MSO
Scenarios and use cases

Hans Philippi
(based on the course slides of Wouter Swierstra)

August 21, 2018

Development Process: Scenarios and use cases

- One of the most popular way to establish (functional) requirements is through writing *scenarios* and *use cases*
- These document the system's behaviour from the users point of view, in a way that customers can understand what the system does – how does the system *add value* to the user?
- Use cases and scenarios are crucial if you want to agree on what the system does

Scenarios: definition

A narrative description of what people do and experience as they try to make use of computer systems and applications – (M. Carroll, 1995)

Scenario: example

- Alice goes to the cash machine
- She enters her card and PIN
- She requests to see her bank balance and withdraws 20 euro
- She takes her money, card and a printed receipt

Note that:

- This only describes a single transaction – it does not capture all possible interactions or all possible outcomes
- The *actors* involved may be people, systems, or even organizations

Use cases – definition

A *use case* is more general than a scenario. It documents an interaction between users and the system that yields an observable result or value.
When writing use cases, many authors distinguish different levels:

- Brief – a one paragraph summary of the main success scenario
- Casual – a few paragraphs that cover various scenarios
- Full dressed – a detailed account of all possible steps and variations, including entry conditions, exit conditions and special requirements

Use cases: brief

Bank Customers should be able to go to the ATM and withdraw money from their account. To do so succesfully, they start by identifying themselves using their bank card and PIN. After selecting the account from which they wish to withdraw money, the necessary information is transmitted to the bank. Once the bank verifies that there is sufficient money in the account, the money is dispensed, together with the bank card and a receipt.

Use cases: casual

Give a brief account several different scenarios:

- What if the PIN number is wrong?
- What if the ATM is out of money? Or cannot dispense the requested amount?
- What if the user does not have enough money on his account?
- . . .

What to write?

A fully dressed use case consists of:

- A unique name
- Participating actors
- Entry conditions
- Flow of events
- Exit conditions
- Special requirements

This general format was originally suggested by Alistair Cockburn and is used throughout Larman's *Applying UML and Patterns*.

Example: withdrawing money

1 Brief Description

This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account

2 Actors

Bank Customer and Bank

3 Entry conditions

- There is an active network connection to the Bank
- The ATM has cash available

Example – Basic Flow of Events

1. Bank Customer inserts their Bank Card
2. Use Case: Validate User is performed
3. The ATM displays the different alternatives that are available on this unit; in this case the Bank Customer always selects "Withdraw Cash"
4. Card ID, PIN, amount and account are sent to Bank as a transaction; the Bank Consortium replies with a go/no go reply telling if the transaction is ok
5. The Bank Card is returned
6. The money is dispensed
7. The receipt is printed
8. The use case ends successfully

Alternative Flows

1 Invalid User

If in step 2 of the basic flow Bank Customer the use case: Validate User does not complete successfully, then:

1. The use case ends with a failure condition

2 Insufficient cash

If in step 5 in the basic flow, the Bank Customer enters an amount that exceeds the amount of cash available in the ATM, then

1. The ATM will display a warning message, and ask the Bank Customer to reenter a lower amount
2. The use case resumes at step 5

Alternative Flows

3 No response from Bank

If in step 6 of the basic there is no response from the Bank within 3 seconds, then

1. The ATM will re-try, up to three times
2. If there is still no response from the Bank, the ATM shall display the message "Network unavailable – try again later"
3. The ATM shall return the card
4. The ATM shall indicate that it is "Closed"
5. The use case ends with a failure condition

. . . *and lots more can go wrong*

Exit conditions

7 Post-conditions

7.1 Successful Completion
The user has received their cash and the internal logs have been updated

7.2 Failure Condition
The logs have been updated accordingly

8 Special Requirements

[SpReq:WC-1] The ATM shall support localizations for all major European languages.
[SpReq:WC-2] The ATM shall keep a log, including date and time, of all complete and incomplete transactions with the Bank

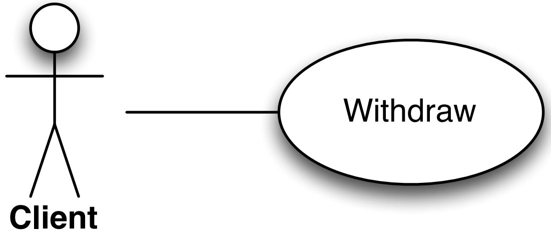
Use cases – advice

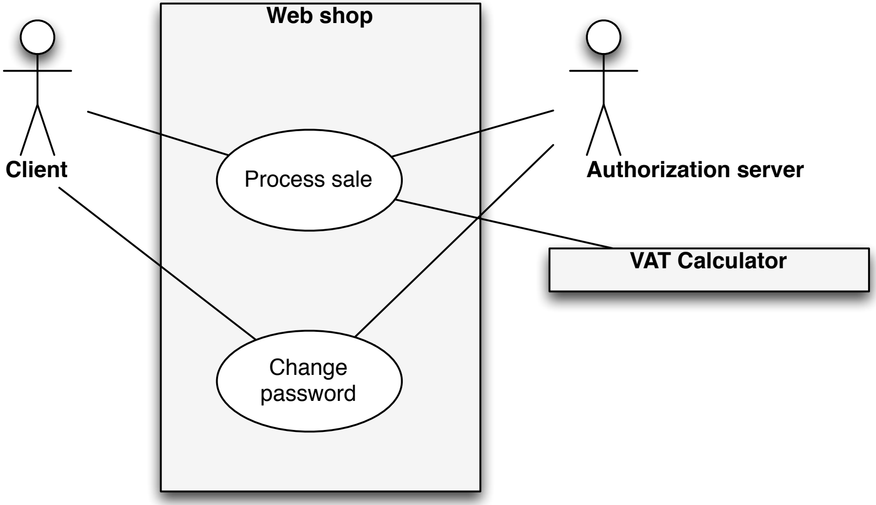
- Don't think about implementation or user interfaces
- Do think about user experience
- Keep the writing as simple as possible – you're not writing a novel or academic textbook, but a piece of text that should be as unambiguous as possible

Use cases – diagrams

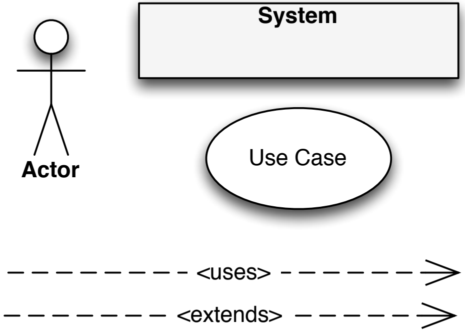
- Sometimes it can be useful to visualize which actors interact during certain use cases
- In that case, it can help to draw UML use case diagrams

UML Use Case Example



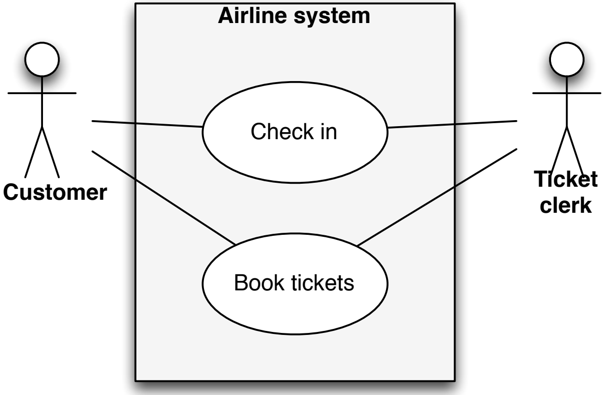

Web shop
Client
Process sale
Authorization server
VAT Calculator
Change password

Elements of UML Use case diagrams

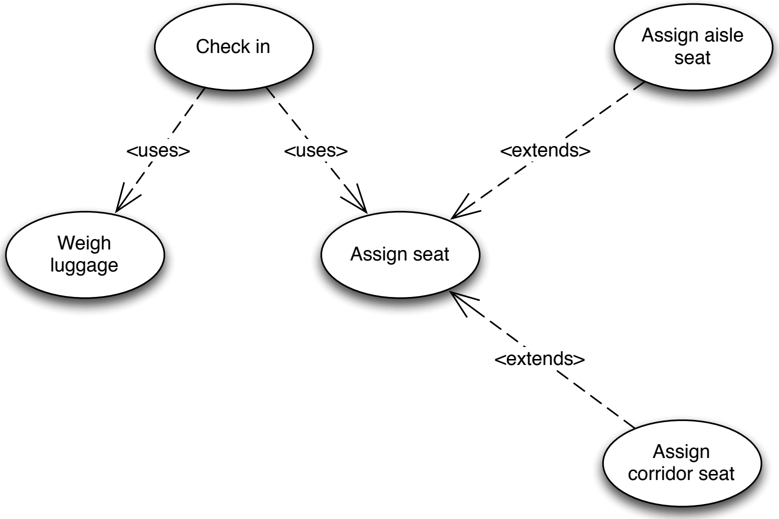


Diagrams

- Actors may be users, other stakeholders, or even software systems
- You may want to organize actors or use cases into system boxes
- You can re-use use cases using the <<`uses`>> arrows – for instance, both cash withdrawal and changing your PIN code require some form of authentication
- You may want to identify special cases of a given use case, using <<`extends`>> arrows – for example, to distinguish Assign Window Seat and Assign Aisle Seat are both extensions of the Assign Seat use case


Airline system
Customer
Check in
Book tickets
Ticket clerk

Detailed view



Requirement verification and validation

Requirements validation is concerned with checking the requirements document for consistency, completeness, and accuracy

Requirements verification is a mathematical analysis, possibly automated, of formal specifications for consistency

Techniques

You need to interact with users and customers to validate your requirements

- Reviews, checklists, discussion
- Prototypes
- Animations

Requirements negotiation

Usually, there are constraints that limit the functionality that can be delivered (time, money, resources) . . .
. . . or there are conflicting views on requirements between different stakeholders

To establish a sensible set of requirements requires **negotiation** with your customer

Back to UP – example planning

- *Inception*
  - start with a two day workshop with all stakeholders aimed at inventorizing use cases, and writing a brief description of each one
- *Elaboration* (first iteration)
  - work out key use cases in greater detail and discuss these with the client
  - design high-risk architectural components
  - start implementation

As iterations proceed, more use cases are developed, together with the prototype system. There are regular checks with the customer to check that they agree with existing requirements and the prototype system.

Epilog

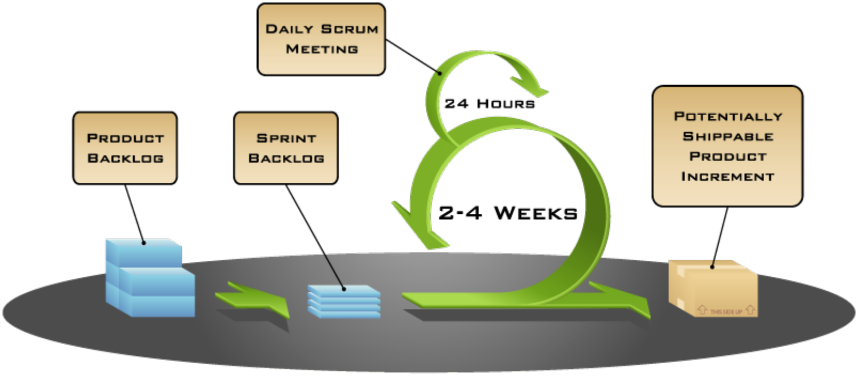
- How can I manage the process of constructing complex software?
  - Use the Rational Unified Process, or any other software development process
- How do I know what the customer wants?
  - Write use cases and requirements documents

UP vs Agile

The Agile Manifesto
"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:"

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile software development with Scrum



Material covered

- Rational Unified Process Whitepaper – available on the MSO website
- Craig Larman. Applying UML and Patterns. Pearson Education. 2002. Chapters 1–7