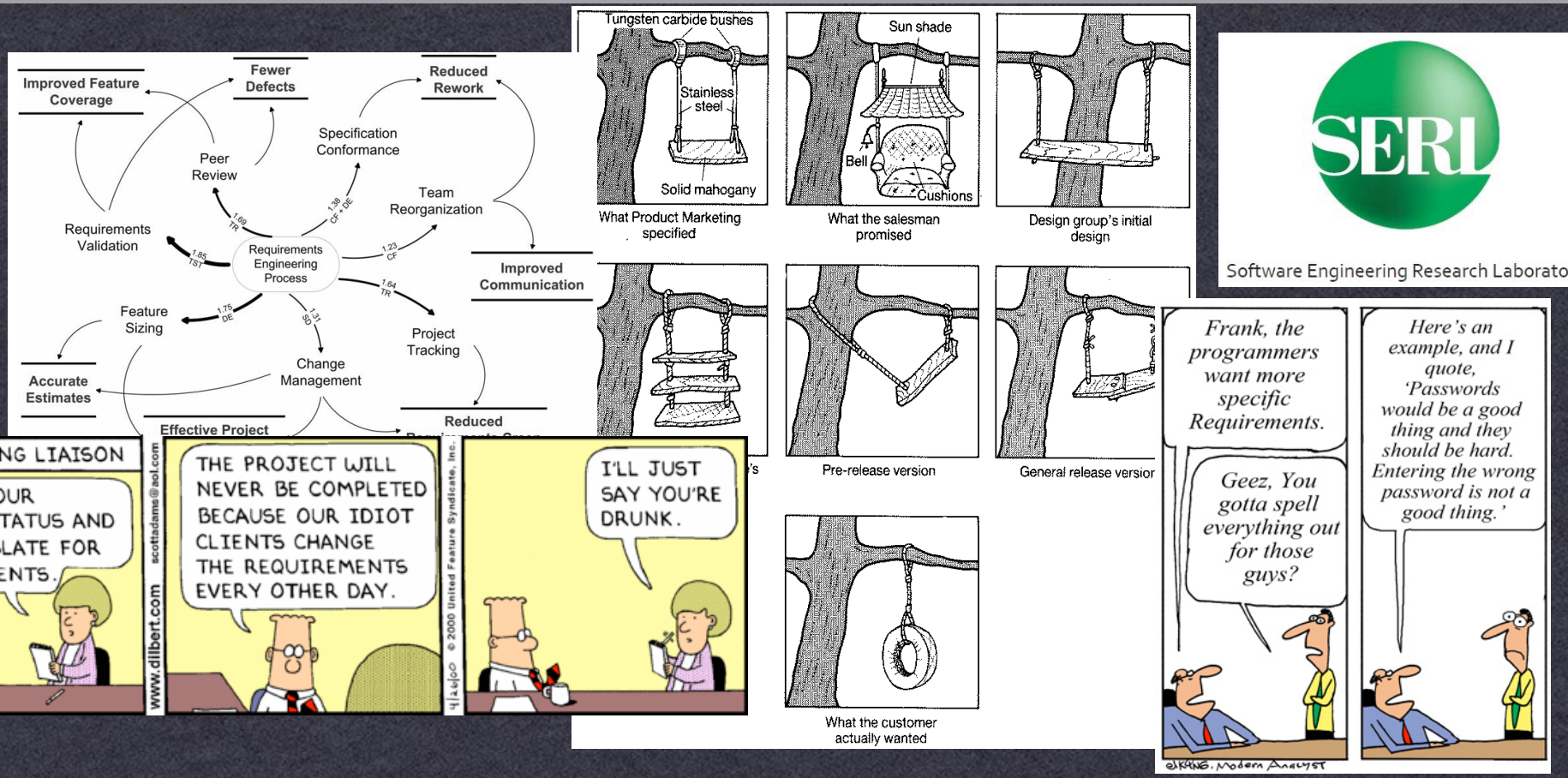


Week 1: Overview of SRE

409220 Software Requirements Engineering

Jim Buchan

AUT COMPUTING + MATHEMATICAL SCIENCES

The aims for Today

- What is Software Requirements Engineering?
- Is it important to project success?
- What are the challenges?
- What are some principles and approaches using proven practice
- Put your hand up now if you read this
- My educational philosophy
- Course Overview

What is a requirement?

(Sommerville and Sawyer 1997):
Requirements are...a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system.

Table 1.1 Types of requirements

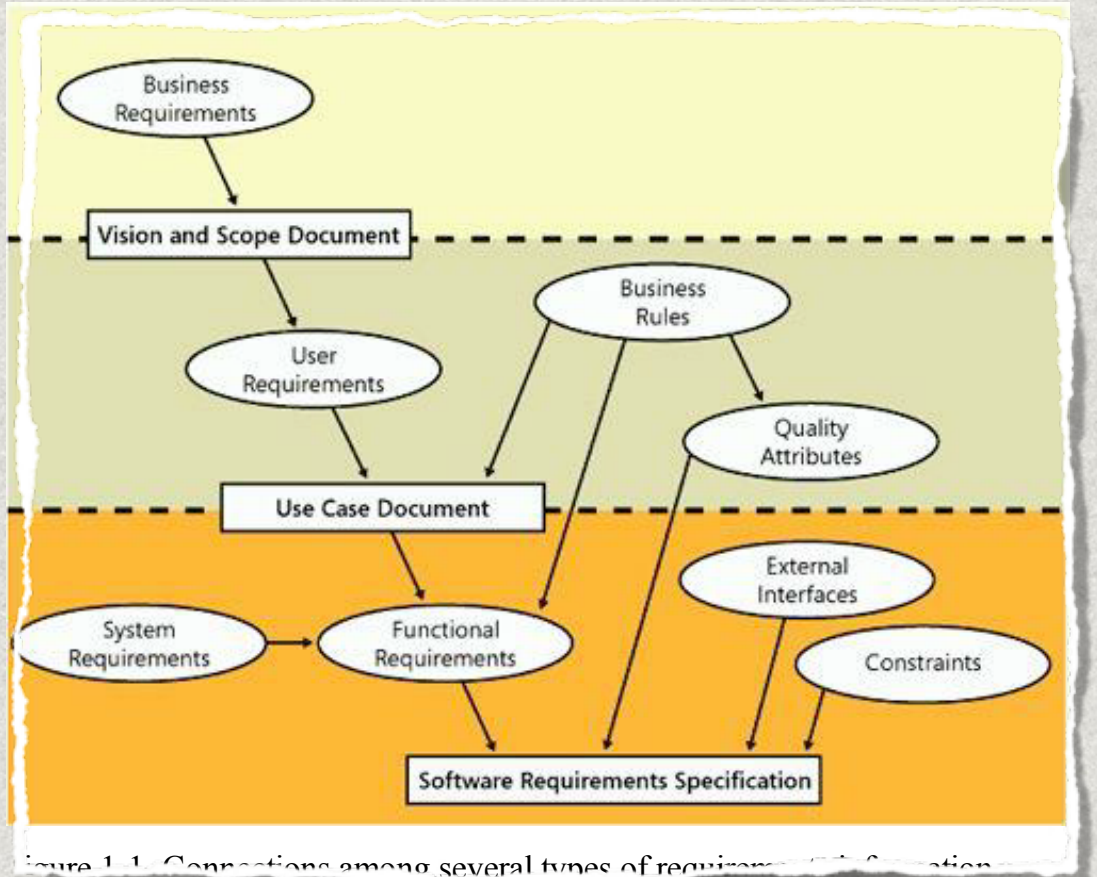
Requirements Classification
• *Functional requirements* — what the system will do • *Non-functional requirements* — constraints on the types of solutions that will meet the functional requirements e.g. accuracy, performance, security and modifiability
• *Goal level requirements* — related to business goals • *Domain level requirements* — related to problem area • *Product level requirements* — related to the product • *Design level requirements* — what to build
• *Primary requirements* — elicited from stakeholders • *Derived requirements* — derived from primary requirements
Others classifications, e.g. • *Business requirements* versus *technical requirements* • *Product requirements* versus *process requirements* —- i.e. business needs versus how people will interact with the system • *Role based requirements,* e.g. customer requirements, user requirements, IT requirements, system requirements, and security requirements

What is SRE?

Discovery

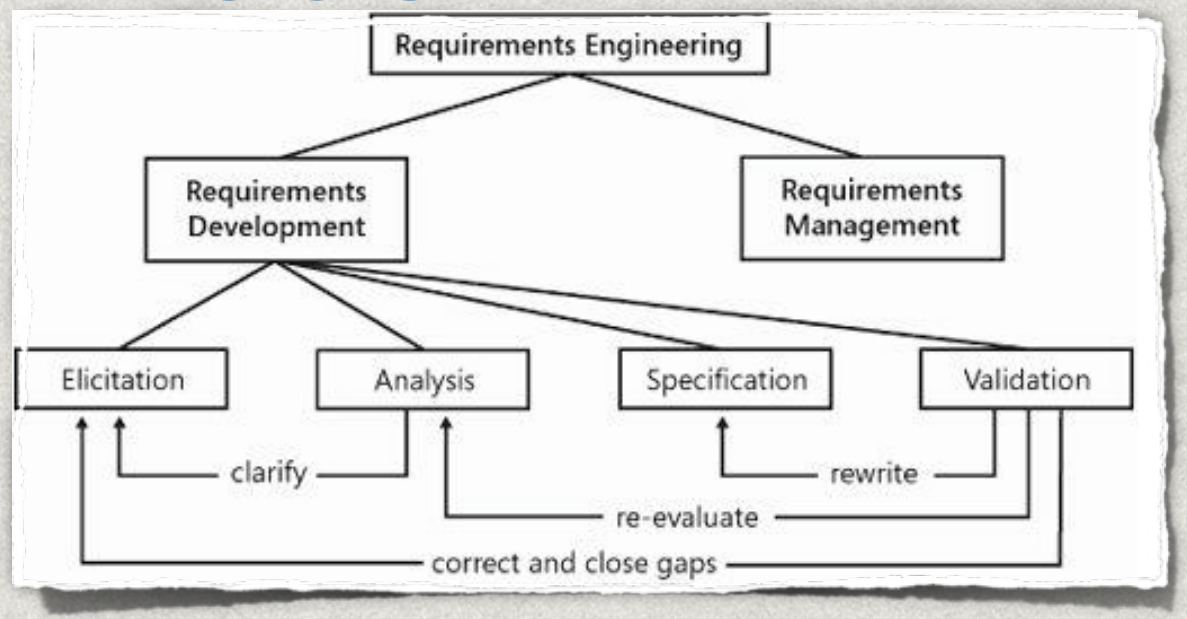
Specification

Verification and Validation



A simple model

- ***Elicitation*** involves the actions we go through to understand our users and discover their needs.



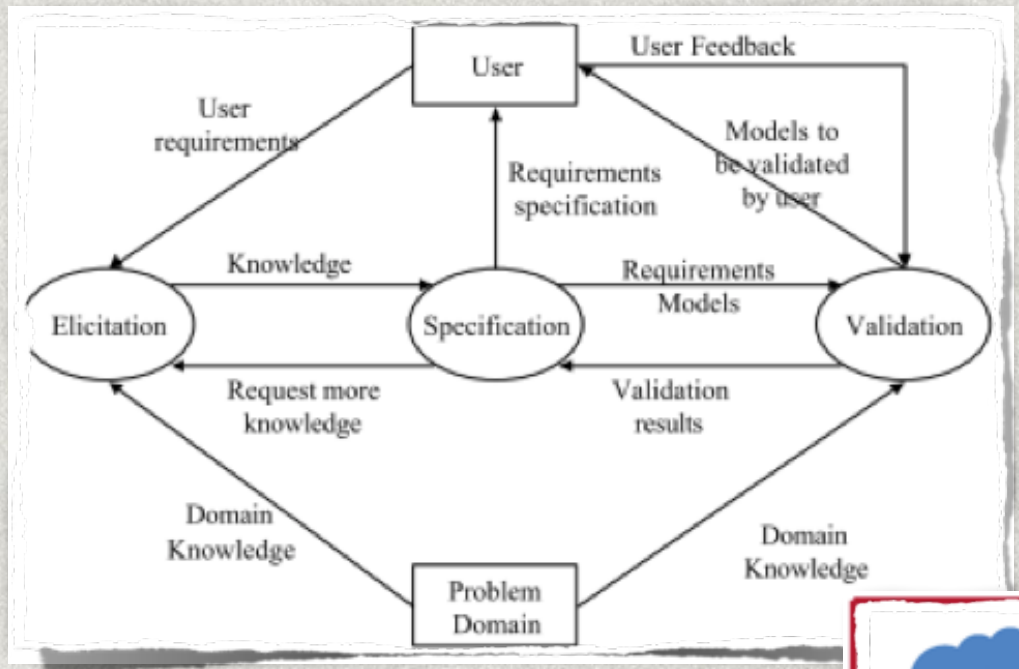
- A major ***requirements analysis*** activity is to derive more detailed requirements from higher-level requirements. Analysis also involves creating multiple views of the requirements, such as prototypes, graphical analysis models, and tests. Other aspects of requirements analysis include negotiating priorities, searching for missing requirements, and evaluating technical feasibility, risk, and failure modes. Analysis provides a feedback loop that refines the understanding that the analyst developed during an elicitation activity

- ***Specification*** involves recording the various types of requirements information in forms that will facilitate communication among the project stakeholders. Traditionally these forms are documents containing natural language text. Other representation techniques also are valuable, such as graphical analysis models, tables, and mathematical expressions.

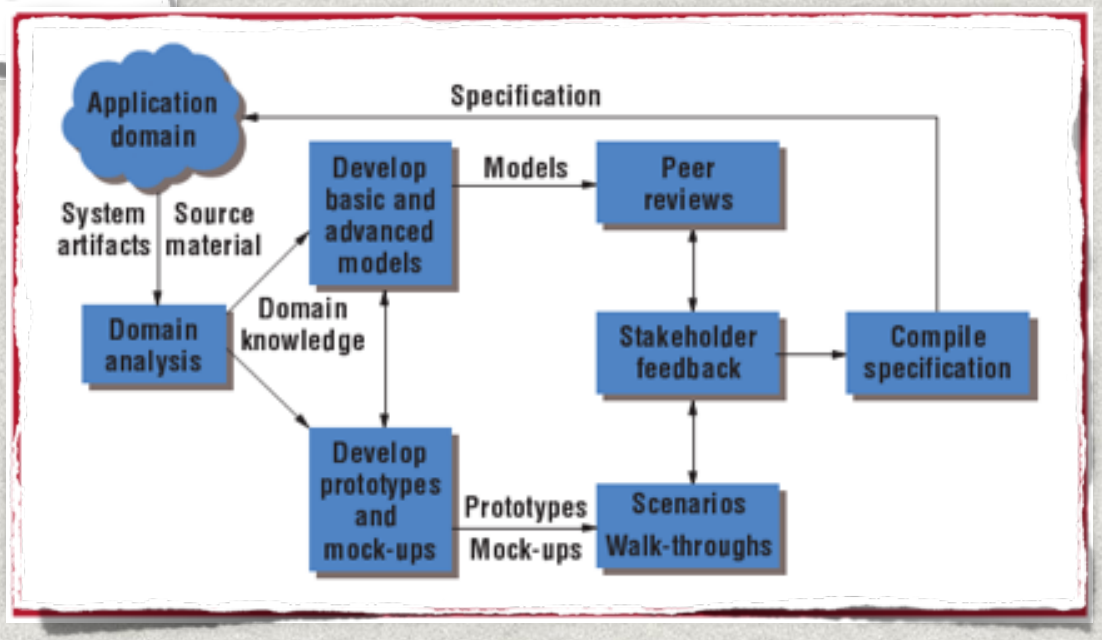
. ***Validation*** ensures that those requirements are correct, will satisfy customer needs, and have all the characteristics of high-quality requirements. Validation might lead the analyst to rewrite some requirements specifications, to reassess the initial analysis, or to correct and refine the set of documented requirements.

AUT COMPUTING + MATHEMATICAL SCIENCES

Lots of Models



(Loucopoulos & Karakostas, 1995)



(Hofmann & Lehner, 2001)

AUT COMPUTING + MATHEMATICAL SCIENCES

A simple model of RE

Call out what you think the requirements **process activities** should be in order to deliver a large piece of software, starting with an idea from a client.

Write down 1 curiosity Question about RE activities

What Curiosity Questions Do You Have about SRE?

- Where Do Requirements Come From?
- What Are Good Requirements?
- How Are Requirements Used And By Whom?
- How Much Detail Is Needed When?
- Who Uses The Requirements And How?
- How Do I Know The Requirements Are The Right Ones?
- What What Processes, Techniques Should Be Used?

AUT COMPUTING + MATHEMATICAL SCIENCES

Most Modern SE approaches are:

Iterative

Incremental

Collaborative

Not a big up-front requirements analysis (for most projects)

Does having a set of good quality requirements mean Project success?

Does a set of poor requirements mean project failure?

feature

requirements

Requirements Engineering as a Success Factor in Software Projects

Hubert F. Hofmann, *General Motors*

Franz Lehner, *University of Regensburg*

Deficient requirements are the single biggest cause of software project failure. From studying several hundred organizations, Capers Jones discovered that RE is deficient in more than 75 percent of all enterprises.[1] In other words, getting requirements right might be the single most important and difficult part of a software project. Despite its importance, we know surprisingly little about the actual process of specifying software. "The RE Process" sidebar provides a basic description.

Based on their field study of 15 requirements engineering teams, the authors identify the RE practices that clearly contribute to project success, particularly in terms of team knowledge, resource allocation, and process.

Most RE research is conceptual and concentrates on methods or techniques, primarily supporting a single activity. Moreover, the rare field studies we actually have do not establish a link between RE practices and performance. We therefore conducted this study to identify the RE practices that clearly contribute to software project success.

Stakeholders and teams

Stakeholders are individuals and organizations that are actively involved in a software project or whose interests the project affects. Stakeholders of any computer system can include customers, users, project managers, analysts, developers, senior management, and quality assurance staff. Table 1 illustrates the wide range of expertise and motivations that stakeholders typically exhibit.[2]

A typical software project team consists of a project manager, analysts, developers, and quality assurance personnel. Often it includes users or their representatives. In the case of commercial off-the-shelf (COTS) software, marketers such as sales representatives and account managers tend to substitute for users and customers.

Field study

Seven field studies have reported on RE in practice.[3–9] Unfortunately, these rare studies have not established a clear link to performance and tend to focus on a narrow set of variables. Our study provides a more integrated view of RE by investigating team knowledge, allocated resources, and deployed RE processes (see Figure 1) and their contribution to project success. In addition, we incorporate the observations of previous field studies.

Fifteen RE teams, including six COTS and nine customized application develop-

0740-7459/01/$10.00 © 2001 IEEE

An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management

Daniela Damian and James Chisan

Abstract—Requirements engineering is an important component of effective software engineering, yet more research is needed to demonstrate the benefits to development organizations. While the existing literature suggests that effective requirements engineering can lead to improved productivity, quality, and risk management, there is little evidence to support this. We present empirical evidence showing how requirements engineering practice relates to these claims. This evidence was collected over the course of a 30-month case study of a large software development project undergoing requirements process improvement. Our findings add to the scarce evidence on RE payoffs and, more importantly, represent an in-depth explanation of the role of requirements engineering processes in contributing to these benefits. In particular, the results of our case study show that an effective requirements process at the beginning of the project had positive outcomes throughout the project lifecycle, improving the efficacy of other project processes, ultimately leading to improvements in project negotiation, project planning, and managing feature creep, testing, defects, rework, and product quality. Finally, we consider the role collaboration had in producing the effects we observed and the implications of this work to both research and practice.

Index Terms—Requirements engineering, process improvement, process interactions, empirical investigation.

1 Introduction

Requirements engineering (RE)—the elicitation, definition, and management of requirements—is often cited as one of the most important, but difficult, phases of software development [4]. Attention to upfront requirements activities has been said to produce benefits such as preventing errors, improving quality, and reducing risk throughout software development projects [4], [28]. Studies conducted by the Standish Group [33] found a striking 74 percent project failure rate, while 28 percent of projects were cancelled completely. The study suggests that the top factors of failure are related to requirements problems, including lack of user input, lack of a clear statement of requirements, and incomplete and changing requirements.

However, this study is based on a survey of past projects and concrete evidence based on systematic studies of the role of good RE practice in software development is very limited [9]. This becomes problematic both for research on the fundamentals of software engineering and RE research exchange with practitioners. Panels of researchers and practitioners at major international conferences on requirements engineering have repeatedly considered the topic of RE research adoption (e.g., [1], [21]). One of the major issues they identified is the lack of concrete knowledge of what organizations can gain from applying state-of-the-art requirements approaches [21].

In this paper, we present research toward filling the gap between claims in the literature and requirements engineering practice. We conducted a 30-month *explanatory* case study at an organization that had revised its RE process (REP) and which provided us with the opportunity to assess the effects of improved RE over an entire project lifecycle. The evidence collected enabled us to explain how RE processes played a part in improving developer productivity, product quality, and project risk management. The case study was conducted in three separate stages. Earlier publications of our research report findings from the first two stages, in particular, the perceived benefits of the improved RE practice in the early [6] as well as the downstream [7] stages of development in the studied organization.

Here, we discuss evidence from the third stage of our three-stage case study. In this stage of our research, we sought to understand how the RE process can interact with other processes and how this interaction may have contributed to the effects we observed earlier in our study. We present the evidence from this work in the context of the entire research study and specifically build on the findings

• *The authors are with the Department of Computer Science, Engineering/ Computer Science Building (ECS), Room 504, University of Victoria, 3800 Finnerty Road, Victoria, BC, Canada V8P 5C2.*
E-mail: danielad@cs.uvic.ca, james@chisan.com.

Manuscript received 10 June 2005; revised 17 May 2006; accepted 1 June 2006; published online 9 Aug. 2006.
Recommended for acceptance by N. Maiden.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0179-0605.

Requirements Eng (2006) 11: 1–3
DOI 10.1007/s00766-004-0206-4

VIEWPOINTS

Alan M. Davis · Didar Zowghi

Good requirements practices are neither necessary nor sufficient

Received: 17 August 2004 / Accepted: 3 September 2004 / Published online: 8 October 2004

The title of this essay was selected to create controversy among the readers and perhaps cause some of you to become defensive. However, we will try to go about demonstrating why we sincerely believe that good requirements practices are indeed neither necessary nor sufficient, even though we are devoted to the field of requirements as active researchers and practitioners. In fact, after you read this article, we trust that you too will say, "Well, of course, I agree with you."

To argue about the necessity or sufficiency of good requirements practices, we must first agree to (1) what is a requirements practice, (2) what is a *good* requirements practice, (3) what is the purpose of a requirements practice, and (4) what is it necessary and sufficient for? Only then can we argue why "good" requirements practices are neither necessary nor sufficient to achieve the intended purpose for requirements.

There is little uniformity in terminology concerning the classes of activities that make up requirements. However, we all seem to agree that the following activities are included:

- Gathering/eliciting/ascertaining/uncovering requirements from stakeholders
- Analyzing (and perhaps modeling and/or refining) those requirements for consistency, completeness, appropriateness, and so on
- Determining what subset of those requirements should actually be addressed given the constraining budgets and schedules
- Documenting the selected requirements
- Verifying that the specified requirements conform to all the quality standards
- Managing changes to requirements

We consider a *requirements practice* to be the use of a principle, tool, notation, and/or method in order to perform any or all of the above activities. Many sources of requirements practices exist, e.g., Davis [1], Gause and Weinberg [2], Gottesdiener [4], IEEE Standard 830 [6], Laueson [7], Leffingwell and Widrig [8], the Robertons [9], Sommerville and Sawyer [10], and Weigers [11].

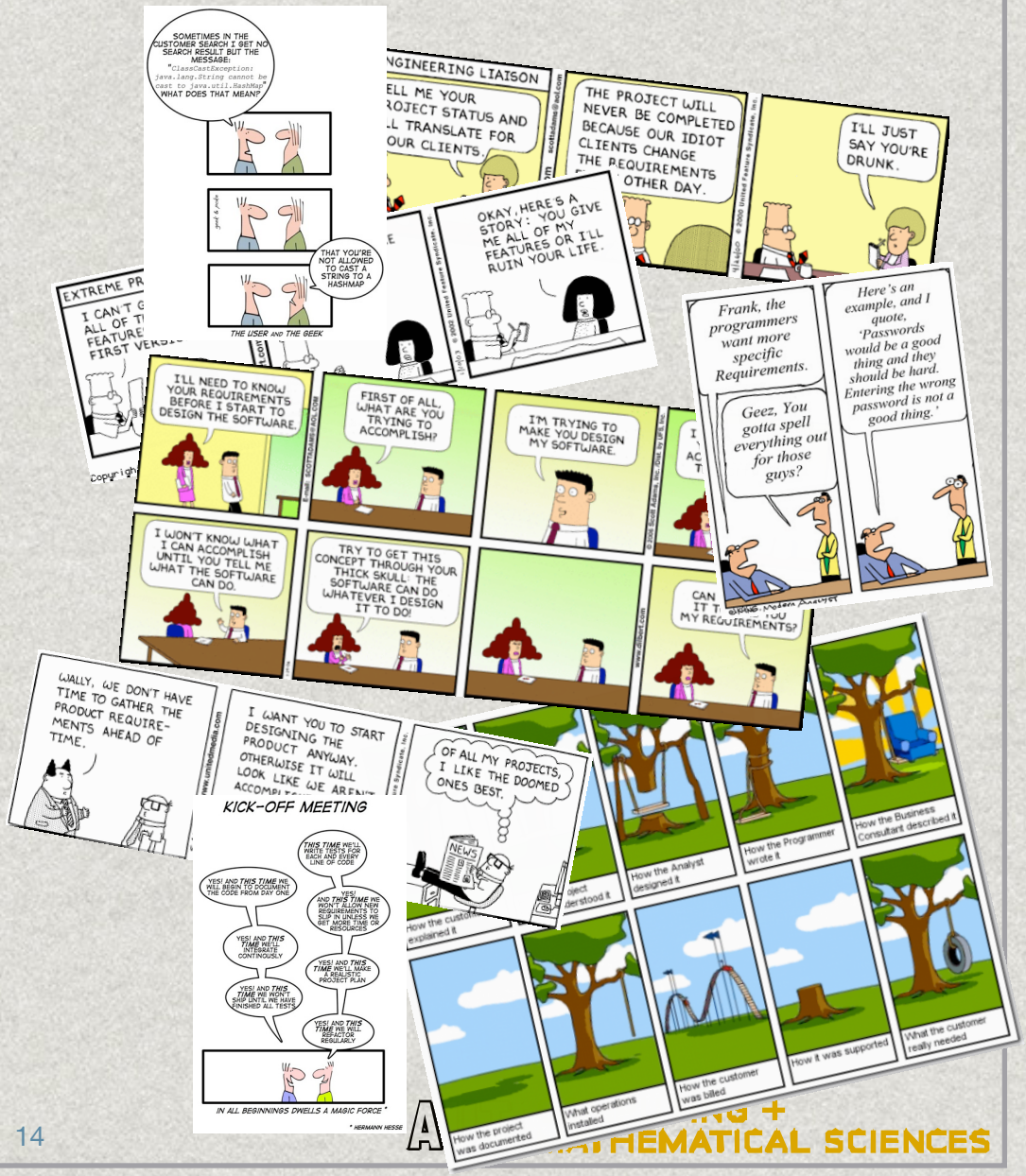
We consider a *good requirements practice* to be a requirements practice that either reduces the cost of the development project or increases the quality of the resulting product when used in specific situations. Few requirements practices have been validated as "good" in practice, and those that have rarely, if ever, describe the specific situations in which they are effective. However, we do have a variety of sources of requirements practices for which the authors do claim goodness. For example, Sommerville and Sawyer wrote a book titled *Requirements Engineering: A Good Practice Guide* [10], Weigers [11] provides a chapter on *good* practices, Young has written a book titled *Effective Requirements Practices* [12] and the Robertsons' book discusses *Mastering* [9] the requirements process. One can think of a set of indicators that you can check your RE practice against to decide if your practices are "good" or not. Table 1 provides three examples of requirements practices that all of the above sources seem to agree are good.

The *purpose of* requirements is to raise the likelihood that the right system will be built, i.e., the system when built satisfies its intended customers and addresses their needs to an acceptable degree. Some may argue that the purpose is more short term, e.g., that its purpose is to ensure communication among all stakeholders, provide

A. M. Davis (✉)
Department of Information Systems, College of Business,
The University of Colorado at Colorado Springs,
1420 Austin Bluffs Parkway, PO Box 7150,
Colorado Springs, CO 80933-7150, USA
E-mail: adavis@uccs.edu

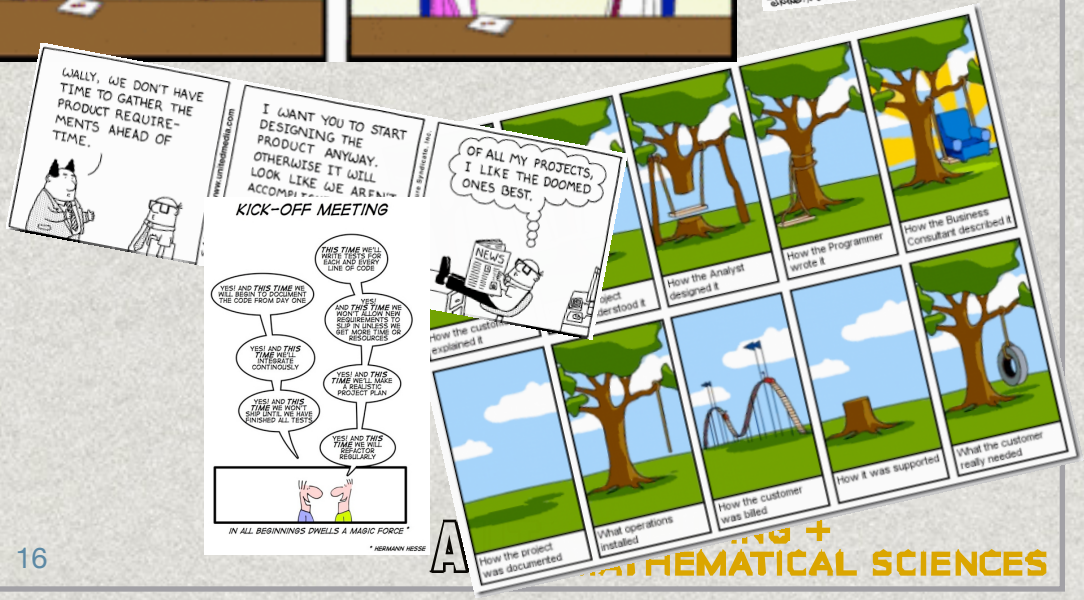
D. Zowghi
Department of Software Engineering,
Faculty of Information Technology,
University of Technology, Sydney,
P O Box 123, Broadway, Sydney,
NSW, 2007, Australia
E-mail: didar@it.uts.edu.au

What are the Challenges?

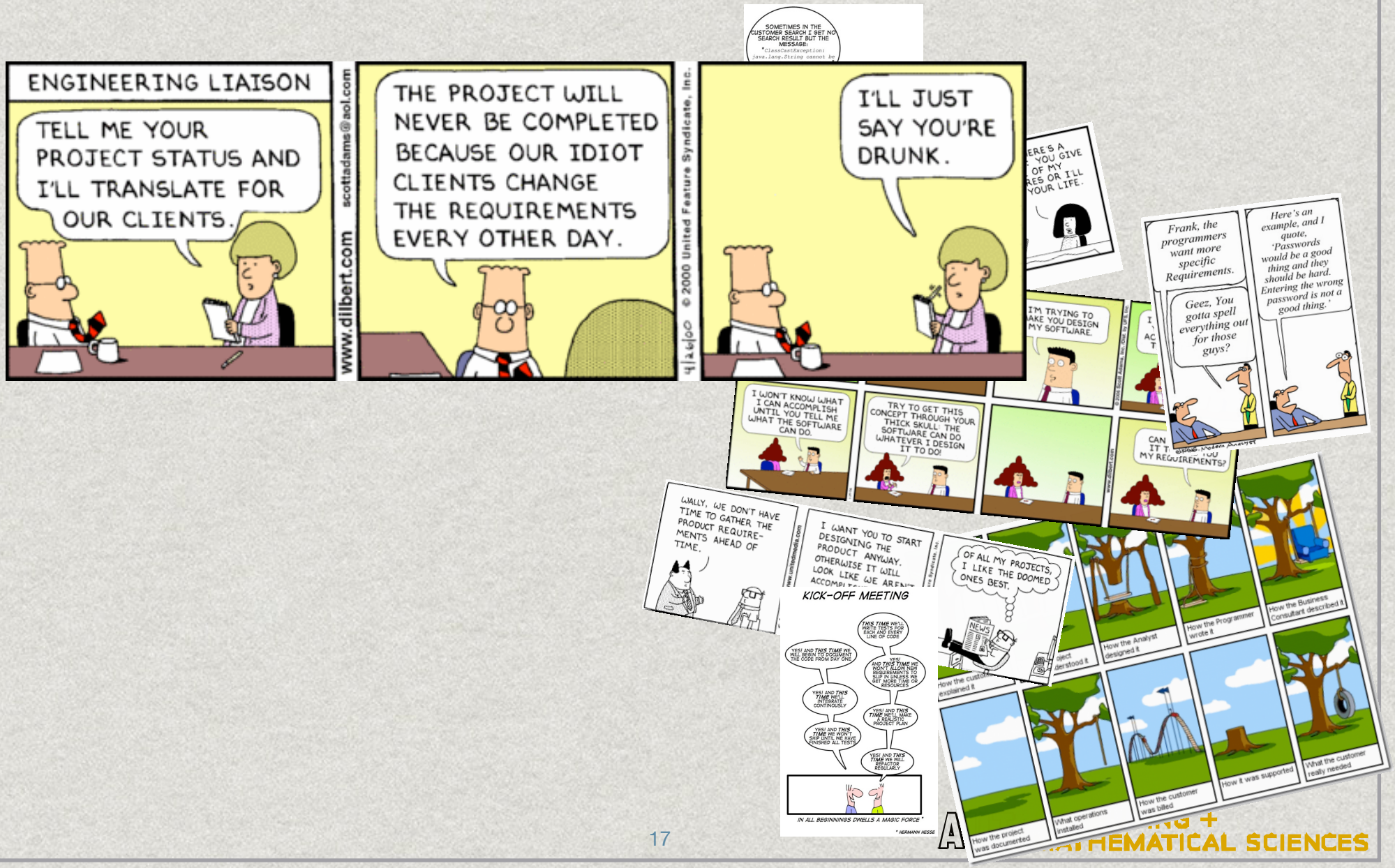


What are the Challenges?

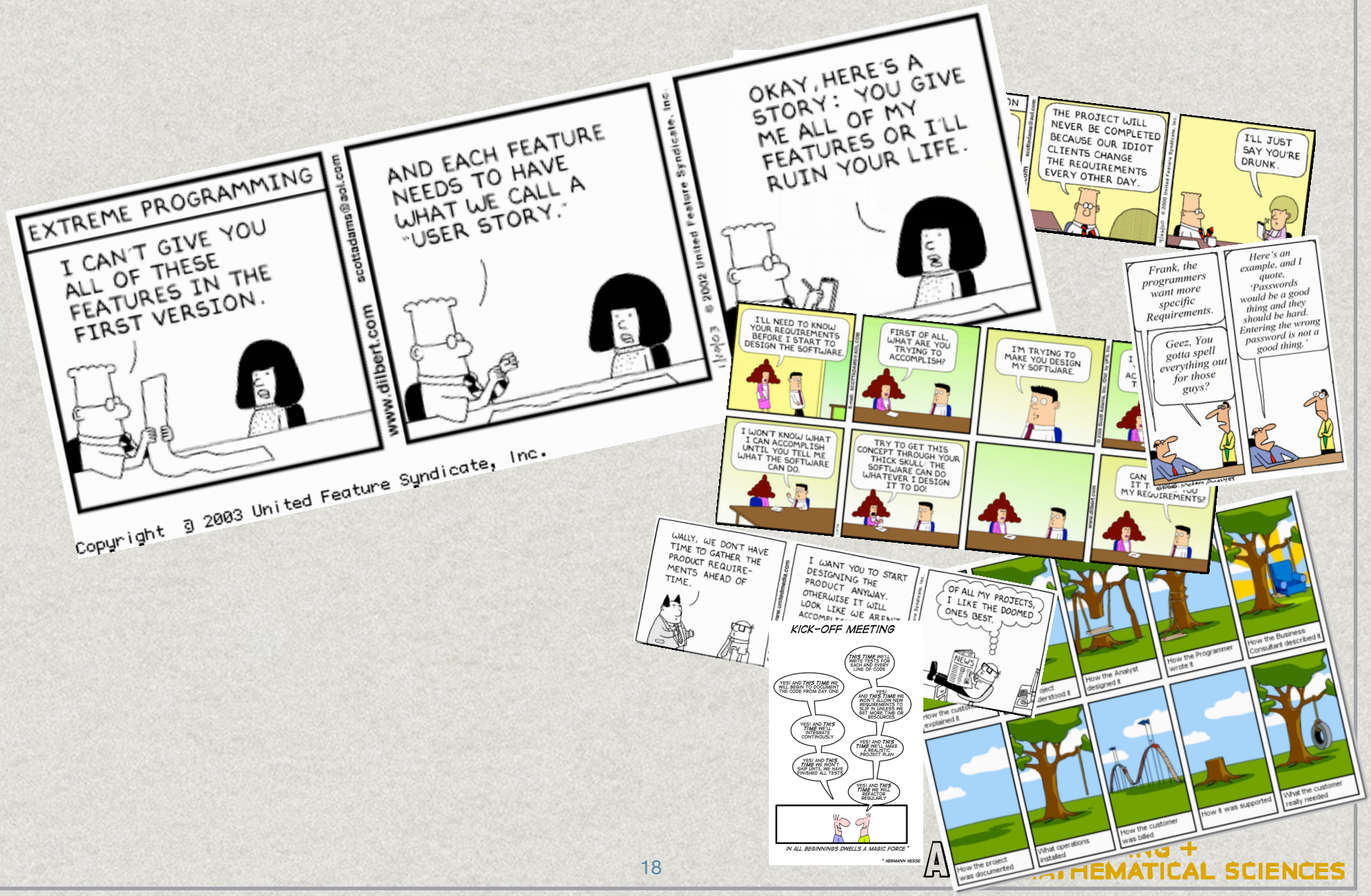
What are the Challenges?



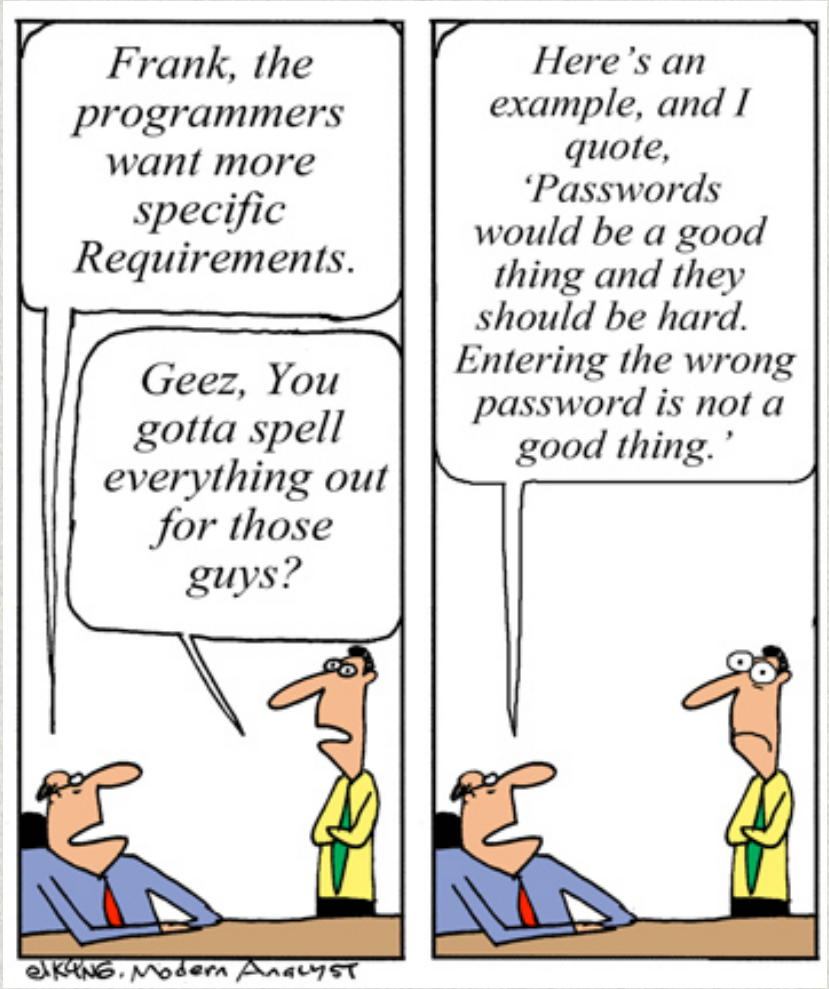
What are the Challenges?

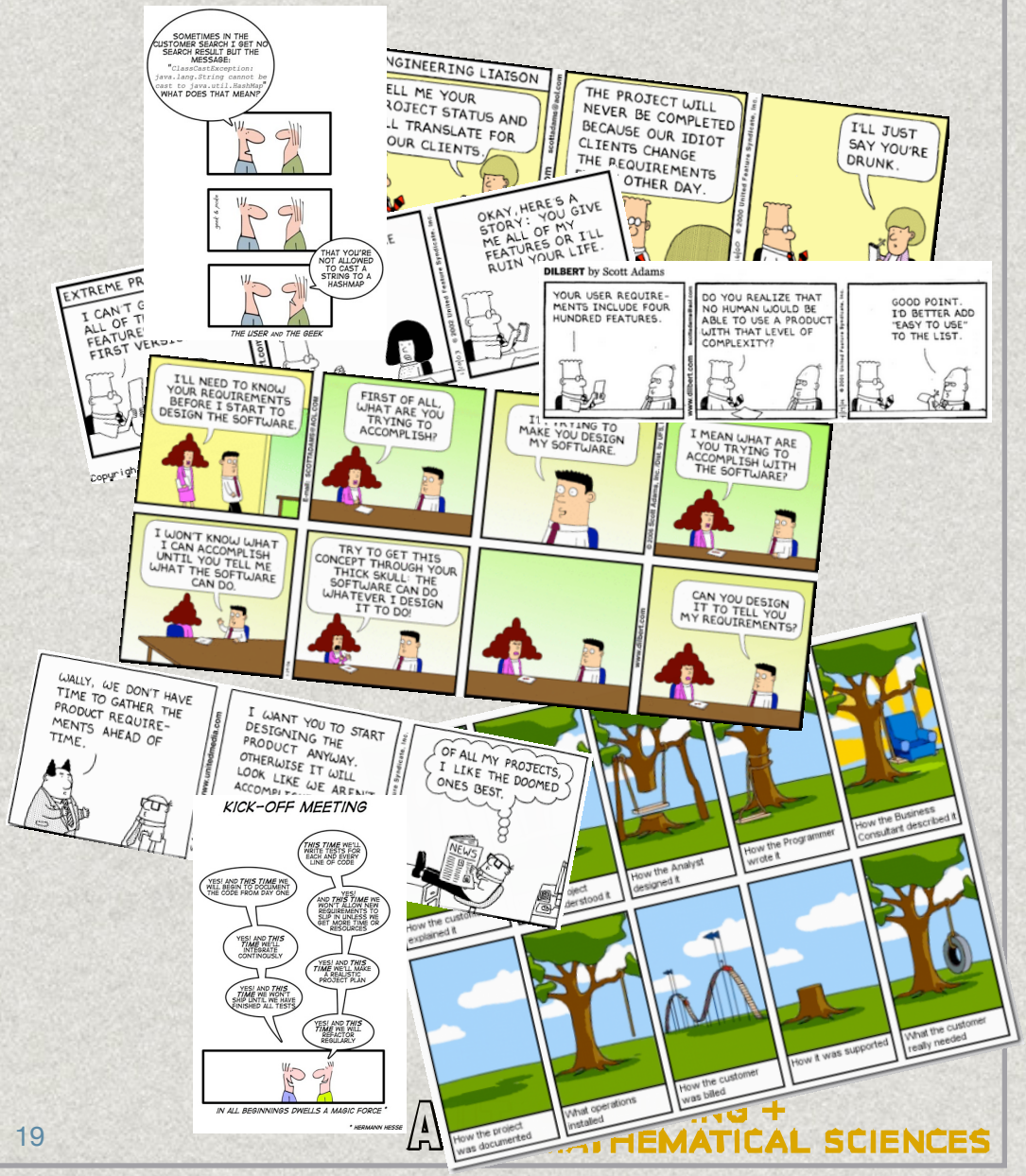


What are the Challenges?

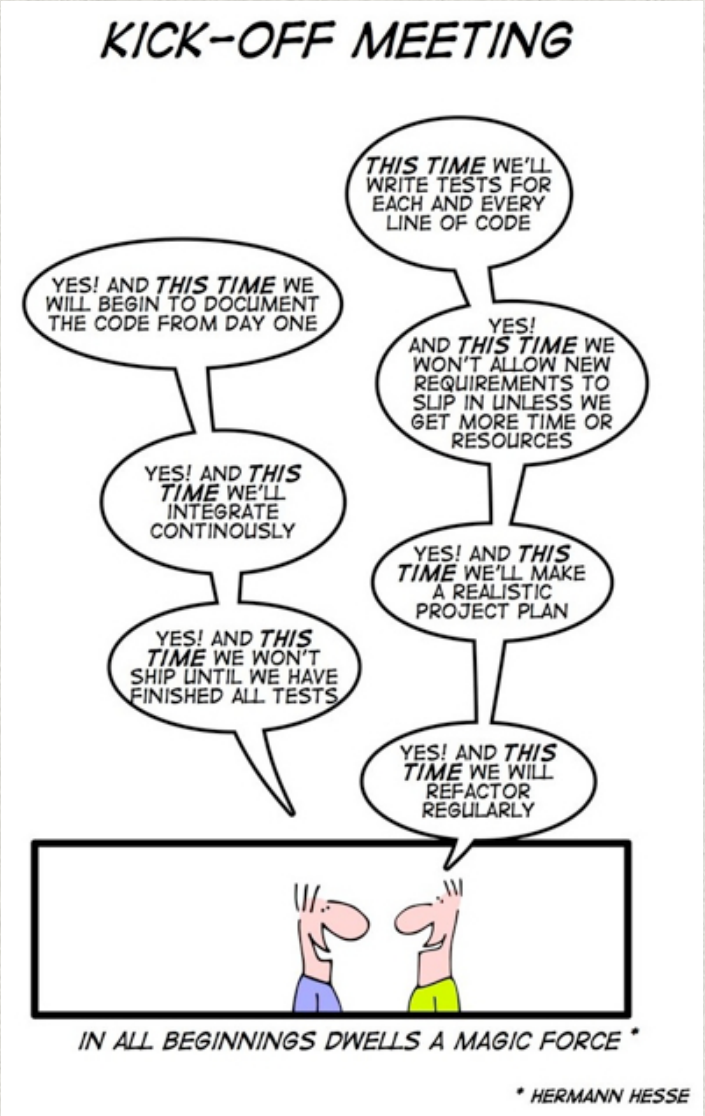


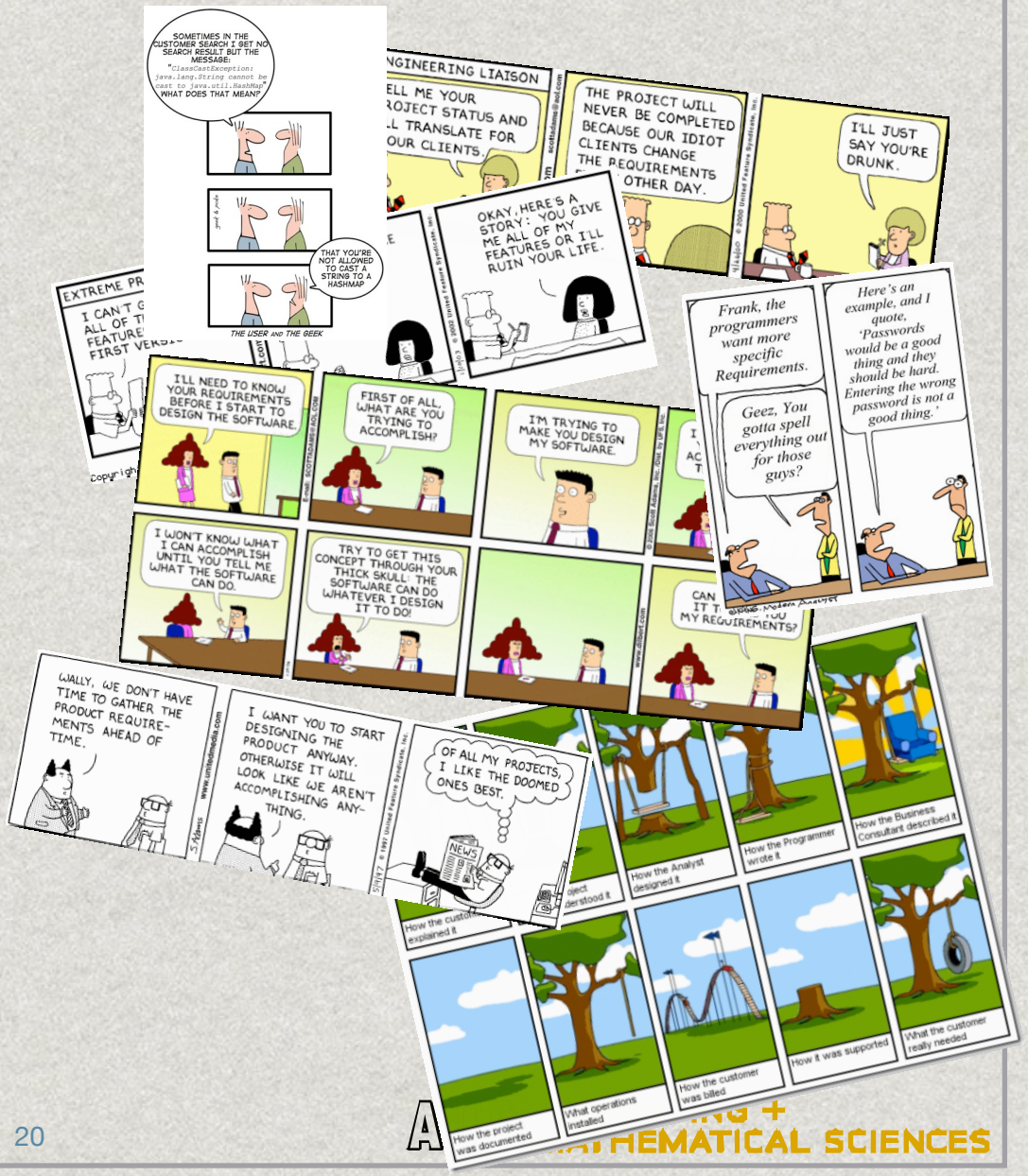
What are the Challenges?



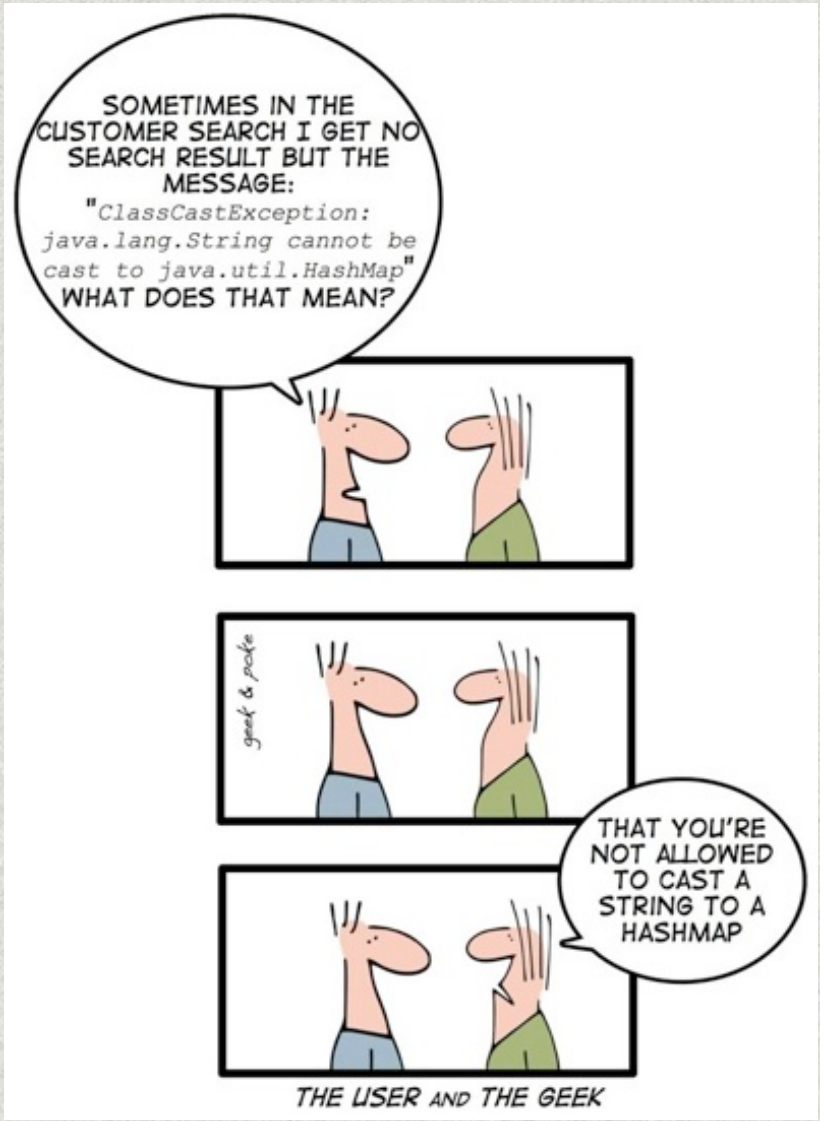


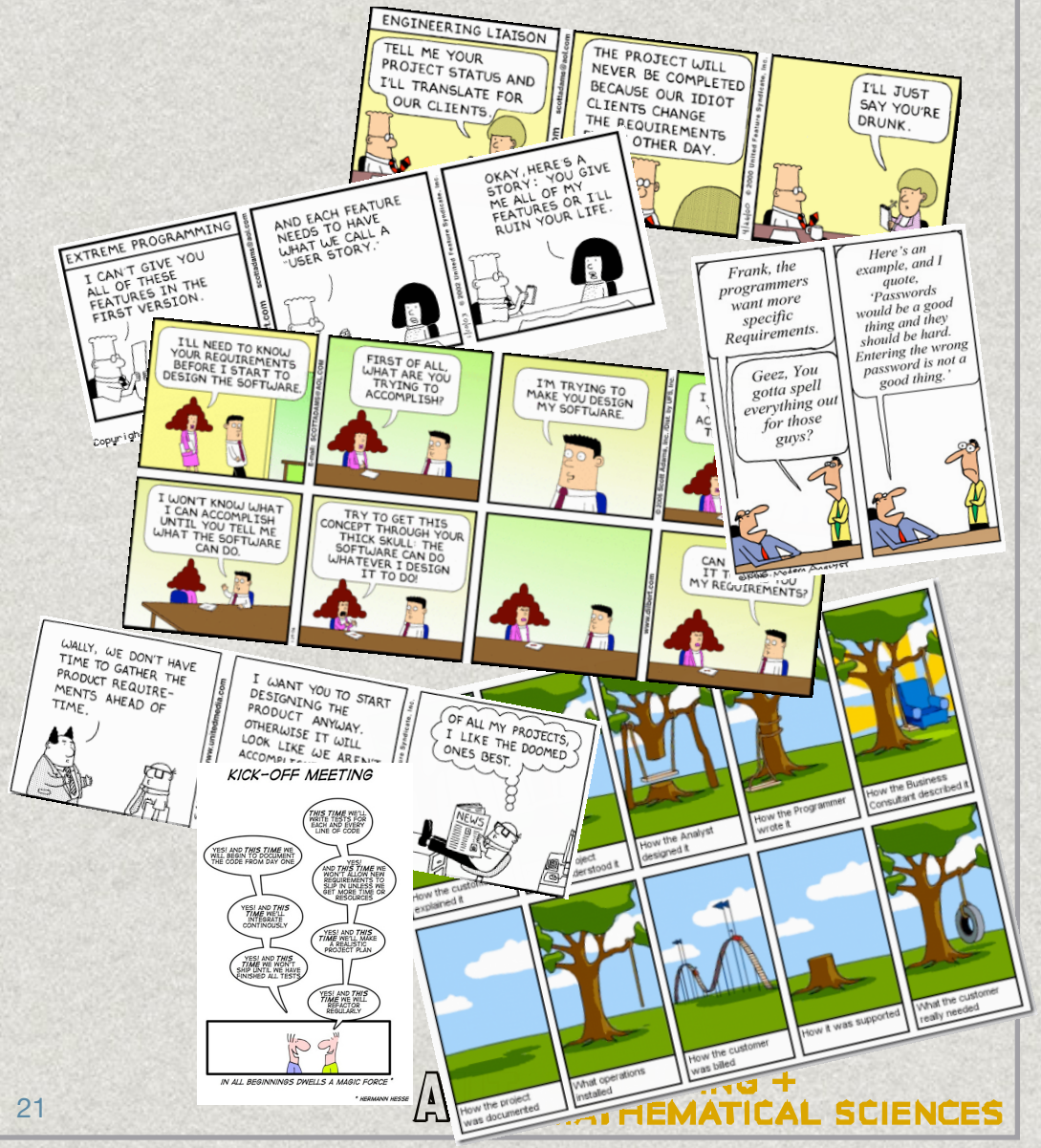
What are the Challenges?



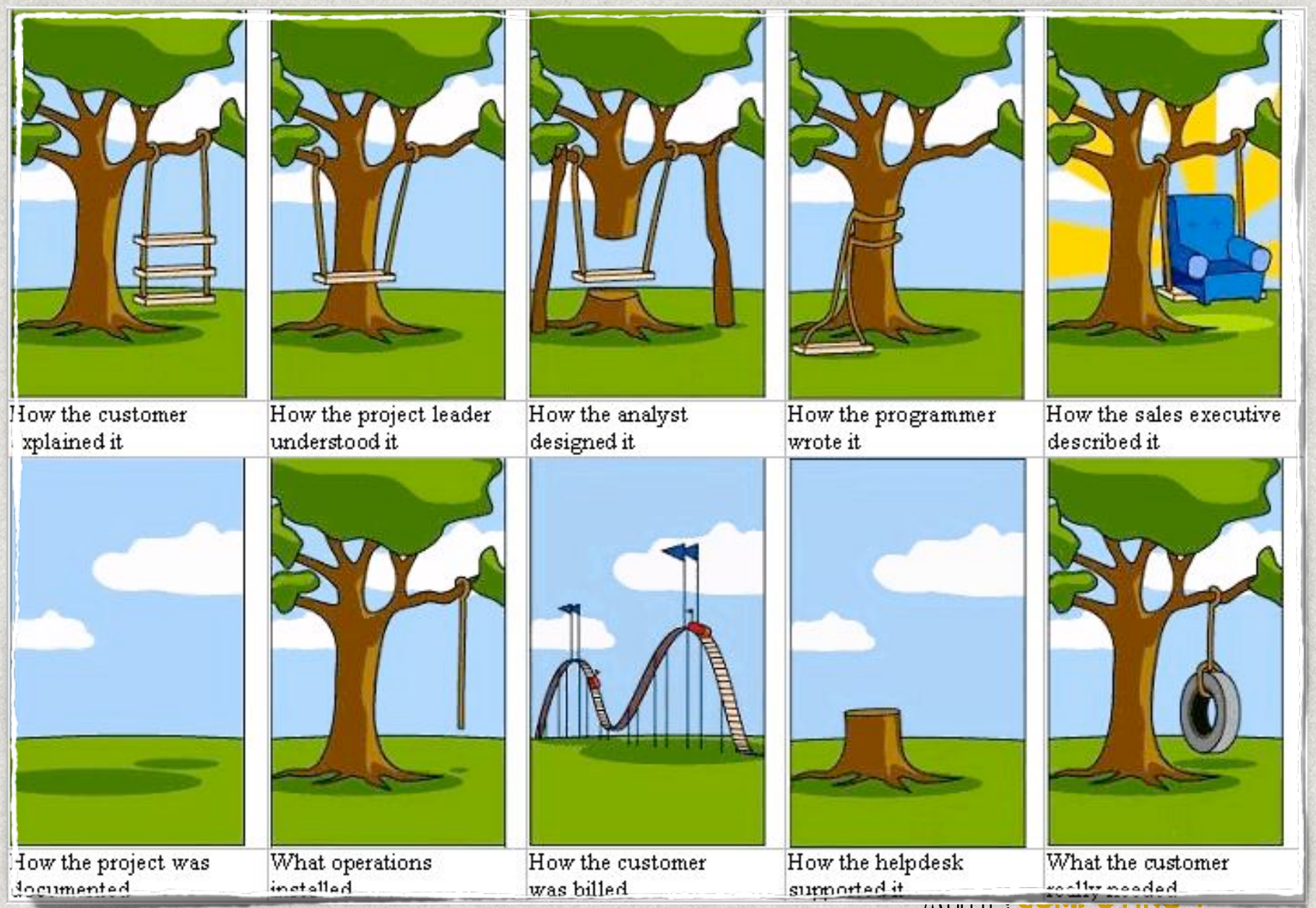


What are the Challenges?





What are the Challenges?

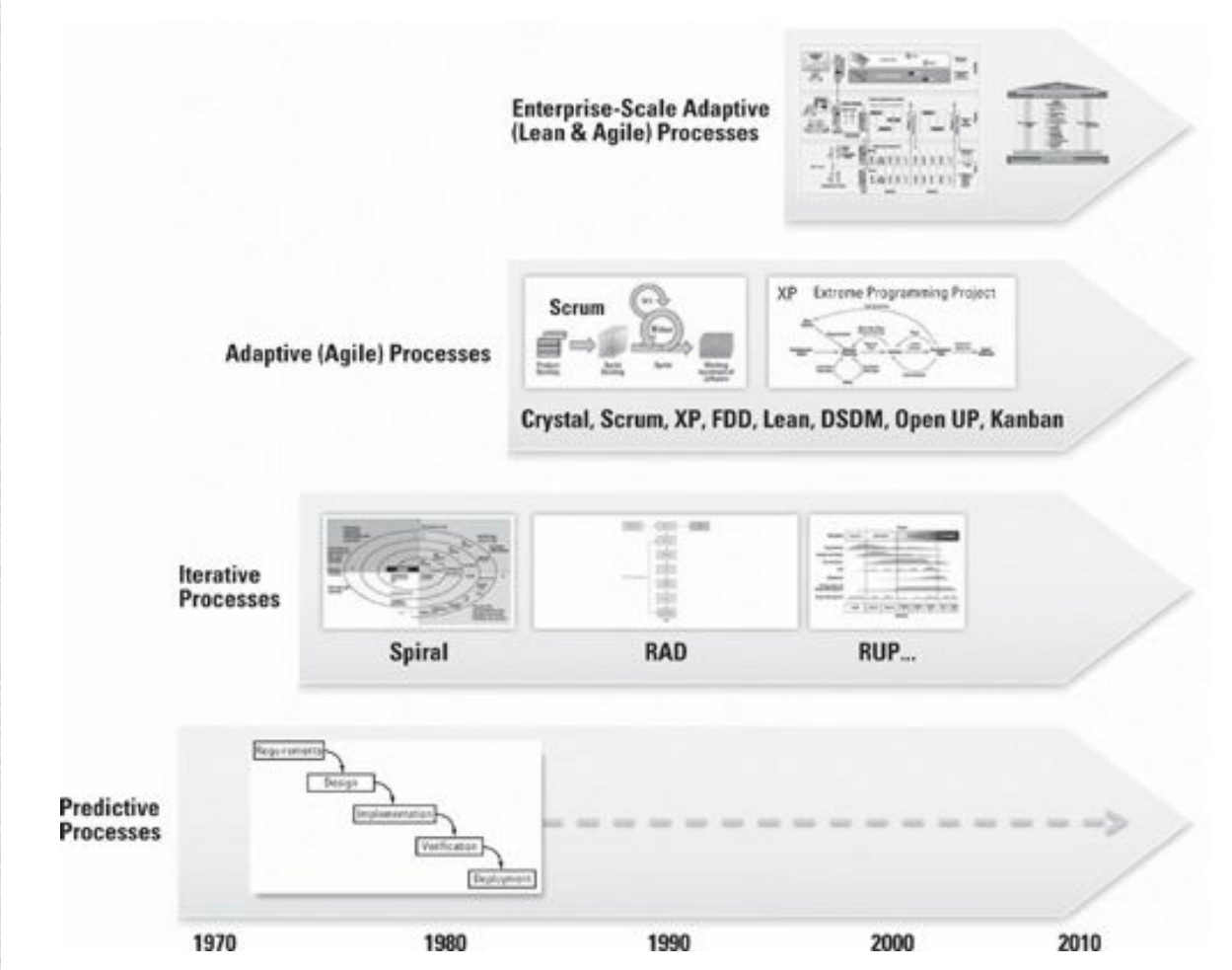


It is not enough to do your best: you must know what to do, and THEN do your best.

-- W. Edwards Deming

What processes are proven practice

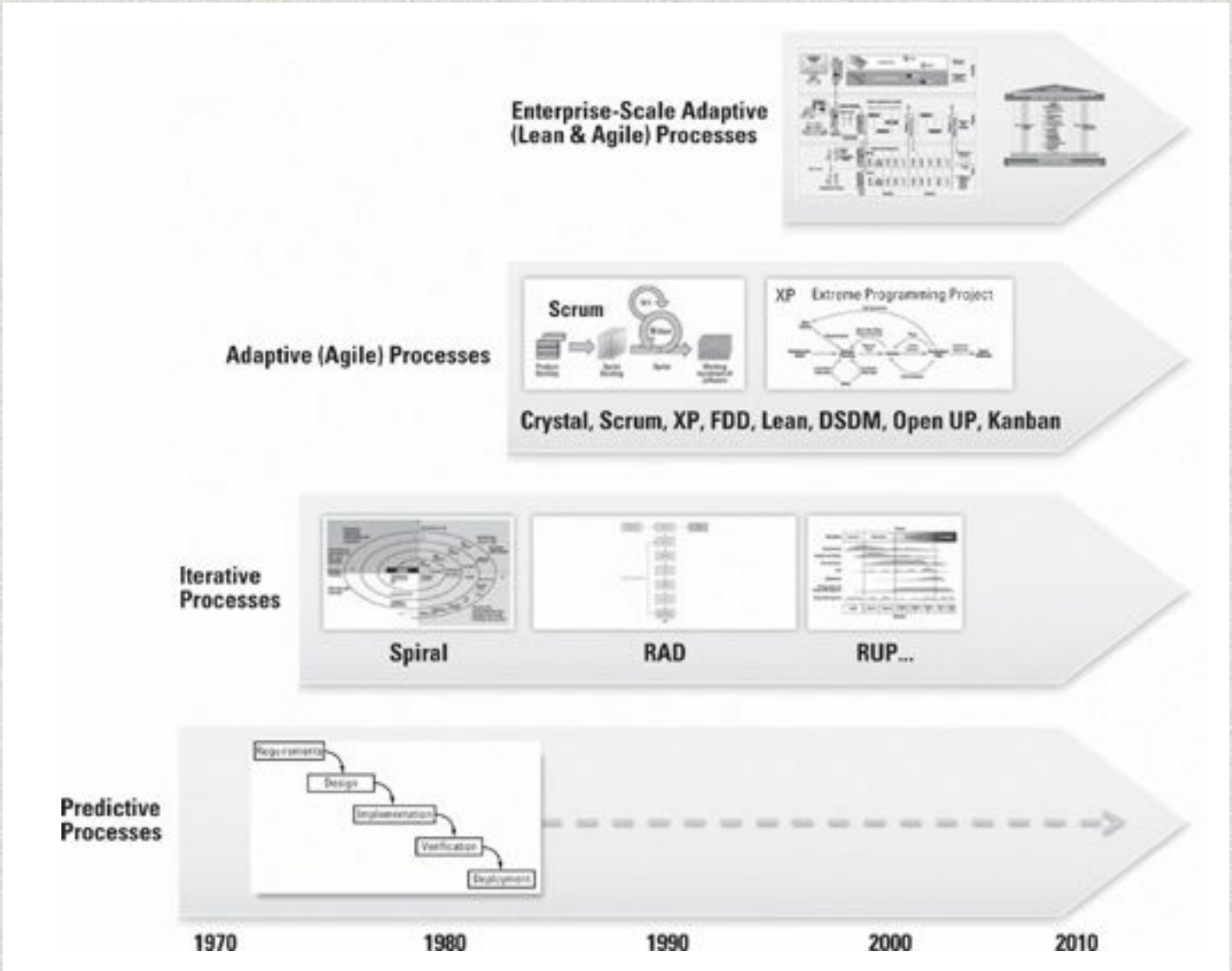
- What are the RE processes in different SE approaches and methodologies?



Name game

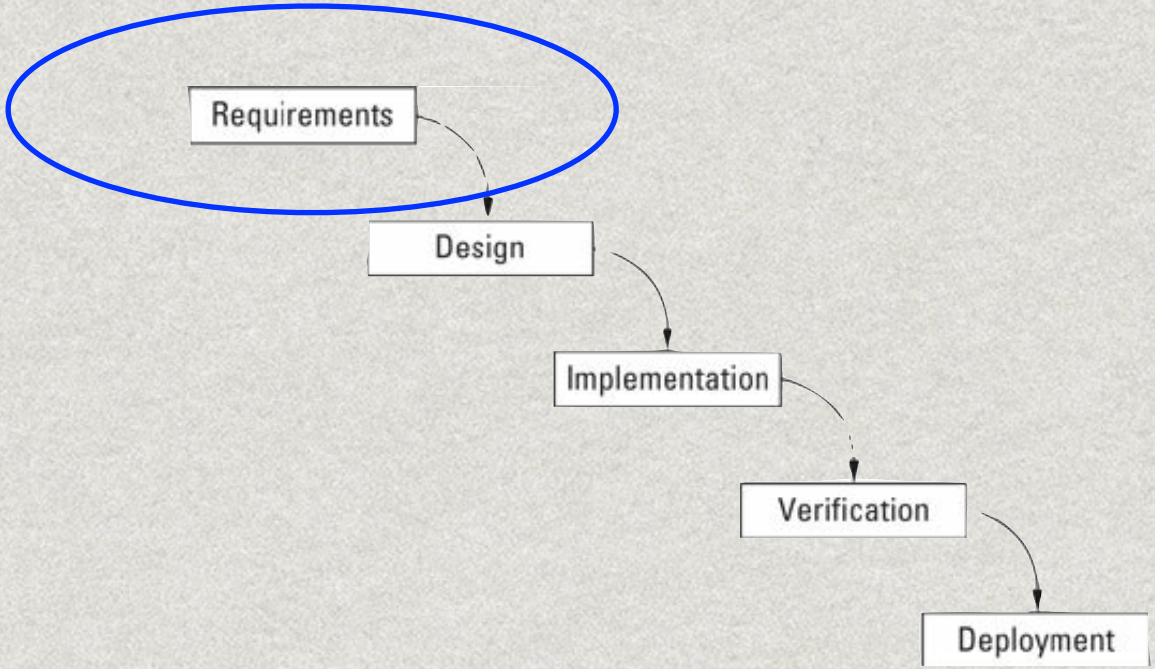
A brief history of SRE

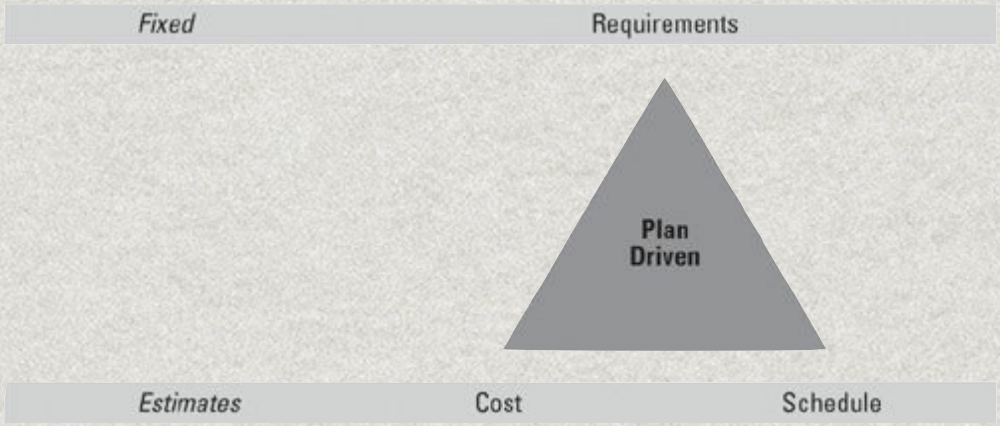
Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

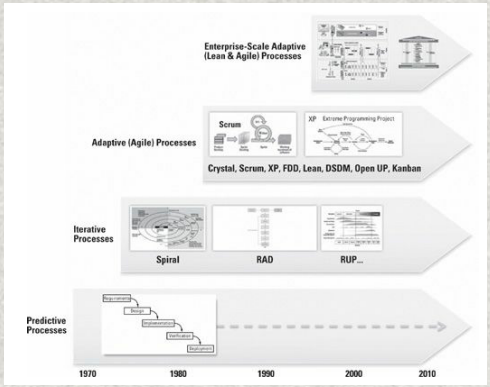


A brief history of SRE

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.





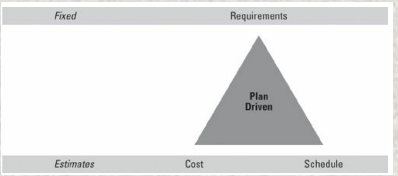


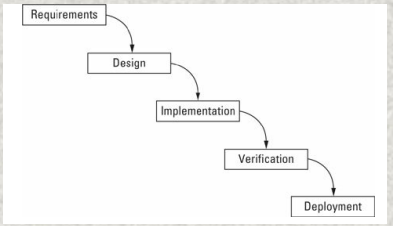
AUT COMPUTING + MATHEMATICAL SCIENCES

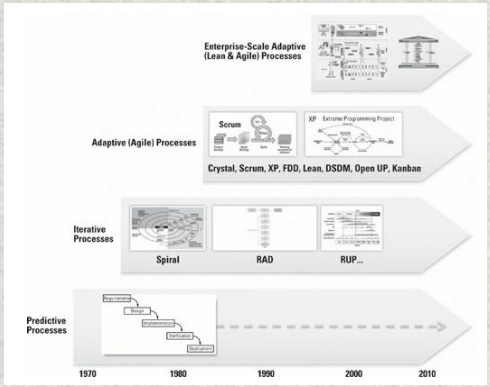
Waterfall

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.
- It appears to be extremely logical and prescriptive. Understand requirements. Design a system that conforms. Code it. Test it. What could be more sensible and logical than that?
- It worked to a point (we did and still do ship a lot of software using the model). As a result, companies built their project and program governance models, including business case and investment approvals, project review and quality assurance milestones, and the like, around its flawed software life cycle.
- It reflects a continuing market reality—customers still do impose fixed-date/fixed- requirements agreements on suppliers, and they will likely continue to do so for years to come. (And, yes, sometimes we impose them on ourselves.)





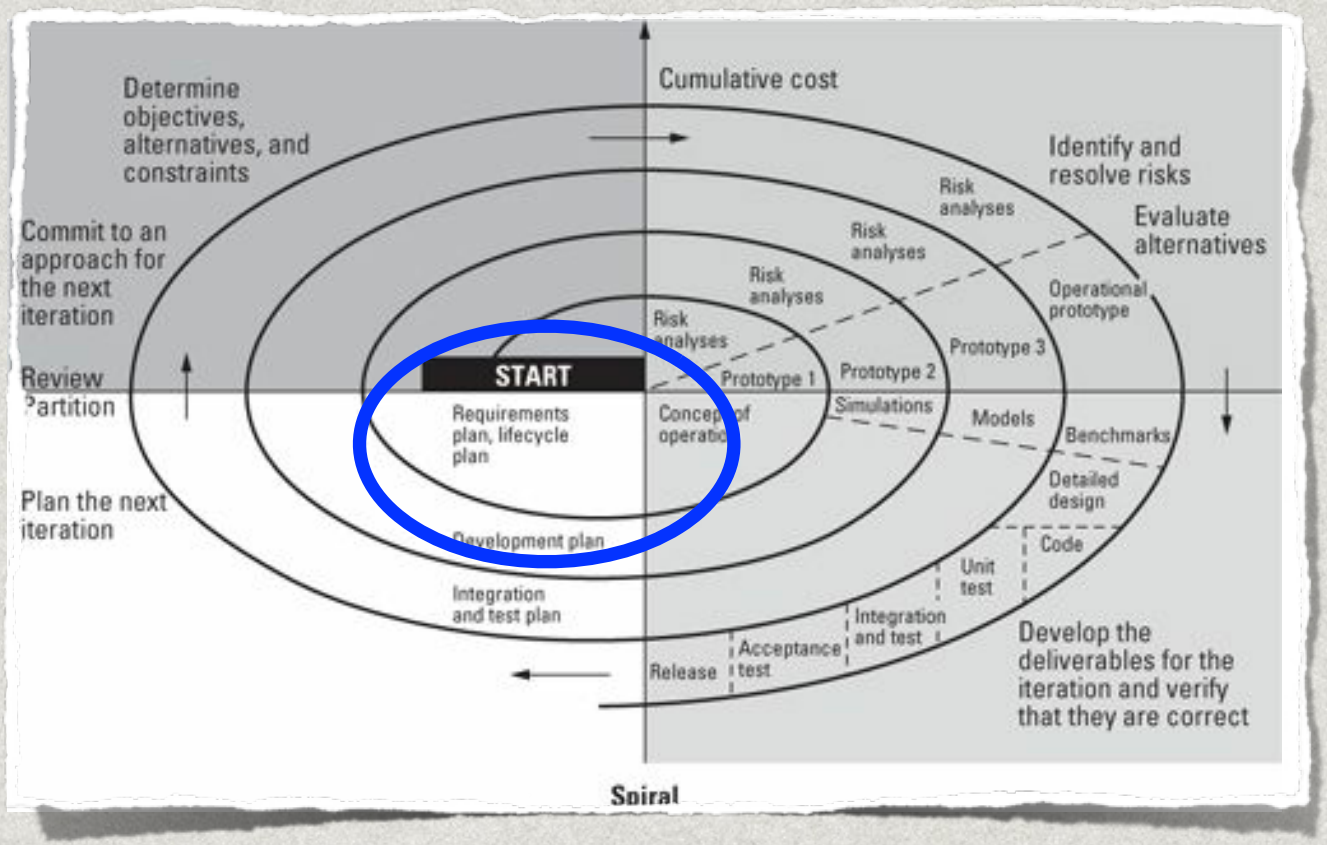


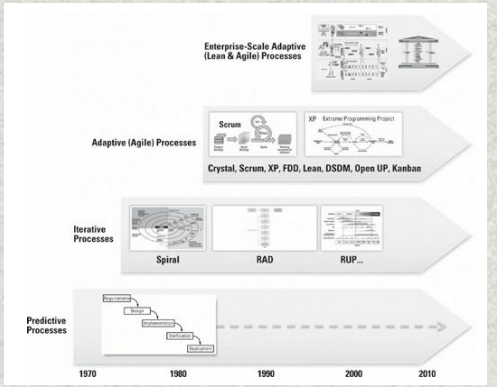
AUT COMPUTING + MATHEMATICAL SCIENCES

Spiral

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Discovery Based Processes Iterative and Incremental



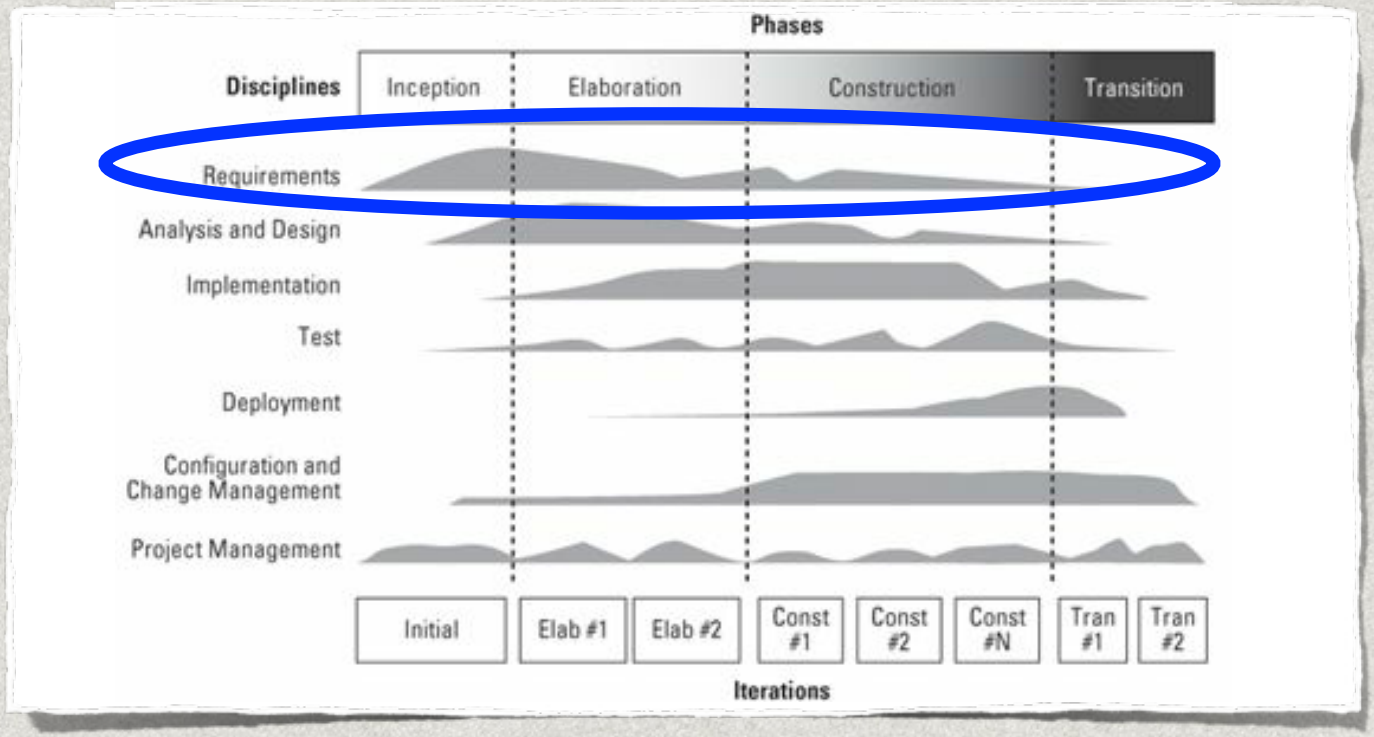


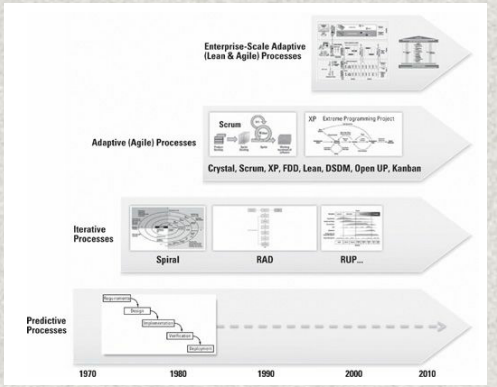
AUT COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Recognise overlapping activities.
- Move away from big up-front requirements and design.
- Based on Spiral- for large-scale projects
- Agile RUP and openUP are recent lightweight versions





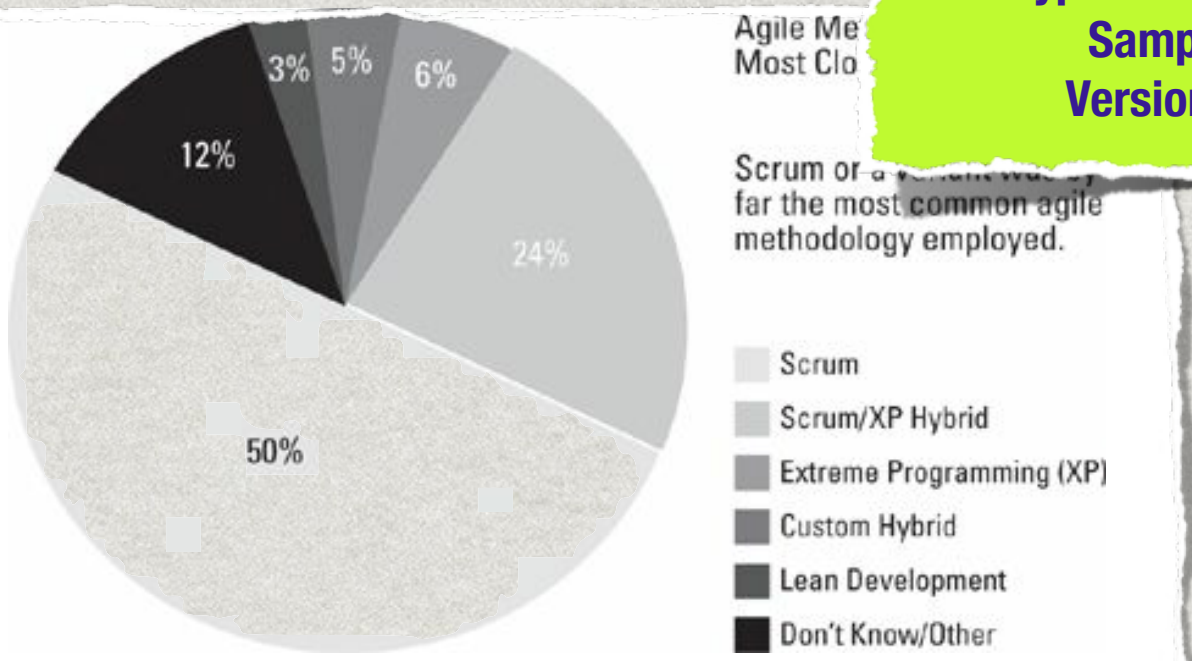
Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

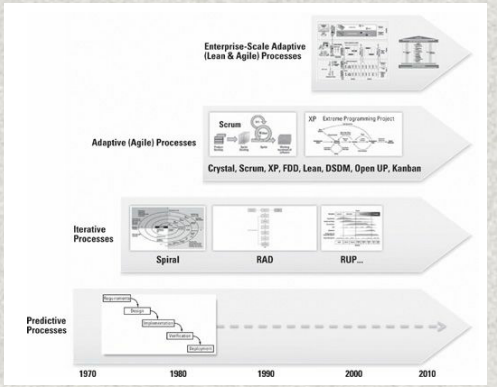
Individuals and interactions	over	
Customer collaboration	over	
Working software	over	
Responding to change	over	

Source: Version One's 2009 Agile Methodology Survey

Sample size?
Types of organisations in sample?
Sample selection method?
VersionOne sells Agile tools?



Courtesy of VersionOne, Inc.

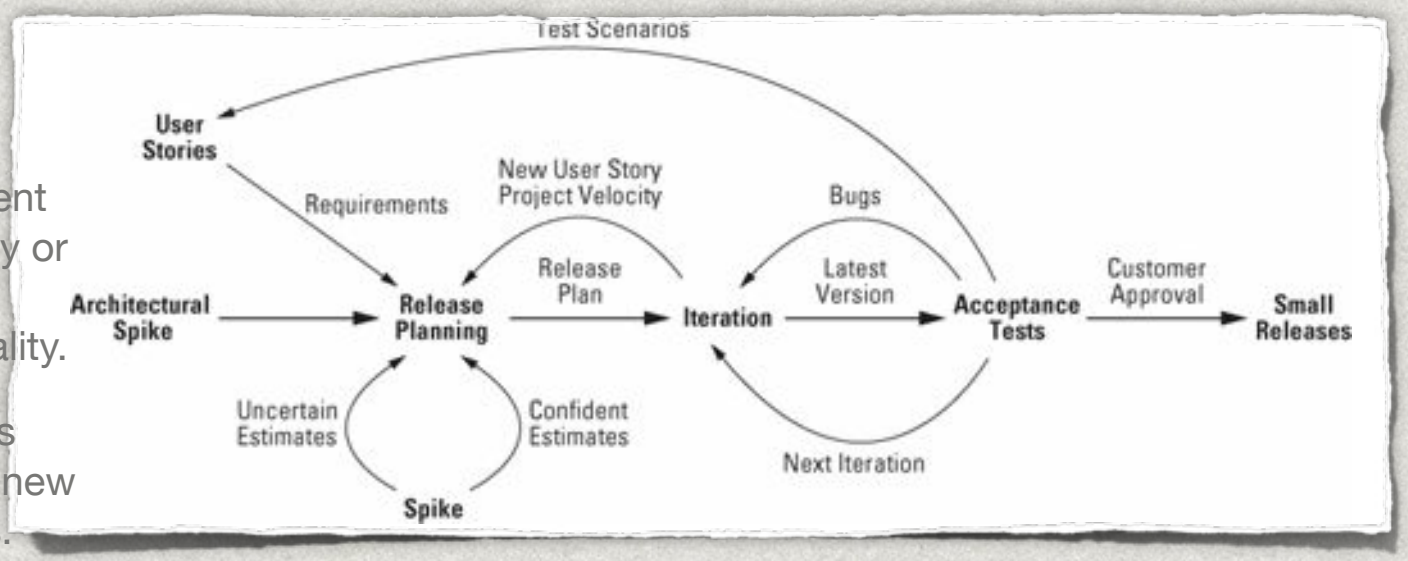


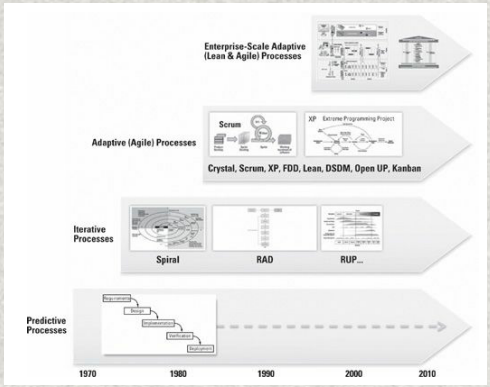
AUT COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.
- Development occurs in frequent builds or iterations, which may or may not be releasable, and delivers incremental functionality.
- Requirements are specified as user stories, each a chunk of new functionality the user requires.
- Programmers work in pairs, follow strict coding standards, and do their own unit testing. Customers participate in acceptance testing.
- Requirements, architecture, and design emerge over the course of the project.



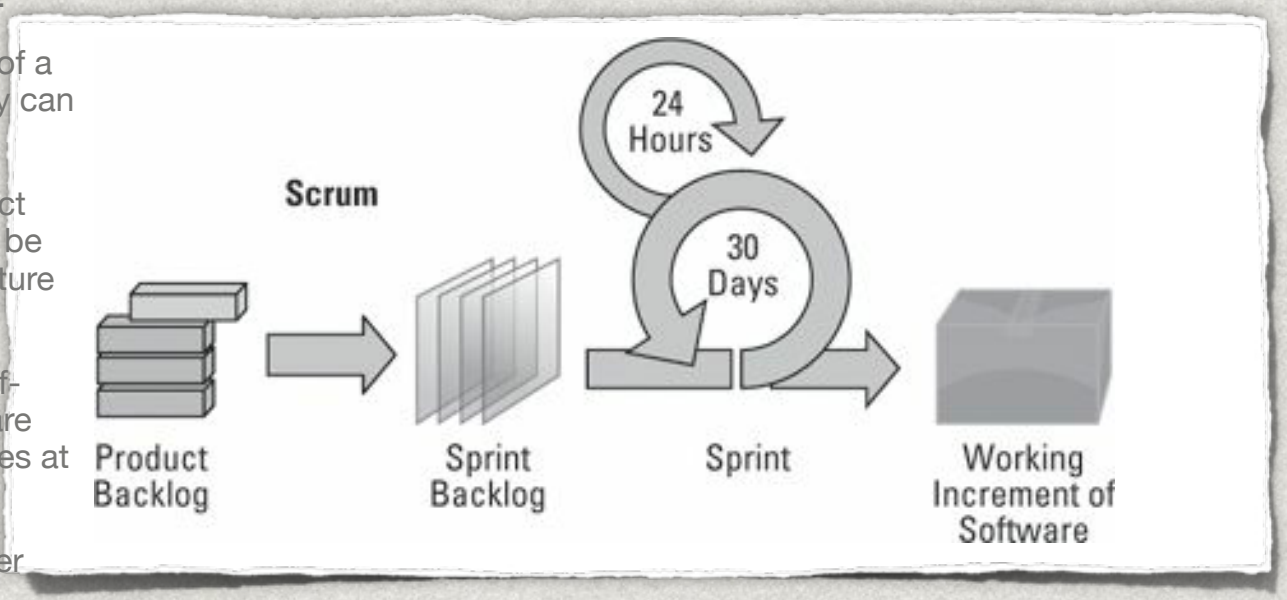


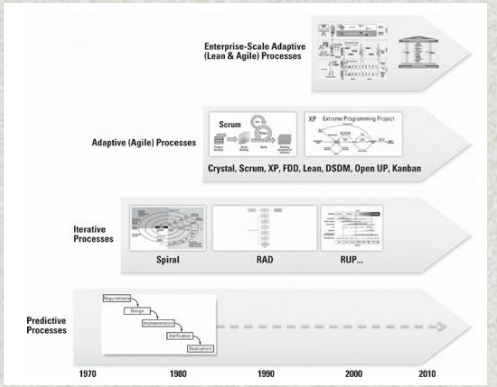
AUT COMPUTING + MATHEMATICAL SCIENCES

Scrum

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Work is done in “sprints,” which are timeboxed iterations of a fixed 30 days or fewer duration.
- Work within a sprint is fixed. Once the scope of a sprint is committed, no additional functionality can be added, except by the development team.
- All work to be done is characterized as product backlog, which includes new requirements to be delivered, the defect workload, and infrastructure and design activities.
- A Scrum Master mentors the empowered, self-organizing, and self-accountable teams that are responsible for delivery of successful outcomes at each sprint.
- A product owner plays the role of the customer proxy.
- A daily stand-up meeting is a primary communication method.
- A heavy focus is placed on timeboxing. Sprints, stand-up meetings, release review meetings, and the like are all completed in prescribed times.
- Typical Scrum guidance calls for fixed 30-day sprints, with approximately 3 sprints per release, thus supporting incremental market releases on a 90-day time frame.





AUT COMPUTING + MATHEMATICAL SCIENCES

(with agile) instead of investing months in building detailed software requirements specifications...teams focus on delivering early, value-added stories into an integrated baseline.

Early delivery serves to test the requirements and architectural assumptions, and it drives risk out by proving or disproving assumptions about integration of features and components.

No longer do management and the user community wait breathlessly for months, hoping the team is building the right thing. At worst, the next checkpoint is only a week or so away, and...users may be able to deploy even the earliest iterations in their own working environment.

--Leffingwell, D, (2007) *Scaling Software Agility*

Agile Requirements is Fundamentally Different

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

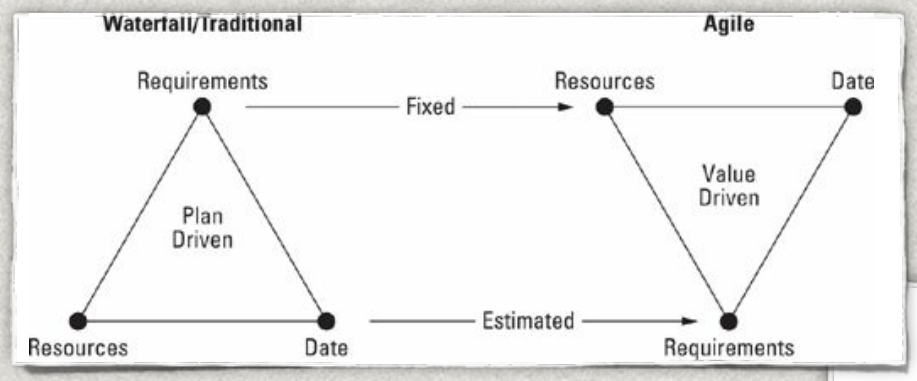


Figure 1-9. Value delivery and ROI in waterfall versus agile

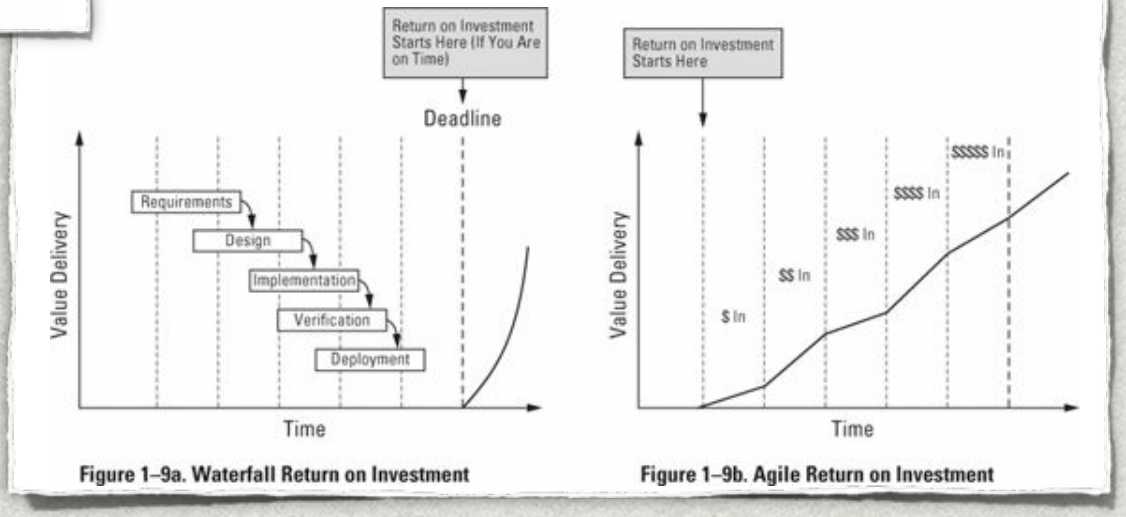
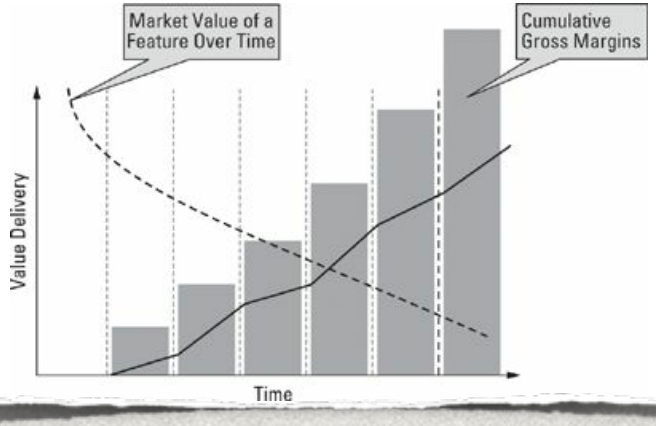


Figure 1–9a. Waterfall Return on Investment

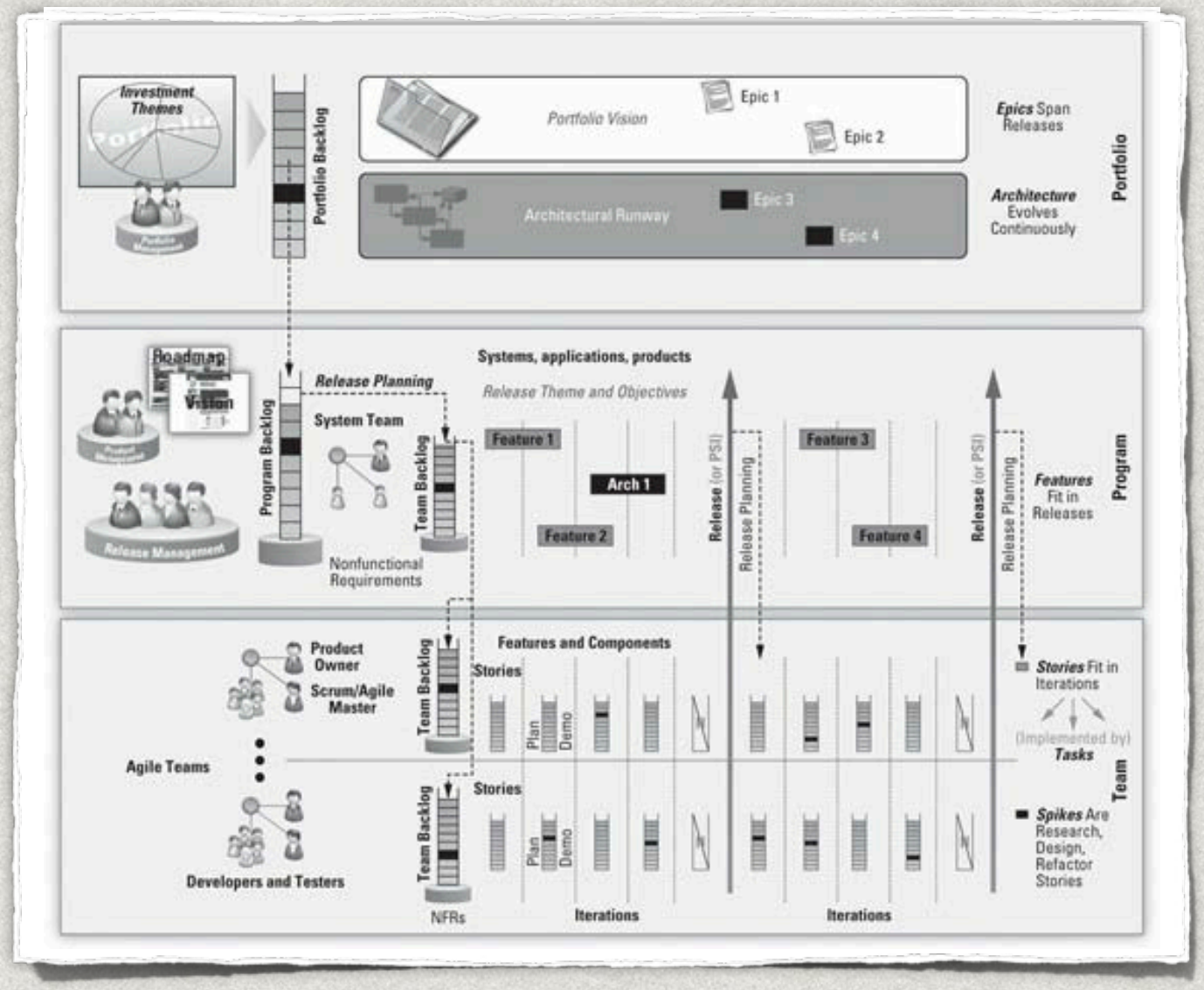
Figure 1–9b. Agile Return on Investment

Figure 1-10. Agile ROI, taking into account differential feature value over time



Agile for the Enterprise

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.



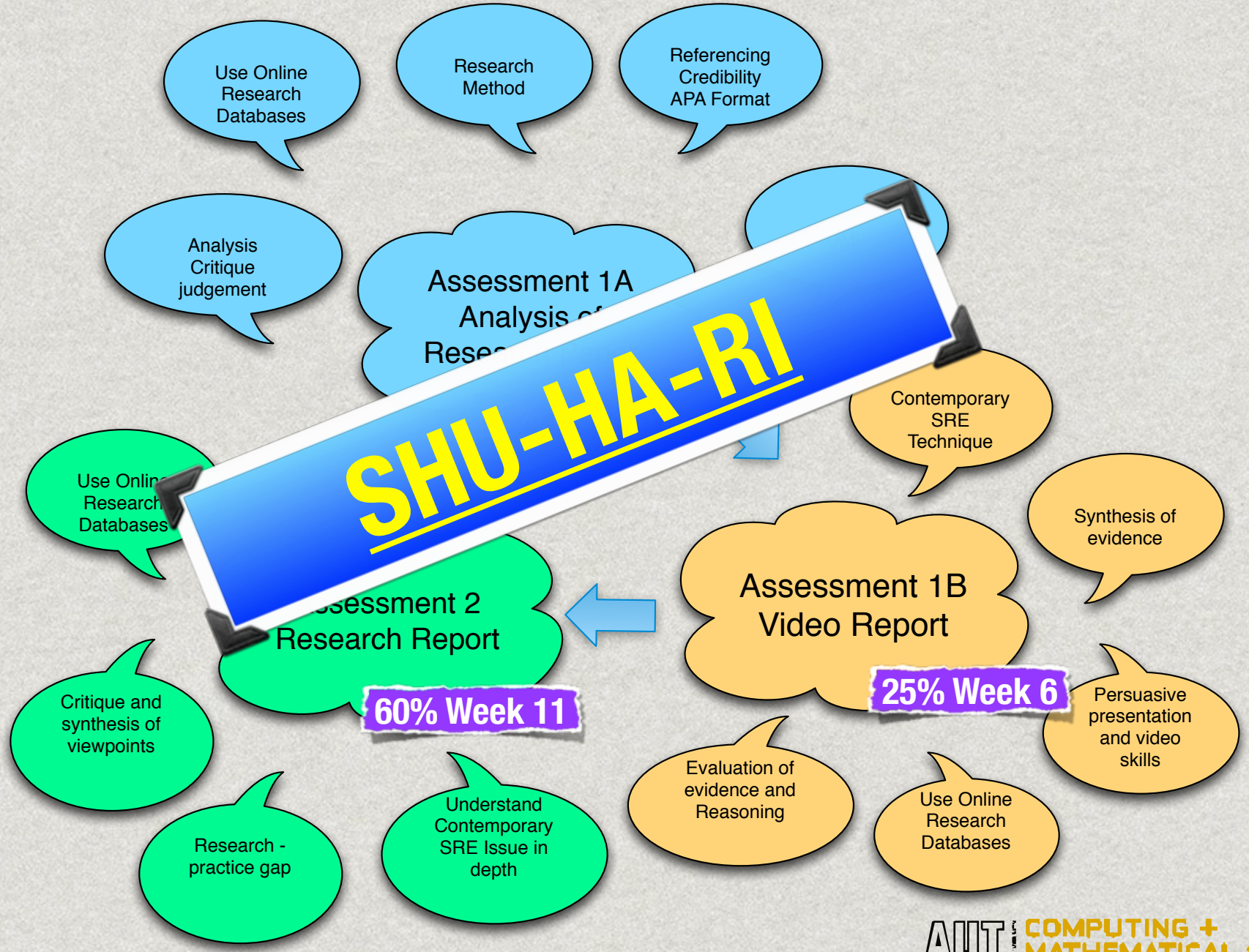
Collaborative and Constructive Learning

Course Overview

- Look at Handbook on AUTOnline
- Get into pairs for Ass1b
  - (Video report)

Guiding learning with the assessments

Guiding learning with the assessments



AUT | COMPUTING + MATHEMATICAL SCIENCES

Assignment 1

Reading for next week

Summary of Today