Avancier Methods (AM)

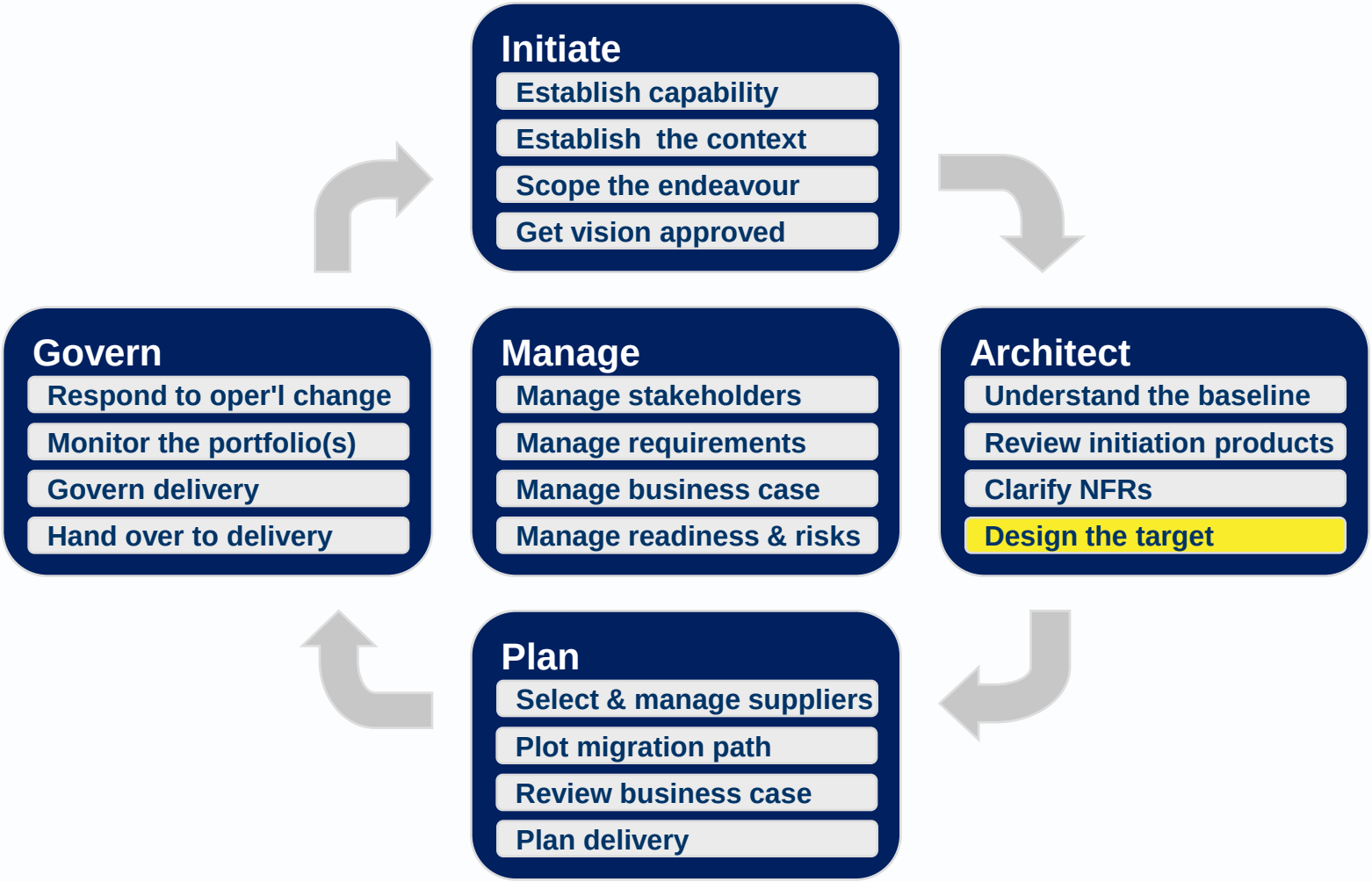
Applications Architecture

Define Application Use Cases / Epics

CONTEXT

- What is the AM level 2 process?
- Which domain are we working in?
- What is the AM level 3 process?

AM level 2 process



Which domain are we working in?

Avancier

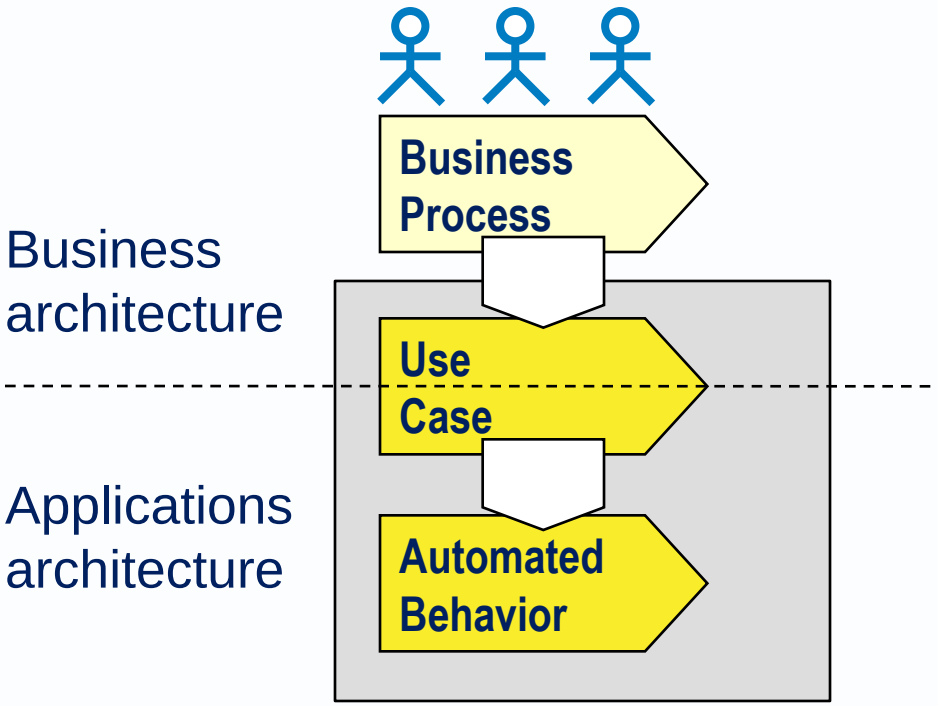
	Passive Structure	*Required Behaviour*	*Logical Structure*	*Physical Structure*
Business		Business Service Business Process	Function Role	Org Unit Actor
Data / Information	Data Entity	Data Flow	Log Data Model	Data Store
Applications		**App Use Case**	**Application Interface**	**Application**
Infrastructure Technology		Platform Service	Platform Interface	Platform Applicat'n

AM level 3/4 process: Design application use cases

Avancier

1. Design application uses cases
    1. Identify use cases
    2. Draw use case diagram
    3. Describe use cases
    4. Identify automated services

The process automation hierarchy



Use case description

Avancier

Service	Name: Buy train ticket
	Inputs: Journey facts
	Outputs or products: Seat reservation
Process flow	1. Identify journey start and end stations 2. Identify outbound start time 3. Identify inbound start time 4. Identify traveller numbers and ages 5. Review buying details 6. Enter payment details 7. Create personal account (optional) 8. Confirm payment details 9. Collect receipt
Non-functionals	Response time
	Throughput
	Availability
	etc

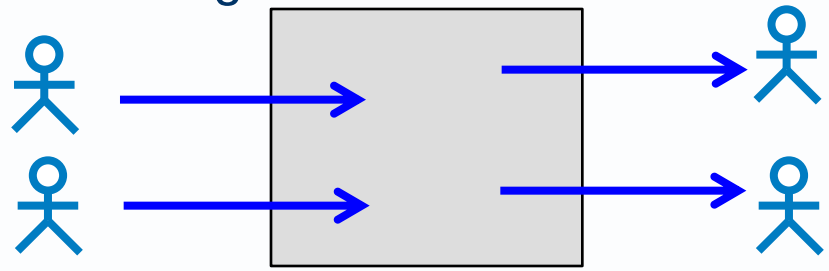
Service "signature"

Each step could be a user story

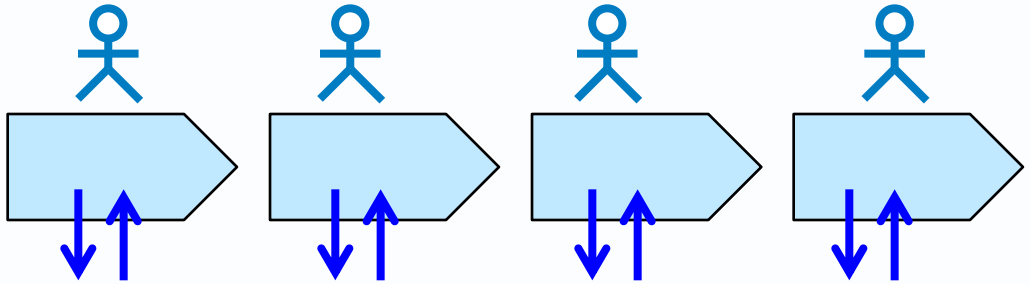
Don't forget the numbers

Define the business context for data creation and use

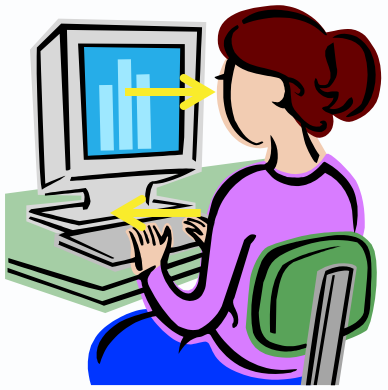
- Context diagram



- Value stream / scenario diagrams (showing OPOPOT activities)

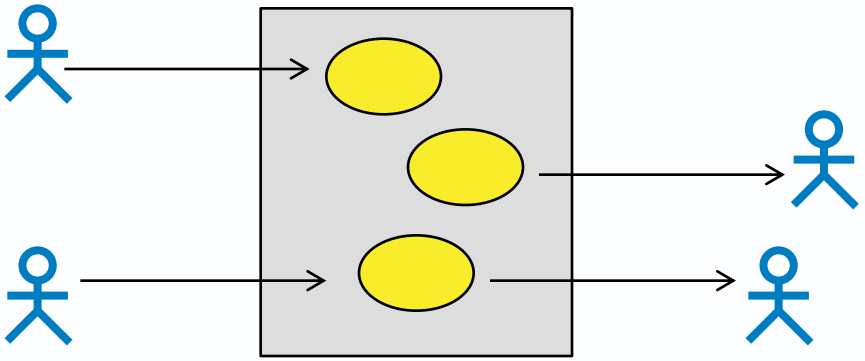


- Client devices and user interfaces

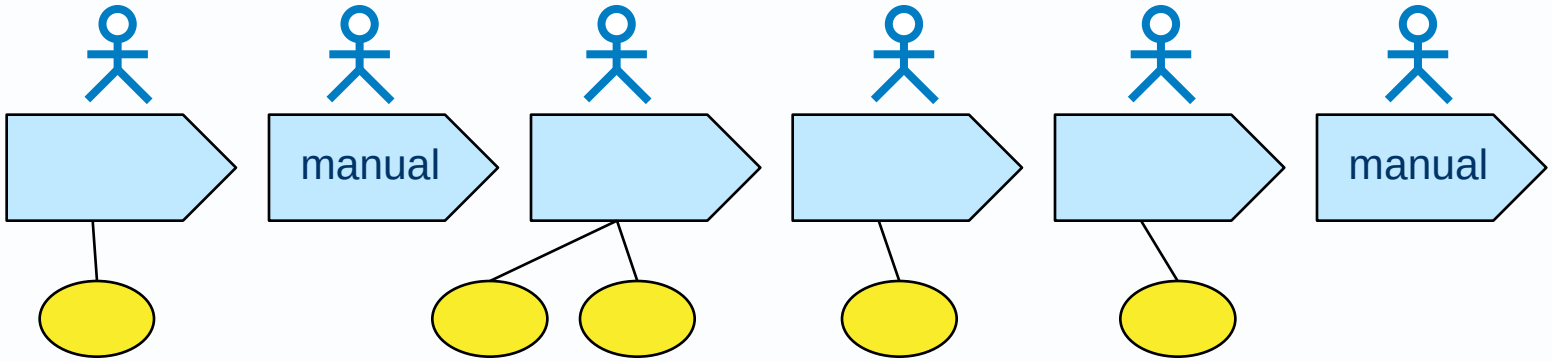


Identify use cases

- Uses cases needed to
  - capture input data
  - provide output data



- Use cases needed to
  - support or enable activities in a business process



Design IS (application) services (AM for SA level 4)

1. Identify use cases
2. **Draw use case diagram**
3. Describe use cases
4. Identify automated services

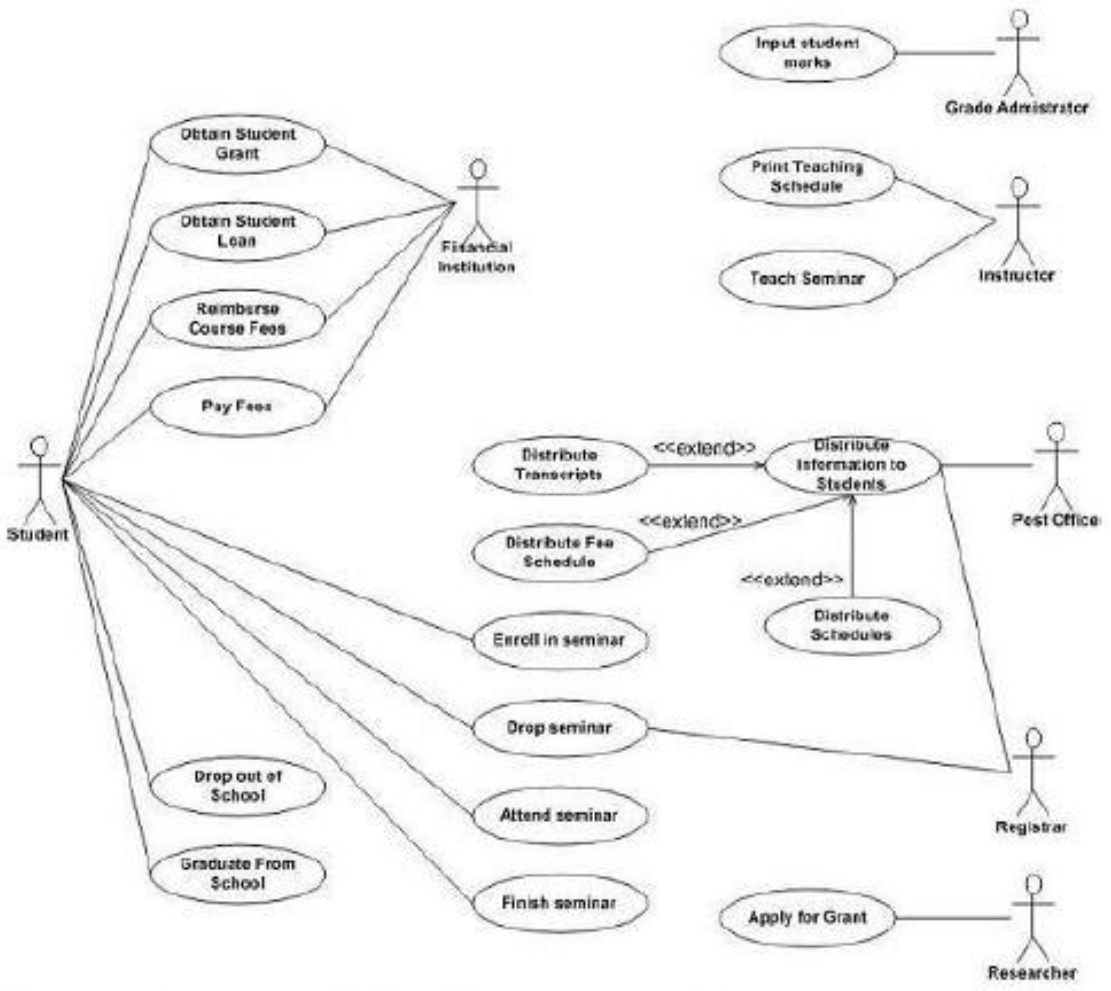
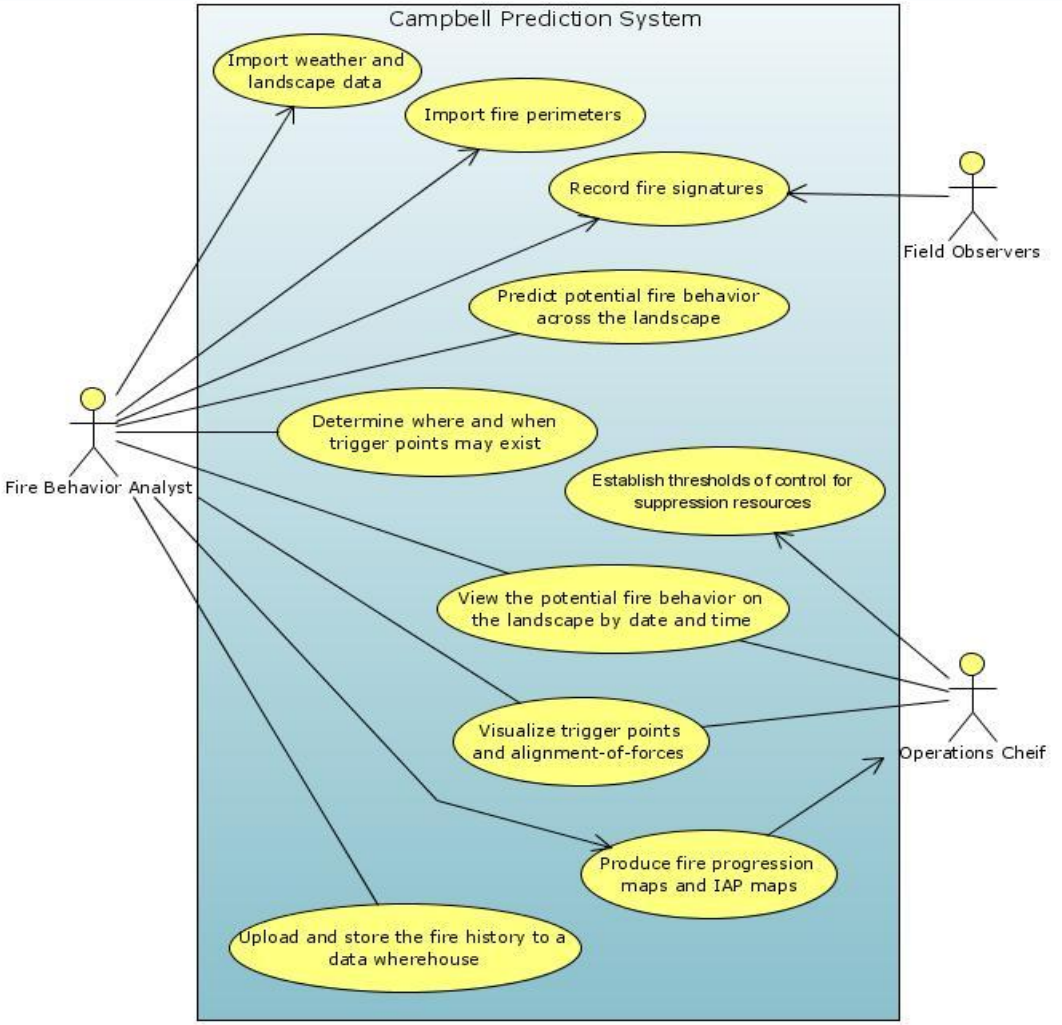


Figure 3-17, Users represented in UML's Use case Diagram

Use case diagram (an application scoping tool)

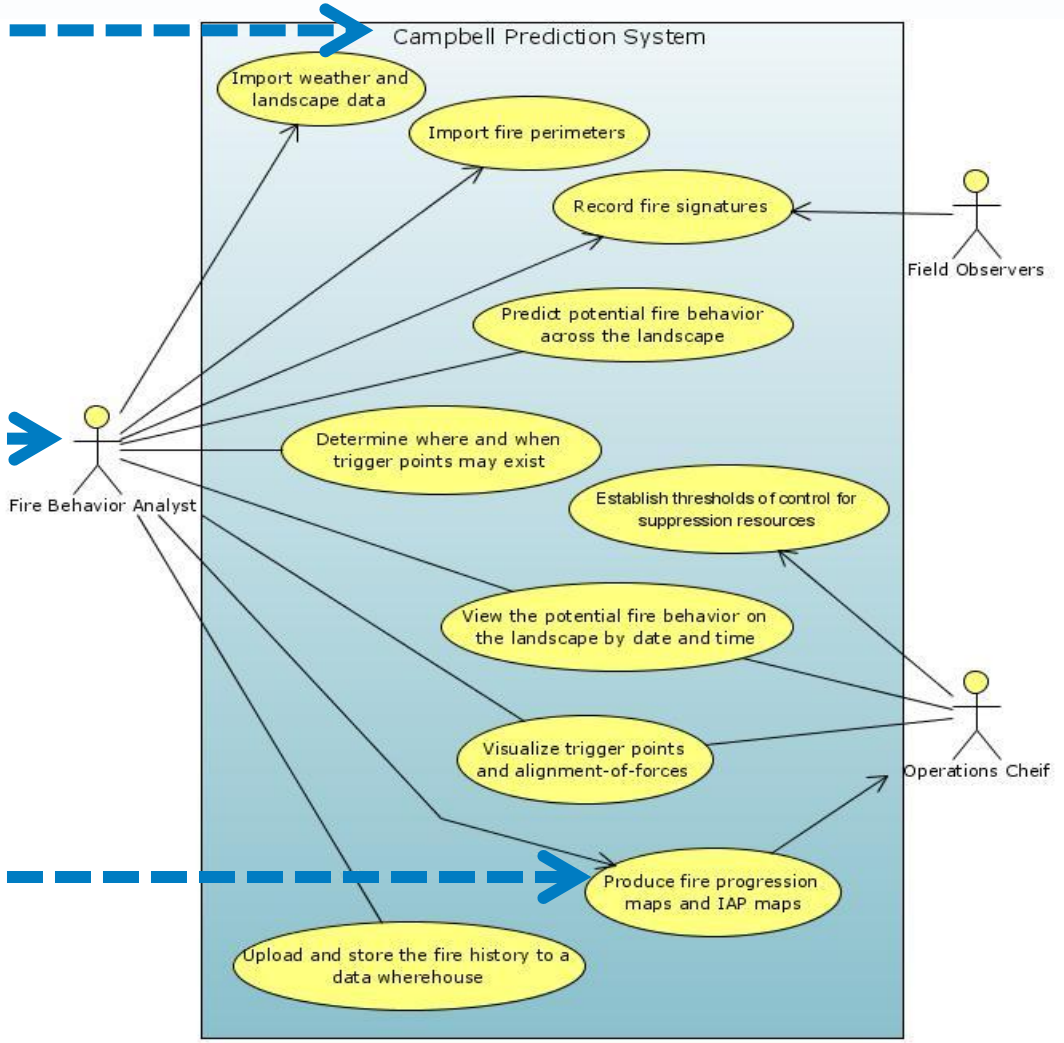
- Shows **uses** that **actors (roles)** can make of a **system**
- A use case is a **process** that ends with a result or service of value



https://kenai.com/attachments/wiki_images/cps/CPS_Main_Use_Case_Diagram_v2.jpg

UML 2.4.1 definitions

- **The subject:** the system under consideration [its] required behavior is specified by one or more use cases.
- **Actor:** entity external to the subject, a **role** played by a user or other system that interacts with the subject, does not necessarily represent a specific physical entity.
- **Use case**: specifies a required use of a system… captures the requirements - what a system is to do.



https://kenai.com/attachments/wiki_images/cps/CPS_Main_Use_Case_Diagram_v2.jpg

Naming use cases

Verb-noun use case names

- Do action on thing(s)

- Place order
- Browse catalogue
- Load price changes
- Get job instructions
- Print history

- How many? Between 2 and 99?

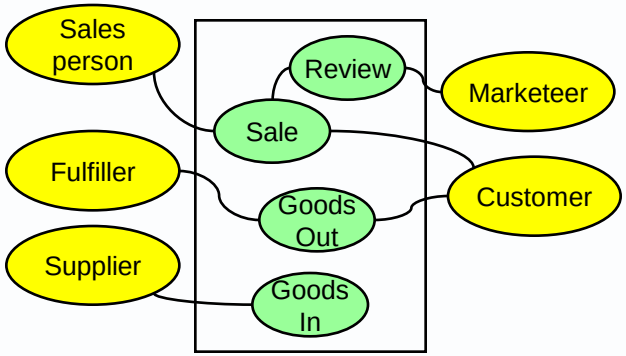
"If you end up identifying a **hundred** use cases, either your system should be broken down into a number of smaller projects,
or you are modelling use case pathways rather than use cases.

If you end up with **one** use case, then you need to move down a level of abstraction or get a bigger project."

After Ivar Jacobsen

Don’t forget the numbers – which inform NFRs

- Capture business capacity and performance measurements

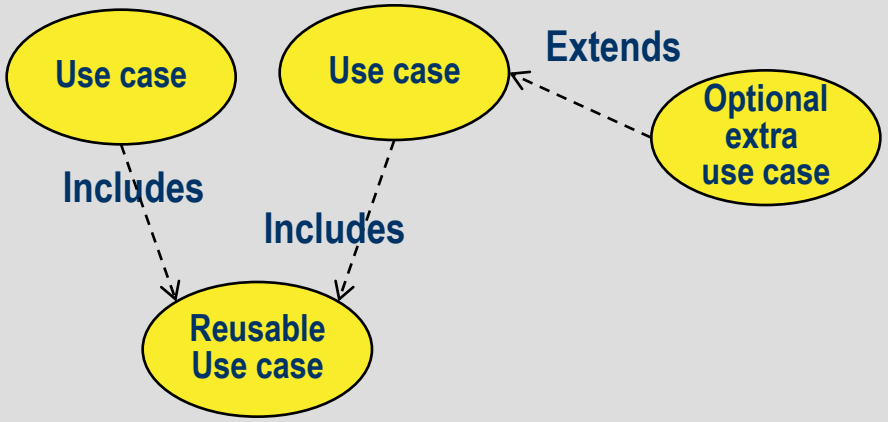


- For each use case / process:
  - Throughput, w peaks and troughs
  - Duration / response time
  - Availability (24*7?)
  - Time constraints (before 4.00 p.m.)
  - Resource cost
  - Etc.

UML 2.4.1 says a use case may **extend** or **include** another

An extends relationship

- specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.

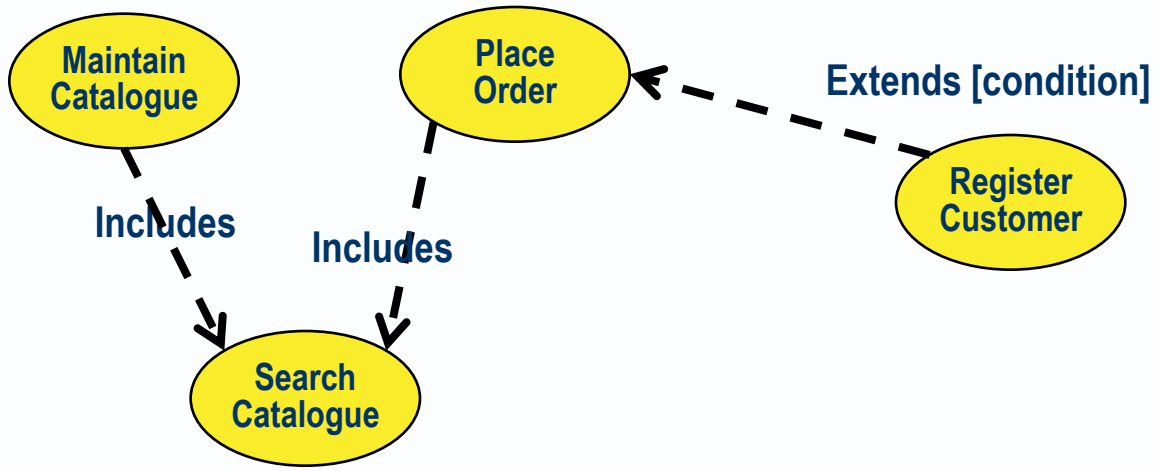


An include relationship

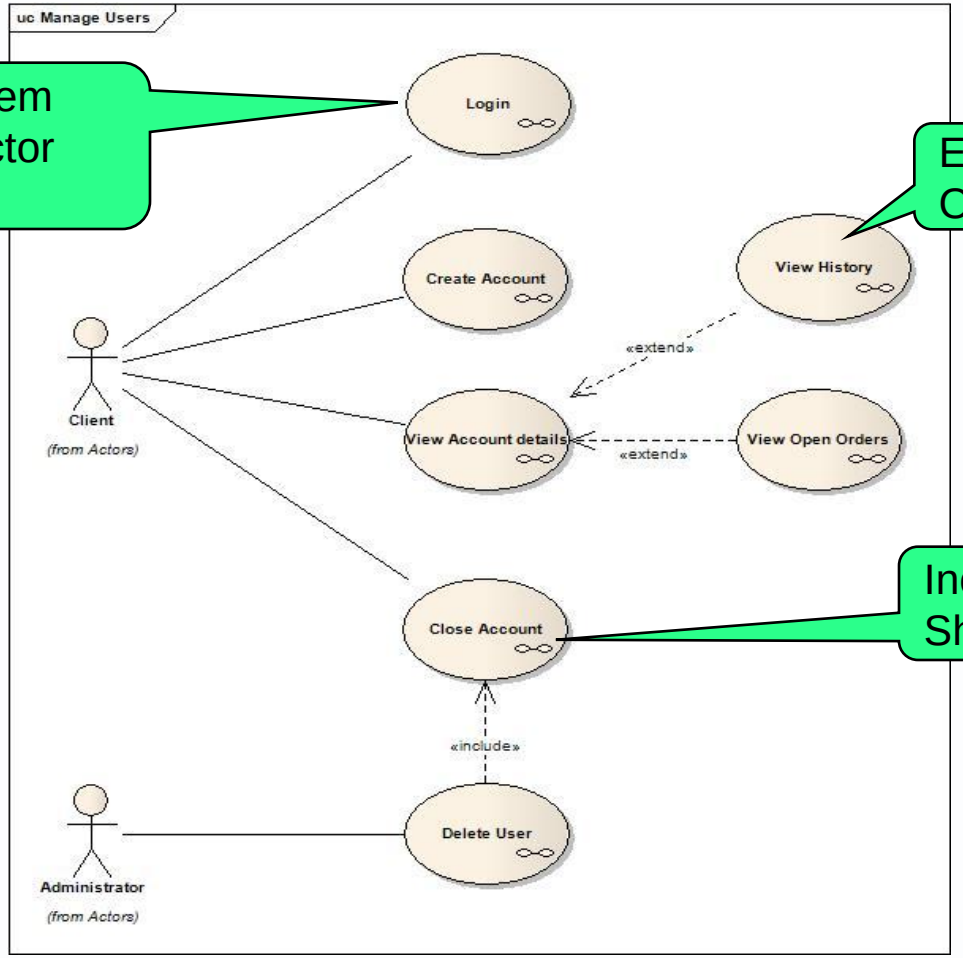
- defines that a use case contains the behavior defined in another use case.
- intended to be used when there are common parts of the behavior of two or more use cases.
- analogous to a subroutine call, which is completed before execution of the including use case is resumed.

UML is ambiguous; the usual interpretation is

- *Extends* use case is optional/conditional
- *Included* use case is mandatory **and/or** reused/shared
- (Use includes if in doubt)



An example of extends and includes



Use case = a system process that an actor engages with

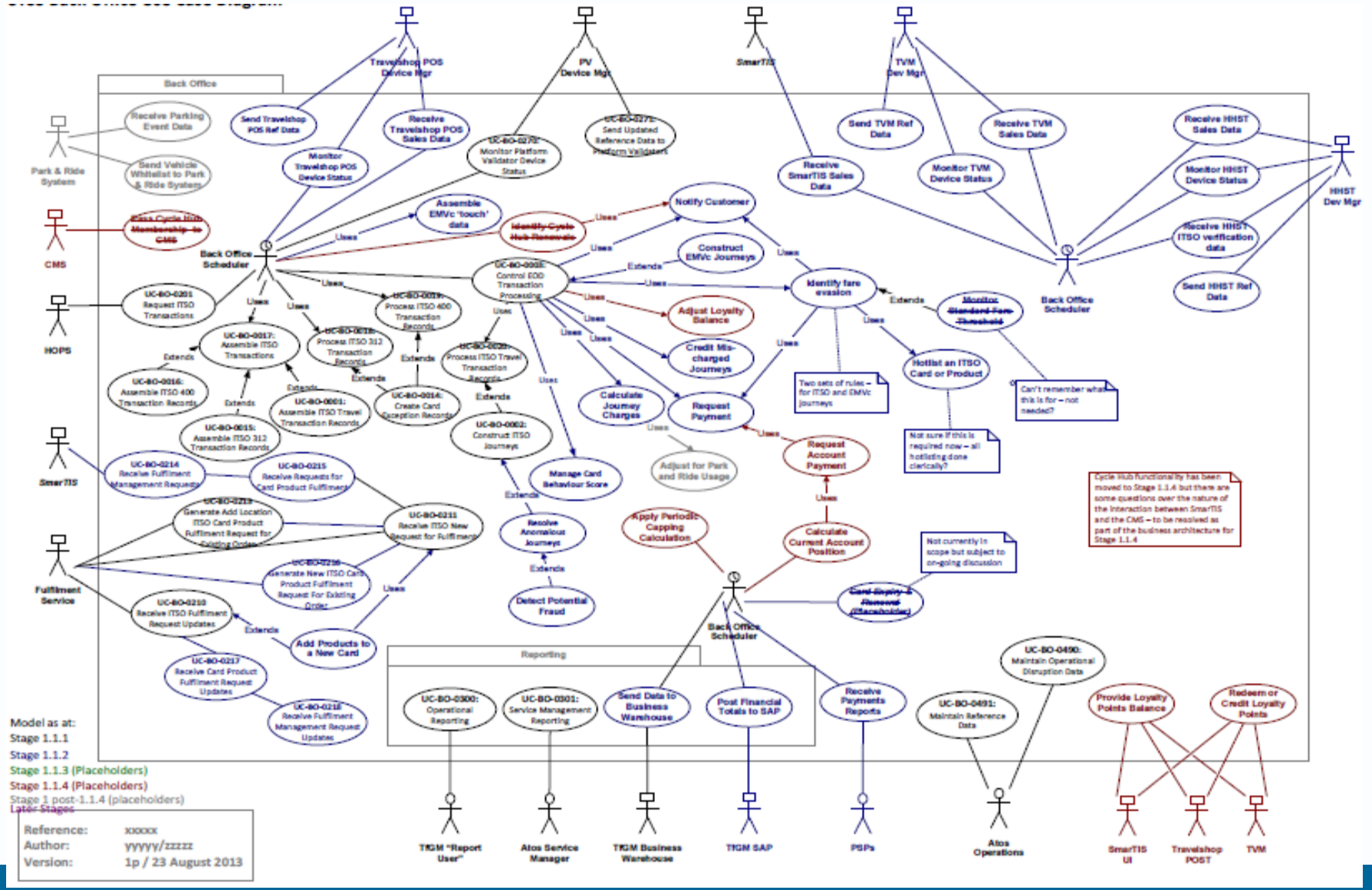
Extends with Optional process

Includes Shared process

http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/353/Enterprise-Architect-for-Business-Analysts.aspx

Real world example, with colour coding of phases

Avancier



Design IS (application) services (AM for SA level 4)

Avancier

1. Identify use cases
2. Draw use case diagram
3. **Describe use cases**
4. Identify automated services

Use Case = process = value stream

- Use cases **define the offered behavior** of the subject without reference to its internal structure.

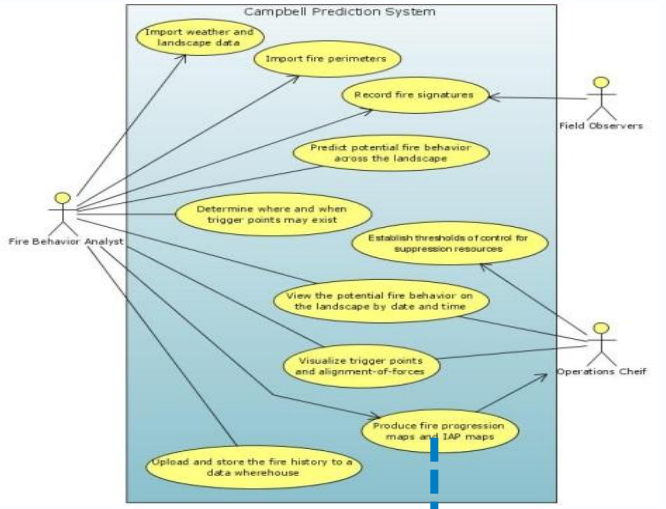
- A use case is the specification of
  - a set of **actions performed** by a system, which
  - yields an **observable result** that is, typically,
  - of value for actors or other stakeholders of the system.

- It is deemed complete if, after its execution, the subject will be in a state in which no further inputs or actions are expected [now] and the use case can be initiated again…

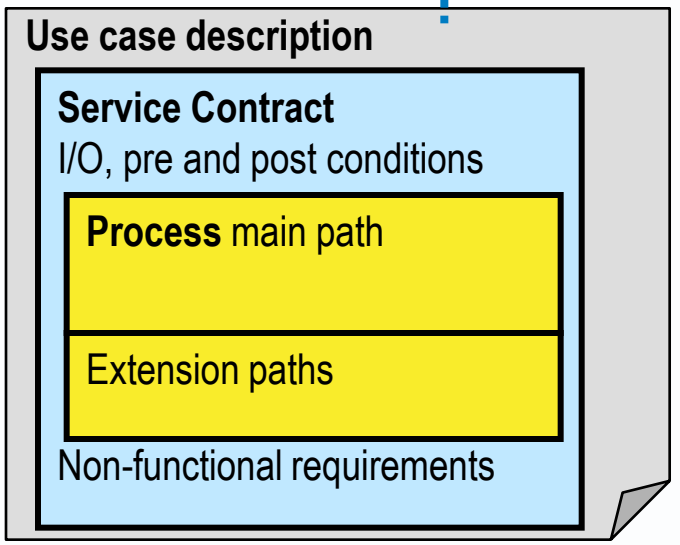
Edited from UML 2.4.1

Describe use cases

- **Application use case diagram**
  - scopes the application system
  - *identifies and names* use cases



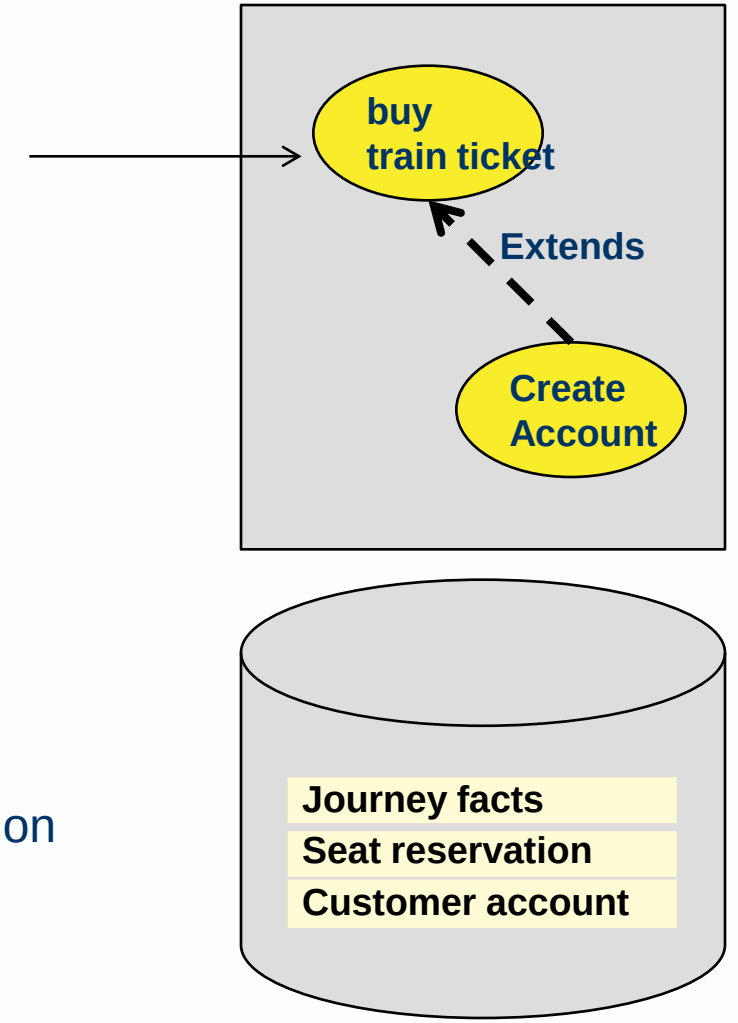
- **Use case description**
  - *describes* use cases
  - **Service** offered by the use case
  - **Process** executed during use case
    - Main path
    - Other paths



Use Case definition as a process

- Iteratively elaborate the use case description
  1. Name a process that handles input or output
  2. Define the service contract's logical elements
  3. Define the service contract's physical elements
  4. Define the process main path
  5. Define the process exception paths

- Create data dictionary
  1. Define data created and used by the process
  2. Give each data group a nickname
  3. Associate data fields with the nickname
  4. Define field types, lengths, validation and derivation rules



Name a process that handles input or output

- Where is the user?
- What time and ability do they need?
- What client device do they have?
- What data do they need?
- What data must they create (for later use)?

Define the service contract's logical elements

Avancier

External behavior (Service contract logic) 	**Name**	Buy train ticket
	Goal, purpose, value added	As implied by the name above
	Roles	Customer, Rail Company web application
	Entry criteria	**Trigger:** Customer accesses web site to buy a ticket **Input:** Journey facts **Preconditions:** Availability of ticket and money
	Exit criteria	**Outputs or products:** Seat reservation **Post conditions:** Seat reservation recorded in one database Payment authorisation recorded in another database
Internal behavior (Processs) 		
External behavior (Service contract measures)		

Define the service contract's physical elements

Avancier

External behavior (Service contract logic) 	**Name**	Buy train ticket
	Goal, purpose, value added	As implied by the name above
	Roles	Customer, Rail Company web application
	Entry criteria	**Trigger:** Customer accesses web site to buy a ticket **Input:** Journey facts **Preconditions:** Availability of ticket and money
	Exit criteria	**Outputs or products:** Seat reservation **Post conditions:** Seat reservation recorded in one database Payment authorisation recorded in another database
Internal behavior (Processs) 		
External behavior (Service contract measures)	**Non-functional qualities**	**Response time:** 5 minutes average **Throughput:** 100,000 per day **Availability:** 99% (24*7) except planned down time **Integrity:** 100% **Scalability:** 200,000 per day **Security:** as per legislation and company standards **Etc.** **Etc.**

Define the process main path

Avancier

External behavior (Service contract logic) 	**Name**	Buy train ticket
	Goal, purpose, value added	As implied by the name above
	Roles	Customer, Rail Company web application
	Entry criteria	**Trigger:** Customer accesses web site to buy a ticket **Input:** Journey facts **Preconditions:** Availability of money
	Exit criteria	**Outputs or products:** Seat reservation **Post conditions:** Seat reservation recorded in one database Payment authorisation recorded in another database
Internal behavior (Processs) 	**Activities**	1. Identify journey start and end stations 2. Identify outbound start time 3. Identify inbound start time 4. Identify traveller numbers and ages 5. Review buying details 6. Enter payment details 7. Create personal account (optional) 8. Confirm payment details 9. Collect receipt
External behavior (Service contract measures)	**Non-functional qualities**	**Response time:** 5 minutes average **Throughput:** 100,000 per day **Availability:** 99% (24*7) except planned down time **Integrity:** 100% **Scalability:** 200,000 per day **Security:** as per legislation and company standards **Etc.** **Etc.**

Main path = happy path = straight-thru path = sunny day path

Avancier

- The end-to-end process that leads to the desired goal/result

buy Rail Ticket

1. Identify journey start and end stations
2. Identify outbound start time
3. Identify inbound start time
4. Identify traveller numbers and ages
5. Review buying details
6. Enter payment details
7. Create personal account (optional)
8. Confirm payment details
9. Collect receipt

buy Air Ticket

Search Flights

Select Flight

Flight Itinerary

Passenger Details

Payment Details

Confirm & Pay

Simple use case description

Avancier

Service	Name: Buy train ticket
	Inputs: Journey facts
	Outputs or products: Seat reservation
Process flow	**1. Identify journey start and end stations** **2. Identify outbound start time** **3. Identify inbound start time** **4. Identify traveller numbers and ages** **5. Review buying details** **6. Enter payment details** **7. Create personal account (optional)** **8. Confirm payment details** **9. Collect receipt**
Non-functionals	Response time
	Throughput
	Availability
	etc

Process flow

NFRs

Define the process exception paths

Avancier

Use case name	Price kitchen
Subject	Kitchen sales system
Actor	Salesman
Stakeholder	Customer
Precondition	Kitchen plan has been populated with kitchen items Kitchen order app is loaded onto the lap top
Trigger event	Customer wants to know price (so far, or for order) Salesman presses price kitchen command
Post condition	Price is displayed in price dialogue box
Main path	**1 Salesman presses price kitchen command** **2 Kitchen drawing app displays price dialogue box** **3 Salesman clicks OK** **4 Kitchen drawing app invokes Kitchen order app** **5 Kitchen drawing app displays the reply in the Price field** **6 Salesman closes the price dialogue box**
Extension paths	**3 Salesman closes the price dialogue box** **5 Kitchen drawing app displays "Kitchen order app not available"**
NFRs	

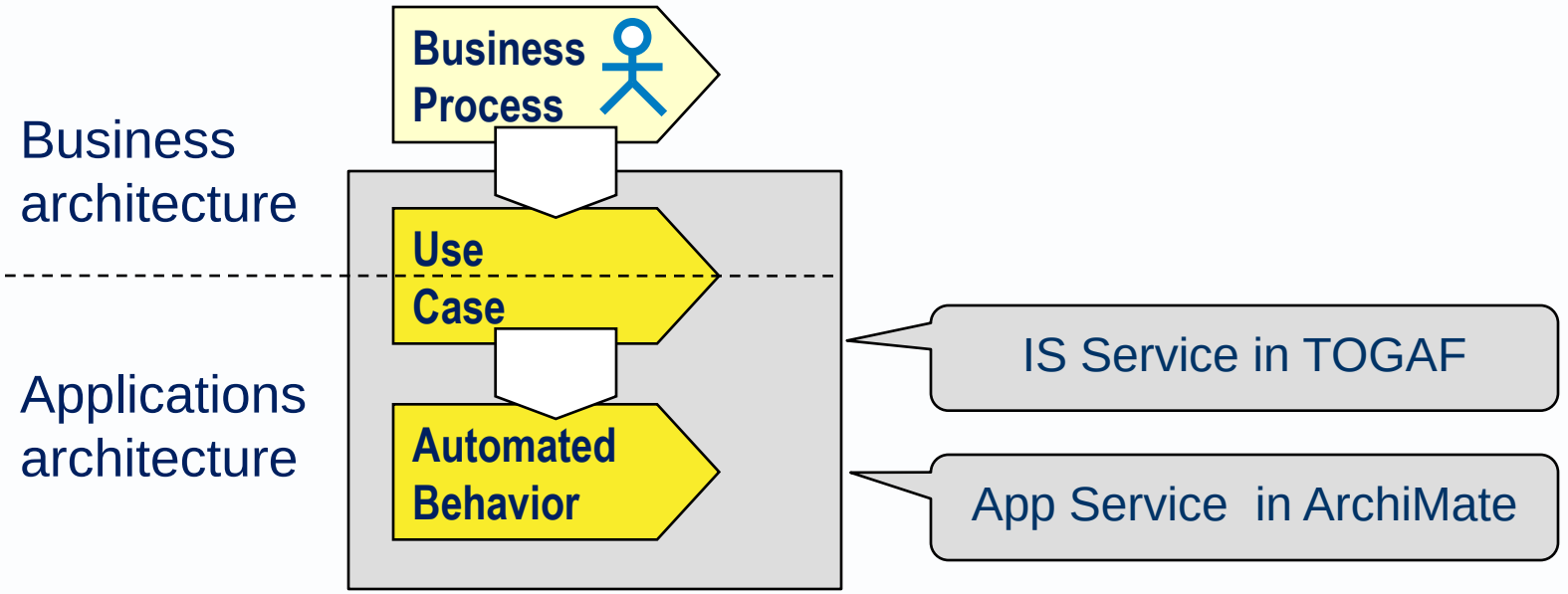
Design IS (application) services (AM for SA level 4)

Avancier

1. Identify use cases
2. Draw use case diagram
3. Describe use cases
4. **Identify automated services**

Automated behaviors

- During the progress of a use case, the use case may require several automated services – which do not require any input or action from the actor to complete



Including the automated service in the use case

Avancier

Goal or name	Capture and confirm order
Subject	Kitchen sales system
Actor	Salesman
Stakeholder	Customer
Trigger event	Customer wants to place order Salesman presses order kitchen command
Precondition	Kitchen plan has been populated with kitchen items Kitchen order app is loaded onto the lap top
Post condition	Order is printed, signed and logged in order app
Main path	1 Salesman presses order kitchen command (OPEN ORDER APP) 2 Kitchen drawing app opens Kitchen Order app in new window 3 Salesman enters customer details 4 Salesman sends order to printer (PRINT ORDER) 5 Salesman presents order to customer 6 Customer signs order 7 Salesman enters confirmation of signature (CONFIRM ORDER) 8 Salesman close order app 9 Kitchen drawing app stores kitchen plan as version n+1
Extension paths	2 Kitchen drawing app displays “Kitchen order app not available” 3 Salesman closes order app before printing the order
NFRs	

Reference to automated service

Automated service: Confirm order

- Defined by a service contract

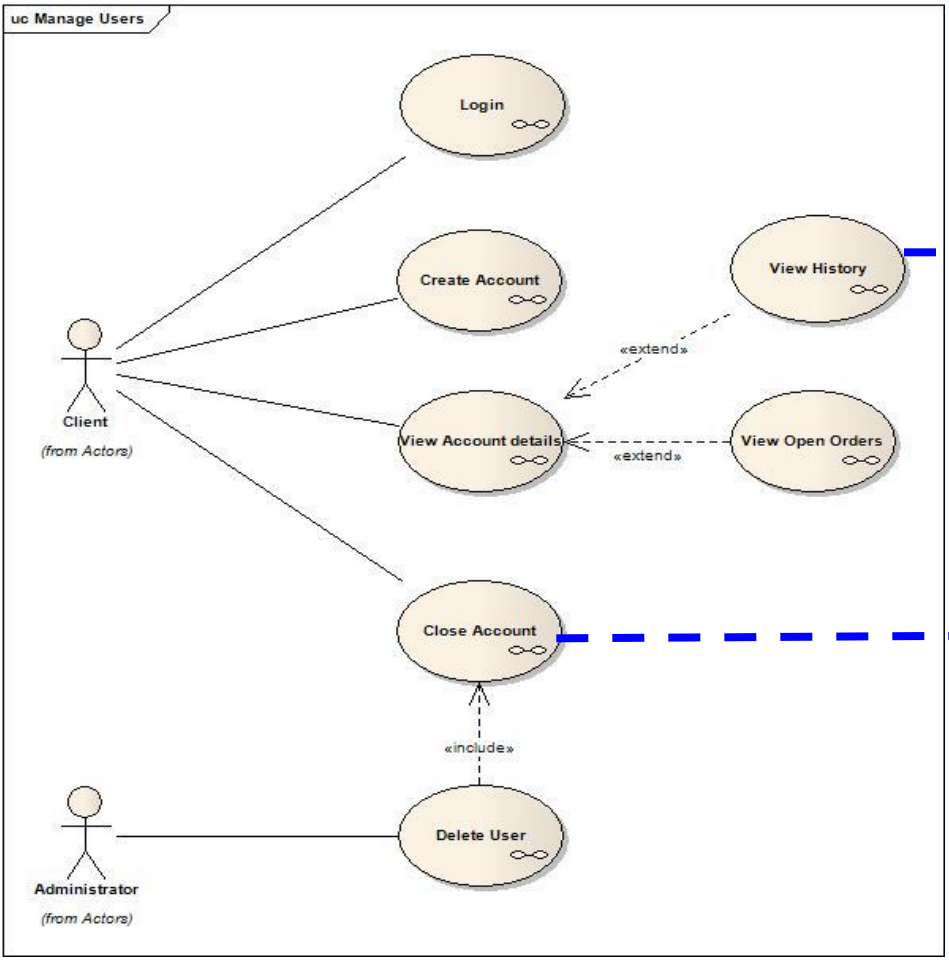
Service Contract	Automated Service	999
Signature	Name	Confirm Order
	Input data	Order number
	Output data	OK or Fail
Semantics or Rules	Preconditions	Order stored in a provisional state
	Post conditions	Order stored in confirmed state
Non-Functional Requirements	Response time	0.5 second
	Throughput	5 per minute
	Availability	99% 07.00 to 19.00
	Security level	2

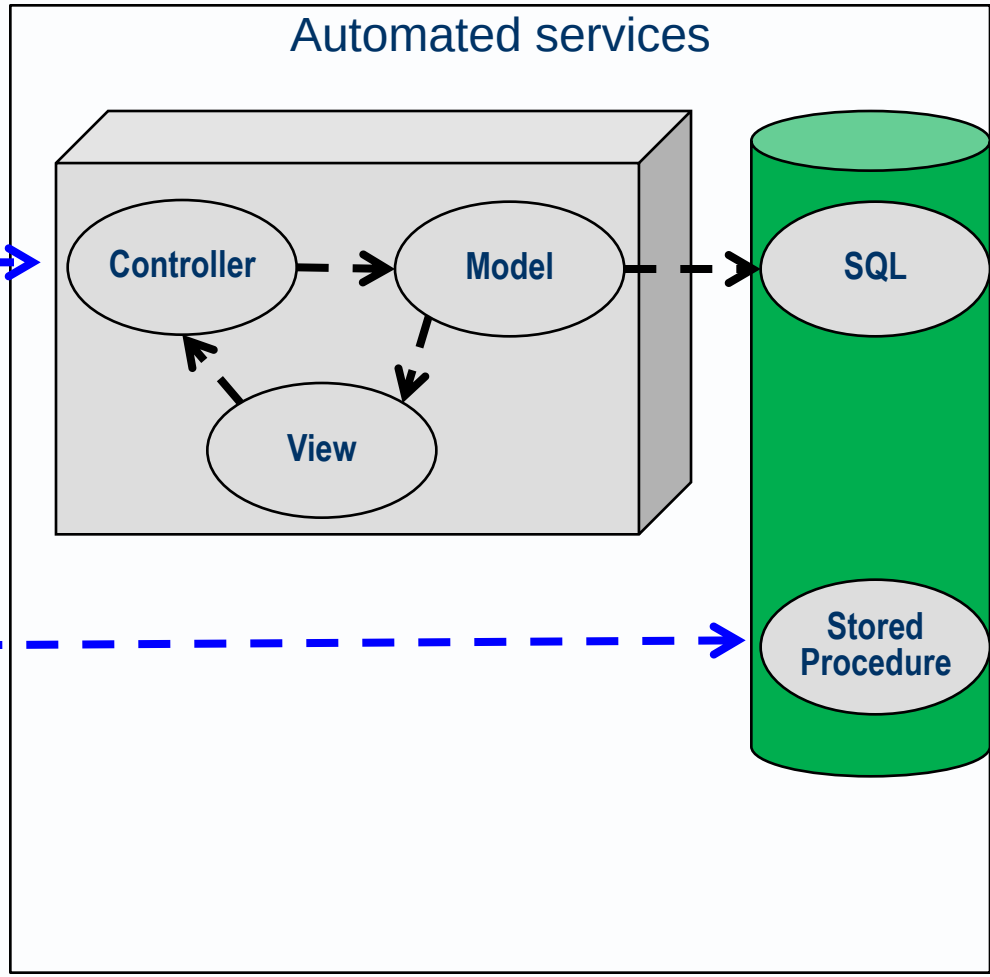
Class example

Service Contract	Automated Service	999
Signature	Name	Get File
	Input data	File name, User credentials, Protocol
	Output data	CSV File
Semantics or Rules	Preconditions	File exists, File picked date null, Valid user, Valid Protocol,
	Post conditions	File picked date set
Non-Functional Requirements	Response time	1 minute
	Throughput	1 day
	Availability	99% (usually business hours)
	Security level	Valid user

HCI "Application use cases" –-> Automated "Service use cases"

Avancier



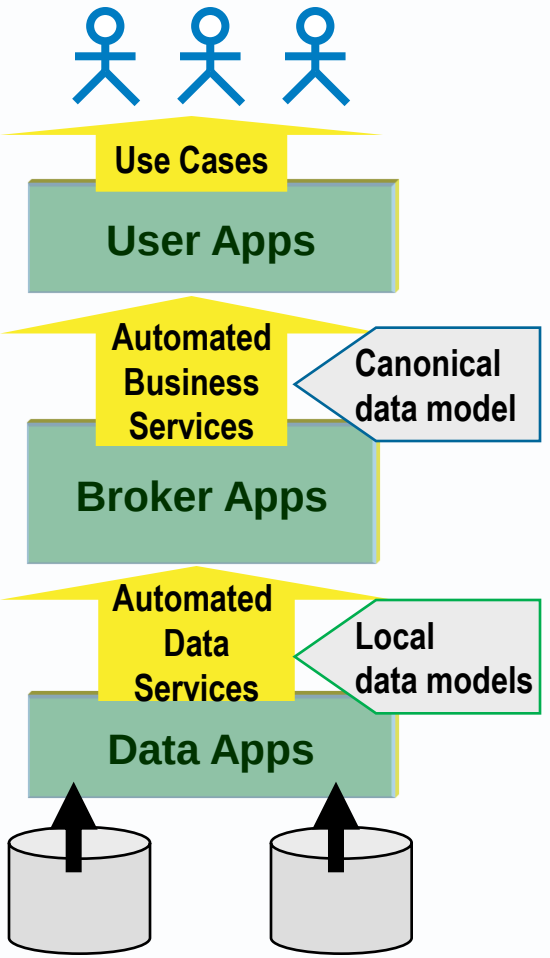


Automated behavior

- [An application service] that can be requested of a software component.
- It is sometimes an ACID transaction.

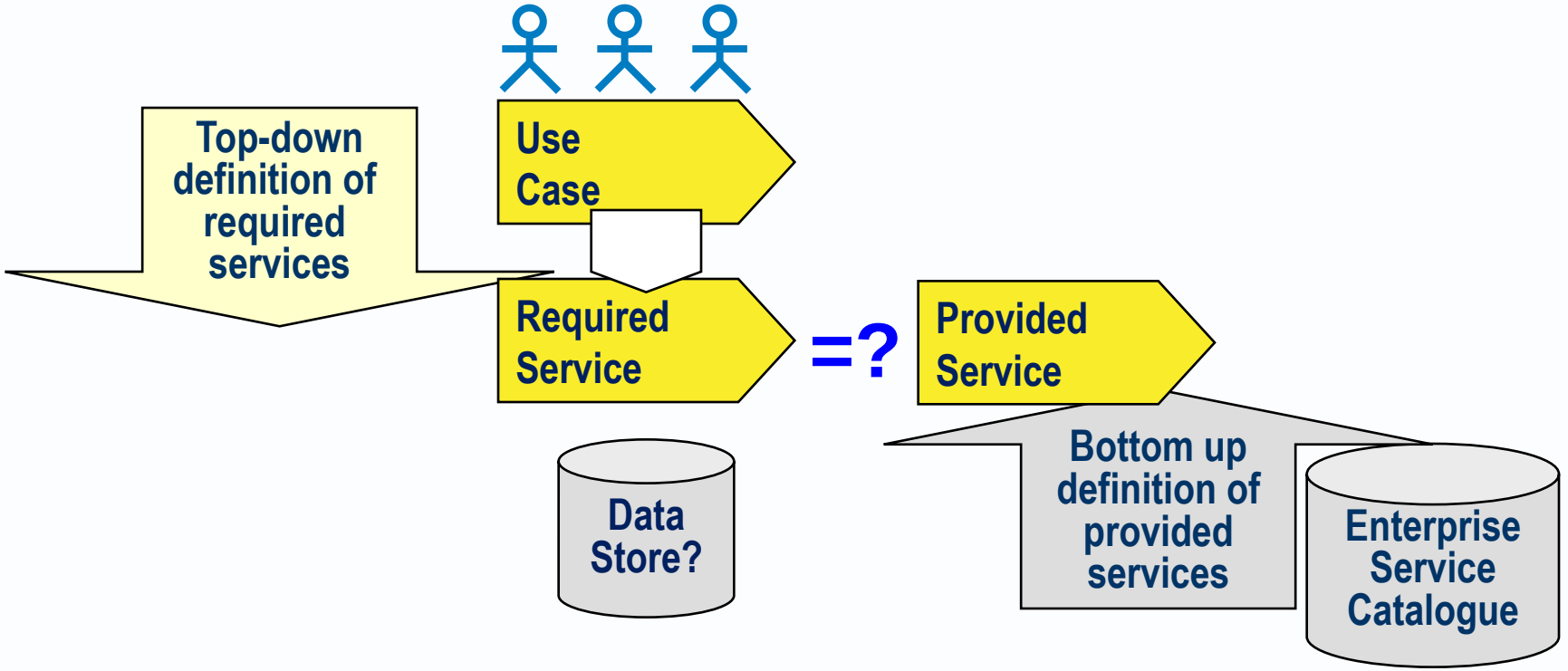
- **Automated business service**
- [An automated behavior] whose input and output data is defined in a canonical data model.

- **Automated data service**
- [An automated behavior] whose input and output data items are defined according to the parochial or physical data model of a specific data source.



Reuse of automated services

- If you want solutions to share automated services
- Then architects have to consider whether there is a match between
  - the services required by application use cases
  - the services provided in an enterprise service catalogue



FOONOTES

- User stories
- Epics
- Interaction diagrams

User story

- User Stories come from Extreme Programming (XP)
  - An agile development model by Kent Beck, Ward Cunningham and Ron Jeffries at Chrysler 1995 to 2000.

- The Scrum Guide does not mention *user stories*.
- It does mention *product backlog items* (features, functions, requirements, enhancements, and fixes)
- However, product backlogs often contain user stories
- A tools like JIRA call items in the backlog user stories

User stories (as used in SCRUM)

- **User story**
  - A self-contained “unit of work” agreed by developers and stakeholders.
  - Independent enough to provide measurable business value to a user.

- **Template – a prompt for discussion**
  - As a [role]
  - I want to [do something)
  - So that [goal/objective/purpose]
  - + Test acceptance criteria

- **Example: “set password”**
  - As a salesperson,
  - I'd like to set my password,
  - So I can log into the system.

Epics (as used in SCRUM)

- **An epic (E.g. “log in”)**
  - A deliverable made up of multiple user stories.
  - A "big story” or complete work flow for a user
  - While each story may be completed independently, their business value isn't realized until the entire epic is deliverable.

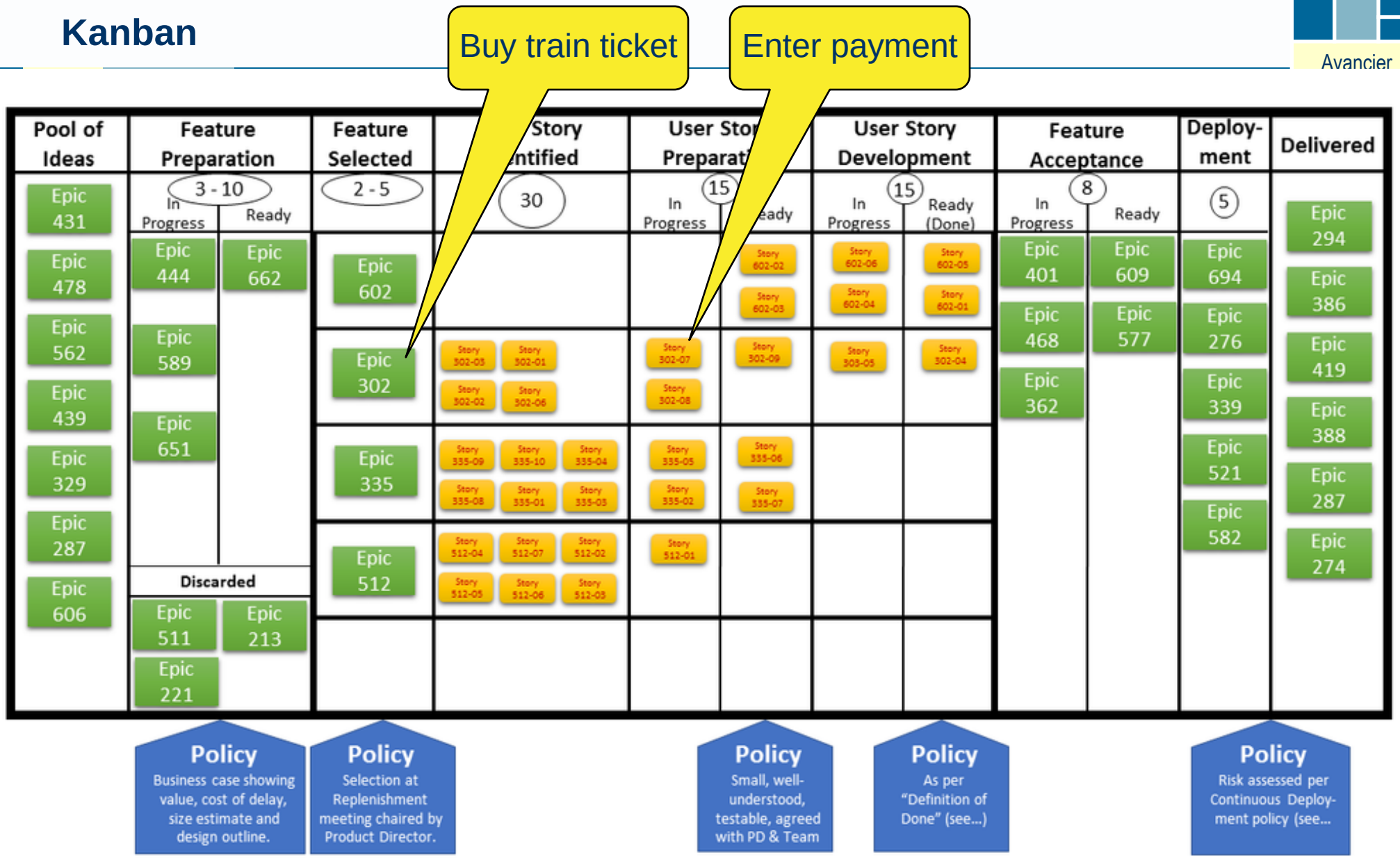
User story
- As a [role]
- I want to [do something)
- So that [goal/objective/purpose]
- + Test acceptance criteria

User story
- As a [role]
- I want to [do something)
- So that [goal/objective/purpose]
- + Test acceptance criteria

User story
- As a [role]
- I want to [do something)
- So that [goal/objective/purpose]
- + Test acceptance criteria

- **A theme: “data entry”**
  - A group of stories related in some other way, such as focusing on a single customer.

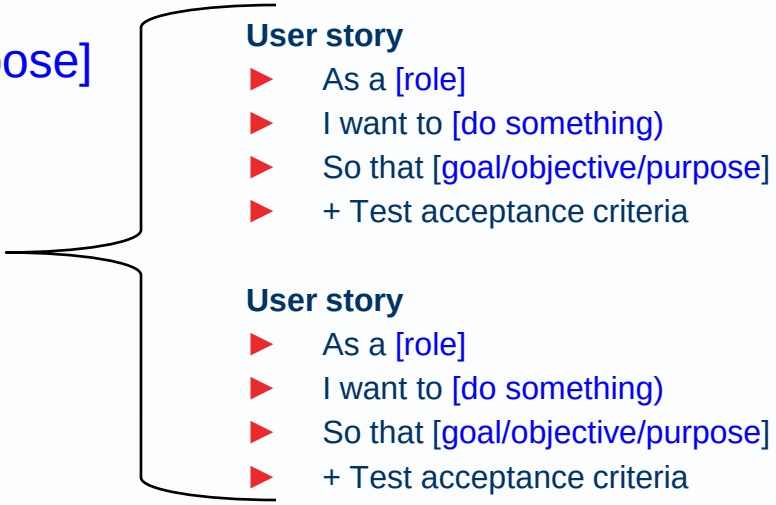
Kanban



Use Cases = Epics

- **Use Case (E.g. “log in”)**
  - An actor uses an application
  - To perform a process
  - To achieve an aim
  - Typically at the OPOPOT level
  - Involving one or more user stories

- **Template**
  - Name: [goal/objective/purpose]
  - Role:
  - Trigger etc.
  - Process: [do something]



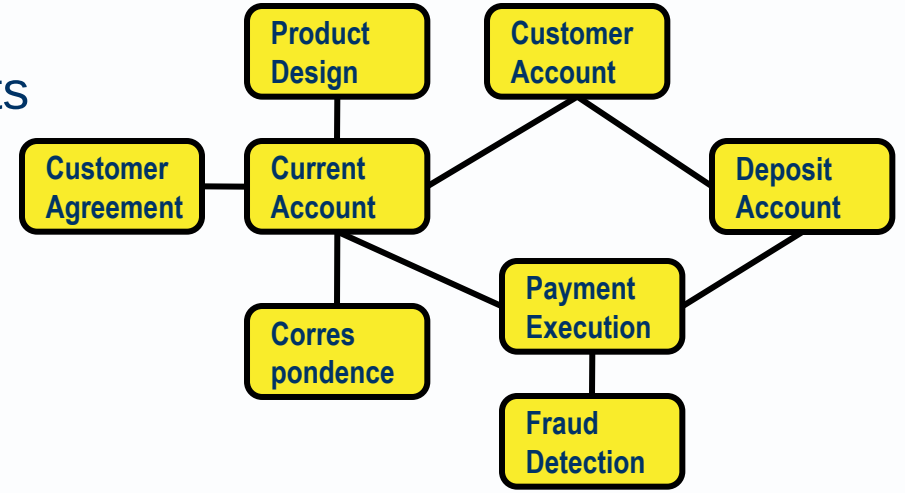
Alternative representations

- Use cases [are] used to specify required and/or offered behaviour by
  - pre-conditions and post-conditions
  - natural language text.
  - **interaction,** activity, or state machine diagrams

Edited from UML 2.4.1

One structure – many processes

- One structure of interrelated components



- One of many behaviours or processes

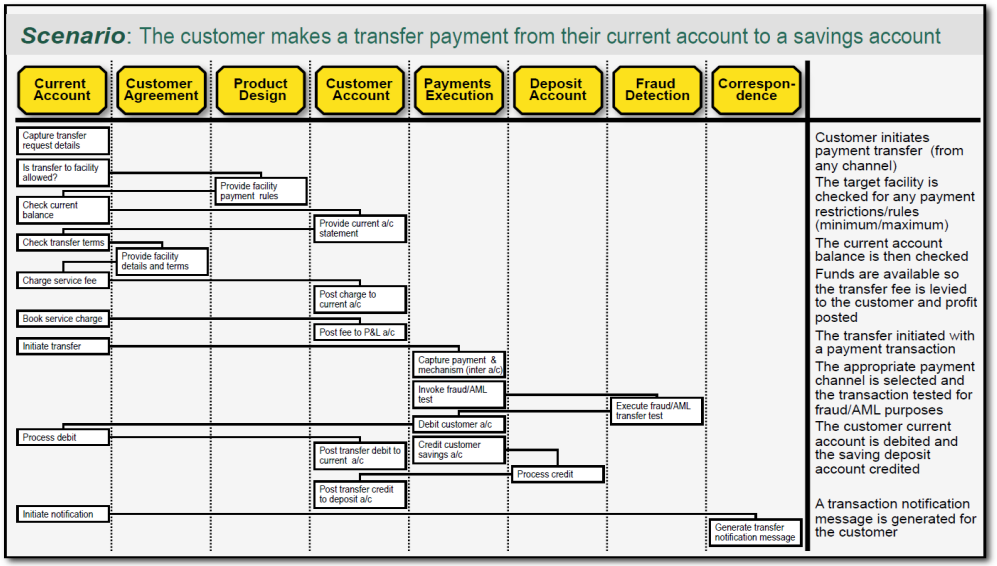


Figure 3 - A Simple Business Scenario

Use case as interaction diagram (from BIAN)

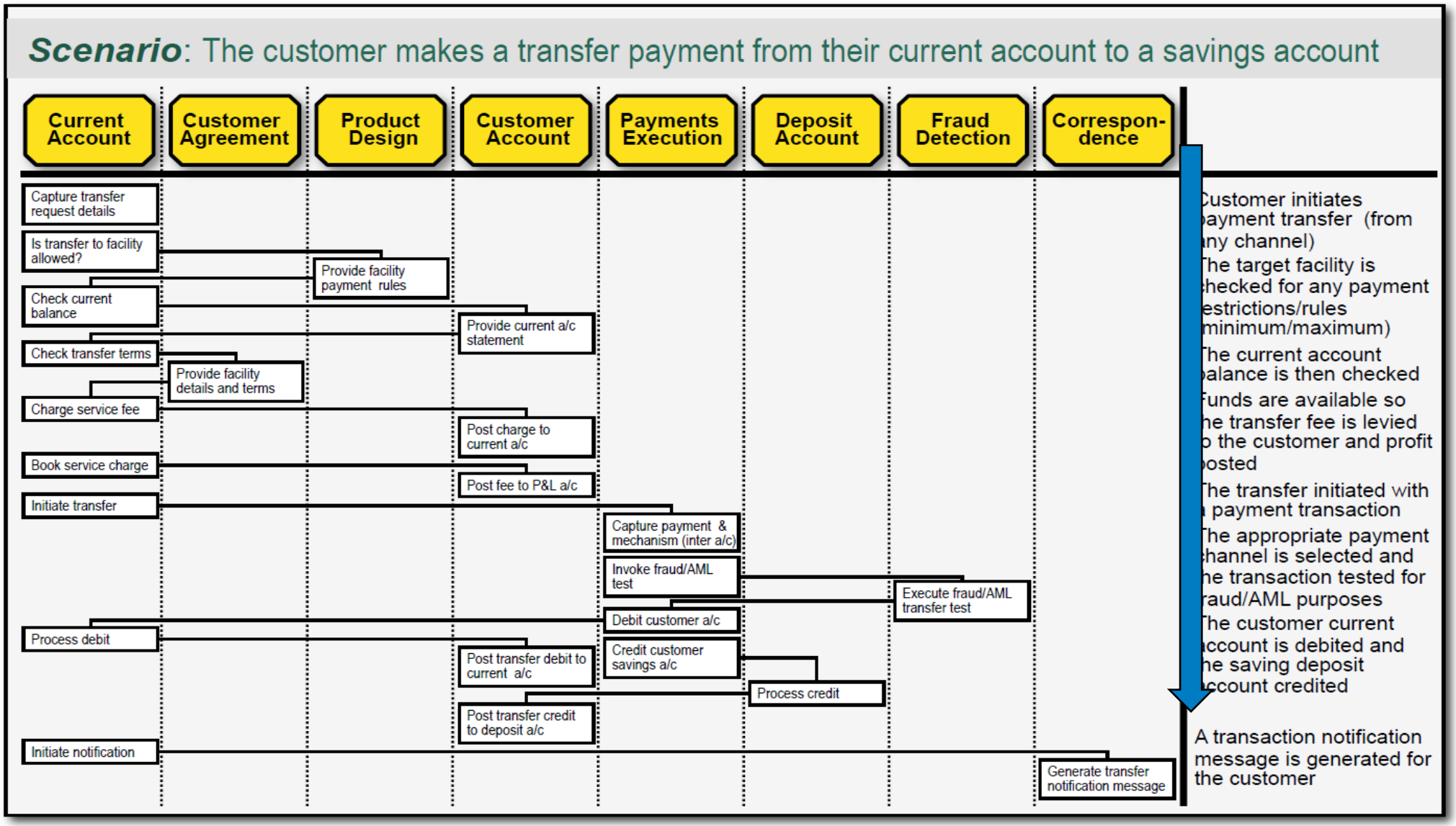


Figure 3 - A Simple Business Scenario