Rationality

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47

Information Systems
March 27, 2008

Rationality (from Wikipedia)

... a decision or situation is often called rational if it is in some sense optimal

... individuals or organizations are often called rational if they tend to act somehow optimally in pursuit of their goals

In this concept of "rationality", the individual's goals or motives are taken for granted and not made subject to criticism, ethical or otherwise

Thus rationality simply refers to the success of goal attainment, whatever those goals may be

Sometimes rationality implies having complete knowledge about all the details of a given situation

It might be said that because the goals are not important in definition of rationality, it really only demands logical consistency in choice making

Limits to Rationality

A system’s users seldom know exactly what they want and cannot articulate all they know.

Even if we could state all requirements, there are many details that we can only discover once we are well into implementation.

Even if we knew all these details, as humans, we can master only so much complexity.

Even if we could master all this complexity, external forces lead to changes in requirements, some of which may invalidate earlier decisions.

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Why a Rational Design Process?

the usual process of designing software is irrational - and serious problems result from this state of affairs

we would like to derive our programs from a statement of requirements in the same sense that theorems are derived from axioms in published mathematical proofs

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Why this is an idealization

The requirements are incomplete and inconsistent

Many facts that become known later, so there are backtracking loops in the design process

Humans cannot manage large amounts of detail

System specs change for external reasons during the process

Using humans implies human errors

We use our favorite ideas, not ideas rationally derived from the particular requirements

It is often sensible to make suboptimal design decisions - especially for reasons of cost

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Why is an idealization useful?

We can follow an idealized rational process as closely as possible, even if we cannot follow it exactly in reality

Designers need guidance

An ideal model is better than an ad hoc process

A rational process provides a basis for a standard method

Provides a model for control and review

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Rational Design Process Elements

At each stage of the process, we need to know:

- What product we should work on next
- What criteria the product should satisfy
- Who should do the work
- What information the workers should use

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Illustration: Requirements Spec.

Every statement should be valid for all acceptable software systems produced

The document should be complete (any system satisfying the stated requirements must be acceptable)

Where information is incomplete the doc should say so

The document should be organized as a reference document - not as an introductory narrative

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Faking the ideal process

The design process should produce the documents

- in order if possible
- with temporary gaps (noted) where information is missing

even if the actual process is nonlinear

... and then fill in the gaps

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Faking the ideal process

By comparison: **Mathematical proofs** are an artifact of the end of a lot of work

not a story of the process of discovery

but can be read as a rational, linear argument or exposition of the proof

Present the **software design** the same way vso that the design document can be read as a

- rational ,
- linear,
- structured

exposition of the design

Parnas and Clements, “A Rational Design Process: How and Why to Fake It”

Poppendieck

In 1986, Parnas [18] equated a ‘rational’ design process to a ‘waterfall’ lifecycle, and suggests that even though such a process is impossible to follow, perhaps we should ‘fake it’.

The software engineering community has gone to great effort to put the waterfall lifecycle behind it, while continuing to acknowledge that this may be the ideal lifecycle, but it is simply impossible to follow.

Perhaps it is time to acknowledge that a software engineering process which demands a detailed scope definition to be fixed at the beginning of a project is not an ideal process, but is instead a “legacy process”.

Poppendieck II

It is time to admit that it is not 'ideal' or even 'rational' to start with a detailed requirements definition at the beginning of a software development process; the requirements specification should be developed as on-going part of the project.

If we want a good system, we must allocate a significant portion of the total time for the really important activities of the project, namely requirements definition and architectural design.

Scope management and work decomposition are simply not important during this fairly large phase of the project.

In fact, if they are emphasized, they will tend to impede the important work that needs to be done to lay the groundwork for an excellent system.

Poppendieck III

The iterative approach is not a new or unique concept. Most software project lifecycles employ some form of iteration.

The problem is, many people still consider the waterfall lifecycle to be an 'ideal', if unattainable goal.

It's time to recognize that the software development process is fundamentally iterative, and stop trying to 'fake it'.

An iteration or two of the system should be developed to help define the architecture and address key risks.

This initial phase of the project should be expected to take perhaps 40% of the allotted timeframe.