Ch 5: Rs Management

!!!: change (of Rs) is **normal**

even **after** approvement !!!

but it also is

cause of serious difficulties

hence:

manage the changes
and their consequences

Rs management covers

- changes of **approved** Rs

- relations between Rs & other Rs
& (possible) consequences for other Rs
from changes

- relations between Rs &
& other documents / SE products
& (possible) consequences for these
docs and products from changes

why is change so normal?:

- errors / bugs	(RE / SE)
- growing insight	(stakeholder)
- active world	(circumstances, fashion, law, policy, clients, technology)

useful means for Rs management

traceability: ~ “connectivity & causality”

- proposed by whom

- why is it a R?

- relation with other Rs

- relation with other info
“before”:
problem domain
organisation
problem situation
stakeholder support
“after”
design
code
test
(P)UM - (Preliminary) User Manual

Rs management is heavily time consuming

- compare the short term profit
with the long term profit

- CASE tools provide support for
so-called change management

- DB for Rs

- document / report
can be starting point for filling DB
via analysis
can be extracted from DB
via generation

- traceability support: allowing different types
of trace-relationships ~ dependency

- cost of change can be “computed” (estimated)
and assessed (compared to the change’ profit)
[this is **change management**]

5.1. stable vs volatile Rs

!!!: Rs change
does not mean
bad Rs Engineering process

again: Rs change is **normal & unavoidable**

Rs change factors:

- errors, conflicts, inconsistencies:
discovered from analysis / validation on
but not only then: **much later too**

- evolving or growing insight or knowledge
of customer and end-user:
customers & end-users
gradually and steadily
learn / better understand
what they **really** want

- realizability: technical, schedule, cost:
no (good enough) solution found
taking too long
getting too expensive

- shifting priorities of customer:
new (ie changing) business situation
new market
new competitors
new staff
new management
new laws /standards

- change in (technical) system environment:
new technology, such as
platforms, OSs, DBs, network,
applications, services,
languages (programming or modelling)
integration, architecture, ...

- change of (business) organisation:
new organisational
structure, goals, strategy,
processes, style

change is **inevitable** indeed

but some Rs change
more often / easier than others

hence: **stable** Rs & **volatile** Rs

stable Rs: **relatively** fixed

they are about;
- kernel / essence of the system
- heart of problem domain

examples

- system for general practitioners:
patients, deseases, medication

- system for personel:
employee, function, career, salary

- system for stock:
article, price, sales, ordering rule

one then always has:
- the standard structure: notions, concepts, items
- the usual functionality: in line with
the normal way of working
- the usual integration / interaction: in line with
the normal way of working

volatile Rs have the following 4 categories:

- emergent Rs: appearing only on the basis of
the functioning of the system as a whole

- consequential Rs: appearing only on the basis
of the usage of the system

these 2 are rather near to each other, eg:
- presentation:
other possibilities / combinations
- performance:
additional restrictions, to make it work
- unforeseen usage
to be prevented or to be cultivated:
eg: security, authorization,
short-cut, integration

- mutable Rs: depending on organisational and social environment
  - tax, term conditions, risk management, pollution restrictions, process support

- compatibility Rs: depending on
  - equipment:
    - computers, sensors (AD), devices (DA)
  - software:
    - concrete systems, architecture rules
  - technical embedding:
    - machines, processes, materials, products

also these 2 are rather near to each other:
as both depend on environment

coping with Rs change:

anticipation

- predicting, expecting, recognizing **possible** changes
  this is localization of (possible) change

- localization of control / guarding
  wrt such R - each R separately

5.2. Rs identification & storage

pre-requisite of Rs management is:

Rs identification

nevertheless, rather often it is absent

if done at all, normally:
numbering according to structure of RsDoc
: chapter, section, subsection, ...

drawbacks:
(a) only **after** RsDoc has been **approved** the numbering is fixed
(b) Rs classification according to RsDoc's structure **obscures** other relationships between Rs

alternatives:

- dynamic renumbering
  ~ figures etc in a text-editor
  drawback (a) repaired
  drawback (b) still valid

- DB key
  so there is a RsDB
  drawback (a) & (b) both repaired

- symbolic identification
  abbreviation as classification
  drawback (a) repaired
  drawback (b) still valid

more properties - all drawbacks - of storing Rs in RsDoc:
RsDoc is a file of a text-editor, so:

- separate configuration / version management (several authors)

- separate traceability

- electronic coupling with change is absent

- different versions of 1 R implies:
  different versions of RsDoc as a whole

- searching is rather restricted:
  text-editors do not offer much

- navigation between related Rs:
  only in the 1order offered by the RsDoc

- navigation between R and model / design / implementation part is unclear / missing

!!! therefore: RsDB !!

main class is: REQUIREMENT

with attributes:

- identifier: key
- statement / description: txt, figure, video, ...
- date-entered: first date
- date-changed: last change
  alternative: history
- source(s); person(s) and/or circumstances
- rationale: why incorporated;
  : txt, figure, video
- relevance / importance:
  must-have, nice-to-have, ...
- status:
  : proposed, reviewed, accepted, ...
  or: in analysis, under review, ...
- comment: plus whatever

furthermore

rejected Rs should be kept: for future (re)use

with relationships:

- has-dependant: other Rs

- is dependent-on: the inverse of has-dependant

- model-link: towards model / design / code

note:

- multimedia DB, as video, graphics, sound, ...

- over 1000 Rs, then larger type of DB

- multi-site access: teamwork

- CASE tool / SPE (Software Process Env.)

  more support change

  more support during / relation with other SE phases
  design, impl., maint., ...

5.3. Change Management

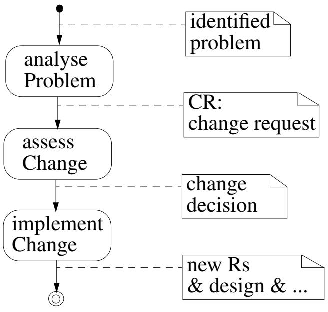
actual process when changing Rs:

- step 1: change request (CR) process
  towards formal CR via a CR form

- step 2: assessment
  impact (based on traceability), costs, time

- step3: the actual change, controlled by
  CR Board / Change Control Board

Change Management process:

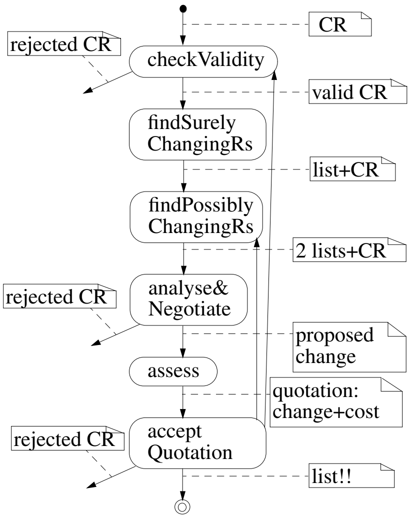


analyseProblem
~ elicit + reuse of Rs formulation

assessChange ~ analyse&Negotiate

implementChange ~ document + validate +
+ remainder of SE lifecycle

more refined assessChange process:



above process has more iteration
as well as other outcomes

note:
different stages of reject / accept:
- easy
- relatively easy
- difficult

note:
rejection handling process is relevant too !!!

in RsDB
- form of CR --> additional Requirement(s)
- status: differentiate between
a stage finished and a next stage started
- Change Control Board
checks / controls process continuation
decides where relevant
groups CRs, analyses, designs. ...

tool support for change management

- electronic CR from
eg to be filled in by various participants:
--> workflow-like

- forms fed into DB

- process support for actual
CR assessment
CR execution

- drawback: process fixed

- research:
customization of process
flexibility
without losing control / guidance

note:
embedded feedback loop pattern is applicable

- 2 concrete Primary Processes
RE process
change management process

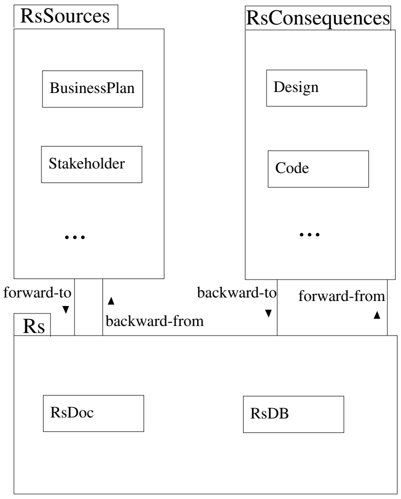
- Management part contains
Change Control Board, relevant for
change management process

- in Environment eg remainder of SE process

- in (M)IS: RsDB and CR-DB

5.4. Traceability

rough data model for RsDB (as class diagram)



about the direction terminology:

- forward: according to waterfall direction

- backward: inverse of forward

- to & from : Rs-centred

drawback:
traceability **among** Rs not covered

to include these to:

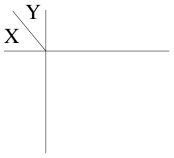
- has-dependants: ~ forward-to

- is-dependent-on: ~ backward-from

traceability tables

comparable to (in)consistency matrix

such a table is a matrix
either of Rs
or of Rs & design (etc) fragments



X is-dependent-on Y

often: 2 lists instead of matrix

one list:

per R: those that are is-dependent-on it

the other list: is inverse list of the former

so per R: those that are dependants of it traceability policy:

keeping traceability info up-to-date
not only the first time, but later too

- which traceability:
  between which kind of Docs; directions

- matrix or list(s)

- when to collect and by whom
  process description

- what if emergency / urgency
  not / summary
  later but when
  how / what

during change management:
also discriminate between
normal, urgency, emergency, other

policy should be

- realistic

- described in a traceability manual
  - it is a task
  - responsibility
  - “gaps” appear easier
  - increase of awareness & trust