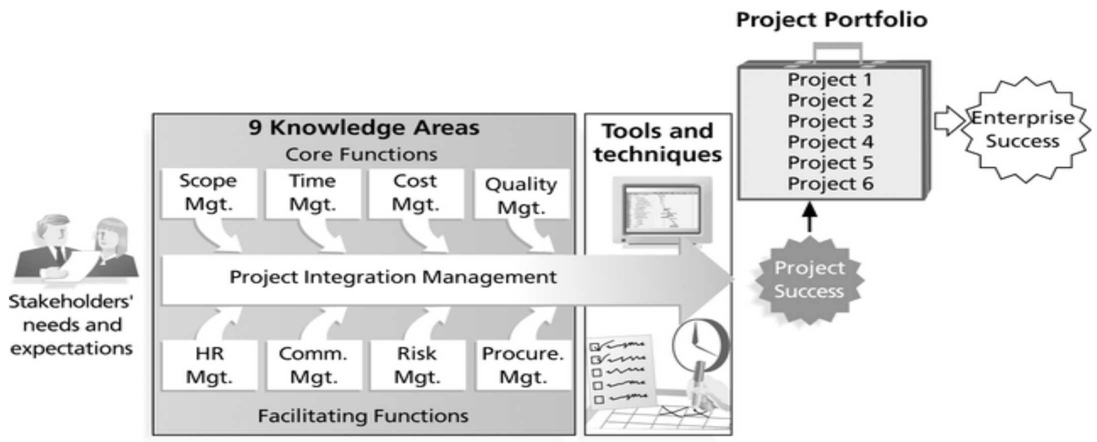
Information Engineering (and Testing)

LN01: Introduction



Marian Gheorghe

Aims of the lecture

- Overview of the module
- Software Engineering principles
- Software Engineering concepts
- Notations, models and diagrams
- Requirements expressed as story cards

Information Engineering (IE)

- **IE –** an approach for designing and developing **Information Systems (ISs)**
- **IS – any system collecting, storing, handling and processing information**
- **IS** is also a generic name for any software system where data is a central element of it

- In this module we will take a more general view on IE: information description and handling by considering various ways of representing information, including concepts, abstract and specific notations and representations

Topics Taught

1. Software Engineering concepts: waterfall model; agile approaches; analysis and design; processes and models
2. Objects (data & functions); UML notation
3. Testing; unit testing
4. Formal notation and OCL (part of UML)
5. Project management
6. Modelling – FSM and EFSM, syntactic diagrams, Petri nets
7. More advanced testing – system testing, model based testing, mutation testing

Assessment

- Coursework covering mainly topics 1-3, released in week 5 and with deadline week 8
    - Will refer to object oriented concepts – notations, implementation, unit testing

- Exam – testing the understanding of the concepts taught and problem solving abilities. Mostly topics 4-6, with some concepts from the previous topics

- Each assessment is worth 50%
- Each of the two parts, reflected in the two assessments, will be revised before each assessment

Comments on the Topics Taught

- It might appear that (some/most of) the topics are already taught somewhere else – EnterprisePro or ISAD (Year 3 students)
- Concepts might be known – requirements, design, some of the UML diagrams etc… but they will be used in different forms and sometimes with specific/different meaning
- Don’t assume that the concepts taught in these modules are enough to understand their use in the context of IE
- The vast majority of the concepts and problems are new and they require attention and good understanding

Enterprise Pro & ISAD

- Both refer to similar concepts: Software Engineering, Analysis & Design, Testing

- ISAD is more about various SE notations, such as UML
- Enterprise Pro is about the pragmatics of SE projects, real clients, group activities

- IE will investigate
    - UML notation from a different perspective: its correct usage and semantics, connections with OCL and testing
    - Different new computational models relevant in expressing IE systems
    - Various testing approaches
    - Project management methods

Software Engineering – Introduction

What is Software Engineering ?

- About handling complex (large) software projects

- Involve whole development cycle: identifying the system features (functions, data, interfaces etc); designing and implementing these features; testing; deployment; maintenance

- Implementation of a process that guarantees good results
- Break down a system into simpler components (parts) and provide solutions
- Project management: identifying what/when to do; organising projects and stages

Small scale coding vs SE

Small scale coding requires:

- personal skills (programming language concepts)
- experience (programming, using a compiler)
- flexible (suitable) materials (simple environments)

SE requires:

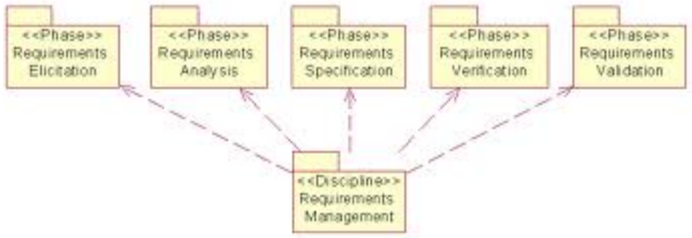
- certain qualification (SE or similar)
- plan, project, tools
- controlled materials (advanced SE environments)

Software Engineering – Key topics & concepts

Key topics

Requirements (capturing and specifying functions, data, interactions of a software system):

- types of requirements
- requirements capture methods
- models (behaviour, data)
- specifications and notations (diagrammatic, formal)

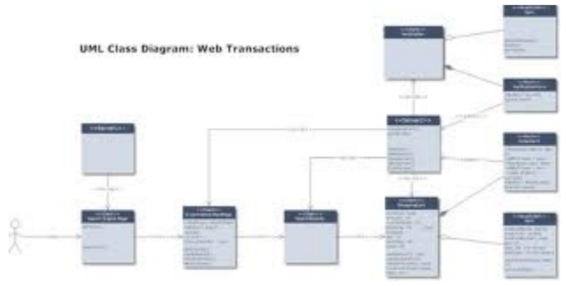


<<Phase>> Requirements Elicitation
<<Phase>> Requirements Analysis
<<Phase>> Requirements Specification
<<Phase>> Requirements Verification
<<Phase>> Requirements Validation
<<Discipline>> Requirements Management

Design (abstract specification of the software system structure and behaviour):

- system architecture
- detailed design (object oriented design)
- data design
- user interface

Logical vs physical design.



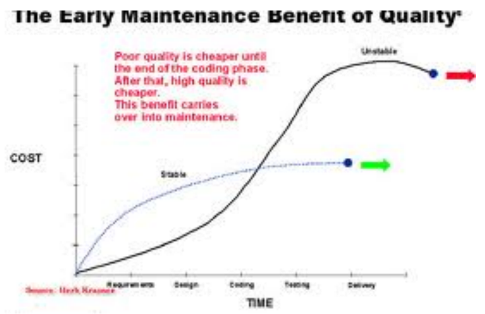
Key topics (cont.)

Verification and validation
(checking the software system works properly)

- testing
- automated static analysis, verification
- code inspection

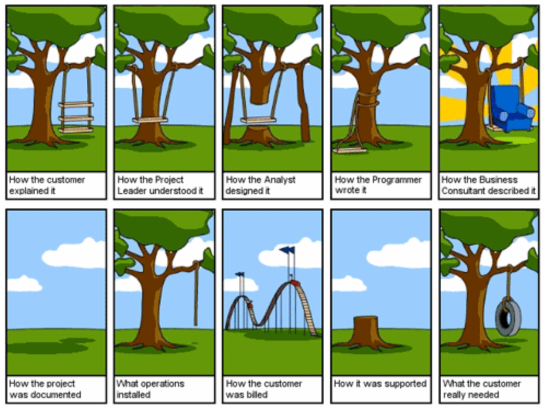
Evolution and maintenance
(changes to software during and after software project life cycle)

- re-engineering
- software and/or data migration
- legacy software



Management
(project, people, software cost, product quality)

Software Engineering Processes (II)


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Processes (II)

- Processes refer to how the product is developed, i.e.
  - activities carried out during the project development
  - methods used to carry out these activities
  - how the quality of the resultant system is assured while these activities are being carried out
- Processes described by sequences of activities
- Fundamental activities:
  - requirements specification (1)
  - design and implementation (2)
  - validation, verification and installation (3)
  - evolution and maintenance (4)

Methods and Models

Methods – hierarchical view

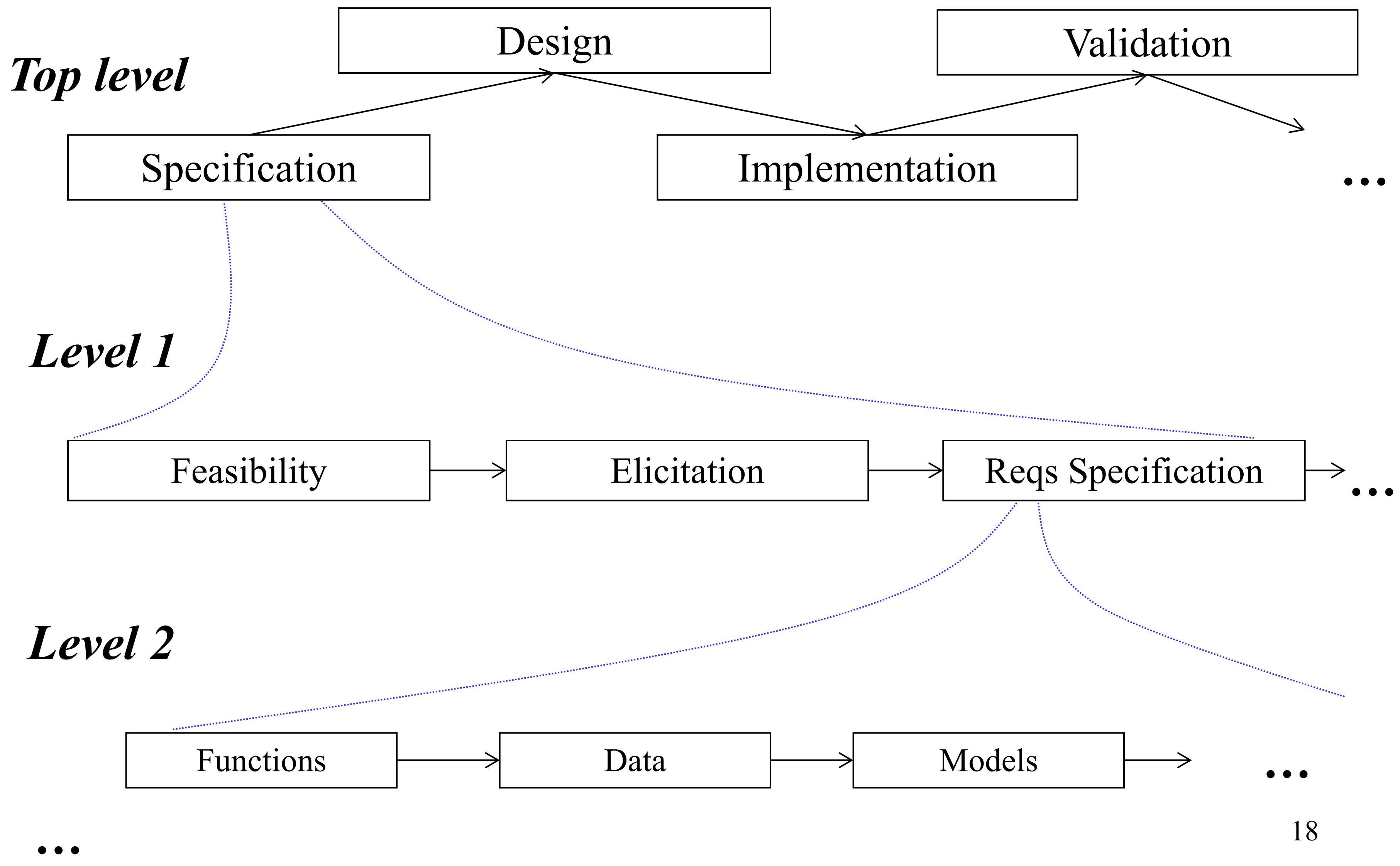
- Processes consist of *activities* (like key fundamental ones and those associated with or derived from) & *methods* to carry them out

- Methods – provide a hierarchical description of the associated activities

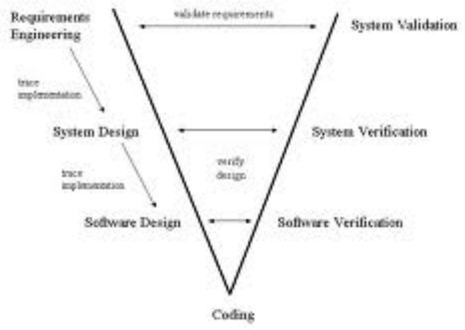
  - Top level: fundamental activities
  - Other levels: activities associated with those from the above levels

& a way to correlate these activities – within various levels and between them

Hierarchies of activities



Software Process Models (fundamental and derived activities)



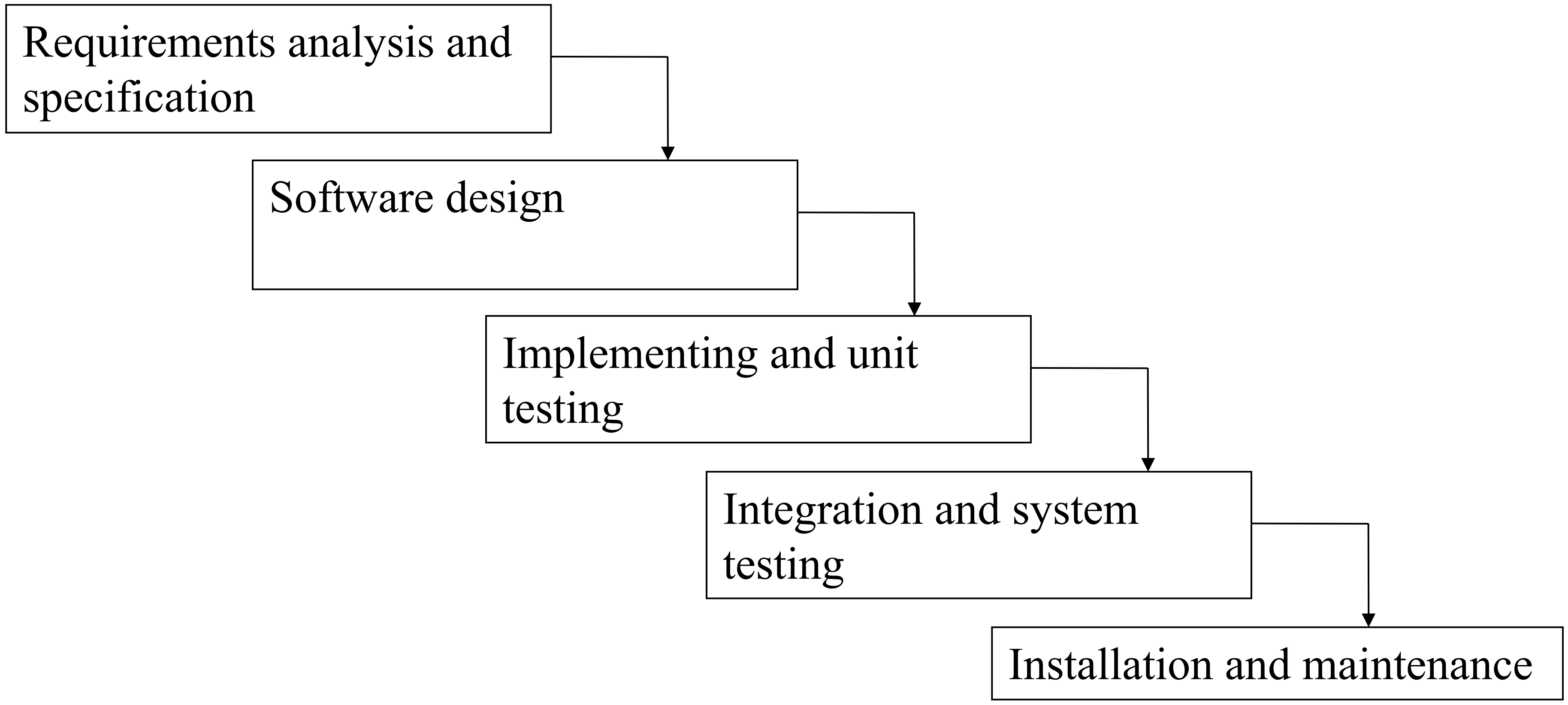
SE process models

Types of process models

- Waterfall
- Evolutionary
- Iterative; Spiral etc
- Agile; XP

Waterfall model

It maps linearly fundamental activities


Requirements analysis and specification
Software design
Implementing and unit testing
Integration and system testing
Installation and maintenance

Evolutionary development

Initial implementation refined into final system

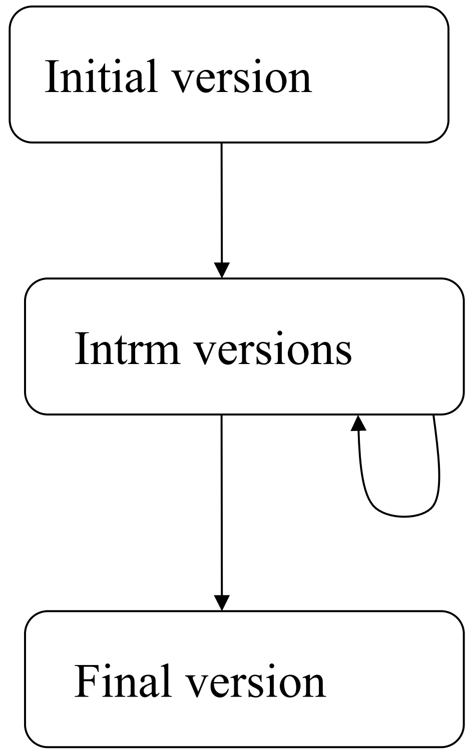
- Exploratory development: the system evolves through stages into a final software product
- Throw-away prototyping

Advantages

- Flexible and suitable for small/medium systems
- Help in early stages or when little information is known about a system

Limitations

- Poorly structured systems
- Special tools/techniques required



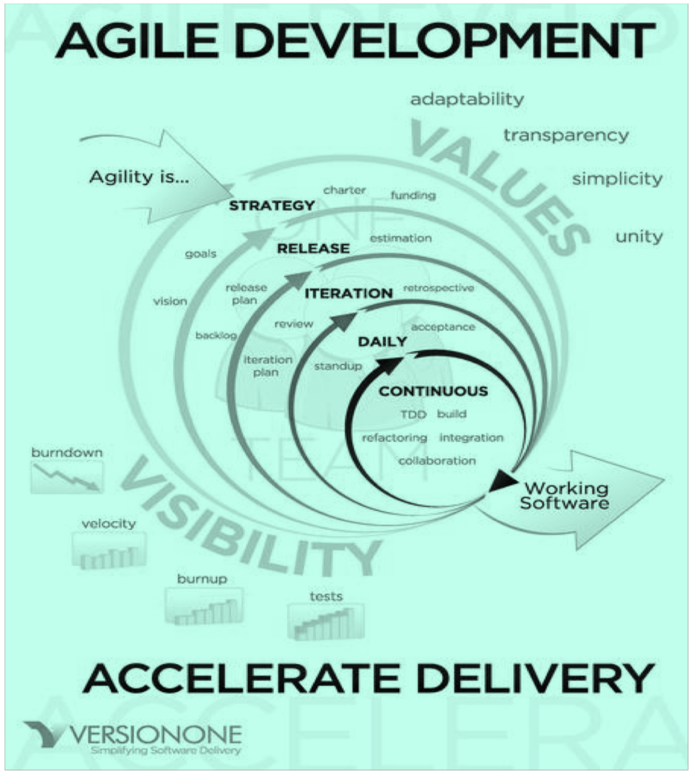
Iterative development (1)

Complex software systems require hybrid approaches (combination of more elaborated stages with partial solutions) involving iterations over certain stages

- Outline requirements
- Assign requirements to increments
- Define system architecture
- Develop, validate & integrate increments (iteratively)
- Validate (partial) system (iterate)
- Obtain final system

Agile methods

- Agile (lightweight) methods appeared as a reaction to classical, heavyweight methods



The agile development method has 4 key principles

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

Agile manifesto: http://agilemanifesto.org/

SE Notations and Diagrams

- Documents and models might use notations and diagrams or a combination of both
- Some notations are standard, others use ad-hoc concepts
- Standard notations and diagrams – UML, Entity-relationships diagrams,
- Non-standard notations – mind maps, story cards (although adopted by agile methodology business)
- Formal notations: EBNF, state machines, Z, sets and functions etc

Requirements (Agile method - story cards)

Project: Library
Story: Login/Logout
Date: 26/10/2009
Requirements number: 0
Task description: (1) The login allows to enter the system to be used by the system administrator; the system administrator is verified through a password; (2) the logout option stops the system

Story cards

- Story cards are utilised by XP agile methodology
- A story card – a description of certain task(s)/function(s) with their connections in a rather structured manner
- Test set aimed to thoroughly check the validity of the implementation
- Requirements are captured as a set of story cards
- Agile approaches based on iterations
- Each iteration is a subset of story cards implemented in one step

Login/Logout

Story card

Project: Library
Story: Login/Logout **Date**: 23/01/2019
Requirements number: 0
Task description: (1) The login allows the system to enter the system; the system administrator enters ID, password; (2) the logout option stops the system;

Initiating event/input: System start up
SC context: IDs, passwords
Observable result/output: ‘success’ or ‘error message’
Tests: 1. insert system Administrator option 2. Insert right ID and password 3. Insert wrong ID (password) 4. Insert wrong option
Associated stories: -

Requirements

Remember from slide 12:

Requirements - capturing and specifying
functions, data, interactions of a software system (interface)

Requirements specification includes the system functions (behaviour)

Functions might handle data

Library system case study – functions and their description (story cards)

Scenario

A cut-down version of a basic library. It is used through the library administrator and assistants only (the administrator has password access).

- Creating catalogue entries for books when they are bought; multiple copies of one book are handled; details may be amended later
- Maintaining records for borrowers: setting them up initially; amending details later; removing them eventually.
- Handling loans: checking them out; checking them back in
- Checking catalogue entries and loan details.

- **Possible extensions**: Check on overdue books and manage fines; notify borrowers of overdue books; record fines due; record fines paid; allow for books to be reserved.

Library System – Story Cards

- A naïve approach based on functions derived directly from the scenario
- A more realistic view includes: data, interactions (discussed later on in a different context) and constraints imposed (non-functional requirements)
- Restrictions regarding the general functionality of the system
    - Run on a certain platform
    - Have a specific interface
    - Allow a certain number of users
    - Optimality criteria...

Library System: Story Cards

List of stories:

- Login/Logout (requirements number 0)
- Register book & copies (1)
- Register borrower (2)
- Edit book & copies (1)
- Edit borrower (2)
- Loan: check out (3)
- Loan: return (3&6) – can be divided into two stories
- Check catalogue (4)
- Reserve (5)

List of users (actors):

Library administrator – interact with the first 5 stories

Library assistants – interact with the rest

LS – Non-functional specification

- Functions above can have certain restrictions or constraints

- **Examples**
  - Only staff members and students are registered
  - Each staff/student member can borrow/reserve up to a certain limit and for a given period
  - Loans – normal, short term
  - Fines might be different
  - Might differentiate between books and journals

Register book & copies

Story card

Project: Library
Story: Register book & copies **Date**: 23/01/2019
Requirements number: 1
Task description: A new book and its copies are registered with the system; unique `isbn` for this book and unique `index` associated with each copy

Initiating event/input: Request to register a book & its copies
SC context: book is not in the system
Observable result/output: success or error message
Tests: 1. A new book, unique `isbn`, `indexes` 2. An existing book 3. A new book, copy with existing `index` 4. Wrong component data (inappropriate `email`, etc)
Associated stories: - Login/Logout

Register borrower

Story card

Project: Library
Story: Register borrower **Date**: 23/01/2019
Requirements number: 2
Task description: A new borrower is registered with the system; unique `uCard`

Initiating event/input: Request to register a borrower
SC context: borrower is not in the system
Observable result/output: success or error message
Tests: 1. A new borrower, unique `uCard` 2. An existing borrower (i.e. existing `uCard`) 3. Wrong component data (inappropriate `email`, etc)
Associated stories: Login/Logout

Edit book & copies

Story card

Project: Library
Story: Edit book & copies **Date**: 23/01/2019
Requirements number: 1
Task description: (1) Add new copies, with distinct `indexes`, of a given book and update `noOfCopies` or (2) remove some copies and update `noOfCopies` or (3) update/remove a book

Initiating event/input: Request to edit a book & its copies
SC context: book in the system & copies are not/are in system
Observable result/output: success or error message
Tests: 1. Existing book, new copies with unique `indexes` 2. Existing book, new copies with existing `indexes` 3. A new book, copy with existing `index` 4. Test `noOfCopies` is right 5. Test book being removed 6. Remove copies/books on loan or reserved
Associated stories: Login/Logout, Register book & copies

Edit borrower

Story card

Project: Library
Story: Edit borrower **Date**: 23/01/2019
Requirements number: 2
Task description: (1) Change component data of a borrower (`name, email` etc) or (2) remove a borrower

Initiating event/input: Request to edit a borrower
SC context: borrower in the system
Observable result/output: success or error message
Tests: 1. Existing borrower (its `uCard` in the system) 2. A new borrower (`uCard` not in the system) 3. Test borrower being removed 4. Remove borrower having books borrowed or reserved
Associated stories: Login/Logout, Register borrower

Loan: check out

Story card

Project: Library

Story: Loan: check out

Date: 23/01/2019

Requirements number: 3

Task description: Available copies or copies of reserved books are allocated to a borrower for a given period

Initiating event/input: Request to borrow books

SC context: borrower and copies involved are in the system

Observable result/output: success or error message

Tests: 1. Existing borrower, available copies; existing borrower, reserved books
2. New borrower with existing and new book
3. Existing borrower, unavailable copy

Associated stories: Login/Logout, Register borrower, Register book &copies, Edit borrower, Edit book & copies, Loan: return, Reserve

Loan: return

Story card

Project: Library
Story: Loan: return **Date**: 23/01/2019
Requirements number: 3; 6
Task description: Borrower returns copies if late return then fines involved

Initiating event/input: Request to return copies
SC context: borrower and copies involved are in the system
Observable result/output: success or error message
Tests: 1. Existing borrower, loaned copies 2. Existing borrower, arbitrary copies 3. New borrower with whatever copies 4. Existing borrower, loaned copies, late return
Associated stories: Login/Logout, Register borrower, Register book &copies, Edit borrower, Edit book & copies, Loan: check out, Reserve

Check catalogue

Story card

Project: Library
Story: Check catalogue **Date**: 23/01/2019
Requirements number: 4
Task description: A `title` or `author` is checked and copies, if exist, are provided with their `status` (`on loan, reserved, in library`)

Initiating event/input: Request provide details about book
SC context: -
Observable result/output: List of copies or error message
Tests: 1. `author` of an existing book 2. `title` of an existing book 3. `author` not in the system 4. `title` not in the system
Associated stories: Login/Logout, Register book &copies, Edit book & copies, Loans, Reserve

Reserve

Story card

Project: Library
Story: Reserve **Date**: 23/01/2019
Requirements number: 5
Task description: A copy of a certain book is reserved by a borrower for a certain date

Initiating event/input: Request to reserve a book
SC context: borrower and books involved are in the system
Observable result/output: success or error message
Tests: 1. Existing borrower, existing book(s) 2. Existing borrower, new book 3. New borrower, existing book 4. New borrower, book not in the system 5. Existing borrower, book in the system, wrong date
Associated stories: Login/Logout, Register borrower, Register book &copies, Edit borrower, Edit book & copies, Loans

Login/Logout

Story card

Project: Library
Story: Login/Logout **Date**: 23/01/2019
Requirements number: 0
Task description: (1) The login allows to enter the system to be used by the system administrator; the system administrator is verified through ID, password; (2) the logout option stops the system

Initiating event/input: System start up
SC context:
Observable result/output: success or error message
Tests: 1. Type in an expected option (1,2,3) 2. Insert system administrator option and right ID, password 3. Insert system administrator option and wrong password or ID 4. Type in wrong option
Associated stories: -

Conclusions

- Some Software Engineering generic concepts and methods introduced and discussed
- Information Engineering is part of SE, but dealing with systems handling data

Reading List

Ian Sommerville, Software engineering, Pearson, 2016

Shari Lawrence Pfleeger, Joanne M Atlee, Software Engineering – Theory and Applications, Pearson, 2009

Martin Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, The Addison Wesley, 2004

Mike Holcombe, Running an agile software development project, Wiley 2008.

Any textbook on object oriented analysis and design, published after 2000

Homework for Tutorial

- Two requirements documents are provided with functions, data and interfaces
- Assess the accuracy of the functions described
- Are there ambiguities or vague statements
- You can work in small groups, 2-3 students each