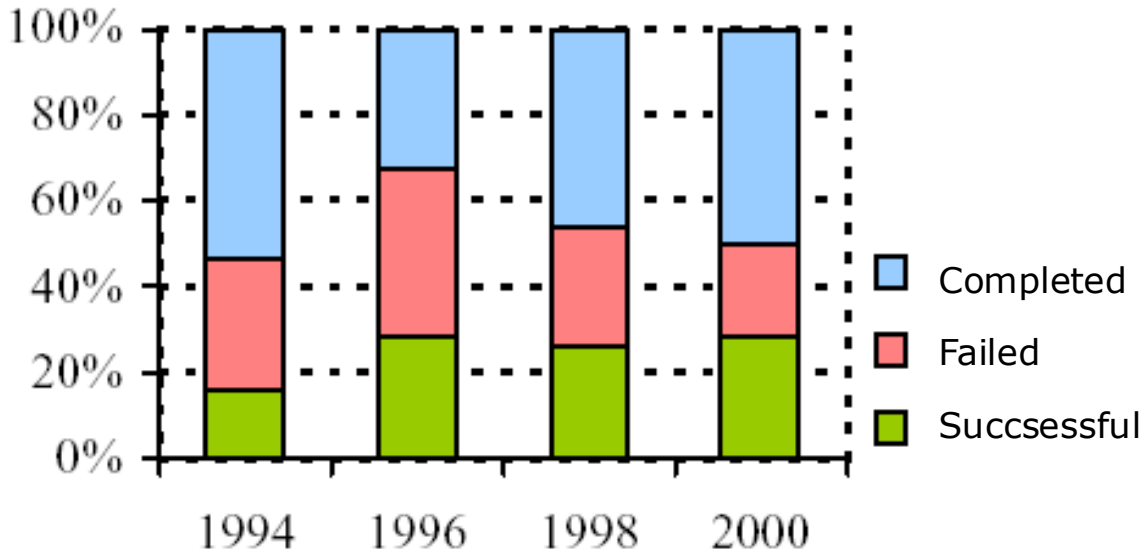

COURSE 1
Software Projects

Success of Software Projects

- Study by *The Standish Group International*
  - 30,000 U.S. software projects analyzed between 1994 and 2000



Success of Software Projects (cont.)

- *CHAOS* Research
  - 50,000 projects in 2004 (58% *U.S.*, 27% *Europe*, 15% *other regions*)

Year	Successful	Completed	Failed
2004	29%	53%	18%
2005	28%	49%	23%
2006	35%	46%	19%
….			
2009	32%	44%	24%

Project Success

- ***Success evaluation***:
  - did the project meet its objectives regarding *schedule*, *costs* and *quality*?
  - does the project satisfy *requirements* & *needs* of the client?
  - can the project results lead to *new projects*?
  - after the project is completed, is the organization able to *continue its operations*?

- ***Ensuring success***:
  - clear objectives,
  - competent staff,
  - management support,
  - adequate resources,
  - communication,
  - monitoring & control

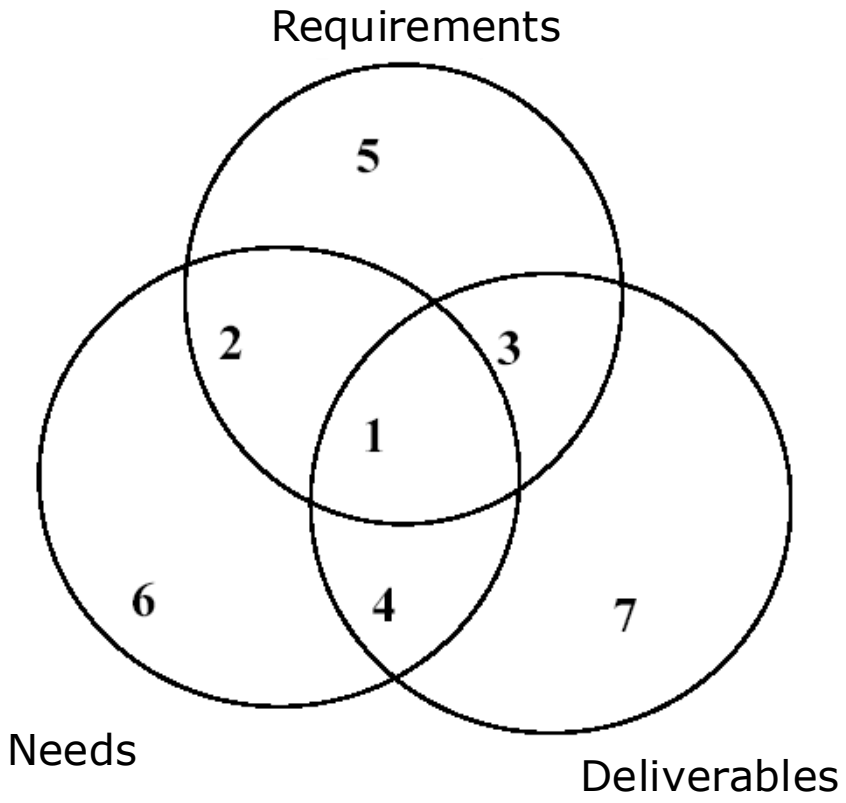
Product / Service Quality

- Product quality = *its ability to satisfy certain implicit or stated needs*

- Quality perspectives:
  - *product perspective*: quality of the product content;
  - *user perspective*: satisfying user needs;
  - *producer perspective*: conformance to specifications (which may or may not match the user perspective);
  - *value perspective*: delivering the result at an acceptable price;
  - *transcendent perspective*: quality cannot be defined precisely; it refers to "intrinsic excellence" (as in works of art).

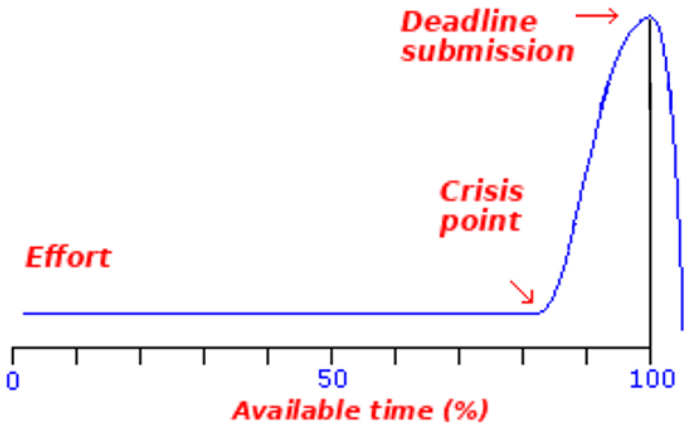
Product / Service Quality (cont.)

1. Conforming quality
2. Product defects
3. Unnecessary quality
4. Quality surplus
5. Claimed quality
6. Unmet needs
7. Quality waste

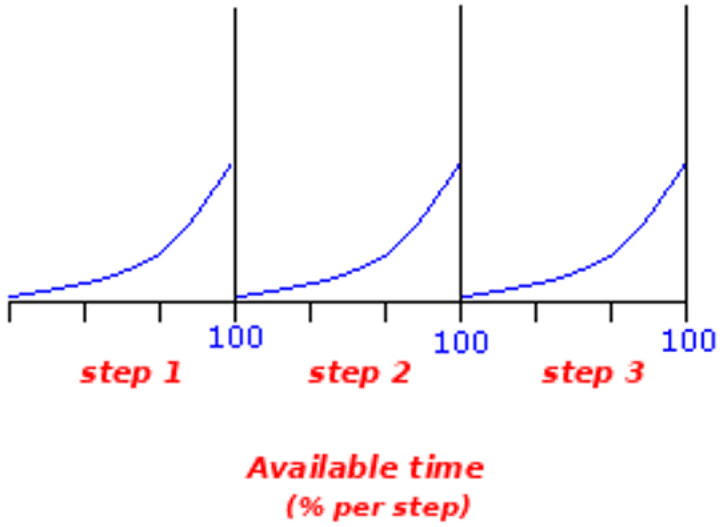


"Small releases"

- Exponential increase in effort toward completion



- Distribution of effort (hours/day) and energy in the iterative and incremental approach



Not measurable, difficult to estimate

- significant differences in estimates by different people, regardless of experience
  - no standard classification
  - consequence: changes are difficult to manage
  - *estimate* +20%
- difficult to monitor / control progress
  - especially during analysis and design

Not measurable, difficult to estimate (cont.)

```
#define         LOWER   0
#define         UPPER   300
#define         STEP    20
main()
{
  int fahr;
  for (fahr=LOWER; fahr<=UPPER;
fahr=fahr+STEP)
   printf("%4d %6.1 f\n", fahr,
(5.0/9.0*(fahr-32)))
}
```

No. of statements	No. of experts
one statement	1
two statements	6
three statements	5
four statements	1
five statements	4
six statements	11
seven statements	4
eight statements	1
nine statements	11

M. Norris, P. Rigby, M. Payne - *The Healthy Software Project: a guide to successful development*, John Wiley & Sons, Chichester,1993

Software: *invisible, intangible*

- produced as texts of different types:
  - design documents,
  - source code,
  - user and operations manuals
- there is nothing concrete to show the client during the requirements analysis phase
  - option: prototype
- initial requirements seem easy to modify

High Complexity

- Software product lifecycle phases:
  - *Functional specification* – natural-language document
  - *Analysis* – analysis model (diagram + annotations)
  - *Design* – design model
  - *Development* – source code in programming language(s)
  - *Compilation/Linking* – executable model
- the results of one phase are transferred to the next phase

$\Rightarrow$ high vulnerability to human error.

Correctness Verification, Testing, Quality

- **impossible** to test all branches
- reworking test scenarios when changes occur
- there are no mechanisms for reliable / precise measurement of an application's quality
- "In software development projects the so-called *quality-control procedures* sometimes have more to do with limiting defects than with guaranteeing the quality of the final product." *Norris et al. (1993)*

Dynamics

- the appeal of technological novelty
  - *stability* vs *stagnation*
- staff turnover
  - reassessment of *tasks*
  - tendency to reject other people's code
- frequent change requests at various stages of the lifecycle

Hasty Decisions in Extreme Situations

- ***adding resources to a delayed project does not eliminate the delay; it increases it***
  - team members' available effort decreases by an amount equal to the effort spent communicating with the new member

… etc

- specifications are too long, dense and detailed, so users fail to identify the main ideas;
- specifications are more a wish list than a prioritized list of features;
- solutions are found that solve one problem but introduce new ones;
- unidentified sub-optimized functionality