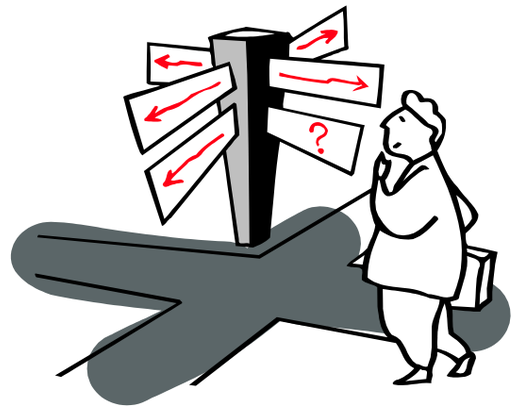
CHAPTER 2 – Requirements



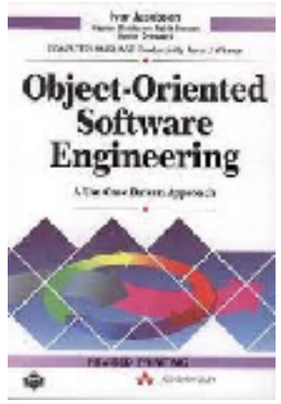
- Introduction
  + When, Why, Where, What
- Iteratively Developing Use Cases
  + Inception
    - Scope Definition + Risk Identification
    - Actors & Use cases + Project Plan
  + Elaboration
    - Primary & Secondary Scenarios
- Scrum: User Stories
    - Behaviour driven (template)
    - Conditions of Satisfaction
    - INVEST Criteria
      > Definition of Ready
    - Granularity: Epic / Features / Sprintable Stories
      > Product Roadmap
      > Minimum Viable Product
- Safety Critical
  + Misuse Cases
  + Safety Stories
- Conclusion
  + Use Cases / User Stories
    - Correctness & Traceability

Literature

Books

- [Ghez02], [Somm05], [Pres00]
  + Chapters on Specification/ (OO)Analysis/ Requirements Engineering

- Use Cases
  + [Schn98] Applying Use Cases - a Practical Guide, Geri Schneider, Jason, P. Winters, Addison-Wesley, 1998.
    - An easy to read an practical guide on how to iteratively develop a set of use cases and how to exploit it for project planning.
  + [Jaco92] Object-Oriented Software Engineering: A Use-Case Driven Approach, I. Jacobson et. al., Addison-Wesley, 1992.
    - The book that introduced use-cases



- User Stories
  + [Rubi13] Essential Scrum: A Practical Guide to the most popular agile process. Kenneth S.Rubin. Addison-Wesley, 2013.
    - The chapter on “Requirements and user stories”

Literature (Safety Critical)

Agile Requirements

- "User stories as lightweight requirements for agile clinical decision support development", Vaishnavi Kannan, et. al. In Journal of the American Medical Informatics Association, Volume 26, Issue 11, November 2019, Pages 1344–1354 https://doi.org/10.1093/jamia/ocz123
  + Mixing use cases with user stories in a medical (= safety-critical) context

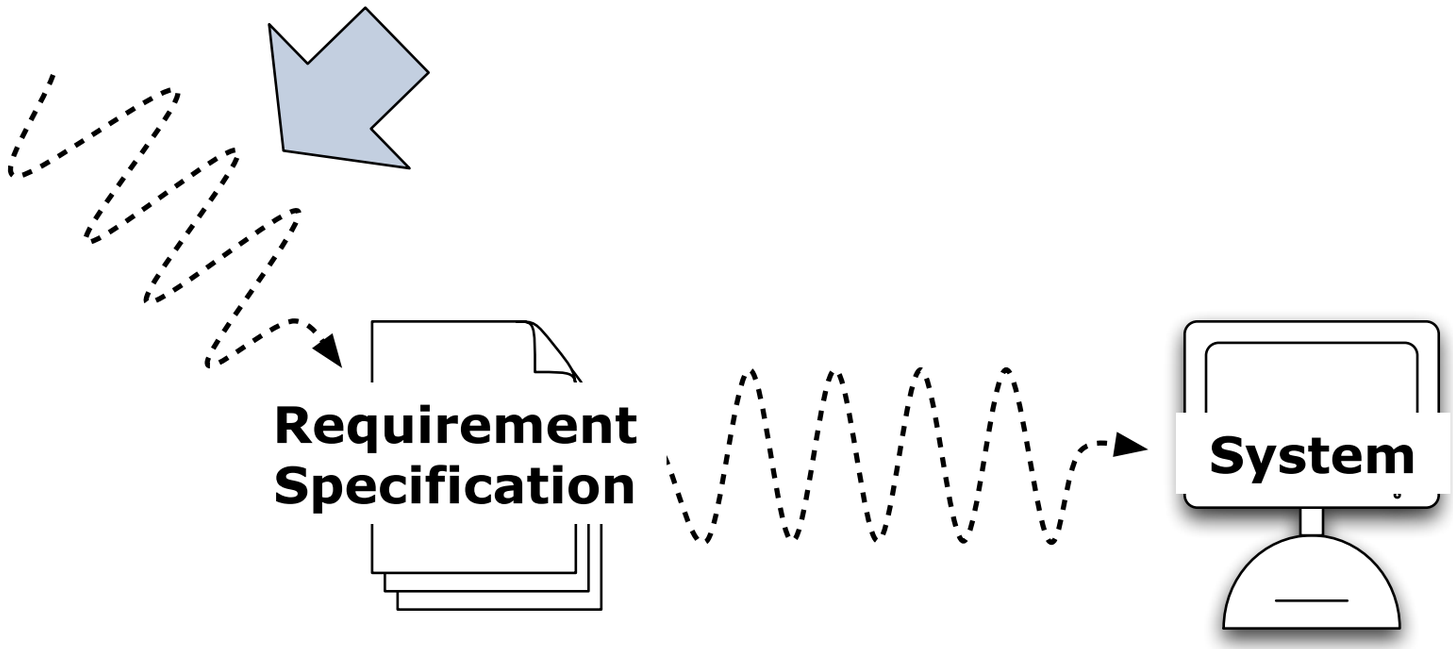
- *Making sense of MVP (Minimum Viable Product) – and why I prefer Earliest Testable/ Usable/Lovable*
  Posted on 2016-01-25 – 12:14 by Henrik Kniberg
  https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp
  + A plea for early feedback from actual users

Safety Critical

- Ian Alexander, "Misuse Cases: Use Cases with Hostile Intent" in IEEE Software, vol. 20, no. 01, pp. 58-66, 2003. doi: 10.1109/MS.2003.1159030
  + Explains a negative form of a use case; a negative scenario. Use- and misuse-case diagrams are valuable in threat and hazard analysis.

- Jane Cleland-Huang and Michael Vierhauser, "Discovering, Analyzing, and Managing Safety Stories in Agile Projects," 2018 IEEE 26th International Requirements Engineering Conference (RE), 2018, pp. 262-273, doi: 10.1109/RE.2018.00034.
  + Addresses the specific problems of discovering, analyzing, specifying, and managing safety requirements within the agile Scrum process.

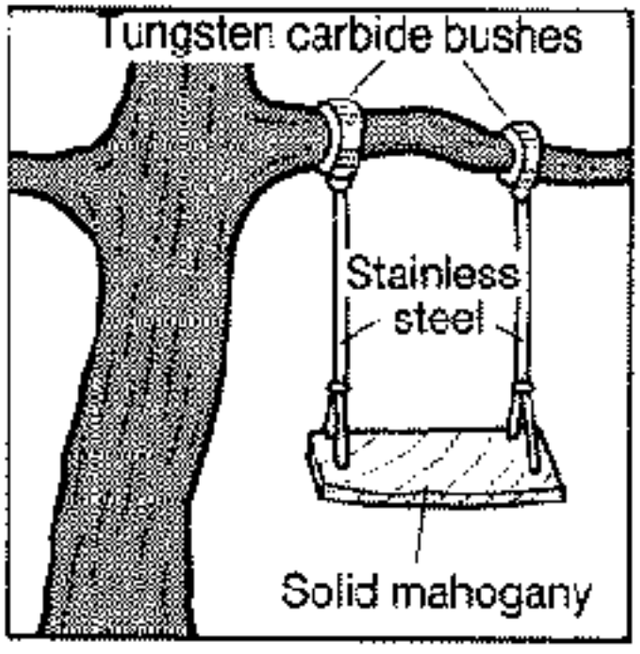
When Requirements?



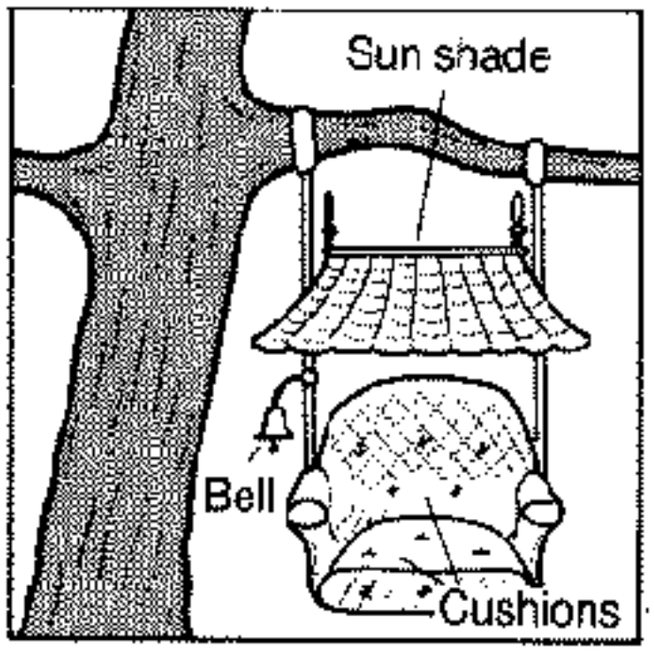
A requirements specification must be

- *understandable*: so that we can decide what is in- and out scope
  + Do all parties agree?
- *precise*: so that parties agree what's inside and outside the system
  + Can you write an acceptance test for each requirement?
- *open*: so that developers have enough freedom to pick an optimal solution
  + Requirements specify the "what", not the "how ".

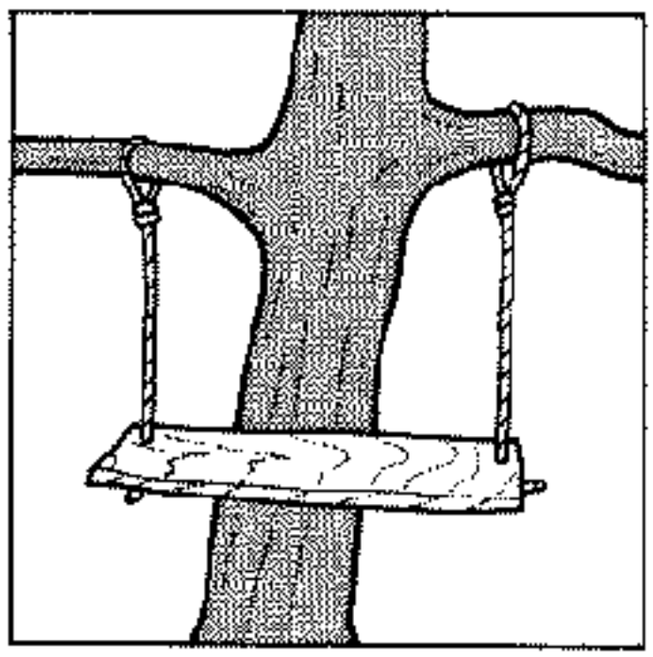
Why Requirements?



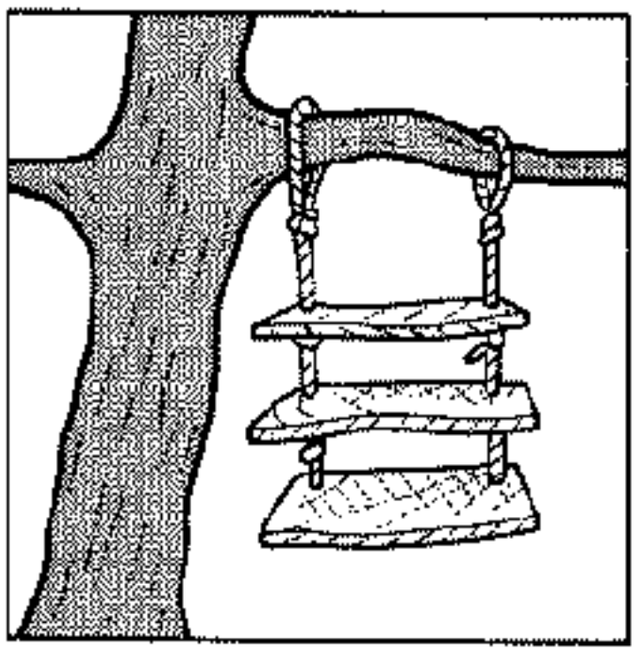
What Product Marketing specified



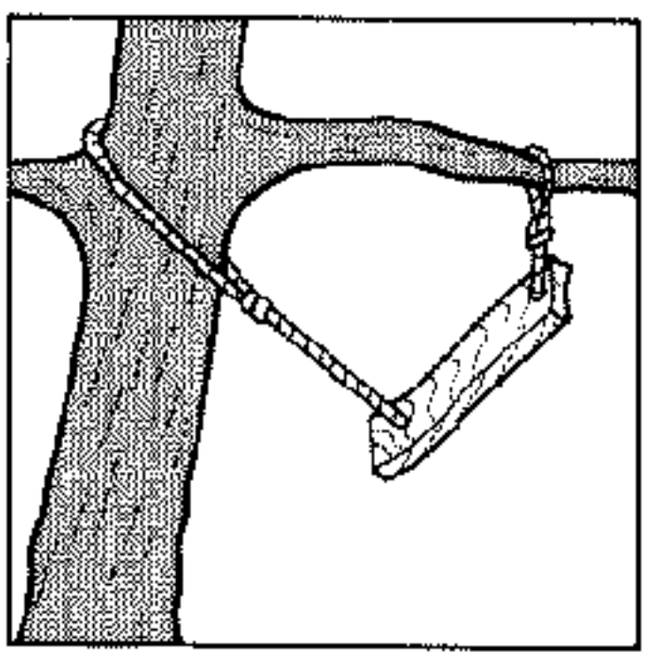
What the salesman promised



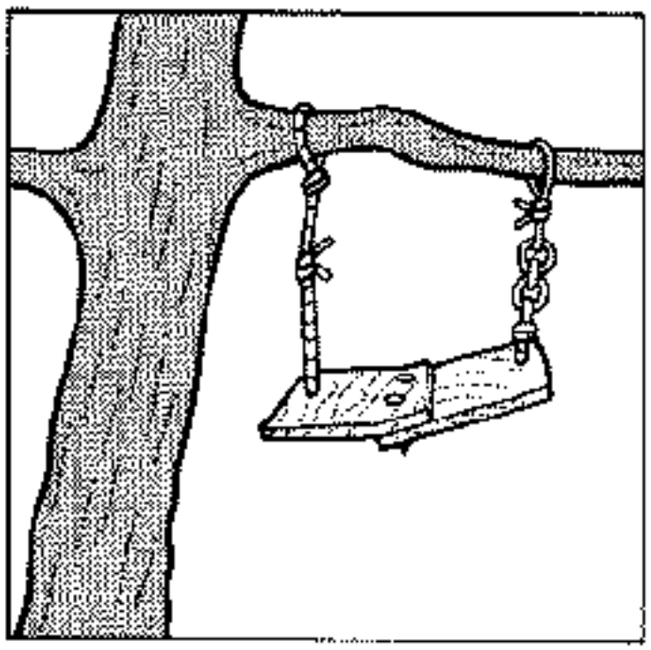
Design group's initial design



Corp. Product Architecture's modified design



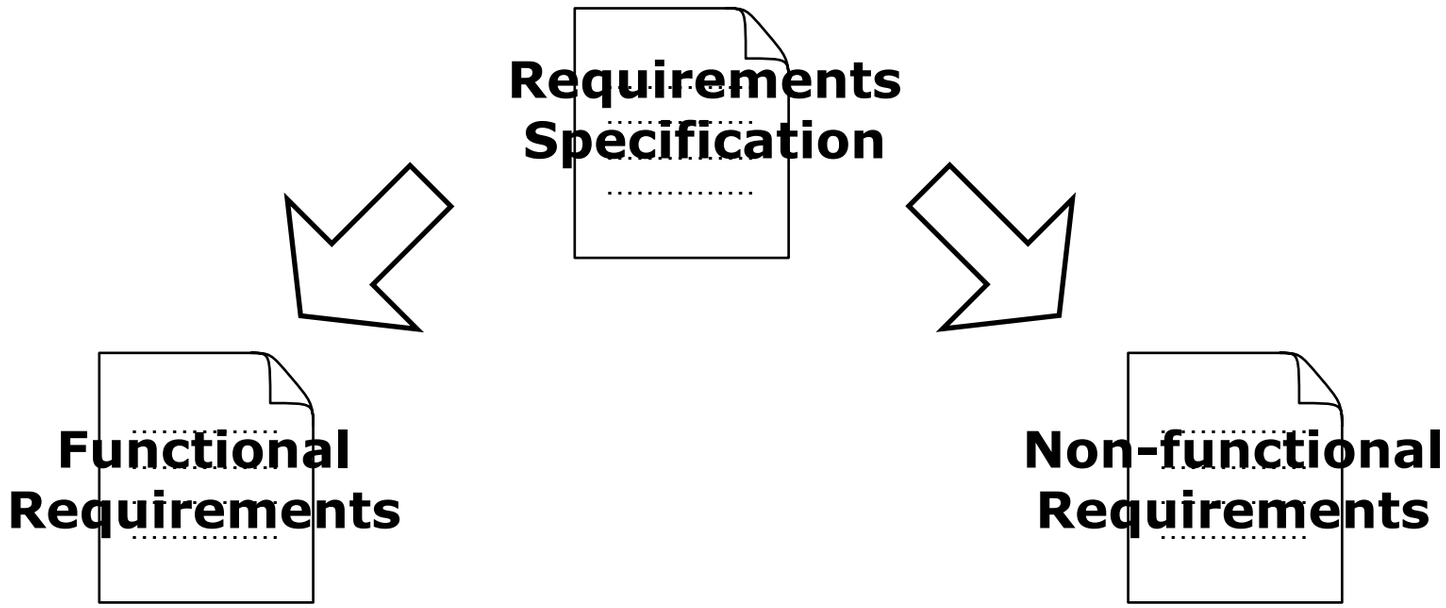
Pre-release version



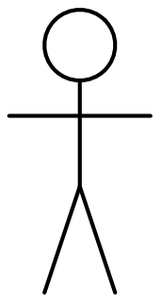
General release version

Numerous stakeholders & Limited resources

Where Requirements?



- functionality as demanded by the end users



- constraints placed on the global system or the development process.
- quality attributes, such as performance, user-friendliness, maintainability, …

What are Use Cases?

- Use Case
  + A use case describes outwardly visible requirements of the system
  + A use-case is a generic description of an entire transaction executed to achieve a *goal* (= the use case goal) and involving several *actors*.

- Actors
  + Actors have responsibilities
  + To carry out responsibilities, an actor sets goals
  + Primary actor (= stakeholder) has unsatisfied goal and needs system assistance
  + Secondary actor provides assistance to satisfy the goal

- Scenario
  + Scenario = an instance of a use-case, showing a typical example of its execution
    - Use case = Primary “success” scenario and secondary “alternative” scenarios
    - Scenario shows how objects interact to achieve the use case goal = UML Sequence diagrams & Collaboration Diagrams

Kinds of Use Cases

There is not a “one size fits all”: use cases depend on your purpose

- Scope
  + Brain-storm mode vs. full-fledged detailed specification

- Intended Audience
  + end-user vs. system development team vs. internal documentation

- Granularity
  + summary vs. detailed; overall system function vs. specific feature
  + Brief Use Case — Casual Use Case — Fully Dressed Use case

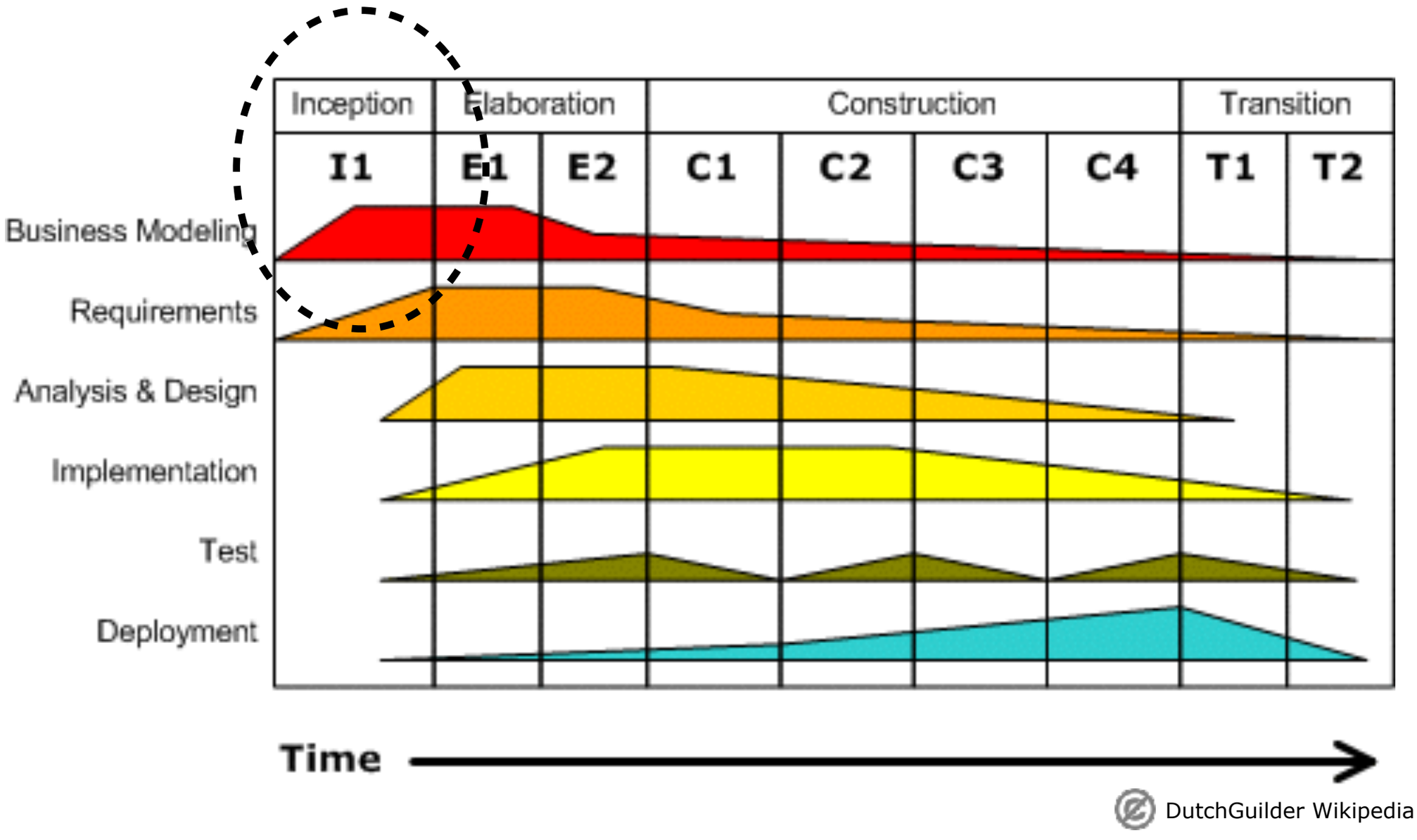
- Black-Box vs. White-Box
  + with or without knowledge about (business) processes used to achieve goal

Depending on your purpose some kind of Use Case Template is selected

Unified Process: Inception

Use cases work well when starting an iterative/incremental process!

+ e.g. *inception* & elaboration phase in Unified Process



DutchGuilder Wikipedia

Inception: System Scope

+ During inception you must define the system's scope
  - used to decide what lies inside & outside the system

- Scope
  + should be *short*
    - (1 paragraph for small projects;
      1/2 a page for mid-size projects;
      2-3 pages for large projects)
        * long statements are not convincing

  + should be *written* down
    - later reference when prioritizing use cases

  + should have end-user *commitment* (*)
    - end-user involved in writing
        * formally approved by a project steering committee

(*) The difference between "involvement" and "commitment"? In a Ham and Egg Breakfast...the chicken is involved and the pig is committed!

System Scope: Example

- (Example from [Schn98])

  + “We are developing order-processing software for a mail-order company called National Widgets, which is a reseller of products purchased from various suppliers.
    - Twice a year the company publishes a catalogue of products, which is mailed to customers and other interested people.
    - Customers purchase products by submitting a list of products with payment to National Widgets. National Widgets fills the order and ships the products to the customer’s address.
    - The order-processing software will track the order from the time its is received until the product is shipped.
    - National Widgets will provide quick service. They should be able to ship a customer’s order by the fastest, most efficient means possible.”

Evaluate the previous scope description

1. Summarise the purpose of the system in a single word

2. What quality criteria are important?

3. What is clearly outside scope?

4. (Technical) opportunities to improve?

Analyzing the Example

- The previous example of a system scope description is
  + short (1/2 a page)
    * quick assessment of what's the system supposed to do
  + goal-oriented (track orders)
    * open for various solutions
  + includes criteria (*quick* service, track *all* of the ordering process, …)
    * will be used to evaluate whether we accomplished the goals
  + provides context
    - National Widgets is reseller ⇒ *external* suppliers & shipment
    * the system will not solve everything, some problems are out of scope

- … and very importantly
  + imperfect (*twice* a year? *on-line* catalogue?)
    - may be improved when understanding increases
    - … but goal and main criteria should not change once approved

Inception: Risk Factors

- During inception you must identify the project's *risk factors*
  + you do not have control over the system's context and it will change
  + projects never go according to plan
    > identify potential problems early (… including wild success)
- Example

Context	Risk Factors	Impact	Likely	Urgency
Competitors	Time to market (too late/too early)			
Market trends	More internet at home			
Potential disasters	Suppliers don't deliver on time			
	System is down			
Expected users	Too many/few users			
Schedule	Project is delivered too early/too late	<< Risky Path (Project Management)		
Technology	Dependence on changing technology			
	Inexperienced team			
	Interface with legacy systems			

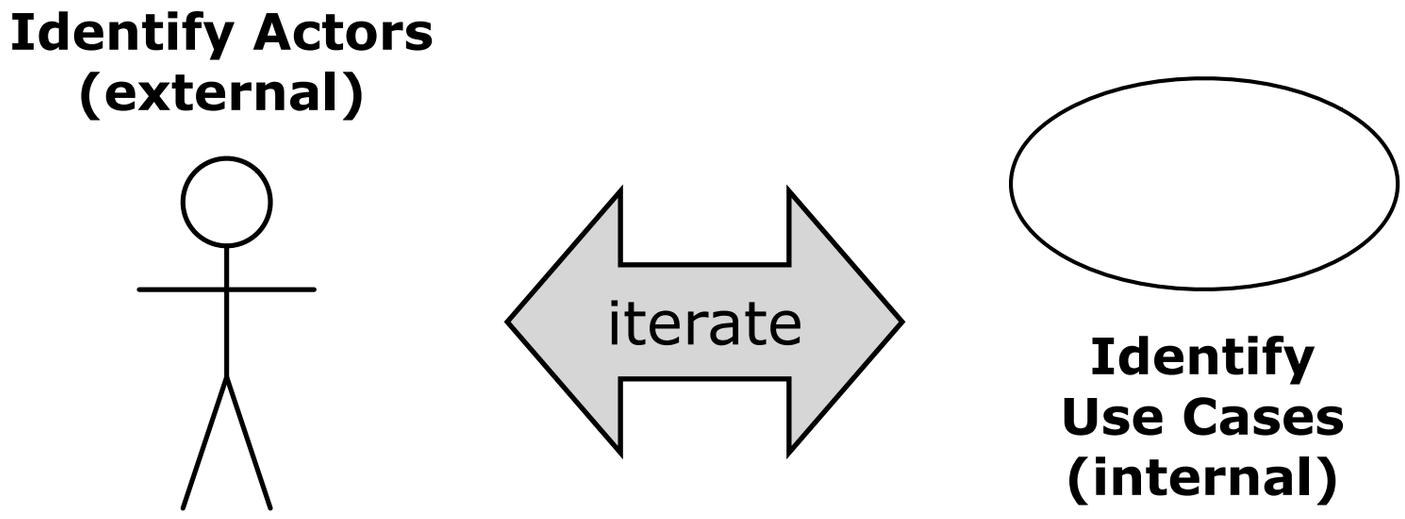
Inception: System Boundaries

During inception you must specify the system boundaries

- what functionality is *internal* to the system (= use cases)
- what functionality is *external* but necessary for internal functionality (= actors)

  \+ ⇒ the distinction is often not as clear as you would like it

  \+ ⇒ iterate: identifying actors + identify use cases



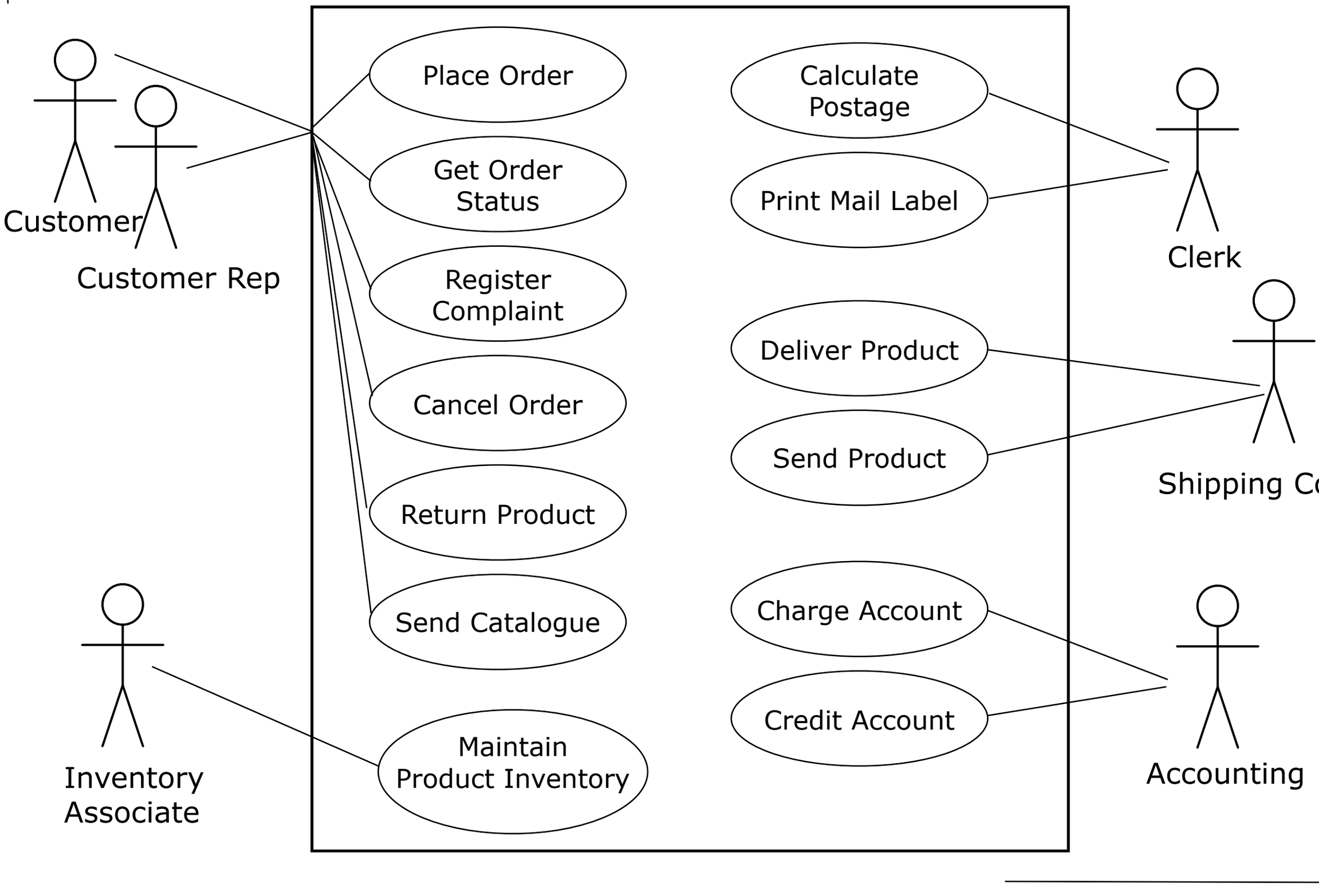
At least one actor must benefit from the use case (i.e. sees the use case value). The corresponding stakeholder will argue to keep the use case in the requirements!

Identifying Actors & Use cases

- Actors
  + Who uses the system?
  + Who installs the system?
  + Who starts up/shuts down the system?
  + Who maintains the system?
  + What other systems use this system?
  + Who provides information to this system?
  + Does anything happen automatically at a preset time?
- Use Cases
  + What functions will the actor want from the system?
  + What actors will create, read, update, or delete information stored inside the system?
  + Does the system need to notify actors about changes in its internal state?
  + Who gets information from this system?
  + Are there any external events the system must know about?
  + What actor informs the system about those events?

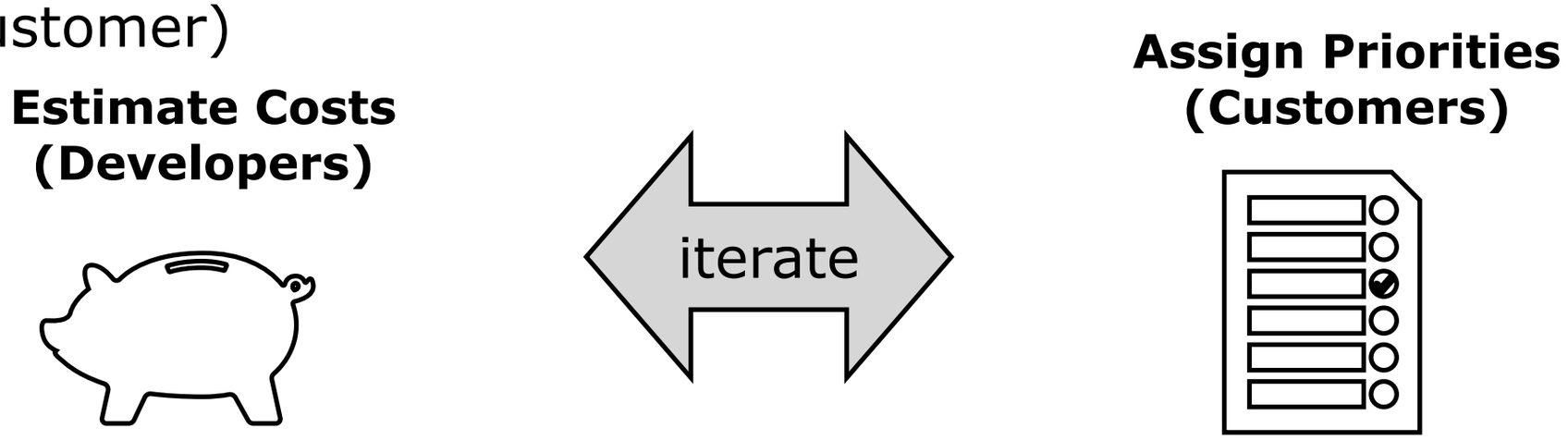
Above questions may help during the identification process.

Example: Actors & Use cases



Inception: Project Plan

- During inception you must specify the project plan
  = when to develop which use case
  + Includes intermediate milestones
  + based on Scope Definition & Risk Factors
  + may result in splitting/merging use cases
  + negotiate: estimate costs (=developer) + assign priorities (=customer)

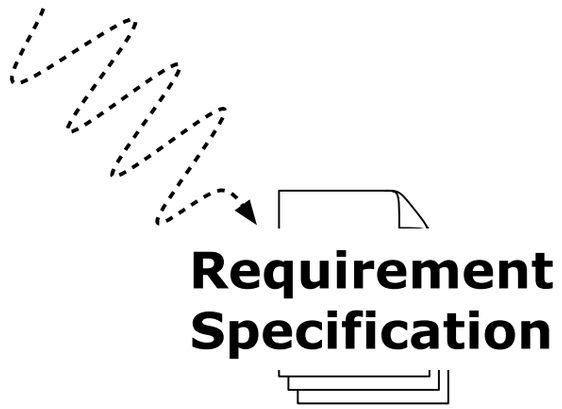


- Good negotiations obey 2 strict rules
  + *Developers estimate cost*; customers do not interfere.
    > Schedule slips are the responsibility of development team.
  + *Customers assign priorities*; developers do not interfere.
    > Deciding where the money is spent is the customers responsibility.

Terminating the Inception Phase

After inception the requirements specification consists of

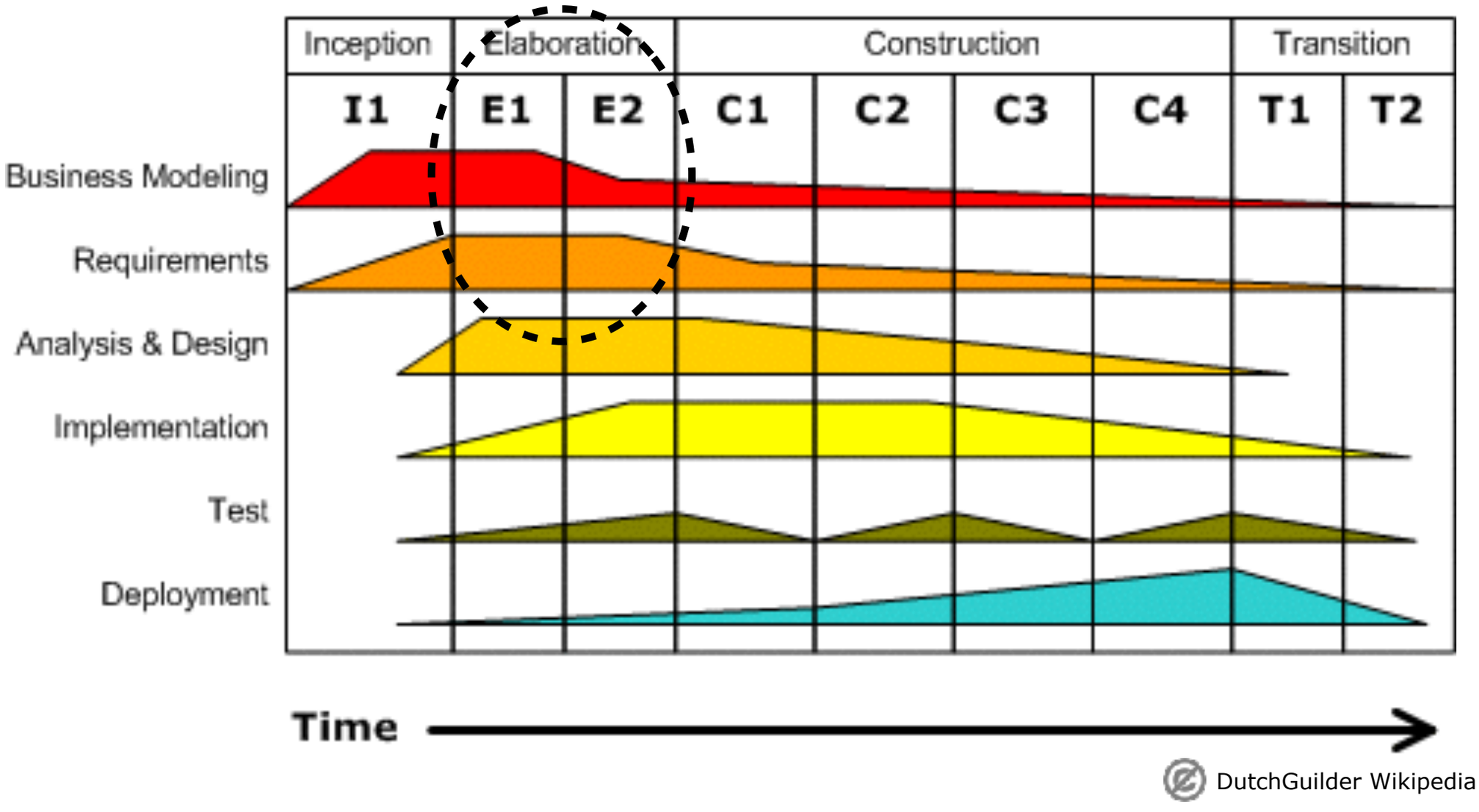
- Scope definition
  + Short description involving goals, criteria, context
- Risk Factors
  + Events that may cause problems during project
- Actors
  + Represent the various stakeholders in the project
- Use cases
  + Represent transactions; valuable for at least one actor
- Project Plan
  + For each use case
    - Cost estimate (assigned by development team)
    - Priority (assigned by customers)
  + Time plan including intermediate milestones



Formal approval by project steering committee

Unified Process: Elaboration

Use cases work well when starting an iterative/incremental process!
+ e.g. *inception* & elaboration phase in Unified Process



DutchGuilder Wikipedia

Elaboration: Primary & Secondary Scenarios

During elaboration you must refine the use cases via scenarios

- Scenario is one way to realize the use case
  - From the actors point of view!
- = List of steps to accomplish the use case goal

- Primary "success" scenario
  + = Happy day scenario
  + Scenario assuming everything goes right
    (i.e., all input is correct, no exceptional conditions, …)
- Secondary "alternative" scenarios
  + Scenario detailing what happens during special cases
    (i.e., error conditions, alternate paths, …)

Example: Place Order Scenario (1/2)

USE CASE 5	Place Order
Goal in Context	Customer issues request by phone to National Widgets; expects goods shipped and to be billed.
Scope & Level	Company, Summary
Preconditions	National Widgets has catalogue of goods
Success End Condition	Customer has goods, we have money for the goods.
Failed End Condition	We have not sent the goods, Customer has not spent the money.
Primary Actors	Customer, Customer Rep, Shipping Company
Secondary Actors	Accounting System, Shipping Company
Trigger	Purchase request comes in.

DESCRIPTION

Step	Action
1.	Customer calls in with a purchase request.
2.	Customer Rep captures customer info.
3.	WHILE Customer wants to order goods.
3.1.	Customer Rep gives Customer info on goods, prices, etc.

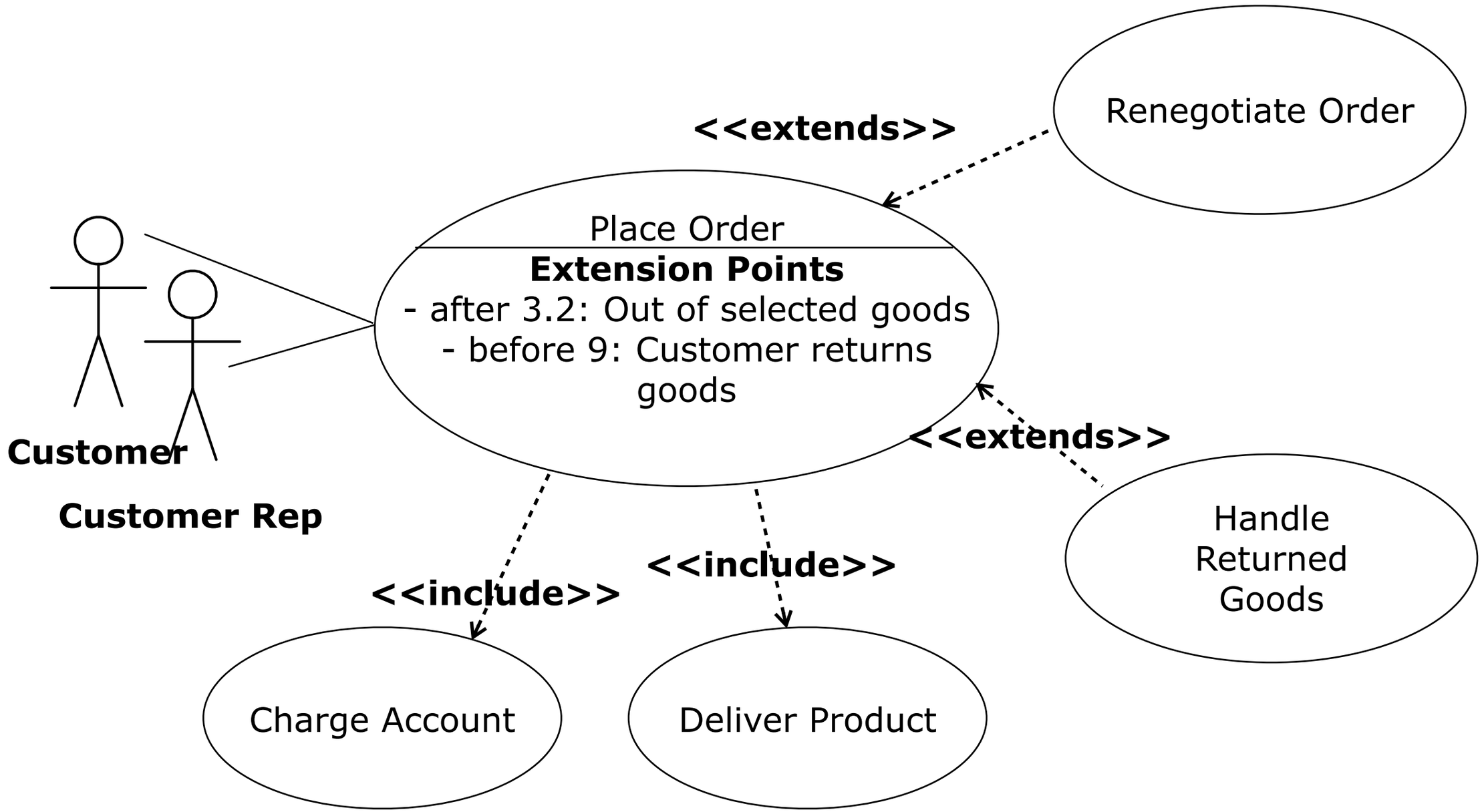
Example: Place Order Scenario (2/2)

3.2.	Customer selects good to add to order list.
4.	Customer approves order list.
5.	Customer supplies payment details.
6.	Customer Rep creates order.
7.	Customer Rep requests Accounting System to Charge Account.
8.	Customer Rep requests Shipping Company to Deliver Product.
9.	Customer pays goods.
Branch	SUBVARIATIONS
1.	Customer may use: (a) phone in, (b) fax in, (c) use web order form.
4.	Customer may pay via: (a) credit card; (b) cheque; (c) cash.
Branch	ALTERNATIVE PATHS
any	Customer may cancel transaction.
Branch	EXTENSIONS
After 3.2	Out of selected good: 3.2.a. Renegotiate Order (Use case 44).
Before 9	Customer returns goods: 9a. Handle returned goods (Use case 45).

Place Order Use Case Diagram

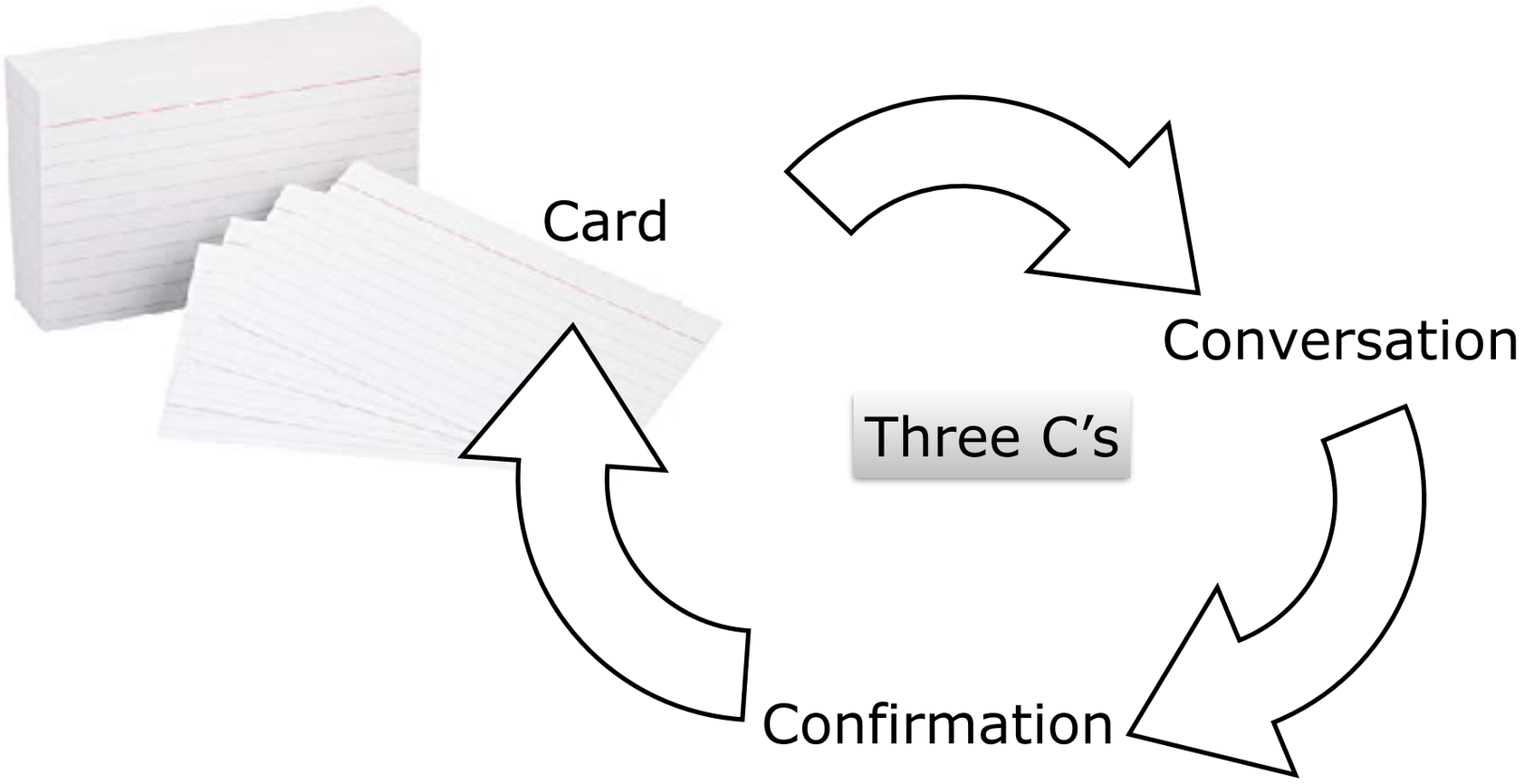
Stereotypes <<extends>> and <<include>> to specify use case relationships.

- Beware the direction of the arrows; it specifies change dependencies!



Scrum: User Stories

Agile Manifesto: We value customer collaboration over contract negotiation.



Behaviour Driven (User Stories)

Template

As a <user role>
I want to <goal>
so that <benefit>.

- …
- …
- …
-

Conditions of Satisfaction

Example

As a *clerk*
I want to *calculate stampage*
so that *goods get shipped fast*.

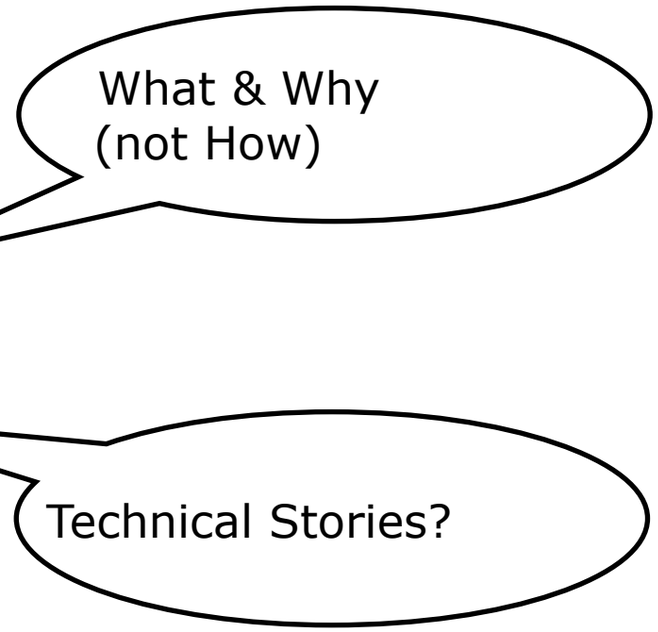
- Verify with nearby address
- Verify with overseas address
- Verify with parcels <= 1kg
- Verify with fragile parcel

As an *inventory associate*
I want to *minimise stock*
so that *we save warehouse costs*.
(See book of Müller for heuristics)

- References / explanations allowed
  + Conversation
- Coarse grained
  + break down in smaller chunks

INVEST Criteria

I	Stories should be *independent* of another and should not have dependencies on other stories
N	*Negotiable*: Too much detail on story limits conversation with the customer
V	Each story has to be of *value* to the customer
E	Stories should be small enough to *estimate*
S	Stories should be *small* enough to be completed in one iteration
T	*Testable*: Acceptance criteria should be available



Independent vs. Interdependent

I	Stories should be *independent* of another and should not have dependencies on other stories

Desirable but …
not as easy as it seems!

- Some user stories depend on one another for technical reasons.
  + e.g.: Storing items in a database before one can retrieve.
    > Work-around: dummy records
- Early stories take longer to implement
  + Creating the technical infrastructure
- Circular dependencies are a nightmare
  + Split the story in smaller chunks

Technical Stories

Technical stories express non-functional requirements in story form.

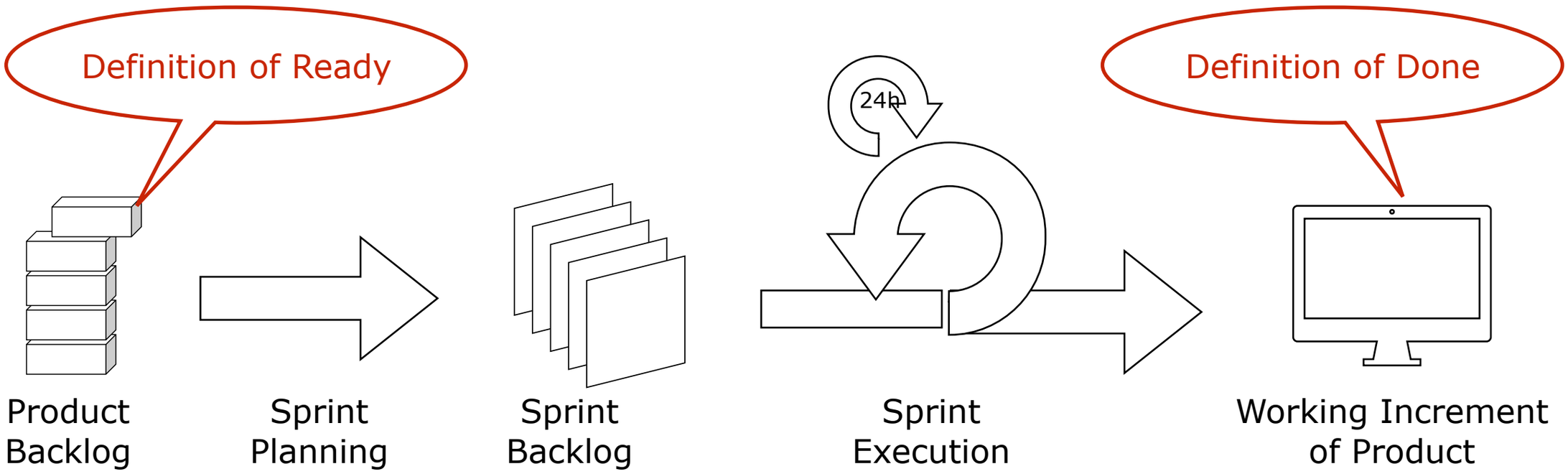
As a *developer*
I want to *migrate to Oracle version 3.2*
so that *we are not operating on a version that soon will retire*.

As a *security engineer*
I want to *ensure that HTTP, Radius SecureID, and LDAP authentication protocols are adhered to*
so that *we avoid backdoor attacks.*

Value?

- **Educate your product owners**
- **Calculate risk (Business case)**

Two sides of the same coin

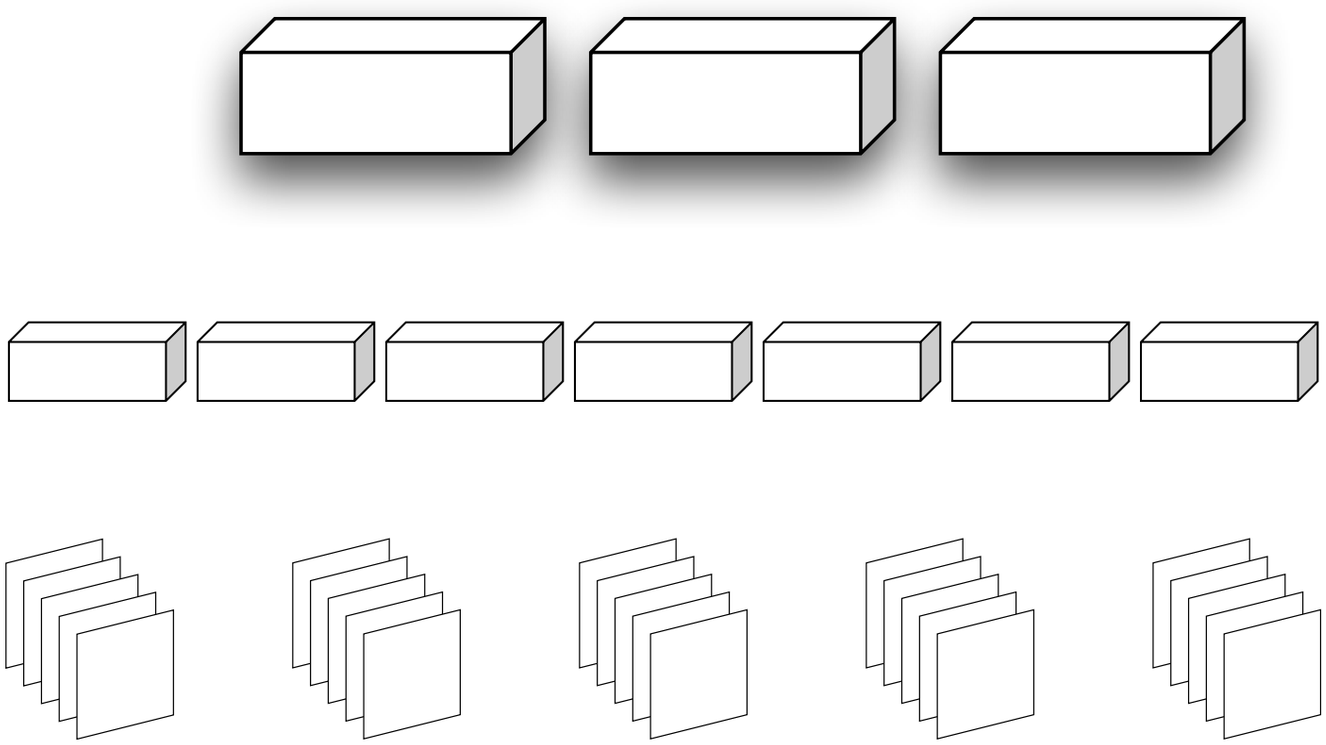


Definition of Ready

definition of ready = a checklist of the properties that must be satisfied before an item in the product back log can be moved into a sprint.

Example “Definition of Ready” checklist
- ✓ Business value is clearly articulated.
- ✓ Details are sufficiently understood by the development team so it can make an informed decision as to whether it can complete the Product Backlog Item.
- ✓ Dependencies are identified and no external dependencies would block the Product Backlog Item from being completed.
- ✓ Team is staffed appropriately to complete the Product Backlog Item.
- ✓ The Product Backlog Item is estimated and small enough to comfortably be completed in one sprint.
- ✓ Acceptance criteria are clear and testable.
- ✓ Performance criteria, if any, are defined and testable.
- ✓ Scrum team understands how to demonstrate the Product Backlog Item at the sprint review.

Product Backlog — Level of Detail

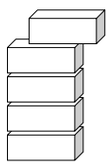
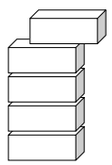
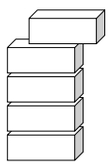


Epic	Months	Bigger than a release
Features	Weeks	Bigger than a sprint
Sprintable Stories	Days	Sprint Ready

CAPSTONE PROJECT

Product Roadmap (a.k.a. Release Roadmap)

Product Roadmap: communicates the incremental nature of how the product will be built and delivered over time, along with the important factors that drive each individual release.

	Q3—2020	Q4—2020	Q1—2021	Q2—2021
Market Map	Market Study	Advertising	Launch	
Market Events			CEBIT Fair	
Feature Map	…	…	…	
Release Plan				
Release Schedule	0.5	0.7	1.0	

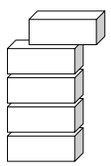
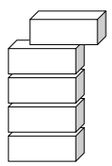
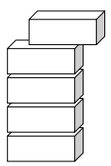
Epic

Minimum Viable Product

New

A minimum viable product (MVP) is a version of a product with just enough features to be usable by early customers who can then provide feedback for future product development.

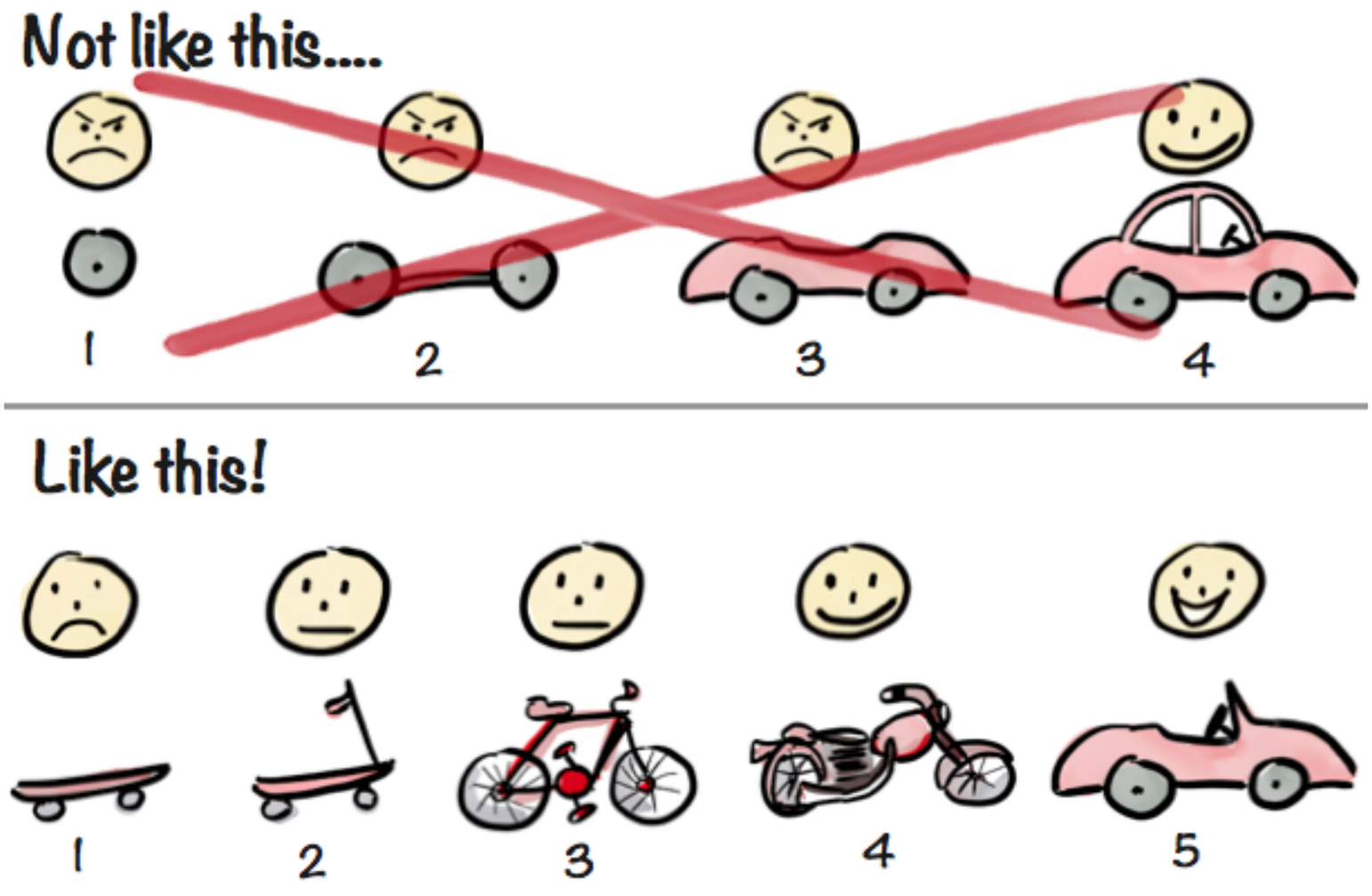
> First milestone for start-ups!

	Q3—2020	Q4—2020	Q1—2021	Q2—2021
Market Map	Market Study	Advertising	Launch	
Market Events			CEBIT Fair	
Feature Map	…	…	…	
Release Plan				
Release Schedule	0.5	0.7	1.0	

Minimum Viable Product: Iterations

New

Making sense of MVP (Minimum Viable Product) – and why I prefer Earliest Testable/Usable/Lovable
Posted on 2016-01-25 – 12:14 by Henrik Kniberg
https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp



Minimum Viable Product: Exercise

New

You an your neighbour are launching a start-up company with a mission to create a revolutionary new product.

- An on-line system for sharing course material in a post covid era (a.k.a. “Blackboard on steroids”)

Which iterations for your minimum viable product do you see?

Failure Mode and Effects Analysis (Example)

	Potential Failure Mode	Potential Effects of Failures	Severity	Potential Causes of Failures	Current Process Control	Occurrence (± Likelihood)	Detection (± Urgency)	Critical (± Impact)	Risk Priority Number	Recommended Actions
Function: Dispense Fuel										
	Does not dispense fuel	- Customer Dissatisfied - Discrepancy in bookkeeping	8	- Out of fuel - Machine jams - Power failure	- Out of fuel alert - Machine jam alert - none					
	Dispense too much fuel	- Company loses money - Discrepancy in bookkeeping	8	- Sensor defect - Leakage	- none - pressure sensor					
	Takes too long to dispense fuel	- Customer annoyed	3	- Power outage - Pump disrupted	- none - none					

Slide Adapted from Intro

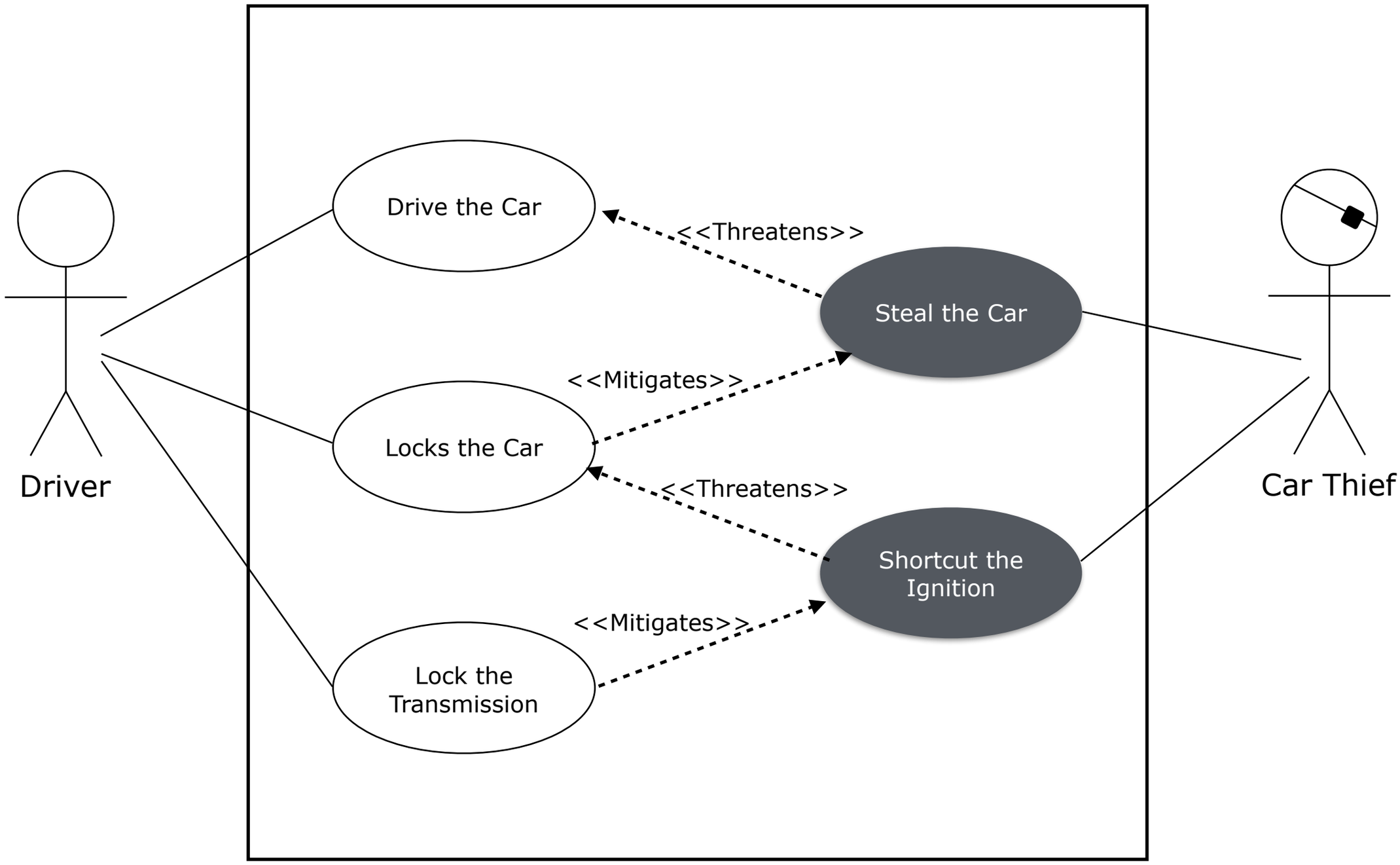
We must be able to express *"can never happen"* scenarios

Misuse Cases

- = a use case from the point of view security of an actor hostile to the system under design.
  + Results from a Failure Mode and Effects Analysis (FMEA)

Adds extra items to a use case diagram
  + Misuse case (coloured black)
  + Negative actor (marked somehow)
  + “Threatens” relationship
    - Between misuse case and ordinary use case
    - ≈ Potential causes of failures (FMEA analysis)
  + “Mitigates” relationship
    - Between misuse case and ordinary use case
    - ≈ Current Process Control (FMEA analysis)

Misuse Case (Example)



© Adapted from Ian Alexander, "Misuse Cases: Use Cases with Hostile Intent"

Safety Stories

- = if satisfied, will prevent a hazard from occurring or reduce the impact of its occurrence

Extended template for Safety Stories (Easy Requirements Syntax — EARS)

- Ubiquitous: The <component name> shall <response>
- Event Driven: When <trigger> the <system name>
- State Driven: While <in a specific state>
  the <system name> shall <system response>
- State Option: Where <feature is included> the <system name> shall <system response>
- Unwanted Behavior: If <optional preconditions> <trigger>, then the <system name> shall <system response>

Safety Stories (Example)

Several variants of stories

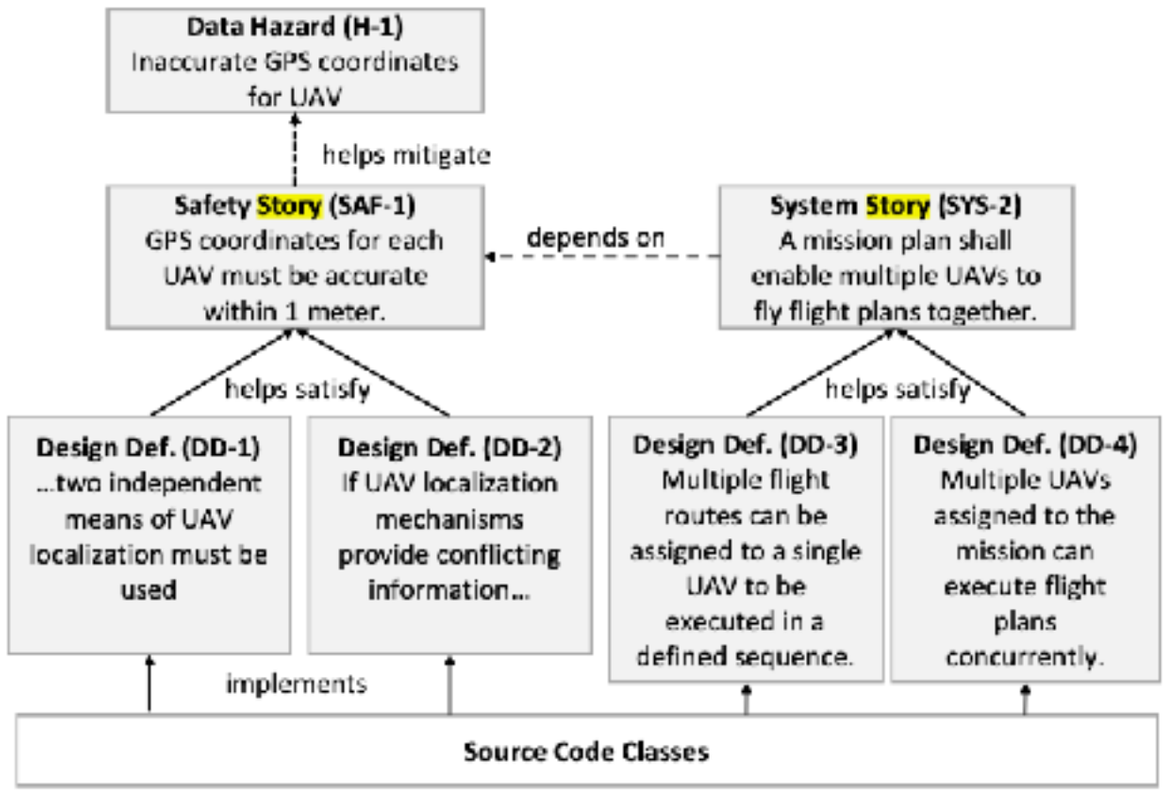
- System Story (SYS-1): A UAV (unmanned aerial vehicle) shall maintain a minimum separation distance from other UAVs at all times.

- Data Hazard (H-1): Inaccurate GPS (Global Positioning System) coordinates for UAV.
  + Failure Mode: GPS provides inaccurate readings.
  + Effect: Violation of minimum separation distance between two UAVs goes undetected, and UAVs collide in midair and then crash onto bystanders.
  + Level: Critical

- Safety Story (SAF-1): The GPS coordinates of each UAV must be accurate within one meter at all times.

- Design Definition (DD-1): When the Dronology system is deployed in an urban environment at least two independent means of UAV localization must be used.

Establish traceability links!



© Adapted from Jane Cleland-Huang et. al, "Discovering, Analyzing, and Managing Safety Stories in Agile Projects,"

Conclusion

Use cases / User Stories help you to specify good requirements because it is easier to make them … (a) understandable; (b) precise; and (c) open.

	Use Cases	User Stories
Understandable	Actors provide an end users perspective	<User Roles> provide an end users perspective
Precise	Scenarios are sufficiently detailed to test (path coverage)	"Conditions of Satisfaction" ⇒ test scenarios
Open	Actors perspective emphasizes the what (and much less the how)	The INVEST criteria

But there is no guarantee, it still requires

- close interaction with various *stakeholders*
- iteration to improve earlier misconceptions
- … and lots of hard work.

Requirements & Correctness

- Are we building the system right?
  + Good requirements will help to validate solution against requirements.
    - Testing
      Writing black box regression tests should be easy.
  + … however, step to system design (architecture)
    + detailed design (objects) is hard.
    - Use cases & User Stories tend to result in hard to maintain systems

- Are we building the right system?
  + Good requirements should be easy to verify
    - Understandable & precise
  + … however
    - we may omit requirements
      * Completeness is not guaranteed!
      * Complementary Failure Mode and Affect Analysis (FMEA)
    - Focus on scenarios restricts evolving requirements
      * Requirements should specify “what” not “how”

Requirements & Traceability

- Requirements ⇔ System

  + Via proper naming conventions
    - … including names of regression tests

- Requirements ⇔ Project Plan

  + Use cases & User Stories form good milestones
    - Less so for misuse cases and safety stories
  + Estimating development effort is feasible
    - Balancing Numerous stakeholders against Limited resources

Use cases form a good base for negotiating the project plan.
User Stories (epics) form a good basis for negotiating the product roadmap.

Summary(i)

- You should know the answers to these questions
  + Why should the requirements specification be understandable, precise and open?
  + What's the relationship between a use case and a scenario?
  + Can you give 3 criteria to evaluate a system scope description? Why do you select these 3?
  + Why should there be at least one actor who benefits from a use case?
  + Can you supply 3 questions that may help you identifying actors? And use cases?
  + What's the difference between a primary scenario and a secondary scenario?
  + What's the direction of the <<extends>> and <<includes>> dependencies?
  + What is the purpose of technical stories in scrum?
  + List and explain briefly the INVEST criteria for user stories.
  + Explain briefly the three levels of detail for Product Backlog Items (Epic, Features, Stories).
  + *What is a minimum viable product?*
  + Define a misuse case.
  + Define a safety story.

- You should be able to complete the following tasks
  + Write a requirements specification for your bachelor capstone project.

CAPSTONE PROJECT

Summary(ii)

- Can you answer the following questions?
  + Why do use cases fit well in an iterative/incremental development process?
  + Why do we distinguish between primary and secondary scenarios?
  + What would you think would be the main advantages and disadvantages of use cases?
  + How would you combine use-cases to calculate the risky path in a project plan?
  + Do use-cases work well with agile methods? Explain why or why not.
  + Can you explain the use of a product roadmap in scrum?
  + Choose the three most important items in your “Definition of Ready” checklist. Why are these most important to you?
  + Can you relate scrum user stories to some of the principles in the Agile Manifesto?
  + How would you turn an FMEA analysis into a misuse case diagram?
  + Elaborate on the relationship between an FMEA analysis and the variants of safety stories.