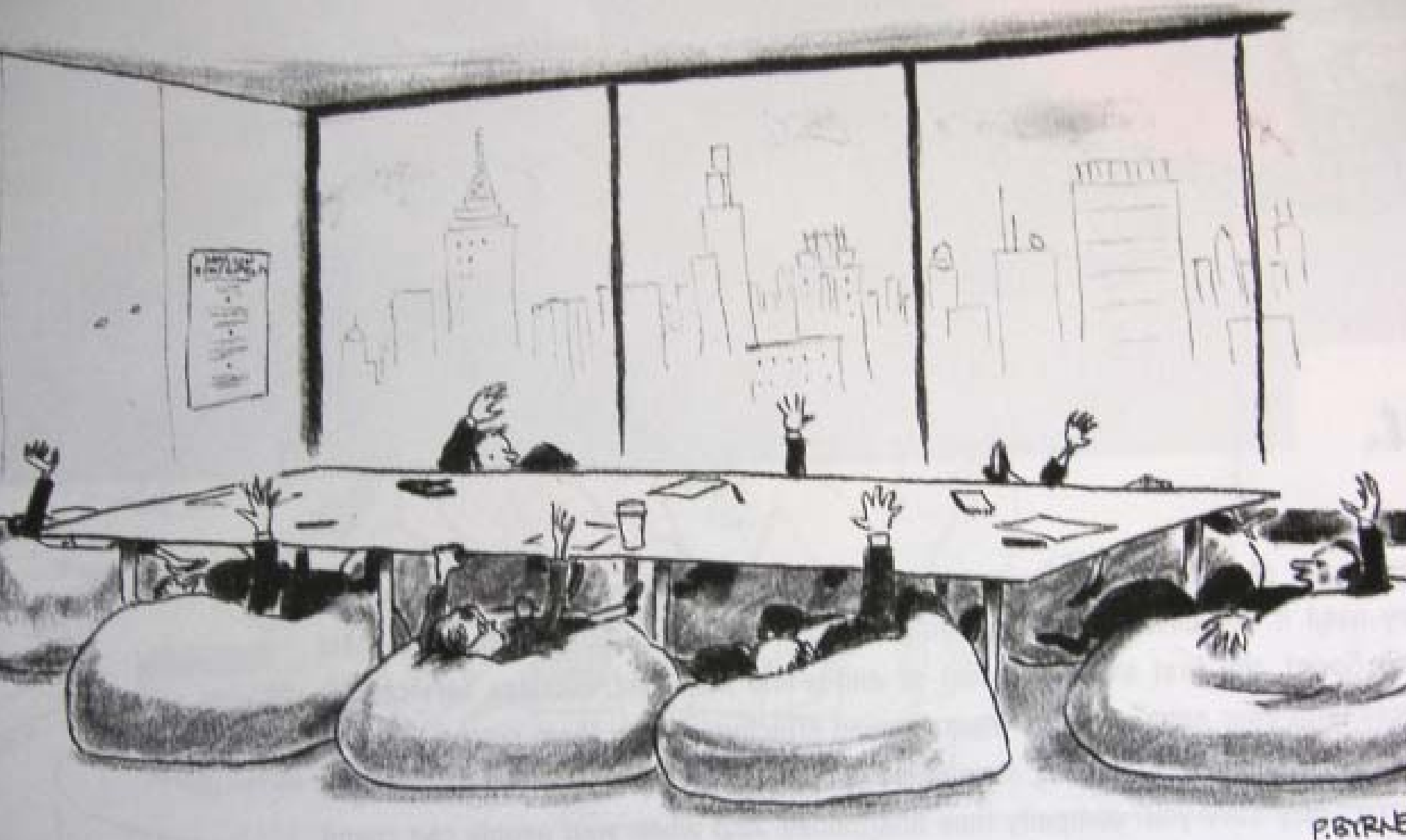
History and Evidence

Iterative , Evolutionary & Agile vs. the Waterfall

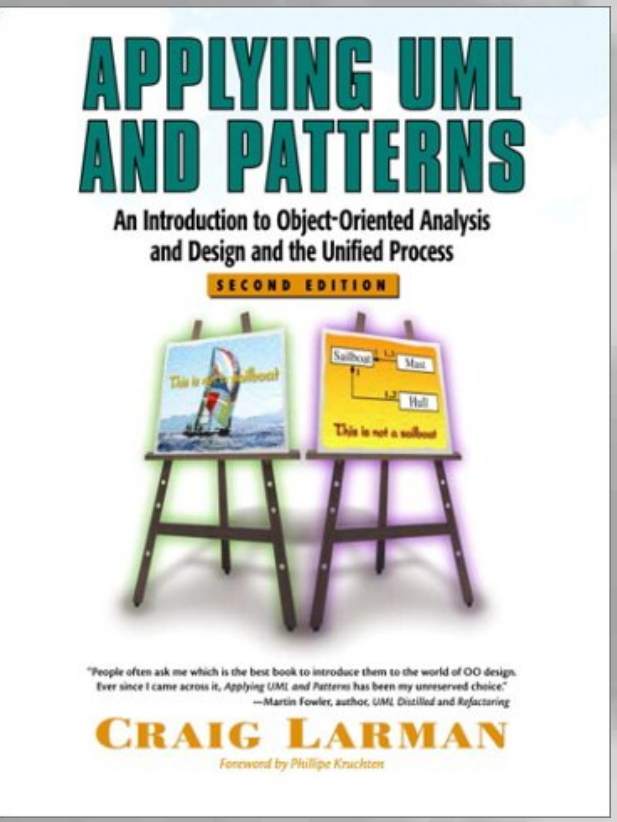
Craig Larman

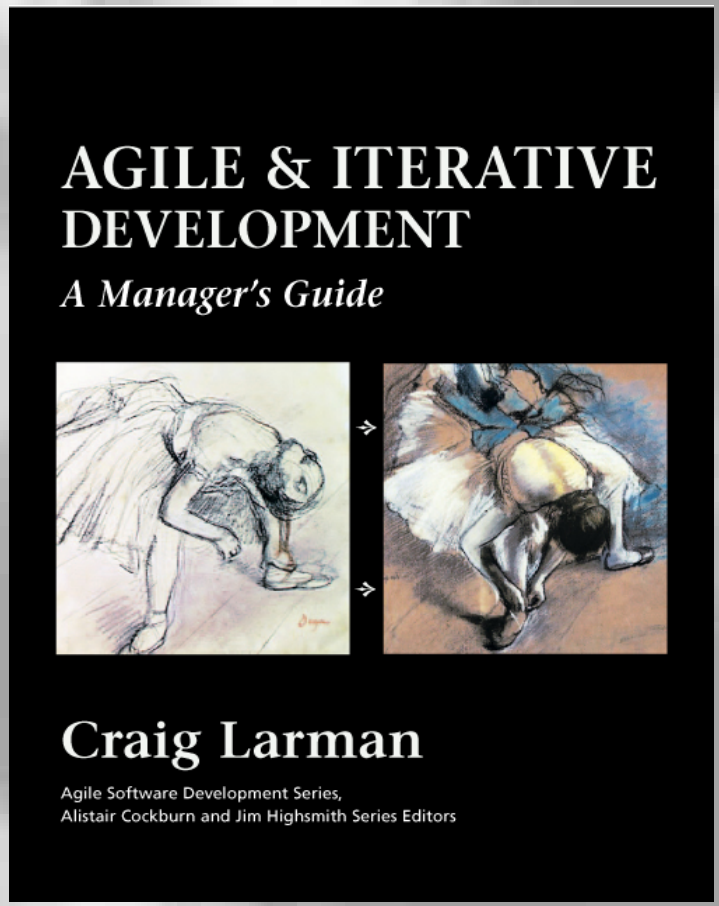
www.craiglarman.com

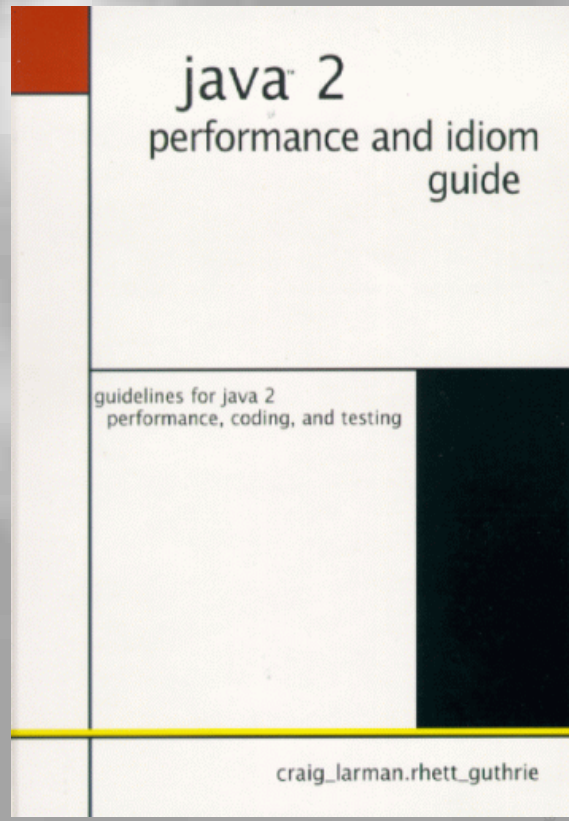


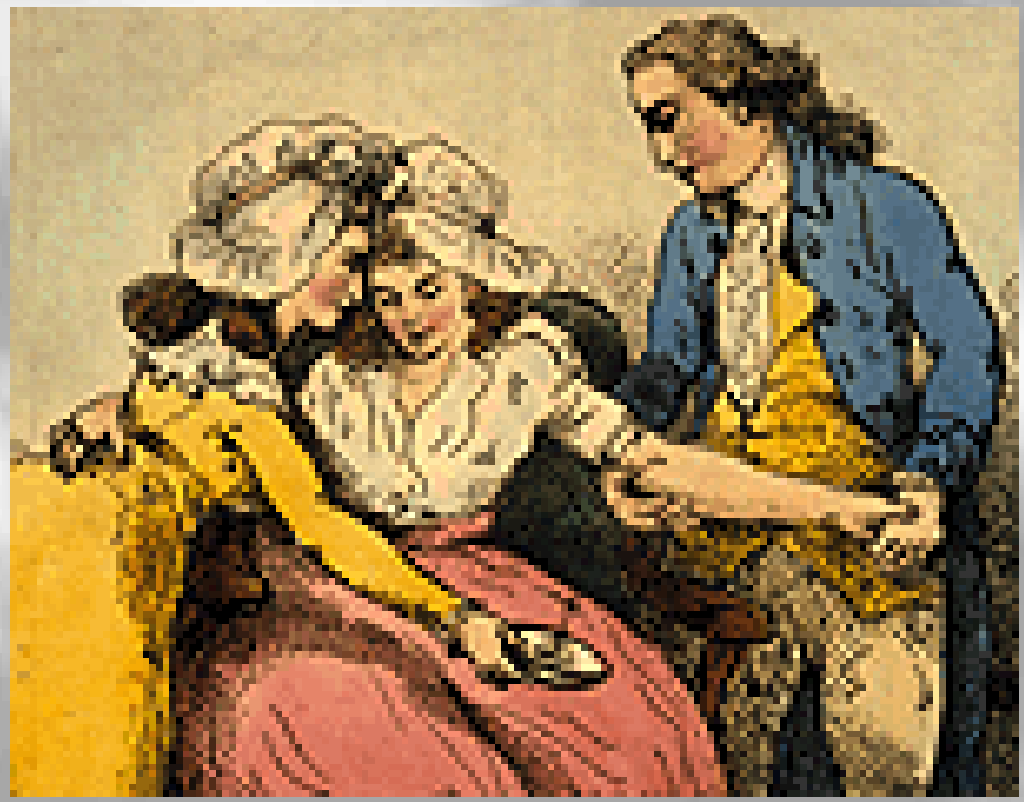
"All in favor of the old, uncomfortable chairs?"

- www.craiglarman.com
- chief scientist, Valtech





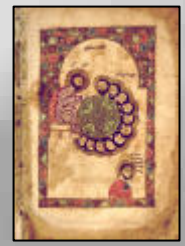


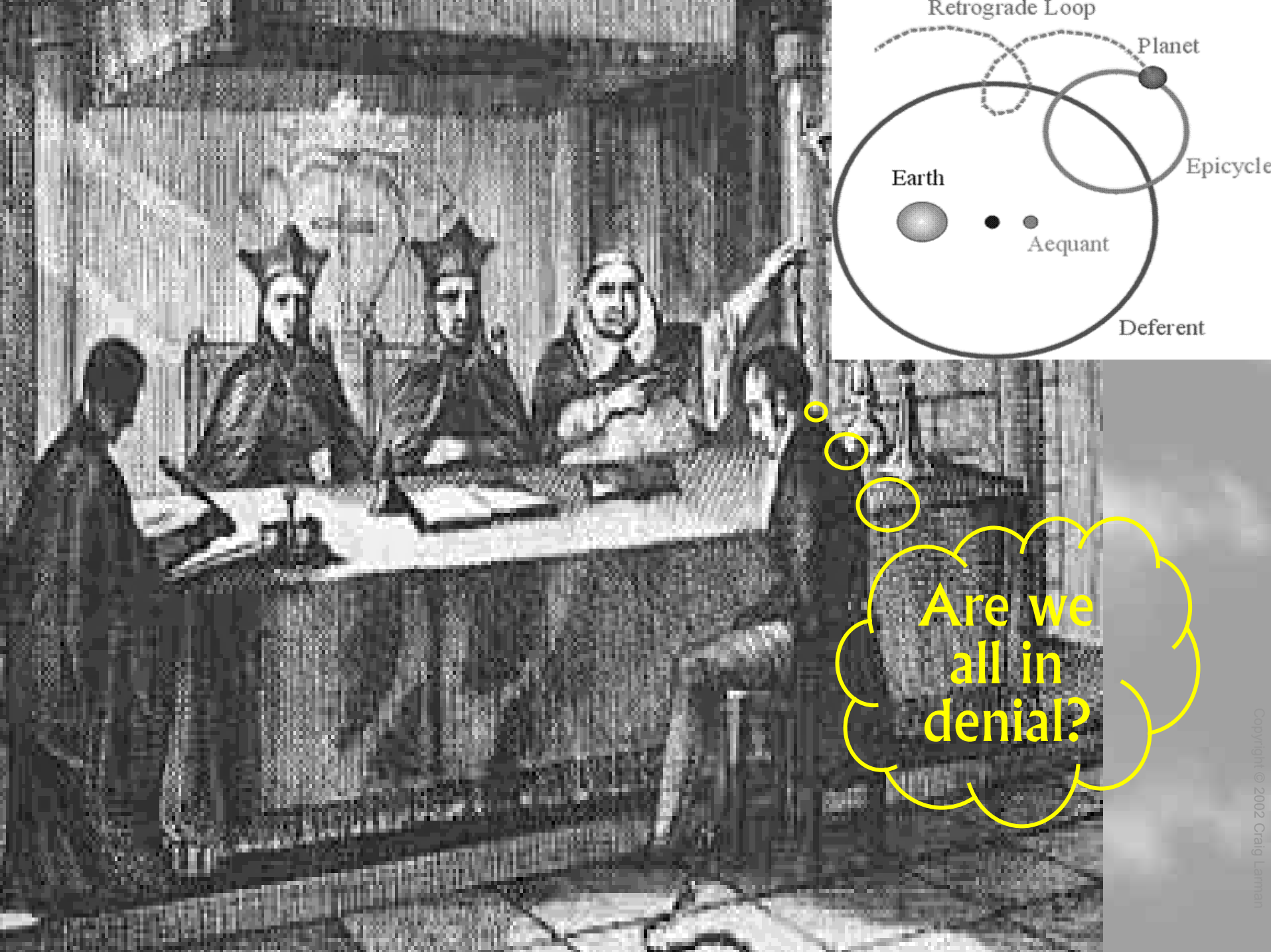


Blood Letting Certified
BL Maturity Model
Level 4

Blood Letting 101

Sacred Blood Letting
Manuscripts






PMI
ISO
Requirements "specialists"
Evidence-Based Iterative Practitioner

THE STRUCTURE OF SCIENTIFIC REVOLUTIONS

THIRD EDITION

THOMAS S. KUHN

- [Jones97] and [BP88]

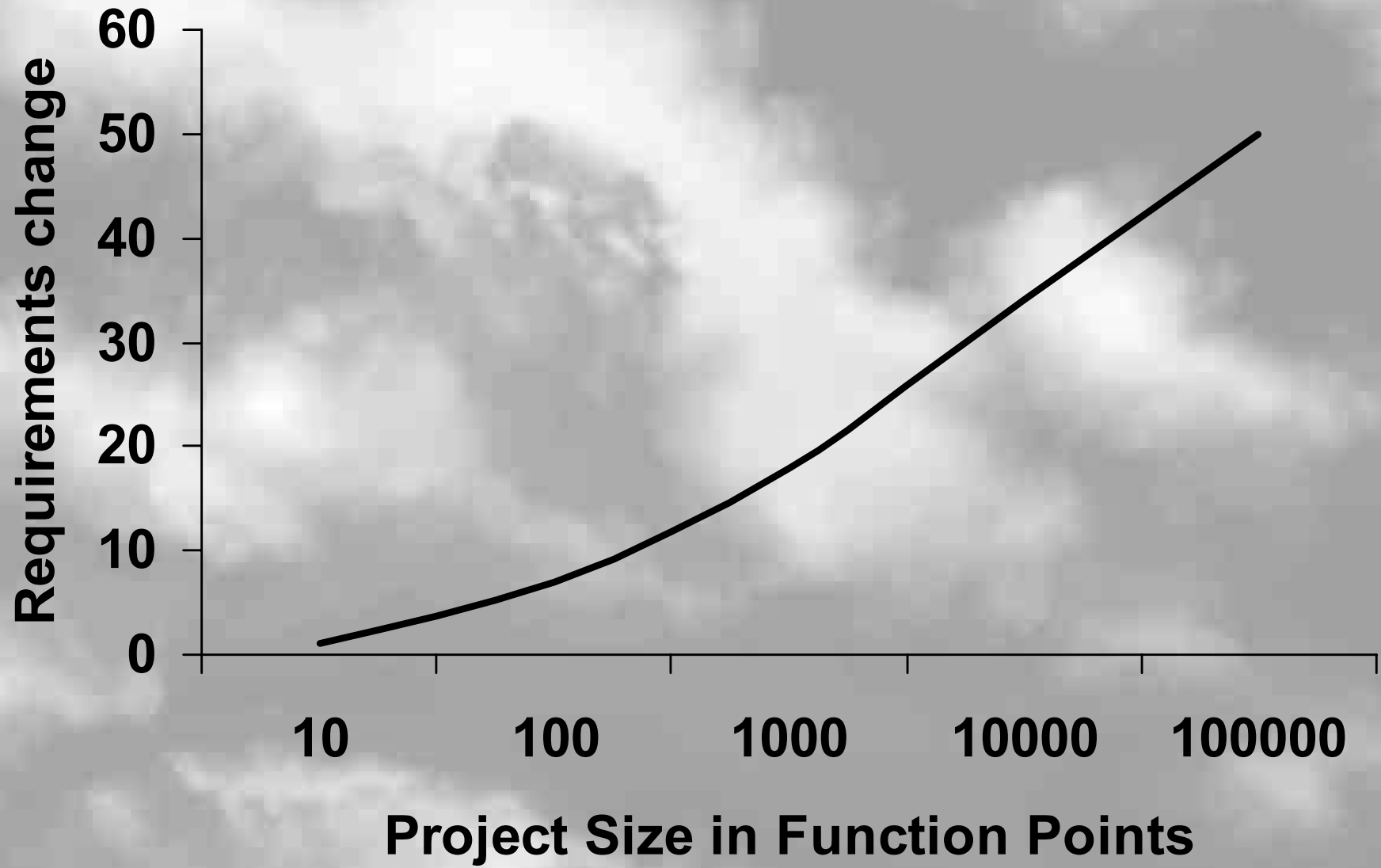
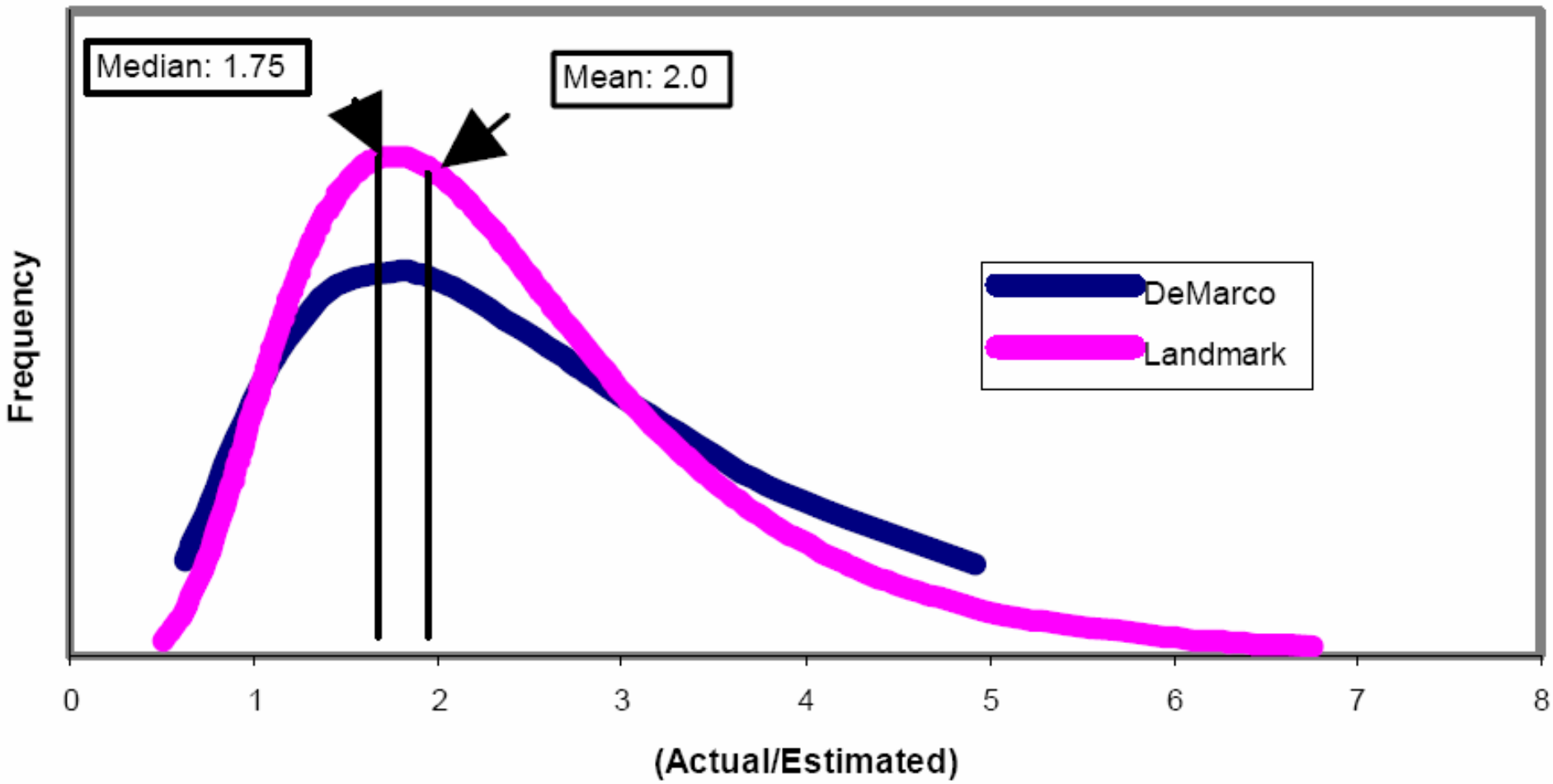


Figure 4: Probability Distribution Curve of Actual/Estimated



Paradigm Shift

Software dev is predictable manufacturing

Waterfall

Big up-front specs

Predictive plans

. . .

Software dev is new product development

Iterative

Evolutionary specs

Adaptive plans

History

Iterative and Incremental Development: A Brief History

Although many view iterative and incremental development as a modern practice, its application dates as far back as the mid-1950s. Prominent software-engineering thought leaders from each succeeding decade supported IID practices, and many large projects used them successfully.

Craig Larman
Valtech

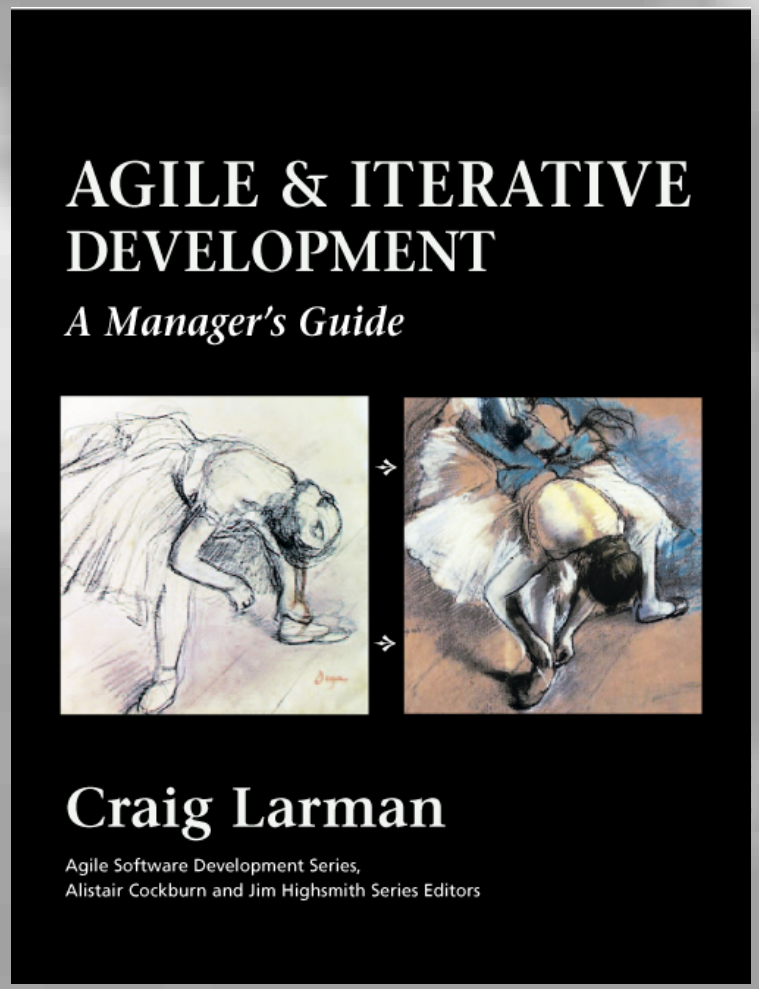
Victor R. Basili
University of

As agile methods become more popular, some view iterative, evolutionary, and incremental software development—a cornerstone of these methods—as the "modern" replacement of the waterfall model, but its practiced and published roots go back decades. Of course, many software-engineering stu-

opment" merely for rework, in modern agile methods the term implies not just revisiting work, but also evolutionary advancement—a usage that dates from at least 1968.

PRE-1970

IID grew from the 1930s work of Walter



- *Q: What are the most exciting, promising software engineering ideas or techniques on the horizon?*

- *A: I don't think that the most promising ideas are on the horizon. They are already here and have been for years, but are not being used properly.*

- *—David L. Parnas*

1970

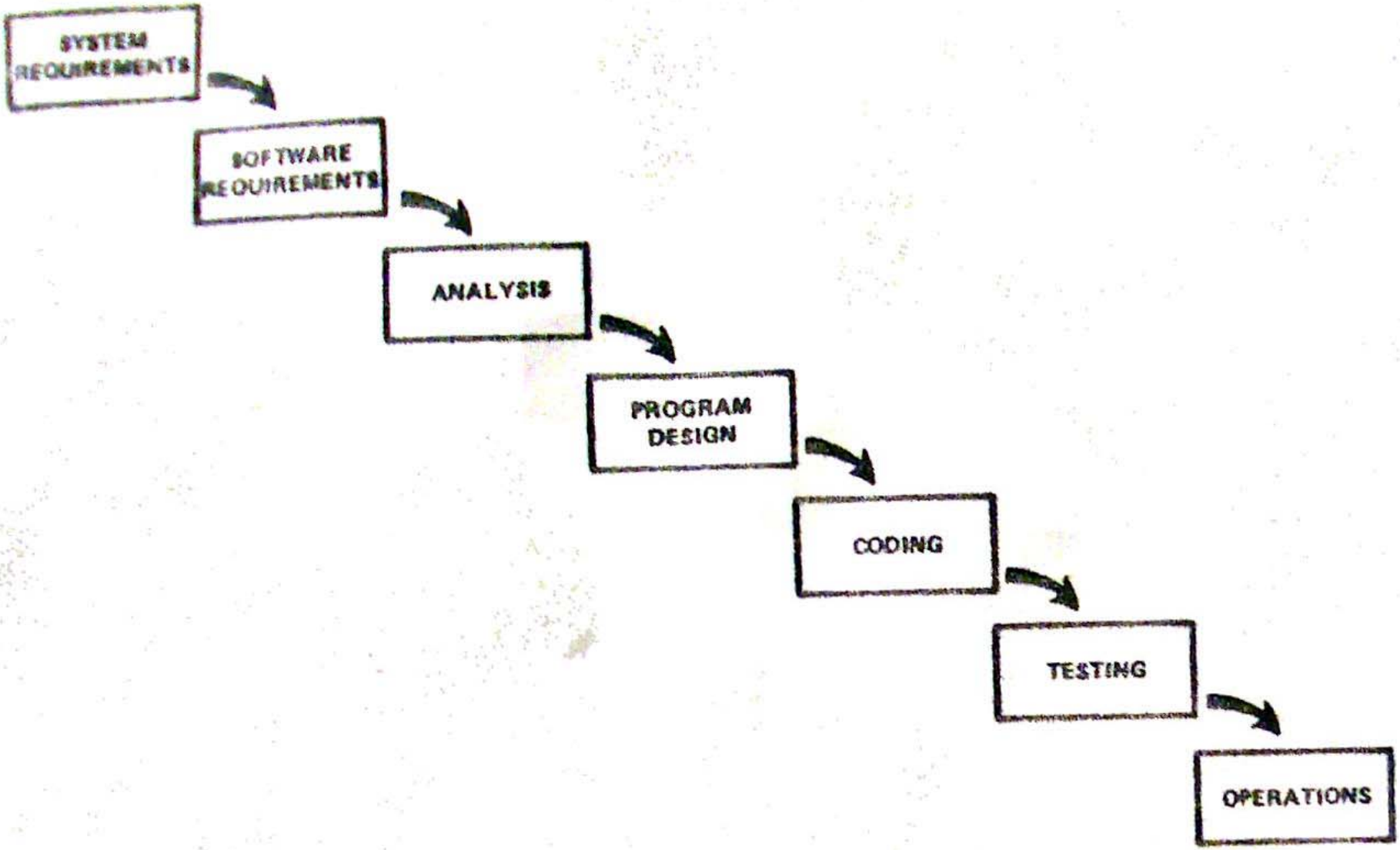
MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS

Dr. Winston W. Royce

INTRODUCTION

I am going to describe my personal views about managing large software developments. I have had

A more grandiose approach to software development is illustrated in Figure 2



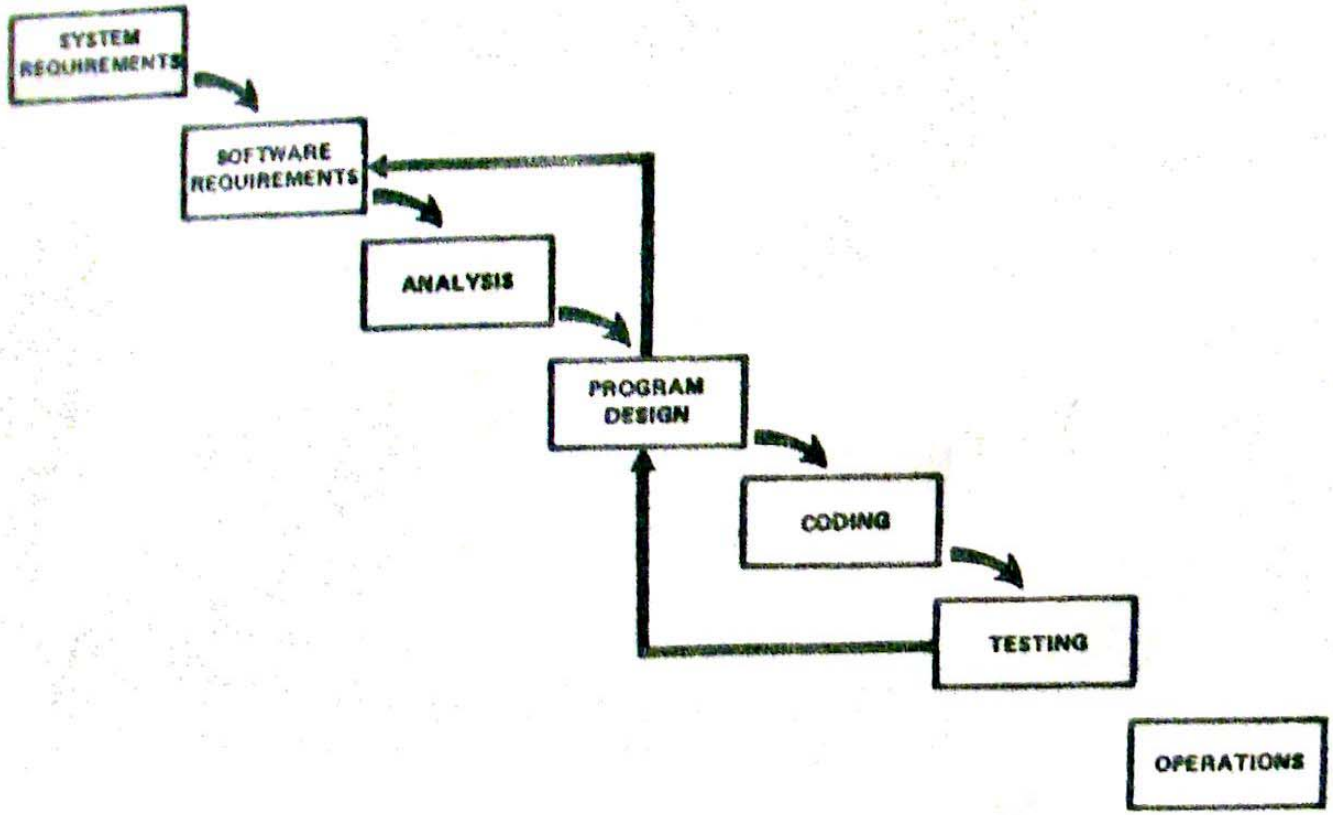
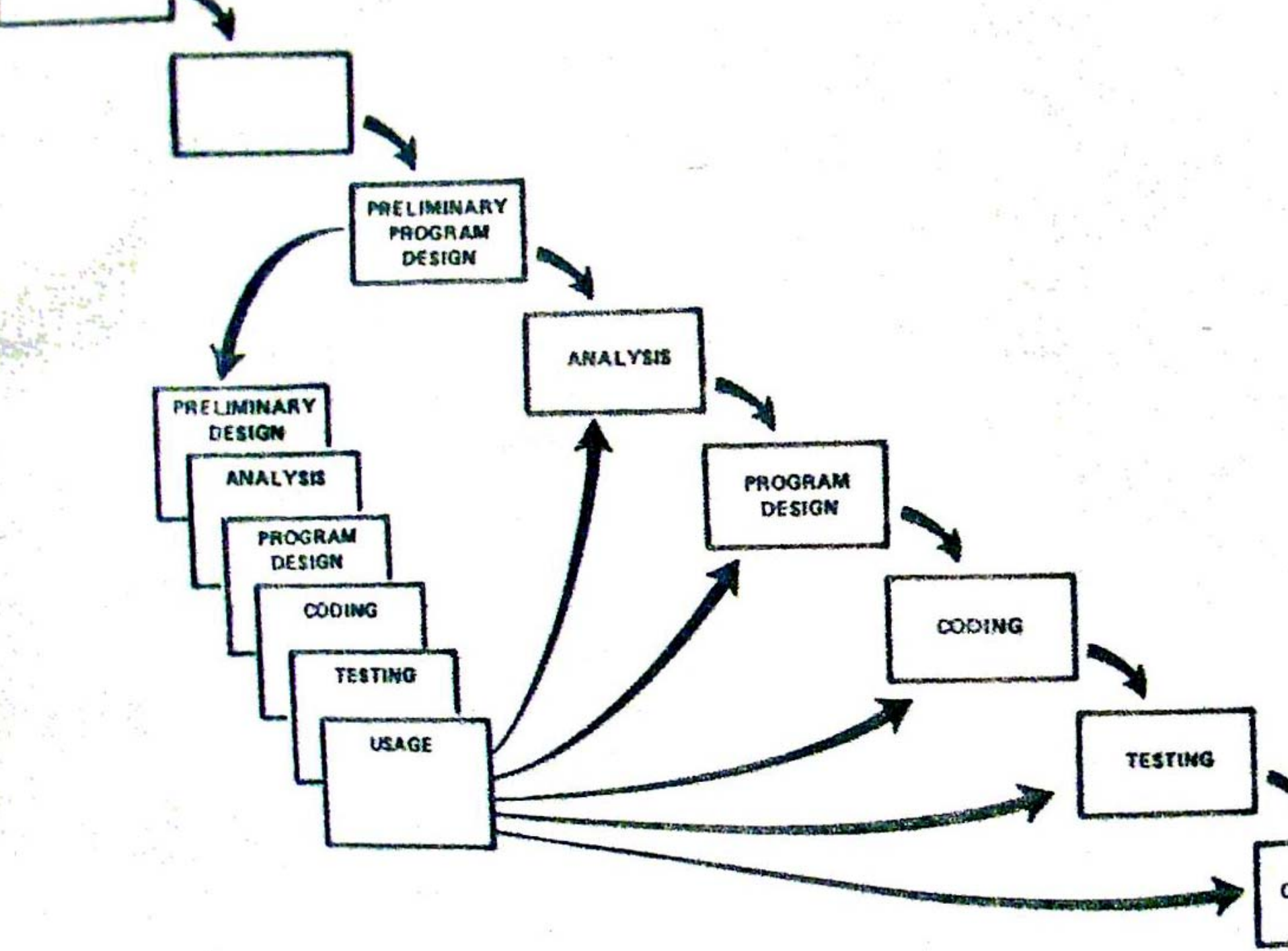


Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.


PRELIMINARY
PROGRAM
DESIGN
PRELIMINARY
DESIGN
ANALYSIS
PROGRAM
DESIGN
CODING
TESTING
USAGE
ANALYSIS
PROGRAM
DESIGN
CODING
TESTING

- Walker Royce, speaking of his father:
  - ***"He was always a proponent of iterative, incremental, evolutionary development. His paper described the waterfall as the simplest description, but that it would not work for all but the most straightforward projects."***

The Polaris System Development

Bureaucratic and Programmatic Success in Government

Harvey M. Sapolsky

Harvard University Press
Cambridge, Massachusetts
1972

4 | PERT and the Myth of Managerial Effectiveness

The Special Projects Office has gained an internation
tation for the innovativeness and effectiveness of the
ment control system it has employed in the develop
the FBM weapon system. PERT, a computerized R8
ning, scheduling, and control technique developed in
the FBM Program, has been extensively used in nume

The Strange Tale of DoD-2167

Report
of the
Defense Science Board
Task Force on

MILITARY SOFTWARE

SEPTEMBER 1987

Frederick P. Brooks, Jr.

Frederick P. Brooks, Jr.
Chairman
Defense Science Board
Task Force on Military Software

MIL-STD-498 (1994)

- ***"DoD must manage programs using iterative development…"***

THE UNDER SECRETARY OF DEFENSE

3010 DEFENSE PENTAGON
WASHINGTON, DC 20301-3010

ACQUISITION,
TECHNOLOGY
AND LOGISTICS

APR 12 2002

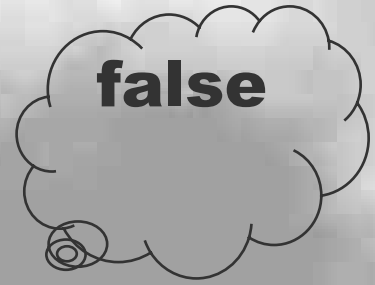
MEMORANDUM FOR SECRETARIES OF THE MILITARY DEPARTMENTS
CHAIRMAN OF THE JOINT CHIEFS OF STAFF
UNDER SECRETARIES OF DEFENSE
ASSISTANT SECRETARIES OF DEFENSE
INSPECTOR GENERAL, DEPARTMENT OF DEFENSE
GENERAL COUNSEL, DEPARTMENT OF DEFENSE
DIRECTORS OF THE DEFENSE AGENCIES

SUBJECT: Evolutionary Acquisition and Spiral Development

Since the publication of DoD Directive 5000.1 and DoD Instruction 5000.2, in which the Department established a preference for the use of evolutionary acquisition strategies relying on a spiral development process, there has been some confusion about what these terms mean and how spiral development impacts various processes such as contracting and requirements generation that interface with an evolutionary acquisition strategy. The purpose of this memorandum is to address those questions.

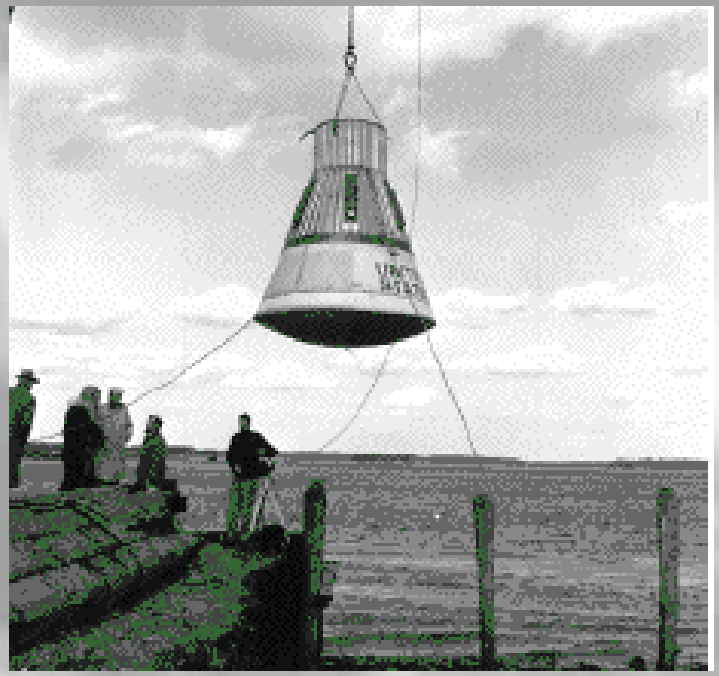
Evolutionary acquisition and spiral development are methods that will allow us to reduce our cycle time and speed the delivery of advanced capability to our warfighters. These approaches are designed to develop and field demonstrated technologies for both hardware and software in manageable pieces. Evolutionary acquisition and spiral development also allow insertion of new technologies and capabilities over time.

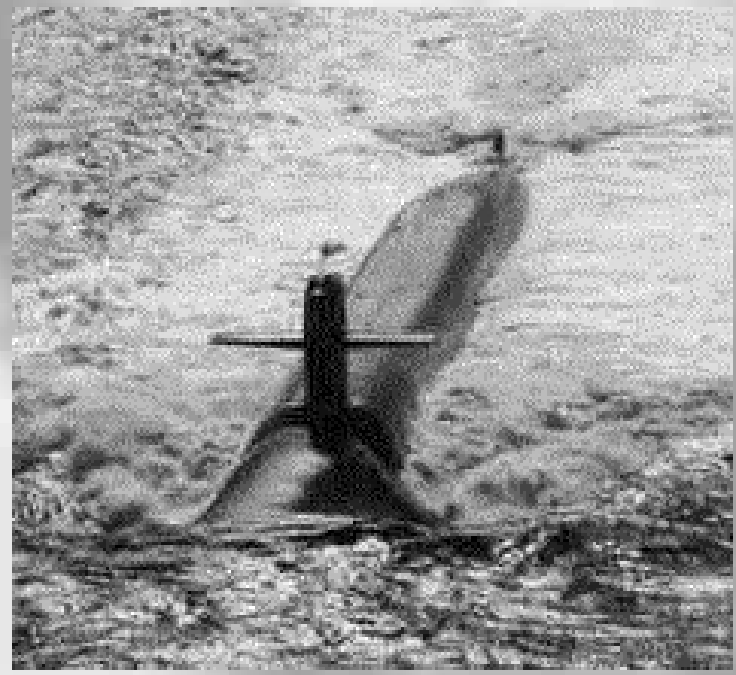
Waterfall → Iterative & Evolutionary



Waterfall →

Iterative & Evolutionary →





- B. Randell and F.W. Zurcher, "Iterative Multi-Level Modeling: A Methodology for Computer System Design," *Proc. IFIP*, IEEE CS Press, 1968
- Describing the Trident practice at IBM FSD:
  - D. O'Neill, "Integration Engineering Perspective," J. Systems and Software, 1983

1970s

Design, Development, Integration: Space Shuttle Primary Flight Software System

The development of Space Shuttle software posed unique requirements above and beyond raw size (30 times larger than Saturn V software), complexity, and criticality.

914 Communications of the ACM

September 1984 Volume 27 Number 9

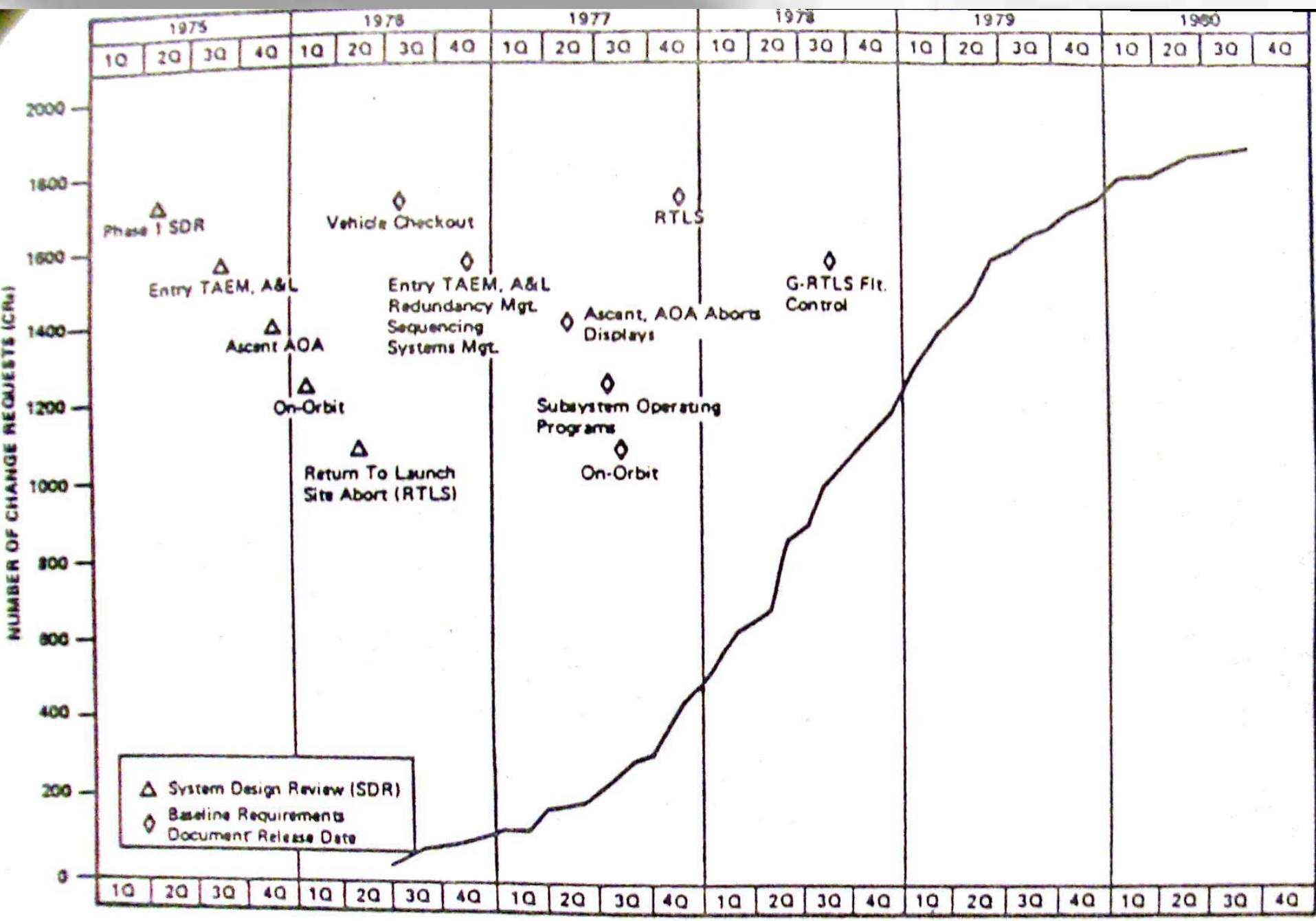


FIGURE 3. STS-1 Flight Software Requirements Change Requests (CRs)

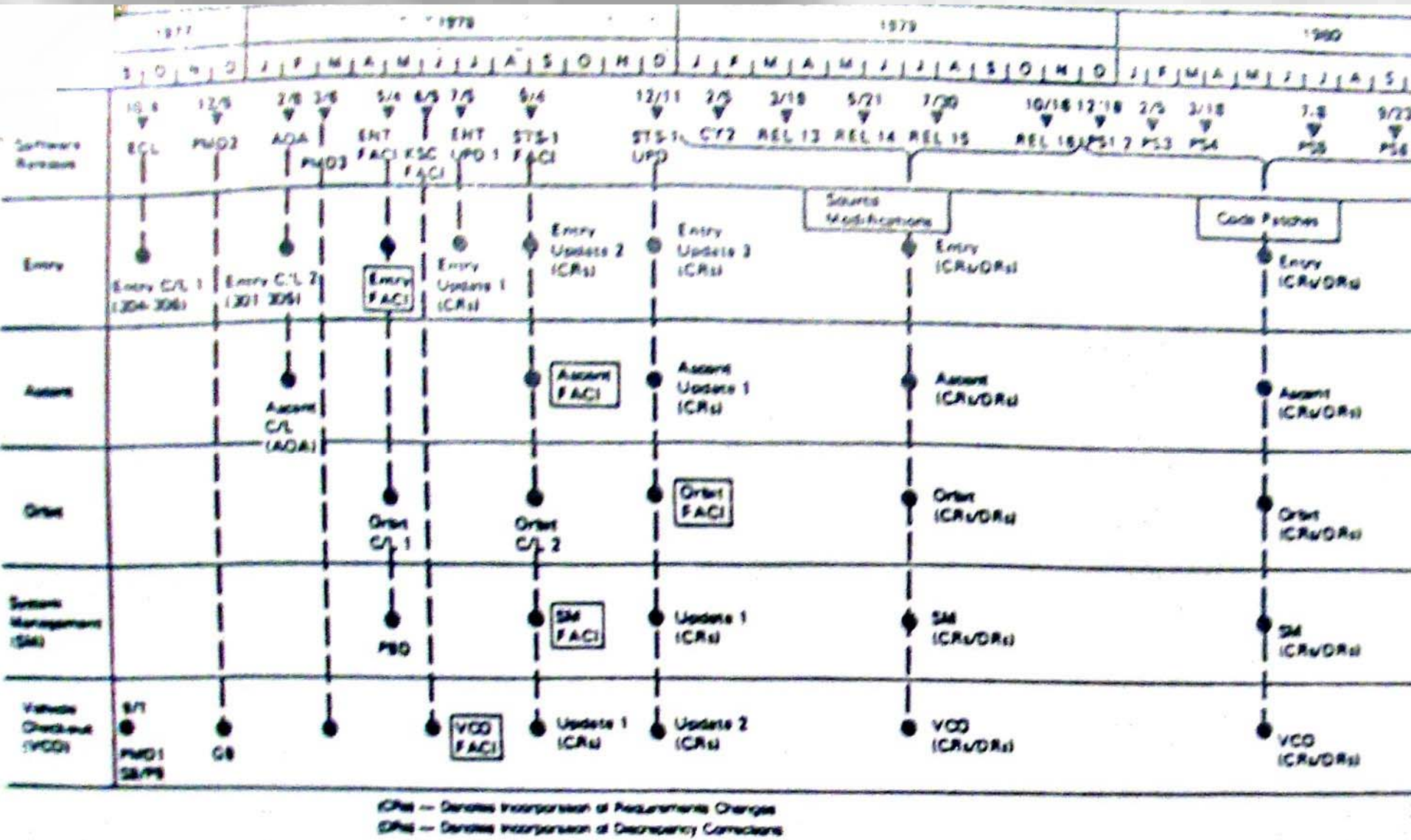


FIGURE 4. Interim Flight Software Releases

process. The key element of this test approach, however, was development of a test management approach that emphasized a hierarchical ordering of development tests that allowed for continual integration of program parts as they were developed and a systematic sequence of evaluation tests on the flight software system (Figure 9).

During the development period, compilation units were added to the master system via the system build process, which was invoked cyclically. Parts of the

1970s: Harlan Mills

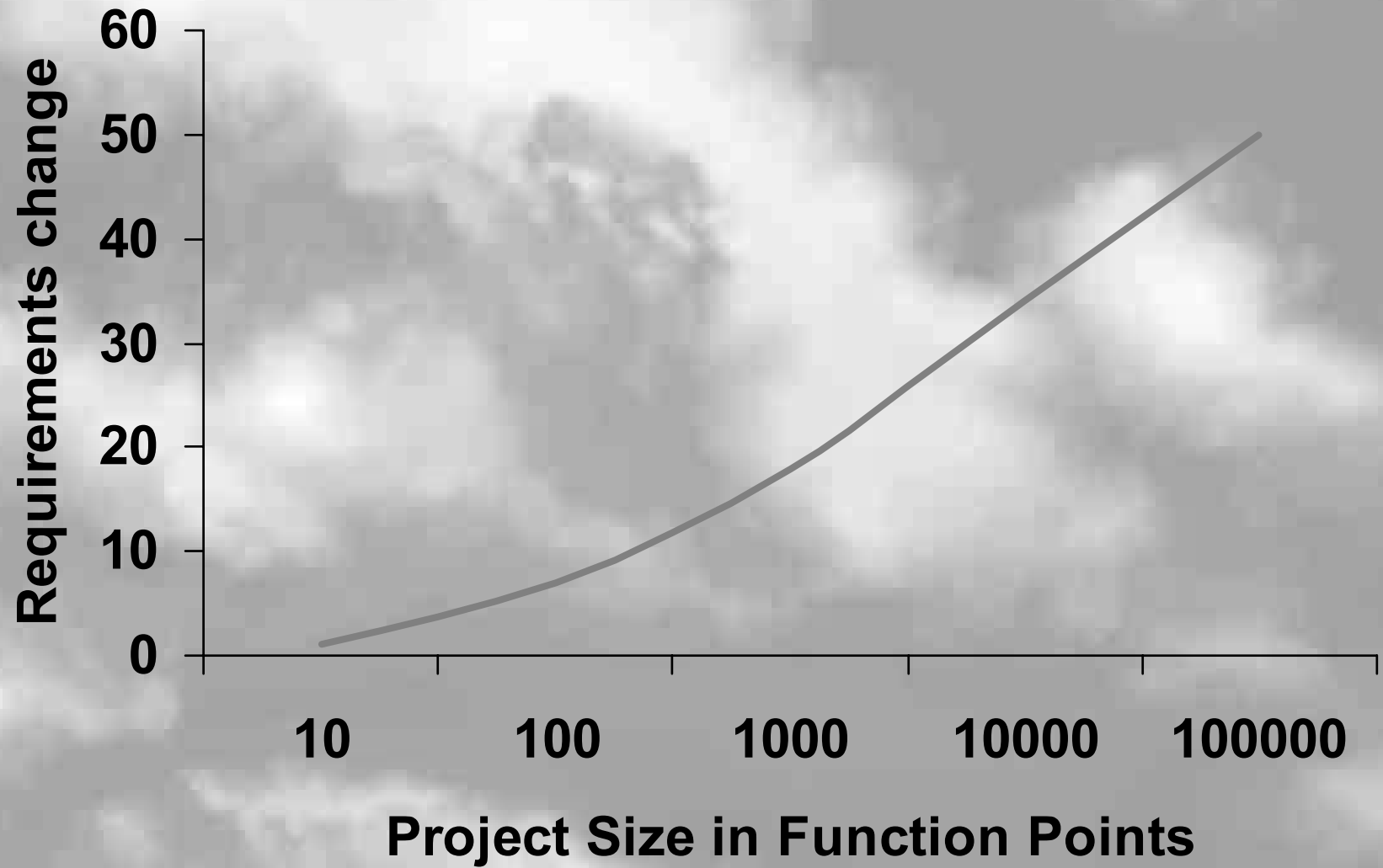
- Harlan Mills, "Software Development," *IEEE Trans. Software Eng.*, Dec. 1976.
  - ***"Software development should be done incrementally, in stages with continuous user participation and replanning and with design-to-cost programming within each stage."***
  - ***"…why do enterprises tolerate the frustrations and difficulties of such [waterfall] development?"***

1980s: Frederick Brooks

- **In his famous 1987 "No Silver Bullet" paper:**
  - **"Nothing in the past decade has so radically changed my own practice, or its effectiveness as [iterative development]."**
- **In his '95 ICSE keynote and in his famous "Mythical Man-Month":**
  - **"The waterfall model is wrong!"**

Evidence

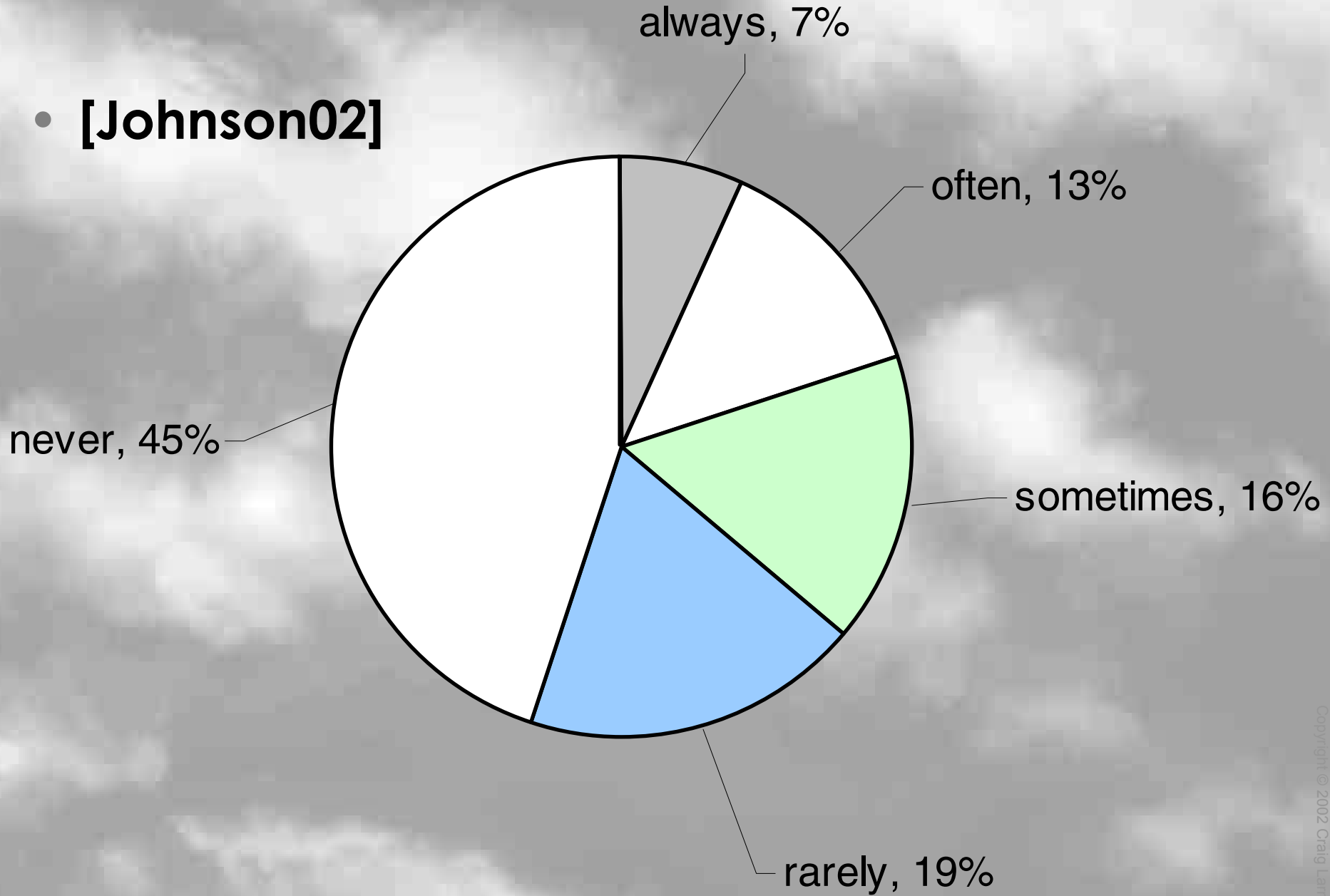
- To repeat: [Jones97] and [BP88]



- Success/failure factors on 1,027 UK projects [Thomas01]
  - Waterfall practices (including detailed up-front requirements and "fixed" schedules)
  - were the single largest contributing factor for failure,
  - being cited in 82% of the projects as the number one problem

- **[Jarzombek99]** – 1995 DoD software project study (of $37 billion USD worth of projects done with waterfall 2167/A)
  - **46% of systems so egregiously did not meet the real needs (although they met the specifications) that they were never used,**
  - another 20% required extensive rework to meet the true needs (rather than the specifications) before they could be used.

- **[Johnson02]**



- **[Johnson98]** study of success/failure factors on 23,000 USA projects: Long waterfall-oriented cycles and infrequent user involvement of the waterfall were correlated with higher failure rates.

- **[MacCormack01]** Two-year study

  "***Now there is proof that the evolutionary approach to software development results in a speedier process and higher-quality products.***"

- Most of the improvement in productivity was related to two factors:
  - iterations with early feedback
  - Daily (or more frequently) integration of all the code, with automated regression testing each build.

- [Standish98] study of 23,000 projects:
  - 2of 5 top success factors

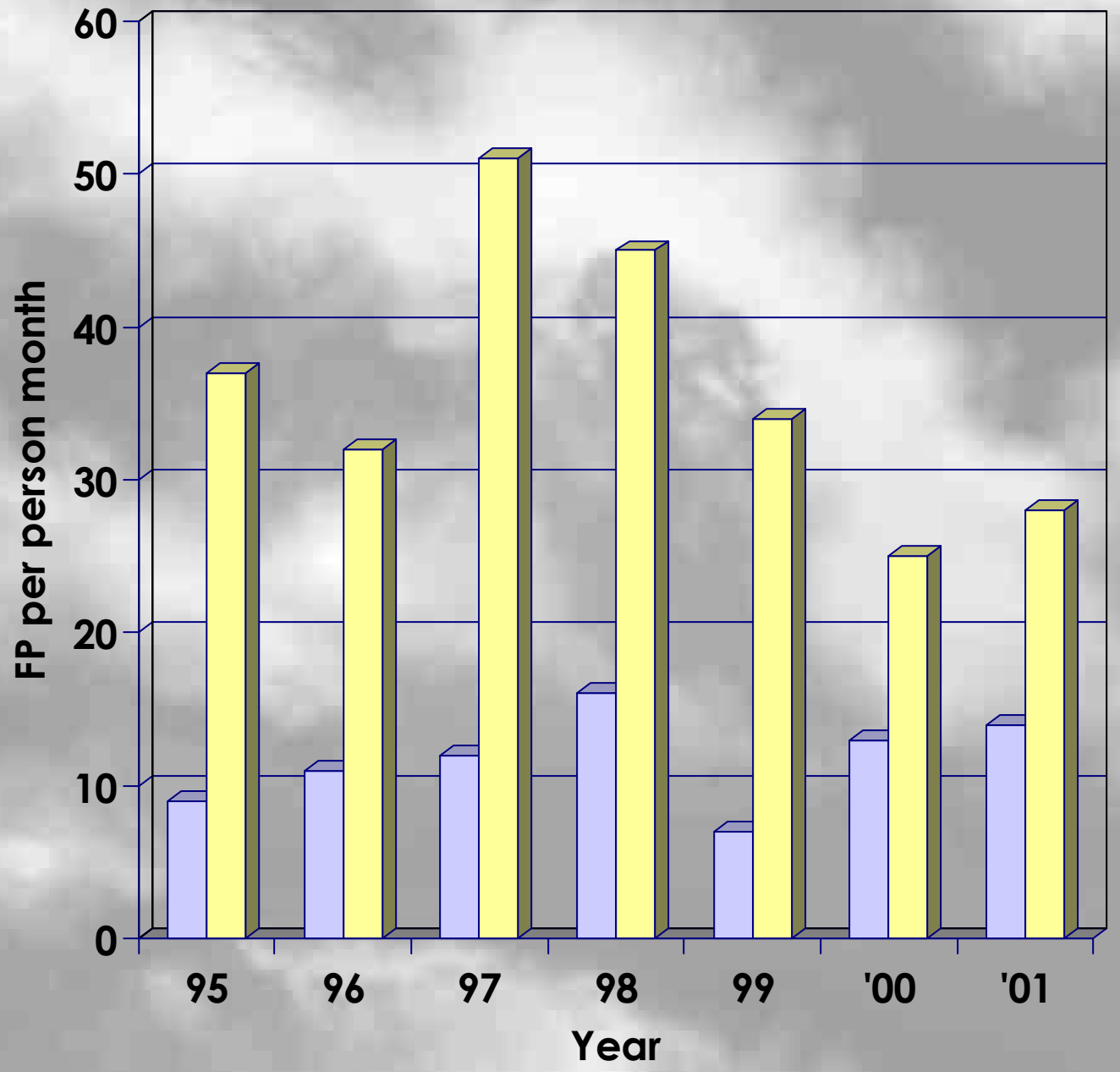
    were strongly associated

    with iterative practices

1. Frequent user involvement
2. Small milestones
3. Clear business objects
4. Experienced PM
5. Executive support

- ***Research also indicates that smaller time frames, with delivery of software components early and often, will increase the success rate. Shorter time frames result in an iterative process of design, prototype, develop, test, and deploy small elements.***

- **[HC96]** Study of hyper-productive development teams. Patterns of success:
  - **Iterative development.**
  - **Simple org structure; fewer roles.**
  - **Architect worked as programmer, especially during early phase**
  - **More direct involvement of developers with other stakeholders.**



Source: Fazackerley, 2003

40,000 person days of effort

7 year study

Large & small projects

- **[Shine03]** Agile method survey.
  88% of organizations cited improved productivity, and 84% improved quality.

  Cost of development, 46% stated no change and 49% stated it was less expensive.

  83% claimed higher satisfaction and 26% claimed "significantly better satisfaction."

- In two studies of 15 teams/projects, research showed (U. Mich., ACM 2000 Conference on Computer Supported Cooperative Work) team ***productivity was double*** over traditional office or cube arrangements.