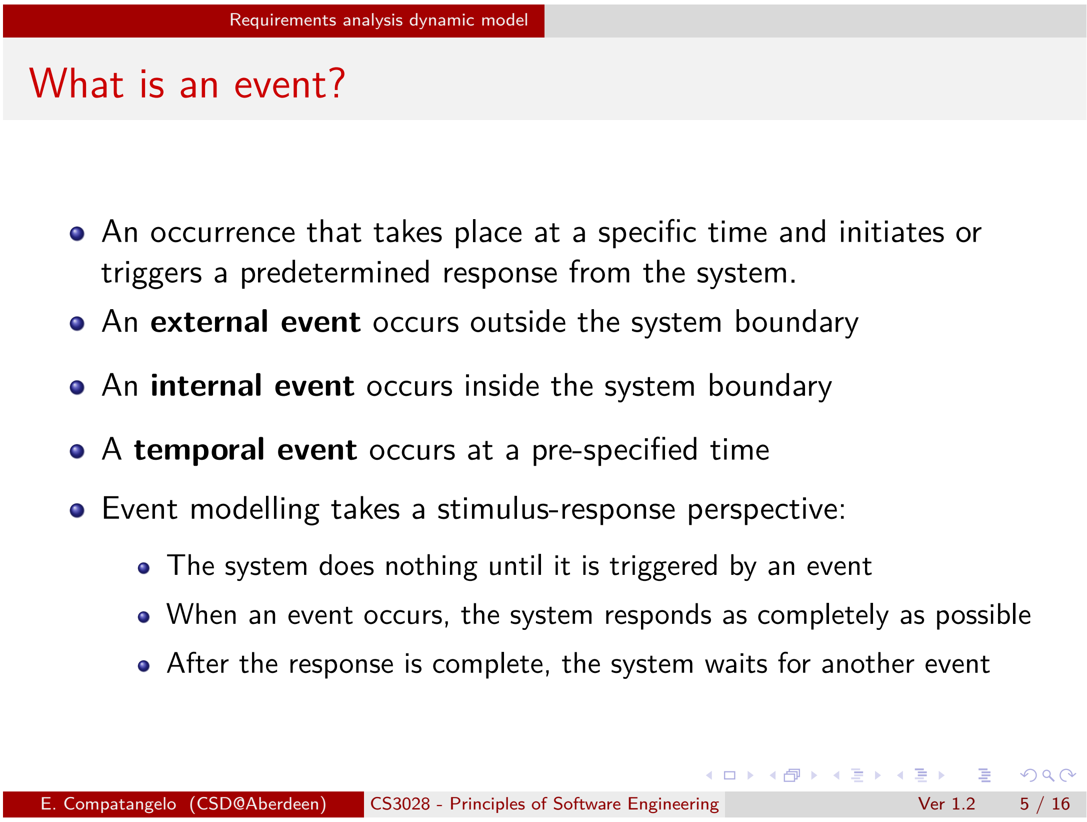
Changing perspective on user stories/use cases

A scenario can be considered from an event-driven perspective where:

- A system is in an **idle state** until when
- An actor triggers an **event** through a stimulus (system input)
- The system **responds** to the stimulus (system output)
- Other stimuli may follow triggering further system responses
- The event is a sequence of actor-initiated stimulus-response pairs



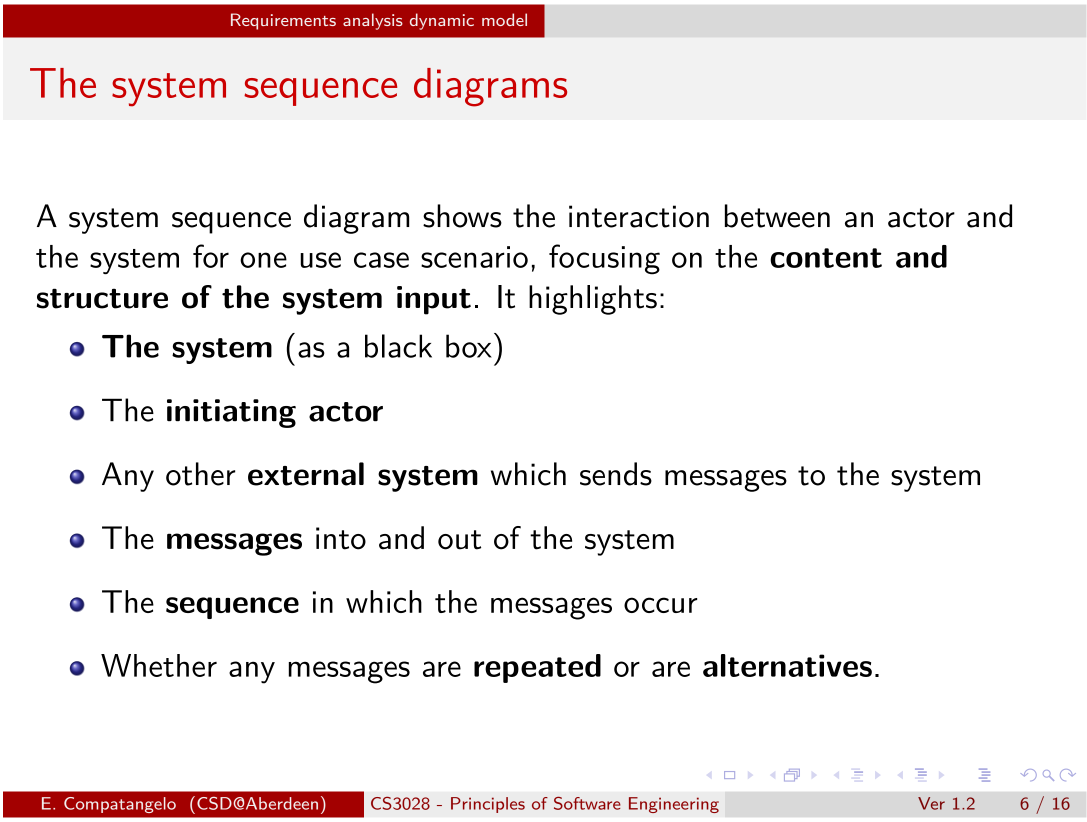
Event-driven scenario perspective



What is an event?

- An occurrence that takes place at a specific time and initiates or triggers a predetermined response from the system.
- An **external event** occurs outside the system boundary
- An **internal event** occurs inside the system boundary
- A **temporal event** occurs at a pre-specified time
- Event modelling takes a stimulus-response perspective:
    - The system does nothing until it is triggered by an event
    - When an event occurs, the system responds as completely as possible
    - After the response is complete, the system waits for another event

10.3 System sequence diagrams

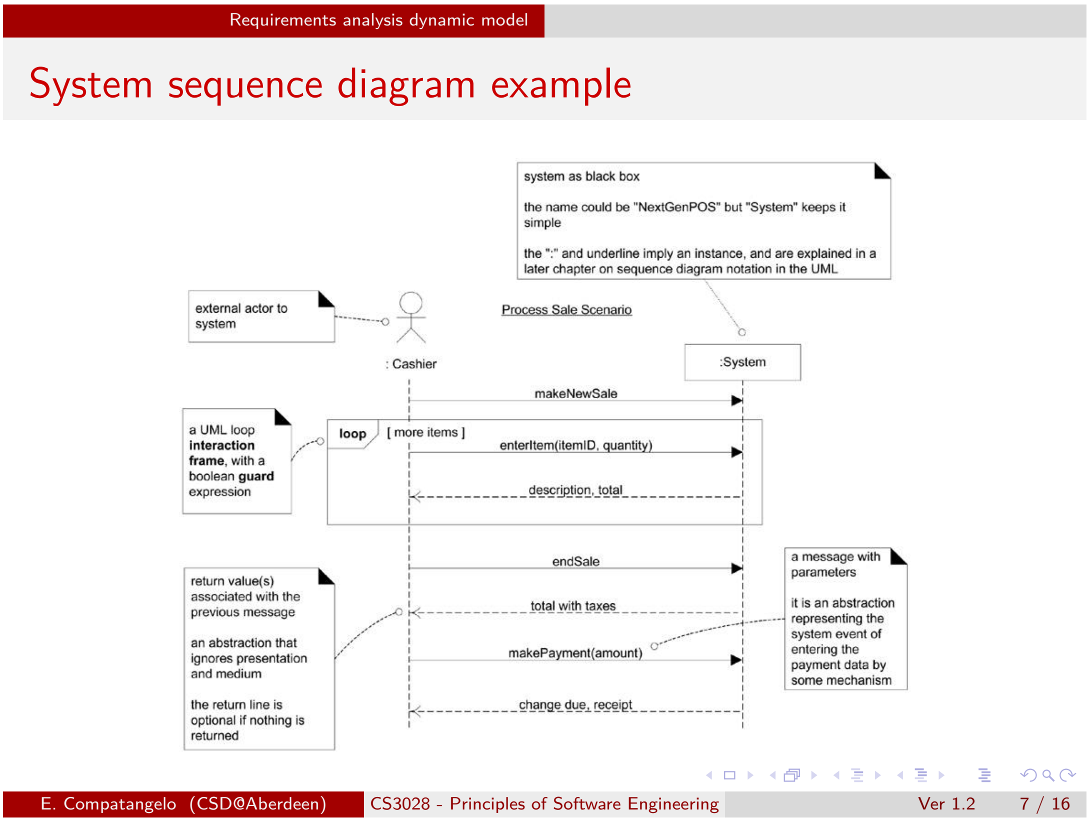


The system sequence diagrams

A system sequence diagram shows the interaction between an actor and the system for one use case scenario, focusing on the **content and structure of the system input**. It highlights:

- **The system** (as a black box)
- The **initiating actor**
- Any other **external system** which sends messages to the system
- The **messages** into and out of the system
- The **sequence** in which the messages occur
- Whether any messages are **repeated** or are **alternatives**.

- Event-driven analysis shows events coming in to our system
- Sooner or later, we have to design the software to handle these events (from the mouse, keyboard, another system) and execute a response
- Hence, it is useful to know precisely the external input events (namely, the **system events**)
- However, modelling system behavior during inception (*i.e.*, during requirements analysis) provides a description of what a system does, **without explaining how it does it** (*i.e.*, as a **Black Box**)



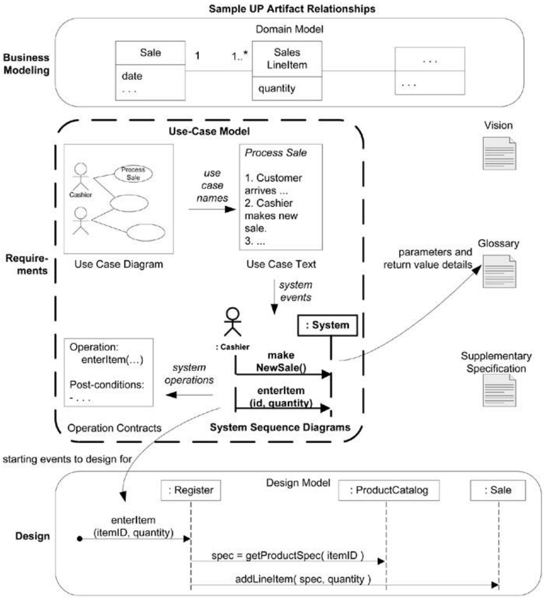
Requirements analysis dynamic model

System sequence diagram example

system as black box
the name could be "NextGenPOS" but "System" keeps it simple
the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML
external actor to system
Process Sale Scenario
: Cashier
:System
makeNewSale
a UML loop **interaction frame**, with a boolean **guard** expression
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale
a message with parameters
it is an abstraction representing the system event of entering the payment data by some mechanism
return value(s) associated with the previous message
an abstraction that ignores presentation and medium
the return line is optional if nothing is returned
total with taxes
makePayment(amount)
change due, receipt

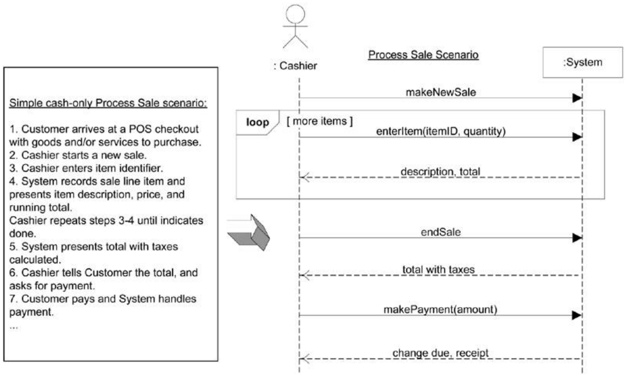
E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 7 / 16

Relationships between SSDs and other UP artifacts



Relationships between SSDs and use cases

An SSD shows system events for one scenario of a use case, therefore it is generated from the inspection of a use case



Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.
Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
...

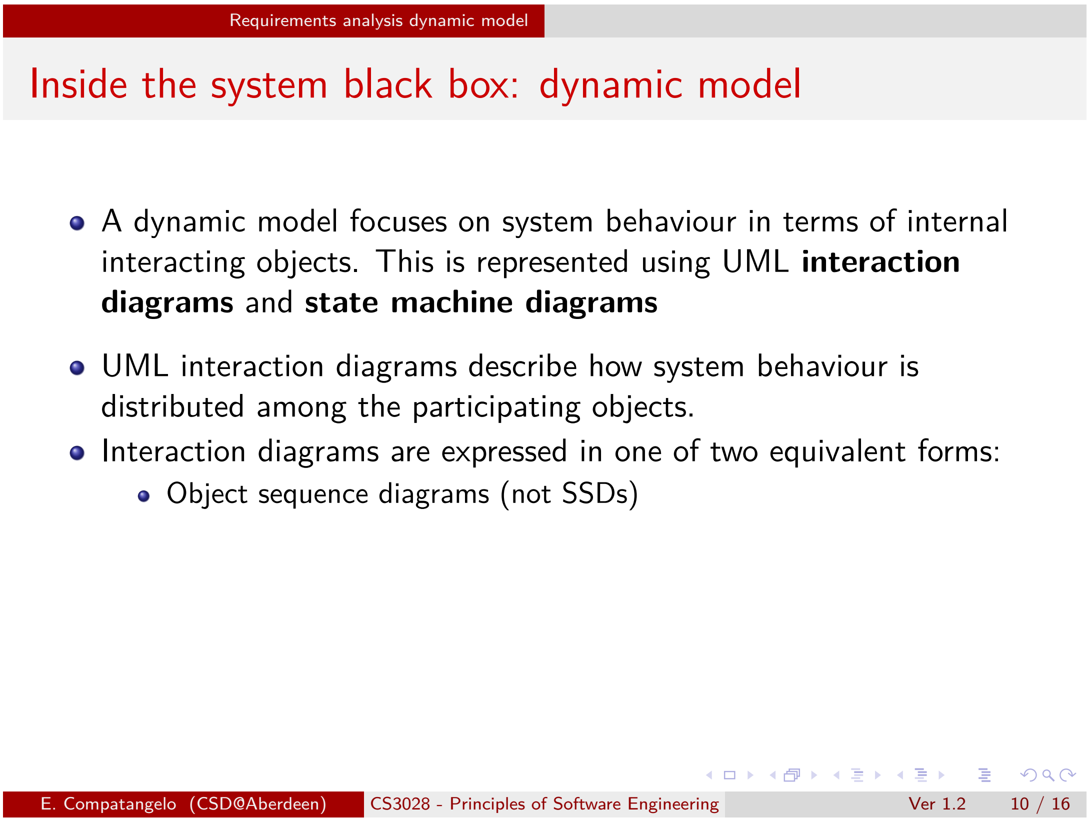
10.3.1 Message formats in SSDs

- The message consists of a message name followed (in parentheses) by a parameter list
- All names begin with a lower-case letter
- There are no spaces in a name
- Upper-case letters separate the words within a name
- Names in the parameter list are separated by commas

10.4 Outgoing messages (System Outputs)

- There are two types of outgoing messages:
  - A **response** of the system which **completes an event**
  - A **message** from the system to an *external system* **requesting action and a reply**
- Remember that **every output** must be derivable from the **input to the use case** combined with **stored data**

10.5 Dynamic models



Requirements analysis dynamic model

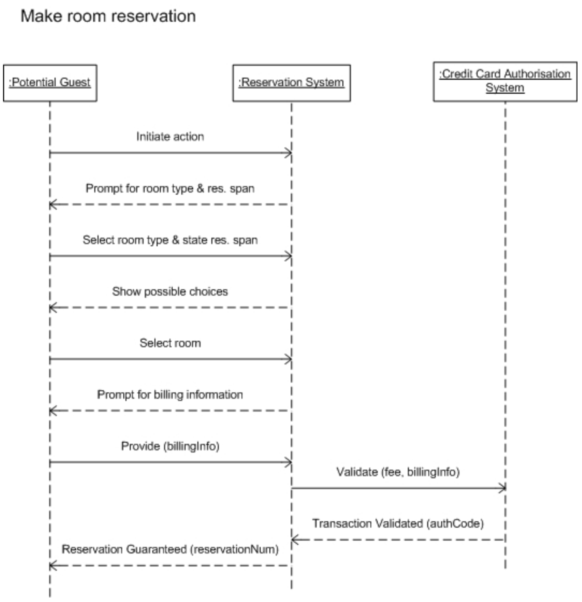
Inside the system black box: dynamic model

- A dynamic model focuses on system behaviour in terms of internal interacting objects. This is represented using UML **interaction diagrams** and **state machine diagrams**
- UML interaction diagrams describe how system behaviour is distributed among the participating objects.
- Interaction diagrams are expressed in one of two equivalent forms:
  - Object sequence diagrams (not SSDs)

E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 10 / 16

10.6 UML sequence diagrams

UML “ internal” object sequence diagram example


Make room reservation
:Potential Guest
:Reservation System
:Credit Card Authorisation System
Initiate action
Prompt for room type & res. span
Select room type & state res. span
Show possible choices
Select room
Prompt for billing information
Provide (billingInfo)
Validate (fee, billingInfo)
Transaction Validated (authCode)
Reservation Guaranteed (reservationNum)

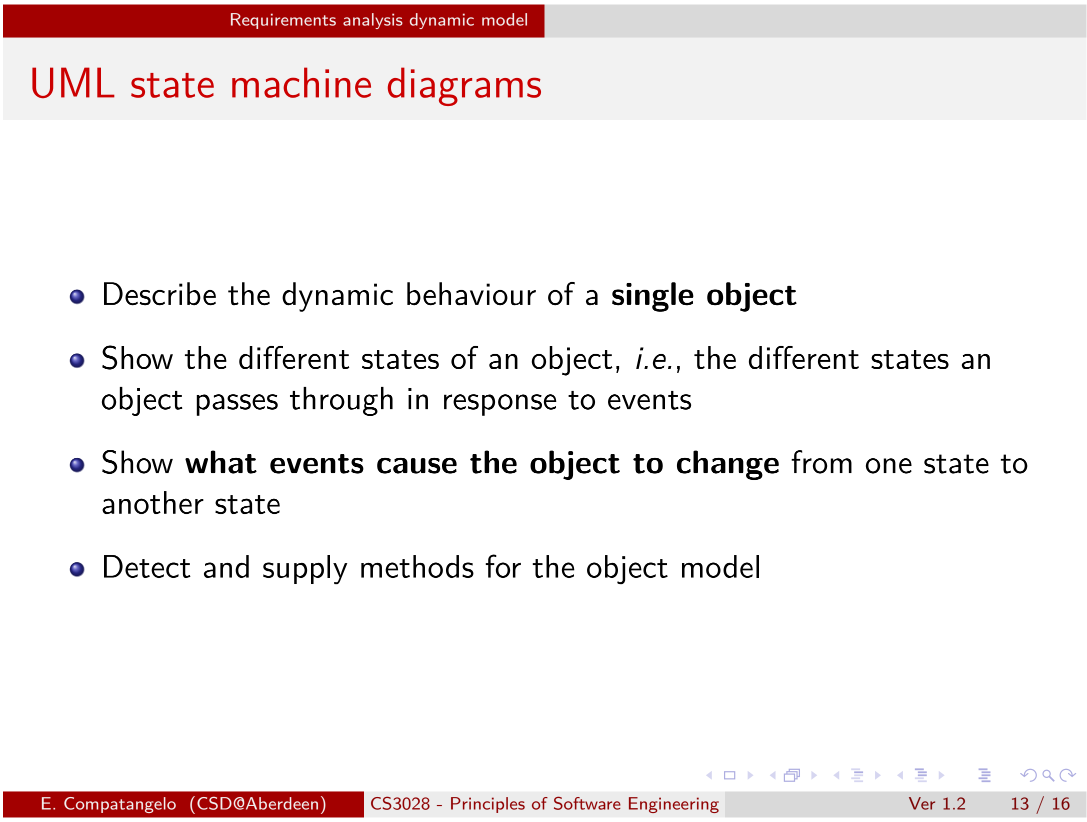
Components of a sequence diagram

- At system level, instances are represented by actor instances and system activation instances. However, at object level class instances (*i.e.* objects) are also shown.
- Lifelines provide the ‘starting point’ and the ‘ending point’ in time for each instance
- Arrows from left to right (thick line) denote messages issued by an actor instance or by a use case activation
- Arrows from right to left (dashed line) denote answers issued by a system or use case activation and triggered by messages
- Time flows from the actor instances/system activations (top of the page) towards the bottom of the page

10.6.1 Building a sequence diagram

- Determine the context of the sequence diagram
- Identify the participating objects
- Set the lifeline for each object
- Add messages
- Place the focus of control on each object's lifeline
- Validate the sequence diagram

10.7 UML state machine diagrams

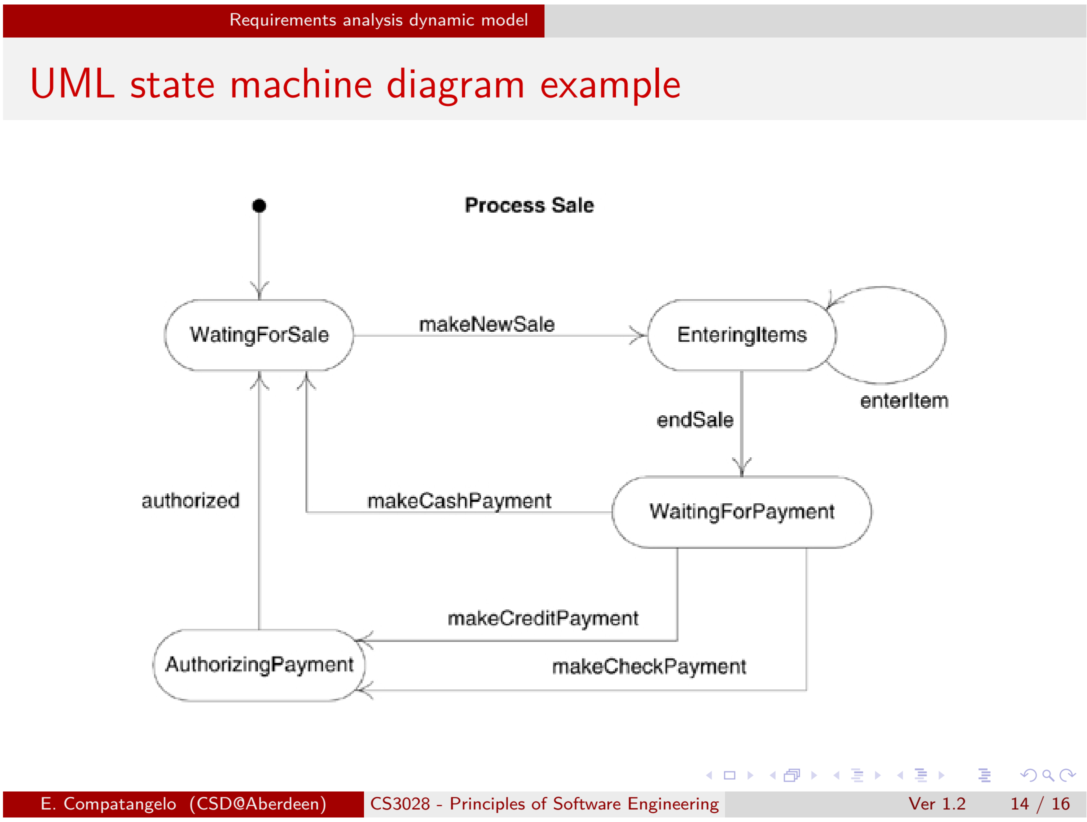


Requirements analysis dynamic model

UML state machine diagrams

- Describe the dynamic behaviour of a **single object**
- Show the different states of an object, *i.e.*, the different states an object passes through in response to events
- Show **what events cause the object to change** from one state to another state
- Detect and supply methods for the object model

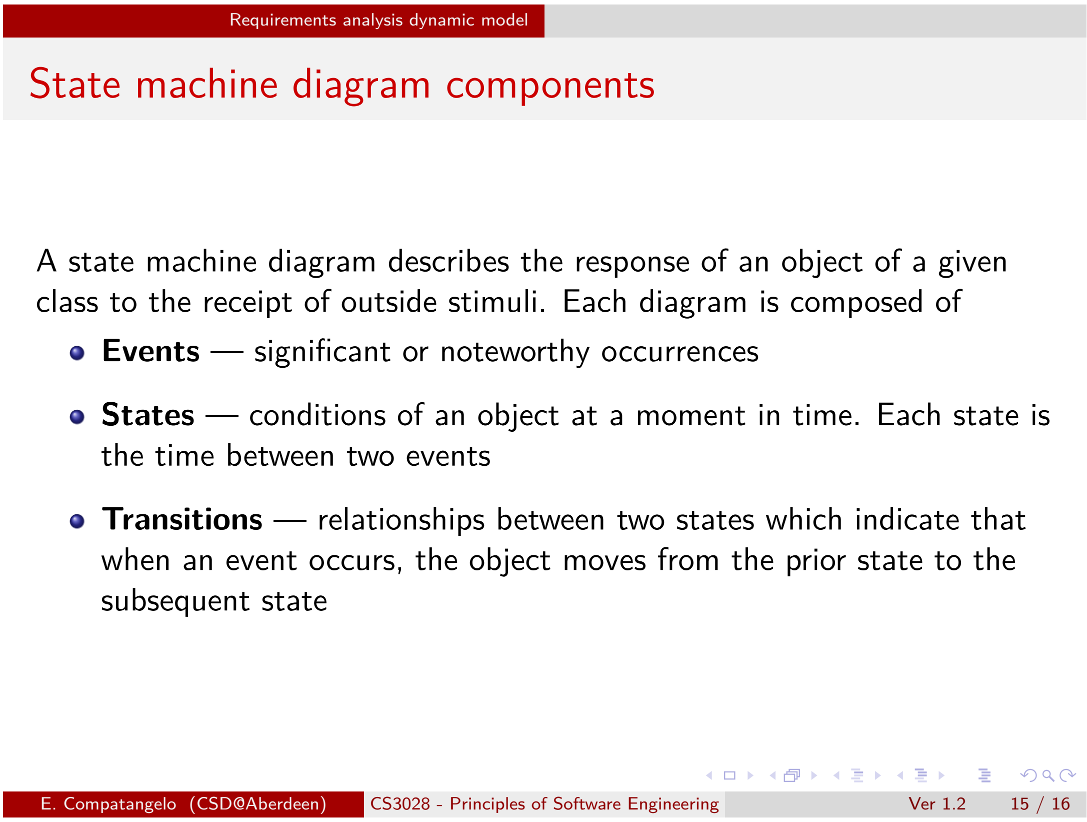
E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 13 / 16



Requirements analysis dynamic model

UML state machine diagram example

E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 14 / 16



Requirements analysis dynamic model

State machine diagram components

A state machine diagram describes the response of an object of a given class to the receipt of outside stimuli. Each diagram is composed of

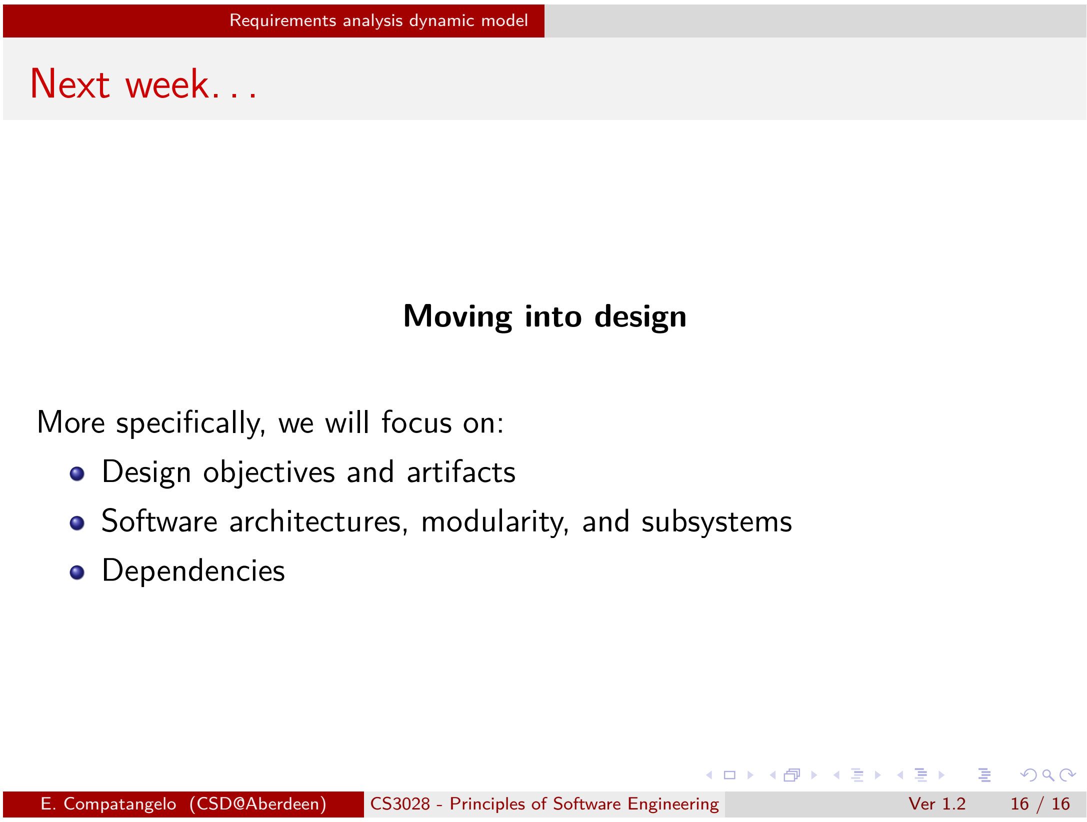
- **Events** — significant or noteworthy occurrences
- **States** — conditions of an object at a moment in time. Each state is the time between two events
- **Transitions** — relationships between two states which indicate that when an event occurs, the object moves from the prior state to the subsequent state

E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 15 / 16

Building a UML state machine diagram

- Set the context
- Identify the initial, final, and stable states of the object
- Determine the order in which the object will pass through stable states
- Identify the events, actions, and guard conditions associated with the transitions
- Validate the state machine diagram

10.8 Preparing for the topic ahead



Requirements analysis dynamic model

Next week. . .

Moving into design

More specifically, we will focus on:

- Design objectives and artifacts
- Software architectures, modularity, and subsystems
- Dependencies

E. Compatangelo (CSD@Aberdeen) CS3028 - Principles of Software Engineering Ver 1.2 16 / 16