LUT University

LAND OF THE CURIOUS

LECTURE 3

SOFTWARE PROCESSES

COURSE LECTURE SCHEDULE

Teachers:
Maria
Susan
Sami Hyrynsalmi

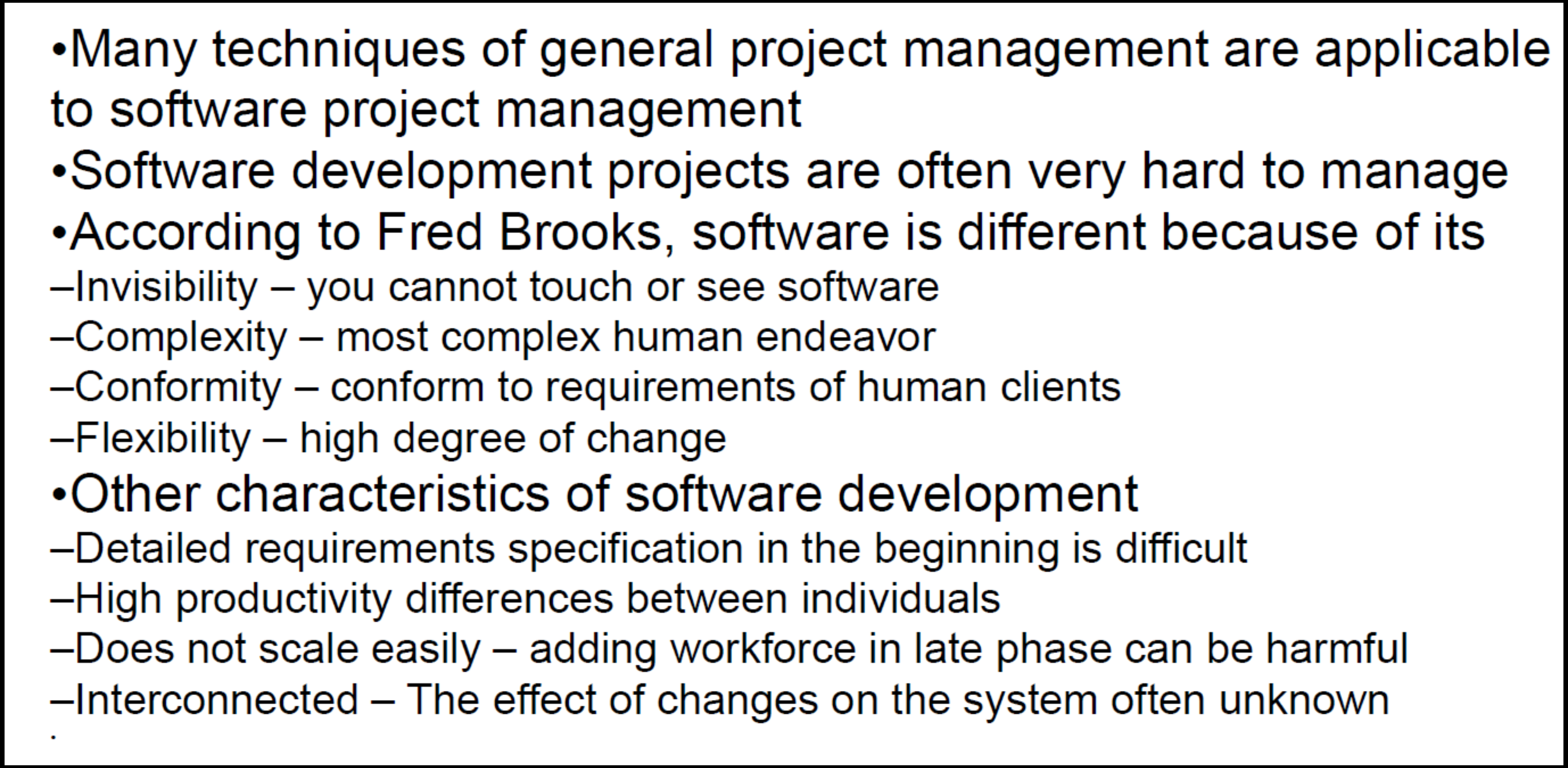
Date	Topic	Book Chapter(s)
Wed 8.9.	Course introduction	
Tue 14.9.	Introduction to Software Engineering	Chapter 1
Tue 21.9.	**Software Processes**	**Chapter 2**
Mon 27.9	Agile Software Engineering	Chapter 3
Tue 5.10.	Requirements Engineering	Chapter 4
Mon 11.10.	Architectural Design	Chapter 6
Wed 20.10.	Modeling and implementation	Chapters 5 & 7
Mon 1.11.	Testing & Quality	Chapters 8 & 24
Mon 8.11.	Software Evolution & Configuration Management	Chapters 9 & 25
Mon 15.11.	Software Project Management	Chapter 22
Mon 22.11.	Software Project Planning	Chapter 23
Mon 29.11.	Global Software Engineering	
Wed 8.12.	Software Business	
Mon 13.12.	Last topics	

GOALS FOR THIS LECTURE:

After this lecture, you know

1. Where does the term “software engineering come from”?
2. Why do we need process models?
3. What kinds software engineering process models exist?

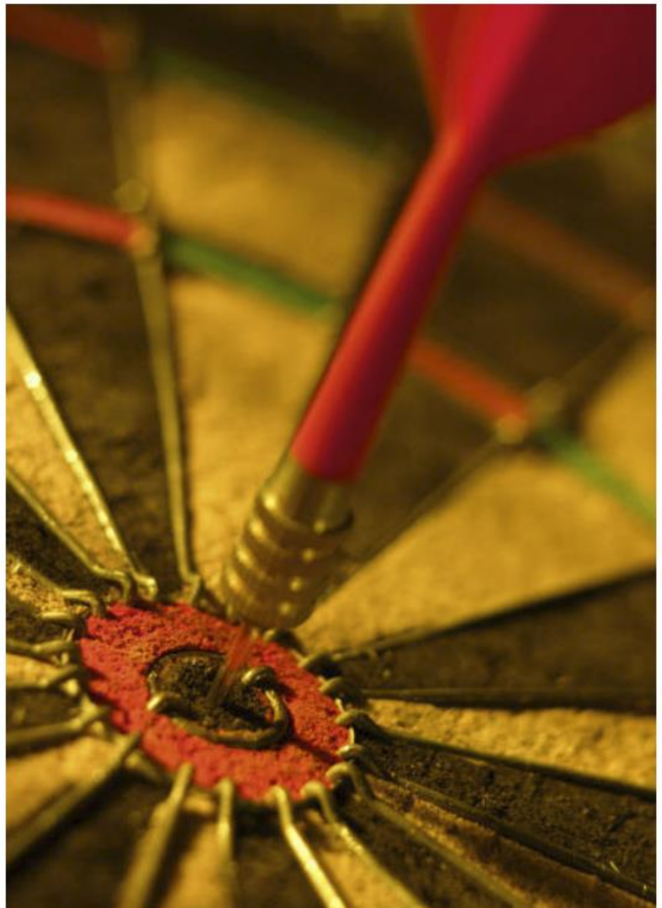
MANAGING SOFTWARE DEVELOPMENT VS. OTHER PROJECTS



- Many techniques of general project management are applicable to software project management
- Software development projects are often very hard to manage
- According to Fred Brooks, software is different because of its
  - Invisibility – you cannot touch or see software
  - Complexity – most complex human endeavor
  - Conformity – conform to requirements of human clients
  - Flexibility – high degree of change
- Other characteristics of software development
  - Detailed requirements specification in the beginning is difficult
  - High productivity differences between individuals
  - Does not scale easily – adding workforce in late phase can be harmful
  - Interconnected – The effect of changes on the system often unknown

SOFTWARE PROJECT SUCCESS RATES

- Standish Group, 2015 (2011) Chaos Report
  - **29% (29%)** Successful (On Time, On Budget, Fully Functional)
  - **52% (49%)** Challenged (Late, Over Budget, and/Or Less than Promised Functionality)
  - **19% (22%)** Failed (Canceled or never used)

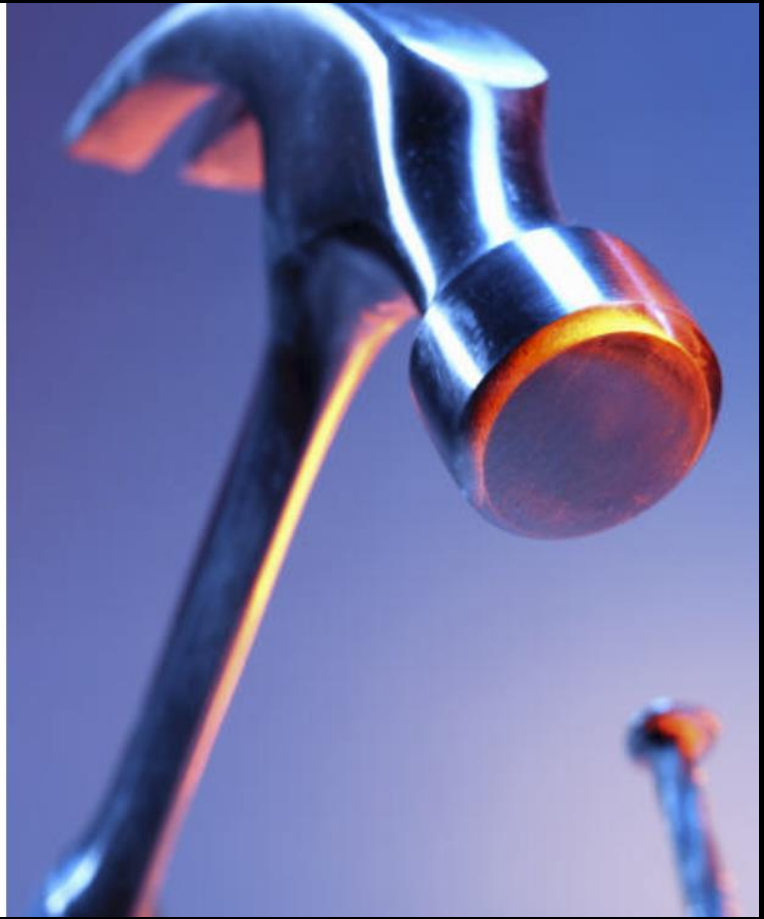


REASONS FOR FAILURE (ACCORDING TO STANDISH GROUP)

- "Most projects failed for lack of skilled project management and executive support"
- "Underestimating project complexity and ignoring changing requirements are basic reasons why projects fail"
- "The problem – and the solution – lay in people and processes"

RECIPE FOR SUCCESS

- Smaller project size and shorter duration
- More manageable
- “Growing”, instead of “developing”, software engages the users earlier and confers ownership.
- -> Iterative and interactive process



LECTURE 3

SOFTWARE PROCESSES

“SOFTWARE CRISIS” IN 1960’S

- Increase of computing power
- Programmers not able to utilize effectively
- Individual approaches were unable to scale up to larger and more complex systems
- Problems:
  - Running over budget
  - Running over time
  - Inefficient software
  - Low quality
  - Not meeting requirements
  - Unmanageable projects
  - Total failure to deliver

TERM: “SOFTWARE ENGINEERING”

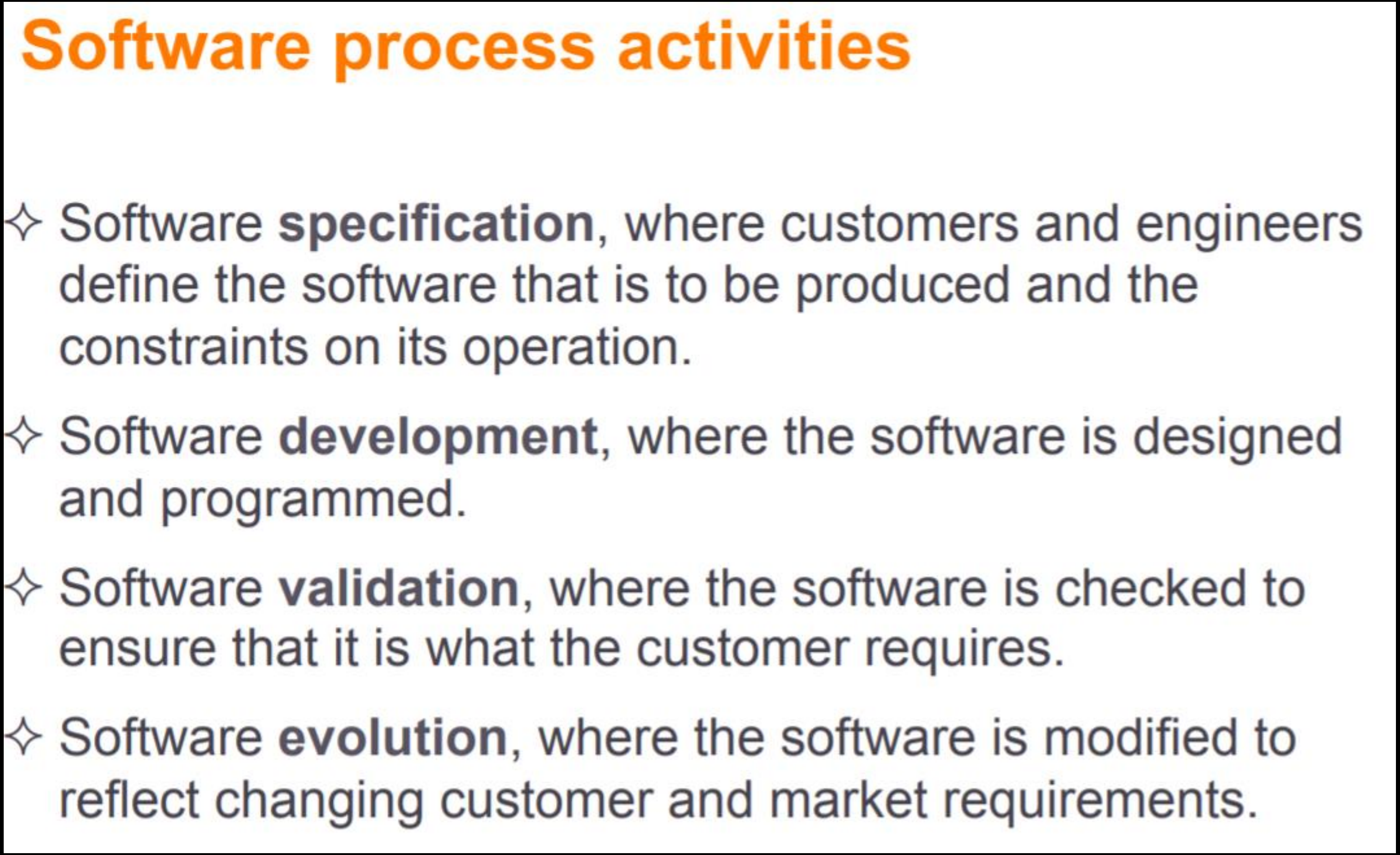
- NATO Software Engineering Conferences 1968 & 1969
- Invited around 50 international experts from all areas concerned with software problems — computer manufacturers, universities, software houses, computer users, etc.
- Agreed on defining best practices for software engineering, “how software should be developed”
- http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF
- ”Real” engineering discipline -> “Software Engineering”

AFTER THE CRISIS…

- Between the 1970s and 1980s, a variety of software engineering techniques and methods were developed together with tools and standardized notations…
- • … this trend is still continuing
- • … however, after over 50 years, developing software is still a challenging endeavour that can fail.

DEFINITIONS

- **Process** according to IEEE:
  - a sequence of steps performed for a given purpose
- **Software process** according to the CMM:
  - a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products
- **Process model**:
  - an abstract depiction of a process from some perspective
- **Software development life-cycle model**:
  - A sequence of steps through which the development progresses



Software process activities

- ✧ Software **specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.
- ✧ Software **development**, where the software is designed and programmed.
- ✧ Software **validation**, where the software is checked to ensure that it is what the customer requires.
- ✧ Software **evolution**, where the software is modified to reflect changing customer and market requirements.

PROCESS MODELING OBJECTIVES

Software Process Models: “Order all or some of the basic software development activities in various ways”

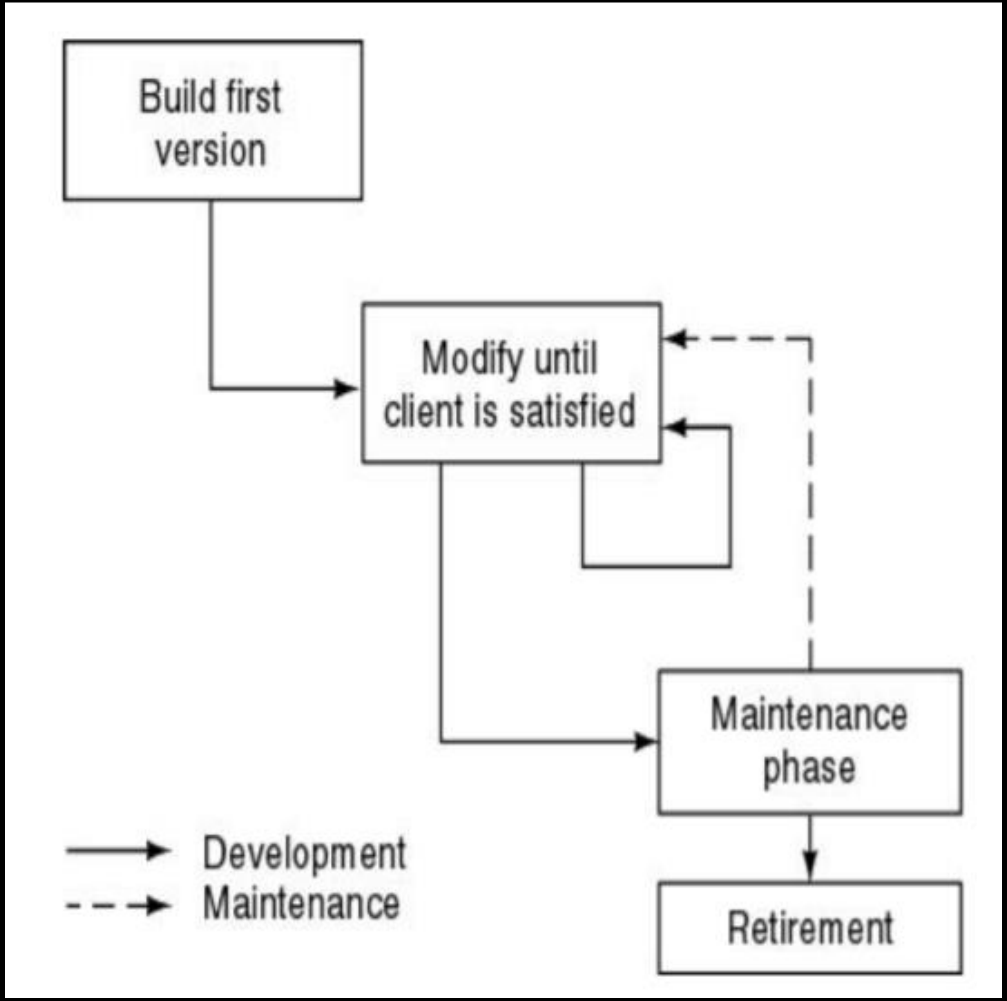
- Enable understanding and communication about the process
- Harmonize and standardize work at the project level
- Support project planning and management
- Gain better overall control of the projects in the process
- Improve the process
- Facilitate process reuse

SOFTWARE PROCESS (LIFE-CYCLE) MODELS

- Build-and-Fix
- Waterfall
- Iterative and Incremental
- Prototyping
- ...

BUILD-AND-FIX

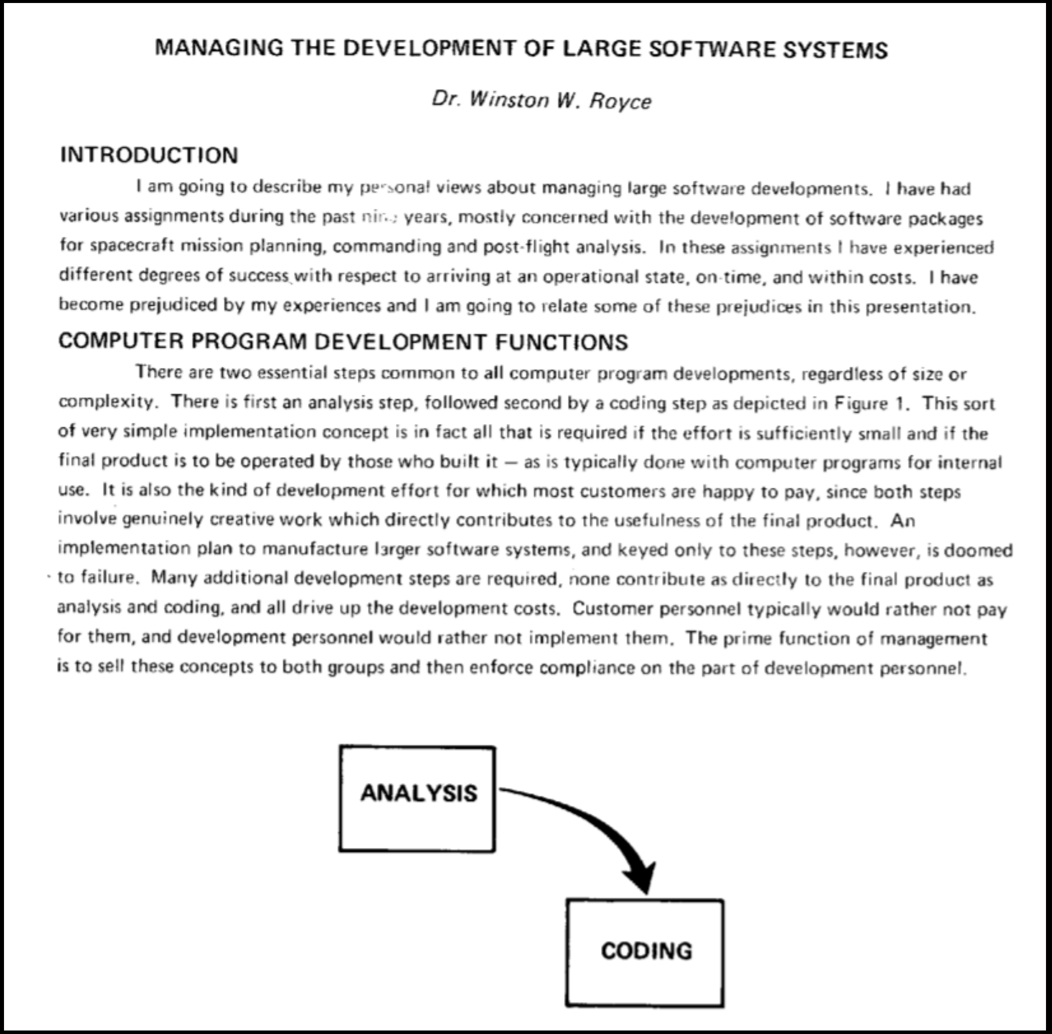
- Problems:
  - No specifications
  - No design
  - Lack of visibility
  - Easily leads to poorly structured systems
  - Totally unsatisfactory
  - Need life-cycle model



WATERFALL

- ”Classical model” for software development
  - 1970 Winston Royce
- Your reading material:

Royce, W. (1970) Managing The Development of Large Software Systems. Reprinted from Proceedings, IEEE WESCON, August 1970, pages 1-9.



MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS

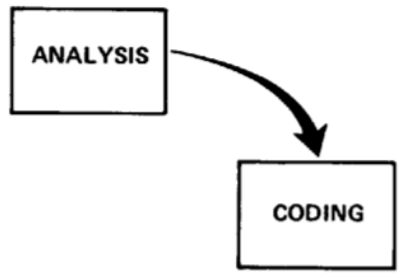
Dr. Winston W. Royce

INTRODUCTION

I am going to describe my personal views about managing large software developments. I have had various assignments during the past nine years, mostly concerned with the development of software packages for spacecraft mission planning, commanding and post-flight analysis. In these assignments I have experienced different degrees of success with respect to arriving at an operational state, on-time, and within costs. I have become prejudiced by my experiences and I am going to relate some of these prejudices in this presentation.

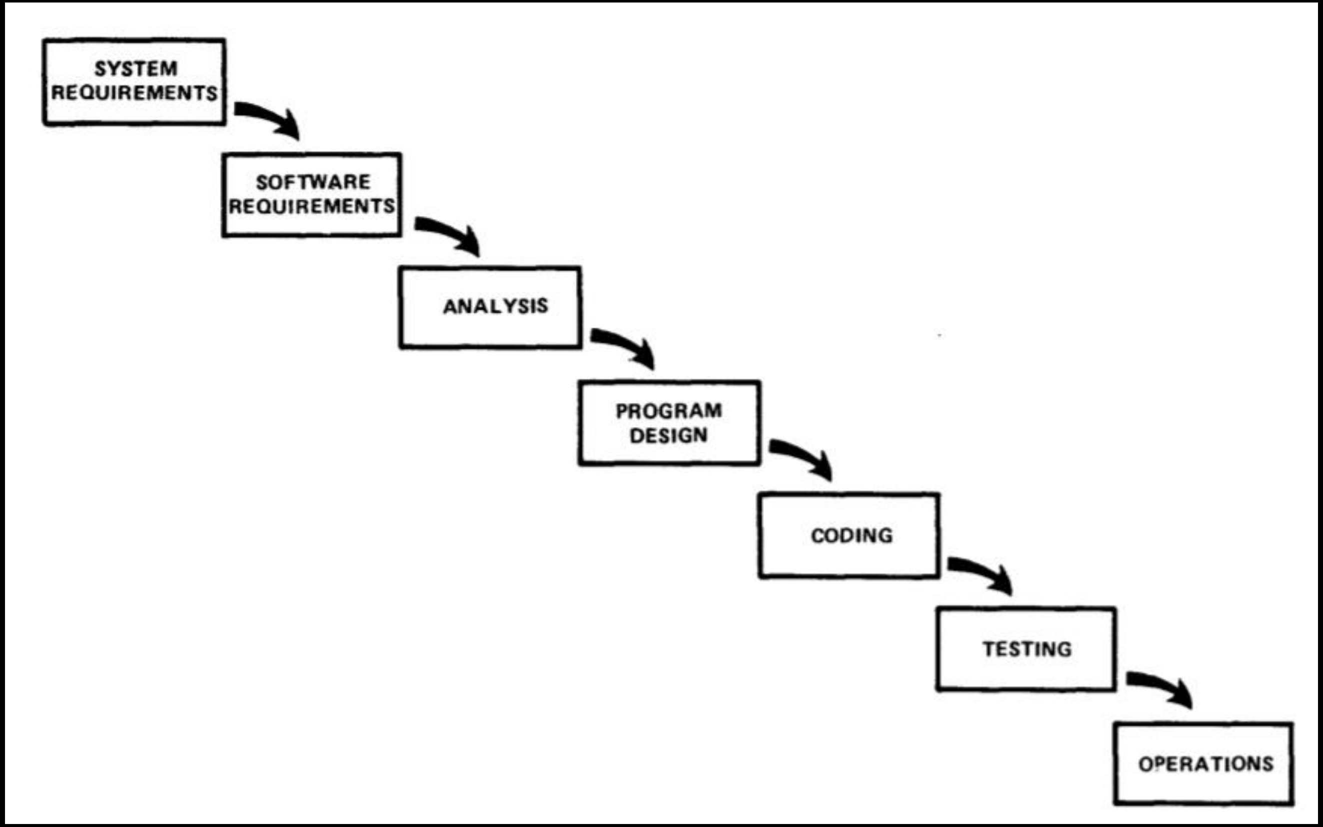
COMPUTER PROGRAM DEVELOPMENT FUNCTIONS

There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed second by a coding step as depicted in Figure 1. This sort of very simple implementation concept is in fact all that is required if the effort is sufficiently small and if the final product is to be operated by those who built it – as is typically done with computer programs for internal use. It is also the kind of development effort for which most customers are happy to pay, since both steps involve genuinely creative work which directly contributes to the usefulness of the final product. An implementation plan to manufacture larger software systems, and keyed only to these steps, however, is doomed to failure. Many additional development steps are required, none contribute as directly to the final product as analysis and coding, and all drive up the development costs. Customer personnel typically would rather not pay for them, and development personnel would rather not implement them. The prime function of management is to sell these concepts to both groups and then enforce compliance on the part of development personnel.



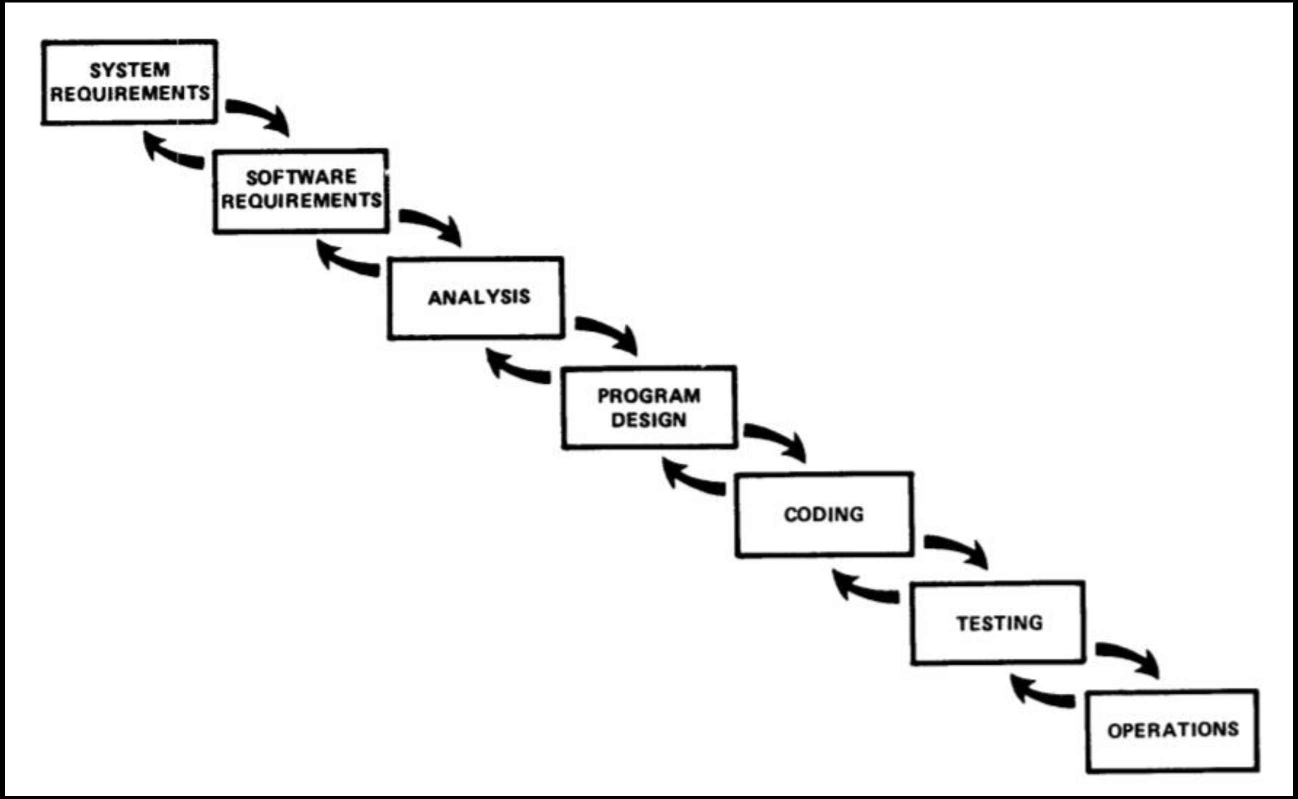
WATERFALL MODEL

- From Royce article:
- “water falls”



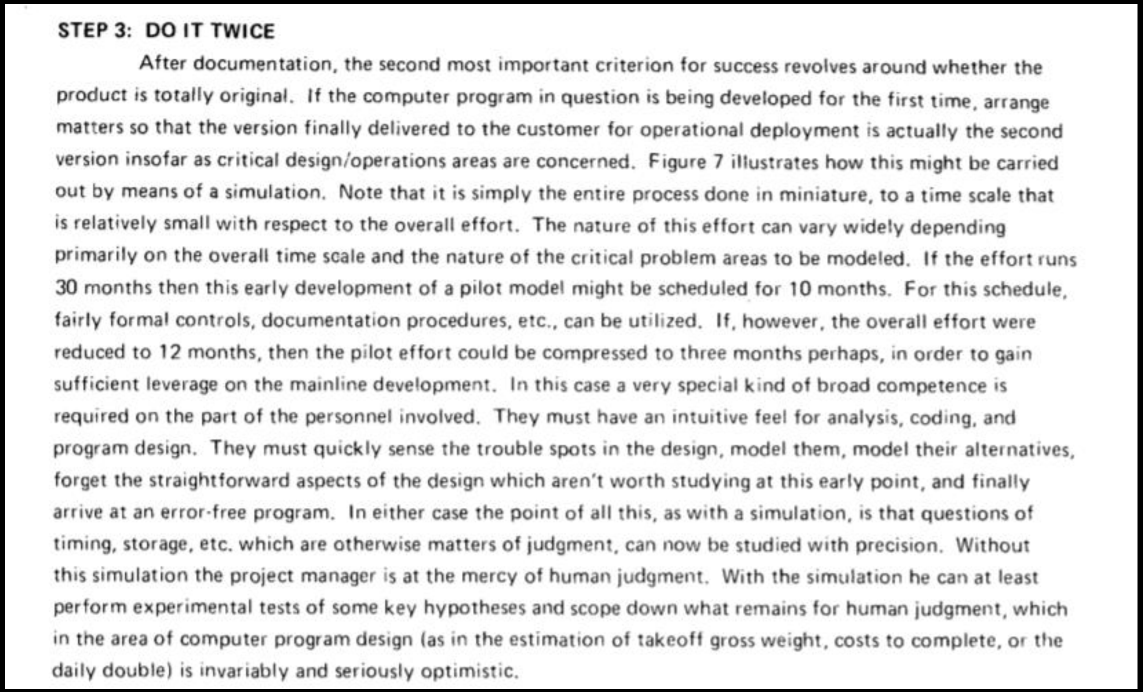
WATERFALL MODEL

- From Royce article:
- “Salmon fall”



WATERFALL MODEL

- From Royce article: “Do it twice”
- *“If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned.”*
- a 30-month project might have a 10-month pilot model



STEP 3: DO IT TWICE

After documentation, the second most important criterion for success revolves around whether the product is totally original. If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned. Figure 7 illustrates how this might be carried out by means of a simulation. Note that it is simply the entire process done in miniature, to a time scale that is relatively small with respect to the overall effort. The nature of this effort can vary widely depending primarily on the overall time scale and the nature of the critical problem areas to be modeled. If the effort runs 30 months then this early development of a pilot model might be scheduled for 10 months. For this schedule, fairly formal controls, documentation procedures, etc., can be utilized. If, however, the overall effort were reduced to 12 months, then the pilot effort could be compressed to three months perhaps, in order to gain sufficient leverage on the mainline development. In this case a very special kind of broad competence is required on the part of the personnel involved. They must have an intuitive feel for analysis, coding, and program design. They must quickly sense the trouble spots in the design, model them, model their alternatives, forget the straightforward aspects of the design which aren't worth studying at this early point, and finally arrive at an error-free program. In either case the point of all this, as with a simulation, is that questions of timing, storage, etc. which are otherwise matters of judgment, can now be studied with precision. Without this simulation the project manager is at the mercy of human judgment. With the simulation he can at least perform experimental tests of some key hypotheses and scope down what remains for human judgment, which in the area of computer program design (as in the estimation of takeoff gross weight, costs to complete, or the daily double) is invariably and seriously optimistic.

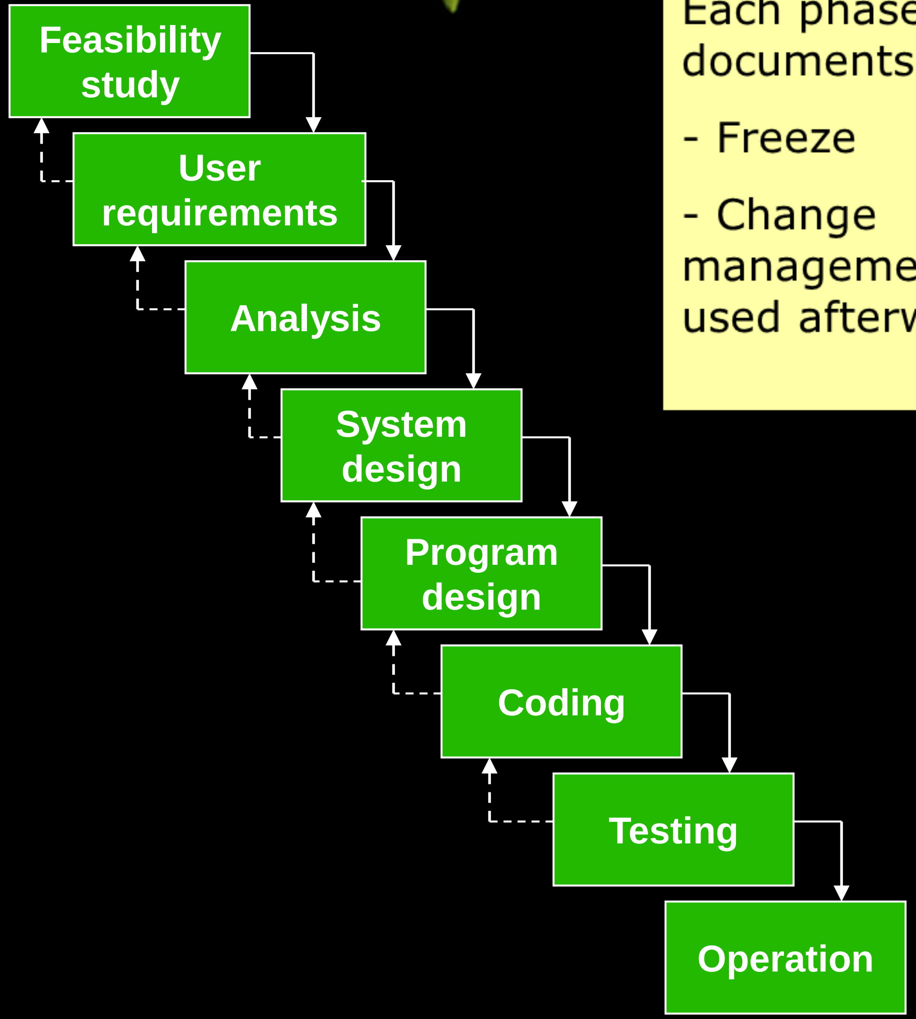
ROYCE: WATERFALL VS. ITERATIVE AND INCREMENTAL DEVELOPMENT (IID)

- His son Walker Royce commented:

“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 rest of his paper describes [iterative practices] within the context of the 60s/70s government-contracting models (a serious set of constraints).”

WATERFALL

- The whole application is developed in one go
- Planning and control
- Document driven
- Limited scope for iteration
- Different people in different phases



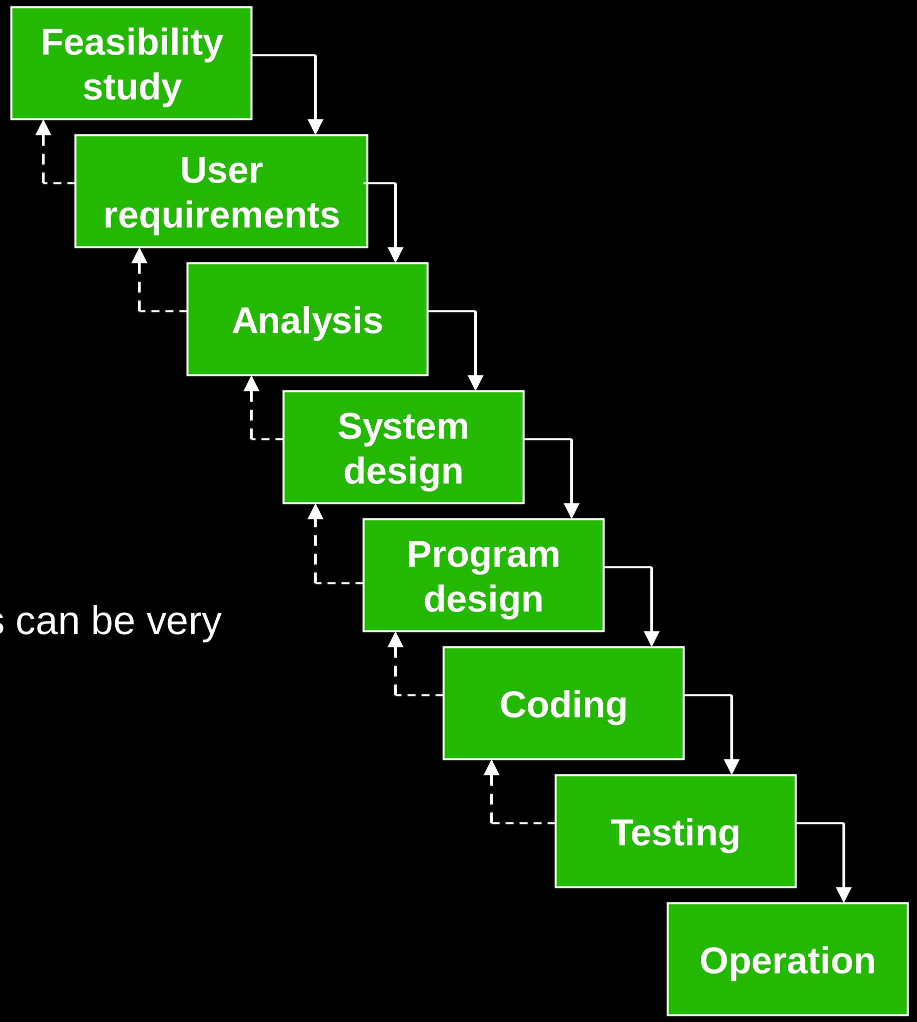
Each phase produces documents

- Freeze

- Change management process used afterwards

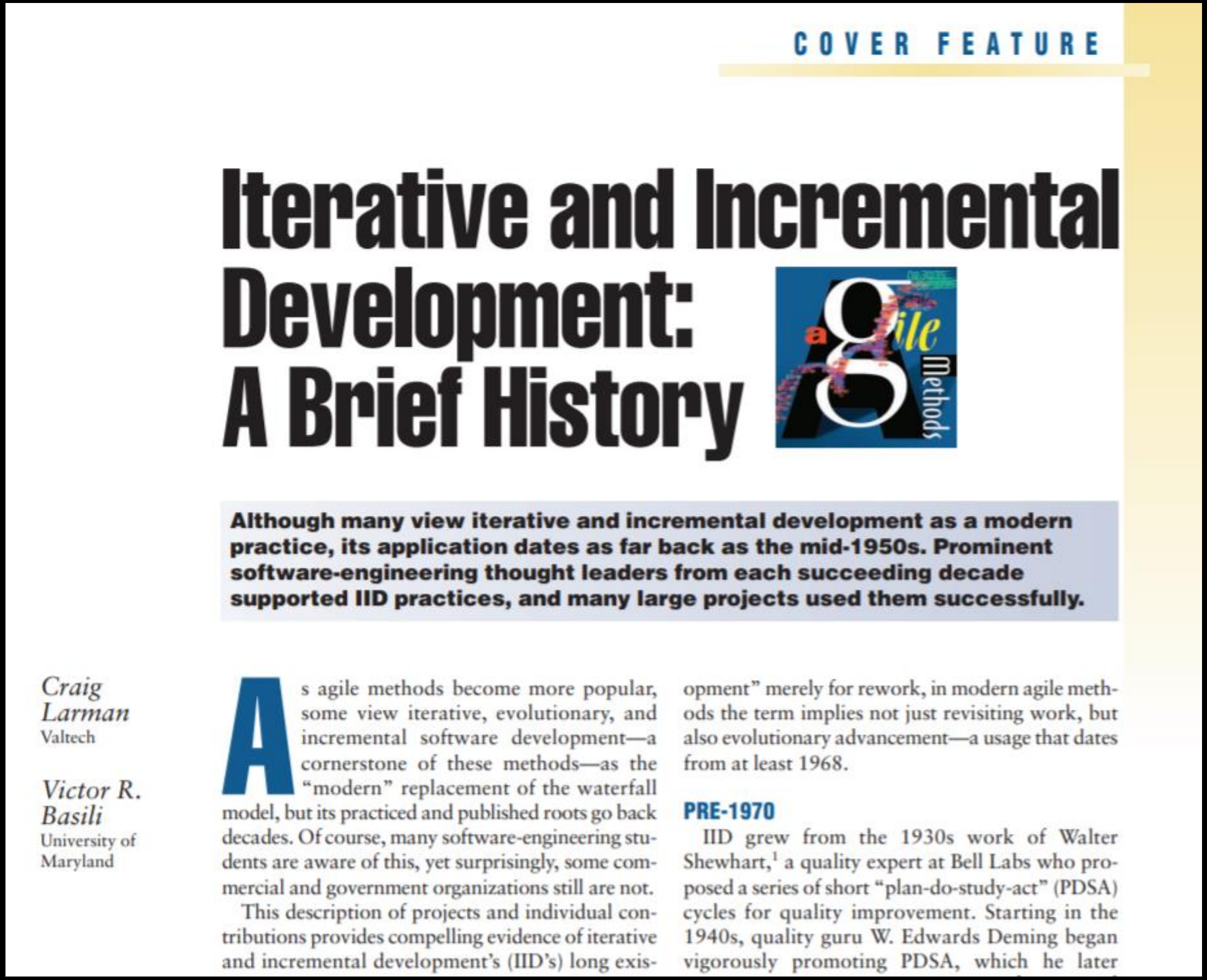
WATERFALL

- Strengths
  - Easily manageable process (manager’s favorite?)
  - Probably the most effective model, if you know the requirements
  - Extensive documentation
- Weaknesses
  - Inflexible partitioning of the project into distinct stages
  - Difficult to respond to changing customer requirements
  - Feedback on system performance available very late and changes can be very expensive
- Applicability
  - Appropriate when the requirements are well-understood
  - Short, clearly definable projects (e.g. maintenance)
  - Very large, complex system development, that requires extensive documentation, safety critical systems



ITERATIVE AND INCREMENTAL DEVELOPMENT (IID)

- The history of IID from 1950's
- Larman, C. and Basili, V. R. (2003) 'Iterative and incremental development: A brief history', *Computer*, 36(6), pp. 47–56. DOI: 10.1109/MC.2003.1204375.



COVER FEATURE

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 Maryland

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 students are aware of this, yet surprisingly, some commercial and government organizations still are not.

This description of projects and individual contributions provides compelling evidence of iterative and incremental development's (IID's) long exis-

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 Shewhart,[1] a quality expert at Bell Labs who proposed a series of short "plan-do-study-act" (PDSA) cycles for quality improvement. Starting in the 1940s, quality guru W. Edwards Deming began vigorously promoting PDSA, which he later

ITERATIVE AND INCREMENTAL DEVELOPMENT

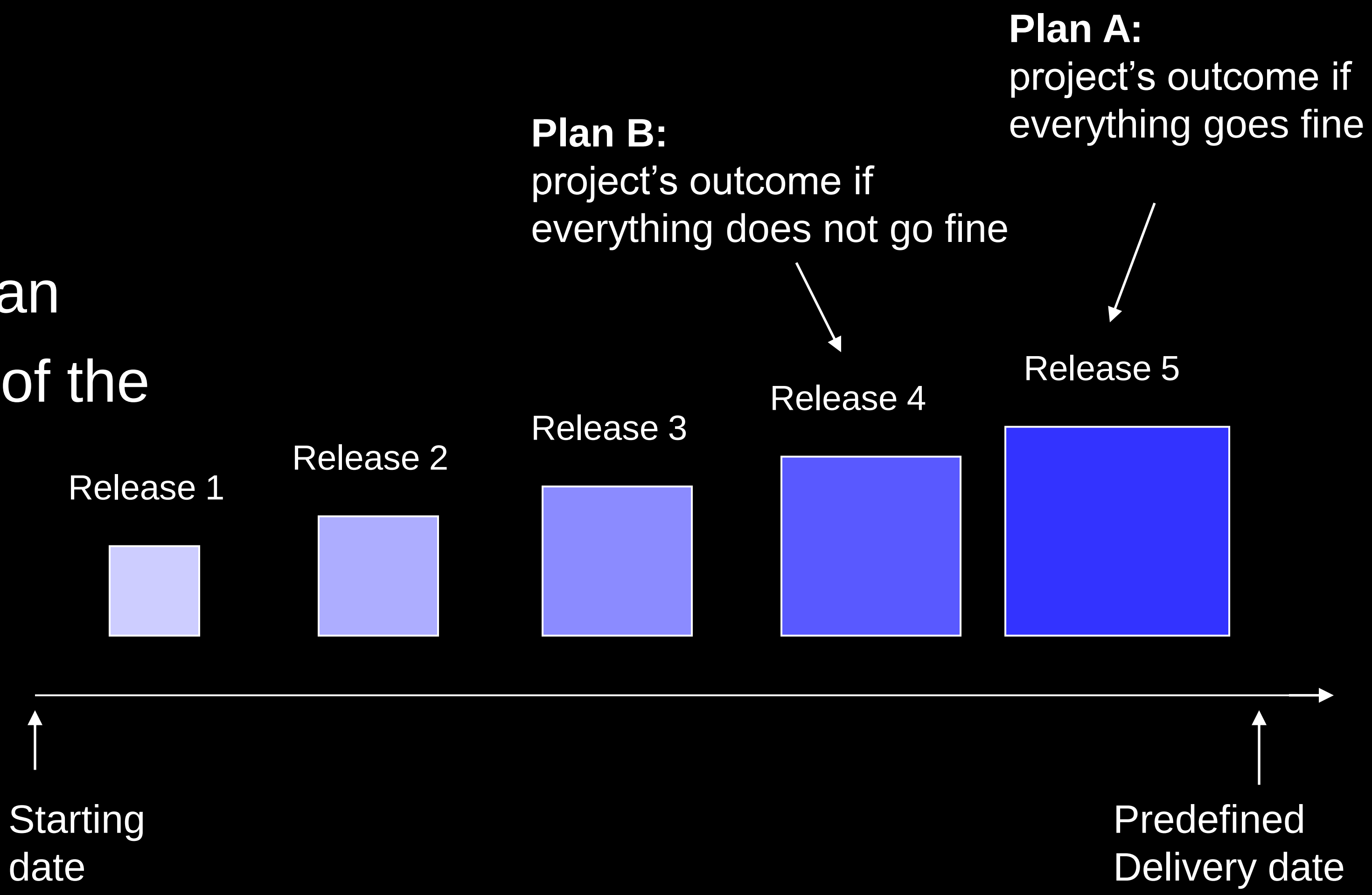
- The concept of growing a system via iterations: iterative and incremental development (IID)
  - Divide the project into increments
  - Each increment adds functionality
  - Each iteration is a self-contained **mini project** composed of activities such as requirements analysis, design, programming and test
  - At the end of the iteration an iteration release: a stable integrated and tested partially complete system
  - Most releases internal, final iteration release is the complete product
- Prioritize user requirements
  - **MOSCOW priorities: must, should, could, want**
  - High-priority requirements into early increments
  - **Freeze requirements during each increment**

TIMEBOXING

- Fixing the iteration end date and not allowing it to change
  - 1-6 weeks is normal length for a timeboxed iteration
  - 3-6 month is usually too long and misses the point and value
  - Shorter steps have lower complexity and risk, better feedback, and higher productivity and success rates
- Involve developers and the customer in planning
- The scope is reduced, rather than slip the iteration end date
  - lower priority requests back on the wish list
- Should not be used to pressure developers to work long hours to meet the deadline -> If normal pace of work is insufficient do less

TIMEBOXING

- Prioritize features
- Make a release plan
- Plan the contents of the first timebox
- Dates fixed
- Content flexible



IID ADVANTAGES

- Customer value can be delivered in the end of each increment so system functionality is available earlier
- Final product better matches true customer needs
- Early increments act as a prototype to help
  - elicit requirements for later increments
  - get feedback on system performance
- Lower risk of overall project failure
- Smaller sub-projects are easier to control and manage
  - A meaningful progress indicator: **tested software**
- The highest priority system services tend to receive the most testing
- Job satisfaction is increased for developers who can see early the results of their work

IID DISADVANTAGES

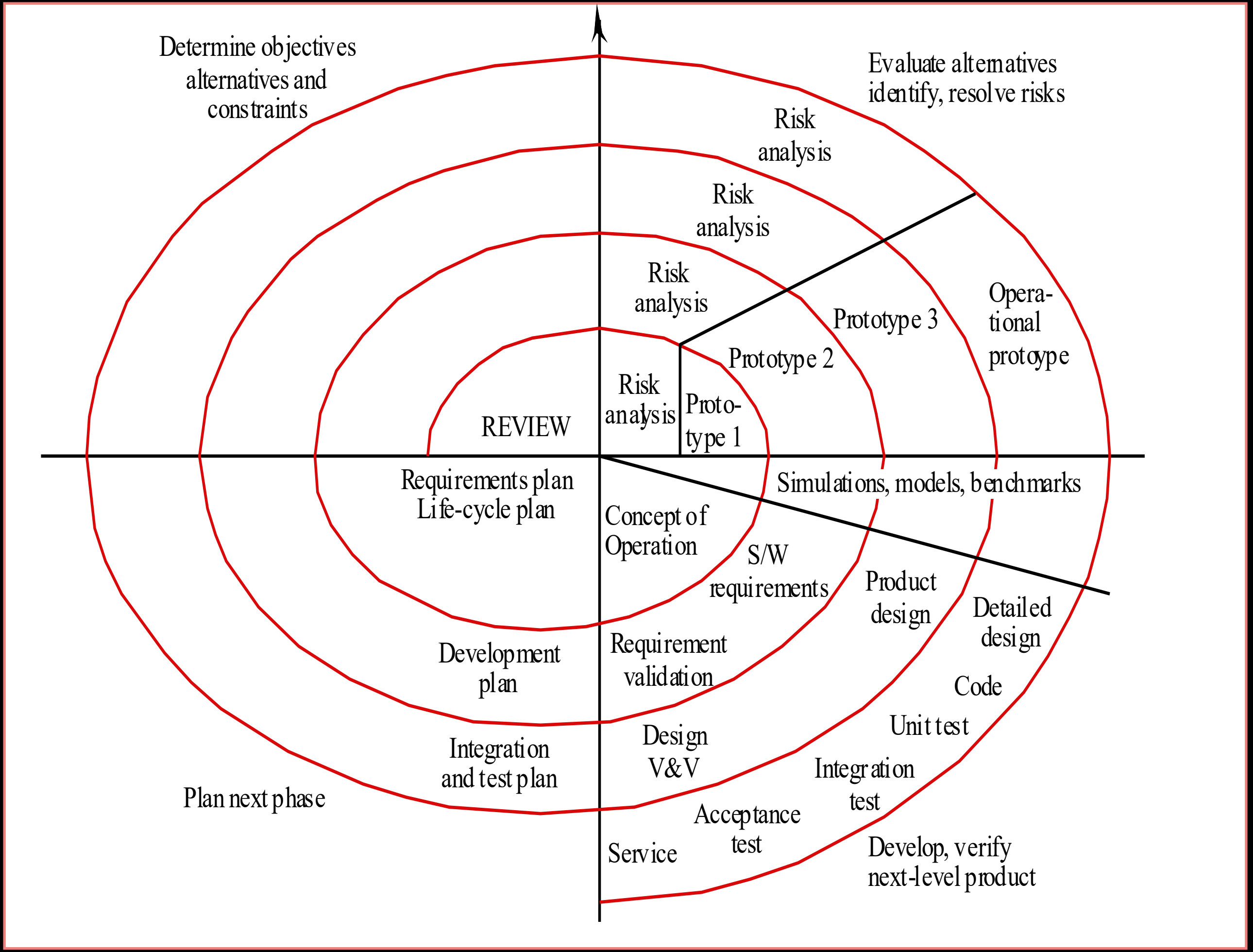
- Can be harder to plan and control than Waterfall development
- Can be more expensive than Waterfall development
- May require more experienced staff
- System architecture must be adaptive to change
- Later increments may require modifications to earlier increments
- Software project contracts are still mostly drawn according to the Waterfall model and all changes cause renegotiations

SOFTWARE PROTOTYPING

- Prototype is a working model of one or more aspects of the projected system -> tests assumptions
- Classification
  - Throw-away: used to test out ideas and then discarded
  - Evolutionary: modified until it can become the operational system
- Strengths
  - Improved user involvement, clarification of requirements
  - Possibility to test a new technical solution, learning by doing
- Weaknesses
  - Users can misunderstand the role of the prototype
  - Additional expense

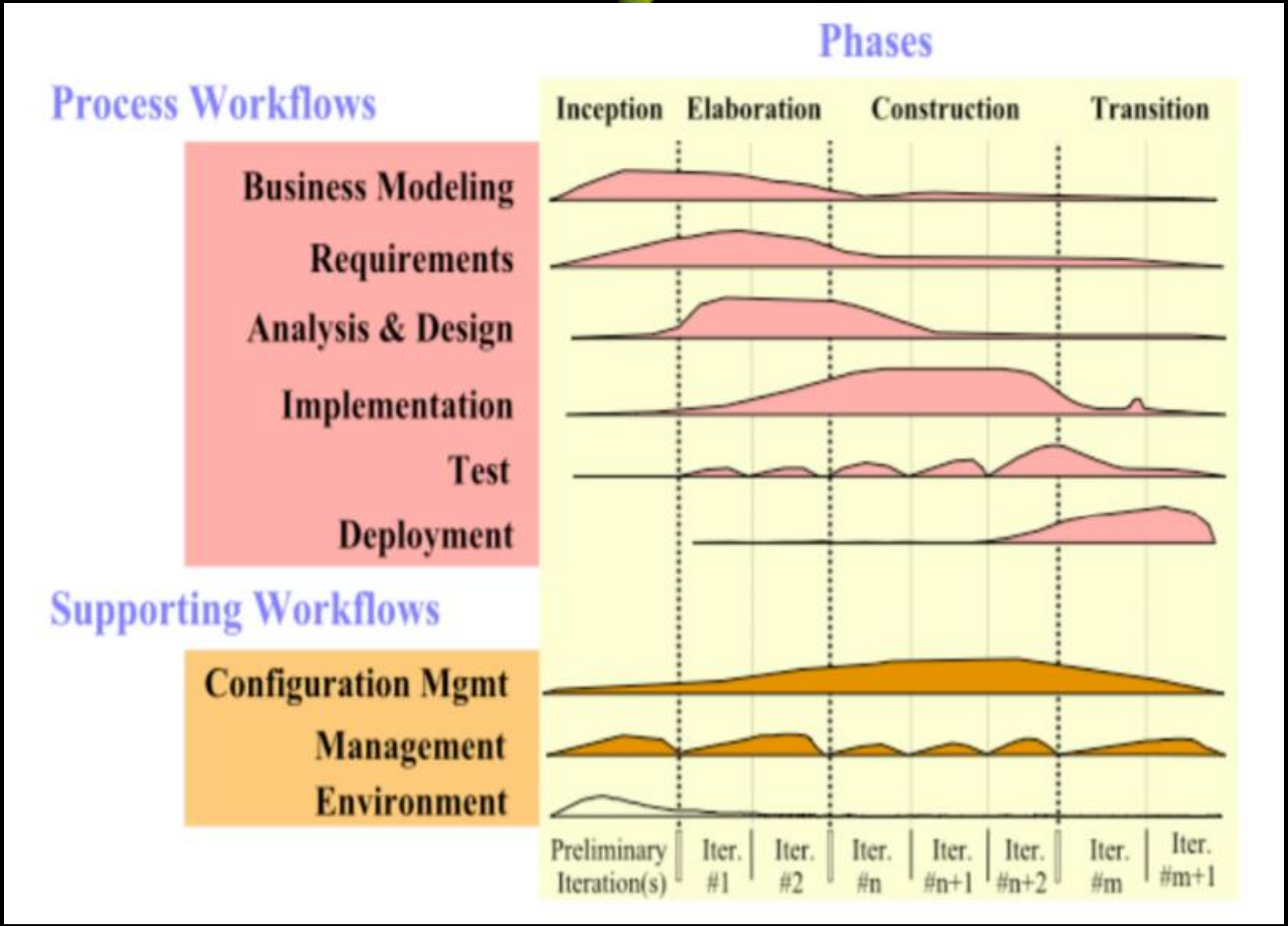
SPIRAL MODEL

- A meta-model -> other models can be derived
- Risk management is central
- Problems
  - Hard to match to contract software
    - Not enough flexibility and freedom in contract mechanisms
  - Great deal of reliance on risk-assessment expertise



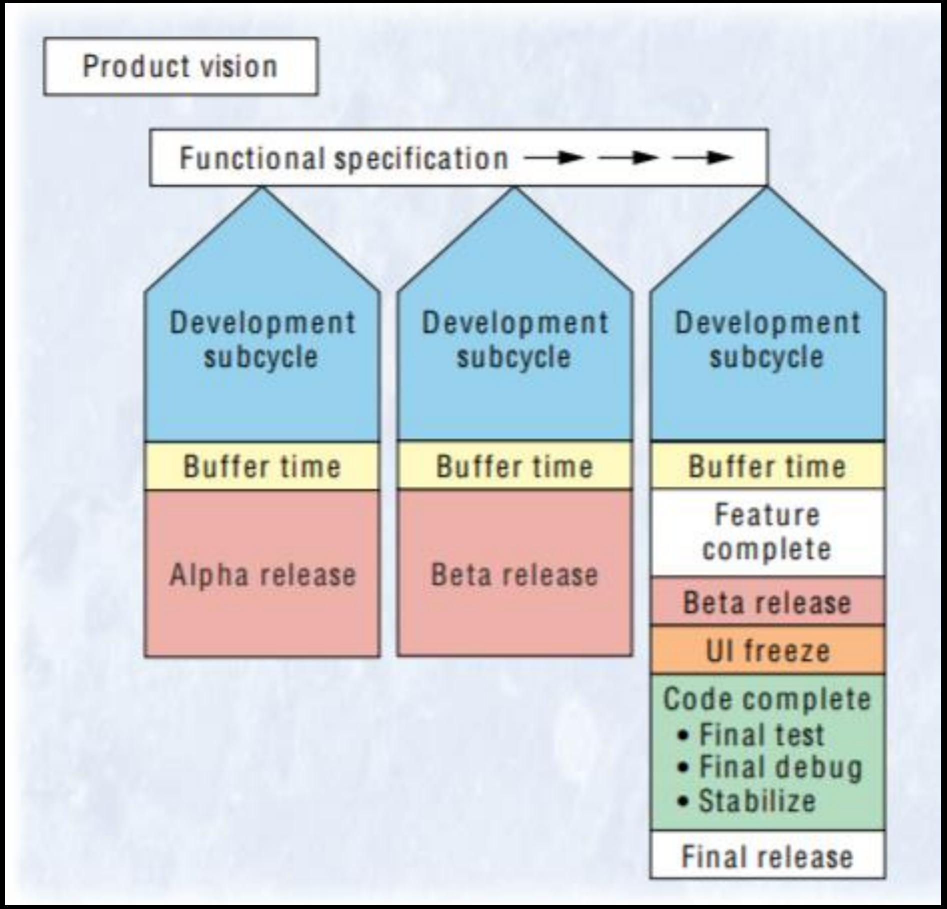
RATIONAL UNIFIED PROCESS (RUP)

- Philippe Kruchten
- Rational
- IBM



SYNCHRONIZE-AND-STABILIZE MODEL

- Microsoft’s life-cycle model
- Requirements analysis—interview potential customers
- Draw up specifications
- Divide project into 3 or 4 builds
- Each build is carried out by small teams working in parallel
- A development subcycle is repeated several times: design, code, usability test, test, daily builds, debugging, integration, and stabilization



PROCESS CHOICE

- There is no uniformly applicable process which should be standardized within an organization if there are, e.g., different types of products and/or markets
  - High costs may occur if you force an inappropriate process on a development team
- However, one organization cannot have a large number of different processes in use
  - Sometimes you don’t have a possibility to choose
  - A chosen process model can be tailored to suite the project

ESSAY 2: COMPARISON OF WATERFALL AND ITERATIVE AND INCREMENTAL MODELS

- Write a 750 word essay **describing and comparing the Waterfall and Iterative and Incremental (IID) software development models**. Describe the principles of each model, its strengths, weaknesses, and applicability. In particular, give practical examples of implementations of the iterative and incremental model.

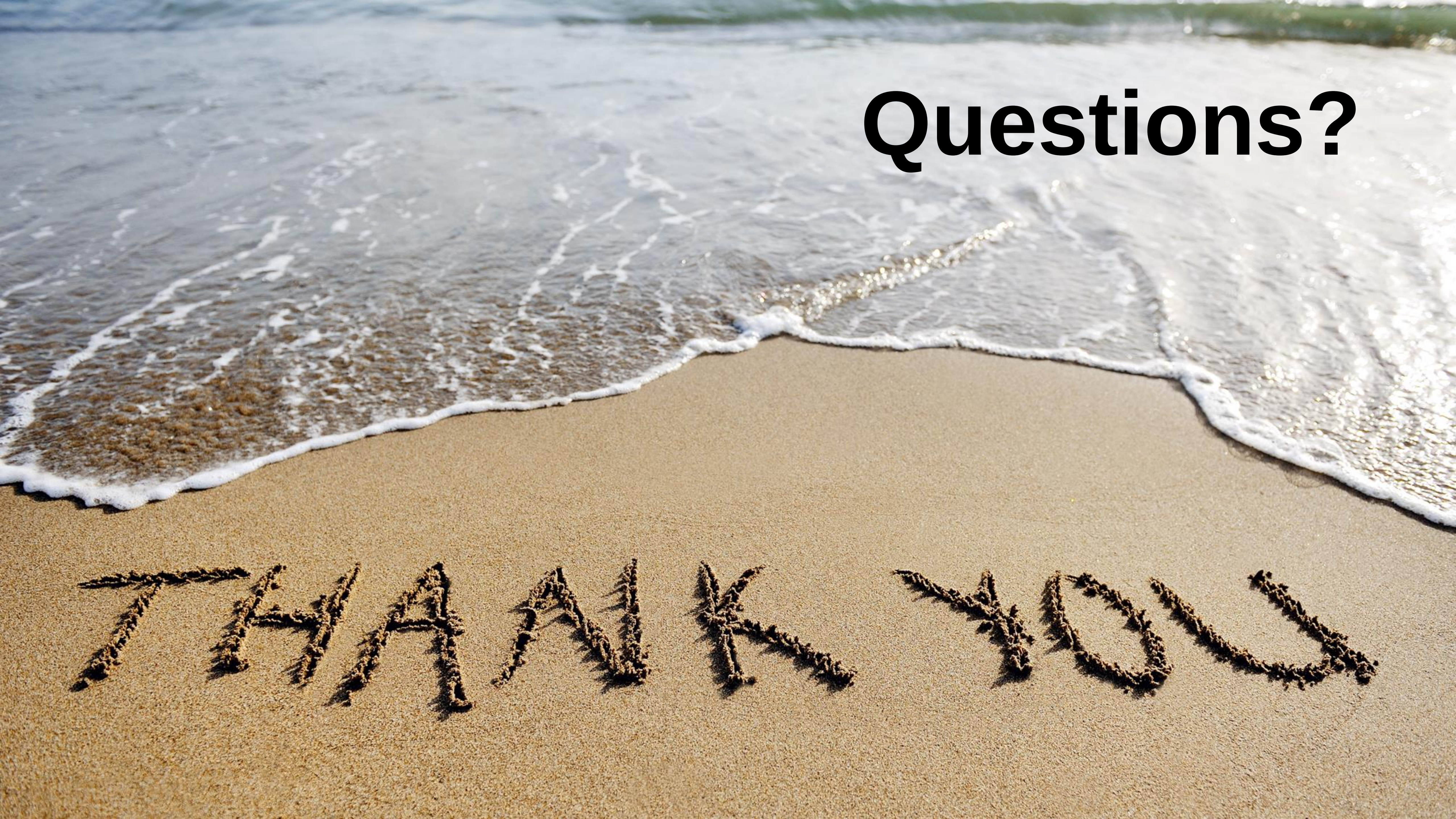
- Use the reading materials (textbook, articles, slides). You may use additional materials (that you have found by yourself), as well.

- Use referencing according to the guidelines, and try to aim for the 750-word limit. Too short or overly long essays will receive a lower score. For the highest score, your essay should be within the 560-940 word range (within 25%).
  - Reference list is NOT included in the word limit

- Remember the importance of referencing, and of avoiding plagiarism.

ESSAY WILL BE GRADED ON:

- Having all the expected elements: description of the models, strengths, weaknesses, applicability, examples.
- The quality and understanding shown in the descriptions and comparison of the models.
- The length, as described above
- The correct use of referencing. Note that absense of references fails the essay completely, as you are required to reference)
- An overall assessment of the essay quality



Questions?

LUT University