Swiss Federal
Institute of Technology
Zürich

Work and Organizational Unit
Prof.Dr. Eberhard Ulich
Nelkenstr. 11, CH-8092 Zürich

Usability Engineering

Matthias Rauterberg

1996

strategic problems...

23% pressure of time

11% cost exceedings

organizational problems...

23% requirements analysis

17% project complexity

15% changes of requirements

8% project management

4% communication between project members

technical problems...

20% software engineering environments

9% hardware constraints

other problems...

12% qualification of project members

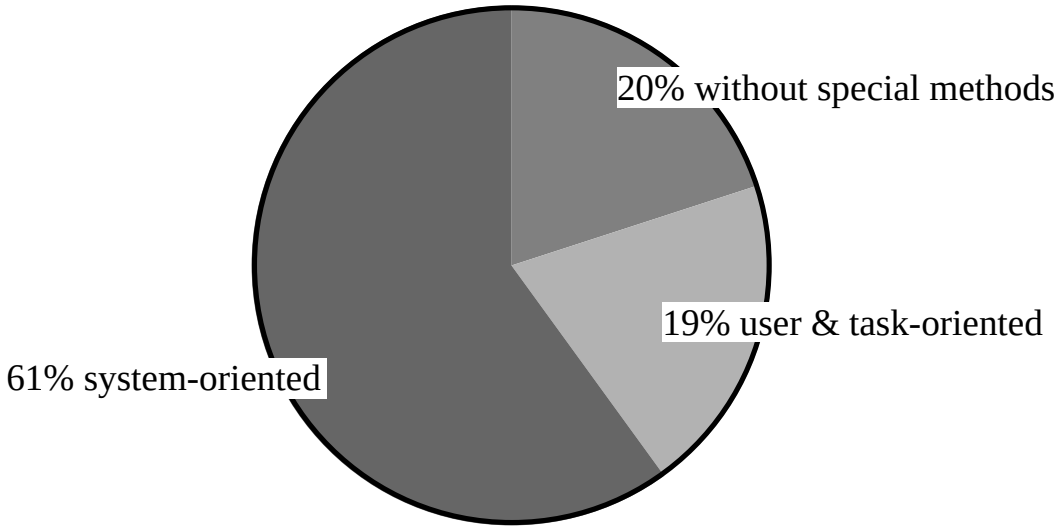
11% continuity of project members

8% software quality

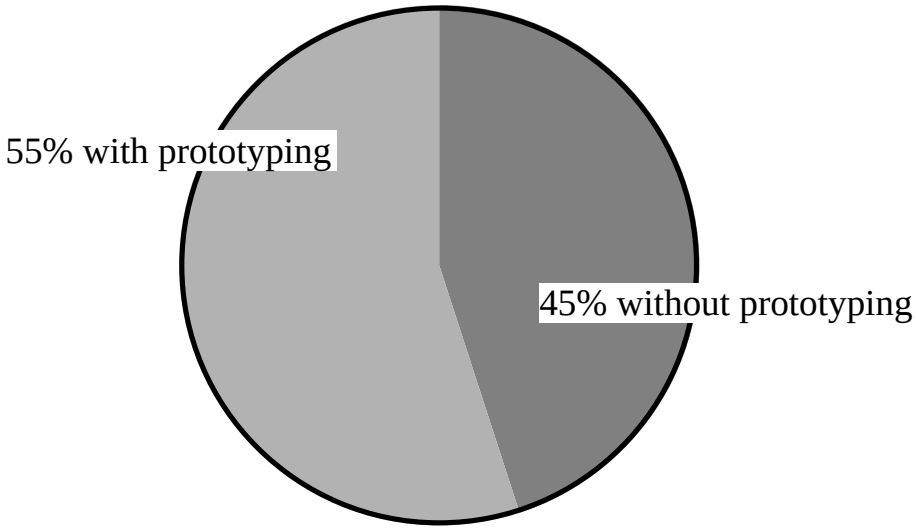
5% external influences

[STROHM & SPINAS,1991, analyzed projects N=75]

methods for requirements analysis

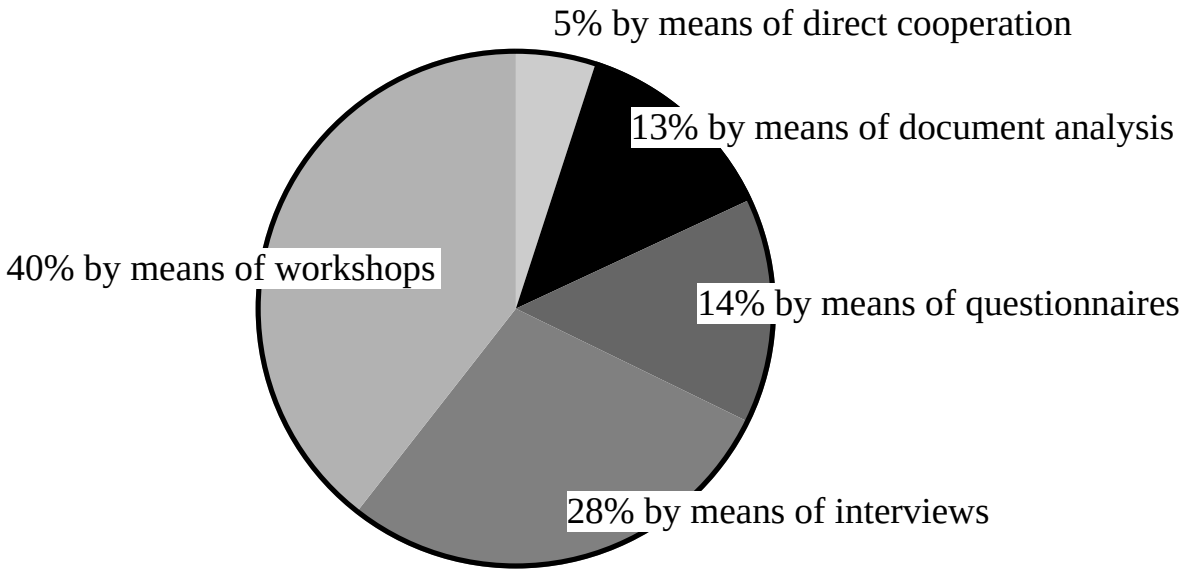


usage of prototyping-methods



[STROHM 1990, analyzed projects N=74]

Contribution of different specification techniques to the final data model



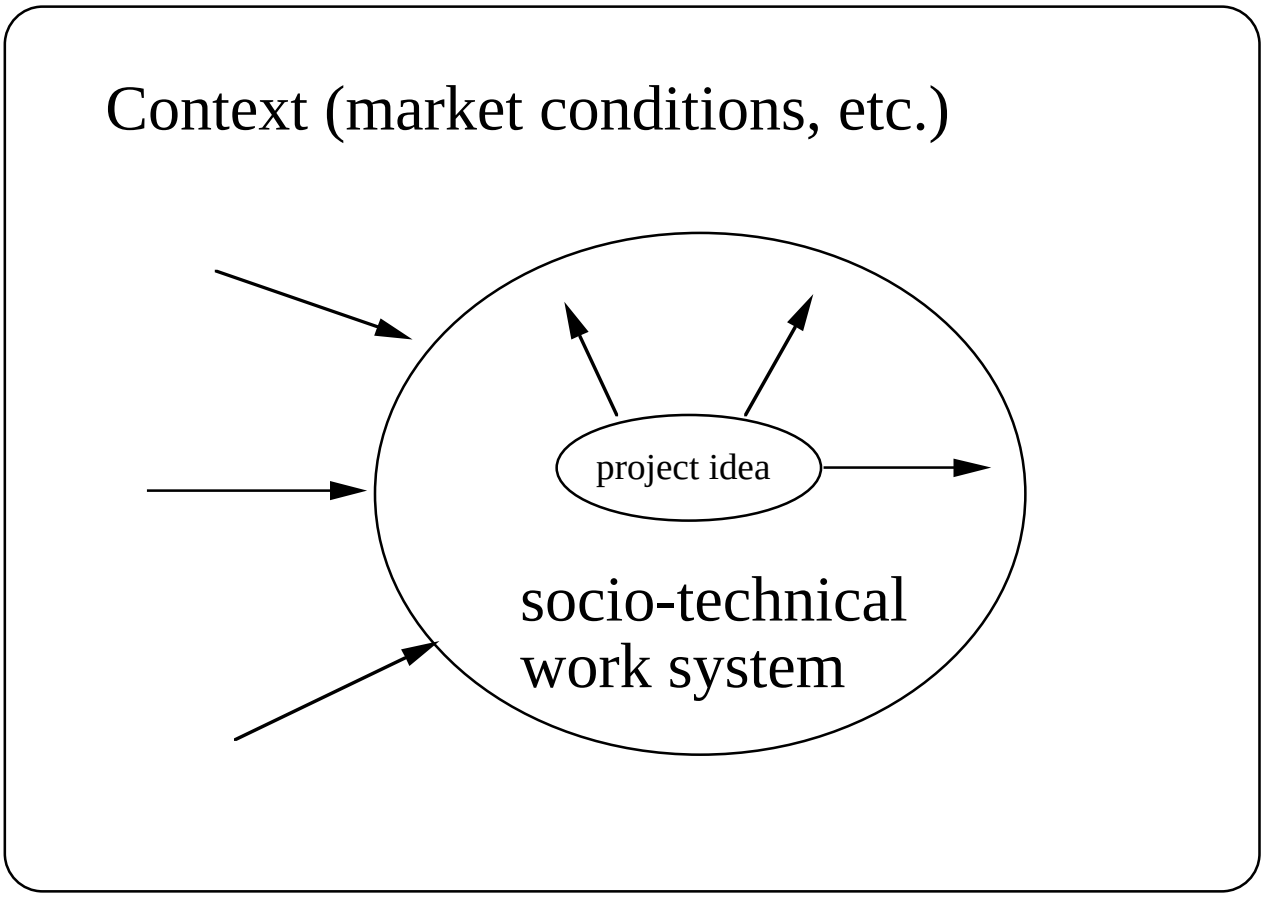
[KIRSCH 1991, 5 methods, 3 project groups with each N=6]

the problem of changing requirements...

two reasons, why requirements can change...

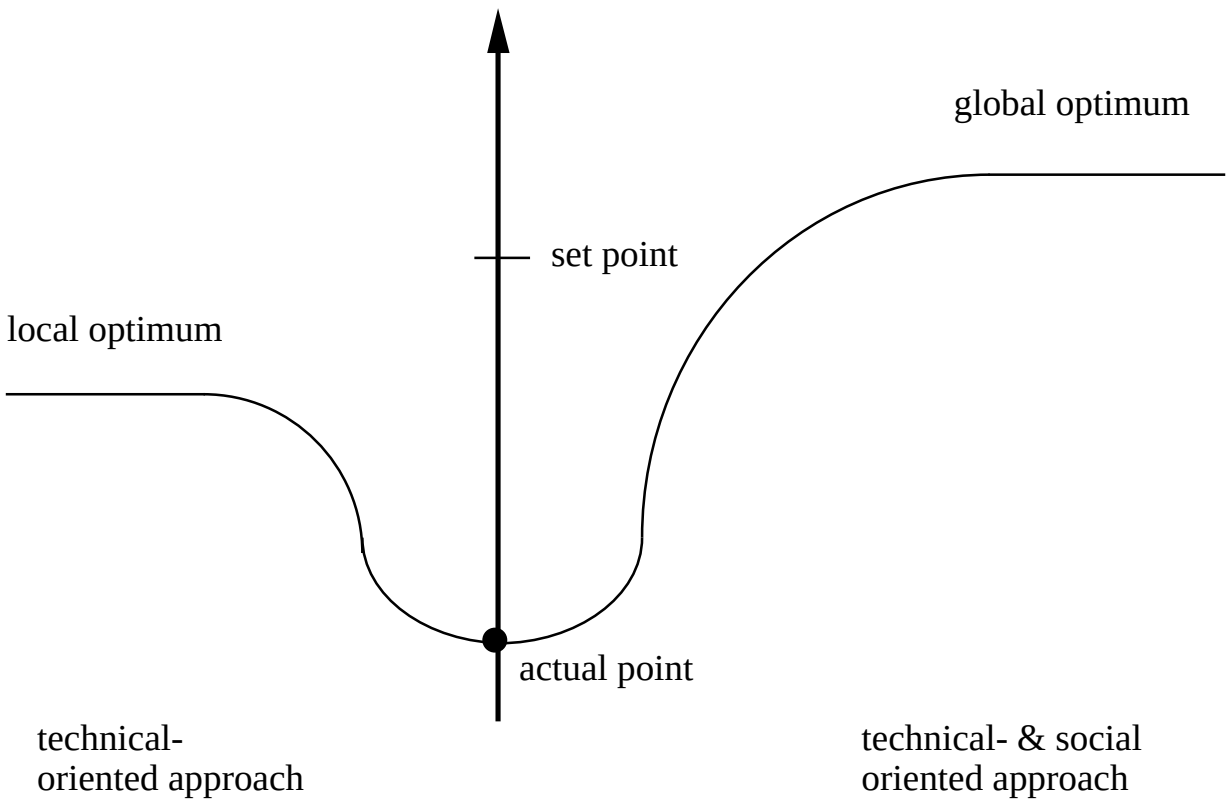
1. changing of the context

2. crystallization of the "final" project idea

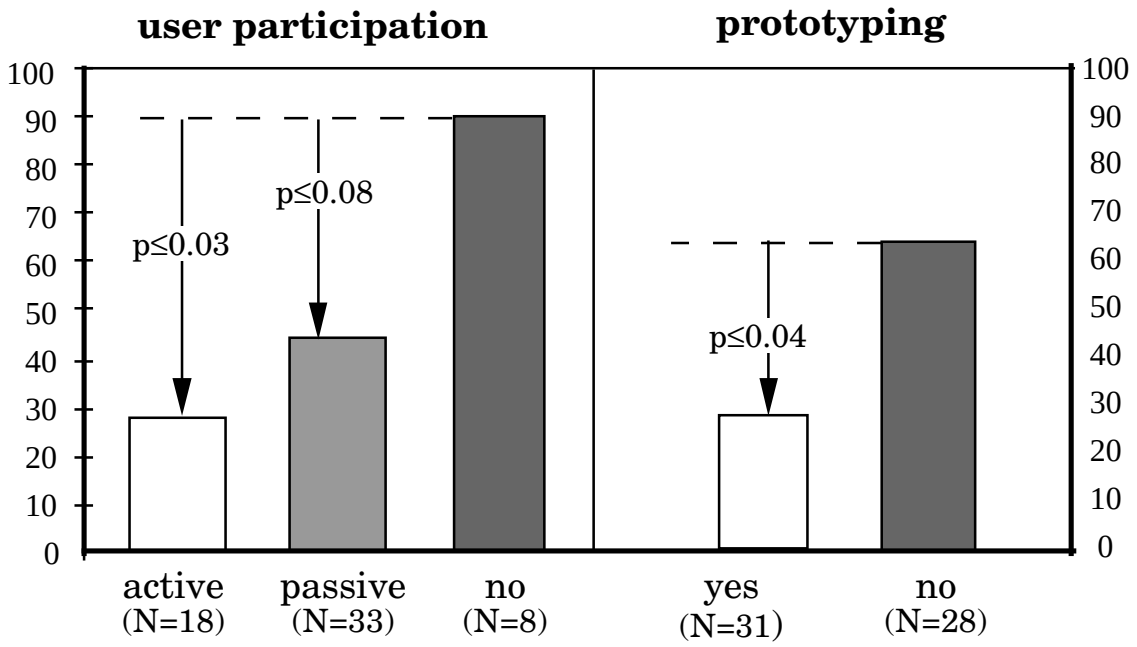


the optimization problem

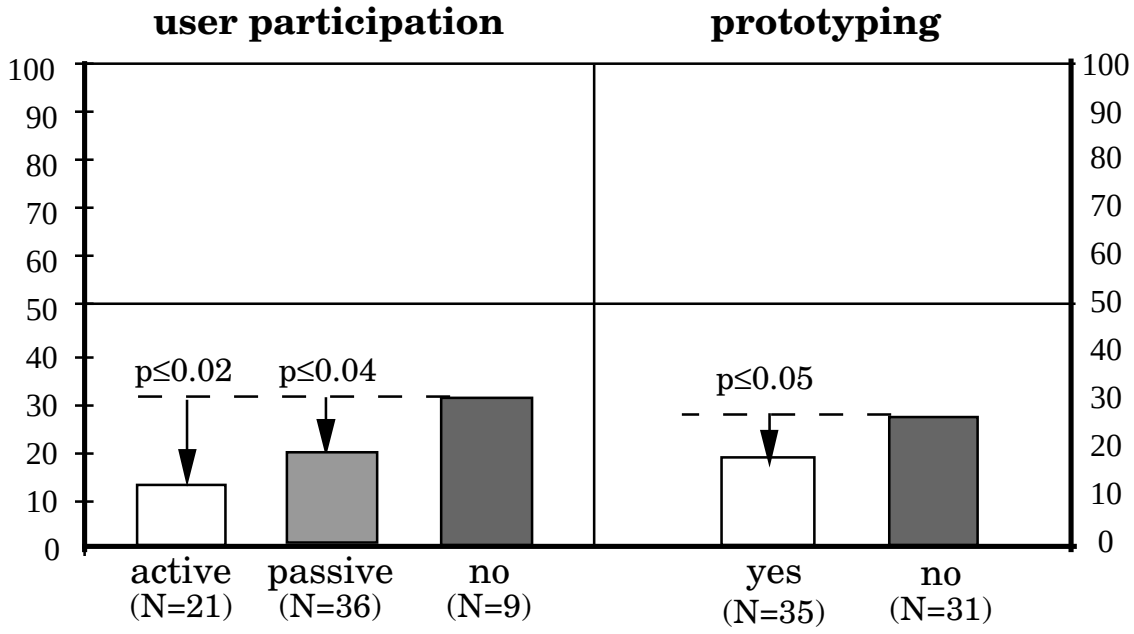
the optimization of a socio-technical worksystem



% cost-exceedings



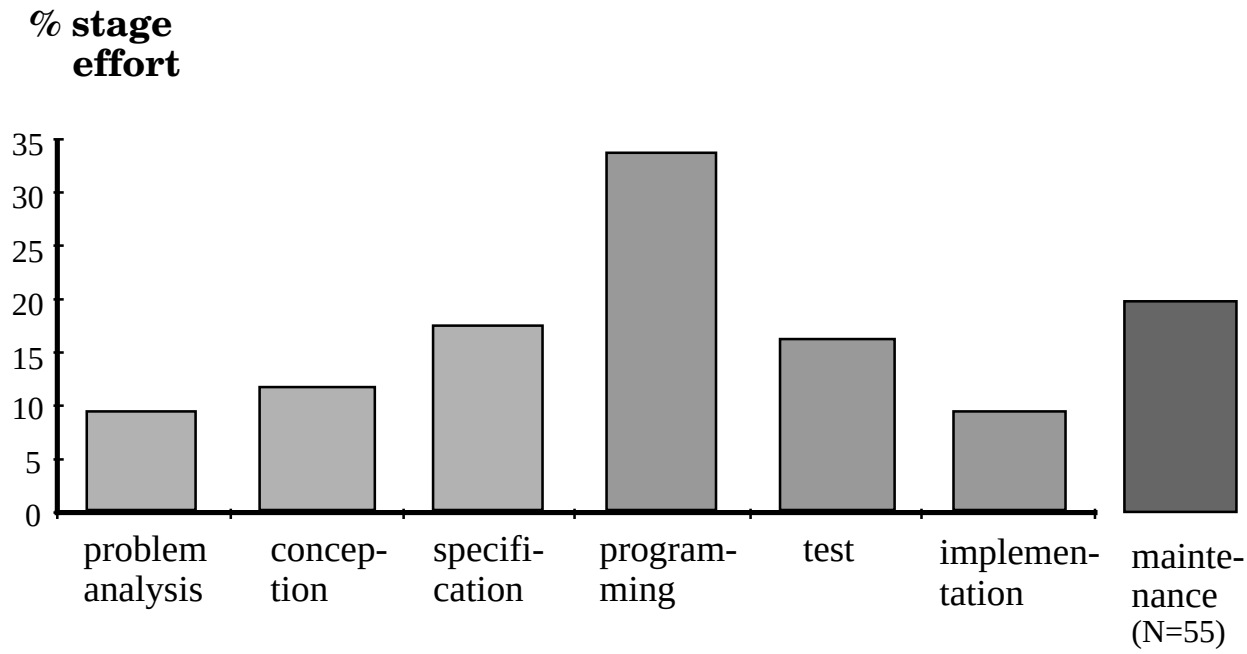
% time-exceedings



[RAUTERBERG & STROHM 1992]

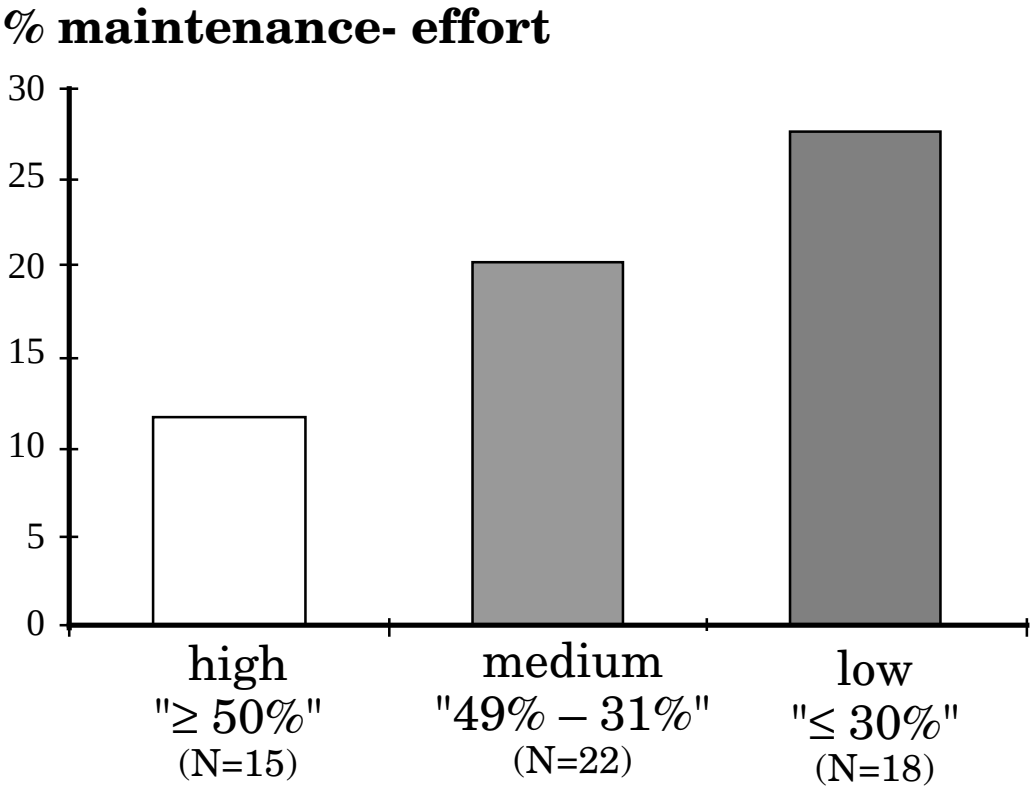
mean stage effort

of 79 software development projects



[STROHM & ULICH 1992]

correlation between effort in the early stages and effort in the maintenance phase

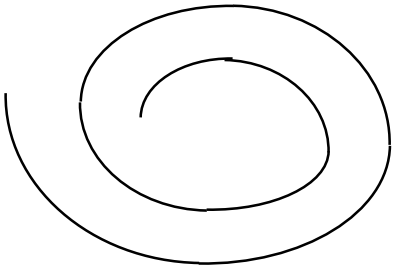


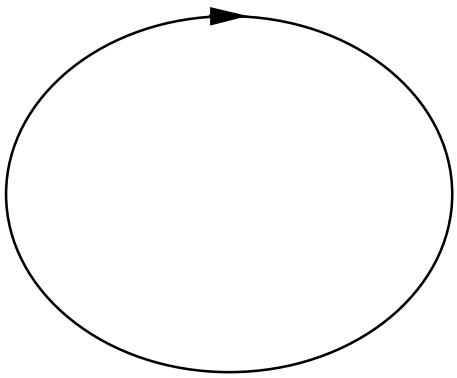
cumulative effort of stage =

"problem analysis"
+ "conception"
+ "specification"

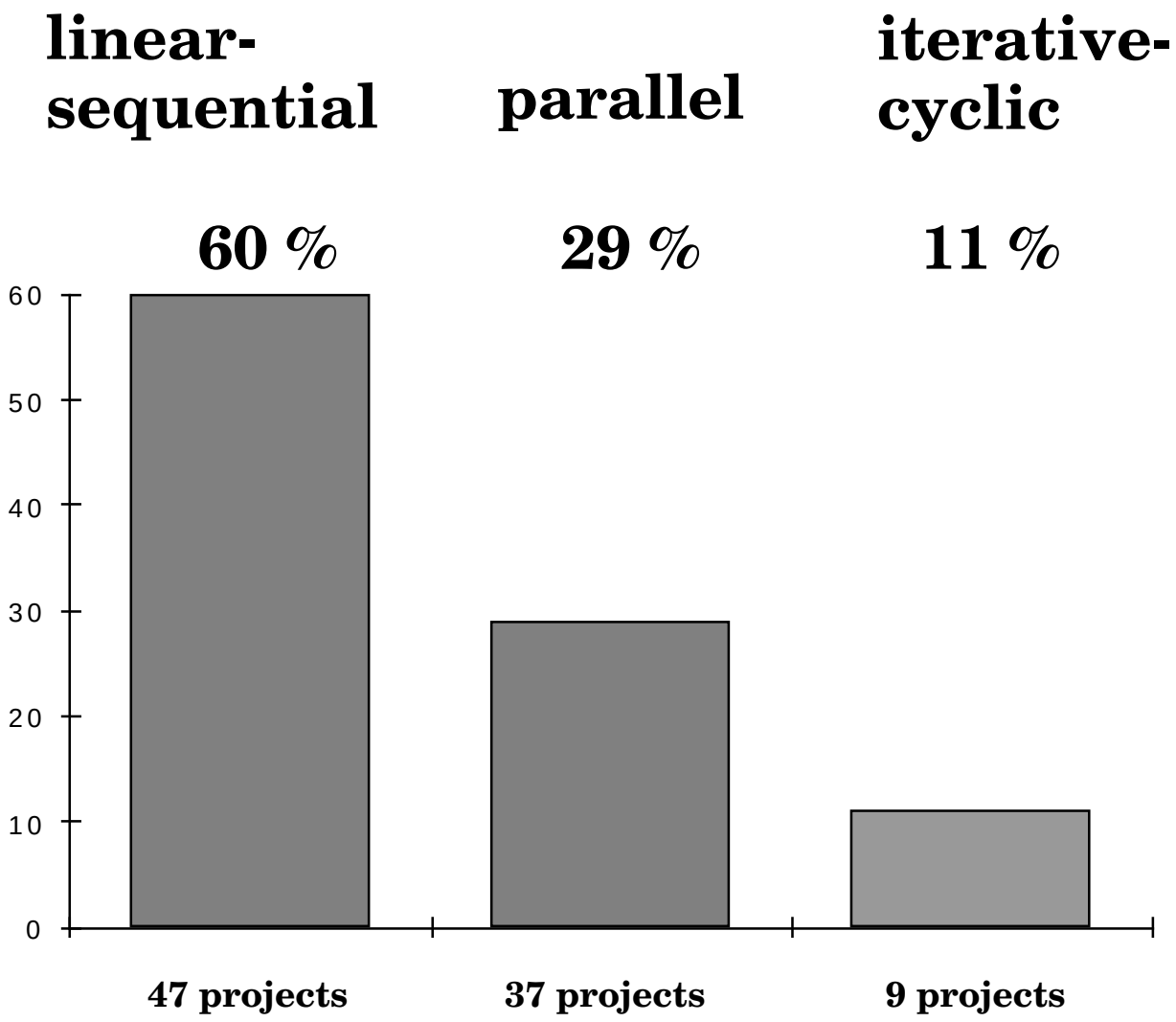
[RAUTERBERG & STROHM 1992]

how can we overcome the obstacles....



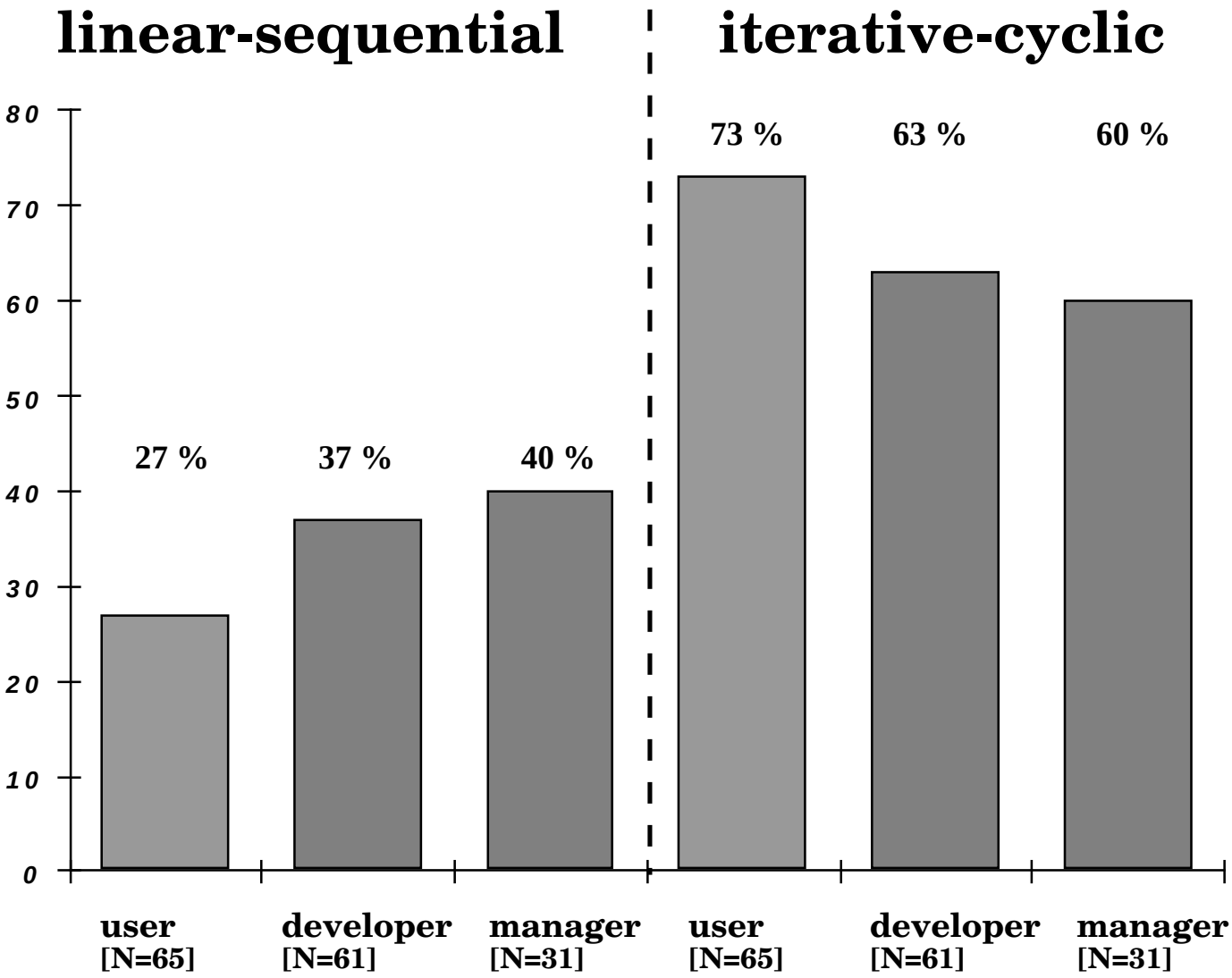


type of the process model (actual in use)



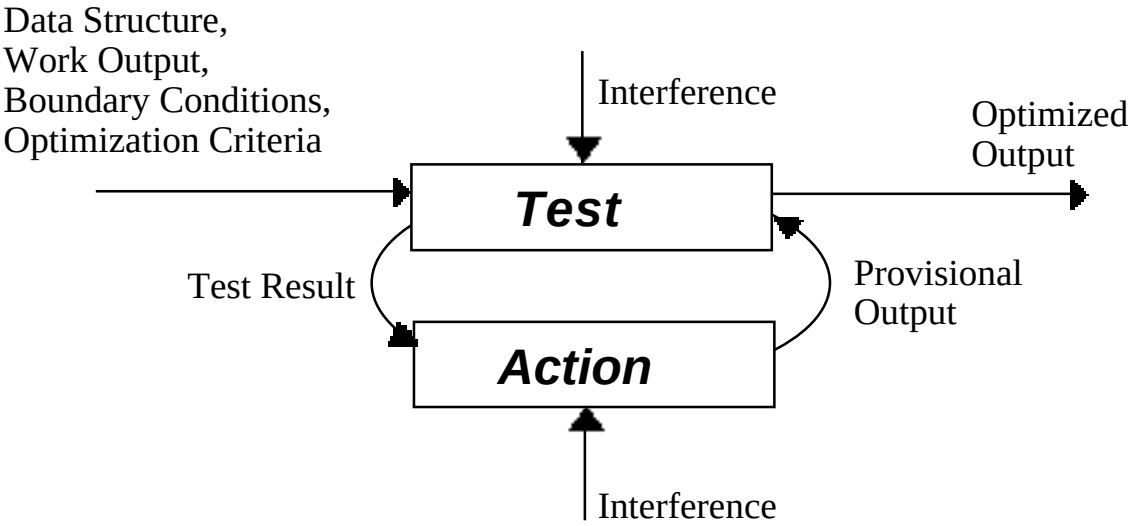
[STROHM 1990, analyzed projects N=79]

type of the process model (ideal procedure)



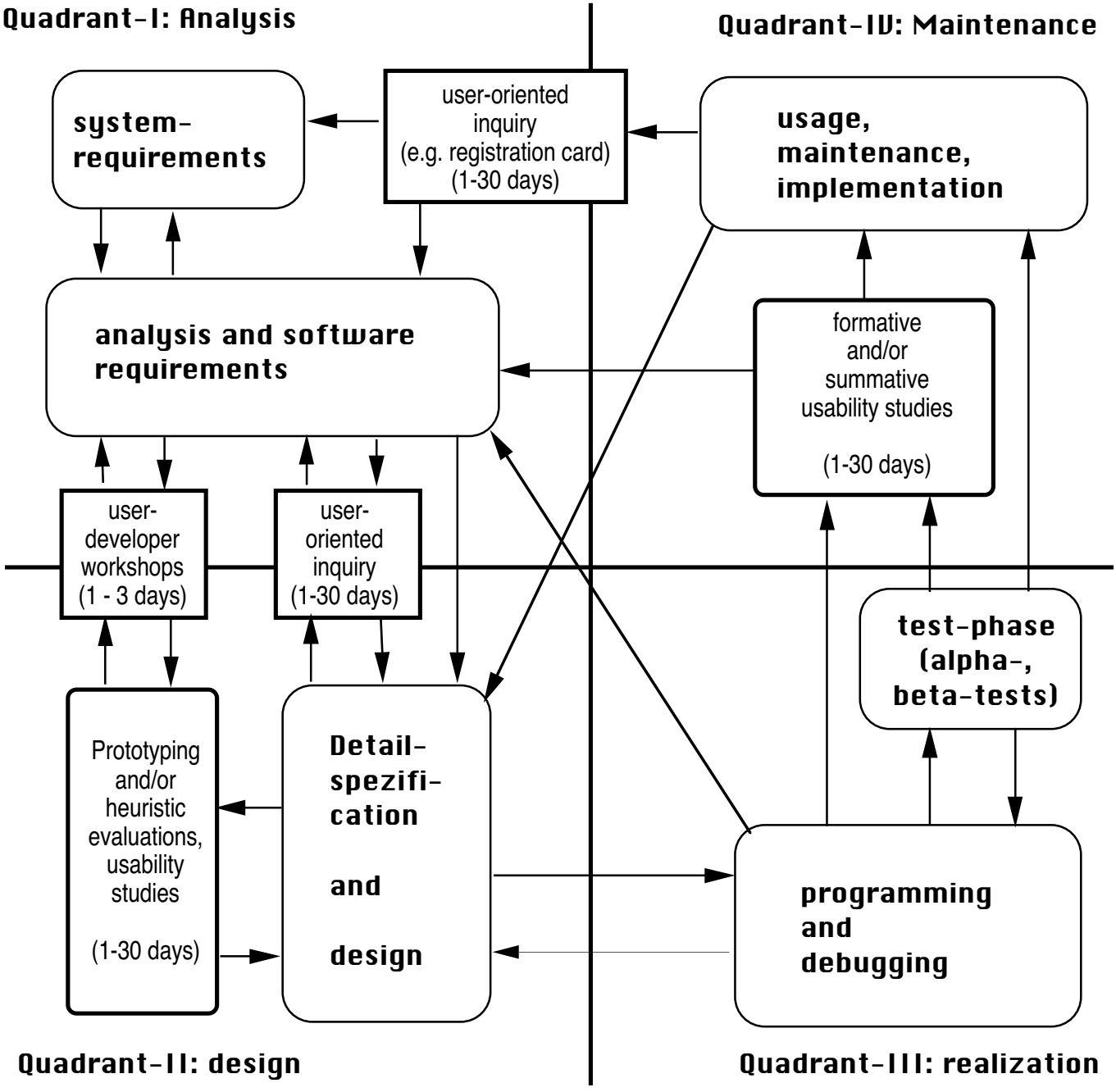
[SPINAS & WAEBER, 1991, analyzed companies N=10]

the optimization cycle



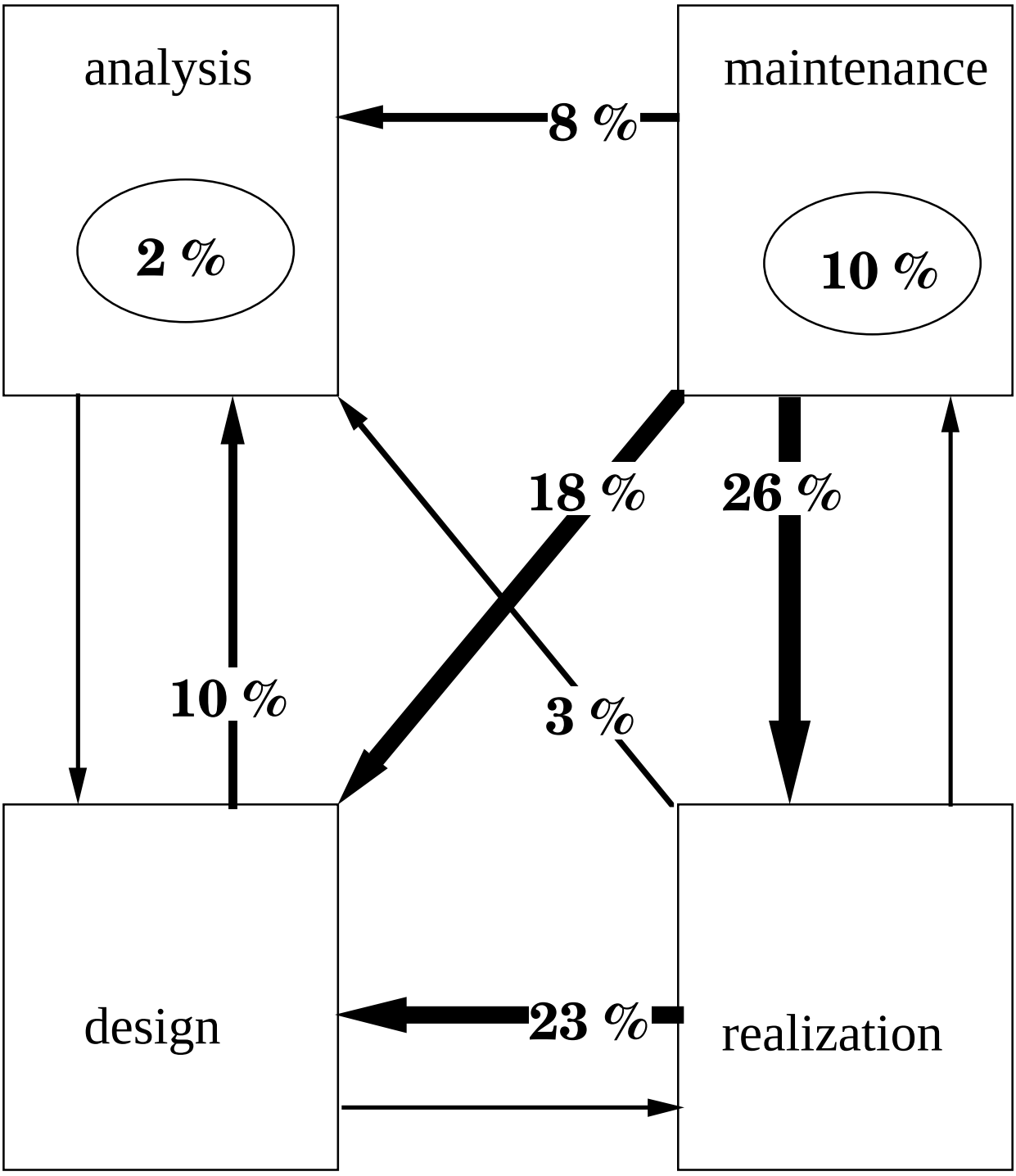
The Four Quadrant-Model

[BOSS-project, Rauterberg 1991]



Number of phase steps back

(IPAS-project, Hesse 1993)



sample:

39 reported explicit loops
29 software projects
156 answers regarding phase progress

methods for user participation

discussion / workshop /fokus group

- verbal
- metaplan
- flip-charts etc.

simulation

- sketches
- mock-ups
- semi-formalistic
- task oriented tests

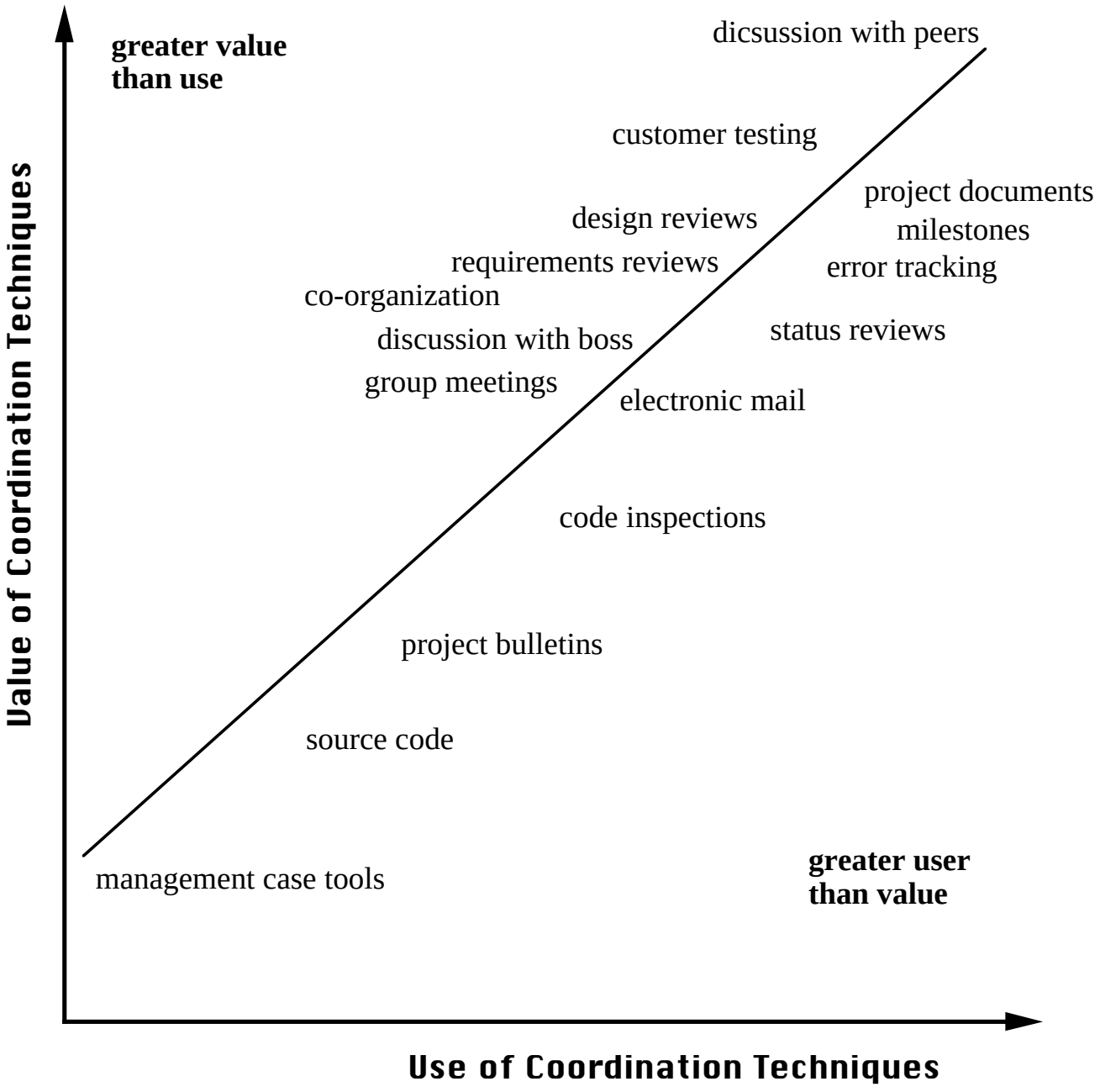
prototyping

- horizontal
- vertical
- formative evaluation

versioning

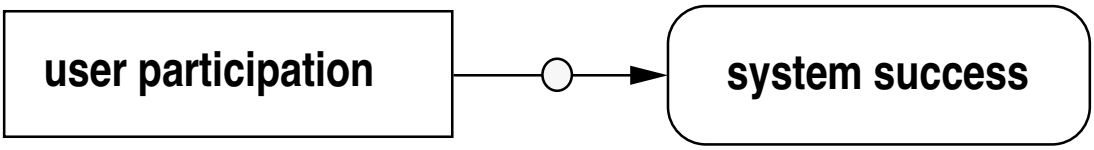
- summative evaluation

comparing the use and value of coordination techniques



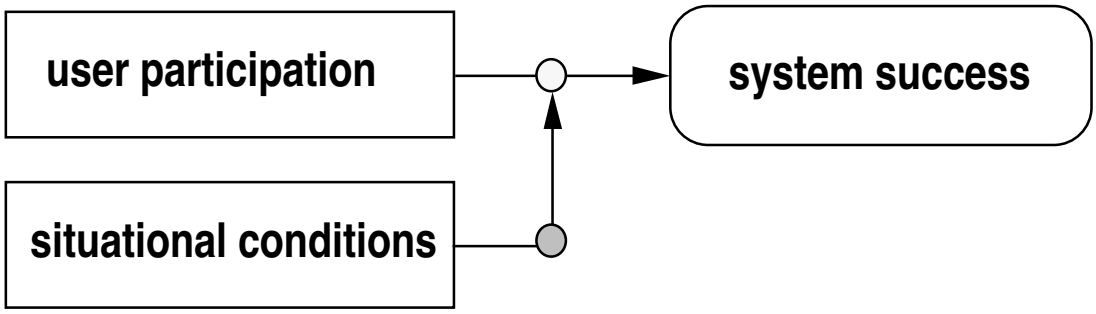
[KRAUT & STREETER, (1992), projects N=65 with 563 software developers]

Conceptual Model-I



○ **empirical results:**
4 studies "*support*", 2 studies "*mix*", 2 studies "*not supported*"

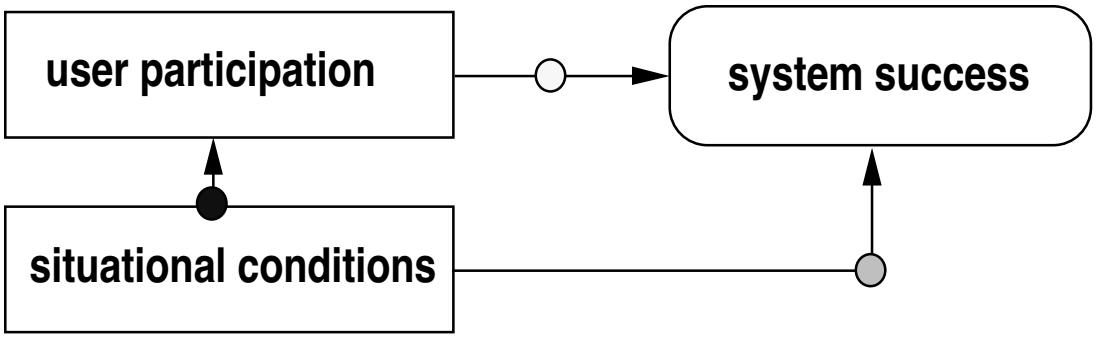
Conceptual Model-II



○ **empirical results:**
3 studies "*support*", 0 studies "*mix*", 1 studies "*not supported*"

● **empirical results:**
3 studies "*support*", 1 studies "*mix*", 0 studies "*not supported*"

Conceptual Model-III



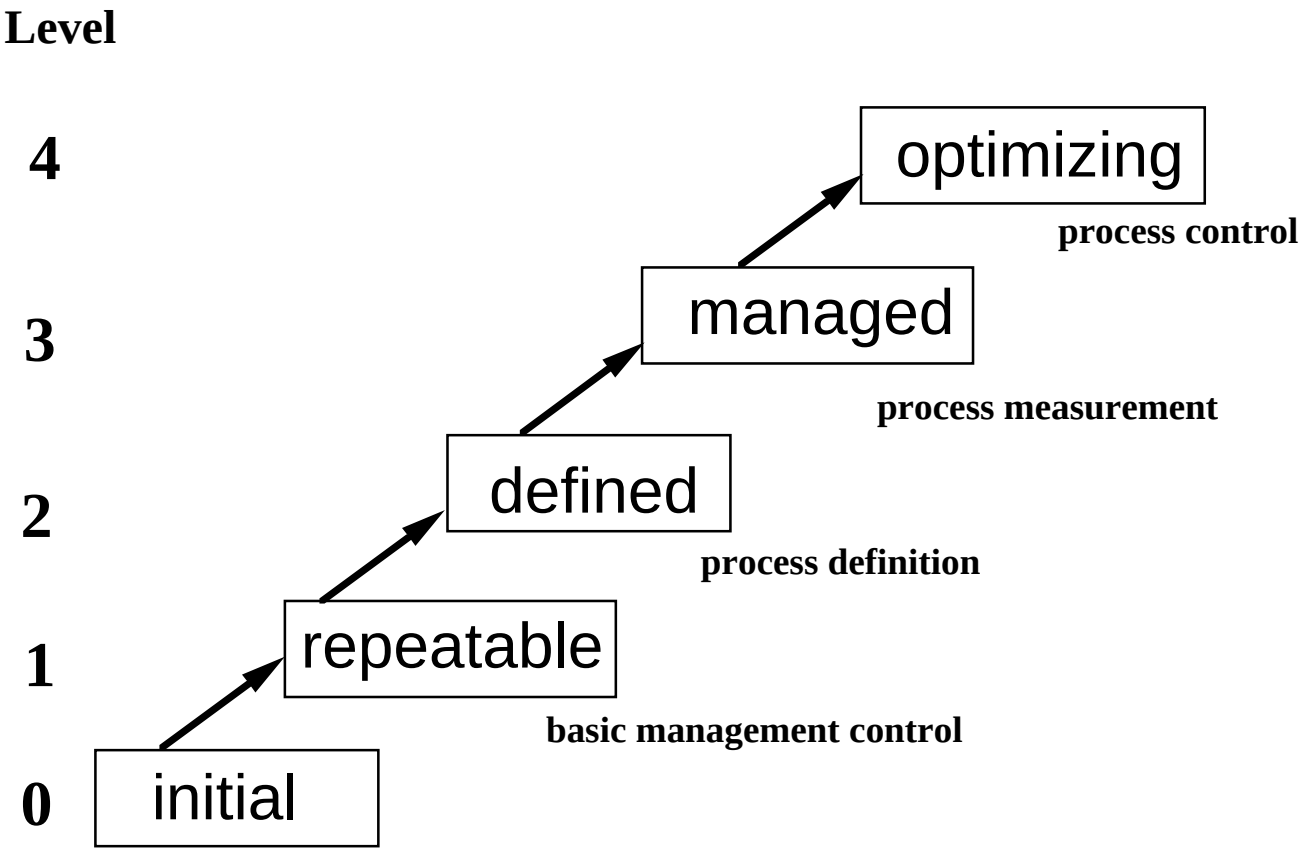
○ **empirical results:**
2 studies "*support*", 0 studies "*mix*", 1 studies "*not supported*"

● **empirical results:**
1 studies "*support*", 1 studies "*mix*", 1 studies "*not supported*"

● **empirical results:**
0 studies "*support*", 3 studies "*mix*", 0 studies "*not supported*"

Process Maturity Level Model

[Humphrey 1990]



Summary

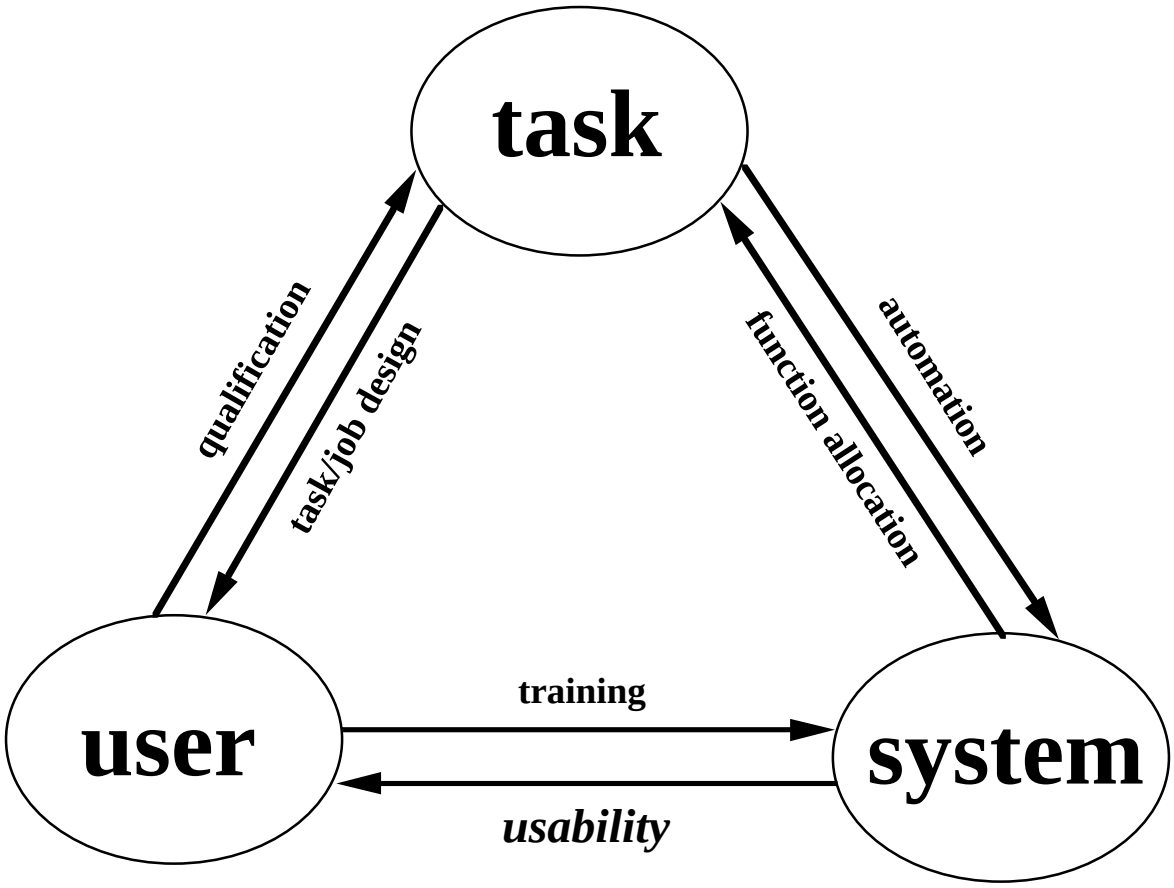
The more effort at the beginning,
the less cost at the end.

Requirements analysis is a social process,
so methods of work and organizational
design are absolutely necessary.

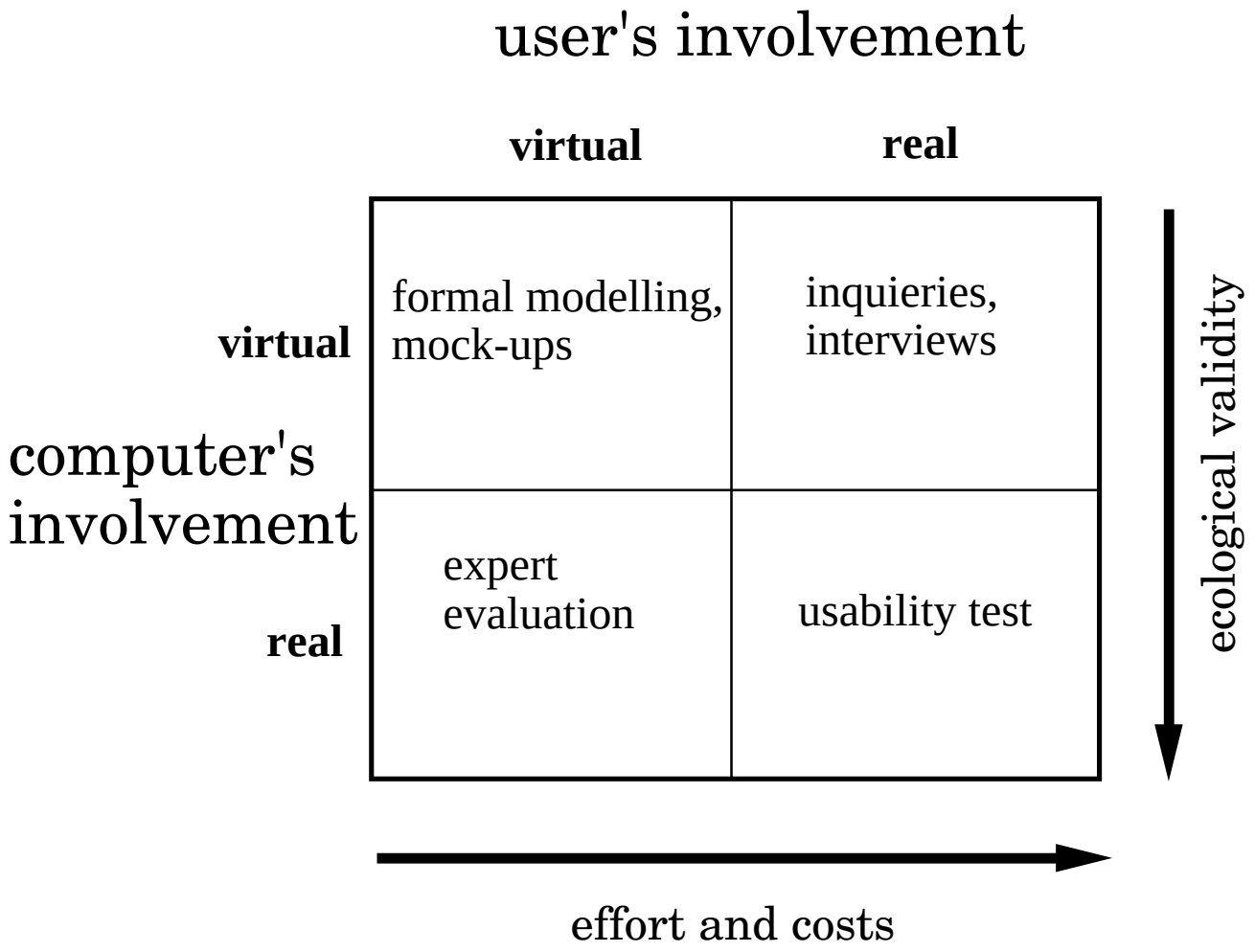
User participation in the context of
iterative-cyclic software development
leads to optimal results.

The iterative-cyclic process model can be
described as a collection of meshed
optimization cycles.

The Bermuda Triangle



a classification of existing methods



Usability Methods

Usability inspection is the name of a set of highly cost-effective methods for finding usability problems and improving the usability of a user interface design by inspection.

Topics to be covered include...

- Definition of usability inspection,
- the heuristic evaluation method,
- other inspection methods.
- Relation between usability inspection methods and user testing.
- Severity of usability problems found by usability inspection.
- Cost-benefit chrarcteristics of usability inspection methods.
- Positioning inspection in the usability engineering lifecycle.

Evaluation

Assessing the usability of an **existing design**

- finding usability problems (to fix them)
- formative evaluation: improve interface, find good/bad parts
- summative evaluation: are goals met?

Only one part of the usability engineering lifecycle
(task analysis, goal setting, design, prototyping, iteration, field studies, etc.)

Inspection methods

• **pluralistic walkthrough** [Bias 1991]

– define a scenario (linear path through interface)

– get users, designers/developers, usability specialists in one room

– show user interface one screen at a time (e.g., overheads)

– have participants write down problems *before* discussion

– discuss the screen (let users speak first)

{may use designer/developer as 'living manual' for early help}

• **standards inspection** [Wixon, Jones, Tse & Casaday 1994]

– have a standard expert inspect interface for compliance

{may cover most of standards without much task knowledge}

• **consistency inspection** [Wixon, Jones, Tse & Casaday 1994]

– team of designers/developers (one from each project) inspects a set of interfaces for more than one system/application, one at a time

• **feature inspection** [Bell 1992]

– imagine typical user task

– list sequence of features used to accomplish the task

– check for long sequences, cumbersome steps, additional knowledge, etc.

• **cognitive walkthrough** [Polson, Lewis, Rieman & Wharton 1992]

– imagine typical user task

– use the system to perform the task, 'defining' the correct solution sequence

– hand-simulate user's problem solving process at each step

– check if user's goal/memory leads to the defined solution sequence

• **quantitative metrics** [Rauterberg 1994]

Evaluation methods

• highly informal evaluation: **heuristic evaluation**

Look at interface and make lists of its problems [Nielsen and Molich 1990]:

– according to checklist of established usability heuristics

– may also apply any additional usability knowledge

Two or more passes through interface:

– inspect flow of interface

– inspect each screen (dialog box, system message, etc.), one at a time

Typical session length: 1–2 hours.

May use observer to help evaluator and note problems mentioned.

Afterwards: aggregate lists of problems from multiple evaluators

• informal evaluation: **usability inspection**

Goals to be met in a somewhat systematic way:

– generate list of usability problems (main goal)

– contribute to building design rationale (artifact inspection)

– provide feedback in design courses [Nielsen et al. 1992]

– evolve a parallel design [Nielsen 1993]

Tools support for inspection:

– mostly none

– online forms for cognitive walkthroughs [Lewis et al. 1992]

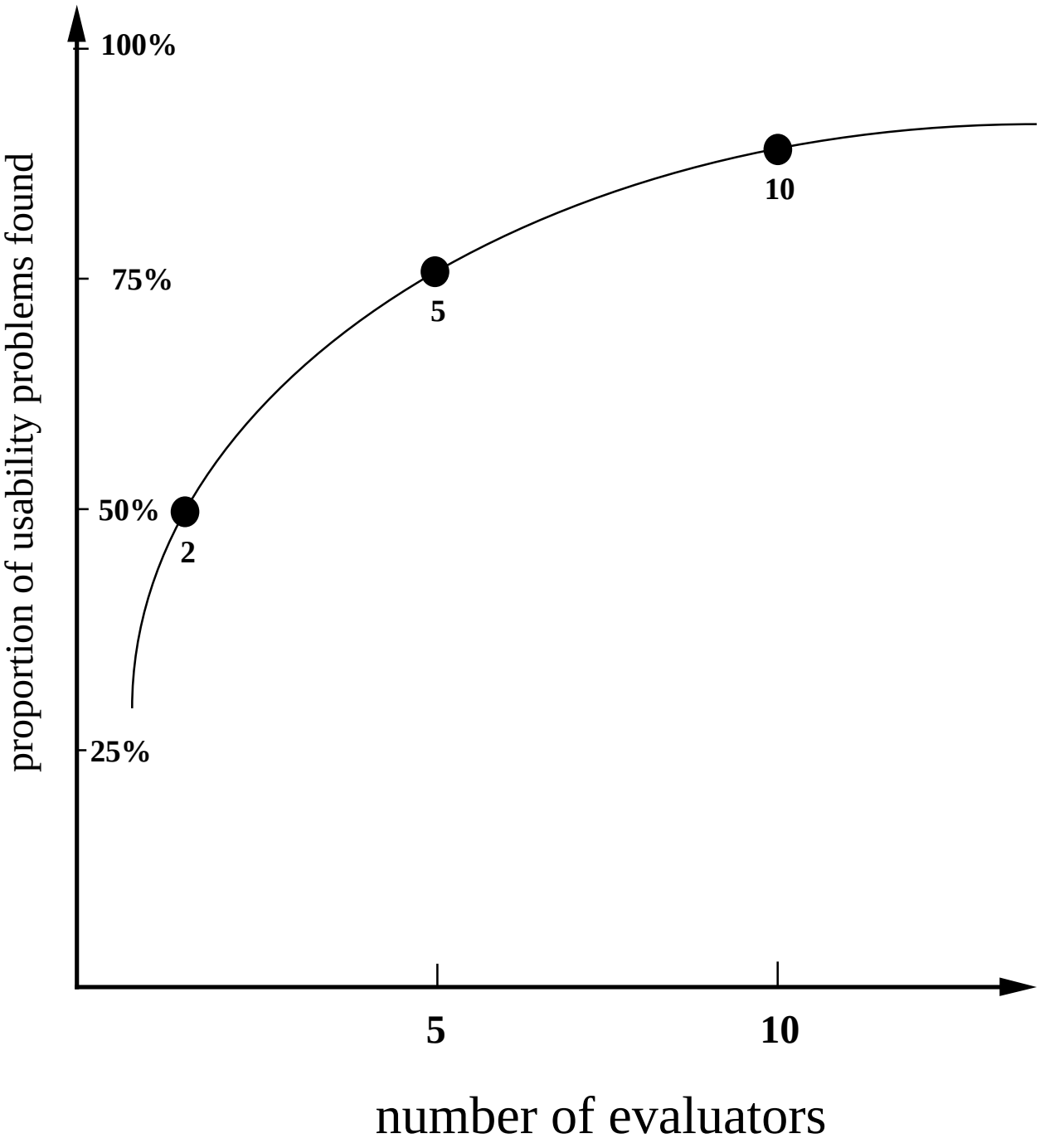
– online/hypertext guidelines/standards documents

– CSCW tools for team heuristic evaluations

(show panel to be discussed for annotation/drawing/pointing)

• structured evaluation: **usability tests**

Efficiency of Inspection Methods



finding problems (mean of 6 evaluation studies) [Nielsen 1993]

Heuristic Evaluation (1):

(see [Molich and Nielsen 1990, Nielsen 1993])

• simple and natural dialog:

Dialogs should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and disminishes their relevative visibility. All information should appear in a natural and logical order.

• speak the user's language:

The dialog should be expressed clearly in words, phrases and concepts familiar to the user, rather than in system-oriented terms.

• minimize the user's memory load:

The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

• consistency:

Users should not have to wonder whether different words, situations, or actions mean the same thing.

• feedback:

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Heuristic Evaluation (2):

(see [Molich and Nielsen 1990, Nielsen 1993])

• clearly marked exits:

Users often choose system functions by mistake and will need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialog.

• shortcuts:

Accelerators–unseen by the novice user–may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.

• good error messages:

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

• prevent errors:

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

• help and documentation:

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Inspection/Evaluation methods and Task scenario

no task scenario	partly	full task scenario
heuristic evaluation **standards inspection** **consistency inspection** **quantitative metrics**	**heuristic inspection** (if evaluator lack domain knowledge) **pluralistic walkthrough**	**feature inspection** **cognitive walkthrough** **user testing**

Note: **without a task scenario you can**
+ reach a broader inspection coverage
+ evaluates 'corners' of interface

– overlook some task-interface mismatches

Inspection/Evaluation methods and Evaluators

team (3-5 indepent eval.)	expert	user
pluralistic walkthrough **consistency inspection** **heuristic evaluation**	without domain knowledge: **standards inspection** **quantitative metrics** with domain knowledge: **feature inspection** **cognitive walkthrough**	**usability testing**

Inspection of Paper Designs

Advantages

• paper mock-up allows evaluation early in lifecycle

• possible because inspection does not rely on use of system

• heuristic evaluation may perform better with paper **interfaces**

Disadvantages

• 'missing dialog element' problems are much harder to find
[they are slightly easier to find in running interfaces]

Note:

mostly, same problems easy/hard to find in heuristic evaluation

• menu choices should have consecutive numbers
[easy to find on paper, hard to find on running system]

• long strings of digits (page #) hard to remember
[easy to find on running system, hard to find on paper]

Methods Summary

Use multiple methods to supplement each other

• feature inspection *early* in design process
• heuristic evaluation of paper mock-ups and running design
• standards and consistency inspection to coordinate projects
• alternate inspection and usability test to avoid 'wasting users'

Aggregate inspection results from multiple evaluators

• individual evaluators only find about 35% of all problems
• three evaluators find about 63%; five find 74%
• base severity ratings on mean of at least three evaluators

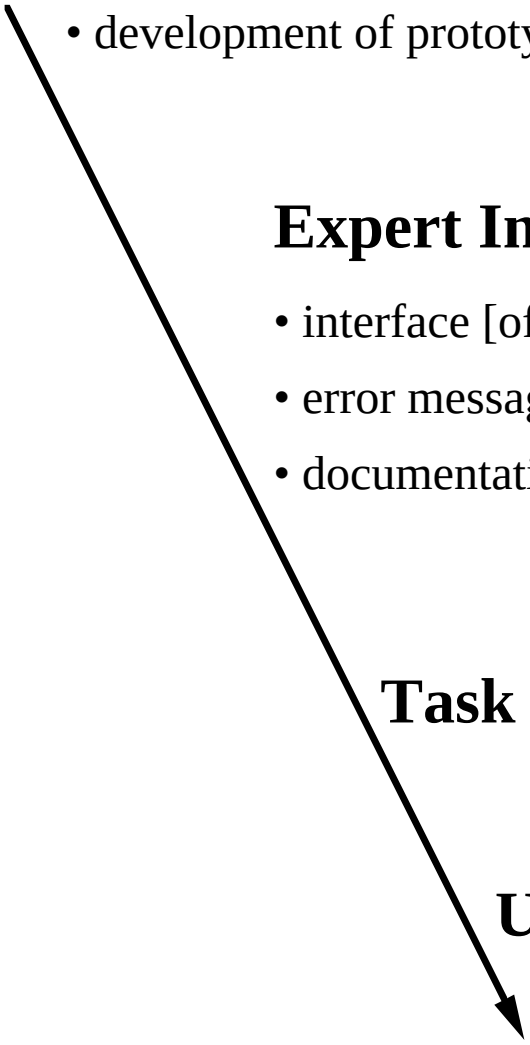
Note:

heuristic evaluation is a 'discount usability engineering' method: cheap, quick, and simple to learn/use (but use it systematically!)

Usability Activities

Participation in the early design phase

- paper mock-ups
- development of prototyps



Expert Inspection

- interface [of running system]
- error messages
- documentation (online / print out)

Task oriented Inspection

Usability Testing

Inspection Methods

'driving questions'

• are there any unneccessary steps?

• has the dialog structure a task oriented design?

• is the design rational/concept appropriate for the target group?

• are the error messages… really helpful?

with context dependent hints?

with detailed instructions?

• are different languages supported?

date/time notation

calendar

numbers/digits

expert's role

• try to think and decide as a user

• solve representative tasks

• don't test product features!

• make errors, behave stupid!

• forget your product knowledge, use the help system

Usability Testing (1)

What is a usability test?

"A method to register and measure the interactive quality of an interactive system."

Laboratory or field testing:

run the test…

- with real end users (the 'tester')
- with representative tasks
- in a 'real' environment

record/protocoll…

- all critical incidents
- all comments of users and observers
- the subjective impressions

control/measure

- mental work load

Usability Testing (2)

formative evaluation

- one tester group (N ≤5)
- one interactive system
- ask for the testers opinion
- *aim*: to find critical incidents

summative evaluation

- two or more tester groups (N ≥12)
- measure the tester's computer experience
- two or more interface versions
- measure the mental work load
- ask for the testers opinion
- compare the performance
- analyse the data with applied statistics
- *aim*: to choose the best system

Usability Testing (3)

neccessary components

- **scenario / tasks**

  select the appropriate task environment

- **real end user as tester**

  select from all parts of the target group

  [tasks, sex, age, experience, hierarchy, departments, languages, …]

- **interactive system**

  minimial hardware / maximal software

- **observer**

  project manager, designer, developer, training instructor, additional end users

- **record data**

  audio/video, logfile, questionnaires,

Usability Testing (4)

important constrains

- **test time ≤ 2.5 hours per tester**

  end users are quickly tired!

- **test only important functions**

  high frequency of usage
  critical task aspects

- **test only main interaction types**

  typical usage, avoid redundancy

- **test complete tasks**

  end users should be successful!

- **socio-psychological aspects**

  don't blame the user/tester!

Usability Testing (5)

important actions

- **briefing of the tester**

  "We test the software, not you!"

  "You can cancel at any time!"

  – explain the test environment

  – ask for user's permission to record data

- **record data/critical incidents**

  is all equipment ready?

  is date/time stamp in the video correct?

  is date/time in the protocol correct?

- **collect all test material**

  clip papers together, put them into a folder

- **debriefing of tester**

  give time for questions, pay, thank the user!

Usability Testing (6)

preparatory actions

- **install the hard/software**
  - run the installed configuration
- **write instructions and task description**
  - can the task be carried out?
  - how much task solving time is needed?
- **write questionnaire**
  - are all questions answerable and relevant?
  - are important questions missing?
- **inform and invite user and observer**
  - make sure that all people will be ready
- **delegate responsibilities**
  - camera, logging, help, observing …
- **test the test setting**
  - run one pre-test!

Usability Testing (7)

important aspects

- **at least one observer has *tool* knowledge**
  - the main effort is getting familiar with the interactive system/tool/product
- **at least one observer has *domain* knowledge**
  - big effort is neccessary to get familiar with the task domain
- **criteria for a critical incident**
  - if you are surprised about the user's action
  - if the user react emotionally
  - discrepancies among observers
- **find out the user's intention**

  thinking aloud, video confrontation

Usability Testing (8)

inquiery and questionnaire

- **subjective impressions**

  – what did you like?
  – what did you dislike?
  – is something missing?
  – is something redundant?
  – what would you change?

- **subjective rating**

  monopolar:
  1 = good , 2= better, 3 = best
  bipolar:
  –2=very bad, –1=bad,…,+2=very good

- **decide for one criteria value**

  – e.g., on average better than 1,5

- **space for additional comments**

Usability Testing (9)

a typical test schedule

- **pro scenario 2 to 2.5 hours**
- **five end users/testers**
- **additional usability methods**
  - design discussion
  - prototyping
  - inspection
  - task oriented simulation

	forenoon	afternoon
monday	pre-test	
tuesday	user 1	user 2
wednesday	user 3	discussion
thursday	user 4	user 5
friday	report writing	

Usability Testing (9)

observers

- **designer, developer**
  – is responsible for software changes
- **information manager**
  – is responsible for manual changes
- **quality manager**
  – is responsible for quality assurance
- **domain/tool expert**
  – is responsible for helping hints
- **end user**
  – is responsible for task-tool mismachtes
- **usability person-1**
  – is responsible for test setting
- **usability person-2**
  – is responsible for data recording

Usability Testing (10)

observation room

- **video recorder**
- **'shadow' screen**
- **logging software**
- **several monitors**
- **the usability person should take care for…**
  - a comfortable atmosphere
  - dimmed lightning
  - comfortable chairs
  - refreshments
  - breaks

test room

- **should looks like a normal working place**

  \+ one way mirror
  \+ video camera, microfon

  [the tester knows, that he will be observed!]

Usability Testing (11)

report writing

- **analyse the protocolls**
- **rate critical incidents for their severity**

  [1] must be fixed at once
  [2] must be fixed before delivery
  [3] should be fixed before delivery
  [4] should be fixed in the next release
  [5] should be fixed in the future

- **analyse quantitative data**
  - frequencies
  - mean plus standard deviation

highlight tape editing

- **video sequences with critical incidents**

References

Joseph Dumas & Janice Redish
A practical guide to usability testing.
Ablex Pub., 1993

Joan Greenbaum & Morton Kyng (ed)
Design at work. Lawrence Erlbaum, 1991

Cem Kaner, Jack Falk & Hung Quoc Nguyen
Testing computer software. van Nostrand Reinhold, 1993

Lei Lei
User participation and the success of information system development.
Tinbergen Inst. Research Series No. 73, 1994

Andrew Monk, Peter Wright, Jeanne Haber & Lora Davenport
Improving your human-computer interface.
Prentice Hall, 1993

Jakob Nielsen
Usability engineering. Academic, 1993

Jakob Nielsen & Robert Mack (eds.)
Usability inspection methods. Wiley, 1994