Hugbúnaðarverkefni 1 / Software Project 1

4. Use Cases

HBV501G – Fall 2018

Matthias Book

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI- OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz Prep

- Please prepare a scrap of paper with the following information:

- ID: _____@hi.is Date: _____
- a) _____
- b) _____
- c) _____
- d) _____
- e) _____

- During class, I'll show you questions that you can answer very briefly
  - No elaboration necessary
- Hand in your scrap after the quiz

- All questions in a quiz weigh same
- All quizzes (ca. 10 throughout semester) have the same weight
  - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam

Quiz #2 Solution: Vision and Scope Document

- **Business requirements**
  - …
  - 3. Business objectives (g)
  - …
  - 5. Vision statement (f)
  - 6. Business risks (c)
  - …
- **Scope and limitations**
  - 8. Major features (d)
  - 9. Scope of initial release (e)
  - 10. Scope of subsequent releases (b)
  - 11. Limitations and exclusions (a)
- **Business context**
  - 12. Stakeholder profiles (h)
  - …

- **Which section answers which question?**
  - a) What are we *not* going to do?
  - b) What can be rolled out later?
  - c) What could jeopardize the product's success?
  - d) What key things should the product be capable of?
  - e) What should be rolled out first?
  - f) What will the product accomplish for whom?
  - g) Which benefits do we expect out of this?
  - h) Who has what interest in the project?

Recap: Team Assignment 1

- By **Sun 23 Sep**, submit in Ugla:
  - A **Vision and Scope document** for your project
    - Following the template on slide 5 – most importantly, sections 1.5, 2.1
      - Use *brief* use case format for the 2-4 most important use cases in section 2.1
    - Write a paragraph on one 1.x section and one 2.x section of your choice as well
      - (“Convince your boss” – there are e.g. business considerations for games, too!)
- On **Thu 27 Sep**, present and **explain** your document to your tutor:
  - What considerations are behind your vision and scope document?
  - Why did you leave out certain sections of the template?
  - How did you come up with your use cases?
  - etc.
- **Grading criteria**
  - Precise vision statement in Sect 1.5 (25%)
  - Precise formulation of 2-4 key use cases in brief format in Sect. 2.1 (25%)
  - Plausible argument in chosen Sect. 1.x (25%)
  - Plausible argument in chosen Sect. 2.x (25%)

Update: General Assignment Format

- Required **artifacts** must
  - be produced by all team members together
  - be submitted in one PDF document by specified deadline in Ugla
  - contain your team number, the names and **HÍ e-mail addresses or kennitölur** of all team members
  - indicate who will present the assignment
- **Submissions** are due at 23:59 on Sundays
  - **No individual extensions** – missing artifacts receive a grade of 0!
  - But undefined grace period until whenever tutors download submissions from Ugla
  - Only the team member who will present must submit a document for the whole team
- The **presentation** must
  - be given by one representative of the team (a different one for each assignment)
  - be based on the submitted document (don't prepare extra slides)
  - take around 5-10 minutes (plus some questions asked by the tutor)

Recap: Vision and Scope Document Template

Assignment 1

1. **Business requirements**
   1. Background
   2. Business opportunity
   3. Business objectives
   4. Success metrics
   5. Vision statement
   6. Business risks
   7. Business assumptions and dependencies
2. **Scope and limitations**
   1. Major features
   2. Scope of initial release
   3. Scope of subsequent releases
   4. Limitations and exclusions
3. **Business context**
   1. Stakeholder profiles
   2. Project priorities
   3. Deployment considerations

- For [target customer(s)]
- who [need/opportunity],
- the [product name]
- is [product category]
- that [major capabilities, key benefit, compelling reason to buy/use].
- Unlike [primary competitive alternative, current system/process],
- our product [primary differentiation and advantage of new product].

- 2-4 in Brief Use Case format

- Plus another Section from 1.x and another Section from 2.x

Use Cases vs. User Stories

- Requirements can be expressed in many different ways. Most common:
  - Informal feature lists (e.g. "• Create account")
  - User Stories (e.g. "As a user, I want to create an account in order to save…"; → HBV401G)
  - Use Cases
    - Brief format (e.g. "The user enters a user name. The system checks... The user…"; → Slide 9)
    - Casual format (→ Slide 10)
    - Fully-dressed format (→ Slide 11ff)

- **User Story**
  - Very brief summary of a requirement that serves as a "handle" in product backlog, effort estimation, sprint planning, progress controlling to represent the respective activities & effort
  - Does not provide any explicit detail on how the feature will look & feel, which constraints it is exposed to, etc.
- **Use Case**
  - **Brief format:** Gives a precise idea of the interaction between user and system
  - **Casual format:** Describes the user interaction in normal and exceptional circumstances
  - **Fully-dressed format:** Provides explicit, detailed description of all aspects, alternatives and constraints that developers need to understand for a high-quality implementation

Recap: Use Cases

- **Official definition in the RUP:**
  - A use case is a **set of scenarios** where each scenario is a **sequence of actions** a system performs that yields an **observable result** of **value** to a particular **actor**.

- **Key writing challenge: Capturing the requirements precisely**
  - Understand and formulate the essence of the requirement
  - Don’t be too detailed, don’t be too high-level
  - Don’t combine several alternatives in one scenario
  - Don’t forget to consider and describe alternate scenarios
  - Don’t interpolate what you’re not sure about – ask!
  - Don’t preempt technical design decisions

Recap: Use Case Formats

- **Brief format**
  - Terse one-paragraph summary, usually of the main success scenario
  - Written during early requirements analysis to get a quick sense of subject and scope
- **Casual format**
  - Multiple informal paragraphs that cover various scenarios
  - Typically including at least the main success scenario and most relevant alternate scenarios
  - Written during early requirements analysis to get a quick sense of subject and scope
- **Fully dressed format**
  - All steps and variations described in detail, with supporting sections on preconditions etc.
  - During first requirements workshop, about 10% of architecturally significant, high-value use cases are written out in this detail; rest may follow in further iterations as needed.
- **[UML use case diagrams]**
  - Secondary, supplementary illustrations only – essential information is contained in the text!

Recap: Brief Use Case Format

- In the Unified Process, we **discover and record functional requirements** by writing **text stories of an actor using a system to meet goals**.

- **Simple example:** “Process Sale” use case **(brief format)**
  - A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates the inventory. The customer receives a receipt from the system and leaves with the items.

- **Emphasis on users’ goals and perspectives:**
  - Who is using the system?
  - What are their typical scenarios of use?
  - What are their goals?

➢More user-centric than just coming up with a list of system features

Recap: Casual Use Case Format

“Handle Returns” use case

- **Main success scenario**
  - A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item. […]
- **Alternate scenarios**
  - If the customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.
  - If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).
  - If the system detects failure to communicate with the external accounting system, […]

Fully Dressed Use Case Format Template

1. **Use case name**
   - What are these scenarios about? (Start with a verb.)
2. **Scope**
   - What is the system under design?
3. **Level**
   - Is this use case describing a user's goal (top level) or a sub-function of some other use case?
4. **Primary actor**
   - Who calls on the system to deliver its services?
5. **Stakeholders and interests**
   - Who cares about this use case, and what do they want?
6. **Preconditions**
   - What must be true at the beginning of the scenario (and worth telling the reader)?
7. **Success guarantee**
   - What must be true on successful completion (and worth telling the reader)?

Fully Dressed Use Case Format Template

8. **Main success scenario**
   - What is the typical, unconditional “happy path” scenario of success?
9. **Extensions / alternate scenarios**
   - What are alternate scenarios of success or failure?
10. **Special requirements**
    - What are related non-functional requirements?
11. **Technology and data variations list**
    - What varying input/output methods and data formats should we be aware of?
12. **Frequency of occurrence**
    - How often does this use case occur? (Influences investigation, timing, priority, testing…)
13. **Miscellaneous / open issues**
    - What open issues are there?

- Template can be tailored to your project’s needs, i.e. you may omit sections if sensible

Example: Fully Dressed Use Case Format

- **Name:** UC1: Process Sale
- **Scope:** NextGen POS application
- **Level:** User goal
- **Primary actor:** Cashier
- **Stakeholders and interests:**
  - **Cashier:** Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from salary.
  - **Salesperson:** Wants sales commissions updated.
  - **Customer:** Wants purchase and fast service with minimal effort. Wants easily visible display of entered items and prices. Wants proof of purchase to support returns.
  - **Manager:** Wants to be able to quickly perform override operations, and easily debug Cashier problems.
  - **Government Tax Agencies:** Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
  - **Company:** Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g. remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.
  - **Payment Authorization Service:** Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
- **Preconditions:** Cashier is identified and authenticated.
- **Success guarantee:**
  - Sale is saved.
  - Tax is correctly calculated.
  - Accounting and inventory are updated.
  - Commissions recorded.
  - Receipt is generated.
  - Payment authorization approvals are recorded.

Example: Fully Dressed Use Case Format

- **Main success scenario:**
  1. Customer arrives at 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. Price calculated from a set of price rules.

  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.
  8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update Inventory).
  9. System presents receipt.
  10. Customer leaves with receipt and goods (if any).

- **Alternate scenarios:**
  - (a) At any time, Manager requests an override operation:
    1. System enters Manager-authorized mode.
    2. Manager or Casher performs one Manager-mode operation, e.g. cash balance change, resume a suspended sale on another register, void a sale, etc.
    3. System reverts to Cashier-authorized mode.
  - (2-4) Customer tells Cashier they have a tax-exempt status (e.g. seniors, native peoples)
    1. Cashier verifies, and enters tax-exempt status.
    2. System records status (which it will use during tax calculations).
  - (3a) Item requires manual category and price entry (e.g. flowers, gift cards).
    1. Cashier enters special manual category code, plus the price.
  - (3b) Item requires weighing.
    1. Cashier places item on scale.
    2. System weighs item, records weight and calculates price.
  - […]

Example: Fully Dressed Use Case Format

- **Further extensions:**
  - System failure
  - Resume suspended sale
  - Invalid item ID
  - Bundled items
  - Remove item from sale
  - Cancel sale
  - Suspend sale
  - Discount item
  - Product rebate
  - Apply customer credit
  - Pay by cash
  - Pay by credit card
  - Pay (in part) with coupons
  - […]

2. Managers performs override.
3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it)

2c. Cashier performs Find Product Help to obtain true item ID and price.
2d. Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above).

3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
1. Cashier can enter item category identifier and the quantity.

3c. Item requires manual category and price entry (such as flowers or cards with a price on them):
1. Cashier enters special manual category code, plus the price.

3-6a: Customer asks Cashier to remove (i.e., void) an item from the purchase: This is only legal if the item value is less than the void limit for Cashiers, otherwise a Manager override is needed.
1. Cashier enters item identifier for removal from sale.
2. System removes item and displays updated running total.
2a. Item price exceeds void limit for Cashiers:
1. System signals error, and suggests Manager override.
2. Cashier requests Manager override, gets it, and repeats operation.

3-6b. Customer tells Cashier to cancel sale:
1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:
1. System records sale so that it is available for retrieval on any POS register.
2. System presents a "suspend receipt" that includes the line items, and a sale ID used to retrieve and resume the sale.

4a. The system supplied item price is not wanted (e.g., Customer complained about something and is offered a lower price):
1. Cashier requests approval from Manager.
2. Manager performs override operation.
3. Cashier enters manual override price.
4. System presents new price.

5a. System detects failure to communicate with external tax calculation system service:
1. System restarts the service on the POS node, and continues.
1a. System detects that the service does not restart.
1. System signals error.
2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):
1. Cashier signals discount request.
2. Cashier enters Customer identification.
3. System presents discount total, based on discount rules.

5c. Customer says they have credit in their account, to apply to the sale:
1. Cashier signals credit request.
2. Cashier enters Customer identification.
3. Systems applies credit up to price=0, and reduces remaining credit.

6a. Customer says they intended to pay by cash but don't have enough cash:
1. Cashier asks for alternate payment method.
1a. Customer tells Cashier to cancel sale. Cashier cancels sale on System.

7a. Paying by cash:
1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

7b. Paying by credit:
1. Customer enters their credit account information.
2. System displays their payment for verification.
3. Cashier confirms.
3a. Cashier cancels payment step:
1. System reverts to "item entry" mode.
4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.
4a. System detects failure to collaborate with external system:
1. System signals error to Cashier.
2. Cashier asks Customer for alternate payment.
5. System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).
5a. System receives payment denial:
1. System signals denial to Cashier.
2. Cashier asks Customer for alternate payment.
5b. Timeout waiting for response.
1. System signals timeout to Cashier.
2. Cashier may try again, or ask Customer for alternate payment.
6. System records the credit payment, which includes the payment approval.
7. System presents credit payment signature input mechanism.
8. Cashier asks Customer for a credit payment signature. Customer enters signature.
9. If signature on paper receipt, Cashier places receipt in cash drawer and closes it.

7c. Paying by check…
7d. Paying by debit…

7e. Cashier cancels payment step:
1. System reverts to "item entry" mode.

7f. Customer presents coupons:
1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
1a. Coupon entered is not for any purchased item:
1. System signals error to Cashier.

9a. There are product rebates:
1. System presents the rebate forms and rebate receipts for each item with a rebate.

9b. Customer requests gift receipt (no prices visible):
1. Cashier requests gift receipt and System presents it.

9c. Printer out of paper.
1. If System can detect the fault, will signal the problem.
2. Cashier replaces paper.
3. Cashier requests another receipt.

Example: Fully Dressed Use Case Format

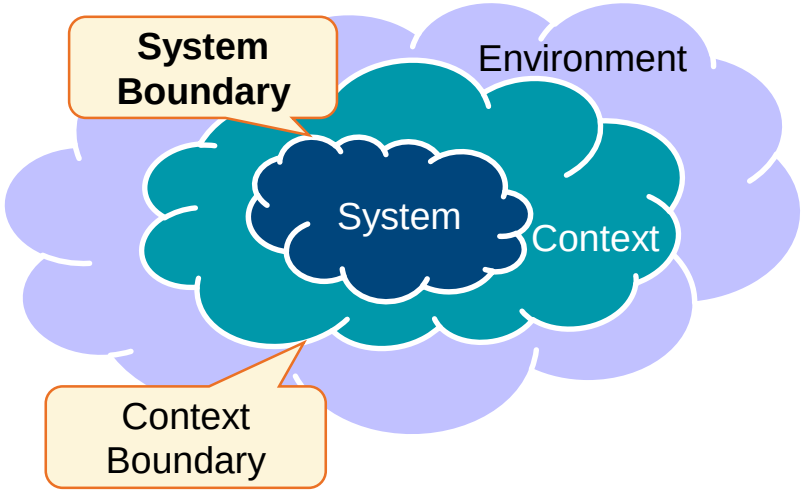
- **Special requirements:**
  - Touch screen UI on a large flat panel monitor. Text must be visible from 1 m.
  - Credit authorization response within 30 seconds 90% of the time.
  - Somehow, we want robust recovery when access to remote services such as the inventory system is failing.
- **Technology and data variations:**
  - (a) Manager override entered by swiping an override card through a card reader, or entering an authorization code via the keyboard.
  - (3a) Item identifier entered by car code laser scanner (if bar code is present) or keyboard.
  - (3b) Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
  - (7a) Credit account information entered by card reader or keyboard.
  - (7b) Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.
- **Frequency of occurrence:** could be nearly continuous
- **Open issues:**
  - What are the tax law variations?
  - Explore the remote service recovery issue.
  - What customization is needed for different businesses?
  - Must a cashier take their cash drawer when they log out?
  - Can the customer directly use the card reader, or does the cashier have to do it?

Finding Use Cases

Note: Not a linear process!
The steps influence each other and will be repeated in iterative-incremental fashion.

1. Choose the system boundary.

- Which aspects of the business domain will be covered/performed by your system?
- Which aspects of the business domain will influence/interact with your system?
- If you're not sure what's in the system, it may be easier to identify what is outside it.



2. Identify the primary actors.

- Who has goals fulfilled through using services of the system?
- Who performs business administrative tasks?
- Who performs technical administrative tasks?
- Beside human primary actors, are there external systems that use services of your system?
- Is "time" an actor because the system reacts to certain events (e.g. deadline expiry)?
- By whom is the system's lifecycle handled (starting, stopping, restarting, error conditions…)?

Finding Use Cases

Note: Not a linear process!
The steps influence each other and will be repeated in iterative-incremental fashion.

3. Identify the goals for each primary actor.

- Note: Actors have goals and use software to help satisfy them.
- Don't ask people: "What are the system's tasks?" or "What do you do?"
  - This will just reflect current solutions and procedures, which may not be optimal.
- Rather, ask first: "Who uses the system? What are their goals?"
- Ask actors next: "What are your goals whose results have measurable value?"
  - This will open up the vision for new and improved solutions, focus on adding business value, get to the core of what stakeholders want from the system.
- Alternatively/additionally: Consider…
  - which real-life events can occur
  - which actors are involved in them
  - what purpose/goal they serve

4. Define use cases that satisfy user goals.

- Write one use case per goal (including its success and failure scenarios).
- Exception: Combine distinct "create/retrieve/update/delete" goals for certain data X into one "manage X" use case

Testing Use Cases' Validity

Here, **"valid" means "valuable" or "justified"** rather than "correct".

("Is this a feature in its own right that merits individual description and consideration?")

- **Example: Which of these are valid use cases?**
  a) Buy an apartment
  b) Get a refund for a returned product
  c) Log in to a system
  d) Check a kennitala's syntactic validity
  e) Subscribe to an e-mail newsletter

- One might say these are all use cases on different levels of detail.
  - But what is helpful for the purpose of understanding what we need to build?

➢ **A valid use case describes scenarios that**
  - Create results that are of **value** to a particular stakeholder
  - Represent **elementary business processes**
  - Are so **complex** that they require several pages to be described in fully dressed format

Testing Use Cases' Validity

- **Boss test**
  - Would your boss be happy if you told him you performed this use case all the time?
  - Checks whether the use case describes actions that **create value**.
- **Size test**
  - Most use cases are on a **time scale** of a few minutes to an hour.
    - Neither something done in one small step, nor something taking several sessions or days to complete is likely a valid use case.
  - Make sure it's not just one step of a more complex use case.
    - If the fully dressed format is less than one **page** long, it's probably not a use case.
- **EBP test**
  - Does the use case represent an elementary business process (EBP)?
    - A task performed by one person in one place at one time,
    - in response to a business event,
    - which adds measurable business value
    - and leaves the data in a consistent state.

Testing Use Cases' Validity

- **Do not take the tests too literally – use common sense!**
- A task requiring two or three people can still be a valid use case, even if it violates the strict Elementary Business Process definition.
- A very complex or frequently recurring sub-task (e.g. "pay by credit card") can be reasonably written in its own use case, so it can be referred to from several other use cases even if it violates the Size or EBP Test.
- A feature such as "authenticate user" may not pass the Boss Test, but still require careful analysis that warrants description as a use case of its own.
- *The purpose of use cases is to **understand** what we need to build.*

Quiz #3: Testing Use Cases' Validity

- **Boss test**
  - Would your boss be happy if you told him you performed this use case all the time?
  - Checks whether the use case describes actions that **create value**.
- **Size test**
  - Most use cases are on a **time scale** of a few minutes to an hour.
    - Neither something done in one small step, nor something taking several sessions or days to complete is likely a valid use case.
  - Make sure it's not just one step of a more complex use case.
    - If the fully dressed format is less than one **page** long, it's probably not a use case.
- **EBP test**
  - Does the use case represent an elementary business process (EBP)?
    - A task performed by one person in one place at one time,
    - in response to a business event,
    - which adds measurable business value
    - and leaves the data in a consistent state.
- **Are the following valid use cases?**
  a) Buy an apartment
  b) Get a refund for a returned product
  c) Log in to a system
  d) Check a kennitala's syntactic validity
  e) Subscribe to an e-mail newsletter
  - Respond to each one with "yes" or "no".

Reality Check

- **Caution:** Written specifications and models can easily give the **illusion** of correctness and completeness.
  - However, in some places, these specs are virtually guaranteed to be flawed, ambiguous, missing critical information, misunderstood by readers…
- It is neither feasible nor economical to try to remedy this by a huge initial requirements and design effort.
  - Neither does it benefit a complex project to forgo requirements engineering altogether and rush straight to coding.

➢ **Use an iterative-incremental approach:** Key requirements, architectural baseline, more requirements, more implementation etc.

➢ **Strive for clear, precise language:** The writing process will prompt you to clarify your requirements.

Use Case Writing Guideline: Essential, UI-free Style

- **Describe the user's intentions and the system's responsibilities, rather than their concrete actions.**
- Do not include user interface details in the use case.
- Consider what is the "root goal" that the user is striving to accomplish.

Bad Example	Good Example
1. Administrator enters ID and password in dialog box (see Fig. 3). 2. System authenticates Administrator. 3. System displays "Edit Users" window (see. Fig. 4).	1. Administrator identifies self. 2. System authenticates identity.

No technology presumptions: Could use password, RFID chip, biometrics, …

- Helps to focus on the problem domain, rather than the solution domain
- Allows developers to come up with new solutions & adapt to different platforms

Use Case Writing Guideline: Terse, Active Voice

- **Write in crisp, terse style.**
- Avoid “noise” words.
- Abbreviations are fine, but have a central place in the document to define them.

- Make sure you don’t eliminate relevant content, and dependencies are clear.
  - Under which conditions does something happen?
  - Where does data go? Who depends on certain data?

- **Use active instead of passive voice.**

Bad Example	Good Example
1. The user’s identity is authenticated.	1. System authenticates user’s identity.

by whom?

Use Case Writing Guideline: Black-Box Style

- Do not describe the internal workings of the system, its components, or design.
- Focus on describing the system's responsibilities.
  - Note: This corresponds to object-oriented thinking:
    Software components have responsibilities and collaborate with other components,
    but don't expose to each other how exactly they fulfill their responsibilities.
- Describe **what** the system will do, but **not how** it will do it.

Bad Example	Good Example
1. The system writes the sale to a database… 2. The system generates an SQL INSERT statement for the sale…	1. The system records the sale.

To decide on this is design and implementation work, not requirements work.

Use Case Writing Guidelines: Actor-Goal Perspective

- **Write requirements focusing on the users/actors of a system, asking about their goals and typical situations.**
- Focus on understanding what the actor considers a valuable result.

- Avoid collecting plain “feature lists”, but consider who is using the system for what purpose.
  - Feature lists easily lead to “gold plating”, where every stakeholder tries to include their pet features, regardless of whether users need or want them.

- Focusing on the actors’ goals helps to
  - capture better what the system actually needs to accomplish
  - evaluate better how much priority should be assigned to a certain requirement

Team Assignment 2

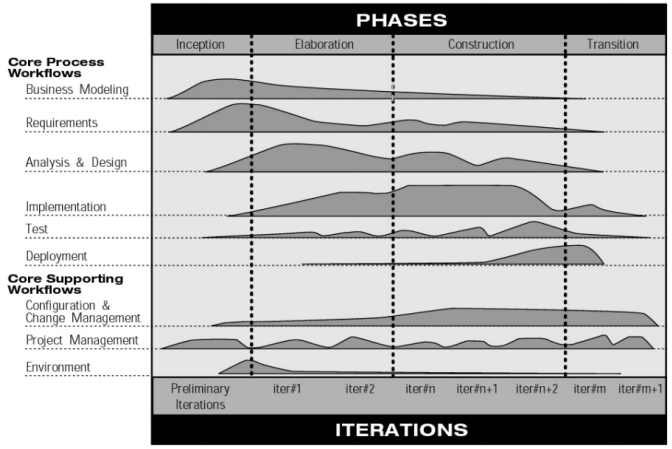
- By **Sun 30 Sep**, submit in Ugla:
  - An initial **Use Case document** (text-only, no UML diagrams)
    - Describe the two most complex requirements mentioned in the Vision & Scope document
      - Use *fully dressed use case* template on next slide
      - Sections 1, 4, 6, 7, 8, 9 and 13 are mandatory for assignment
- On **Thu 4 Oct**, present and **explain** your document to your tutor:
  - How did you come up with your use cases and scenarios?
  - Why are these the most important ones?
  - What further information might have to be considered before implementation?
- **Grading criteria**
  - 2 main use cases in fully-dressed format (50% each)
    - Precise and plausible information in Sect. 1, 4, 6, 7, 13 (10%)
    - Precise formulation of main scenario in Sect. 8 (20%)
    - Precise formulation of alternative scenarios in Sect. 9 (20%)

Recap: Fully Dressed Use Case Format Template

Assignment 2

1. **Use case name**
   - What are these scenarios about? (Start with a verb.)
2. **Scope**
   - What is the system under design?
3. **Level**
   - Is this use case describing a user's goal (top level) or a sub-function of some other use case?
4. **Primary actor**
   - Who calls on the system to deliver its services?
5. **Stakeholders and interests**
   - Who cares about this use case, and what do they want?
6. **Preconditions**
   - What must be true at the beginning of the scenario (and worth telling the reader)?
7. **Success guarantee**
   - What must be true on successful completion (and worth telling the reader)?
8. **Main success scenario**
   - What is the typical, unconditional "happy path" scenario of success?
9. **Extensions / alternate scenarios**
   - What are alternate scenarios of success or failure?
10. **Special requirements**
    - What are related non-functional requirements?
11. **Technology and data variations list**
    - What varying input/output methods and data formats should we be aware of?
12. **Frequency of occurrence**
    - How often does this use case occur? (Influences investigation, timing, priority, testing…)
13. **Miscellaneous / open issues**
    - What open issues are there?

Recap: Unified Process: The Big Picture



1. **Inception**
   - Approximate vision
   - Business case
   - Scope
   - Vague estimates

2. **Elaboration**
   - Refined vision
   - Identification of most requirements and scope
   - More realistic estimates
   - Iterative implementation of core architecture
   - Resolution of high risks

3. **Construction**
   - Iterative implementation of lower-risk and other elements
   - Preparation for deployment

4. **Transition**
   - Beta tests
   - Deployment

- ***See this as a tool box, not a chore!***