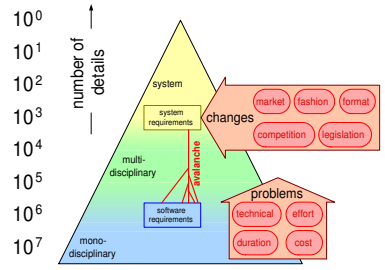
What is a Good Requirement Specification?

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

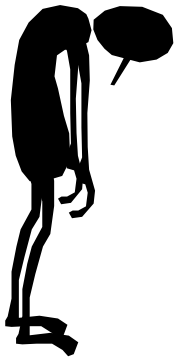
Requirements play a driving role during product creation. The requirements are captured in a requirements specification. How can we assess the requirements specification? What are the criteria for a good specification?
We discuss these aspects by positioning the requirements specification in the broader context of customers, market, product creation and product life-cycle. We zoom in to the software requirements specification, to discuss the criteria for this mono-disciplinary specification.

September 9, 2018
status: concept
version: 0.2



Did you ever program a VCR?

A ☐



depressed

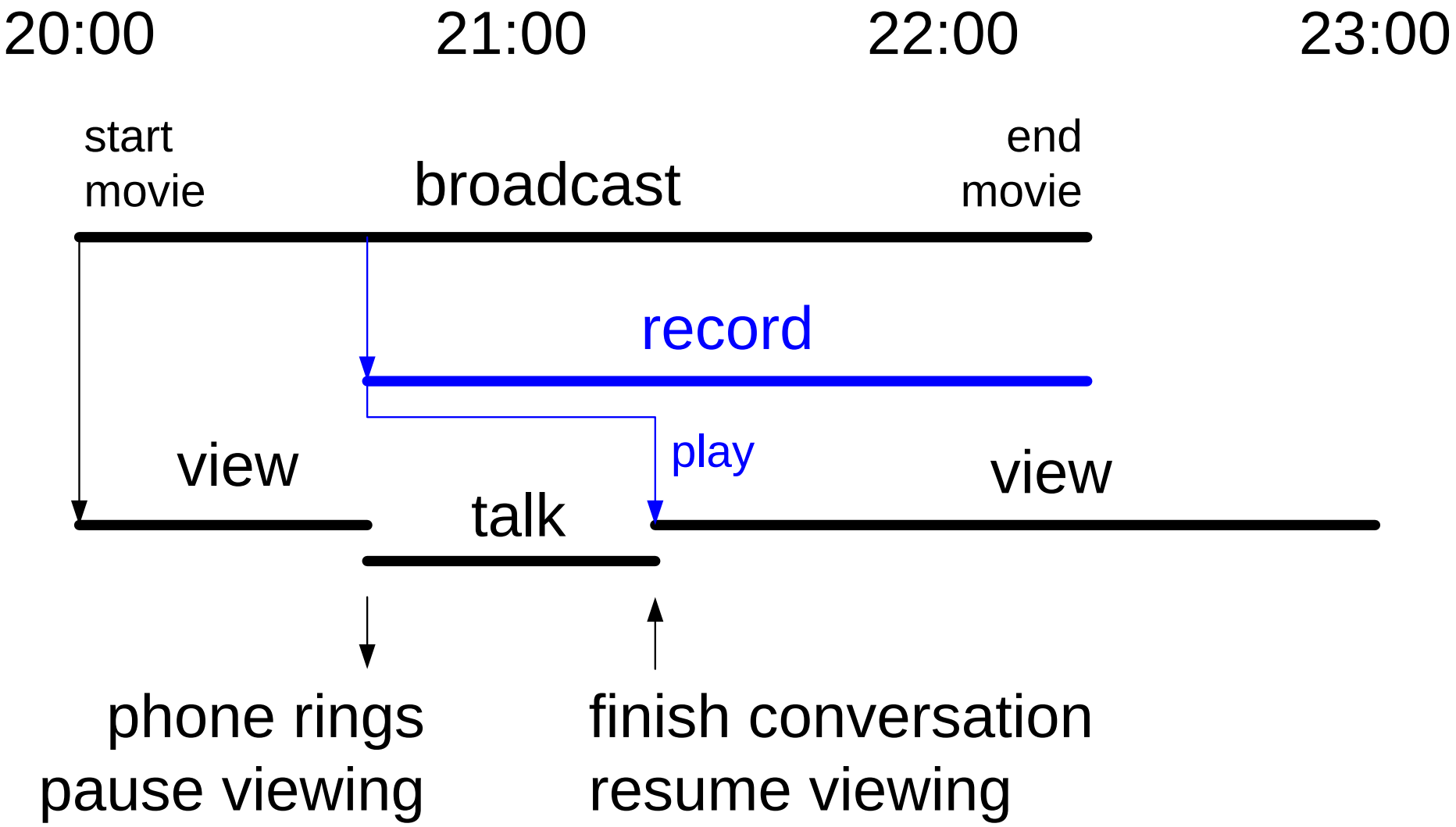
B ☐

desparate

C ☐

hysteric

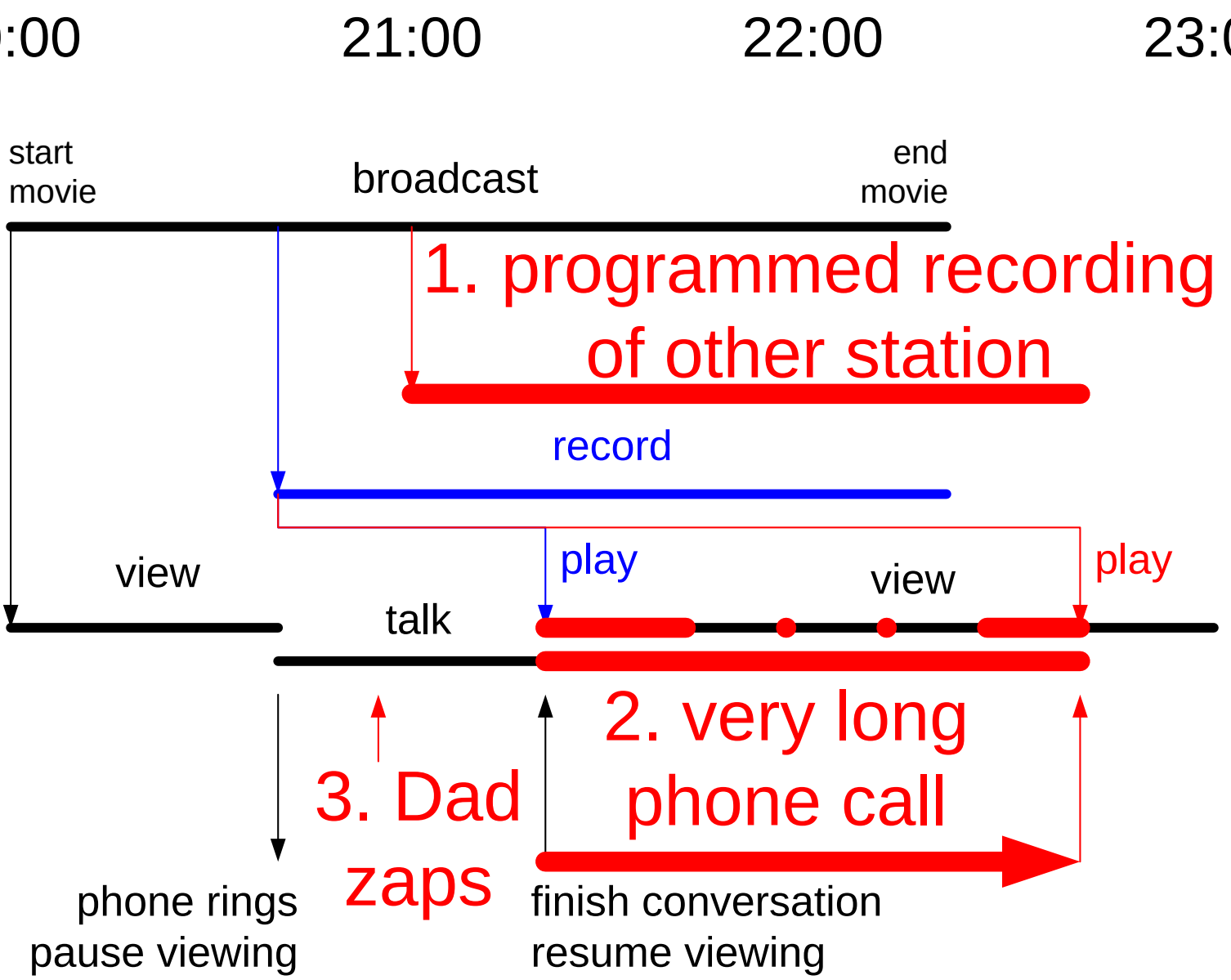
Example Time Shift recording



Construction limits intrude in User Experience

- number of tuners
- number of simultaneous streams (recording and playing)
- amount of available storage
- management strategy of storage space

What if?


20:00
21:00
22:00
23:00
start movie
broadcast
end movie
1. programmed recording of other station
record
view
play
view
play
talk
2. very long phone call
3. Dad zaps
phone rings
pause viewing
finish conversation
resume viewing

OOTI workshop 2001: “Requirements Engineering”

Visual Basic Prototype:

enables "experiencing"

- ☐ play
- ☐ pause
- ☐ record
- ☐ EPG

Requirements specification

Many tables, mostly addressing details

- 2.1.1 Real-time data requirements
- 2.1.2 Implementation detail
- 2.1.3 Non-real time data requirements

1.1 Software Requirements

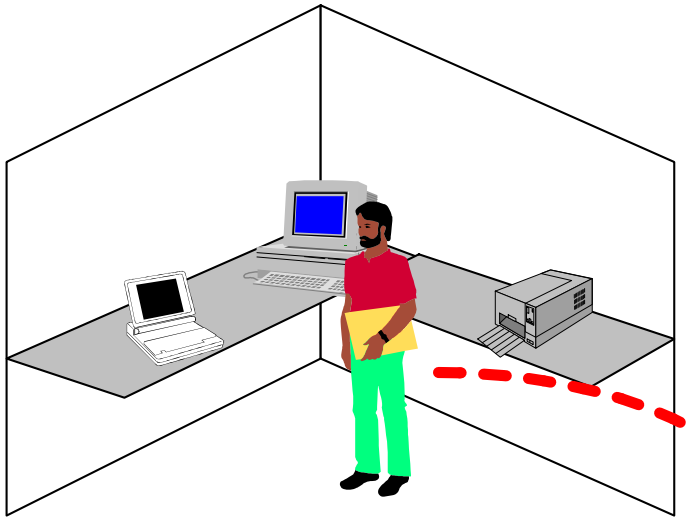
1.1.1	Real-time data requirements	*1.1.1.1 Access to the non-real-time data must be done in such a way that it does not interfere with the real-time data* *1.1.1.2 There must be no disruptions in output of video signal during the operation of VCR* 1.1.1.3 *Responsiveness for non real-time data is less then 150ms (the time for writing a block on HDD) for 2KB of non-video data*	
1.1.2	Implementation detail	*1.1.2.1 Management of HDD content must only be possible through the TOC in order to prevent unauthorized access to content of HDD* *1.1.2.2 Visual feedback is provided to the user via On-Screen Display* 1.1.2.3 *User input is provided via the RC*	
1.1.3	Non-real time data requirements	*1.1.3.1 User must be able to pause and unpause a title, played from HDD, while (s)he is watching it* *1.1.3.2 User can jump forward and backward in a title, from HDD, during watching of this title* *1.1.3.3 Names of titles should be derived from the information from the EPG (name of the program to be recorded, time and date of registration)*	

Key Success Factor: Feedback

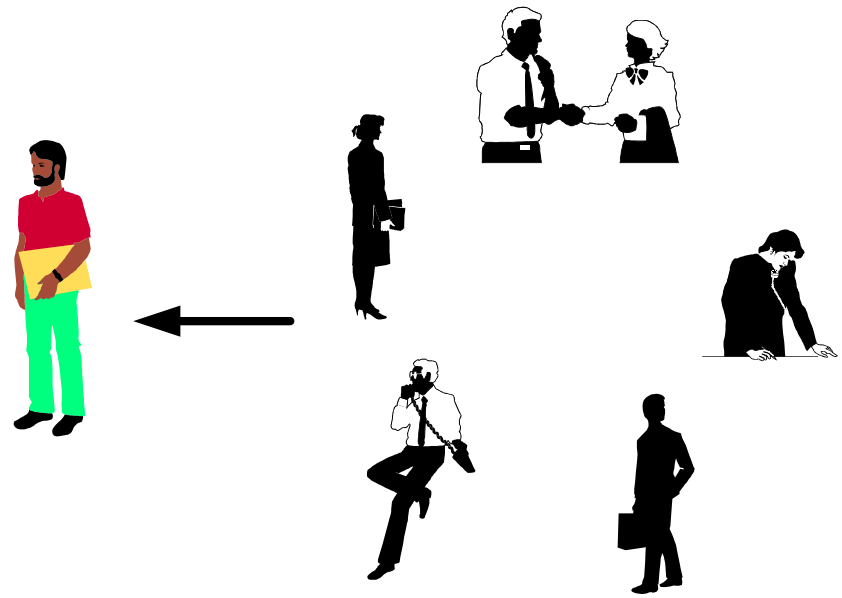
Obtain feedback from real users:

- Observe
- (Dare to) Listen

- Experiment
- Use short development cycles



Don't stay in the development lab

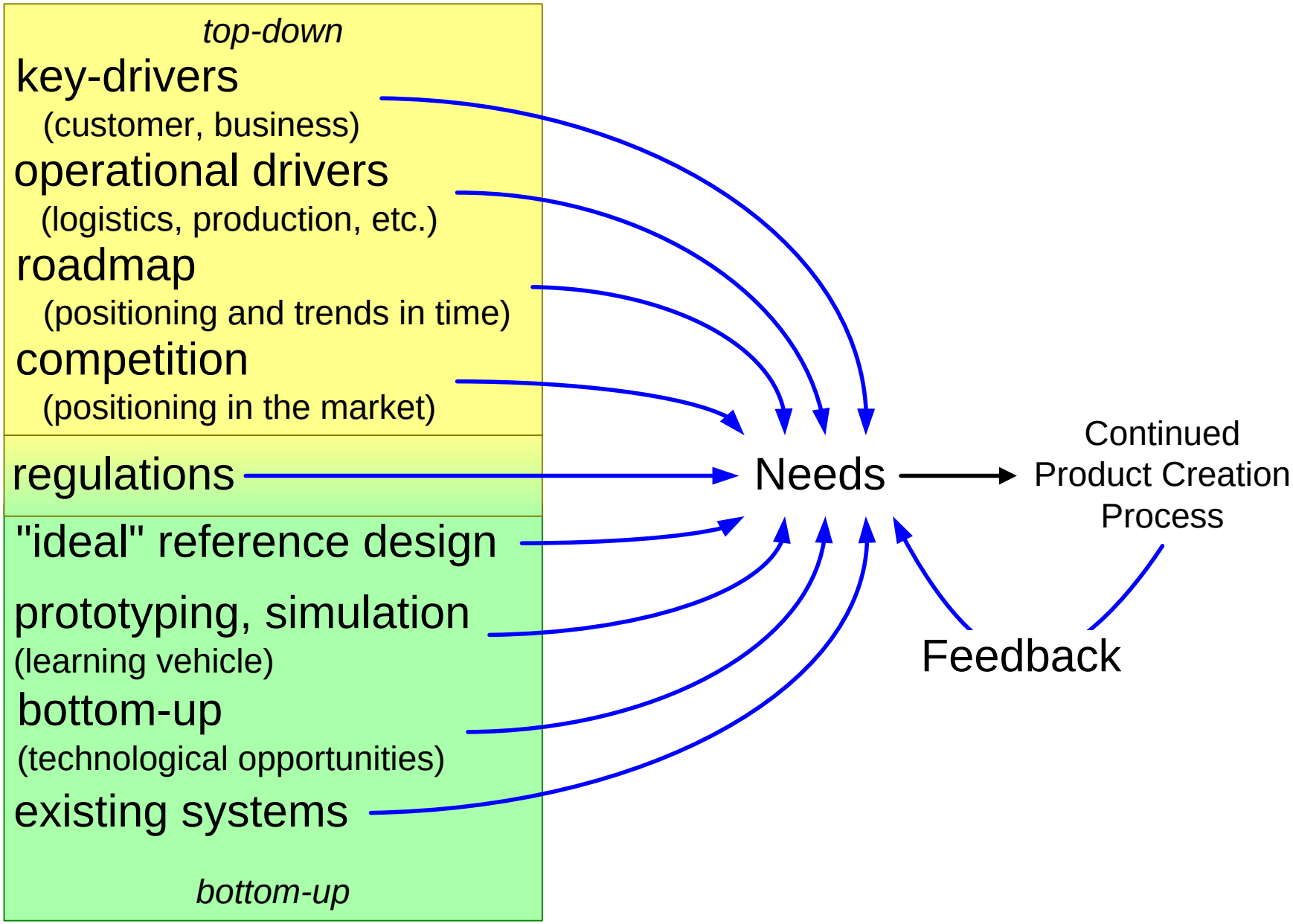


Criteria for a Good Requirements Specification

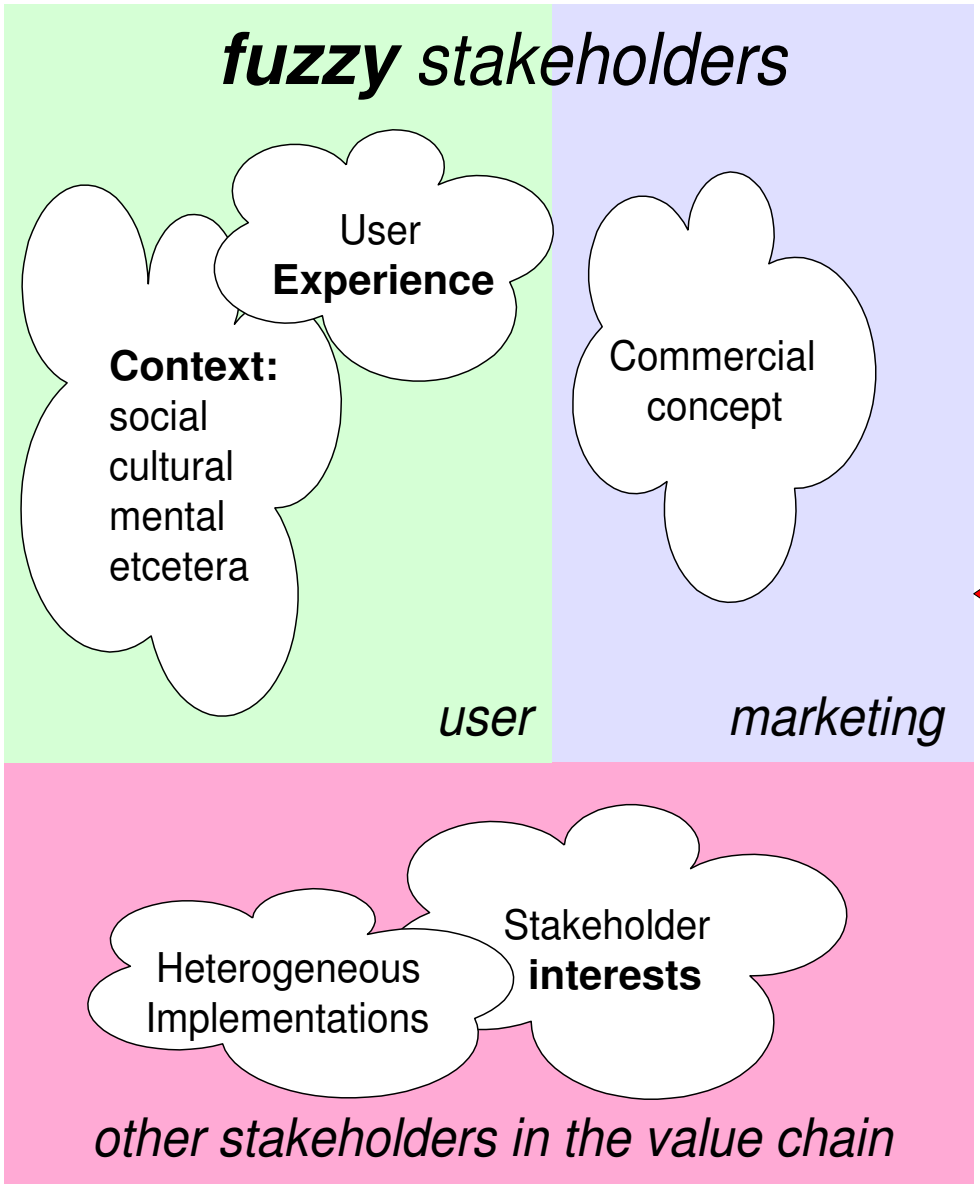
A good requirements specification:

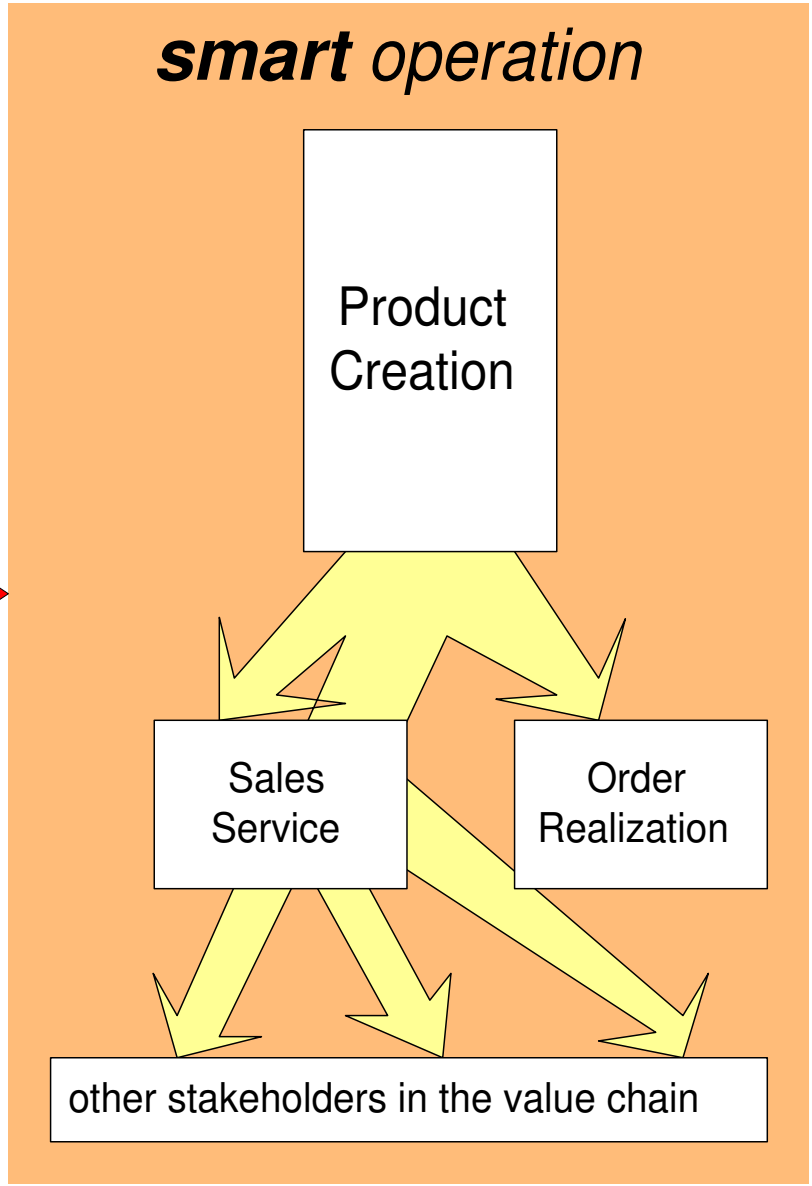
- reflects the real needs of all stakeholders
  - especially customers
  - implicit, latent
  - sense
  - simplicity
  - i.e. more than hard factors!
- describes a feasible product
  - sales
  - manufacturing
  - logistics
  - service
- answers most critical design questions
  - non-linear choices
  - discrete options
  - (e.g. hard disk vs flash memory)
- is useful for human product creators
  - SMART, but also
  - understandable
  - accessible

Multiple Viewpoints to Understand Needs and Feasibility



How SMART can requirements be described?





Requirements must be SMART and Usable

fuzzy *stakeholders*

- Accessible
- Understandable
- Low threshold

smart *operation*

- Specific
- Unambiguous
- Verifiable
- Quantifiable
- Measurable
- Complete
- Traceable

System or Software?

When SW engineers demand "requirements",

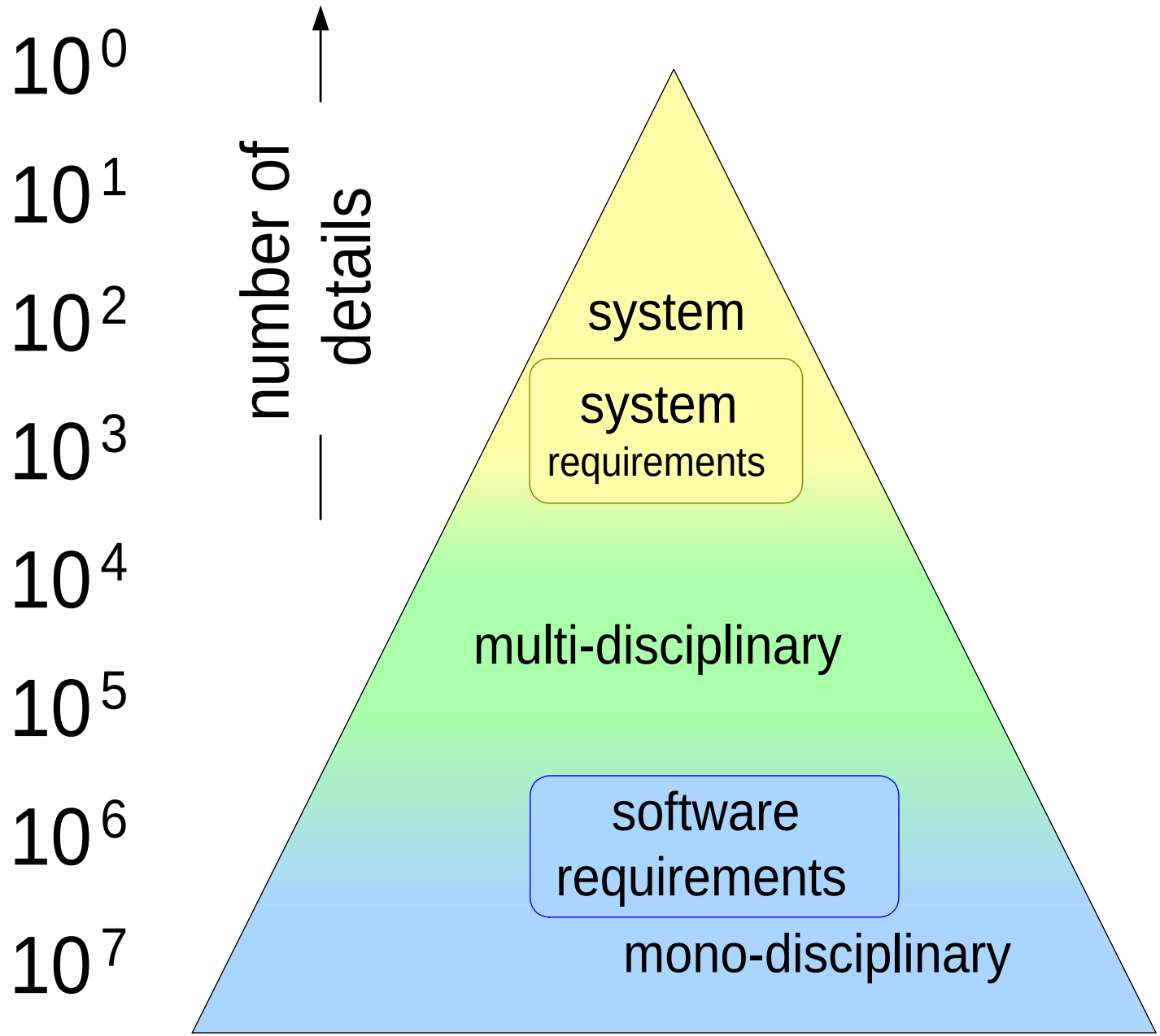
then they expect *frozen* inputs

to be used for

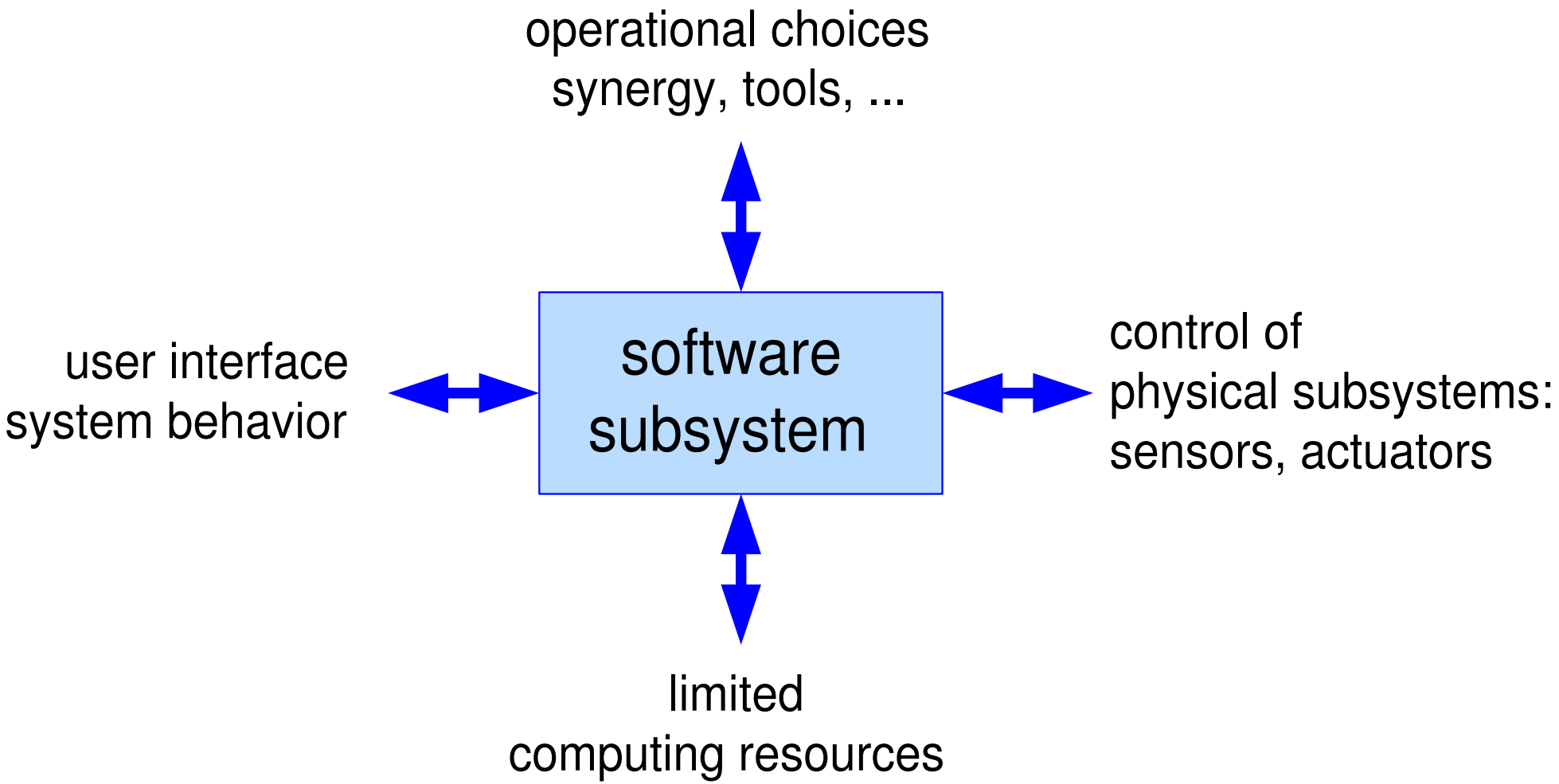
the design, implementation and validation

of the software

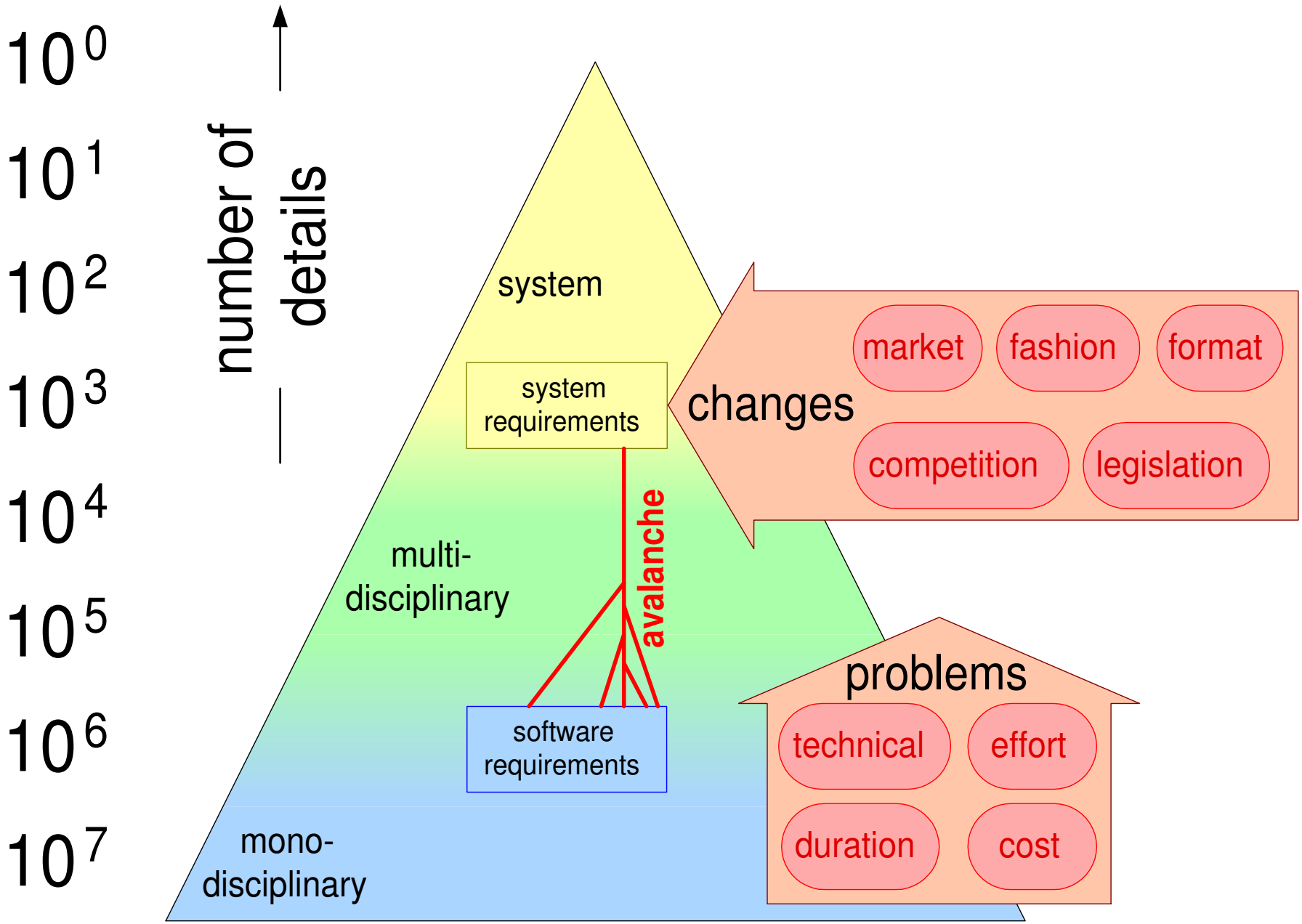
System vs Software Requirements



Why is the Software Requirement Specification so Large?



And why is it never up-to-date?



Conclusions and Recommendations

Never wait for the software requirements specification to be complete

1) it is never complete

2) it is never up-to-date

3) the product will be too late.

Be creative to cope with uncertainty and dynamics

for instance, use prototype as specification "WYSIWYG"

use incremental development strategies (XP, EVO, ...)

focus on most important and critical issues