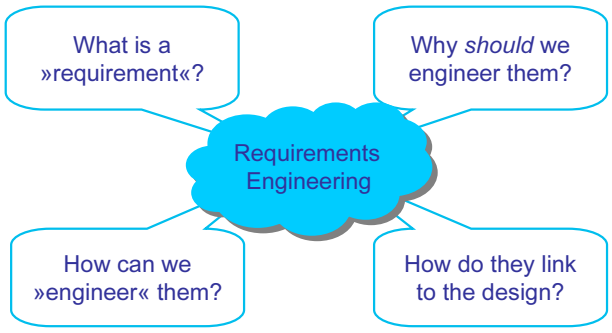
Three Lectures on Requirements Engineering

Dr. Frank Padberg
May, 2009

Lecture 1

Raising questions



engl. *requirement*: Anforderung, Bedingung, Maßgabe

IEEE: »Requirement«

(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).

Standard Glossary of Software Engineering Terminology
(ANSI/IEEE Standard 610.12-1990)

Why should we engineer requirements?

the task is important:

- the reqs are the basis for the downstream development
  - the reqs must be correct and complete in order to build the right software product
  - errors in the reqs are costly
- to a large extent, development communicates with the customer via reqs

Why should we engineer requirements? (cont.)

the task is difficult:

- the reqs are often unclear in the beginning
- projects must handle large numbers of reqs

the task is recurring:

- every project must handle reqs, all the time

> we can no longer do reqs "ad-hoc" but need a systematic approach – that is, engineering

IEEE: »Software Engineering«

...is the application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance, and retirement of software, that is, the application of engineering to software

Standard Glossary of Software Engineering Terminology
(ANSI/IEEE Standard 610.12-1990)

IEEE: »Requirements Analysis«

(1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.

(2) The process of studying and refining system, hardware, or software requirements.

(1) studying user needs
(2) digging deeper into the reqs

Standard Glossary of Software Engineering Terminology
(ANSI/IEEE Standard 610.12-1990)

Analysis versus design

- clearly distinguish between "what" software to develop and "how" to make it work
- this distiction is indispensable for today's large systems
- analysis is about the software's functionality and its properties – *not* about technical solutions or even implementations

Business Process Analysis

Business process analysis (1)

Your customer wants to do business.
He thinks in terms of his business, not in software engineering terms. As a software analyst, it is your task...

- to study your customer's application domain;
- to learn your customer's business language;
- to find out what his business processes look like and model them;

Business process analysis (2)

- to find out which business steps your customer wants to keep or change;
- to find out which business steps your customer wants to support by software.

This is often difficult and time-consuming, but inevitable in order to meet your customer's needs.

The requirements engineer

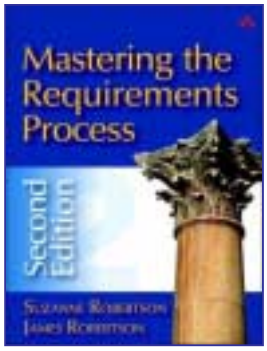
You as a requirements analyst and engineer

Your customer's business world

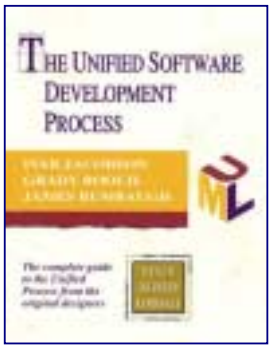
Your software world

RE Processes

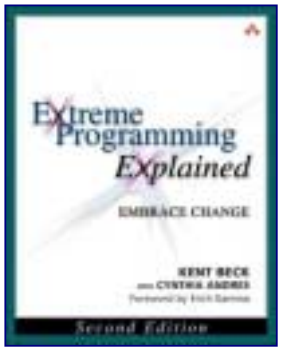
»Volere« RE Process



Rational Unified Process (RUP)



Extreme Programming (XP)



General SE, including RE

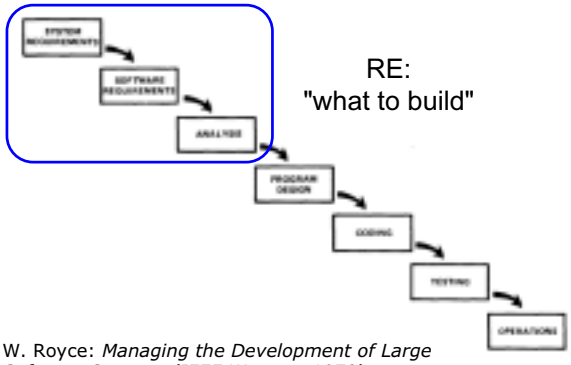
Rationale for up-front RE

"We must know exactly what to build before we can build it"

- the classical engineering viewpoint
- makes sense – in general
- also results in "design before coding"
- leads to the waterfall software process

engl. *up-front*: im Voraus

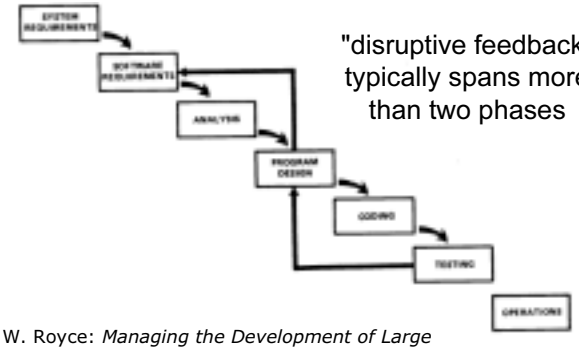
Classical waterfall process (1)



RE:
"what to build"

W. Royce: *Managing the Development of Large Software Systems* (IEEE Wescon, 1970)

Classical waterfall process (2)



"disruptive feedback" typically spans more than two phases

W. Royce: *Managing the Development of Large Software Systems* (IEEE Wescon, 1970)

Agile requirements processes

Agile processes, such as XP, start with requirements, too...

– but in much smaller units
– and in much shorter cycles
– and much less formal
– and much closer in touch with the customer
– and much closer in sync with coding

...than traditional software processes.

The Volere Requirements Process

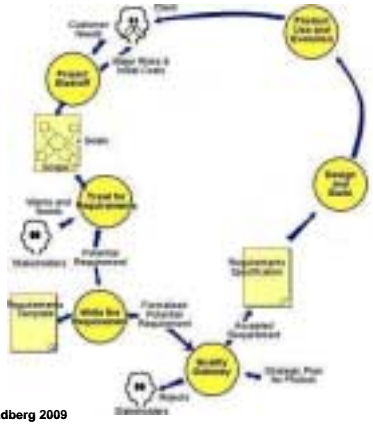
»Volere«

the dictionary tells us: (ital.) *volere* means

»to want« »wollen«
»to wish« »mögen«

> "the RE process is about finding out what the customer wants the product to be"

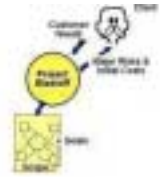
Volere RE process loop



taken from Robertson & Robertson

simplified – shows only the core loop

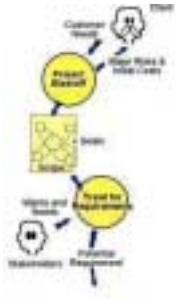
Step 1: »Project Blastoff«



Main Activity:
outline the business context of the project

Main Output:
+ work context diagram
+ project goals

Step 2: »Trawl for Requirements«

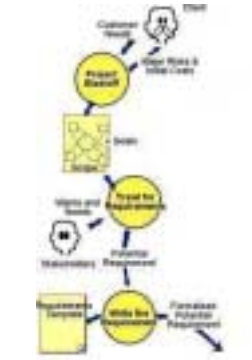


Main Activity:
analyze the business processes that the software must support

Main Output:
+ process descriptions
+ potential requirements for the software

this is the most laborious and comprehensive RE step

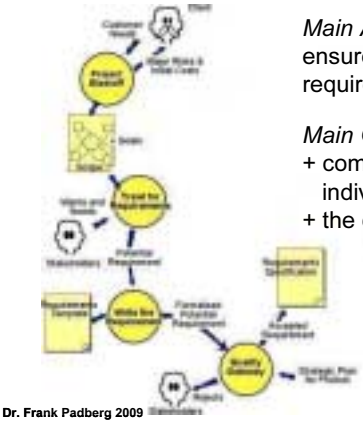
Step 3: »Write the Requirements«



Main Activity:
cast the requirements and the other analysis results into a structured form

Main Output:
+ formalized potential requirements for the software
+ the initial specification document

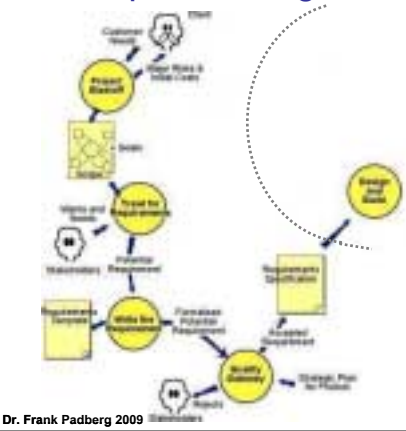
Step 4: »Quality Gateway«



Main Activity:
ensure a high quality of the requirements

Main Output:
+ complete and accurate individual requirements
+ the complete specification

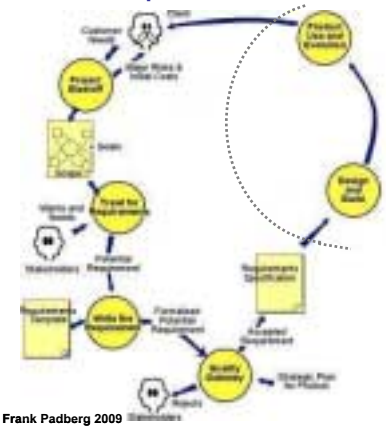
Step 5: »Design and Build«



Main Activity:
software development
(design, coding, test)

Main Output:
the software

Step 6: »Product Use«



Main Activity: field usage of the software

Main Output: new customer needs

Volere Step 1: Project Blastoff

Project Blastoff

- also known as "kickoff" or "initiation"
- explore, understand, and outline the business context of the project
- provides the basis for the detailed analysis of business processes and reqs
- requires a lot of communication with the client and customer

Blastoff – Deliverables

a) Purpose of the project.
b) The scope of the work.
c) The stakeholders.
d) Constraints.
e) Names.
f) Relevant facts and assumptions.
g) The estimated cost.
h) The risks.
i) Go/no-go decision.
~~j) A first-cut low-fidelity prototype.~~

this is an inconsistency in Robertson's book

engl. *deliverable*: Lieferung

Purpose of the project

- a short, quantified statement of what the product is intended to do and what advantage it brings to the business
- explains why the business is investing in the project and the benefit it wants to achieve
- justifies the project
- the project goal is the highest-level requirement

Running Example

Your customer, the County Highway Dept., tells you:

> Roads freeze in winter, and icy conditions cause road accidents that kill people. We need to be able to predict when ice will form on a road so we can schedule a de-icing truck to treat the road in time. We expect a new system to provide more accurate predictions of icy conditions. This will lead to more timely de-icing treatment than at present, which will reduce road accidents. We also want to eliminate indiscriminate treatment of roads, which wastes de-icing compounds and causes environmental damage.

Robertson p. 42

Ex: Project goals (1)

Roads freeze in winter, and icy conditions cause road accidents that kill people. We need to be able to predict when ice will form on a road so we can schedule a de-icing truck to treat the road in time. We expect a new system to provide more accurate predictions of icy conditions. This will lead to more timely de-icing treatment than at present, which will reduce road accidents. We also want to eliminate indiscriminate treatment of roads, which wastes de-icing compounds and causes environmental damage.

Ex: Project goals (2)

- Purpose:
  "accurately predict road freezing times and timely schedule the de-icing treatment"
  Benefit:
  reduce road accidents
- Purpose:
  "eliminate unnecessary treatment of roads"
  Benefit:
  save money on winter road maintenance and reduce damage to the environment

Work Scope

Scope of the work

- "the work" means the business area affected by the installation of your product
- you must first understand and scope the business area, then the future product
- you need to identify the conceptual domains that are relevant in this business context
- you need to identify the external systems that the work interfaces with

engl. *scope*: Abgrenzung, Rahmen, Geltungsbereich, Reichweite, Umfang
engl. *domain*: Bereich, Domäne

Ex: Relevant domains

domain "roads" domain "trucks" domain "weather"

Roads freeze in winter and icy conditions cause road accidents that kill people. We need to be able to predict when ice will form on a road so we can schedule a de-icing truck to treat the road in time. We expect a new system to provide more accurate predictions of icy conditions. This will lead to more timely de-icing treatment than at present, which will reduce road accidents. We also want to eliminate indiscriminate treatment of roads, which wastes de-icing compounds and causes environmental damage.

domain "scheduling" domain "predictions"

Ex: Systems in the domains (1)

- domain "weather":
  - weather forecasting services
  - weather stations owned by the customer
  - suppliers of thermal maps (temperature differentials along the roads)



- domain "roads":
  - the county's road engineering dept. (builds and maintains the roads)

- domain "trucks":
  - the county's truck depot

Ex: Systems in the domains (2)

- domain "scheduling"?
- domain "predictions"?

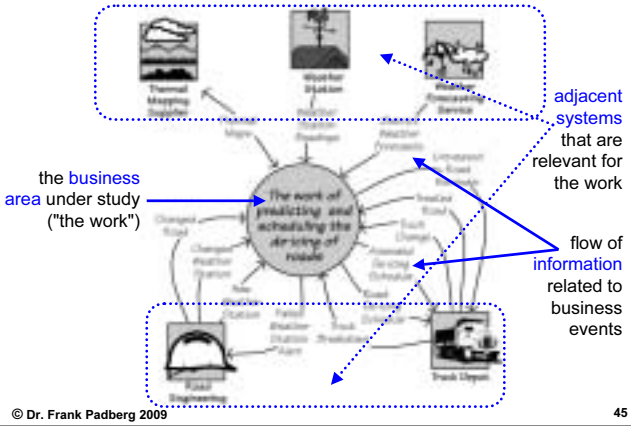
these domains are part of "the work"!

Work context diagram

- a simple diagram that shows the scope of the customer's business area ("the work")
- the diagram depicts adjacent systems and the flow of information between the systems and "the work"
- coarse-grained
- the software to be built eventually becomes part of "the work"

engl. *adjacent*: angrenzend

Ex: Work context diagram



Ex: Information flows (1)

input from an adjacent system – triggers a business event

Ex: Information flows (2)

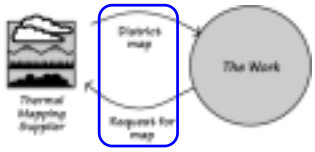
output to an adjacent system – part of the response to a business event

Ex: Information flows (3)

output to an adjacent system – triggered by an internal timing event (this *is* a business event)

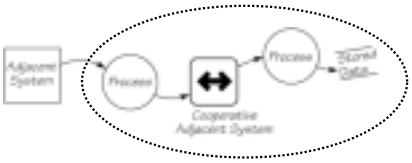
Ex: Information flows (4)

request and response to and from a cooperative adjacent system – as part of processing a business event



Cooperative adjacent system

- provides data upon request during the processing of a business event
- that is, might *appear* as part of "the work," but actually *is* a separate, independent system



processing a business event

Ex: Business events (1)

Event	Information (I/O)
Weather Station transmits reading	Weather Station Readings
Weather Bureau forecasts weather	District Weather Forecasts
Road engineers advise changed roads	Changed Road
Road Eng. installs new weather station	New Weather Station
Road Eng. changes weather station	Changed Weather Station
Time to test weather stations	Failed Weather Station Alert
Truck Depot changes a truck	Truck Change
Time to detect icy roads	Road de-icing Schedule
Truck treats a road	Treated Road
Truck Depot reports problem with truck	Truck Breakdown Amended de-icing Schedule
Time to monitor road de-icing	Untreated Road Reminder

Ex: Business events (2)

- have a total of 11 business events
- ...but 13 information flows?!
- one particular business event ("Truck Depot reports problem with truck") involves two different information flows ("Truck Breakdown" and "Amended De-icing Schedule")
- one particular information flow ("Thermal Maps") is to a cooperative adjacent system ("Thermal Mapping Supplier") and, hence, does *not* trigger a business event

Stakeholders

Stakeholders (1)

- the dictionary tells us:
  - *stake* = a share or interest in a business or a given situation
  - *stakeholder* = a person or organisation with a legitimate interest in a given situation, action, or enterprise
- wikipedia tells us:

  corporate stakeholder = a person, group, organization, or system who affects or can be affected by an organization's actions

engl. *stakeholders*: Interessengruppe

Stakeholders (2)

- in RE: replace "organization's actions" with "system [or software] requirements"
- Robertson (p. 45): stakeholders are the "source of requirements"

Client ≠ Customer ≠ User

three important groups of stakeholders in RE:

- client:
  pays for the development of the software product
- customer:
  buys the software product
- user:
  actually uses the software product

Ex: Stakeholders

- client = Saltwork Systems, represented by chief executive Mack Andrews
- customer = Northumberland County Highways Dept., represented by director Jane Shaftoe
- potential additional customers = other counties in the UK
- users = highways dept. clerks; truck depot supervisors; road eng. dept. supervisors

More stakeholders

other potential stakeholders in RE:

- managers
- consultants
- technical experts
- marketing people
- lawyers
- negative stakeholders
  (they do *not* want the product)

Blastoff – More deliverables (1)

- constraints (solution and project)
  - Are there any existing technical solutions that must be used?
  - How much time and money are available?
- names, relevant facts, assumptions
  - Is any special terminology used by this part of the organization?
  - Are there any facts or assumptions that may affect the outcome of the project?

special terminology typically is captured in a "glossary"

Blastoff – More deliverables (2)

- estimated project cost
- short analysis of the main risks faced by the project
- go/no-go decision
  - Is the project viable?
  - Does the cost of producing the product make it worthwhile?
  - Do you have enough information to proceed with the requirements activity, or should you ask for more time to learn more?

connects to project management

engl. *viable*: durchführbar

Costing and Go/No-go

- at this stage, there is too little information to sensibly estimate the *software* cost – but it may be possible to estimate the cost of the *reqs analysis* phase
- the go/no-go decision often is made just for the reqs analysis phase
- in today's contracts, the client usually has several "exit points" for the project

Cross-check against Pressman's Book

Pressman: RE steps

Pressman (5th edition) chapter 10 – taken from Sommerville's book on RE:

1. reqs elicitation
2. reqs analysis (and negotiation)
3. reqs specification
4. *system modeling* ← should come first!
5. reqs validation
6. reqs management

engl. *to elicit*: entlocken

SafeHome example

We would like to enter the market for home security systems by building a microprocessor-based system that would protect against and/or recognize a variety of undesirable "situations" such as illegal entry, fire, flooding, and others. The product, tentatively called *SafeHome,* will use appropriate sensors to detect each situation, can be programmed by the homeowner, and will automatically telephone a monitoring agency when a situation is detected.

Pressman (5th) p. 277

SafeHome context diagram

Pressman (5th) p. 322

Volere Step 2: Trawling for Requirements

Trawling (1)

- the analyst seeks to understand the desired functionality of the future software
- to this end, he must analyze the business processes that the software must support
- he describes and models the business processes, then "trawls" the models for requirements

engl. *to trawl*: mit dem Netz fischen

Trawling (2)

- starting point are the results from the blastoff:
  - work context diagram
  - table of business events
  - list of stakeholders
- main output:
  - business process descriptions
  - potential requirements
- concept for partitioning: business use case

Business use cases

- the scope of "the work" is bounded by the communication with the adjacent systems
- business events happen in the adjacent systems (or, are time-triggered)
- the data flow resulting from a business event reaches "the work"
- "the work" responds to the event with pre-planned actions = a business use case
- that is, business use cases link to events

Use case – original definition

a sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor

I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard: *Object-Oriented Software Engineering – A Use Case Driven Approach* (ACM Press, 1992)

Current and to-be processes

- the first goal is to understand the work as-is
- models help by providing different views of the business processes
- ask questions about...
  - what the analyst doesn't know yet
  - what the analyst doesn't understand yet
  - what the business people don't know yet
- study the current work, but also possible improvements to it as a secondary goal

Trawling – Important techniques

- use case workshops
- interviews
- apprenticing
  (observing a user "in action")
- studying existing documents
  (such as process descriptions, problem reports, work artifacts)
- prototyping
- diagramming

how to "squeeze" all the needed information from the stakeholders?

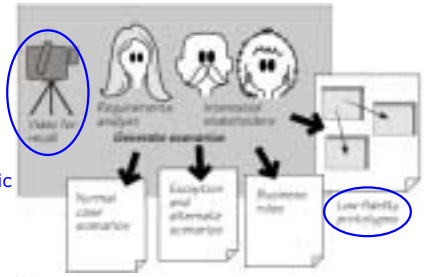
typically, the analyst employs a combination of these techniques

engl. *to apprentice*: in die Lehre gehen

Volere Step 2a: Analyzing the Business Processes

Use case workshops

the analyst works with a manageable group of stakeholders to find the scenarios for each use case



somewhat problematic

Scenario

- describes what is being done by the business use case as a series of steps
  (recall: in response to a business event)
- 3 to 10 steps
- stakeholder-friendly description
- "broad-brush picture"
- stakeholders are invited to participate and revise the scenarios until they represent a consensus view of what the work should be

Ex: Business scenario

BE-10: *Truck depot reports problem with truck*

1. Truck breaks down while on the road.
2. Truck radios the truck depot.
3. Truck depot supervisor tells the de-icing work about the truck problem.
4. "The work" reschedules trucks so as to disperse the broken truck's allocation among the remaining trucks.
5. "The work" updates the truck depot supervisor on the amended de-icing schedule.
6. Truck depot supervisor implements the new schedule.

Kinds of scenarios

- normal case scenario:
  shows the actions if all goes well – that is, no mistakes and nothing unusual happens
- alternative case scenario:
  shows alternative normal case arising from intentional choices made by the user
- exception case scenario:
  shows the actions taken to safely rejoin the normal case after an exception occured

Outlook

Is all this relevant for me?

Yes, absolutely!

- Don't believe that you won't have to do RE when you become a developer in industry after finishing your degree.
- Successful software companies assemble a team for each client, and each team member actively participates in all phases of the software project – including requirements.
- So much for that.

Warning notice

The second edition of Robertson's book includes "agility guides" at the beginnings of the chapters – keep in mind that the Volere process is traditional, not agile, and that these sections have been pasted to the book's first edition mainly for marketing reasons.

Next lecture's topics

- business scenarios
- product scoping
- diagramming
- functional requirements

End of Lecture 1