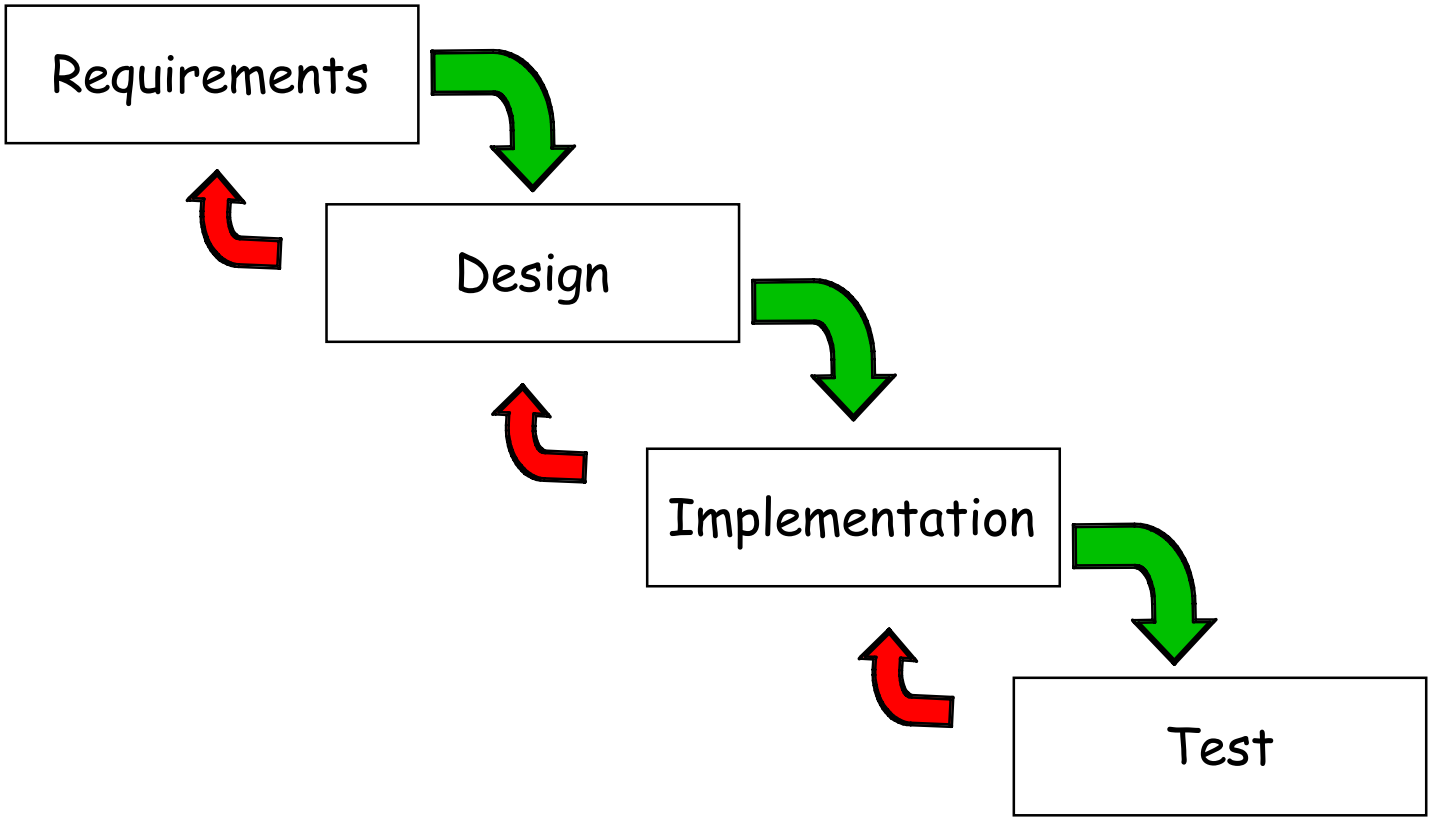
Software Engineering 2007

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
E2-201
Information Systems
February 6, 2007

Concepts (Warm Up)

Software
Software Engineering
Vs. Computer Science
Vs. System Engineering
Software Process
Software Process Model
Software Development Costs
Software Engineering Methods
CASE
Software Attributes

Waterfall Model



Agile & Iterative Development

Larman, Chapters 1-5

Background & Problems

waterfall: response to ad-hoc code-and-fix 1960's, Winston Royce (proponent of iterative dev.) - e.g. DoD standard 2167

system should be clearly specified before its design and implementation

clients are not sure what they want

details will only be revealed during development

as the product develop, clients change their minds

external forces (competitor's product)

-> high change rate, not predictable manufacturing

Manufactoring vs. Development

Predictable Manufacturing	New Product Development
It is possible to first complete specifications, and then build.	Rarely possible to create upfront unchanging and detailed specs. and then build.
Near the start, one can reliably estimate effort and cost.	Near the beginning, it is not possible. As empirical data emerge, it becomes increasingly possible to plan and estimate.
It is possible to identify, define, schedule, and order all the detailed activities.	Near the beginning, it is not possible. Adaptive steps driven by build-feedback cycles are required.
Adaptation to unpredictable change is not the norm, and change-rates are relatively low.	Creative adaptation to unpredictable change is the norm. Change rates are high.

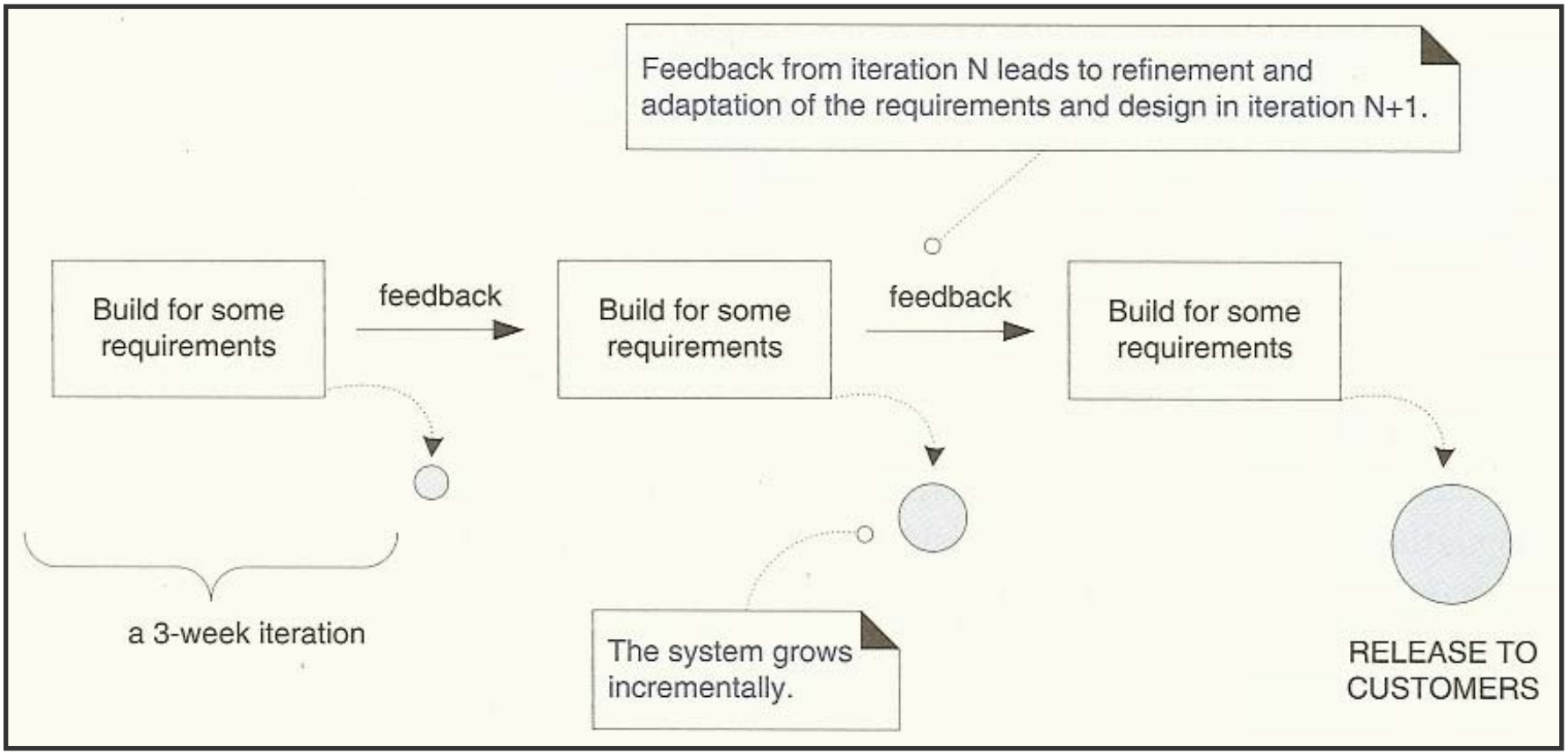
ID - Iterative Development

ID is an approach to build software of several iterations
each iteration a mini-project

- requirements, design, implementation, test
- iteration release: a stable, integrated and tested partially complete system

IID - Iterative and Incremental Development

Fig. 2.1



Iterative and incremental development

Agile methods are a subset of iterative and evolutionary methods

The key practices:

- risk-driven and client-driven
- timeboxing
- evolutionary and adaptive development
- evolutionary requirements analysis
- evolutionary and adaptive planning rather than predictive planning

Risk-and Client-Driven Iterative Planning

What to do in the first/next iteration?

Risk-driven: the riskiest, most difficult elements for the early iterations

Client-driven: the choice of features comes from client, the currently highest business value

Timeboxing

the practice of fixing the iteration end-date and not allow it to change

if short of time, reduce the scope and leave features to the next iteration

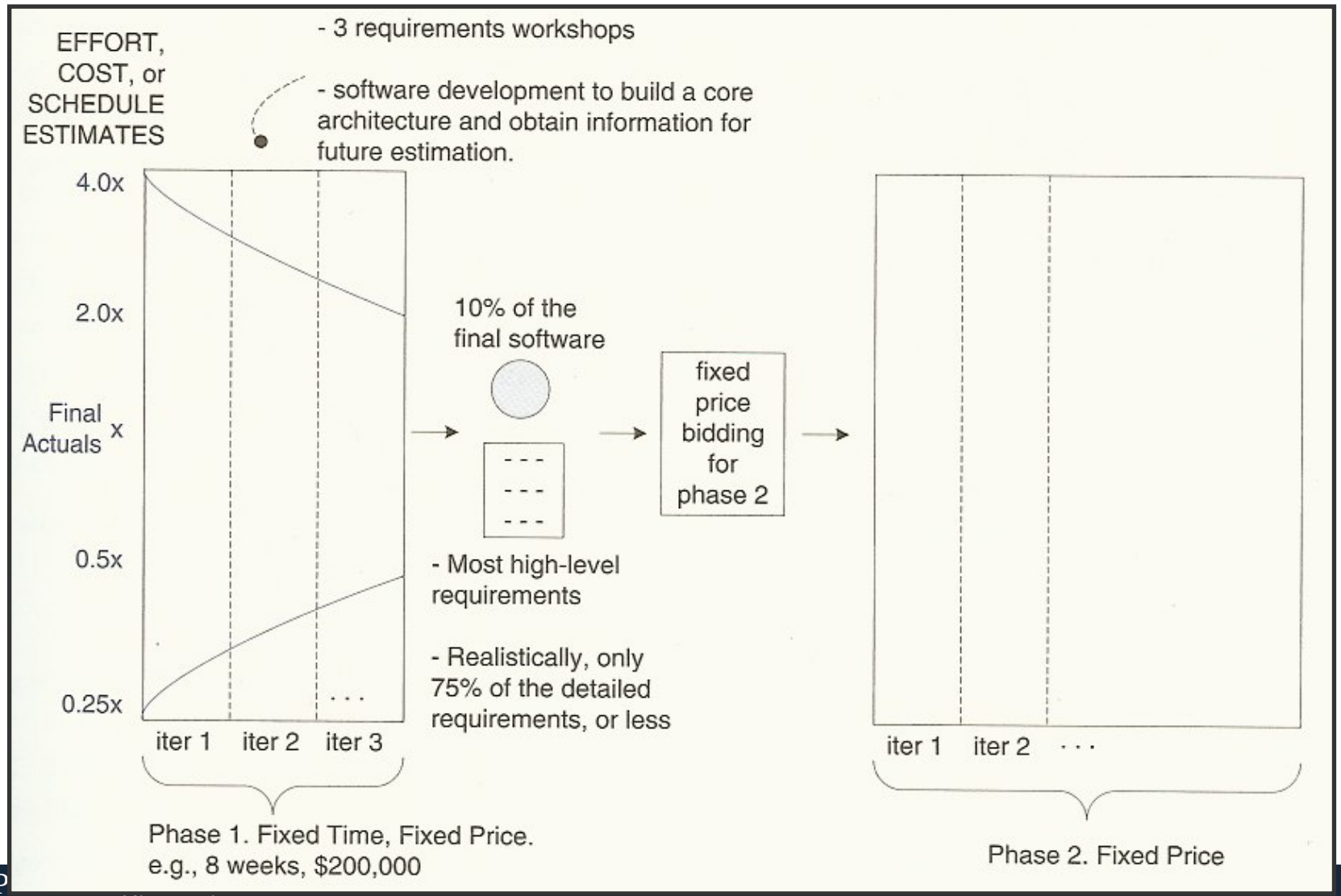
no adding of new tasks to iteration

Evolutionary and Adaptive Development

requirements, plan, estimates, solution evolve and are refined
requires feedback from users, tests, developers

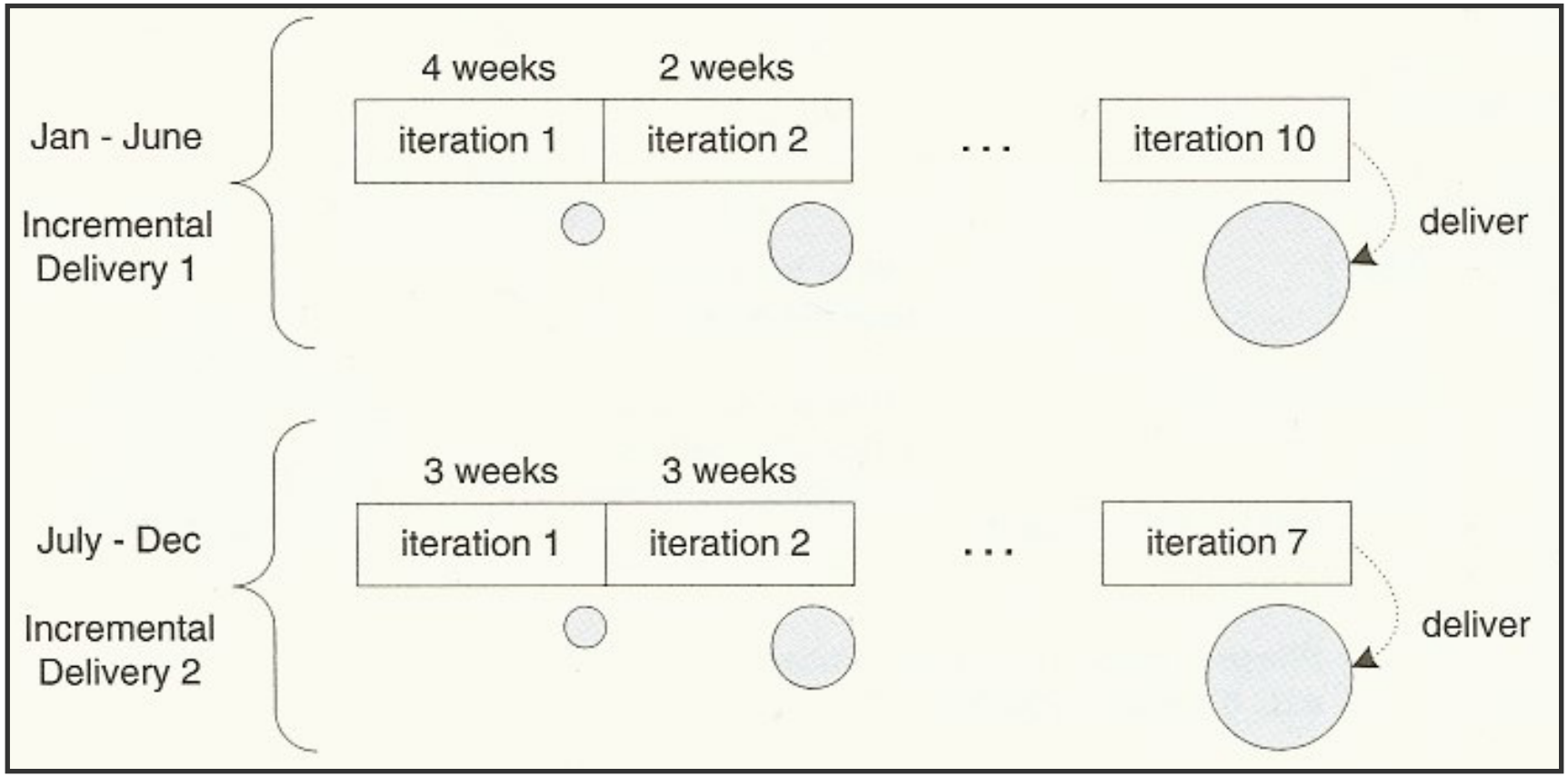
Two contract phases

Fig. 2.6



Incremental delivery with iterations

Fig. 2.7



The Agile Manifesto

Individuals and interactions	*over process and tools*
Working software	*over comprehensive documentation*
Customer collaboration	*over contract negotiation*
Responding to change	*over following the plan*

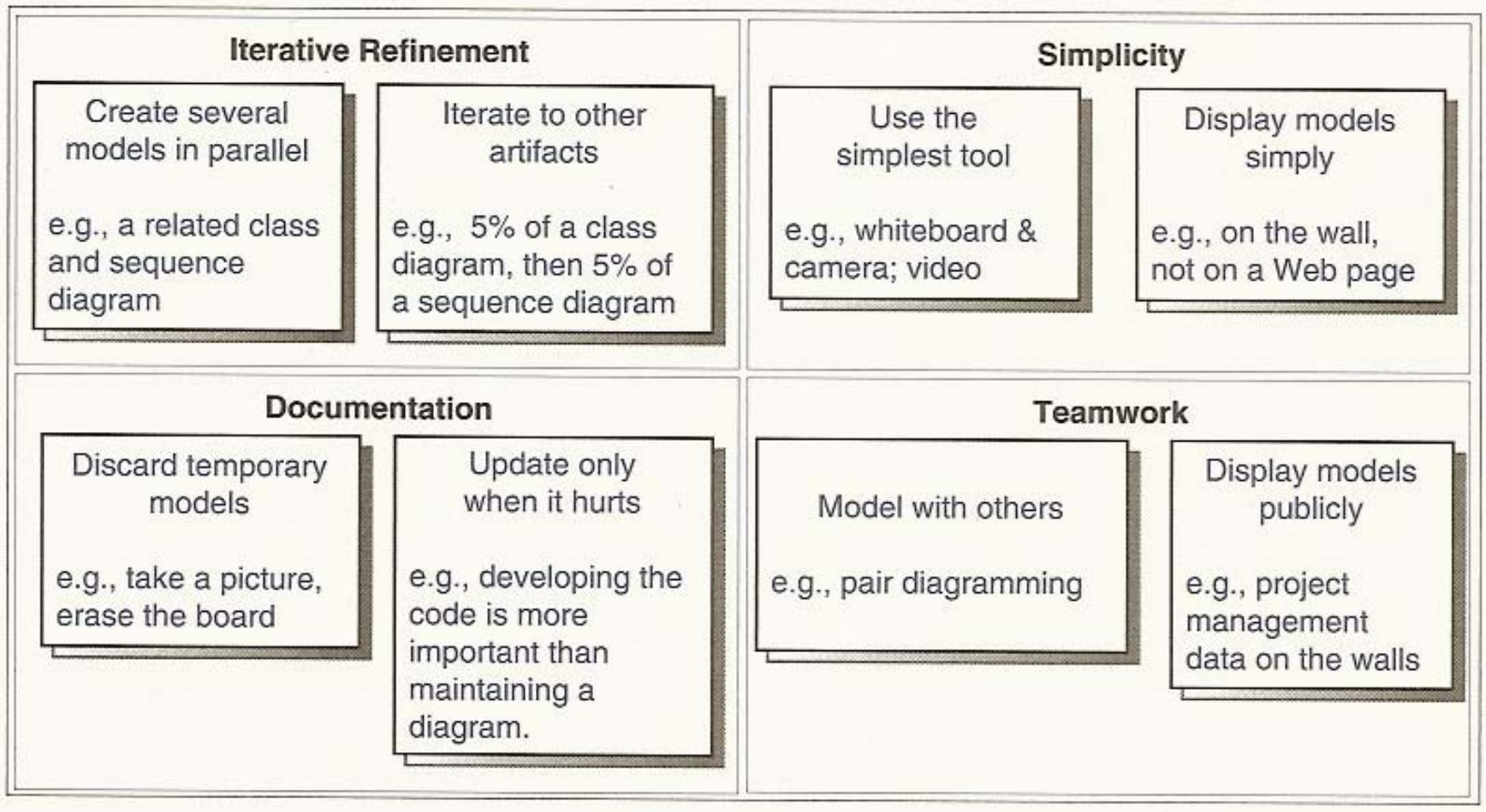
That is, while there is value in the items on the right, we value the items on the left more

The Agile principles

1 early and continuous delivery of valuable software	8 agile processes promote sustainable development
2 welcome changing requirements, even late	9 sponsors, developers, and users maintain constant pace
3 deliver working software frequently	10 attention to technical excellence and good design enhances agility
4 business people and developers together daily	11 simplicity – the art of maximizing the amount of work not done – is essential
5 motivated individuals	12 the best architectures, requirements and designs emerge from self-organizing teams
6 face-to-face conversation	13 regular reflections in the team on how to become more effective
7 working software is the primary measure of progress	

Agile Modeling

Fig. 3.3



Complete Requirements Not Known From Start

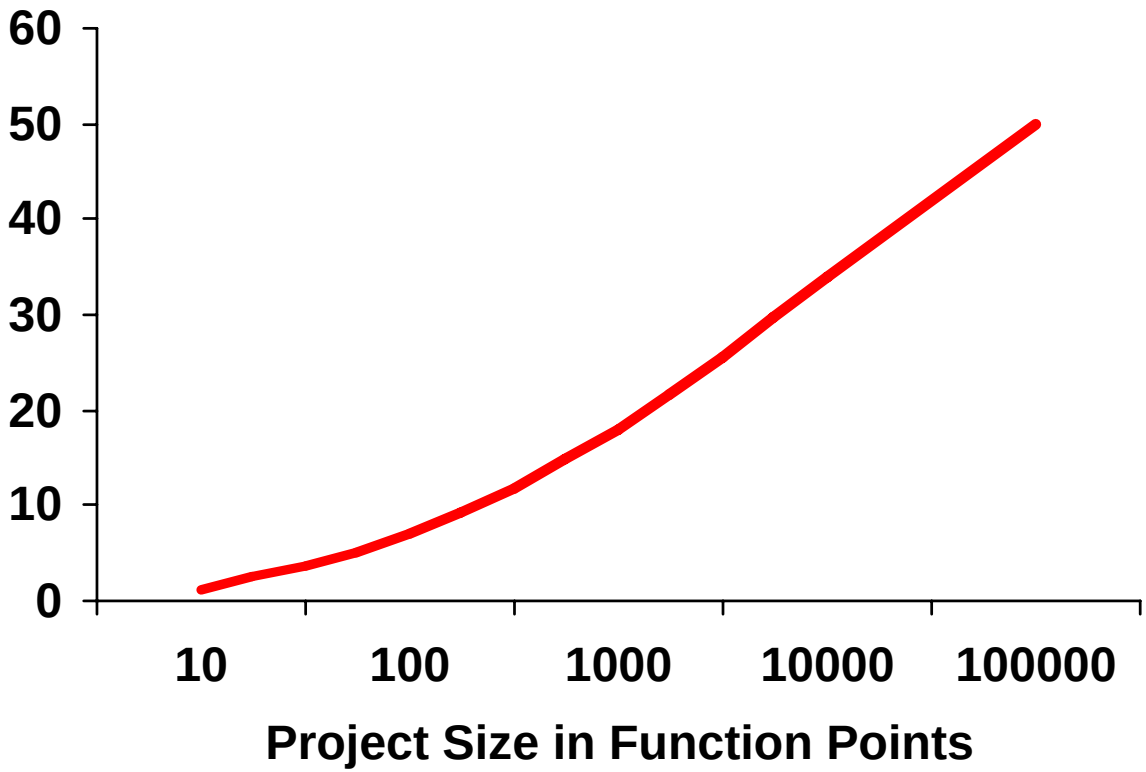
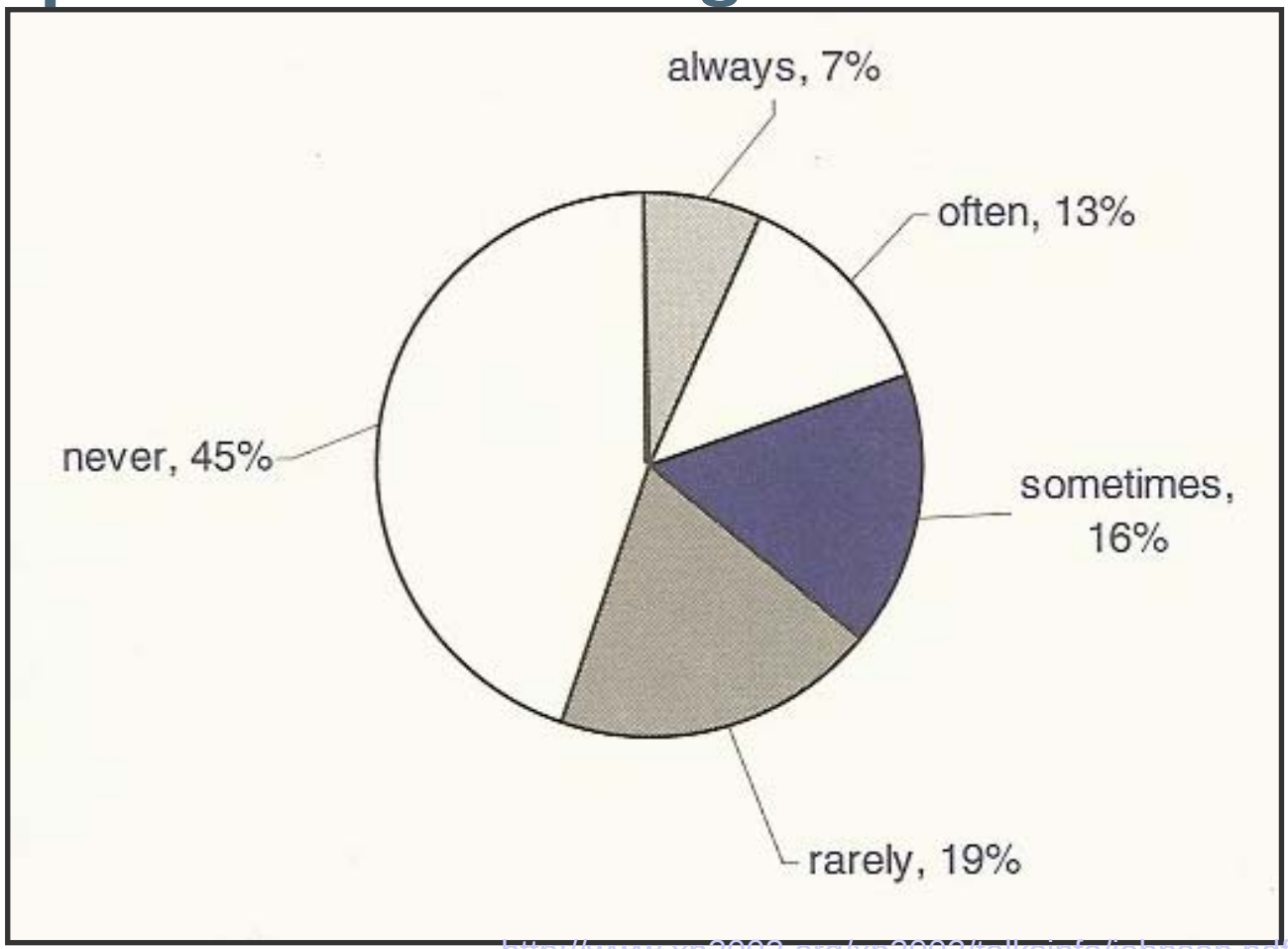


Fig. 5.1 rates of change on software projects

Source: *Applied Software Measurement*, Capers Jones, 1997. Based on 6,700 systems.

The Requirements Challenge



http://www.xp2003.org/xp2002/talksinfo/johnson.pdf

Fig. 5.3 actual use of requested features