

2. Kravspecifikation

IEEE-standarden for kravspecifikationer
Typer af tekstkrav
Funktionelle og ikke-funktionelle krav
Typer af grafiske krav
Teknikker til at finde kravene

Efter denne lektion (og projektet) skal du:

- Kunne analysere en given problemstilling og skrive en simpel kravspecifikation for et mindre IT-system
- Vide hvordan en kravspecifikation efter IEEE standard 830-1993 ser ud
- Kunne anvende en række forskellige formater til at formulere krav
- Kunne anvende interview-teknikken til at afdekke krav

Kilder til lektionen

Da Pressman IKKE dækker kravspecifikationer specielt godt har jeg valgt at lade mig inspirere af:

- Søren Lauesen (1999). Software Requirements. Samfundslitteratur. ISBN 87-593-0794-3
- Karl Wiegers (1999). Software Requirements. Microsoft Press. ISBN 0-7356-0631-5
- IEEE Std 830-1993, Recommended Practice for Software Requirements Specifications, December 2, 1993

Bemærk at slides til denne lektion er på engelsk

IEEE standard 830-1993

Recommended Practice for Software Requirements Specifications

- Introduction
- Overall description
- External Interface Requirements
- System Features
- Non-functional Requirements
- Other Requirements
- Appendix A - C

This version of IEEE-830 adapted and extended by Wiegers (1999)

1. Introduction

- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience and Reading Suggestions
- 1.4 Product Scope
- 1.5 References

2. Overall Description

- 2.1 Product Perspective
- 2.2 Product Functions
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 Assumptions and dependencies

3. External Interface Requirements

- 3.1 User interfaces
- 3.2 Hardware interfaces
- 3.3 Software interfaces
- 3.4 Communications interfaces

4. System Features

- 4.x System Feature X
 - 4.x.1 Description and Priority
 - 4.x.2 Stimulus / Response Sequences
 - 4.x.3 Functional Requirements

5. Non-functional Requirements

- 5.1 Performance requirements
- 5.2 Safety requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes
- 5.5 Business Rules
- 5.6 User Documentation

Alternative: ISO 9126
5.1 Efficiency Requirements
5.2 Usability Requirements
5.3 Maintainability Requirements
5.4 Functionality Requirements
5.5 Reliability Requirements
5.6 Portability Requirements

Why use an SRS ?

- For a small project, the IEEE 830 is an excellent checklist for ensuring that all project participants know the functional requirements, the stored data model and the constraints on the system.
- Often, the objective of the SRS can be satisfied with a few bullet lists and a couple of diagrams
- AVOID creating making the creation of an SRS a time-consuming, paper-wasting exercise - One SRS for a defense system (approximately 100 staff-years in effort) is said to have been shipped in a panel van!
- The usefulness of the SRS is not in the volume of paper produced but rather in how well it communicates what the system should be to all project participants.

Requirements can be documented 3 ways:

- Textual styles that use well-structured and carefully written natural language
- Graphical styles that illustrate transformational processes, system states and changes between them, data relationships, logic flows, or object classes and their relationships
- Formal specifications that define requirements by using mathematically precise formal logic languages – *Formal Spec. is not covered here*

Different textual styles for Behavioral/Functional Requirements

- Feature style
- Workflow description style
- Design style
- Event or Function list style
- Standards style
- Development process style

Feature style

Examples:

- The hotel reservation system shall accept requests for bookings and within 10 seconds produce a list of available rooms in the requested period
- The hotel reservation system shall be able to record that a room is occupied for maintenance in a specified period
- The hotel reservation system shall accept bookings from guests of a certain room kind. The actual room allocation is not done until the guest check-in

Workflow description style

Example - The hotel reservation system shall work as follows:

- Find a free room
 - receive request for room type and period
 - List rooms of requested kind
 - Allow user to select room for requested period
- Check-in
 - Check that guest is booked and room is free
 - If guest_record exists then find it, else record information for new guest_record
 - Set room_status to occupied for given period

Design style

Examples:

- The hotel reservation system shall use the screen layouts found in appendix XYZ
- The menu points and buttons shall work according to the description in appendix XYZ
- The hotel reservation system shall be able to show the total number of rooms occupied and booked as a bar graph with one bar per day.

Event or Function list style

Example:

The hotel reservation system shall support the following business events:

- A guest books a room
- A guest with a booking checks in
- A guest without booking checks in
- A guest checks out
- A checked-in guest wants another room

Standards style

Examples:

- The hotel reservation system user interface shall follow Microsoft Windows Style Guide version x.y.z
- The Microsoft user interface shall be used as a model where appropriate
- The hotel reservation system shall run under Microsoft Windows version XX, release YY

Development process style

Examples:

- The hotel reservation system shall use prototyping in the development. At least 3 prototypes shall be created and tested during development
- The hotel reservation system shall be developed using the Rational Unified Process development model

For Non-functional Req. the ISO 9126 Quality Model can be used

Efficiency

Time behaviour, resource utilisation

Usability

Understandability, learnability, operability

Maintainability

Analysability, changeability, stability, testability

Functionality

Accuracy, Suitability, interoperability, compliance, security

Reliability

Maturity, fault tolerance, recoverability

Portability

Adaptability, installability, conformance, replaceability

Efficiency style

Examples:

- The hotel reservation system shall be able to process 100 payment transactions per second in peak load
- For simple editing functions, response time shall be less than 1 second
- Scrolling one page up or down in a 200 page document shall that at most 1 second. Jumping to any of the 200 pages shall take at most 5 seconds.

What is Usability?

- **Ease of learning:** How fast can a user who has never seen the user interface before learn it sufficiently well to accomplish basic tasks?
- **Efficiency of use:** Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?
- **Memorability:** If a user has used the system at some earlier date, can he or she remember enough to use it more effectively next time (or does the user have to start over again learning everything every time)?
- **Error frequency and severity:** How often do users make errors while using the system, how serious are these errors (burning down a cement plant is worse than getting the wrong player's score on a golf site), and how easy is it to recover from a user error?
- **Subjective satisfaction:** How much does the user *like* using the system?

Usability style

Examples:

- Novice users of the hotel reservation system shall be able to perform task X and Y in 15 minutes
- Experienced users of the hotel reservation system shall be able to perform task X, Y and Z in 5 minutes
- 80% of the users shall find the system easy to learn and efficient in daily use

Maintainability style

Examples:

- Supplier's hotline for the hotel reservation system shall analyze and classify 95% of defect reports within 2 working hours
- Installation of new and updated versions of the hotel reservation system shall leave all database content unchanged
- Every program module must be assessed for maintainability according to procedure XYZ. At least 75% must obtain "highly maintainable" and none must obtain "poor"

Functionality style

Examples:

- The hotel reservation system shall be able to export room reservation data as an Excel spreadsheet (interoperability)
- To obtain unauthorized access to the system it should take more than one hour for an experienced hacker using a modem or Internet connection to interface with the hotel reservation system (security)

Reliability style

Examples:

- After a power-out failure only one transaction (the current) must be lost.
- When starting after failure the system should notify the user of the last transaction correctly recorded.

Portability style

Examples:

- The hotel reservation system shall be designed with a persistent architecture that is the same for any available database interface. The purpose is to make the application independent from the underlying database system, as well as make it feasible to port the application program between platforms and from one database system to another

Using Partitions to achieve a persistent architecture

Presentation tier

Dept ID	Department Name	Department Manager
100	R&D	David Scott
200	Finance	Michael Hartley
300	Marketing	May Ann Dab
400	Shipping	Scott Stevens
500	Support	John Malin

Business logic tier



Data access tier



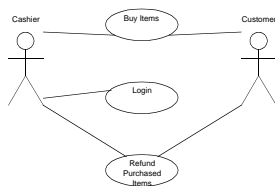
One can also use graphical styles to write Requirements – The details will be explained in later lessons

- Use Cases
- Conceptual Model
- Data Model
- Context Diagram
- Dataflow Diagrams
- State Transition Diagrams

Use Case style

Example:

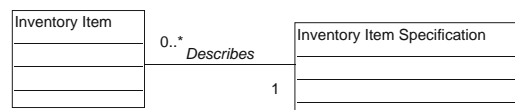
The product shall be able to handle the following Use Cases:



Conceptual Model style

Example:

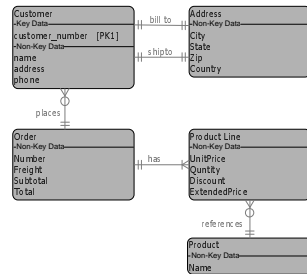
The product shall be able to handle the following concepts:



Data Model style

Example:

The product shall be able to store the following data:



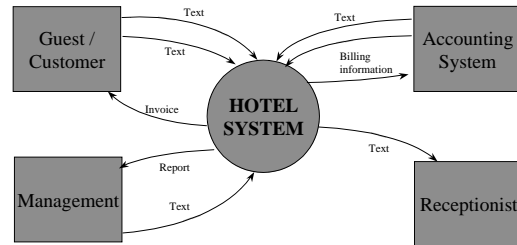
© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 31

Context Diagram style

Example: The product shall have the following interfaces:



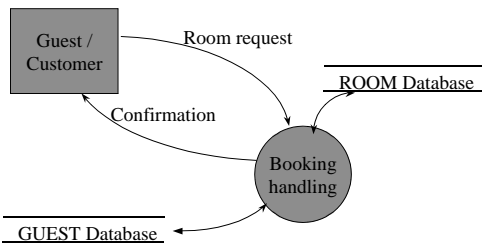
© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 32

Dataflow Diagram style

Example: The product shall support these activities:



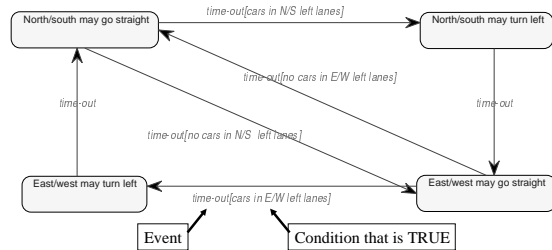
© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 33

State Transition Diagram style

Example: The Traffic Light shall change as shown in the following State Transition Diagram:



© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 34

Exercise:

Look at an existing Requirements Specification (see course web site) and try to identify examples of:

1. Behavioral requirements – different styles
2. Non-Behavioral requirements – different styles
3. Things to help the reader
4. Examples of solutions



© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 35

Requirements Elicitation Problems

- System requirements change
- Articulation of requirements is difficult
 - Functions and processes are not easily described
 - End users have difficulty expressing what they do or what they want
- User motivation is difficult
 - May dislike idea of having to use a new system
 - Rewards are not seen until the system is implemented

Scharer, Datamation, 1981.

© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 36

The Many Faces of Requirements

- Many stakeholders
 - Users, administrators, owners, system builders ...
- Many concurrent activities
 - Elicitation, modeling, specification, documentation ...
- Many uses of requirements
 - Communication, contracts, development ...



© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 37

Us Vs. Them

Scharer, Datamation, 1981.



- | | |
|---|--|
| <ul style="list-style-type: none"> • How we see users <ul style="list-style-type: none"> – Don't really know what they want – Can't articulate what they want – Have too many needs which are politically motivated – Can't prioritize needs – Refuse responsibility for the system – Are unwilling to compromise | <ul style="list-style-type: none"> • How users see us <ul style="list-style-type: none"> – Don't understand the business – Handle company politics awkwardly – Tell them how to do their job – Say no all the time – Are always over budget – Are always late – Are unable to respond quickly to changing needs |
|---|--|

© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 38

Users' Defense Strategies

- The kitchen sink
 - Throw in everything as a requirement
- Smoking
 - Ask more to use later for bargaining
- The same thing
 - Laziness or lack of knowledge



Scharer, Datamation, 1981.

© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 39

Techniques: Interviewing

- Structured vs. Unstructured
- Selection of interviewees
- Arranging an interview
- Conducting an interview
 - Stage setting
 - Establishing rapport
 - Directing the interview
 - Listening
 - Biases and non-verbal cues



© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 40

Unstructured interviewing

- A meeting between two people without a fixed agenda or pre-planned questions
- Example 1: "Could you tell me about a day at work, starting with the beginning?" (Grand Tour)
- Example 2: "Could you tell me how you do this / use this tool?" (Specific Tour)
- Example 3: "Could you show me how you do this?" (Physical Tour)

© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 41

Structured interviewing

- A meeting between two people with a fixed agenda and 100% pre-planned questions
- Questionnaire can be used

Question no. 10
In what type of projects do you use Use Cases?

- Organisational change project
- Systems development project
- General problem solving
- Process evaluation
- Education and training
- Other (please specify)

© Jan Pries-Heje

2. Kravspecifikation

Slide no.: 42

Semi-structured interviewing

- A meeting between two people with a checklist of issues and possible questions

Product Issues

What primary products & services do you produce at 'internet speed'? Do they differ from other products in terms of complexity, scalability, connectivity or interfaces, or other characteristics? If so, how?
Do you build individual products or product lines or families?

Interview Planning Steps

- Read background material
- Establish interview objectives
- Decide who to interview
- Prepare the interviewee--memo/letter
- Prepare for the interview
- Decide on question types and structure

Steps for Conducting the Interview

- Make an appointment
- Prepare! Know the interviewee
- Be on time
- Have a planned beginning
- Have a planned middle
- Have a planned closing
- Follow up

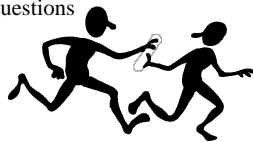
Have a Planned Beginning

- Introduce yourself and your role on the project
- Use open-ended general questions to begin the discussion
- Be interested in all responses. Pay attention!



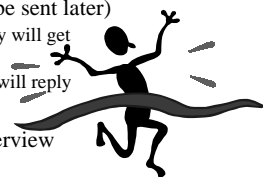
Have a Planned Middle

- Combine open and closed-ended questions
- Follow up with probing questions
 - Active listening
- Provide feedback
 - “Let me tell you what I think I heard...”
 - Paraphrasing
- Limit note taking to avoid distraction



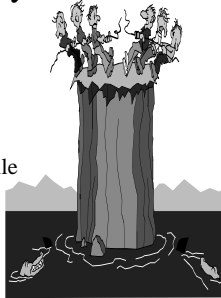
Have a Planned Closing

- Summarize what you heard
- Ask for corrections if necessary
- Request feedback on your notes & interview summaries (to be sent later)
 - Provide date by which they will get information for review
 - Ask for date when he/she will reply
- If follow-up is needed, set date & time for re-interview



Interviewing Summary

- Objectives
- Interviewee assessment
 - Top, middle, user
 - Friendly, indifferent, hostile
- Anticipated obstacles
 - Expected outcomes



Preparation for Project:
Prepare a semi-structured interview guide to be used for an In Class Interview

