

Three Lectures on Requirements Engineering

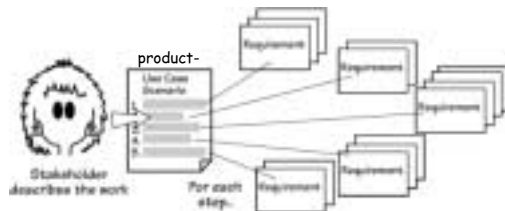
Dr. Frank Padberg
May, 2009

Lecture 3

© Dr. Frank Padberg 2009

2

Where are we now?

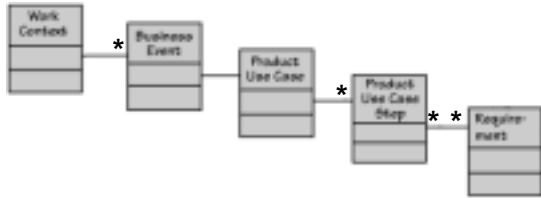


we are still in Step 2, trawling the product scenarios for requirements...

© Dr. Frank Padberg 2009

3

Recall: From work context to requirements



Recall: Functional requirements

Functional requirements describe **what the product must do** to carry out the work for which it is intended. They are independent of any technology used by the product.

⇒ The product shall record the new weather stations.



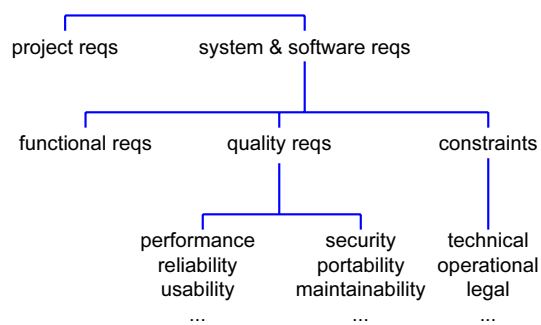
Quality Requirements

More requirements

- software always has to meet a certain quality level; f.e., think of throughput or usability
- software may also have to comply with certain technological or legal constraints
- such properties are *additional requirements* – the qualities that the product and its functions must have

such reqs are often called "non-functional,"
but this is a nonsense term...

A requirements hierarchy



Scope of quality requirements

quality reqs have different scopes:

- some apply to a particular functional req
- some apply to a use case as a whole
- some apply to the entire product

the quality reqs often are crucial
for the success of the product

Trawling for quality reqs (1)

- quality issues naturally come up when trawling for functional requirements
- hence, quality reqs get collected "over time" during analysis
- in addition, the analyst must ask specific quality-related questions when discussing product use cases and functional reqs
- use a checklist of quality categories (see, f.e., Robertson p. 175)

© Dr. Frank Padberg 2009

10

Trawling for quality reqs (2)

- when talking to users of the product, ask for
 - usability – look and feel – security – cultural reqs
- when talking about the operating environment of the product, ask for
 - performance – interface – maintainability – security reqs
- when talking to managers and marketing people, ask for
 - look and feel – political – security – legal reqs

© Dr. Frank Padberg 2009

11

Ex: A functional requirement

Produce road de-icing schedule

6. Product schedules available trucks from the relevant depots.
7. Product advises the engineer of the schedule.

this product use case scenario will give a functional requirement:

⇒ The product shall produce a de-icing schedule showing the roads to be treated and the trucks allocated to them.

© Dr. Frank Padberg 2009

12

Ex: A usability requirement

- for sure, the stakeholders want the schedule to enable an engineer to *easily* direct the trucks to the correct roads
- this gives a usability requirement:

⇒ The product shall produce a schedule that is easy to read.

Ex: Another usability requirement

- when thinking about the targeted users of the product, more usability reqs come up
- f.e., the customer tells you that they often employ new junior engineers
- this gives another usability requirement:

⇒ The product shall be intuitive to use.

- but how to make sure that this property has been achieved?

Fit criteria

- to clarify a quality req, it helps to establish a **fit criterion** that will measure whether the software meets the req or not

⇒ The product shall be intuitive to use.
Criterion: A new engineer shall be able to produce a correct de-icing forecast within 10 mins of encountering the product for the first time without reference to any out-of-product help.

- conformance could be tested in a user study

Ex: Look and feel requirements

⇒ The product shall conform to the established look and feel of the organization's products.

Criterion: Sixty percent of the target audience will recognize the product as belonging to the corporation within 5 secs of encountering it for the first time.

note that this req does *not* say the company logo must be prominent, nor does it talk about the colors to be used; simply conform to whatever the standards are

Ex: Performance requirements

⇒ The product shall accommodate the largest geographical area of any road authority in the UK.

again, a fit criterion should be added

alternatively, when specifying numbers a **rationale** should explain the numbers:

⇒ The product shall have the capacity for 5000 roads.

Rationale: The maximum number of roads in the area of any potential customer for the product.

Ex: Performance fit criteria

⇒ The product must produce the road de-icing schedule in an acceptable time.

gets clearer when thinking about how **failure** of this req could be measured:

⇒ The product must produce the road de-icing....

Criterion: The road de-icing schedule shall be available to the engineer within 15 secs from making his request for 90 percent of the times. It shall never take longer than 20 secs.

Ex: Maintainability requirements

⇒ The product shall enable the addition of new road authority areas within two days.

already contains its fit criterion; conformance could be tested in a user study

Ex: Legal requirements

⇒ The product shall comply with ISO 93.080.99.

legal reqs often lead to additional *functional* reqs, f.e.:

⇒ The product shall produce an audit report of all road schedules and their subsequent treatments.

⇒ The product shall retain all ice predictions for all occasions the schedule is run.

Ex: Operational constraints

⇒ The product shall interface with the DM31 weather stations.

Criterion: The interfaces to the Rosa Weather Station shall be certified as complying with the National Transportation Communication ITS Protocol (NTC/IP).

again, this constraint will lead to additional functional reqs, such as establishing a radio network connection, matching data formats ...

Ex: Project constraints

time...

⇒ The product must be available at the beginning of the new tax year.

...and money

⇒ The budget for building the product is £500,000.



Volere Step 2d: Modeling the Data

Business Data

- during analysis, model **only business data**
- business data objects naturally occur in the stakeholder's process descriptions
- model the **statics** – business data objects and their logical relationships
- **caution**: it is tempting, but doesn't make much sense to specify "operations" for data objects during analysis – operations are *design* information

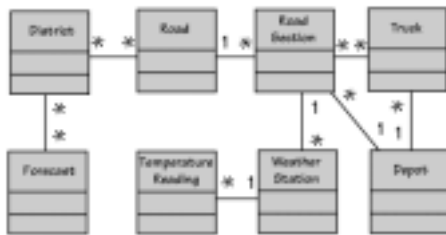
Variety of data diagrams

- long-standing:
 - entity-relationship diagram (ERD)
 - statechart
 - state transition table
- with UML:
 - class diagram
 - state diagram

© Dr. Frank Padberg 2009

25

Ex: Business objects as a class diagram



a simple UML class diagram that shows key **business data** objects and their relationships

© Dr. Frank Padberg 2009

26

Ex: Functional reqs from data

- when talking about the business data objects, additional reqs naturally come up
- f.e., business data often must be stored in the product – a functional req:

⇒ The product shall record air-temperature readings, humidity readings, precipitation readings, and road-temperature readings received from weather stations.

engl. *precipitation*: Niederschlagsmenge

© Dr. Frank Padberg 2009

27

Ex: Quality reqs from data

- data often lead to additional quality reqs
- f.e., performance reqs:

⇒ The product shall record data transmitted from a weather station within half a second.

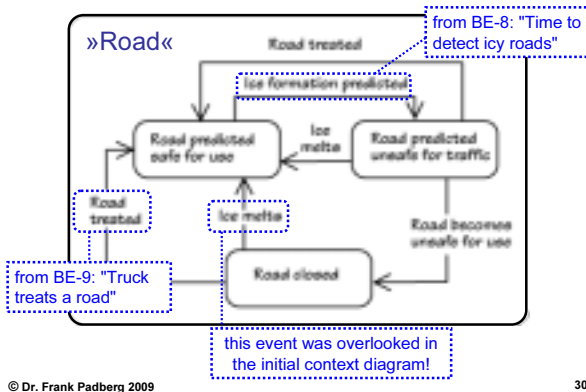
- f.e., security reqs:

⇒ The product shall ensure that data recorded from a weather station cannot be altered, except by a supervising engineer.

State diagrams

- show the states of an object, plus the events that make the object move to its next state
- here: business data object and business events
- often helps to identify additional requirements
- **caution:** state diagrams have a tendency of getting too close to design...

Ex: Business object states



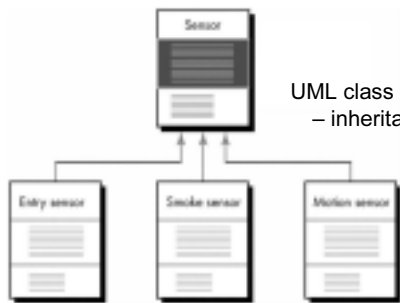


Cross-check against Pressman's Book

© Dr. Frank Padberg 2009

31

SafeHome: Sensor data model



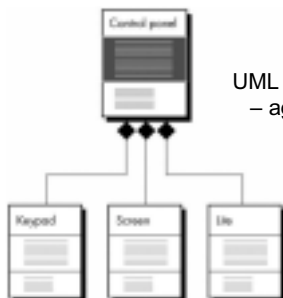
UML class diagram
– inheritance –

© Dr. Frank Padberg 2009

Pressman (5th) p. 589

32

SafeHome: ControlPanel data model



UML class diagram
– aggregation –

© Dr. Frank Padberg 2009

Pressman (5th) p. 589

33

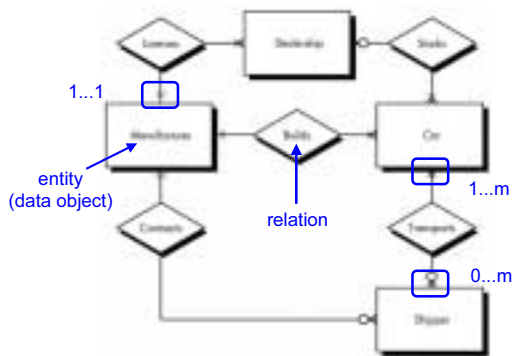
Entity-relationship diagram (ERD)

- originally invented in the 70s for the design of relational databases
- shows data objects, attributes, relationships, and cardinalities
- is the point of origin of the OO class diagrams of various sorts

© Dr. Frank Padberg 2009

34

Sample ERD



© Dr. Frank Padberg 2009

Pressman (5th) p. 308

35

SafeHome: Multiple relationships



simple ERD form where relationships are not drawn as diamonds but appear as labels at the arrows;
it almost looks like a UML class diagram...

© Dr. Frank Padberg 2009

Pressman (5th) p. 321

36

Data dictionary

- an organized listing of all data elements that are pertinent to the system
- gives precise, rigorous definitions
 - inputs and outputs (information flows)
 - component of stores
- may even elaborate on intermediate data calculations
- invented in the late 80s

engl. *pertinent*: gehörig

© Dr. Frank Padberg 2009

37

Sample data dictionary entry

Name:	telephone number
Aliases:	phone number, number
Where/how used:	read-phone-number (input) display-phone-number (output) analyze-long-distance-calls (input)
Description:	telephone no. = {local extension outside no. 0} outside no. = 0+{area code domestic no.} service code = {211 411 811 911} domestic no. = {0}+area code + local number area code = "three numerical digits"
Format:	alphanumeric data

Pressman (5th) p. 330

© Dr. Frank Padberg 2009

38

Ex: Data dictionary entry for weather station reading



- weather station reading =
- + weather station identifier
 - + temperature
 - + moisture content of road surface
 - + date and time

© Dr. Frank Padberg 2009

39

Writing individual requirements

- is actually done during trawling
- the individual reqs must be written down in a **structured form** ("formalized")
- helps with bookkeeping
- helps **avoid defects**, such as reqs that are incomplete or not traceable to a use case
- uses a **template**, in paper form or electronic form

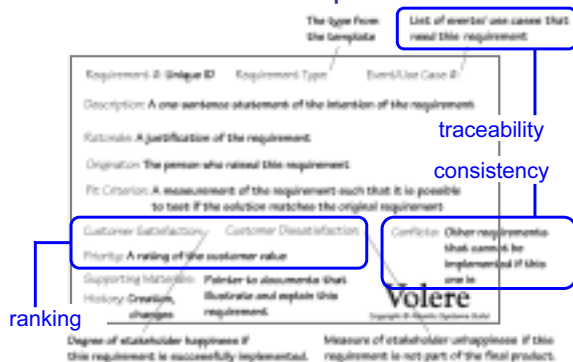
Volere »Shell«

Requirement #: _____	Requirement Type: _____	Event/User Case #: _____
Description: _____		
Rationale: _____		
Originator: _____		
Fit Criterion: _____		
Customer Satisfaction: _____	Customer Dissatisfaction: _____	Conflicts: _____
Priority: _____		
Supporting Materials: _____		
History: _____		

Volere
Copyright © Holm Systems GmbH

an index card with pre-printed structure for writing **one** req

Volere shell explanation



Conflicting requirements

- one particular req can make another one less feasible; for example:

⇒ The product shall compute the **shortest** route to each destination.

⇒ The product shall compute the **quickest** route to each destination.

- nothing unusual when asking different people
- must be resolved, with the client

© Dr. Frank Padberg 2009

46

Ex: Functional req in a shell

Requirement #: 75 Requirement Type: B Event/Use Case #: 7, 8

Description: The product shall record all the roads that have been treated.

Rationale: To be able to schedule untreated roads and highlight potential danger.

Originator: Arnold Bross, Chief Engineer

Fit Criterion: The recorded treated and untreated roads shall agree with the drivers' road treatment logs.

Customer Satisfaction: 3 Customer Dissatisfaction: 5

Priority: Conflicts:

Supporting Materials:

History: Created February 28, 2006

Volere
Support of Teams Systems Ltd.

© Dr. Frank Padberg 2009

47

Ex: Quality req in a shell

Requirement #: 110 Requirement Type: M Event/Use Case #: 6

Description: The product shall be easy for the road engineers to use.

Rationale: It should not be necessary for the engineers to attend training classes to be able to use the product.

Originator: Soela Henning, Road Engineering Supervisor

Fit Criterion: A road engineer shall be able to use the product to successfully carry out the critical use cases within 1 hour of first encountering the product.

Customer Satisfaction: 3 Customer Dissatisfaction: 5

Priority: Next release Conflicts: None

Supporting Materials:

History: Raised by AG, 25 Aug 05.

Volere
Support of Teams Systems Ltd.

© Dr. Frank Padberg 2009

48

»The spec«

- a **document** that contains all requirements
- **SRS** = Software Requirements Specification
- must contain clear, complete, and testable instructions about what to build
- is the basis for the contract with the client
- is clearly structured – uses some sort of **template**
- can be paper, but is often stored using a software tool for reqs management

© Dr. Frank Padberg 2009

49

Writing the spec

this means:

assembling all the analysis results

- blastoff deliverables – project purpose, constraints, work context diagram
- trawling results – product use case diagram, product scenarios and process diagrams, data models, functional reqs and quality reqs
- think of this as *building*, not writing

© Dr. Frank Padberg 2009

50

Volere »Reqs Spec Template«

Project Drivers

- Purpose of the project ✓
- Client, customer, and other stakeholders ✓
- Users of the product ✓

Project Constraints

4. Mandated constraints
5. Naming conventions and definitions
6. Relevant facts and assumptions

Functional Requirements

7. The scope of the work ✓
8. The scope of the product ✓
9. Functional and data reqs ✓

Nonfunctional Requirements

10. Look and feel reqs ✓
11. Usability and humanity reqs ✓
12. Performance reqs ✓
13. Operational and environmental reqs ✓
14. Maintainability and support reqs ✓
15. Security reqs
16. Cultural and political reqs
17. Legal reqs ✓

Project Issues

18.
[f.e., risks; cost; solution ideas]
27.

© Dr. Frank Padberg 2009

51

IEEE Guide to SRS

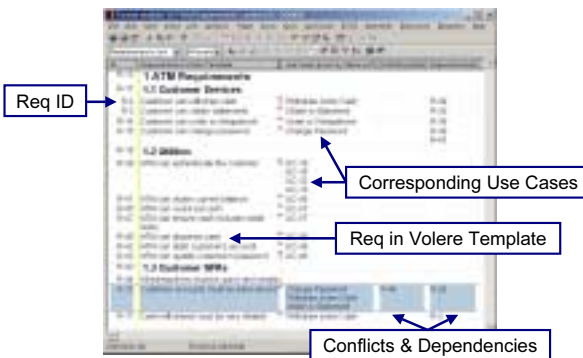
1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, Acronyms, and Abbreviations
 - 1.4 References
 - 1.5 Overview
 2. General Description
 - 2.1 Product Perspective
 - 2.2 Product Functions
 - 2.3 User Characteristics
 - 2.4 General Constraints
 - 2.5 Assumptions and Dependencies
 3. Specific Requirements
 - 3.1 Functional Reqs
[organized by req id]
 - 3.2 External Interface Reqs
[UI, HW, SW]
 - 3.3 Performance Reqs
 - 3.4 Design Constraints
 - 3.5 Attributes
[security, maintainability...]
 - 3.6 Other Reqs
[data base, operations...]
- Appendixes
Index

(ANSI/IEEE Standard 830-1984, resp., -1998)

© Dr. Frank Padberg 2009

52

Telelogic DOORS



© Dr. Frank Padberg 2009

53

A word on tools

"Despite the claims of some vendors, there is no requirements tool that can actually interview a user or determine what your client really needs."

(Robertson, p. 348)

© Dr. Frank Padberg 2009

54

Where are we now?



© Dr. Frank Padberg 2009

55

The flowchart on the left shows a sequence of steps: 'Use Case', 'Requirements', 'Analysis', 'Design', 'Implementation', 'Testing', and 'Deployment', connected by arrows. To the right is the cover of the book 'Mastering the Requirements Process' by Dr. Frank Padberg, featuring a classical column.

Volere Step 4: Quality Gateway

© Dr. Frank Padberg 2009

56

Quality gateway

- is the gateway into the specification
- each potential req must be tested before it enters the spec
- that is, we are talking about the **quality of the reqs** and spec itself
- can be (and should be) done **incrementally**, whenever the analysis of a use case is completed

engl. *gateway*: Einfahrt, Tor

© Dr. Frank Padberg 2009

57

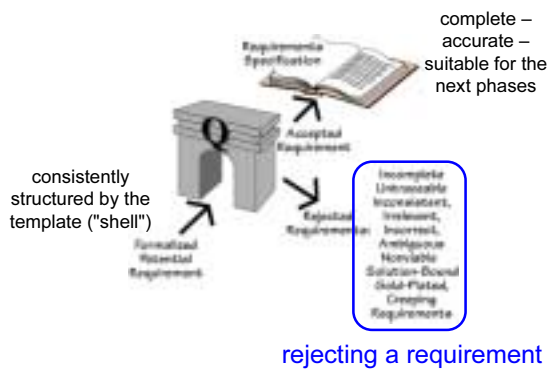
Why test requirements?

recall that most defects originate in requirements or design, and are expensive to correct downstream

© Dr. Frank Padberg 2009

58

The gateway



© Dr. Frank Padberg 2009

59

Completeness and consistency

- completeness and traceability of reqs to business use cases should be ensured by using the template (or reqs tool)
- but better check again...
- for consistency, check that terms are used consistently and match their definition given in the glossary



© Dr. Frank Padberg 2009

60

Ex: Ambiguity (1)

⇒ The product shall show the weather for a day.

a somewhat ambiguous requirement:

- its meaning depends on the context within the spec
- forecast for the next 24 hours, or until the end of this day?
- could easily be specified more precisely

Ex: Ambiguity (2)

⇒ The product shall communicate all roads predicted to freeze.

another somewhat ambiguous requirement:

- its meaning depends on the context within the spec
- every road known to the product, or just the set of roads currently examined by the user?
- it helps to group reqs by product use case
- but beware of "overspecifying" – this can make specs really hard to read; recall that you are just giving a *description* of the req

Ex: Relevance (1)

⇒ The product shall maintain a lookup table of the times of sunrise and sunset throughout the year.

an irrelevant requirement:

- the temperature of the road surface matters, not day or night
- discard this req

Ex: Relevance (2)

⇒ The product shall record the durations of truck activity.

looks irrelevant, but actually is not:

- de-icing trucks may not be scheduled for more than 20 hours out of each 24-hour period
- hence, this req **indirectly** contributes to the project goal
- this rationale should be given in the template

Ex: Not solutions

⇒ The product shall **beep** and put a **flashing message** on the screen if a weather station fails to transmit readings.

a badly shaped functional requirement:

- it contains the means of implementation
- it consigns the alert message to a screen
- there might be better solutions for the alert (e-mail, diagnostic application, telephone)

⇒ The product shall **issue an alert** if a weather station fails to transmit readings.

better!

Not solutions – more examples (1)

⇒ The product shall **require a password** to access account data.

a badly shaped security requirement:

- again, it contains the means of implementation
- it adopts the current standard solution to access control even if better solutions come up

⇒ The product shall ensure that account data can be **accessed only by authorized users**.

better!

Not solutions – more examples (2)

⇒ The product shall **use a mouse**.

a badly shaped usability requirement:

- it contains the means of implementation
- changing "mouse" to "pointing device" is no better

⇒ The product shall allow the user to **directly manipulate** all interface items.

better!

express any req in its essential form, leaving the designer free to choose the best way to implement it

© Dr. Frank Padberg 2009

67

Ex: Viability

⇒ The truck drivers shall receive weather forecasts and schedule their own de-icing.

an invariable requirement:

- truck drivers cannot predict when a road will freeze and do not know which roads have been treated
- contradicts the constraint "targeted user group"
- discard this req
- other common reasons for invariability:
 - lack of technological skills to build the req
 - lack of resources to build the req
 - lack of acceptance among the stakeholders

© Dr. Frank Padberg 2009

68

Other common reqs problems

- *gold plating:*
features that contribute more to the cost of a software than to its usefulness
- *creep:*
any reqs that enter the spec after it has been considered complete
- *leakage:*
unrecognized and uncontrolled creep

© Dr. Frank Padberg 2009

69


Implementing the gateway

- try to keep it "lean"
 - often done by one analyst, supported by one tester
 - may involve a stakeholder
 - reqs are marked off when approved
- formal reviews may be needed for complex, technical subject matters

engl. *to mark off*: abhaken

© Dr. Frank Padberg 2009

70

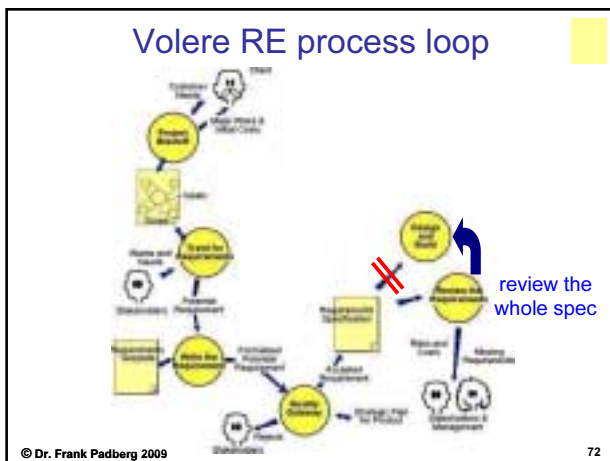


Step 4+: Reviewing the Specification

© Dr. Frank Padberg 2009

71

Volere RE process loop



© Dr. Frank Padberg 2009

72

Quality of the specification

- we need a quality gateway for the spec as a whole
- the gateway must be in the main RE process loop – *before* entering design and build
- the spec must be correct and complete (w/t the current release or iteration)
- the spec must be suitable for its purpose (which is to serve as a basis for the design)

requirements validation

© Dr. Frank Padberg 2009

73

Reviewing the spec

- spans from informal reviews to full, formal software inspections
- review a small pack of use cases at a time – this is an ongoing activity
- check for
 - conflicting reqs
 - missing reqs
 - more quality reqs
 - inconsistent diagrams
 - incomplete data handling
 - risky and costly reqs

connects to software quality assurance, cost estimation, and risk analysis

© Dr. Frank Padberg 2009

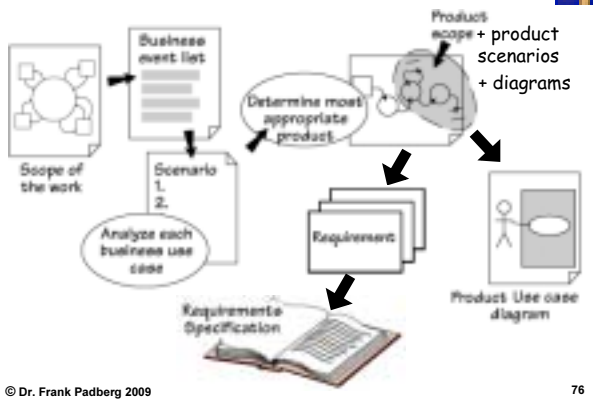
74

Wrap-up

© Dr. Frank Padberg 2009

75

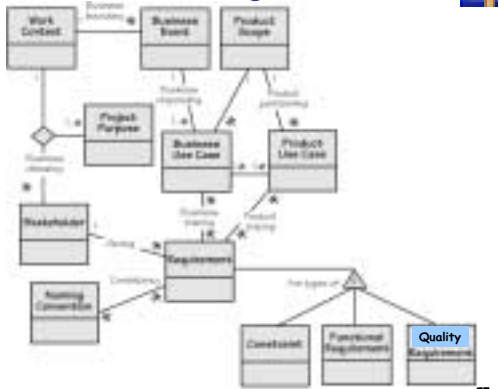
From work scope to specification



© Dr. Frank Padberg 2009

76

Volere knowledge model



© Dr. Frank Padberg 2009

77

Tools

- mostly "low-tech paper"
 - scenario template
 - reqs shell
 - reqs specification template
 - [paper prototypes]
 - some software tools
 - diagramming tools
 - reqs management tools
- recall that low-tech is a good thing when trawling for reqs

© Dr. Frank Padberg 2009

78

The Volere RE process



© Dr. Frank Padberg 2009

79

RE »Leftovers«

- low-tech analysis prototypes
- reuse of requirements
- prioritizing the requirements

see



- inspections for specifications
- requirements change management
- cost estimation

see the
upcoming
lectures

- transition to design

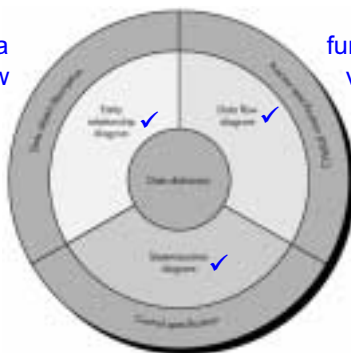
© Dr. Frank Padberg 2009

80

Elements of the analysis model

data
view

function
view



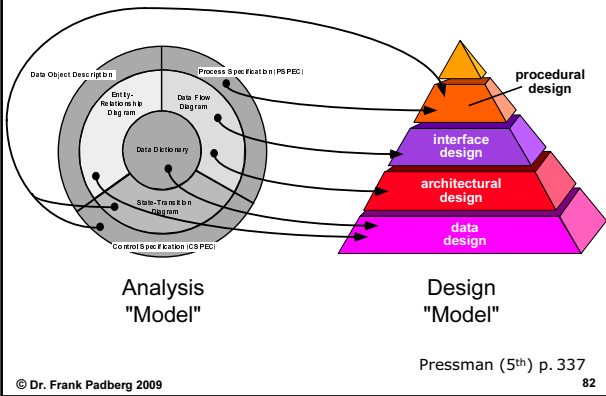
behavior view

Pressman (5th) p. 301

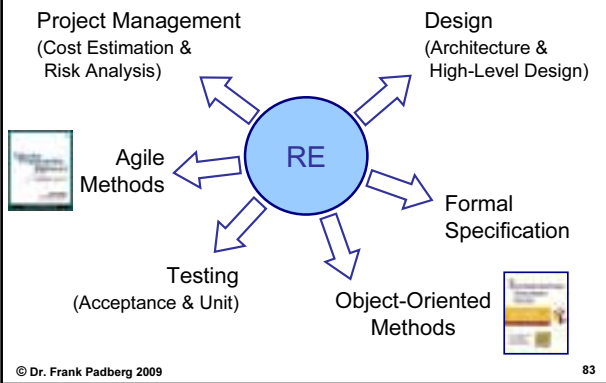
© Dr. Frank Padberg 2009

81

Analysis maps to design



RE – reaching out...



End of the Lectures
