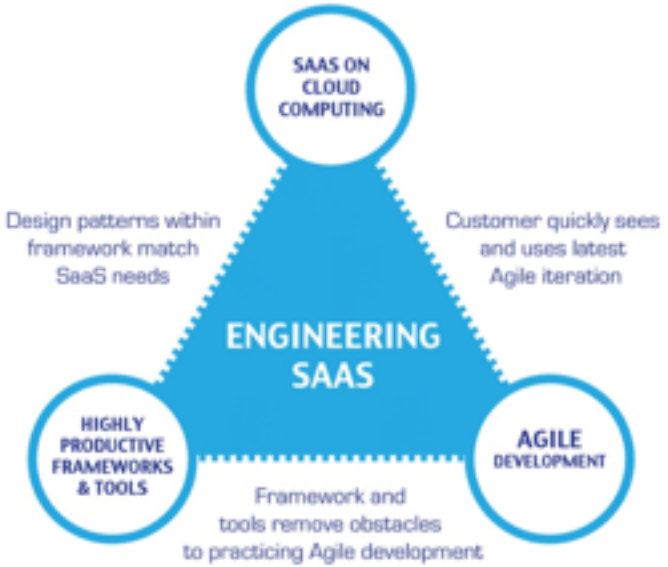
AUT UNIVERSITY

COMPUTER + MATHEMATICAL SCIENCES



Week 1: Overview of SDM

409232 Software Development Methods

Jim Buchan

SERL

Software Engineering Research Laboratory

Today's Goals

20,000m view of the course and why

Learning collaboratively

What is SaaS

What is Service Oriented Computing

What is Cloud Computing

Setting up the Virtual Machine that will be our development environment

Based on Chapter 1 of the Textbook

Today's Goals

20,000m view of the course and why

Learning collaboratively

What is SaaS

What is Service Orient[illegible]puting

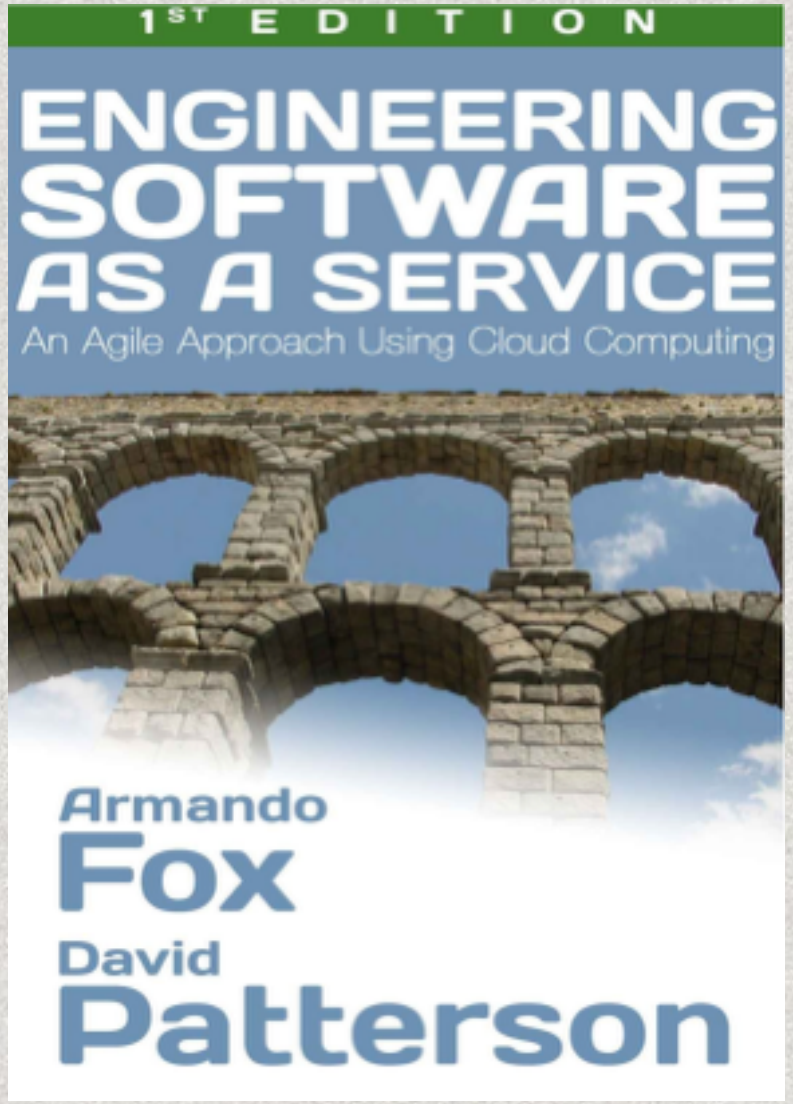
What is Cloud C[illegible]g

Setting up [illegible]dual Machine that will be our dev[illegible]t environment

You will need a pen and paper for all lectures

Based on Chapter 1 of the Textbook

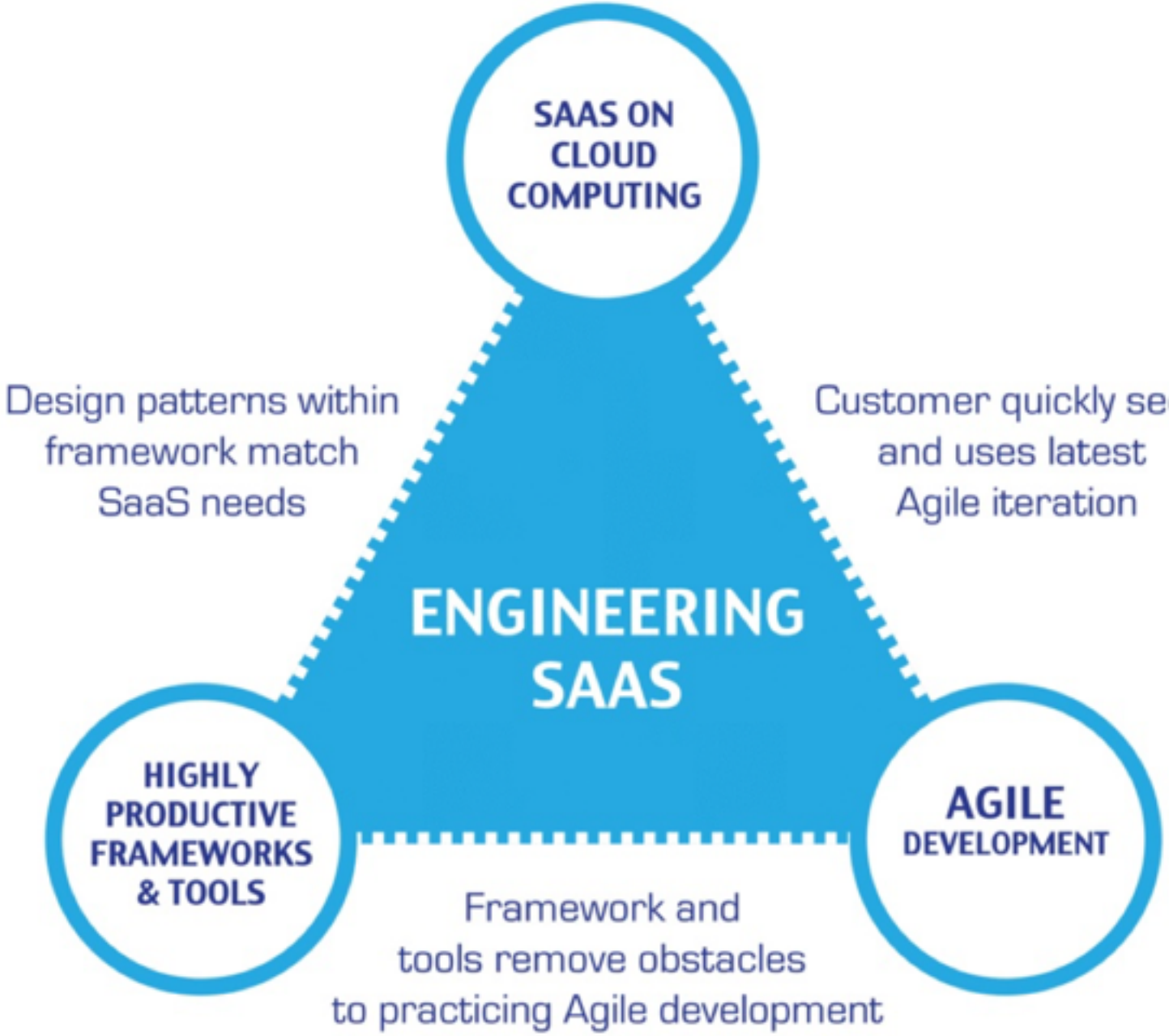
The Course



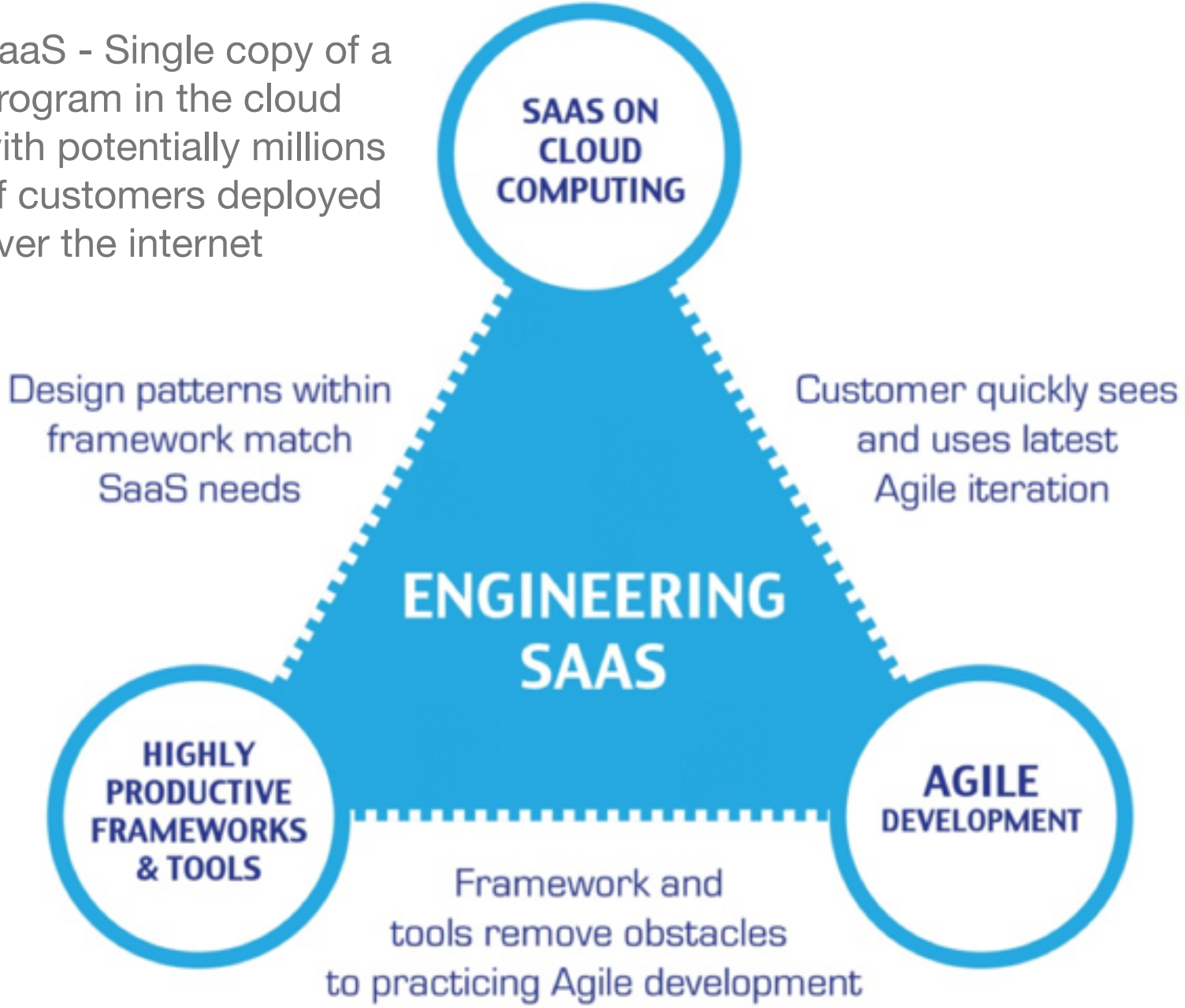
The course will be based on parts of this book.

- Virtual Machine
- Videos
- Exercises

Kindle version $10 from Amazon

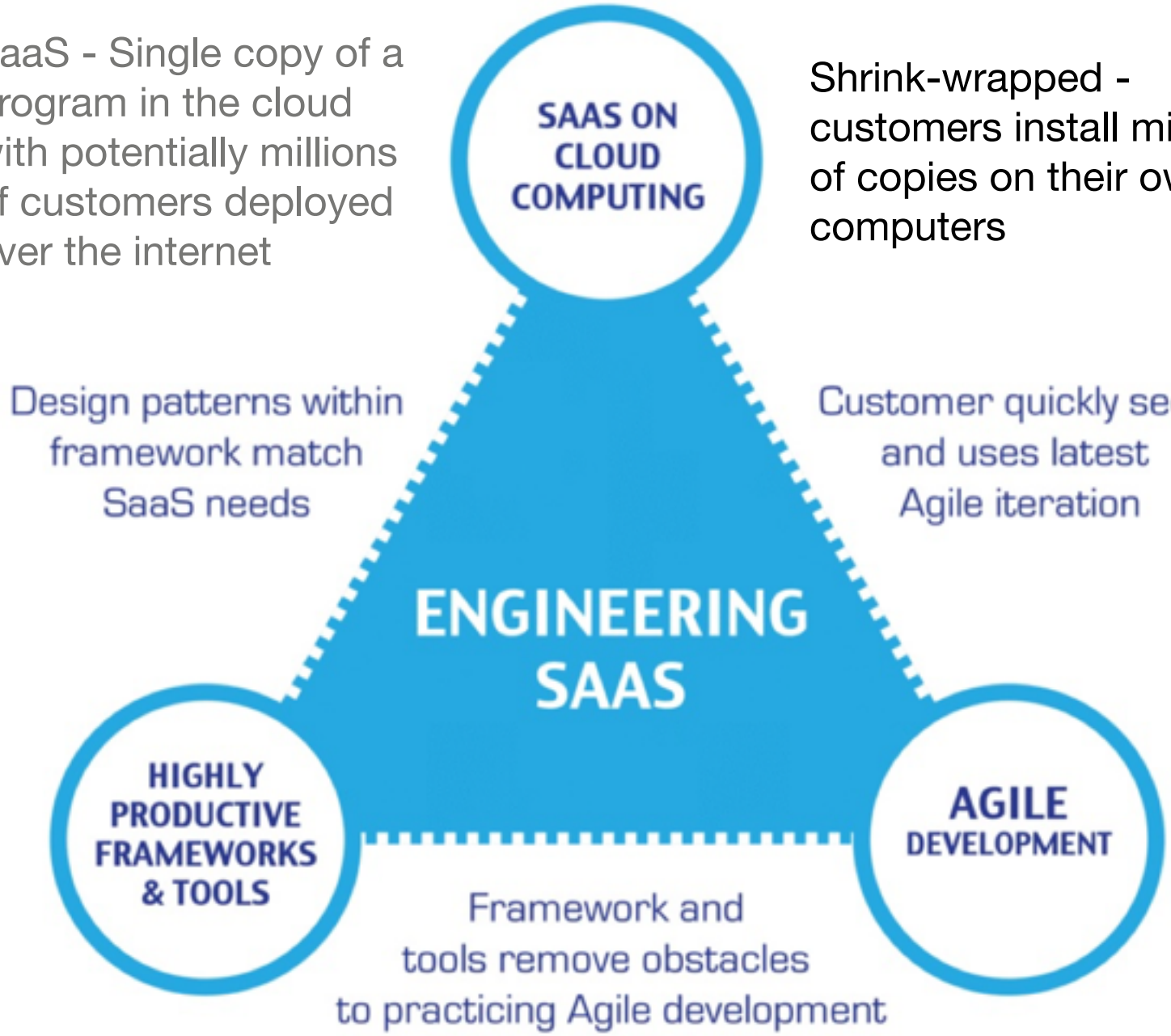

SAAS ON CLOUD COMPUTING
Design patterns within framework match SaaS needs
Customer quickly sees and uses latest Agile iteration
ENGINEERING SAAS
HIGHLY PRODUCTIVE FRAMEWORKS & TOOLS
AGILE DEVELOPMENT
Framework and tools remove obstacles to practicing Agile development

SaaS - Single copy of a program in the cloud with potentially millions of customers deployed over the internet



SaaS - Single copy of a program in the cloud with potentially millions of customers deployed over the internet

Shrink-wrapped - customers install millions of copies on their own computers



SaaS - Single copy of a program in the cloud with potentially millions of customers deployed over the internet

Shrink-wrapped - customers install millions of copies on their own computers

Easy to deploy new features quickly and incrementally

SaaS evolves more rapidly than shrink-wrapped software

Change is normal rather than the exception

SaaS - Single copy of a program in the cloud with potentially millions of customers deployed over the internet

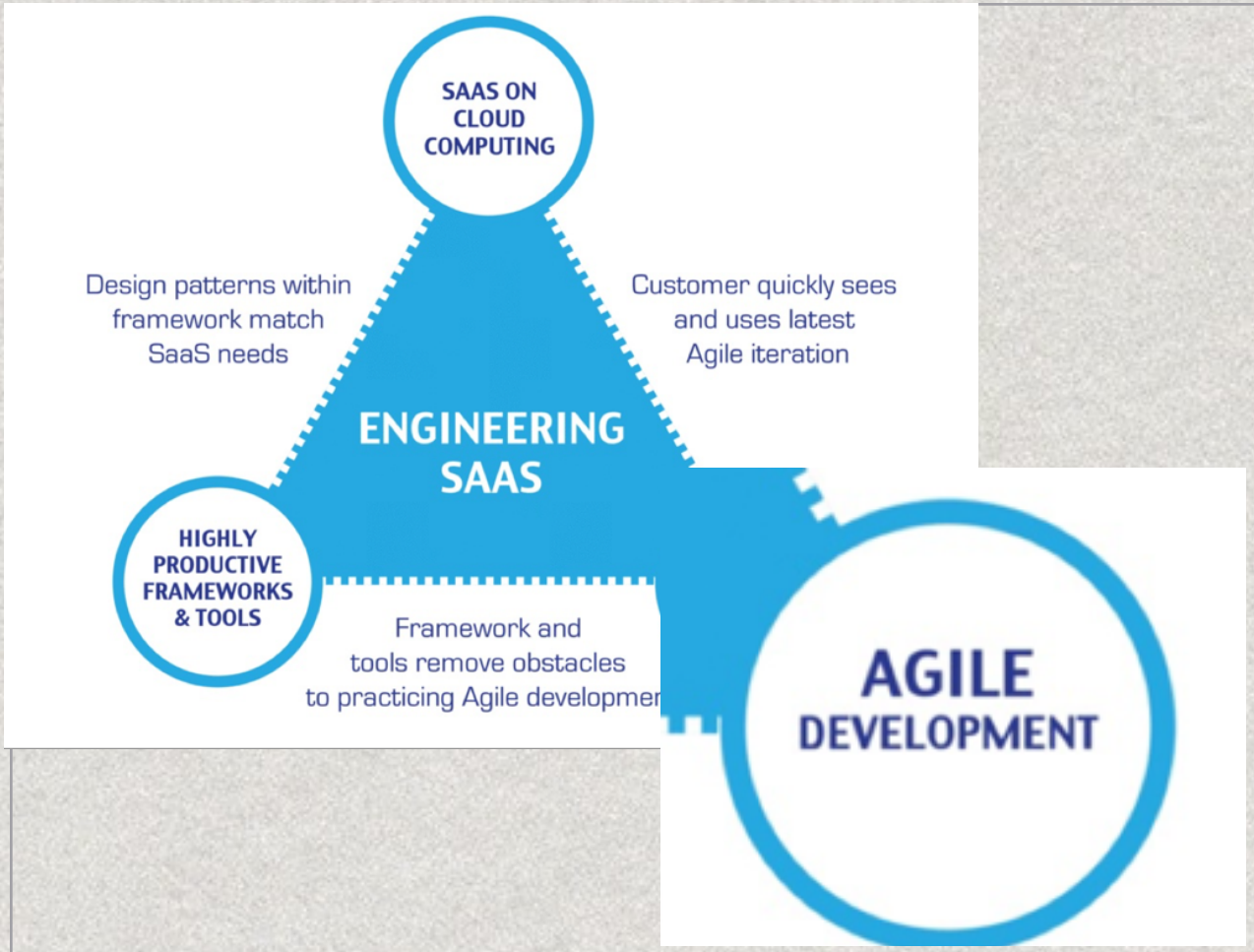
Shrink-wrapped - customers install millions of copies on their own computers

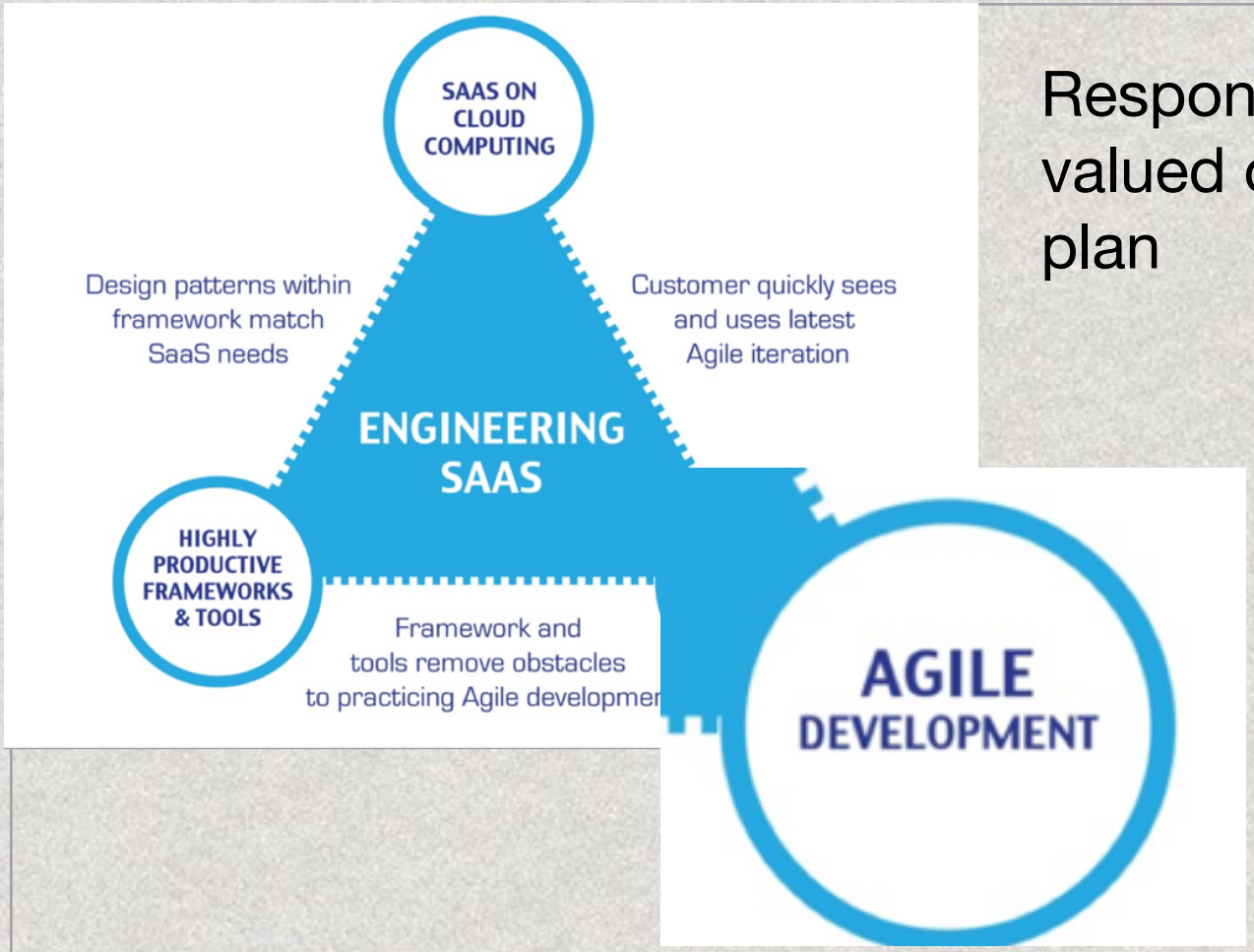
Easy to deploy new features quickly and incrementally

SaaS evolves more rapidly than shrink-wrapped software

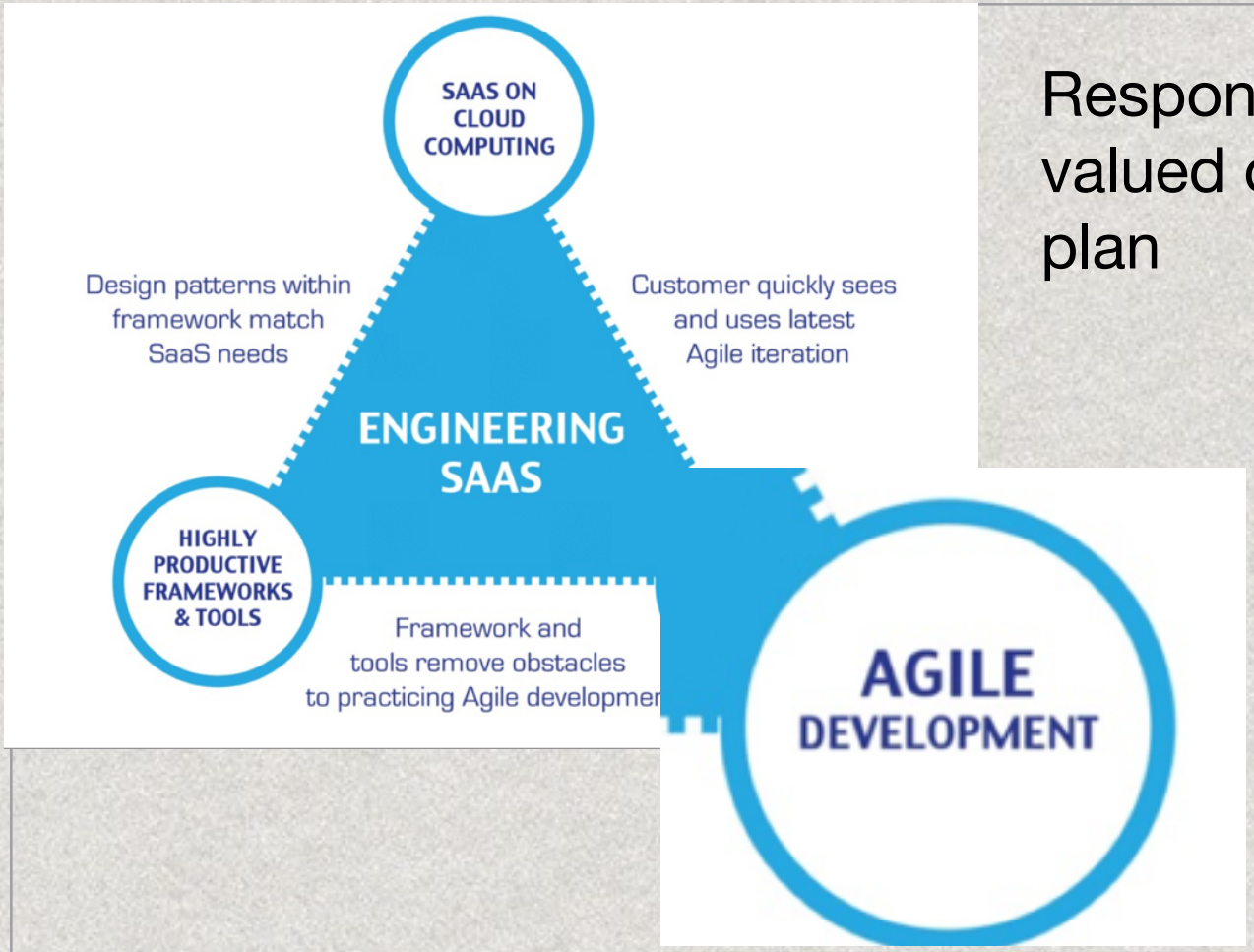
Change is normal rather than the exception

We need a software development methodology that fits this type of rapid change and deployment


SAAS ON CLOUD COMPUTING
Design patterns within framework match SaaS needs
Customer quickly sees and uses latest Agile iteration
ENGINEERING SAAS
HIGHLY PRODUCTIVE FRAMEWORKS & TOOLS
Framework and tools remove obstacles to practicing Agile developmer
AGILE DEVELOPMENT



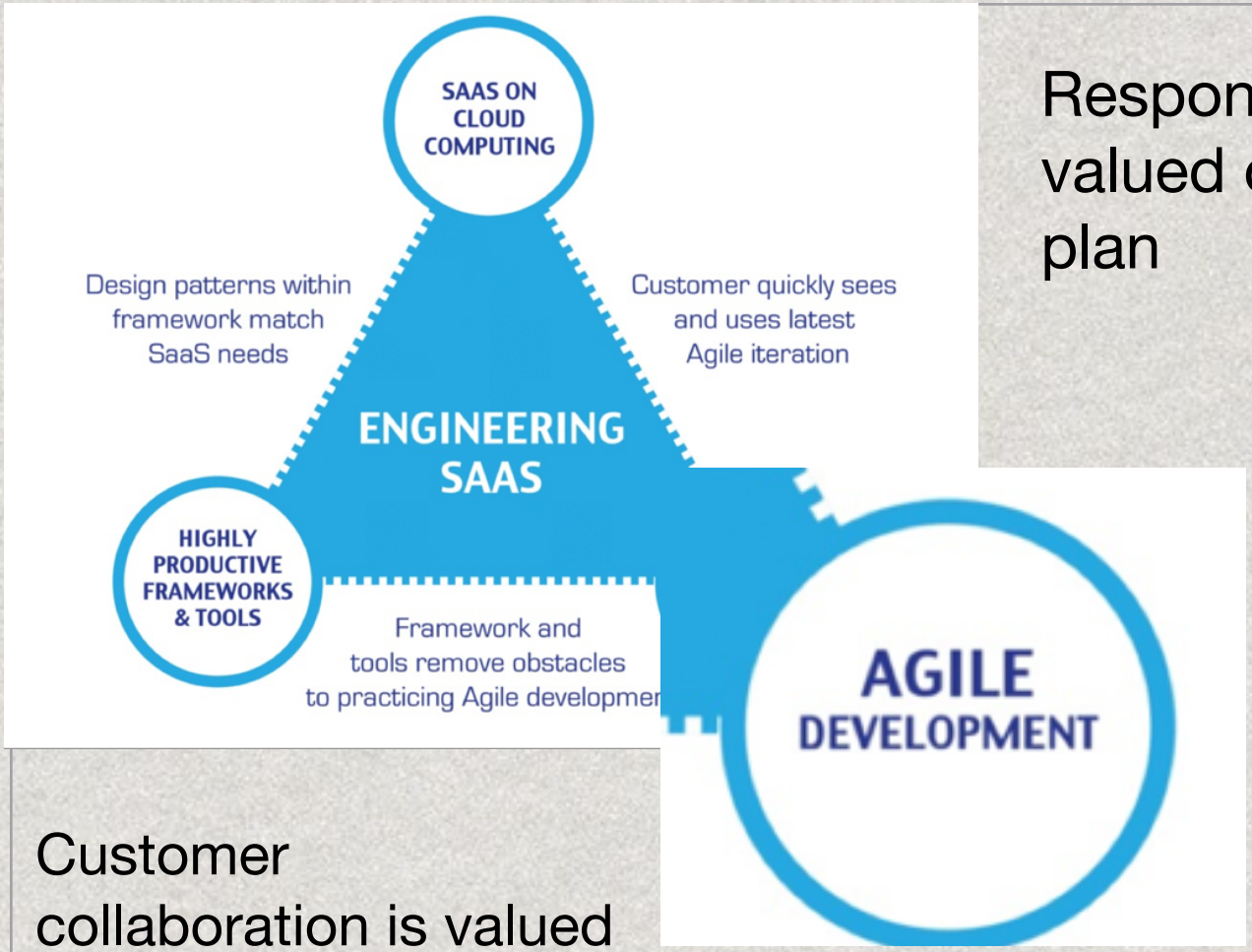
Responding to change is valued over following a plan



Responding to change is valued over following a plan

Plan-and-Document approaches focus more on getting predictability by stages of big effort.

- Waterfall
- Spiral
- Rational Unified Process



Responding to change is valued over following a plan

Plan-and-Document approaches focus more on getting predictability by stages of big effort.

- Waterfall
- Spiral
- Rational Unified Process

Customer collaboration is valued more than contract negotiation

Agile Principles

Organizational

1 Put the customer at the center.

2 Let the team self-organize.

3 Work at a sustainable pace.

4 Develop minimal software:

- 4.1 Produce minimal functionality.
- 4.2 Produce only the product requested.
- 4.3 Develop only code and tests.

5 Accept change.

Technical

6 Develop iteratively:

- 6.1 Produce frequent working iterations.
- 6.2 Freeze requirements during iterations.

7 Treat tests as a key resource:

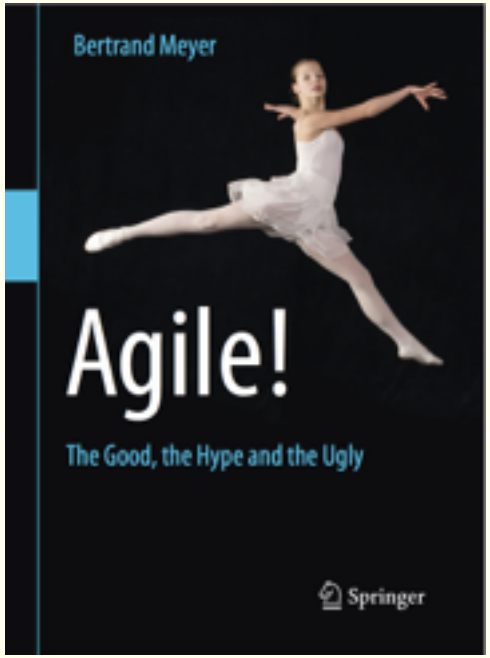
- 7.1 Do not start any new development until all tests pass.
- 7.2 Test first.

8 Express requirements through scenarios.

Agile Principles

Organizational

1 Put the customer at the center.
2 Let the team self-organize.
3 Work at a sustainable pace.
4 Develop minimal software:
  4.1 Produce minimal functionality.
  4.2 Produce only the product requested.
  4.3 Develop only code and tests.
5 Accept change.



Technical

6 Develop iteratively:
  6.1 Produce frequent working iterations.
  6.2 Freeze requirements during iterations.
7 Treat tests as a key resource:
  7.1 Do not start any new development until all tests pass.
  7.2 Test first.
8 Express requirements through scenarios.

Agile Principles

Organizational

1 Put the customer at the center.

2 Let the team self-organize.

3 Work at a sustainable pace.

4 Develop minimal software:

- 4.1 Produce minimal functionality.
- 4.2 Produce only the product requested.
- 4.3 Develop only code and tests.

5 Accept change.

Technical

6 Develop iteratively:

- 6.1 Produce frequent working iterations.
- 6.2 Freeze requirements during iterations.

7 Treat tests as a key resource:

- 7.1 Do not start any new development until all tests pass.
- 7.2 Test first.

8 Express requirements through scenarios.

Agile Principles

Organizational

1 Put the customer at the center.
2 Let the team self-organize.
3 Work at a sustainable pace.
4 Develop minimal software:
  4.1 Produce minimal functionality.
  4.2 Produce only the product requested.
  4.3 Develop only code and tests.
5 Accept change.

Technical

6 Develop iteratively:
  6.1 Produce frequent working iterations.
  6.2 Freeze requirements during iterations.
7 Treat tests as a key resource:
  7.1 Do not start any new development until all tests pass.
  7.2 Test first.
8 Express requirements through scenarios.

Scrum

Extreme Programming (XP)

- **Pair programming**
- **Story Boards**

Kanban

Agile Principles

Organizational

1 Put the customer at the center.
2 Let the team self-organize.
3 Work at a sustainable pace.
4 Develop minimal software:
  4.1 Produce minimal functionality.
  4.2 Produce only the product requested.
  4.3 Develop only code and tests.
5 Accept change.

Technical

6 Develop iteratively:
  6.1 Produce frequent working iterations.
  6.2 Freeze requirements during iterations.
7 Treat tests as a key resource:
  7.1 Do not start any new development until all tests pass.
  7.2 Test first.
8 Express requirements through scenarios.

Scrum

Extreme Programming (XP)

- **Pair programming**
- **Story Boards**

Kanban

Behaviour Driven Development (BDD)

Test Driven Development (TDD)

Continuous Integration

Continuous Delivery

What skills do employers want in their software developers?

What skills do employers want in their software developers?

How to enhance and maintain existing (legacy) code that has little documentation

What skills do employers want in their software developers?

How to enhance and maintain existing (legacy) code that has little documentation

How to work with non-technical people to understand what to build and verify what is being built

What skills do employers want in their software developers?

How to enhance and maintain existing (legacy) code that has little documentation

How to work with non-technical people to understand what to build and verify what is being built

How to work in teams rather than alone to understand what to build and build it.

What skills do employers want in their software developers?

How to enhance and maintain existing (legacy) code that has little documentation

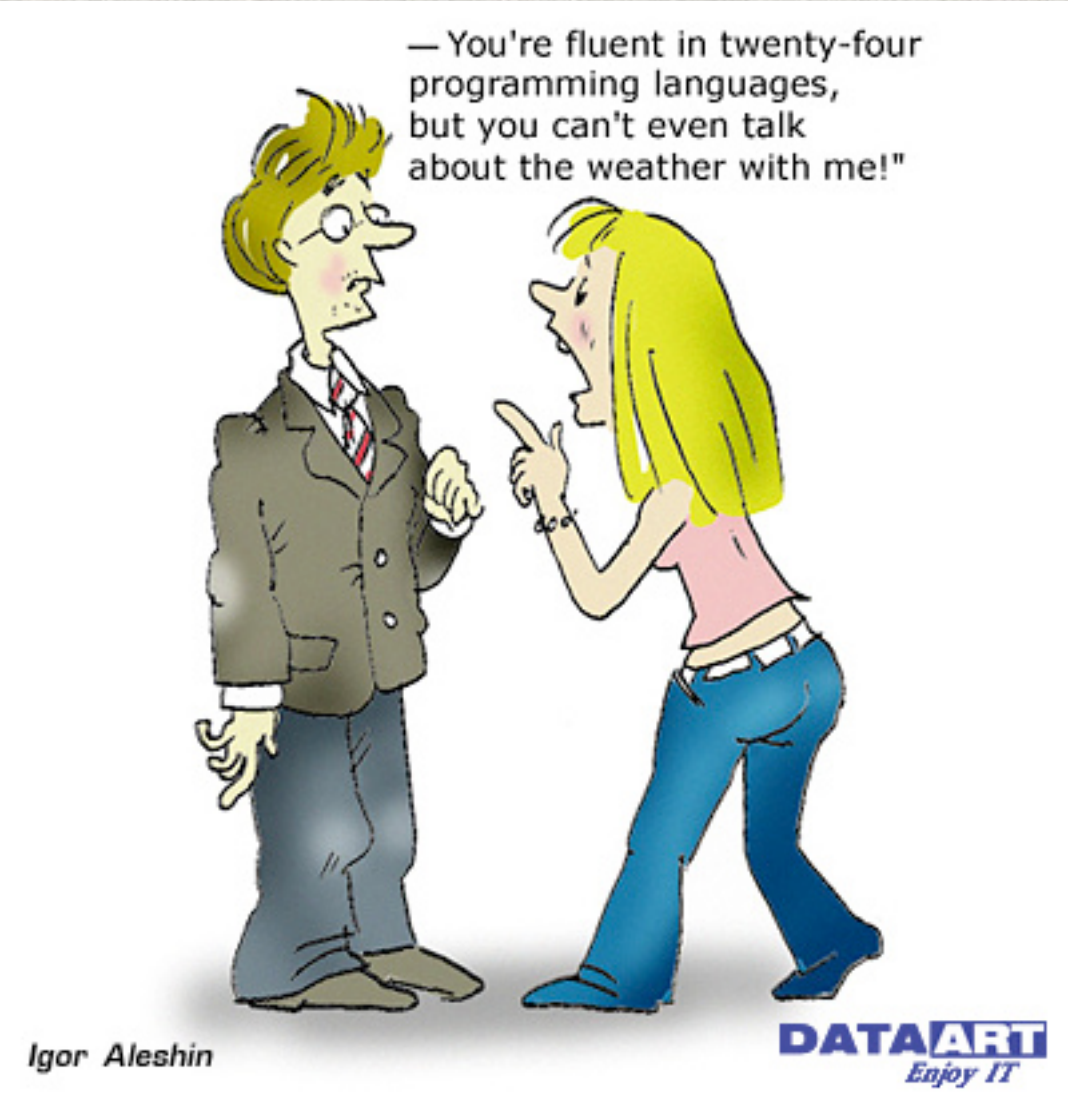
How to work with non-technical people to understand what to build and verify what is being built

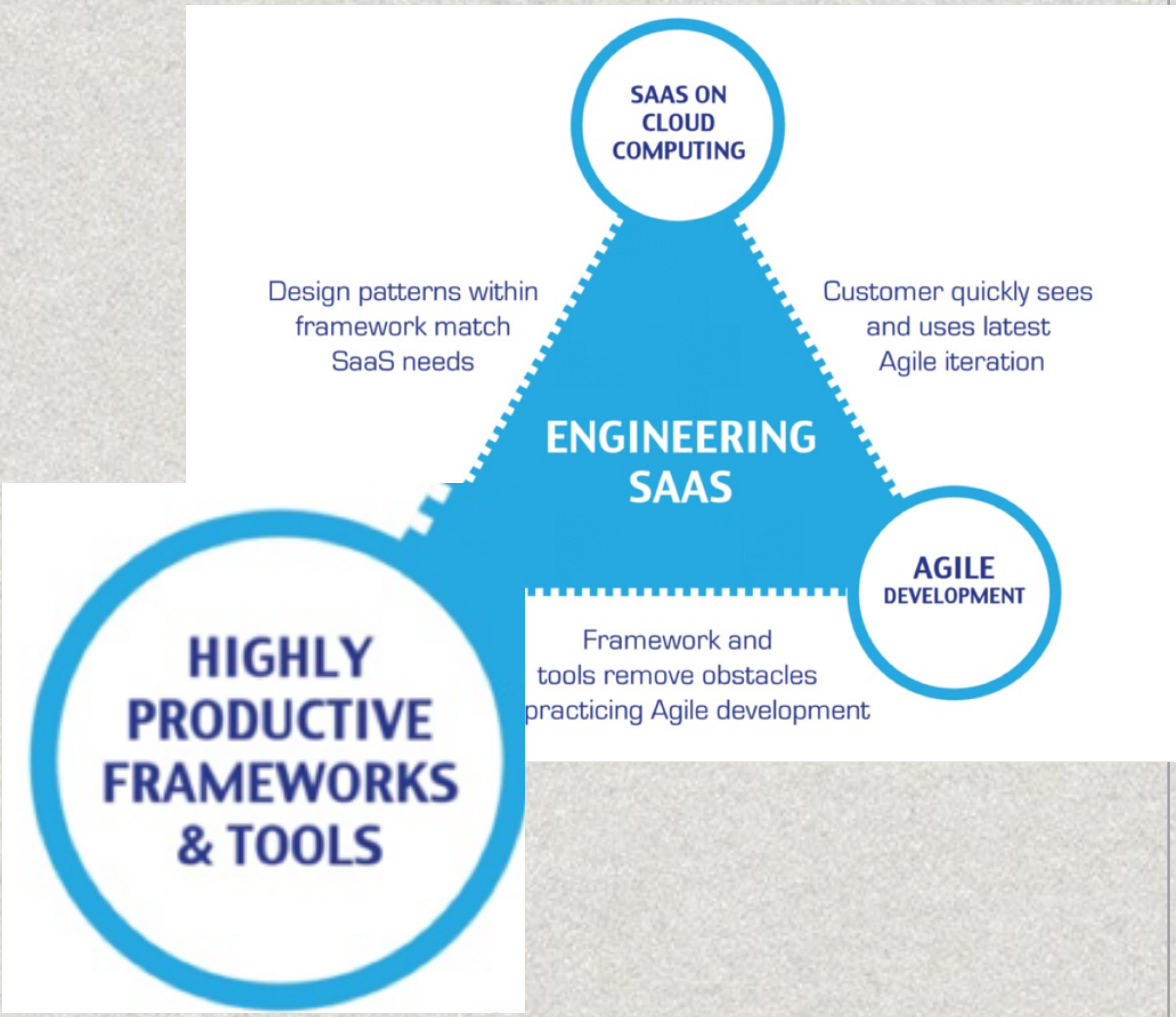
How to work in teams rather than alone to understand what to build and build it.

Treat testing as important as coding

Challenges

Here is what they always forget to tell new testers in the job interview

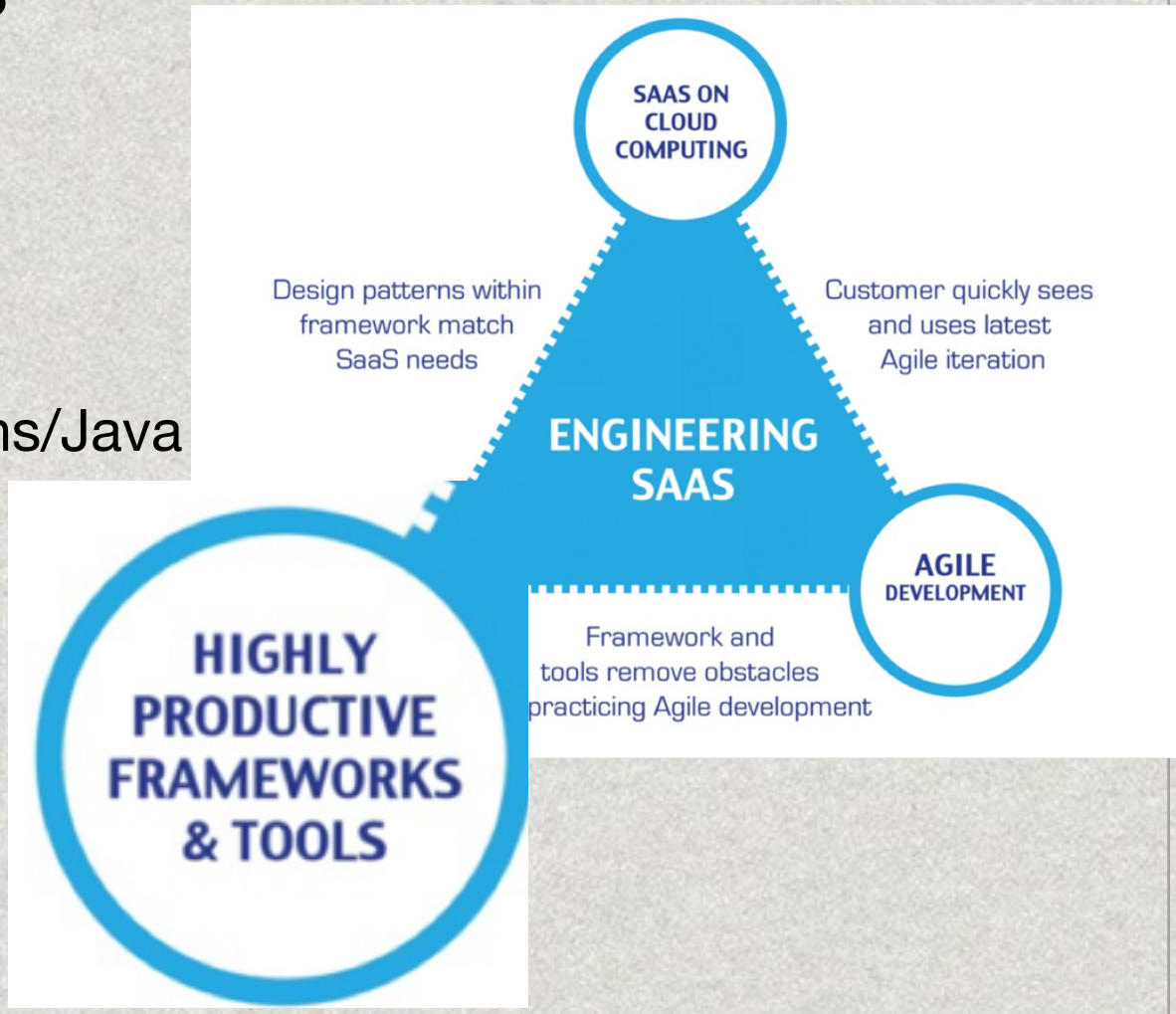

— You're fluent in twenty-four programming languages, but you can't even talk about the weather with me!"
Igor Aleshin
DATAART
Enjoy IT


SAAS ON CLOUD COMPUTING
Design patterns within framework match SaaS needs
Customer quickly sees and uses latest Agile iteration
ENGINEERING SAAS
AGILE DEVELOPMENT
Framework and tools remove obstacles practicing Agile development
HIGHLY PRODUCTIVE FRAMEWORKS & TOOLS

Framework for SaaS Development

Rails/Ruby

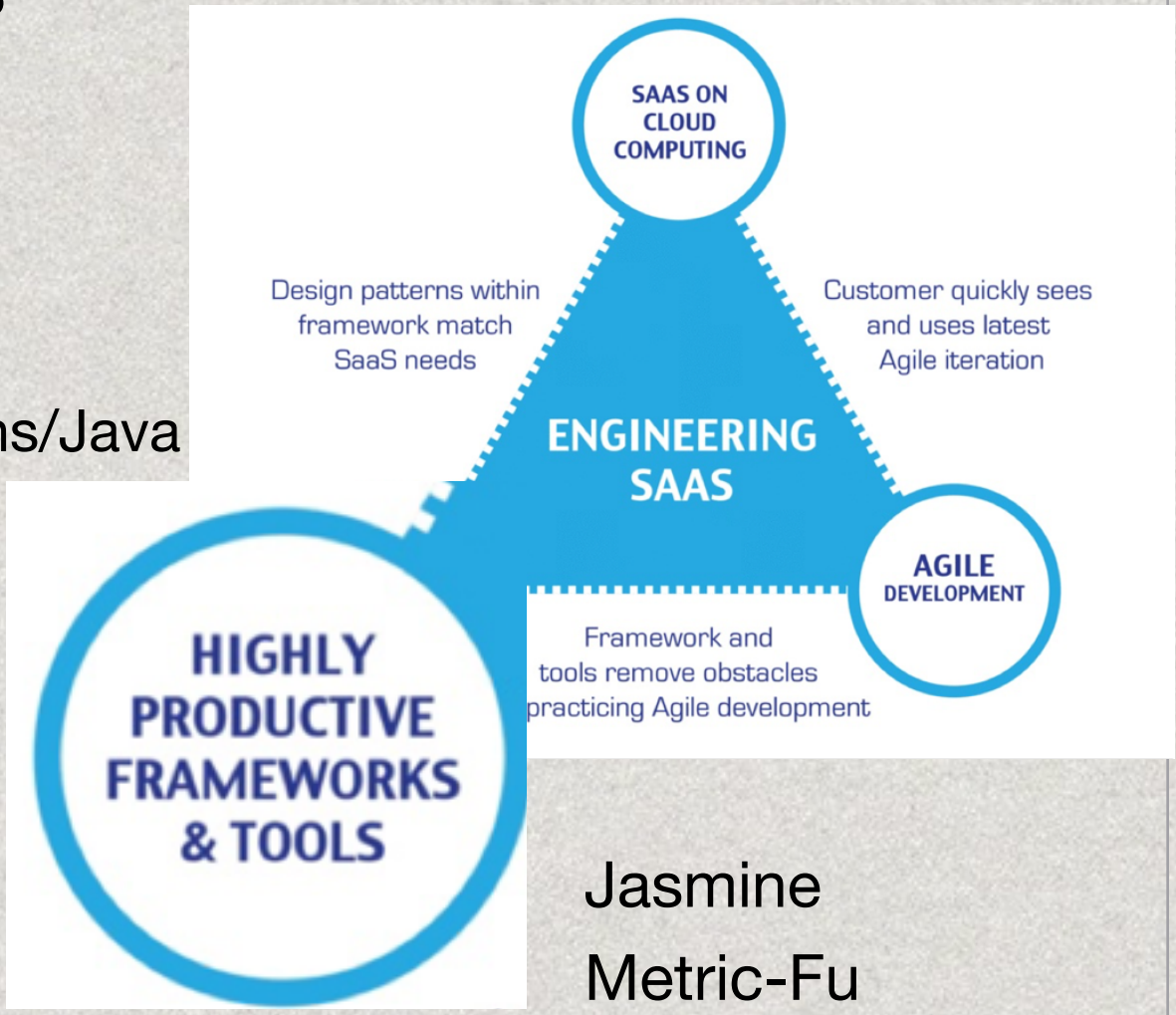
- Django/Python
- Sails/Javascript/
- Enterprise Java Beans/Java



Framework for SaaS Development

Rails/Ruby

- Django/Python
- Sails/Javascript/
- Enterprise Java Beans/Java



- Heroku
- Github
- RSpec
- CodeClimate
- Cucumber

- Jasmine
- Metric-Fu
- SimpleCov
- Factory-Girl

What is Software Engineering?

An early definition was given at the first NATO conference (Naur and Randell, 1968):

Software engineering is the establishment and use of sound ***engineering principles*** *in order to obtain* ***economically*** *software that is* ***reliable*** *and works* ***efficiently*** *on* ***real machines***.

What is Software Engineering?

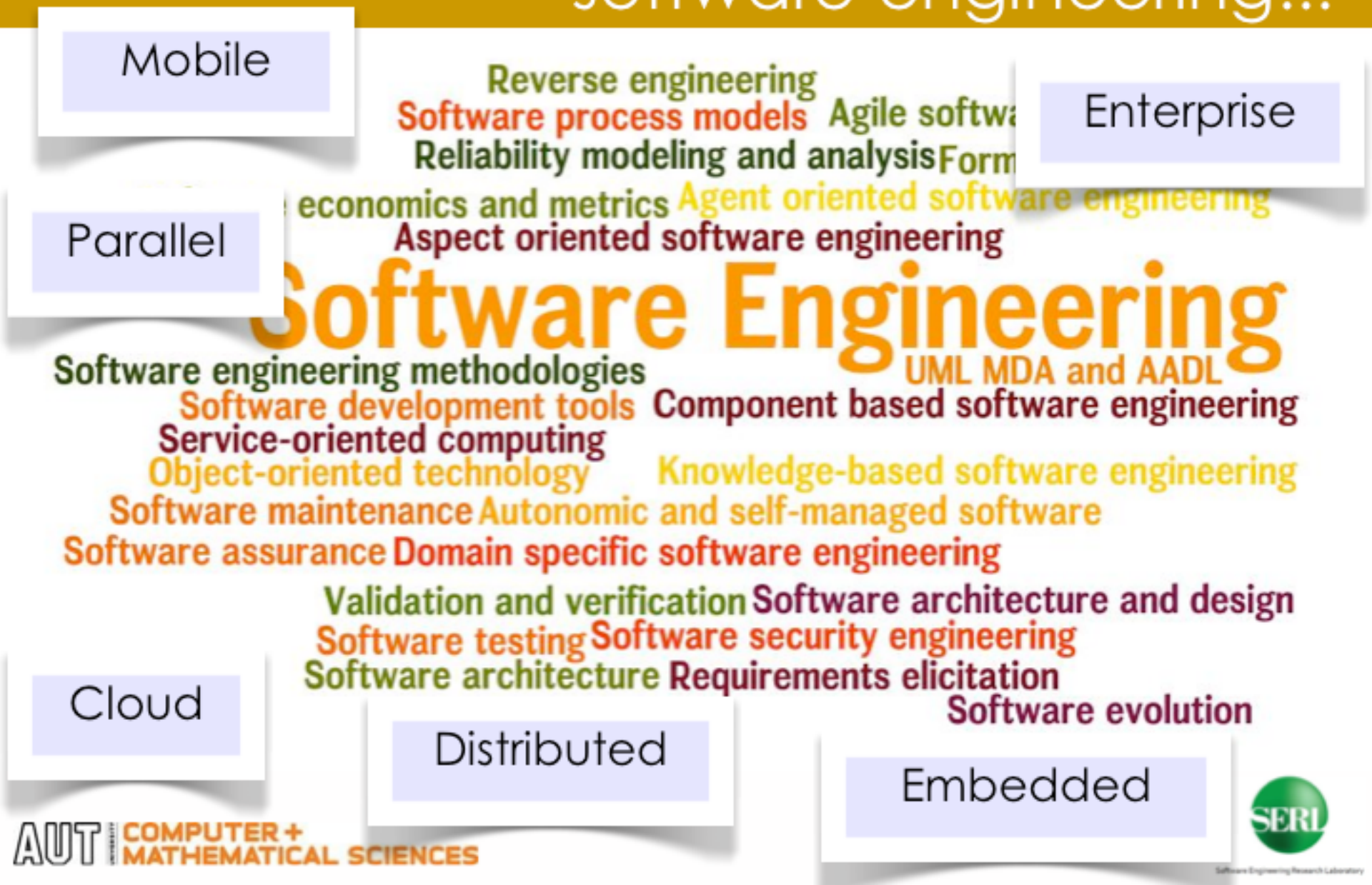
An early definition was given at the first NATO conference (Naur and Randell, 1968):

Software engineering is the establishment and use of sound ***engineering principles*** *in order to obtain* ***economically*** *software that is* ***reliable*** *and works* ***efficiently*** *on* ***real machines***.

The definition given in the IEEE Standard Glossary of Software Engineering Terminology (IEEE610, 1990) is as follows:

Software engineering is the application of a **systematic**, **disciplined**, **quantifiable** approach to the **development**, **operation**, and **maintenance** of software; that is, the **application of engineering** to software.

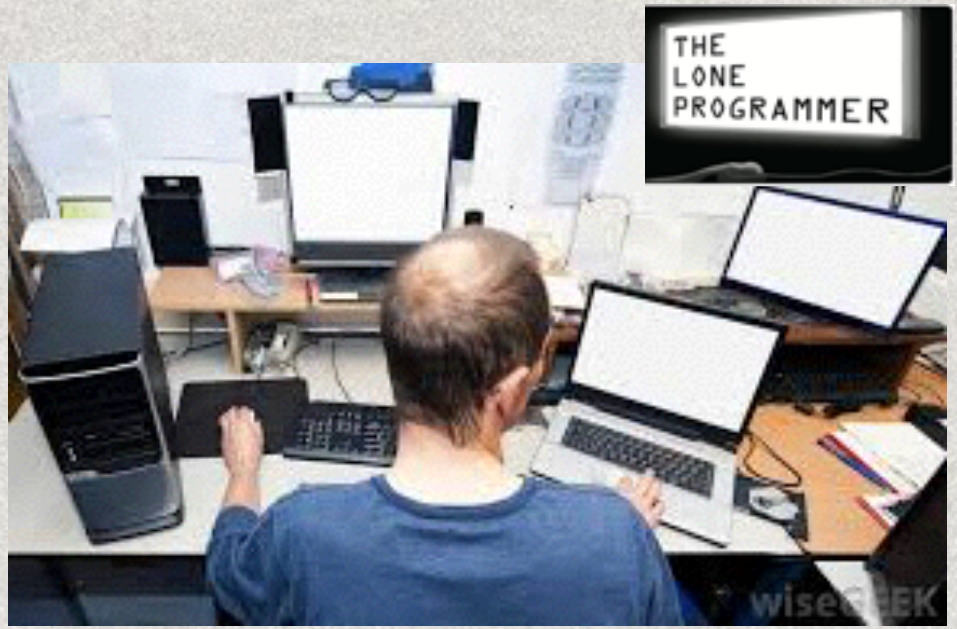
The many faces of software engineering...



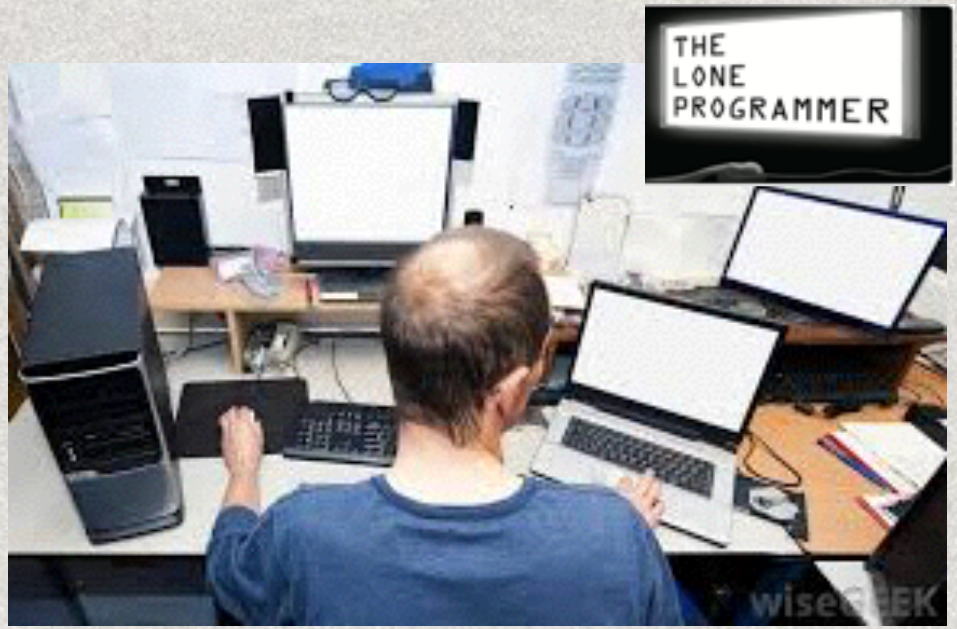
AUT | COMPUTER + MATHEMATICAL SCIENCES

SERL

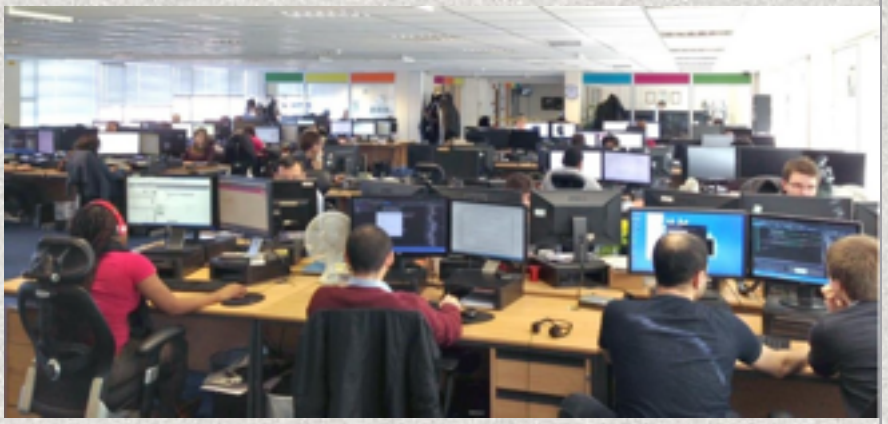
What is Software Engineering?



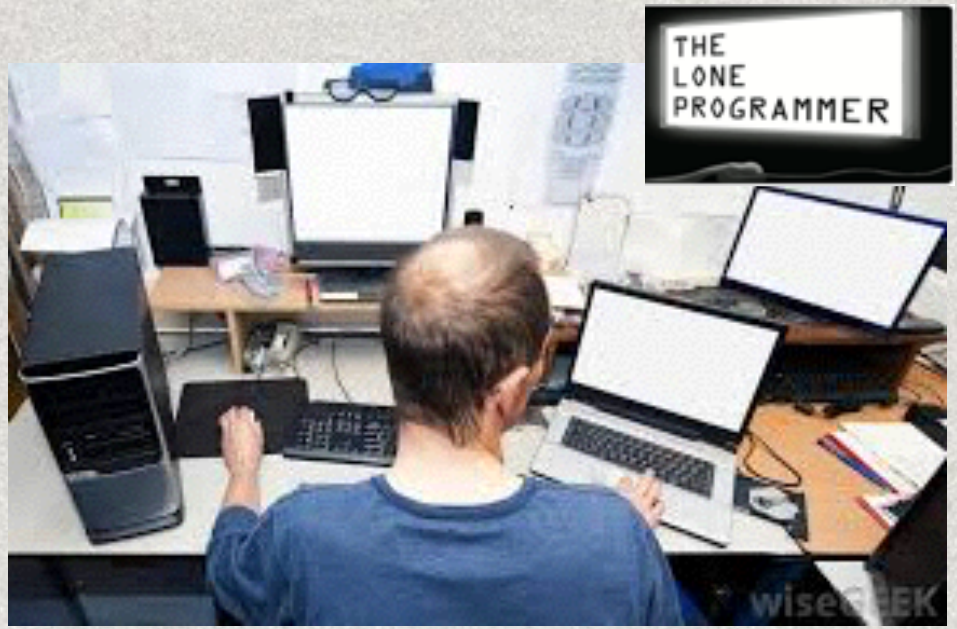
What is Software Engineering?



OR....

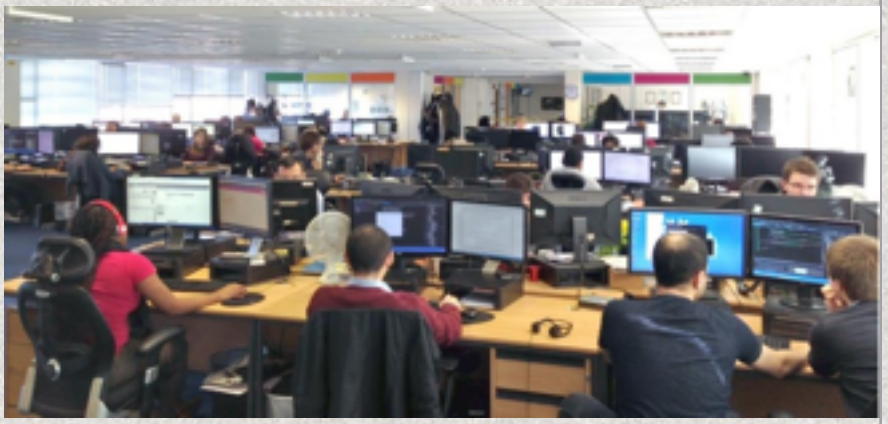


What is Software Engineering?

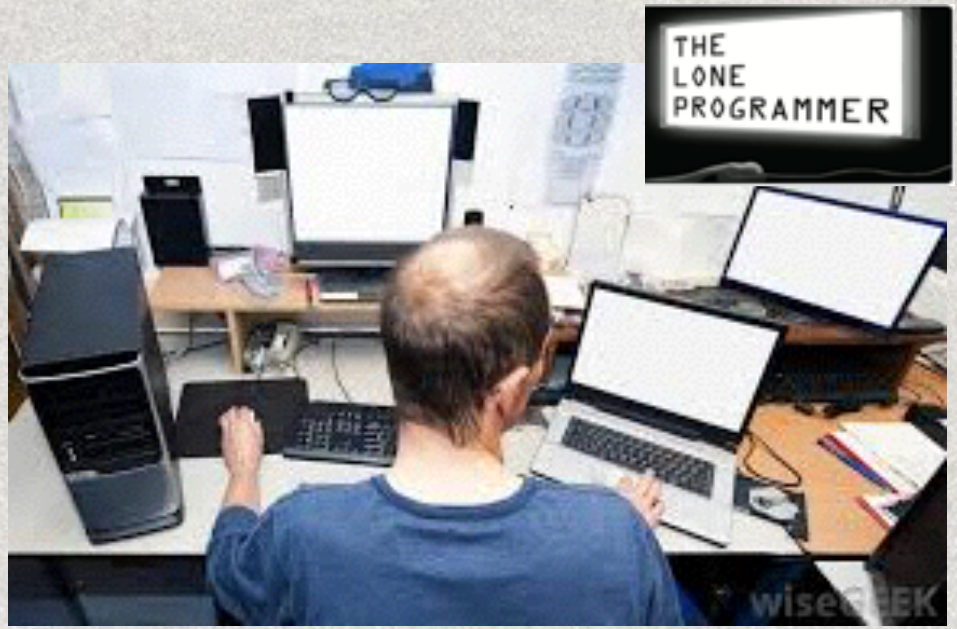


WHY is most software development collaborative now?

OR….



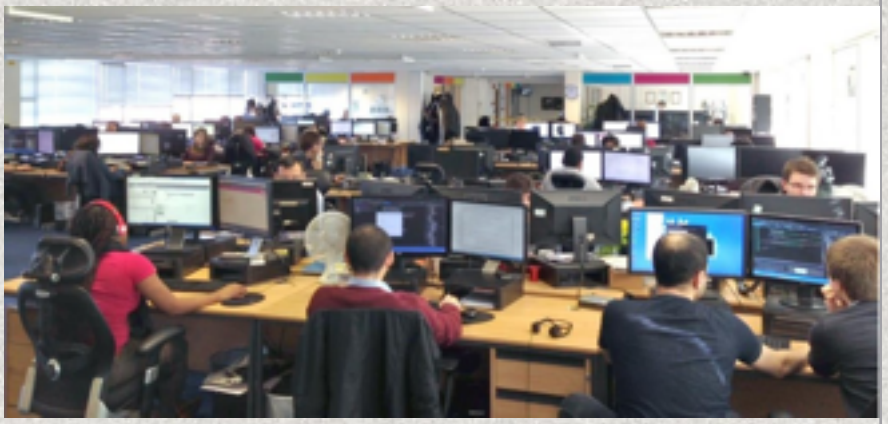
What is Software Engineering?



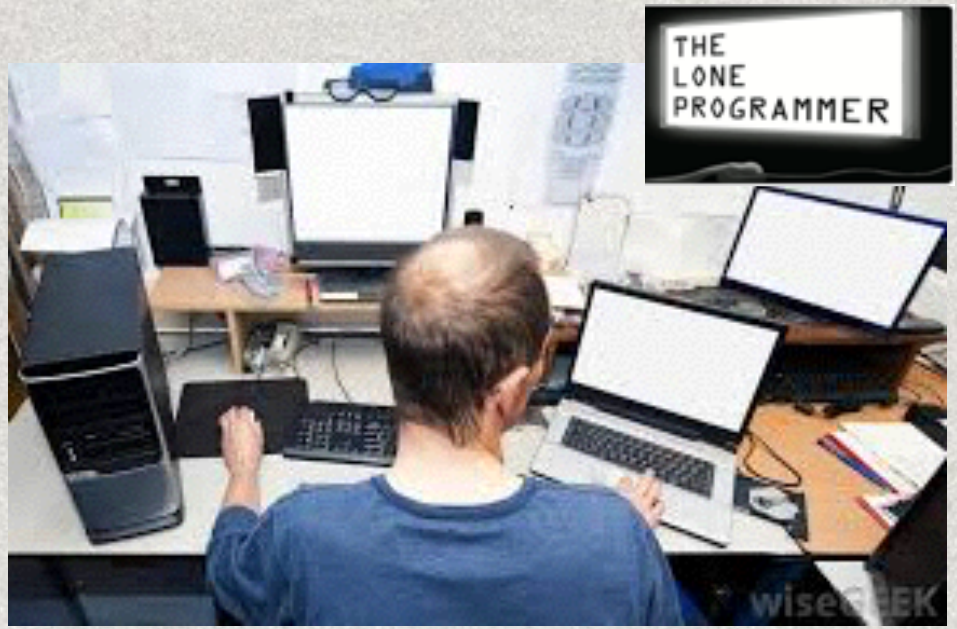
WHY is most software development collaborative now?

What challenges does this add?

OR….



What is Software Engineering?

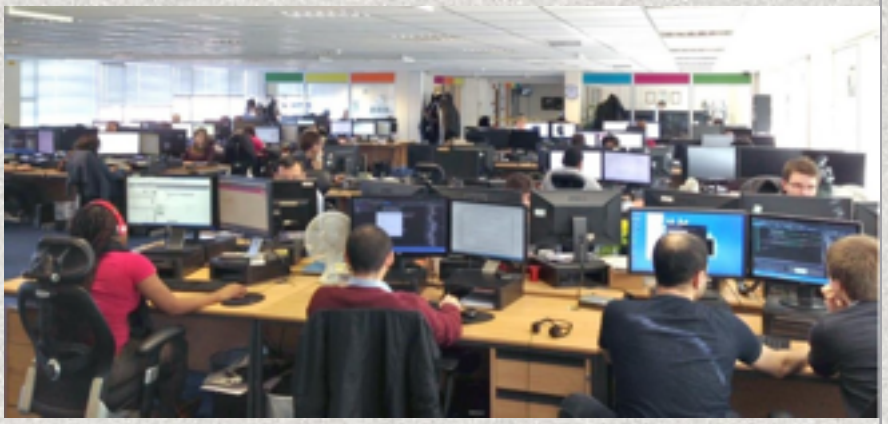


WHY is most software development collaborative now?

What challenges does this add?

Will learning how to do SE collaboratively help me get a…?

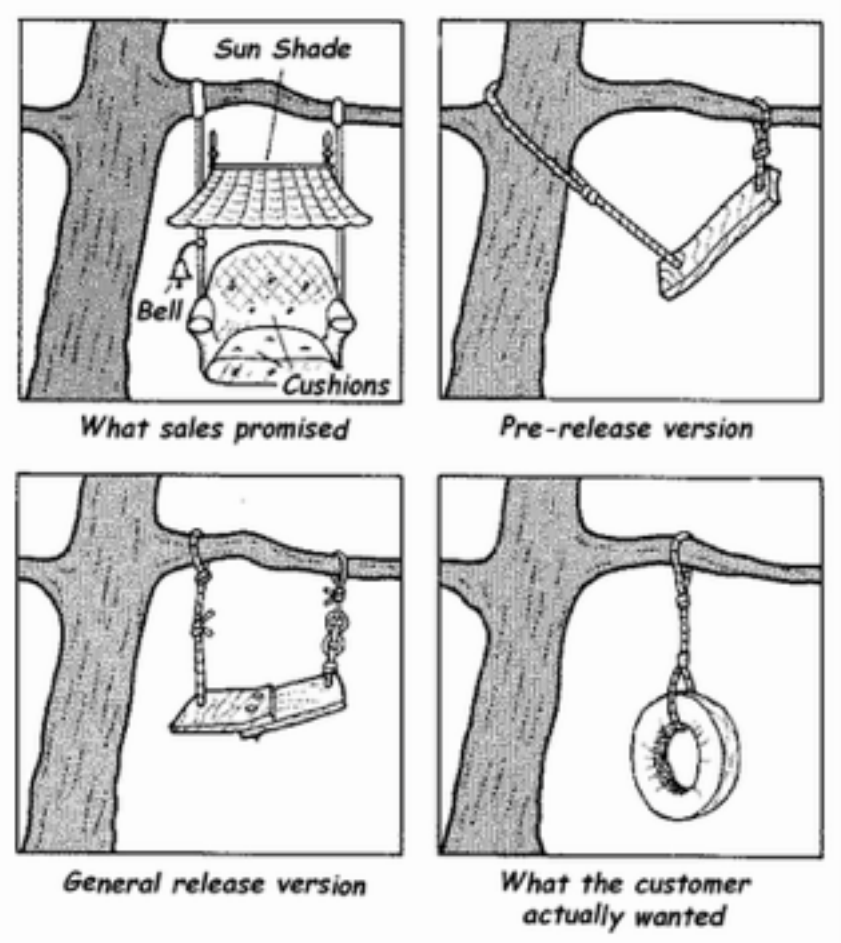
OR….



What are the challenges?

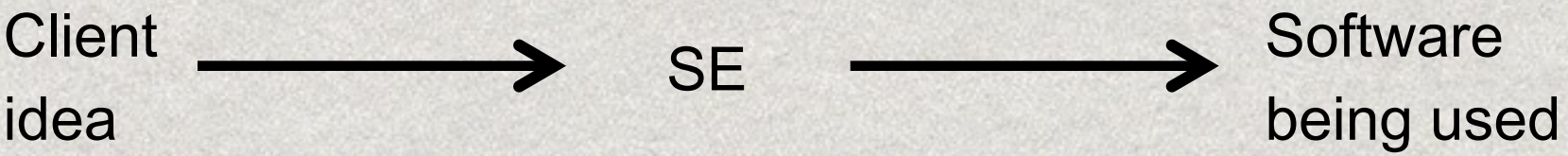
A compass to help keep our team on course

Challenges


Sun Shade
Bell
Cushions
What sales promised
Pre-release version
General release version
What the customer actually wanted

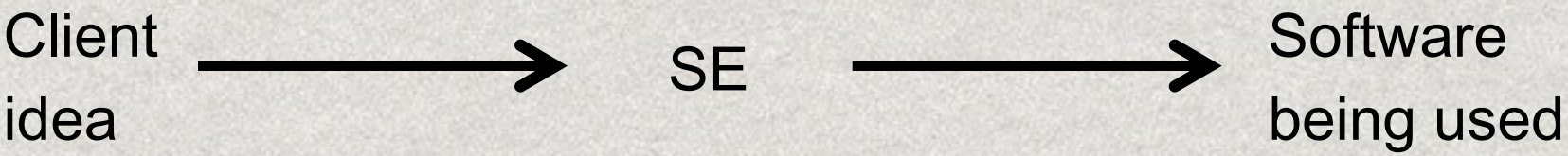
A simple model of SE

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



A simple model of SE

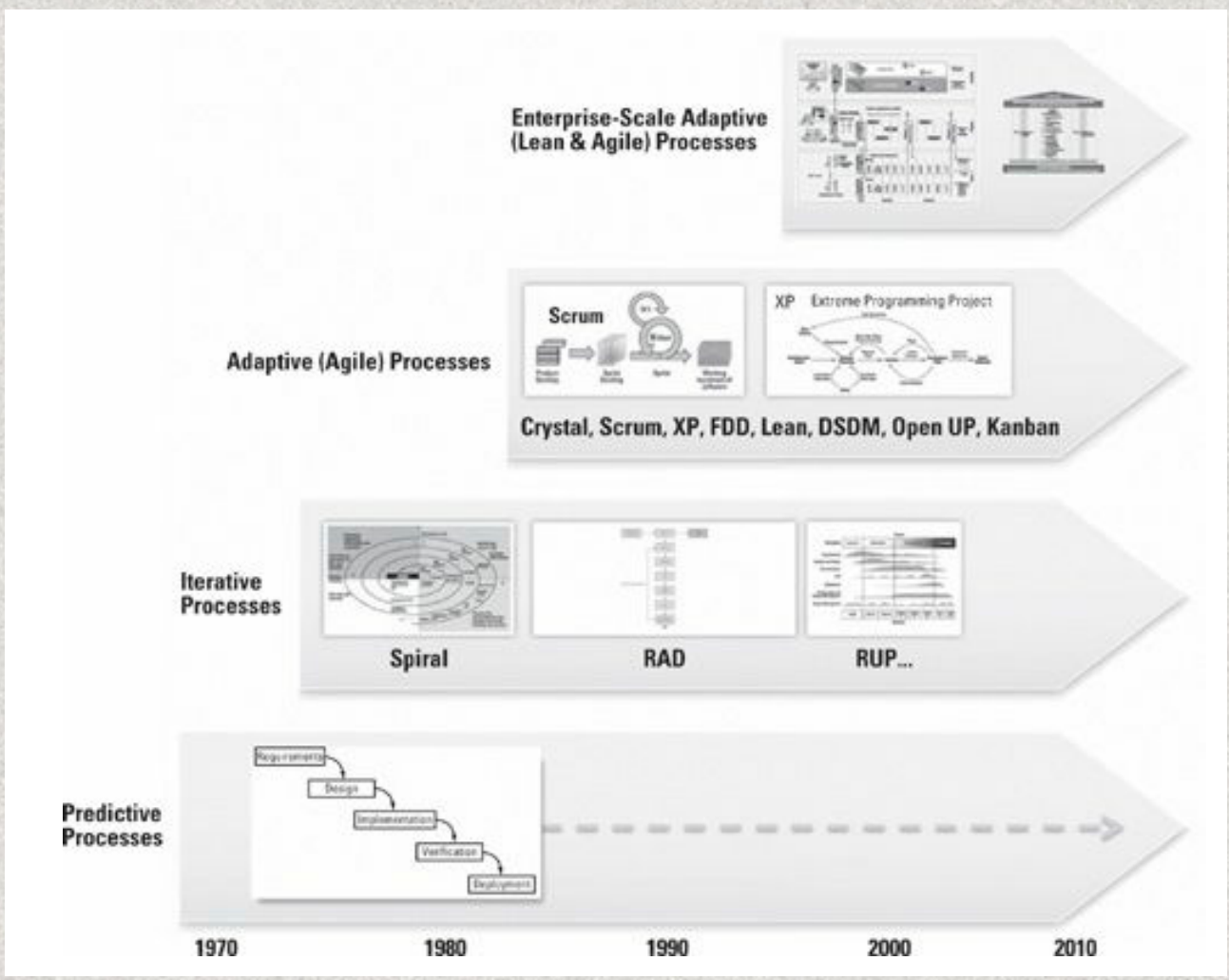
Call out what you think the software development **process activities** should be in order to deliver a large piece of software, starting with an idea from a client.



Write down 1 curiosity Question about software development activities

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.



Plan and Document

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization. —Gerald Weinberg, Weinberg's Second Law

Plan and Document

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization. —Gerald Weinberg, Weinberg's Second Law

The general **unpredictability** of software development in the late 1960s, along with the software disasters, led to the study of **how high-quality software could be developed on a predictable schedule and budget**.

Drawing the analogy to other engineering fields, the term software engineering was coined (Naur and Randell 1969).

The goal was to discover methods to build software that were as predictable in quality, cost, and time as those used to build bridges in civil engineering.

Plan and Document

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization. —Gerald Weinberg, Weinberg's Second Law

The general **unpredictability** of software development in the late 1960s, along with the software disasters, led to the study of **how high-quality software could be developed on a predictable schedule and budget**.

Drawing the analogy to other engineering fields, the term software engineering was coined (Naur and Randell 1969).

The goal was to discover methods to build software that were as predictable in quality, cost, and time as those used to build bridges in civil engineering.

The goal of all these "Plan-and-Document" software development processes is to **improve predictability** via **extensive documentation, which must be changed whenever the goals change.**

Plan and Document

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization. —Gerald Weinberg, Weinberg's Second Law

The general **unpredictability** of software development in the late 1960s, along with the software disasters, led to the study of **how high-quality software could be developed on a predictable schedule and budget**.

Drawing the analogy to other engineering fields, the term software engineering was coined (Naur and Randell 1969).

The goal was to discover methods to build software that were as predictable in quality, cost, and time as those used to build bridges in civil engineering.

The goal of all these "Plan-and-Document" software development processes is to **improve predictability** via **extensive documentation, which must be changed whenever the goals change.**

Documentation should be written at all stages of development, and includes requirements, designs, user manuals, instructions for testers and project plans. —Timothy Lethbridge and Robert Laganiere, 2002

Plan and Document

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization. —Gerald Weinberg, Weinberg's Second Law

The general **unpredictability** of software development in the late 1960s, along with the software disasters, led to the study of **how high-quality software could be developed on a predictable schedule and budget**.

Drawing the analogy to other engineering fields, the term software engineering was coined (Naur and Randell 1969).

The goal was to discover methods to build software that were as predictable in quality, cost, and time as those used to build bridges in civil engineering.

The goal of all these "Plan-and-Document" software development processes is to **improve predictability** via **extensive documentation, which must be changed whenever the goals change.**

Documentation should be written at all stages of development, and includes requirements, designs, user manuals, instructions for testers and project plans. —Timothy Lethbridge and Robert Laganiere, 2002

Documentation is the lifeblood of software engineering. —Eric Braude, 2001

Plan and Document

An early version of this Plan-and-Document software development process was developed in 1970 (Royce 1970). It follows this sequence of phases:

1. Requirements analysis and specification
2. Architectural design
3. Implementation and Integration
4. Verification
5. Operation and Maintenance

Team-based SE Challenges

- discovering requirements and staying on the same page (sharing understanding)
- integrating different developers' code,
- keeping track of progress
- scheduling and planning
- who does what - resource management

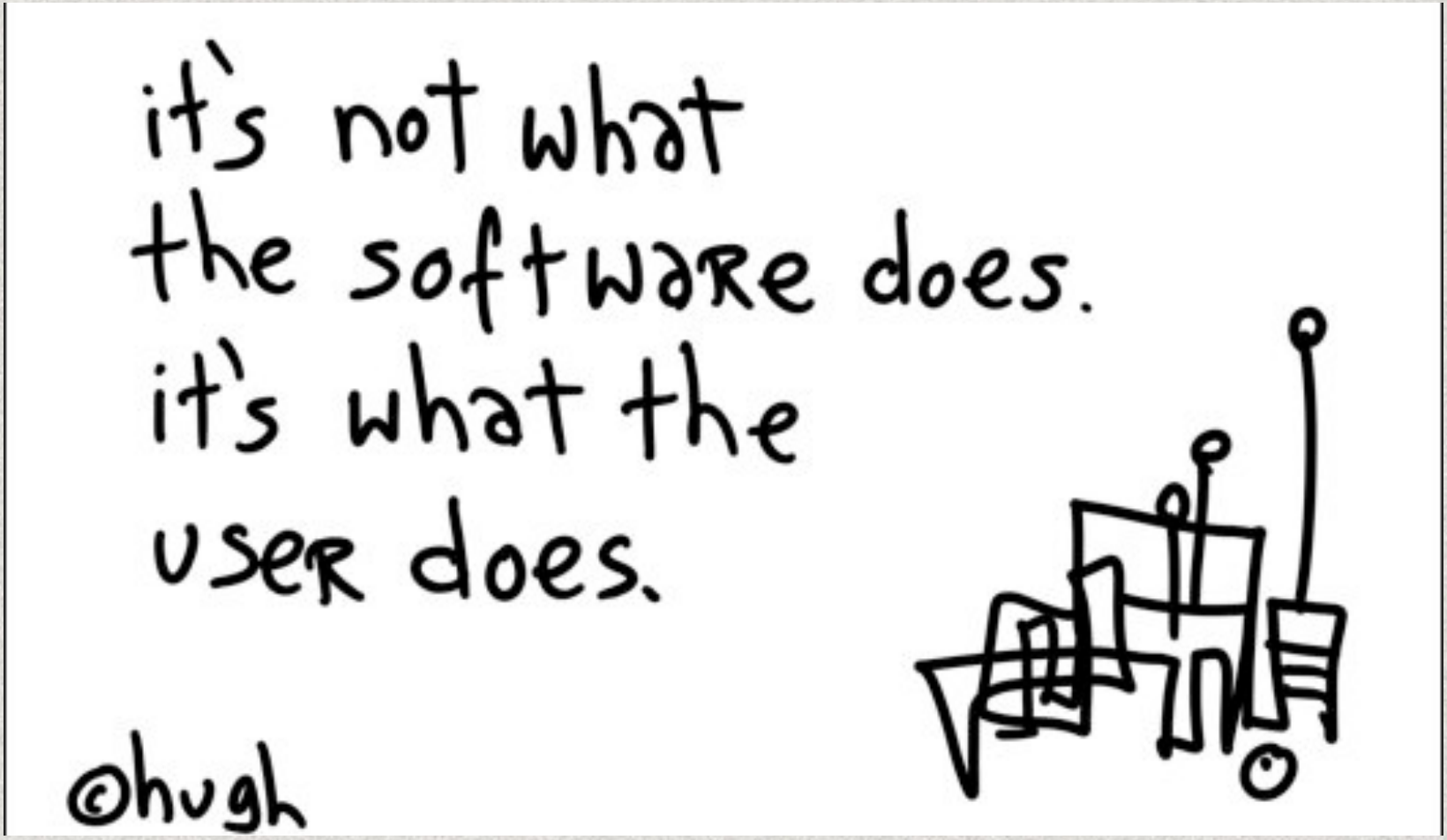
Most Modern SE approaches are:

Iterative

Incremental

Collaborative

Challenges



Is it important?

Is it important?

Expenditure over a trillion $US worlwide

Is it important?

Expenditure over a trillion $US worlwide

It’s pervasive

Is it important?

Expenditure over a trillion $US worlwide

It's pervasive

If it goes wrong…

Is it important?

Expenditure over a trillion $US worlwide

It's pervasive

If it goes wrong…

Who will do it that's smart enough if you don't…?

Is it important?

Expenditure over a trillion $US worlwide

It's pervasive

If it goes wrong…

Who will do it that's smart enough if you don't…?

It has helped with some big problems facing humanity and may continue to help…

Is it important?

Expenditure over a trillion $US worlwide

It's pervasive

If it goes wrong…

Who will do it that's smart enough if you don't…?

It has helped with some big problems facing humanity and may continue to help…

Hey … my mobile phone needs software…

How might the SDM differ for…

Software for my own home inventory and

the software for an inventory system at The Warehouse

Software for my watch and

software for my car

Software for my sales team at the bank

software for making stock trades at the bank

Challenges

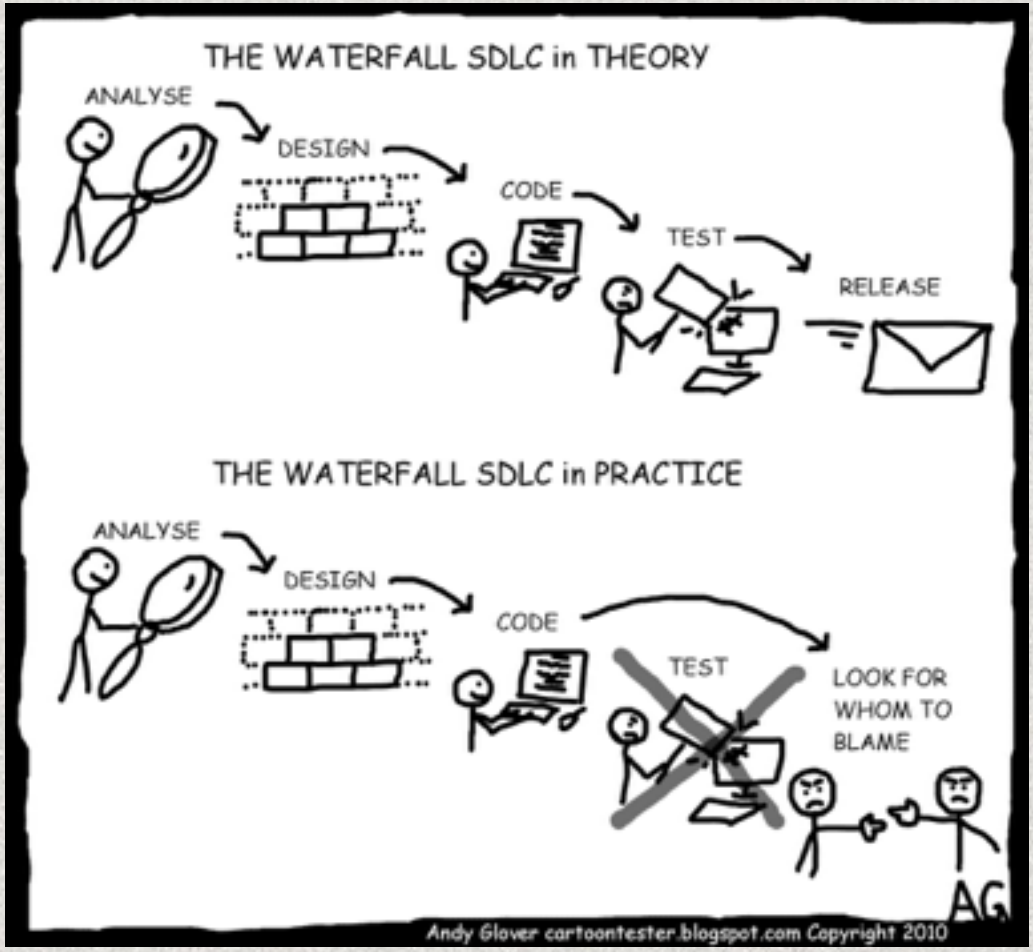

OBSOLETE: Any piece of software or hardware that you bought last week for mega bucks.
COMPUTER DICTIONARY
VERSION 9.8
VERSION 8.7
search ID: tmcm35
© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com

What is the goal, purpose, aim of SE?

What is the goal, purpose, aim of SE?

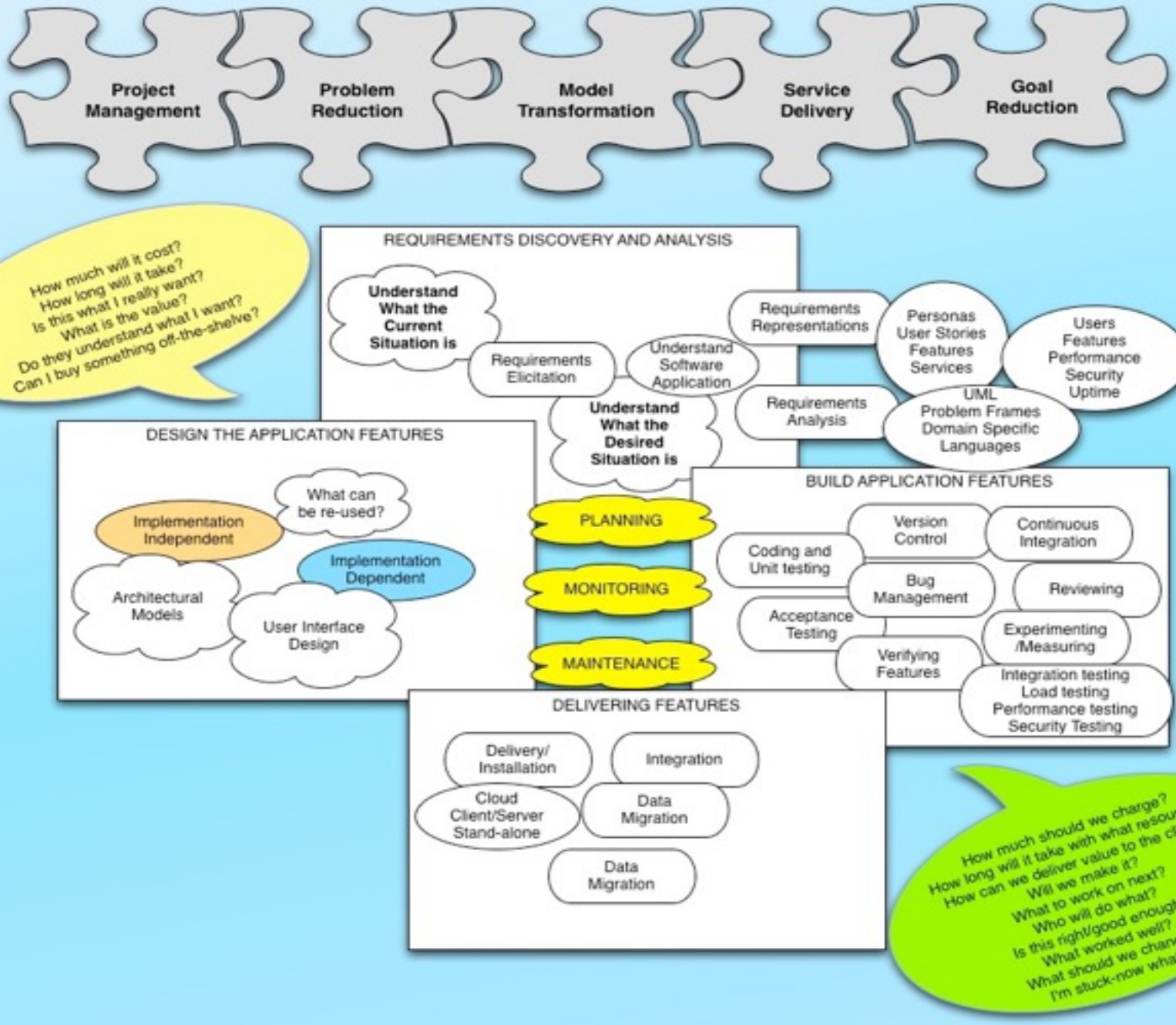
To create value

Challenges



Paper Outline

1. Describe the emerging challenges in current software development contexts
2. Describe emerging trends in software development methods and explain the rationale for some of these
3. Critically assess, compare and contrast the distinguishing features of a variety of software development approaches and methods
4. Recommend and justify the selection of development approaches, methods and practices across the full range of development activities for different development contexts
5. Use a selection of industry standard models, tools and techniques that support development methods
6. Critique research literature in the area of software development methods
7. Evaluate the strength of evidence for the efficacy of significant methods and tools
8. Apply some software development methods and critically reflect on the experience.


Project Management
Problem Reduction
Model Transformation
Service Delivery
Goal Reduction
How much will it cost?
How long will it take?
Is this what I really want?
What is the value?
Do they understand what I want?
Can I buy something off-the-shelve?
REQUIREMENTS DISCOVERY AND ANALYSIS
Understand What the Current Situation is
Requirements Elicitation
Understand Software Application
Requirements Representations
Personas
User Stories
Features
Services
Users
Features
Performance
Security
Uptime
Requirements Analysis
UML
Problem Frames
Domain Specific Languages
Understand What the Desired Situation is
DESIGN THE APPLICATION FEATURES
What can be re-used?
Implementation Independent
Implementation Dependent
Architectural Models
User Interface Design
PLANNING
MONITORING
MAINTENANCE
BUILD APPLICATION FEATURES
Version Control
Continuous Integration
Coding and Unit testing
Bug Management
Reviewing
Acceptance Testing
Experimenting /Measuring
Verifying Features
Integration testing
Load testing
Performance testing
Security Testing
DELIVERING FEATURES
Delivery/ Installation
Integration
Cloud
Client/Server
Stand-alone
Data Migration
Data Migration
How much should we charge?
How long will it take with what resources?
How can we deliver value to the client?
Will we make it?
What to work on next?
Who will do what?
Is this right/good enough?
What worked well?
What should we change?
I'm stuck-now what?

SWEBOK® V3.0

Guide to the Software Engineering Body of Knowledge

Editors

Pierre Bourque
Richard E. (Dick) Fairley

IEEE computer society

SWEBOK

International Conference on Software Engineering

Modern Software Development Methods

Modern Agile ITPM is a mash-up from a selection of a variety of good ideas and practices from a variety of sources.

Modern Software Development Methods

Modern Agile ITPM is a mash-up from a selection of a variety of good ideas and practices from a variety of sources

Scrum

Kanban

PMBoK

Lean

reme Programming

PMI

Are Agile Methods Really Used?

7th ANNUAL STATE of AGILE

VERSIONONE Agile Made Easier

Plan on using Agile Project Management for future projects

2011 - 59%
2012 - **83%**

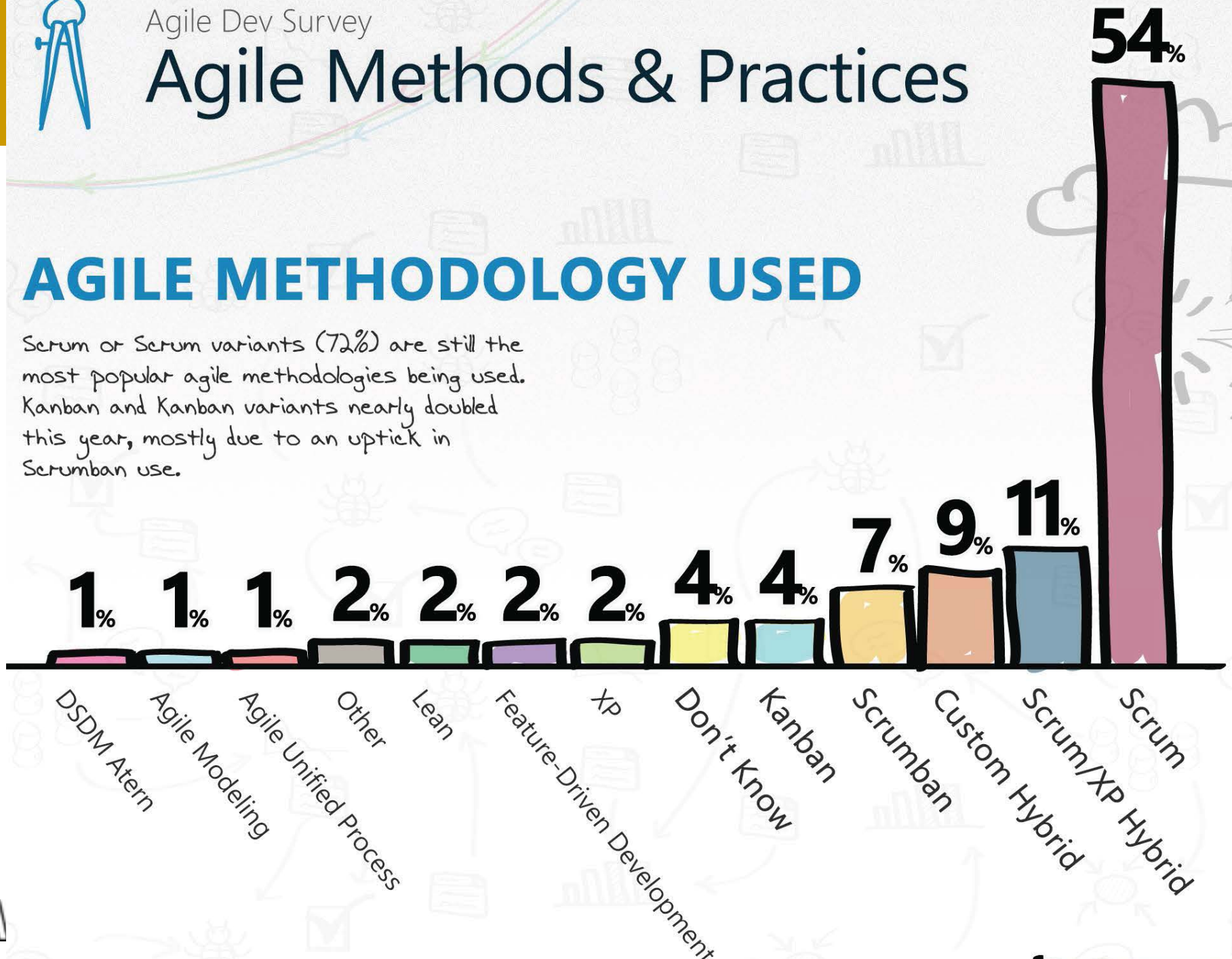
AUT | COMPUTER + MATHEMATICAL SCIENCES

Agile Dev Survey

Agile Methods & Practices

AGILE METHODOLOGY USED

Scrum or Scrum variants (72%) are still the most popular agile methodologies being used. Kanban and Kanban variants nearly doubled this year, mostly due to an uptick in Scrumban use.

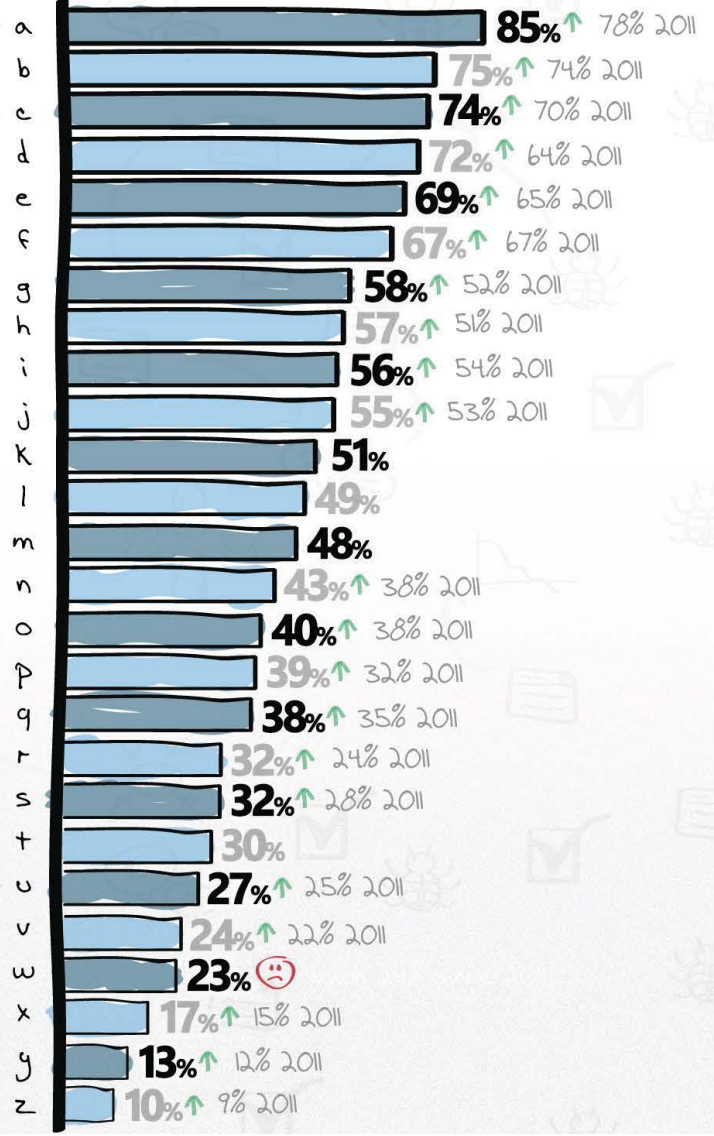


AGILE TECHNIQUES EMPLOYED

Again this year, core agile tenets currently in use are* Daily Standup, Iteration Planning and Unit Testing. The two techniques that grew the most in usage from this year to last year were Kanban and Retrospectives; yet, agile techniques increased in every area but one (Continuous Deployment).

*Respondents were able to select multiple options.

- a Daily Standup
- b Iteration Planning
- c Unit Testing
- d Retrospectives
- e Release Planning
- f Burndown/ Team-Based Estimation
- g Velocity
- h Coding Standards
- i Continuous Integration
- j Automated Builds
- k Dedicated Product Owner
- l Integrated Dev/QA
- m Refactoring
- n Open Workarea
- o TDD
- p Digital Taskboard
- q Story Mapping
- r Kanban
- s Collective Code Ownership
- t Pair Programming
- u Automated Acceptance Testing
- v Analog Taskboard
- w Continuous Deployment
- x Agile Games
- y Cycle Time
- z BDD



AUT | COMPUTER + MATHEMATICAL SCIENCES

SERL
Software Engineering Research Laboratory

Agile Dev Survey

Reasons for Adopting Agile

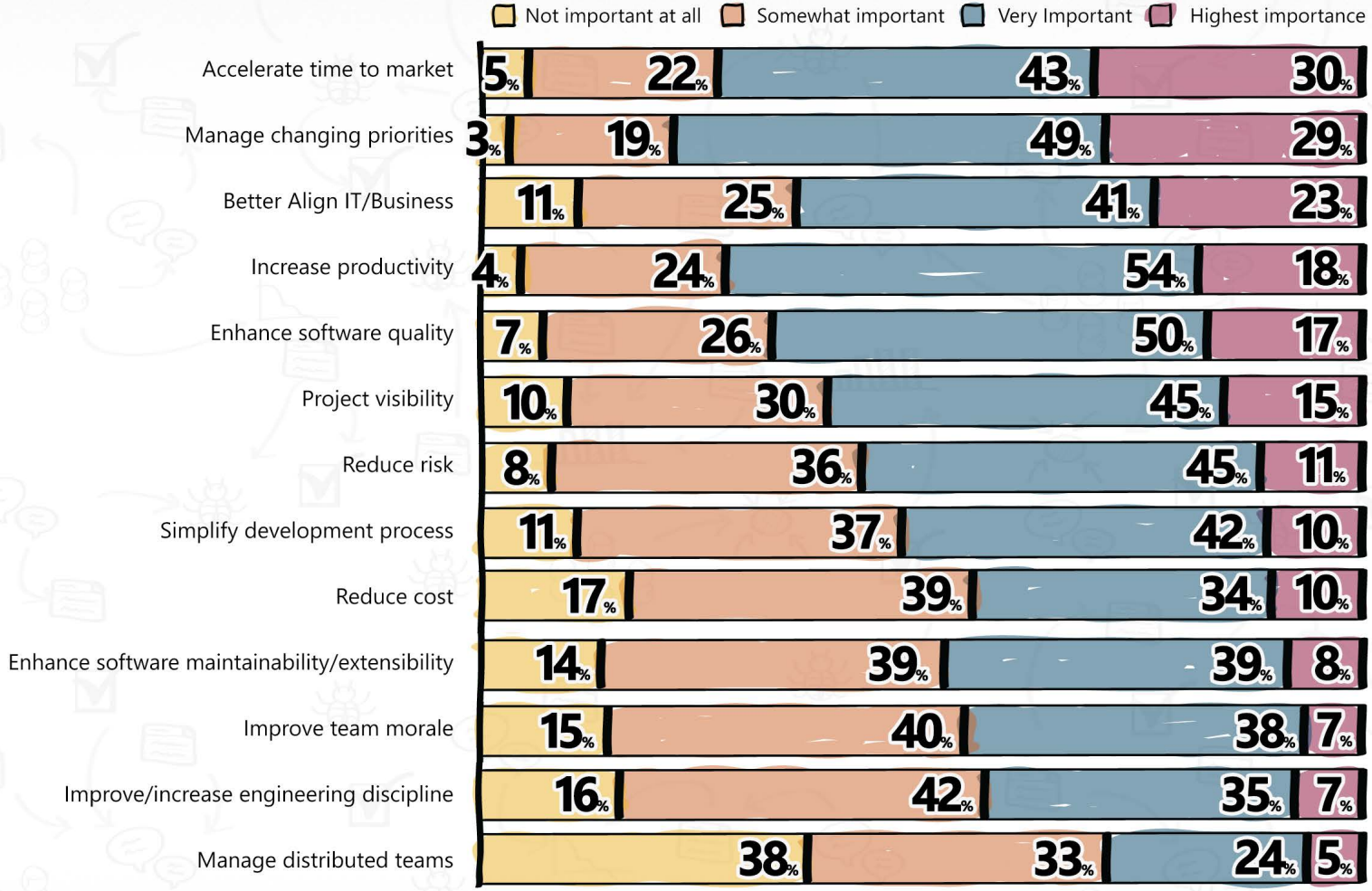
7th ANNUAL STATE of AGILE

WHY AGILE?

Most responses centered on better customer focus and increased predictability.

Once again, top 3 reasons* respondents cited for adopting agile were to accelerate time to market, more easily manage changing priorities, and to better align IT and business objectives:

*Respondents were able to select multiple options.



SERI
Software Engineering Research Laboratory

Agile Dev Survey

Reasons for Adopting Agile

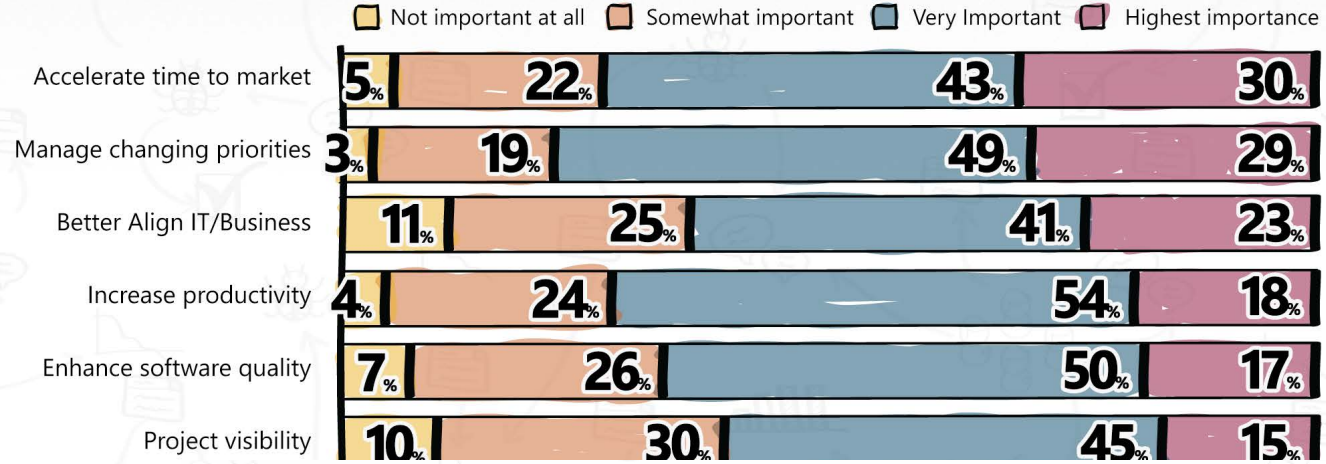
7th ANNUAL STATE of AGILE

WHY AGILE?

Once again, top 3 reasons* respondents cited for adopting agile were to accelerate time to market, more easily manage changing priorities, and to better align IT and business objectives:

*Respondents were able to select multiple options.

Most responses centered on better customer focus and increased predictability.



Simplify developm

Enhance software maintainability/

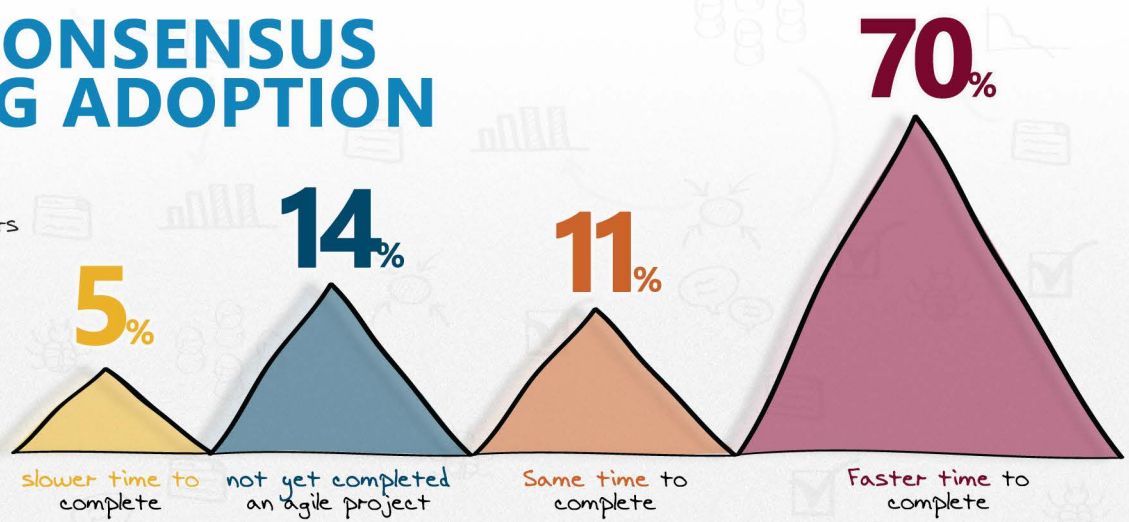
Improve t

Improve/increase engineerir

Manage distril

OVERALL CONSENSUS REGARDING ADOPTION OF AGILE

The vast majority of respondents felt that agile projects have a faster time to completion.



My Empirically Based Educational Philosophy

Constructivist Learning

Collaborative Learning

Student centred learning

Lecturer as facilitator and guide

Learning to Learn

Deep thinking/learning rather than shallow, non-critical

Questioning is a master skill

Communicating ideas is as important as having them