Ch. 4: (Rs) Validation

observed above:

validate **versus** analyze

both have:
- analysis if Rs
- evaluation of Rs
- impact of Rs

but

A&N: rough Rs
"have we got **the right Rs**?"

Val: polished Rs
balanced, agreed, well-formulated
"have we got **the Rs right**?"
(and also, self-evidently
"have we got the right Rs?")

typical problems appearing:

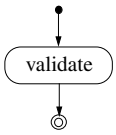
- not conform standard / quality

- badly phrased
ambiguous
sloppy language (English, Dutch)
sloppy formalism: formally incorrect

- flaws in models
form (syntax)
meaning (semantics)

- what has been missed in A&N
conflicts / consistency, completeness

rough process model for validation



input is aggregation of
RDoc
organizational knowledge
general: form, meaning, frame
organizational standards

output is aggregation of
list of problems
agreed actions, among which:
where to resume REprocess

if list is empty, then agreed action is:
stabilize RDoc: approve (&PHD&PUM)

4.1. Rs Review

process-like description

- *plan Rv*
- *distribute Docs (all)*
- *prepare Rv*
- *hold Rv meeting*
- if necessary:
repeat E&A&N
- *check actions*
- *revise RDoc*

more globally:

- actual validating
- if necessary:
repeat E&A&N
- settlement of validation

similar to Fagan inspection, above & below:

rules for prepare & hold meeting

neutral chair
neutral rapporteur
boss absent
different backgrounds & stakeholders
per R:
discuss
choose action
system designers present, e.g. 3 - 10(!)

possible actions

accept / approve -->	ready
clarify -->	rewrite RDoc
supply -->	E&A&N
solve conflict -->	A&N
solidify -->	A&N

pre-review to shorten Rv:

easy form checking,
automatically done / supported

such as

standard structure
spelling
all figures, tables
references
NoToDos

2 continuations after pre-review:
- repair
- list of points, sent with the distribution

full Rv also has a checklist

- clearness
- superfluous
- complete
- ambiguous
- consistent
- structure; similar to standard?
- relations (traceability)

resulting list
- description of each problem, with reference
- agreed action for each problem

note
one can imagine some further aggregation
before restarting E&A&N(&D)

4.2. Prototyping

here it is similar to "program" testing

rough process description

- choose prototype testers:
(the) right people, sufficiently neutral

- develop test scenarios
i.e. build special purpose prototype
OR
extend existing E&A&N prototype

- let the testers execute the scenarios

- document problems

spin off:

- test scenarios are
reusable for eventual system testing

- results provide material for (test) comparison

- thus extended prototype might serve as
spare system / stop-gap system

- (P)UM for prototype
can indicate how to rewrite RDoc
might be first step towards eventual UM

4.3. Model Validation

review of model(s) is much more technical

because of

- syntax
- semantics
- consistency
  - within one model
  - between models
  - with reality

e.g. wrt UML
- semantics are only partially defined
- consistency is very weak,
  - in particular wrt dynamics

moreover:
consistency with reality always is
  notoriously hard

reformulation of models helps

- in plain living language
  - just for better understanding
  - by modeler as well as by others

- in (more) formal language(s)
  - just for better analysis
  - such as model checking, theorem proving

similarly, animation / simulation helps
  - is like prototype
  - presents scenarios
  - can handle more general **what-if**s

4.4. Rs Testing

scenario for testing:
  a Requirement in the eventual system

developing tests thus provides
  a critical Rs study

- where is R relevant

- correspondence with other Rs

- ?: one or more scenarios per R
  if more, actually more than one R??

- reformulation of R:
  more conforming to test scenario

above gives rise to test record form:

- R's identification
- corresponding Rs
- test scenario description and explanation
- testing problem --> R's problem
- comment / recommendation

testing is less suitable / impossible for

- overall Rs:
  having a system-wide characterization

  a test could be done, but relation between
    R and system remains unclear
    as everything is involved

- exclusive Rs:
  those forbidding specific behaviour

  they are non-operational, declarative
    thus one has to try everything

- non-functional Rs:
  such as reliability, security, evolvebility,

  typically: aspect-like
    cross-cutting everything
    thus one has to try (nearly) everything

so we see,
- a too large test set is common
this often makes the underlying R less clear