DAT231/DIT284 Requirements Engineering

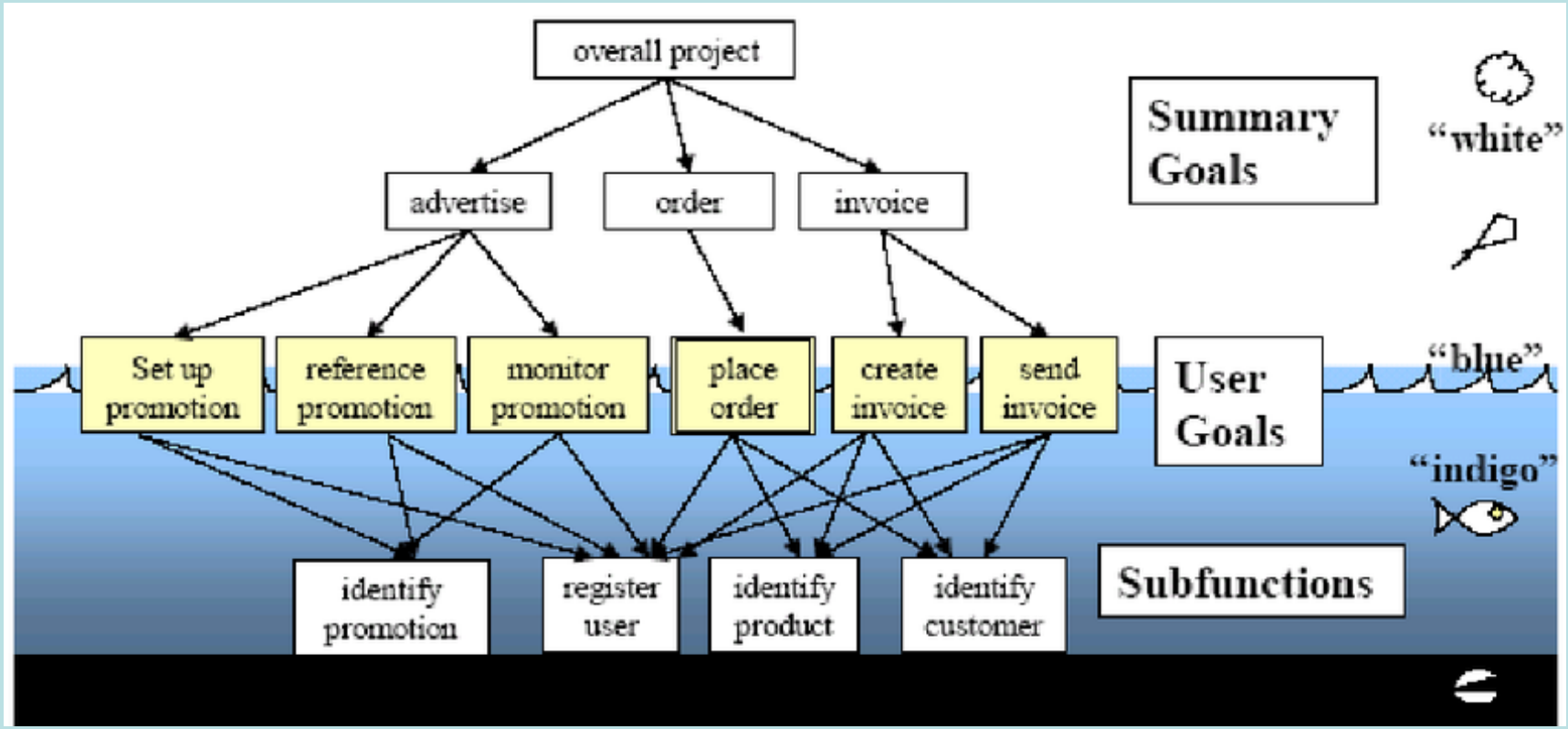
Workshop 4: Requirements Documentation

Why do we have misunderstandings

- *Gulfs of Understanding*
  - Between people that do not know and never meet each other
- Developer and Marketing
  - Marketing must behave like real customer and fight for requirements
  - But: Requirements increase costs
- Developer and User
  - Overcome *Symmetry of Ignorance*
  - Learn to think like the other
  - Requirements are *common ground*
- Customer and internal staff
  - Take customers serious!
  - Without customer no business.

A few slides with general advise
and principles (mind background!)

Abstraction Levels



[Coc2001]

Abstraction Levels

[GW2006]

In agile:

- Themes
- Epics
- User Stories
- Tasks

Simple rules for good requirements

Fundamentals

- Perfectionism is trouble in the beginning
  - –Quick sketch in the beginning: just enough for overview and coordination
  - –Do not forget to improve! Otherwise, requirements will stay in raw state
  - –Important: Specifiy maturity of requirements and keep it up to date!
- How to structure single requirements/features (Facets)

User? Who? Substantive	Bank customer	Foreign student
Can do what? Verb	withdraws	searches
With what? Object	Money	program webpage
How? Adverbial Ext.	at ATM	for admission restrictions

Simple rules for good requirements

Fundamentals

- Simple, active voice sentences
  - "The customer can specify whether she/he prefers small or big bank notes"
- Use simple vocabulary
  - Especially if customers and developers have mixed language background
  - Complicated terms must be defined in the glossary!
- Identify the user of a requirement!
  - Who should be able to do what? Specify role or person and their profile
- Formulate goal oriented: specify the desired result
  - Property desired by identified user
    - "Parking garage customer can see where free parking spaces are from entrence"
- Specify verifiable criteria
  - Testable, objective, quantitative: "as far as possible"

What to avoid

Sounds easy in theory

- Ambiguities
  - "or" is ambiguous: OR or EXOR or even AND?
    - "Designer chooses background color or text color"
  - "he/she/it / the system": Careful with pronouns
- Combined requirements
  - Conjunctions between requirements: OR, AND, WITH, ALSO
    - "…the screen shall go black for a short period and if data might be lost, it will be stored…"
- Exceptions from the rule
  - Specifying some exceptions within a requirement, will often lead to unclear/incomplete cases and combinations
  - Indications: *if, when, with exception of, in the case of, unless, even though, always, …*
  - Handle separately!

What to avoid

Sounds easy in theory

- Verbose requirements
  - Long sentences are boring, hard to read, often not completely thought out
- Drift into design and planning
  - Design and planning ideally based on requirements
  - Details are signs of warning: Component names, DB fields, etc.
- Speculation, Guessing, Over-Generalization
  - No wish lists. No fantasies about future users
  - Avoid possible extensions
  - Avoid wishful thinking: Will only lead to frustration
- Vague formulation
  - User friendly, secure, flexible, customer oriented, modern, …

More on writing style

Based on Chris Rupp / Neurolinguistic Programming

- Goal: Understand the exact meaning of a statement
- Assumption: Transformations of language distort requirement
- Method: Content clarity through writing rules that avoid transformations

Omission	*Generalization*	*Distortion*
Incomplete statements	Universal quantifiers	Nominalization
Degenerated modal verbs	Incomplete conditions	Functional verbs
Implicit assumptions	Substantives without relation	

Examples: Omission

Problematic reqt	Questions to ask	
In the start dialogue, the person number is provided.	Who is responsible?	The user provides their person number in the start dialogue.
Books, which are repeatedly borrowed, are listed.	Who borrows, how, when, where?	
Other customers can easily get information online.	What other customers? What is easy? In comparison to what?	

Examples: Omission

Problematic reqt	Questions to ask	
The system must never forward private annotations	Who is enabling that? How is making sure it does not happen?	Check modal verbs: have to, can, must not
The system uses only annotation data from local sources		Check quantifiers: for all, never, always, …
The system forwards the list of all books.	Really no exception? What if the list is empty?	Check for exceptions
…	Are there any implicit assumptions?	Identify and express them concretely!

Examples: Omission

Problematic reqt	Questions to ask	
Every book can be extended two times.	Really every book?	The loan of every book that is *available* and *not reserved* can be extended two times.
If the user is an adult, (s)he will gain access.	All possible branches covered?	
The customers log in. They notice that many other customers…	Do not use plural unless absolutely necessary.	A customer logs in. (S)he notices that many other customers…

Examples: Omission

Problematic reqt	Questions to ask	
On request, the authentication will be executed.	Are there too many nouns (request, authentication,…)?	The ATM requests authentication. The user provides their PIN.

Fig 9.1 Quality criteria for a specification

Classic: A good requirement spec is:

Correct
Each requirement reflects a need.

Complete
All necessary requirements included.

Unambiguous
All parties agree on meaning.

Consistent
All parts match, e.g. E/R and event list.

Ranked for importance and stability
Priority and expected changes per requirement.

Modifiable
Easy to change, maintaining consistency.

Verifiable
Possible to see whether requirement is met.

Traceable
To goals/purposes, to design/code.

Additional:

Traceable from goals to requirements.
Understandable by customer and developer.

Fig 3.6A Task descriptions

Work area: 1. Reception
Service guests - small and large issues. Normally standing. Frequent interrupts. Often alone, e.g. during night.
Users: Reception experience, IT novice.

R1: The product shall support tasks 1.1 to 1.5

Task: 1.1 Booking
Purpose: Reserve room for a guest.

Task: 1.2 Checkin
Purpose: Give guest a room. Mark it as occupied. Start account.
Trigger/
Precondition: A guest arrives
Frequency: Average 0.5 checkins/room/day
Critical: Group tour with 50 guests.

Sub-tasks:
1. Find room
2. Record guest as checked in
3. Deliver key

Variants:
1a. Guest has booked in advance
1b. No suitable room
2a. Guest recorded at booking
2b. Regular customer

Missing sub-task?

Task: 1.3 Checkout
Purpose: Release room, invoice guest.
. . .

From: Soren Lauesen: Software Requirements

Fig 3.6B Triggers, options, preconditions

Task:	**Look at your new e-mails**
Purpose:	Reply, file, forward, delete, handle later.
Trigger:	A mail arrives.
-	Someone asks you to look.
-	You have been in a meeting and are curious about new mail.
Frequency:	. . .

Task:	**Change booking**
Purpose:	. . .
Precondition:	Guest has booked?
Trigger:	Guest calls

. . .

Sub-tasks:

1. Find booking
2. Modify guest data, e.g. address (optional)
3. Modify room data, e.g. two rooms (optional)
4. Cancel booking (optional)

Makes sense?

Fig 3.8A Tasks & Support

From: Soren Lauesen: Software Requirements.

Task: 1.2 Checkin Purpose: Give guest a room. Mark it . . . Frequency: . . .	
Sub-tasks:	**Example solution:**
1. Find room. **Problem:** Guest wants neighbor rooms; price bargain.	System shows free rooms on floor maps. System shows bargain prices, time and day dependent.
2. Record guest as checked in.	(Standard data entry)
3. Deliver key. **Problem:** Guest forgets to return the key; guest wants two keys.	System prints electronic keys. New key for each customer.
Variants:	
1a. Guest has booked in advance. **Problem:** Guest identification fuzzy.	System uses closest match algorithm.

Past: Problems

Domain level

Future: Computer part

Fig 3.9 Vivid scenario

Scenario: The evening duty

Doug Larsson had studied all afternoon and was a bit exhausted when arriving 6 pm to start his turn in the reception. The first task was to prepare the arrival of the bus of tourists expected 7 pm. He printed out all the checkin sheets and put them on the desk with the appropriate room key on each sheet.

In the middle of that a family arrived asking for rooms. They tried to bargain and Doug always felt uneasy about that. Should he give them a discount? Fortunately Jane came out from the back office and told them with her persuading smile that she could offer 10% discount on the children's room. They accepted, and Doug was left to assign them their rooms. They wanted an adjoining room for the kids, and as usual he couldn't remember which rooms were neighbors.

Around 10 pm, everything was quiet, and he tried to do some of his homework, but immediately became sleepy. Too bad - he wasn't allowed to sleep at work until 1 AM. Fortunately the office computer allowed him to surf the net. That kept him awake and even helped him with some of his homework.

From: Soren Lauesen: Software Requirements

Fig 3.10 Good tasks

Good tasks:
- Closed: goal reached, pleasant feeling
- Session: Small, related tasks in one description
- Don't program

Examples:

1 Manage rooms?
2 Book a guest?
3 Enter guest name?
4 Check in a bus of tourists
5 Stay at the hotel?
6 Change the guest's address etc?
7 Change booking?
8 Cancel entire booking?

Frequent mistake

Got them all?
- All events covered?
- Critical tasks covered?
- At least as good as before?
- CRUD check

How to deal with that?

From: Soren Lauesen: Software Requirements

Fig 3.11 High-level tasks

Task:	**1. A stay at the hotel**
Actor:	The guest
Purpose:	. . .
Sub-tasks:	**Example solution:**
1. Select a hotel. **Problem:** We aren't visible enough.	?
2. Booking. **Problem:** Language and time zones. Guest wants two neighbor rooms	Web-booking. Choose rooms on web at a fee.
3. Check in. **Problem:** Guests want two keys	Electronic keys.
4. Receive service	
5. Check out **Problem:** Long queue in the morning	Use electronic key for self-checkout.
6. Reimburse expenses **Problem:** Private services on the bill	Split into two invoices, e.g. through room TV.

From: Soren Lauesen: Software Requirements

Fig 3.12B Human and/or computer

Human and computer separated

Use case: Check in a booked guest

User action	**System action**
Enter booking number	
	Show guest and booking details
Edit details (optional)	
	Store modifications
Push checkin	
	Allocate free room(s)
	Display room number(s)
Give guest key(s)	

Fig 3.15 Standards as requirements

R1: Data transfer to the account package shall be done through a file with the format described in WonderAccount Interface Guide xx.yy. The account numbers shall be . . .

R2: The user interface shall follow MS Windows Style Guide, xx.yy. The MS Word user interface should be used as a model where appropriate.

R3: Shall run under MS-Windows release xx.yy. Supplier shall port product to new releases within _____ months.

R4: Shall follow good accounting practice. The supplier shall obtain the necessary certification.

R5: The supplier shall update the payroll computations in accordance with new union agreements within one month after release of the agreement.

From: Soren Lauesen: Software Requirements

Fig 3.16 Development process as requirement

R1: System development shall use iterative development based on prototypes as described in App. xx.

Generates new requirements?

R2: Supplier shall deliver additional screens with a complexity like screen S3 at a price of $____ per screen.

R3: All developers shall spend at least two days working with the users on their daily tasks.

R4: A special review shall be conducted at the end of each development activity to verify that all requirements and system goals are duly considered. The customer's representative shall participate in the review.

R5: Customer and supplier shall meet at least two hours bi-weekly to review requests for change and decide what to do, based on cost/benefit estimates of the changes.

From: Soren Lauesen: Software Requirements

General feedback

- Specification template: both good and bad
  - Make room for information that you feel is important!
    - What about: List elicitation activities, validation activities, changelog, additional information,…
  - Delete sections you do not need
- If you have a Figure that is not referenced in the text, please delete it.
  - Also: Have good references between requirements, figures and models
  - If you have a requirement that has a figure (which might be a good idea sometimes), you should still have a concrete requirement.
- Not always clear how the elicitation activities (creativity!) contribute.

Into the Workshop

- I will show examples
- Sit in your project groups and discuss
  - Do you have similar examples?
  - Do you have a solution you are particularly proud of?
- Bring up any questions from your documents!