Requirements Engineering

- Goal
  - Understand the importance to identify requirements
  - Learn that requirements are documented as result of the process named requirement engineering
- Contents
  - Software lifecycle
  - Definitions & roles
  - Requirement life cycle
  - Requirement properties
  - Best practices
  - Overview diagrams & charts for documentation

Product Lifecycle

- Strategy & Conception
- Development
- Market-Entrance
- Evolution / Maintenance
- End of Life (no further support)

Definitions

- Requirement
  - Def: a value to be offered by the system
  - Analytical Specification
    - Functional Req. describe the functions “What”
    - Non-functional Requ. Deals with how the system operates (answer-time, load-capacity, up-time) instead of how it behaves
  - Technical Specification
    - Technical Req. describe the technical implementation “How”
- Requirements Engineering
  - Def: covers activities involved in discovering, documenting and maintaining a set of requirements
  - Very intense & complete in V-Modell / Wasserfall-Modell
  - Approximatively for small set of functionality in agile Development

Roles dealing with requirements

- Product Manager
- Project Manager
- Customer
- Tester
- End-User
- Developer
- UX-Designer / Designer
- Requirements-Engineer / Analyst

Requirement life cycle

1. Identify requirements coming from
   - Market research
   - Customer Interviews
   - Retailer, Partner, Sales, Marketing
   - Customer-Study
2. document requirements as analytical specification “WHAT” (Lastenheft)
   - Documentation as high level user stories
   - Iterative collection using prototyping, simulation, concepts & testing
   - Modeling and analysis using images & charts

Requirement life cycle

3. Requirements evaluation
    - Evaluate / sort requirement wrt to “Does it add any business value”
    - Compare with standards
    - Resolve contradictory requirements
4. Requirement Analysis as technical specification document “HOW” (Pflichtenheft)
    - Dependencies and technical details
    - Specify system boundaries
5. Requirements are implemented in software
6. software is tested against documented requirements

Attributes of requirements

- Complete
- Correct (to the actual knowledge)
- Consistent
- Testable
- Understandable
- Necessary
- Non-ambiguous
- Realizable

Requirement Pattern

- Number, Titel, Status, Priority, Stakeholder
- Description, Limitations
- References, Influences, Dependencies
- Cost / effort
- Acceptance test
- Comments from team members

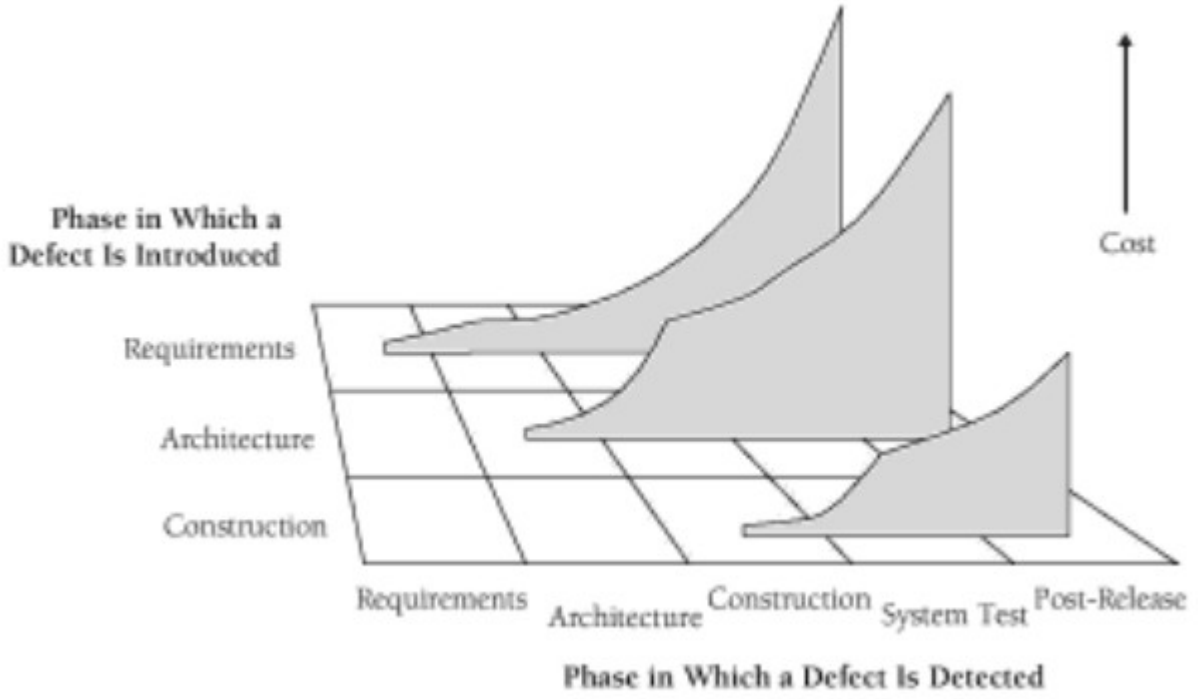
Aufnahmedaten	
Name (ID)	LOI Erstellung (#12)
Datum	2007-12-18
Personen	Grünwald, Herzog
User Story	
Akteure	Projektpartner, Projektleiter, ITM Studiengangsleitung
Vorbedingungen	Alle Projektpartner bekannt, LOI gewünscht
Nachbedingungen	LOI bei Studiengang ITM archiviert
Priorität	Soll
Beschreibung	Der Letter of Interest bildet die langfristige Willensbekundung zwischen den Projektpartnern für eine Zusammenarbeit. Der LOI wird zwar unterschreiben und gibt die Richtung (z.B. Projekt im Bereich Prozessmanagement oder Usability) vor, liefert aber keine rechtliche Bindung und ist daher nicht zwingend für den Projektauftrag nötig. Jedoch kann man versuchen aus dem LOI Projekte abzuleiten. Wird ein LOI gewünscht wird er aus einer Vorlage des Studienganges oder Projektpartners durch den Projektleiter erstellt, von der Studiengangsleitung in Kooperation mit den Projektpartner korrigiert, fertiggestellt und unterschrieben. Danach wird das Dokument durch den Projektleiter im Studiengang archiviert.

http://www.hausarbeiten.de/faecher/vorschau/135430.html

Critics from the view of Agile Dev

- An idea that has never been implemented before is hard to describe in its completeness
- A 100+ pages requirements document cannot be consumed easily and is hard to understand in its whole
- Many ideas are not realizable in its specified form, only prototypes can proof that

Cost of inadequate requirements



True in both system:

- upfront-design (100% requirements are described before testing)
- Agile methods (requirements are discovered iteratively, 5% per iteration)

From book: Steve McConnell, Code Complete: A Practical Handbook of Software Construction, 2004.

Best practices

- Glass' law: Requirement deficiencies are the prime source of project failures.
- Boehm's first law: Errors are most frequent during the requirements and design activities and are the more expensive the later they are removed.
- Boehm's second law: Prototyping (significantly) reduces requirement and design errors, especially for user interfaces.

Glass' law: Glass, R.L.: Software Runaways. Lessons Learned from Massive Software Project Failures. Upper Saddle River, NJ: Prentice Hall 1998.

Boehm's first law: Boehm, B.W., McClean, R.K., Urfrig, D.B.: Some Experience with Automated Aids to the Design of Large-Scale Reliable Software. IEEE Trans on Software Engineering 1, 1 (1975), 125–133.

Boehm's second law: Boehm, B.W., Gray, T.E., Seewaldt, T.: Prototyping Versus Specifying: A Multiproject Experiment. IEEE Trans on Software Engineering 10, 3 (1984), 290–302

Best practices

- Requirements are not stable, use of change management recommended
- Intense and long-during analysis leads to paralysis at the cost of the realization phase
- Use short term contracts (less fuction in less time) over long term contracts (all functions with behavior testing after years)

Charts in Requirement Analysis

	Structure	Behavior
Analytical Specification (Lastenheft)	Analytical Class- Object- Packet-	User Story Use-Case- Activity-
Technical Specification (Pflichtenheft)	Technical Class- Components- Distribution-	State- Sequence- Communication- Interaction-

Above assignment are not that strict, take what best suits your needs