

Course Introduction

Week 1





Mihi

Hāere mai, Haere Mai, Haere Mai.

Tēnā koutou katoa.

Ko Tony Clear taku ingoa.

Nō Pōneke ahau.

Ko Maungakiekie taku maunga.

Ko Waitematā taku moana.

I te taha o taku matua, no Enniscorthy Ireland ahau.

I te taha o aku whaea, no Cork Ireland ahau.

Ko Tainui Raua Ko Ngapuhi nga iwi o nga mokopuna

Tēnā koutou, Tēnā koutou, Tēnā tatou katoa.

A bit about me

First Degrees in Latin and English Language

Stint as a high school teacher

Joined IT industry as a programmer

Worked on large government and corporate systems projects

Managed teams of developers and other functions

Joined then AIT as staff manager, gained a Master's degree in IS and PhD in Computer and Information Sciences

Held roles of discipline group leader, associate head of school, head of school, faculty associate dean research

Co-ordinated R&D project course for many years,

Now associate professor with teaching, research and service roles

Enjoy software and development!

Researching aspects of global SE and computer science education



```

import React from "react";
import { useForm } from "react-hook-form";
import "./styles.css";

export default function App() {
  const { register, handleSubmit, errors } = useForm();

  const onSubmit = (data) => {
    console.log(data);
  };

  return (
    <div className="App">
      <form onSubmit={handleSubmit(onSubmit)}>
        <div className="form-control">
          <label>Email</label>
          <input
            type="text"
            name="email"
            ref={register({
              required: true,
              pattern: /^[^@]+@[^@]+\.[^@]{2,}$/
            })}
          />
          {errors.email && errors.email.type === "required" && (
            <p className="errorMsg">Email is required.</p>
          )}
          {errors.email && errors.email.type === "pattern" && (
            <p className="errorMsg">Email is not valid.</p>
          )}
        </div>
        <div className="form-control">
          <label>Password</label>
          <input
            type="password"
            name="password"
            ref={register({ required: true, minLength: 6 })}
          />
          {errors.password && errors.password.type === "required" && (
            <p className="errorMsg">Password is required.</p>
          )}
          {errors.password && errors.password.type === "minLength" && (
            <p className="errorMsg">
              Password should be at-least 6 characters.
            </p>
          )}
        </div>
        <div className="form-control">
          <label></label>
          <button type="submit">Login</button>
        </div>
      </form>
    </div>
  );
}

```

// validation function

```

const validatePassword = (value) => {
  if (value.length < 6) {
    return 'Password should be at-least 6 characters.';
  } else if (
    !/(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?!.*\s)(?=.*[@#$%])/ .test(value)
  ) {
    return 'Password should contain at least one uppercase letter, lowercase letter, a digit and a special character.';
  }
  return true;
};

// JSX
<input
  type="password"
  name="password"
  ref={register({
    required: 'Password is required.',
    validate: validatePassword
  })}
/>

```

What does this code do and
tell me where the mistake is.....

Taking Stock

The schedule for the course

Where are we now?

Class Representative

Perspectives Quiz

The Assessment Schedule

Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

Taking Stock

Week

No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Review
Questionnaire

Assgt 1A -
Techstack

Worksheets

Assgt 1B - Team
Project

Iteratio
ns



Why have this course – (Jim Buchan's) vision

The aim of the Software Development/Engineering majors is to give you opportunities to build the **knowledge, capabilities** and **attitudes** to become a **good software engineer**

So, what capabilities and attitudes does a good software engineer have?

Many! That's why good ones are in such demand..

The important thing is that you can apply these capabilities and attitudes to collaborate with a team of others to

**...create, deploy and maintain
high quality software that is of
value to some users.**



(Tony's) views - Reflecting Over Time



<https://inroads.acm.org/about.cfm>

About *ACM Inroads*

The *ACM Inroads* magazine serves professionals interested in advancing computing education on a global scale.

Each issue of *ACM Inroads* presents the latest work, insights, and research in computing education as written by educators and professionals *for* educators. Authors represent an international community of scholars and professionals who reflect on and contribute to the computing profession.

The Association for Computing Machinery (ACM), the largest educational and scientific computing society in the world, publishes *ACM Inroads* quarterly.

My reflections here come from experiences expressed in roles as a regular Columnist and Associate Editor

Clear, T. (2003, Dec). **The Waterfall is Dead - Long Live the Waterfall!** *SIGCSE Bulletin*, 35(4), 13-14.

(Tony's) views – SE & Tensions - 1

Reflections on Software Development/Engineering Over time

- Requirements analysis
- Feasibility/Design
- Construction
- Implementation and testing

...core distinctions between **programming-in-the small** and **programming-in-the-large**. Key questions such as “what is programming?” come to mind. Is programming “the implementation of a design”.

(Tony's) views – SE & Tensions - 2

Reflections on Software Development/Engineering Over time

So it seems to me that we have **a tension between four opposing forces**:

- a **force for change** built upon an **initial and evolving vision**, which drives the software process
- a **commercial force for certainty** of cost and outcomes
- a **project management force for certainty of delivery** against targets
- a **professional force for delivering quality software**

Highsmith's [1] **interaction, cooperation and collaboration** within the software process.

Clear, T. (2003, Dec). **The Waterfall is Dead - Long Live the Waterfall!!!** SIGCSE Bulletin, 35(4), 13-14.

(Tony's) views – *Feature Driven Dev't - 1*

“Information systems **success is achieved** when an information system is **perceived to be successful** by the stakeholders and other observers”.

...Feature Driven Development [7] ...delivering **scheduled releases** on a **regular set of cycles** for a demanding client operating in an entrepreneurial climate.

“...improved the relationship between our client and us by allowing the client to consider priorities and goals when planning each release, and accordingly **change their requirements specifications as the business evolves**” [2].

(Tony's) views – Feature Driven Dev’t - 2

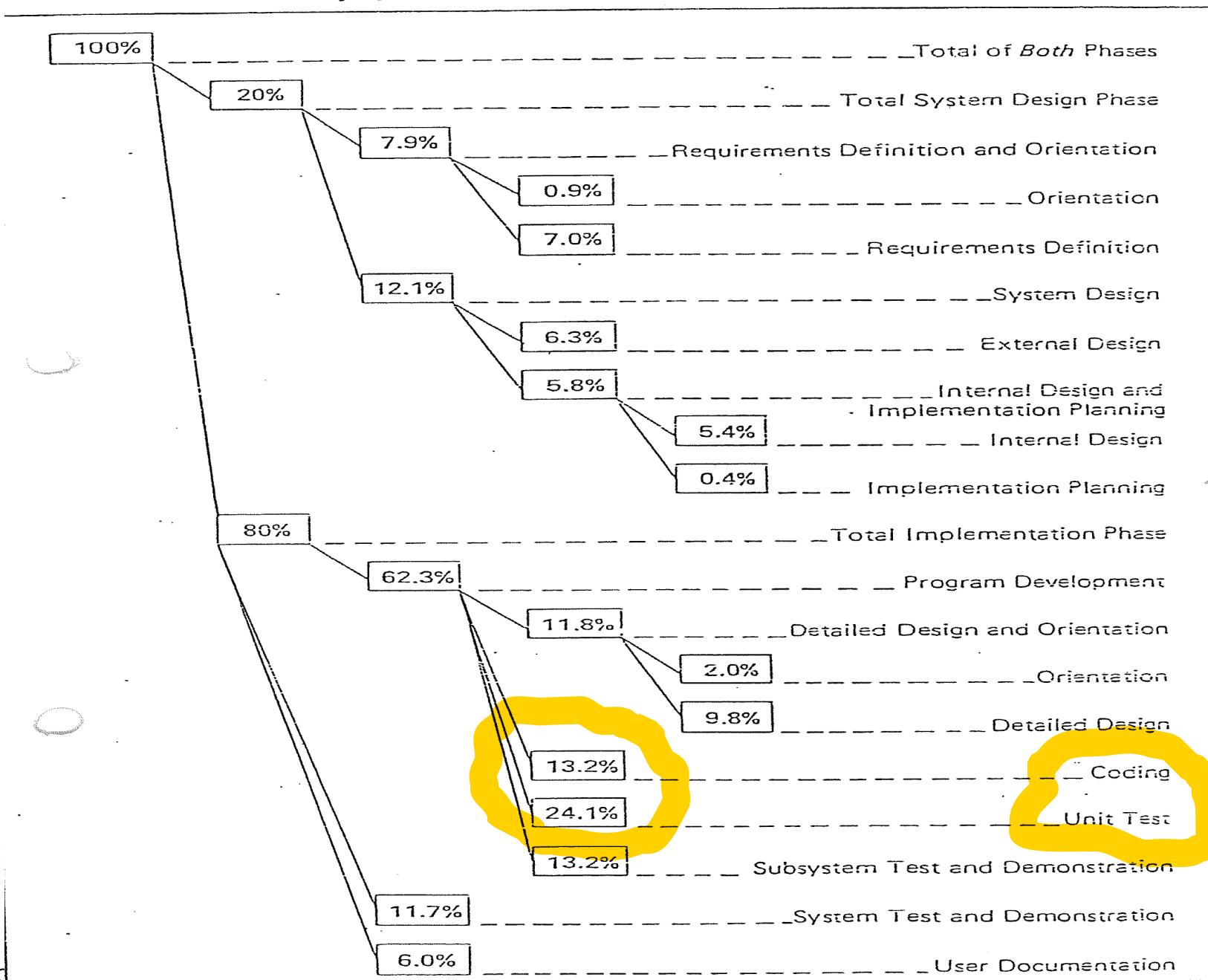
...**delivering functionality constantly** helped them “**better estimate** the cost of each feature, allow for project creep and to therefore have better control over the project as a whole”.

... ‘**time-boxing**’,(namely setting short time delimited windows within which results must be produced). ..“**came to realise that time boxes were about forcing tough decisions**”.

...customers and developers have **features as a common ground** in which to “discuss, debate and decide on critical product features” [4]. Certainly in the online talent agency project [2], **packaging four features into each release** enabled the client to **adjust priorities** within defined delivery windows.

Clear, T. (2004). **Students Becoming Political and "Incorrect" Through Agile Methods**. SIGCSE Bulletin, 36(4), 13 - 15.

Figure 3.4 shows the distribution of work-effort among the standard technical tasks experienced by DP Services on previously completed total projects* where the same function was implemented as was designed.



(Tony's) views - Documentation- 1

the views of Naur ... the notion of **programming as “theory building”**, during which the programming team develops a jointly owned “theory of the world” to become frozen into software.

...**documentation as a secondary construct** to the programmers' internalised theory of the program or system, and based upon this “Theory Building View, **for the primary activity of the programming there can be no right method**”[2], since the creative process of theory building is inherently not a method or rule driven activity.

Clear, T. (2003). **Documentation and Agile Methods: Striking a Balance**. SIGCSE Bulletin, 35(2), 12 - 13.

(Tony's) views - Documentation- 2

...documentation is not necessary **if the programming team can jointly own and hold the theory of the world in their head.**

...“**the code, the code and nothing but the code**” as the key artifact from a software project.

documentation as something external, something "other" than the primary work of coding, since it is only through the coding that the theory becomes encapsulated.

Ambler [5] suggests **two primary reasons for documentation**, namely that we should model (or document) **to communicate**, or model (or document) **to understand**.

Clear, T. (2003). **Documentation and Agile Methods: Striking a Balance**. SIGCSE Bulletin, 35(2), 12 - 13.

Working Globally in Software Engineering

Student Conceptions of the Discipline?

What does it mean to develop software as a professional software engineer?

Peters, A., Hussain, W., Cajander, A., Clear, T., & Daniels, M. (2015). **Preparing the Global Software Engineer**. In M. Nordio & B. Al-Ani (Eds.), *Proceedings 2015 IEEE 10th International Conference on Global Software Engineering* (pp. 61-70). IEEE. <https://doi.org/10.1109/ICGSE.2015.20>

Clear, T. (2016). THINKING ISSUES: **Computer science education---: challenging thinking in the small?** ACM Inroads, 7(2), 31-33. <https://doi.org/10.1145/2927016>

Participation in Computer Science is experienced ...

- ...A. as *using*, i.e. to make use of what exists for various purposes.
- ...B. as *inquiry*, i.e. activities that aim at understanding, learning, informing.
- ...C. as *creating* things, i.e. to produce things that were not there before.
- ...D. as (*systematic*) *problem solving*. This includes using methods, ways of thinking and (systematically) working with others to create things.
- ...E. as *creating for others*. This includes taking into account the user's perspective in the process of creating and problem solving.
- ...F. as *continuous development*, i.e. as a continuous process of improvement.
- ...G. as *creating knowledge* to develop new solutions, i.e. to do research.

Figure 1: Categories Describing Qualitatively Different Ways of Experiencing The Phenomenon Participation in CS/IT [10]

Working Globally in Software Engineering – Maturing?

Student Conceptions of the Discipline?

What does it mean to develop software?
as a professional software engineer?

Table 1: Student Progression across Categories of Participation in CS/IT [11]

Student Progression Across Categories of Participation In CS/IT	Categories of Experiencing Participation In CS/IT						
	A	B	C	D	E	F	G
Reflection	Using	Inquiry	Creating things	Systematic Problem Solving	Creating for others	Continuous Development	Creating Knowledge/ Research
Pre-course	0%	8%	13%	52%	23%	4%	0%
Post-course	0%	3%	10%	24%	39%	9%	15%

Clear, T. (2016). THINKING ISSUES: Computer science education---: challenging thinking in the small? ACM Inroads, 7(2), 31-33.
<https://doi.org/10.1145/2927016>

(Tony's) views – *Dispositions and Agility* - 1

“..agilisme”... core principle of agility from the agile manifesto [9] states that “**Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**”

Implication by a high performing team

Unpacking the **three key aspects** then of agile development they can be said to comprise: 1) **customer satisfaction**, 2) **delivery of working software**, and 3) **provision of value..**

AUT Students and Healthy Together Technology

This year, the Healthy Together Technology team has been supporting a group of AUT Bachelor of Computer and Information Management students.

The students all majored in either Software Development or Service Science and have been completing their final year projects. They chose to work with CM Health on two projects – a chat-bot for the main CM Health website to help answer FAQs for the public, and a Wayfinding App to assist patients and whaanau to navigate around the Middlemore Hospital site, particularly to the Galbraith Infusion Centre and the Cardiac investigation unit.

They have been mentored by Megan Milmine, Deputy CIO, and Sally Dennis, IS Clinical Change Manager. Read more [here](#).



(Tony's) views – *Dispositions and Agility* - 2

...disposition “**concerns not what abilities people have, but how people are disposed to use those abilities.**” [10] “So here we are talking about a mindset and attitudinal dimensions, which raises the question can a disposition be taught or is it some innate part of a person’s character?”

AUT Students and Healthy Together Technology

This year, the Healthy Together Technology team has been supporting a group of AUT Bachelor of Computer and Information Management students.

The students all majored in either Software Development or Service Science and have been completing their final year projects. They chose to work with CM Health on two projects – a chat-bot for the main CM Health website to help answer FAQs for the public, and a Wayfinding App to assist patients and whaanau to navigate around the Middlemore Hospital site, particularly to the Galbraith Infusion Centre and the Cardiac investigation unit.

They have been mentored by Megan Milmine, Deputy CIO, and Sally Dennis, IS Clinical Change Manager. Read more [here](#).



(Tony's) views – *Dispositions and Agility* - 3

...producing a full quality assurance plan or change management plan when the project approach had not yet been determined, **did not make sense**, or as we discussed at the mid-project review

'did not produce value for the client,'

team who **internalized the need for discretion and judgement** when deciding **which tasks and deliverables** contributed to providing value, **so made sense to produce, and at what time.**

AUT Students and Healthy Together Technology

This year, the Healthy Together Technology team has been supporting a group of AUT Bachelor of Computer and Information Management students.

The students all majored in either Software Development or Service Science and have been completing their final year projects. They chose to work with CM Health on two projects – a chat-bot for the main CM Health website to help answer FAQs for the public, and a Wayfinding App to assist patients and whaanau to navigate around the Middlemore Hospital site, particularly to the Galbraith Infusion Centre and the Cardiac investigation unit.

They have been mentored by Megan Milmine, Deputy CIO, and Sally Dennis, IS Clinical Change Manager. Read more [here](#).



DevOps, Agility and Dispositions...

Complementing the required knowledge and skills, **job ads also showed a desire for recruits who had specific** (Attributes, **Dispositions**, Attitude & Philosophy) detailed in Appendix 1.

Schussler observes that “**dispositions are different from knowledge and skills**” and “**concerns not what abilities people have, but how people are disposed to use those abilities**” [41].

The dispositions and attributes sought were **typically human and team centric**, demanding flexibility and adaptability

a **wider mindset** including customer/business awareness, relationship management, communication, general business acumen and respect in collaborative relationships

- Clear, T. (2021). THINKING ISSUES: Is Agility a Disposition and Can it be Taught? . ACM Inroads, 12(1), 13-14.
<https://doi.org/10.1145/3447870>
- Clear, T. (2017). Meeting Employers Expectations of DevOps Roles: Can Dispositions Be Taught? . ACM Inroads, 8(2), 19-21.
<https://doi.org/10.1145/3078298>
- Hussain, W., Clear, T., & MacDonell, S. (2017). Emerging Trends for Global DevOps: A New Zealand Perspective. In D. Cruzes & A. Sharma (Eds.), Proceedings 2017 IEEE 12th International Conference on Global Software Engineering (pp. 21-30). IEEE.
<https://doi.org/10.1109/ICGSE.2017.16>

Leadership Atributes...

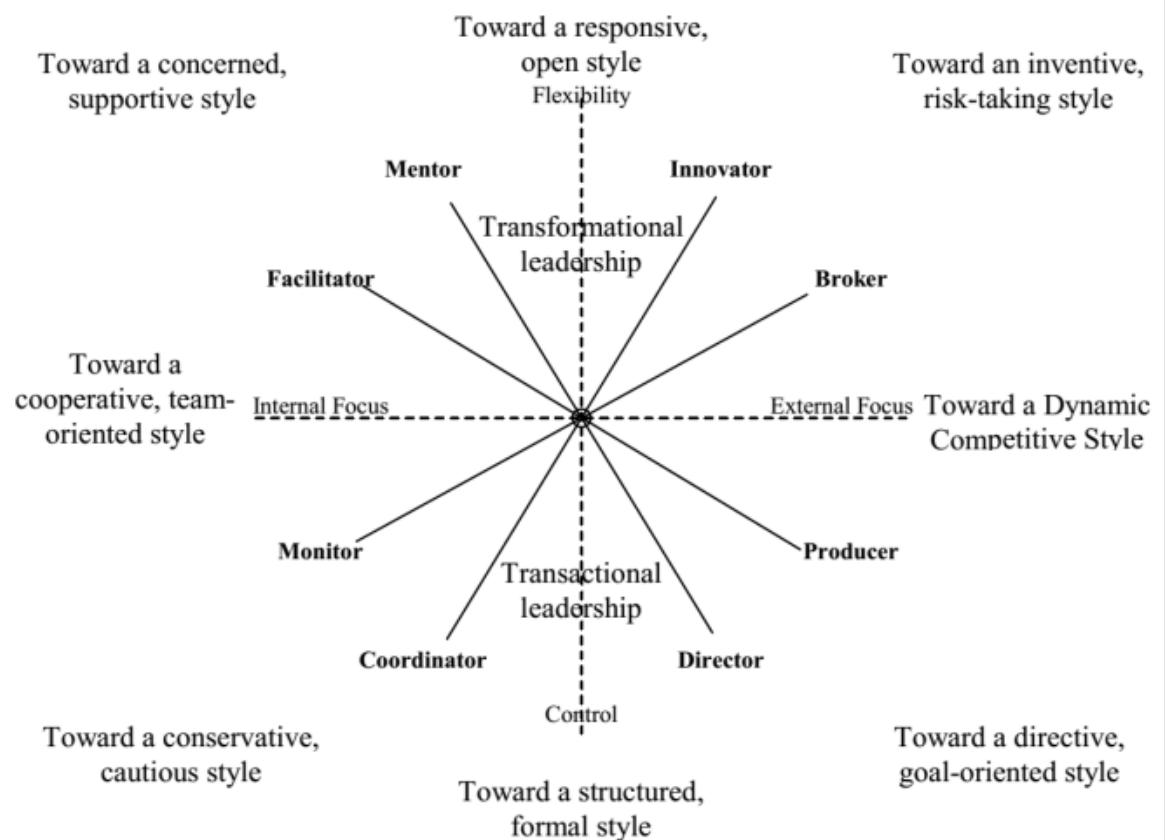


Figure 5: Competing Values Framework of Leadership Roles [43]

...leadership attributes were actively sought. From our data we saw the strong emphasis on '**Transformational Leadership'** roles such as **Mentor, Facilitator, Innovator and Broker**. These stood in contrast to the more traditionally viewed expectations of technical employees engaged in projects and task related activities, better fitting a '**Transactional Leadership**' style.

Expectations of **taking responsibility for other team members** through **training and mentoring them** to develop new Knowledge Skills and Capabilities were very evident in the job ads.

- Hussain, W., Clear, T., & MacDonell, S. (2017). Emerging Trends for Global DevOps: A New Zealand Perspective. In D. Cruzes & A. Sharma (Eds.), *Proceedings 2017 IEEE 12th International Conference on Global Software Engineering* (pp. 21-30). IEEE. <https://doi.org/10.1109/ICGSE.2017.16>

Your SE journey in the BCIS

Programming 1	Coding basics. Good coding practice, code design. A few tools.
Programming 2	Coding in Object Oriented view. Good OO design and coding. IDE. Unit Tests
Program Design and Construction	Extending good coding practice – Design patterns, version control, architecture, ...
Software Development Practice	Practice working in teams. Scrum -What and How. Own PO. Own Tech stack selection. Some tools
Contemporary Issues in Software Engineering	How to work in teams. Collaborate with External PO. The entire SDLC – from vision through evolving requirements to release. Specified tech stack. More tools. QA. More WoW. Hands on experience of how practices integrate with technologies and values to produce valuable high-quality results!
R&D Project	Working in teams for an EXTERNAL client. The entire SDLC. More tools, tech stack.

Problem-based learning

A real-world problem/opportunity for a client which can be addressed with a software product. You have to build the product as a team using good practice and tools. Planning-doing and reflecting on this drives the learning.

You understand theory and evidence - WHY you do something (WHAT) a particular way (HOW) and alternative options.

You practice to build capability and continuous learning from mistakes and reflection and team collaboration and technical skills and use of tools

You reflect on experience to compare theory to practice and evidence

You interact with a client/Product Owner

Developed and delivered in teams incrementally and iteratively using Agile informed ways of working and thinking and behaving

What work do we need to do and how and why?

Understand what to build – collaborating with a client - ongoing

Build it

Plan?

Deploy it

Maintain it

What work system, process, flow?

What are the values and principles that will guide how we decide to work, interact and behave?

What documentation is needed

How will this course help... a view of learning

- SE language used by other developers
- SE concepts used by other developers
- SE Values and principles that apply to many situations (patterns) and guide behaviour, interactions and work patterns
- SE knowledge – understand how practices and tools address SE problems, what is available and when to use them
- SE skills to use the tools, practices effectively

You will learn best when you are **motivated**...

- You understand the purpose of the learning
- You have some choice in learning (interested)
- It is safe to ask questions and give opinions
- It is safe to make mistakes and learn from them
- You develop mental models – Experiences - Interrupt or reinforce your models
- You are exposed to a diversity of options, experiences and people – no one way is right
- You get the chance to teach others
- You APPLY theory and knowledge to make software and interact as a team
- The work is at a difficulty level where it is a struggle but you get the buzz of succeeding

My expectations of you

- Hard working – minimum 8 hours per week outside of class
- Curious – willing (enjoy) to ask questions
- Committed to understanding my expectations
- Honest with me and each other
- Committed to learning (not just a grade?)
- Respectful of me and each other
- Open to new ideas and ways of working
- Support each other – have empathy
- Willing to think and probe and critique and disagree
- Willing to compromise in teams



Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)

(Individual)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,

Setup the tools needed to work with the selected

Tech Stack (front-end/backend)

Set up tools to assure quality of product

Set up tools to deploy the product to the cloud

Set up tools to monitor and alert issues post deploy

Learn how to use the tools

Learn how to use the Tech Stack

Understand the product goals -> Product Backlog

Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)

Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)

(small team 4)

Capability building by Developing a Product in a small team

Apply a new tech stack and tool set

Practice DevOps and Scrum WoW

Collaborate with a Product Owner and team

Three sprints to learn fast – fast feedback

Submit – reviews weeks 8,10,12 (tutorials)

Capability and learning Portfolio with Evidence

Product increments

Sprint 1 weeks 6 and 7

Sprint 2 weeks 8 and 9

Sprint 3 weeks 10 and 11

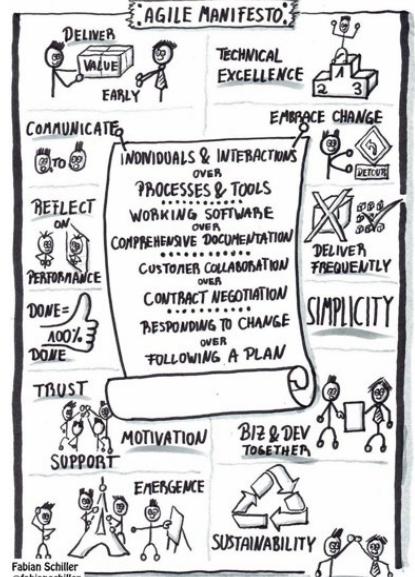
Ass 2 Knowledge Check (30%)

(Individual, online questions)

A set of questions about scenarios to confirm you have understood main language and principles

Sometime in Revision weeks (Faculty schedules)

Choose your team's WoW!



SEMAT (Software Engineering Methods and Theory)

Ways of Working

XP (eXtreme Programming)

Agile ways of working

DevOps
DevSecOps

Scrum

Kanban

Lean

Waterfall (plan-driven)

User stories

Test driven Development

Continuous deployment

Code craft

Mob programming

Continuous Integration

Pair programming

Code Review

Our findings indicate that few participants run their projects in a purely agile or a purely traditional manner, and **most of them use home-grown hybrid development methods.**

Kuhrmann et al. (2022)

Tools to support WoW

Visual Studio. Git. GitHub.

JUnit. Cucumber. Selenium.

TravisCI. Ansana. Jira. AWS.

Heroku. Puppet. Ansible. Maven. Docker. Lint. SonarQube. etc

Learn how to learn a new Technology Stack... and supporting tools

Javascript/Typescript based applications deployed in the cloud

Backend – Nest

Front end – NEXT.js

Database – MongoDB?

Other tools to optimize coding and quality

Visual Studio code, Eslinter, Prettier etc

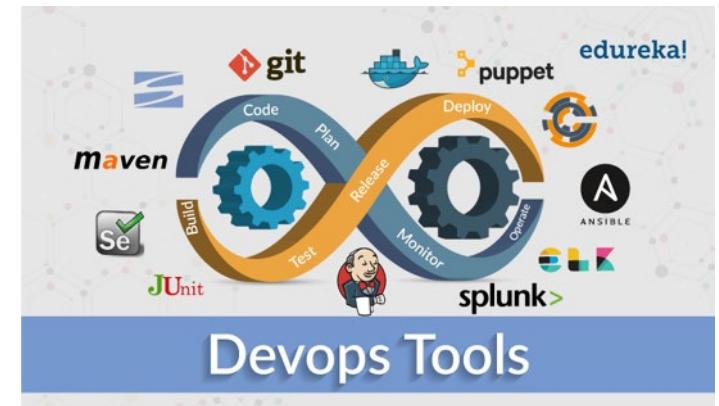
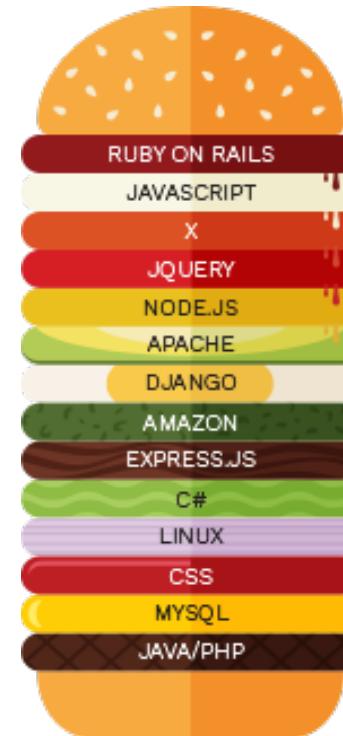
Automated testing/checking from the start – tool pipeline

GitHub for **version control, code sharing, code integration**

GitHub Actions for **(CI/CD)** auto build and test, and deploy

Jest for **end-to-end testing**

Vercel for **cloud deployment**



Diverse sources of knowledge/learning....

- **Online Tutorials** - Freecode.com, YouTube
- **Online documentation** - tools, libraries eg Docker, GitHub
- **Podcasts** Engineering Culture by InfoQ
- **Newsletters** - InfoQ
- **Meetup groups** Agile Auckland, DevOps, Ministry of testing,
- **Blogs** - Medium, Freecode.com,
- **GitHub Repositories** - Open source projects,
- **Company websites** -Google, Xero, Microsoft, Facebook, Netflix, Basecamp
- **Published research**- ACM, IEEEExplore, SpringerLink, ScienceDirect
- **Online Q&A** -Stackoverflow
- **Guest speakers from industry**
- **Work in Industry or Open source projects**
- **Each other** – teaching and listening
- **Your lecturer**

Use of CANVAS and MS teams

- This semester ENSE701 will run as an **on-campus** course for **lectures and labs**
- Canvas will be used as a Repository of course content
 - For communicating announcements from time to time
 - And storing lecture recordings
- Teams channels will also be used for communicating with the product owner and the teaching team, so that they are all in the one place rather than lost in a deluge of emails
- As illnesses are still disrupting our lives... ☹
 - Teams may be used as a backup for meetings with students who have to isolate

Takeaways from today...a mental model of SE and this course



Engineering large software products is complex, can be seen from diverse perspectives and brings many challenges
(but it is great fun and rewarding and needed!)



There is no recipe – you need a box of tools and techniques to pull from and experiment with and adapt



This paper gives you the chance to extend your capability in ways of working, techniques and tools that focus on collaborative software development



Focus on building capability and knowledge as much as you can through doing and experimenting



You will need to keep on learning new techniques, ways of working together and tools from a variety of sources and sometimes just-in-time



Let's build a safe, collaborative, creative and curious learning environment



#171678965



Questions and Comments....



Tony Clear S2 2024

CISE ENSE701

I has a question...



34

Ways of Working in SE

Week 2



Taking Stock

Class Representative, Communication Protocol and Concerns – Announcement Review

Perspectives Quiz – Implications

The schedule for the course

Where are we now?

The Assessment Schedule

Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

Taking Stock

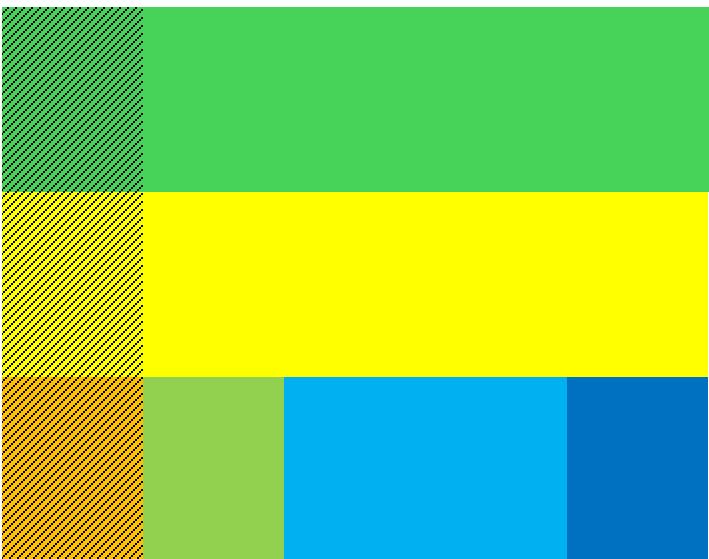
Week
No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Review
Questionnaire

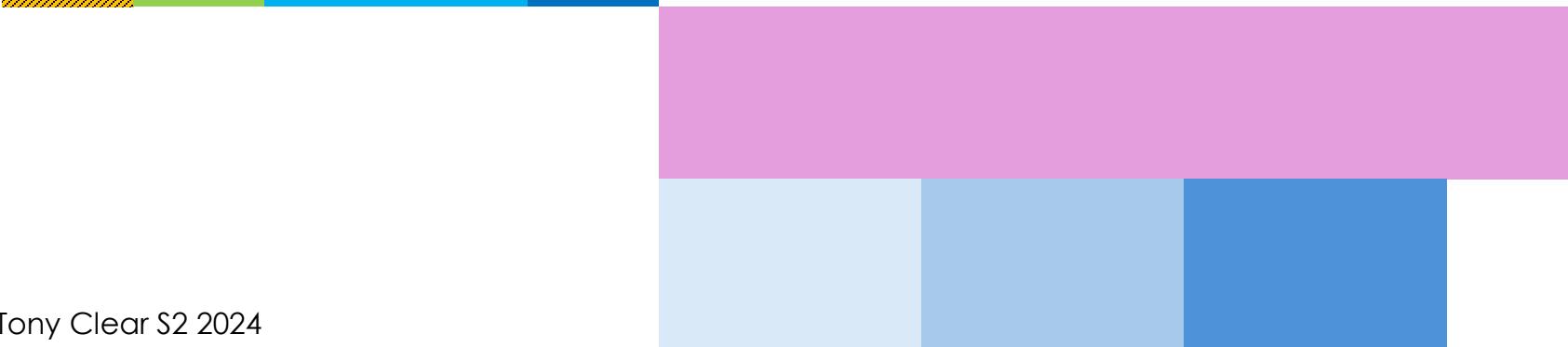


Assgt 1A -
Techstack



Worksheets

Assgt 1B - Team
Project



Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)

(Individual)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,

Setup the tools needed to work with the selected

Tech Stack (front-end/backend)

Set up tools to assure quality of product

Set up tools to deploy the product to the cloud

Set up tools to monitor and alert issues post deploy

Learn how to use the tools

Learn how to use the Tech Stack

Understand the product goals -> Product Backlog

Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)

Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)

(small team - 4 Including QA)

Capability building by Developing a Product in a small team

Apply a new tech stack and tool set

Practice DevOps and Scrum WoW

Collaborate with a Product Owner and team

Three sprints to learn fast – fast feedback

Submit – reviews weeks 7,9,11 (tutorials)

Capability and learning Portfolio with Evidence

Product increments

Sprint 1 weeks 5 and 6

Sprint 2 weeks 7 and 8

Sprint 3 weeks 9 and 10

Ass 2 Knowledge Check (30%)

(Individual, online questions)

A set of questions about scenarios to confirm you have understood main language and principles

Sometime in Revision weeks (Faculty schedules)

Ass1B Team Development

What do we have to **decide**, **do** and **plan** to get ready to start developing and deploying features as a team?

What types of work do we have to do and get prepared for?

Ideas?

Traditional Waterfall approach

Requirements

Design

Build

Test

Deploy

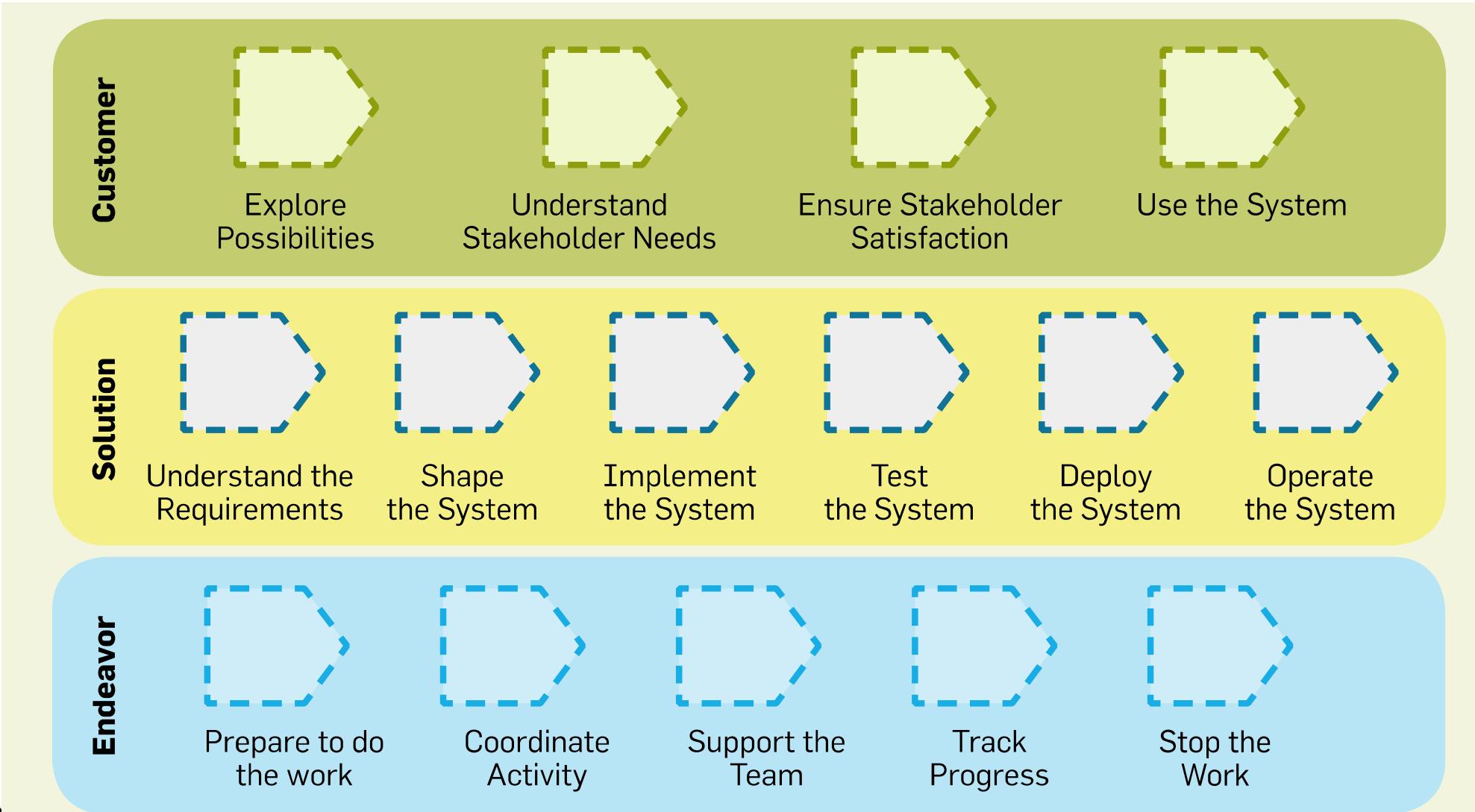
Maintain

Things to do – types of work

There are certain types of work that all software development includes...

SEMAT (Software Engineering Methods and Theory)

(Ivor Jacobsen)



What work do software development teams have to do?

Understand the problem to be solved

Plan the work

Design the solution

Do the work

Deliver/Deploy the solution

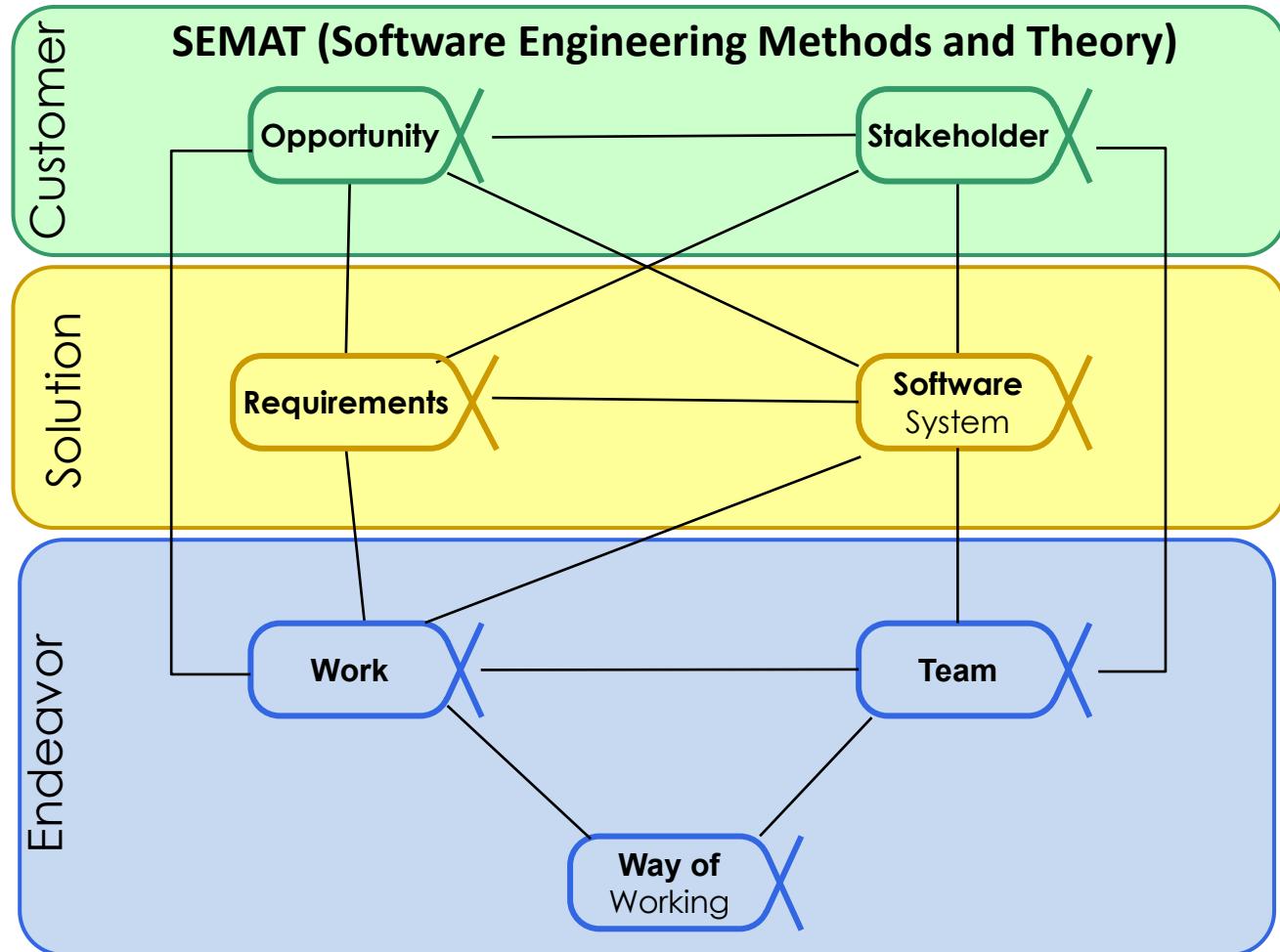
Maintain/update the work

Work = ?????

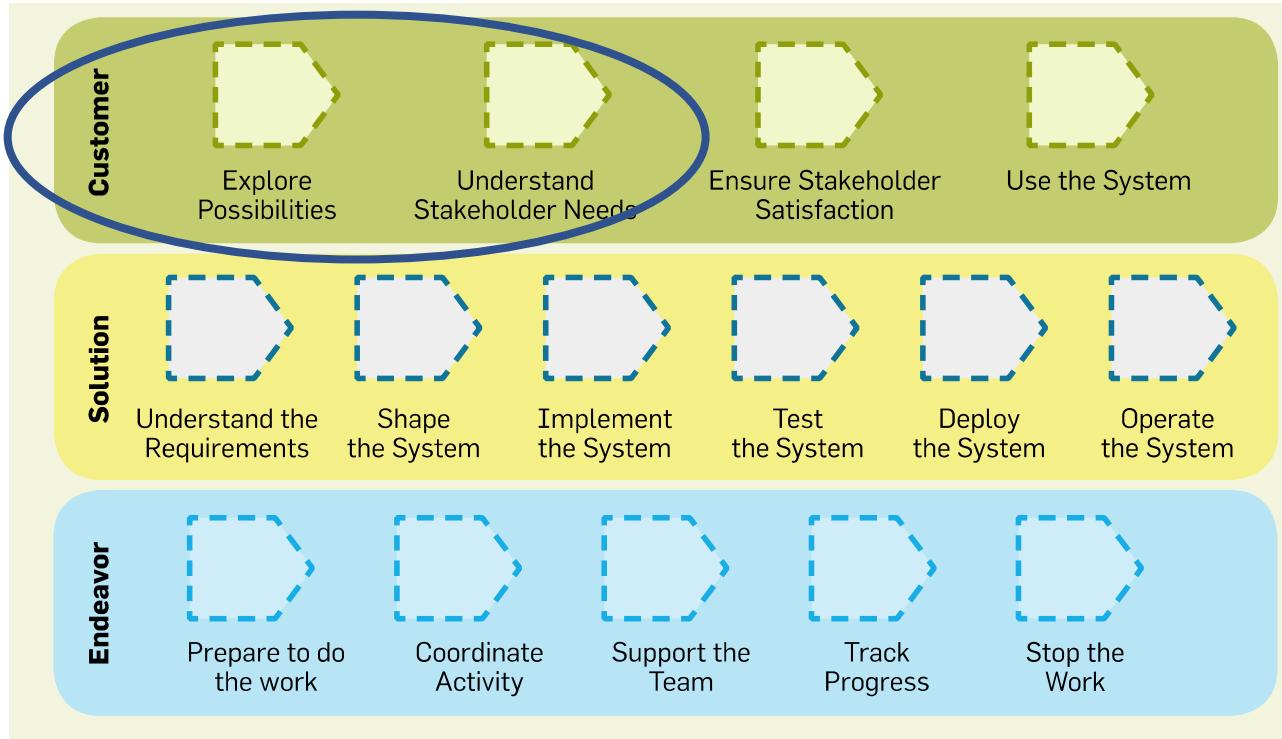
IS IT LINEAR?

Who is involved and when and role?

How do we know quality is good enough?



Agree on a system for working together to get this work done
- time and cost and tech and skill constraints

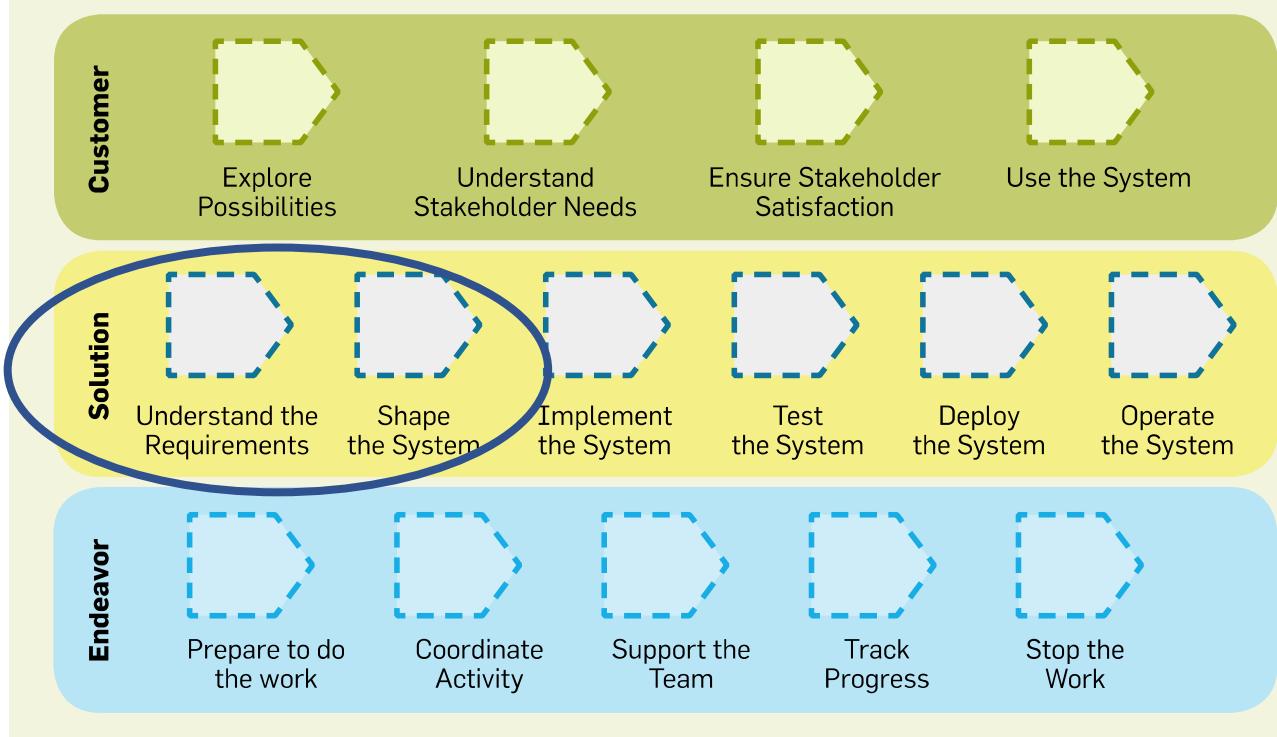


Work closely with the users and client – frequent interactions and coordination and feedback

Need a product goal /vision we share

Iterative incremental development

- so we can learn and discover by doing and getting feedback in rapid learning loops
- so we can deliver small bits of value to the client frequently
- break down the problem and solution



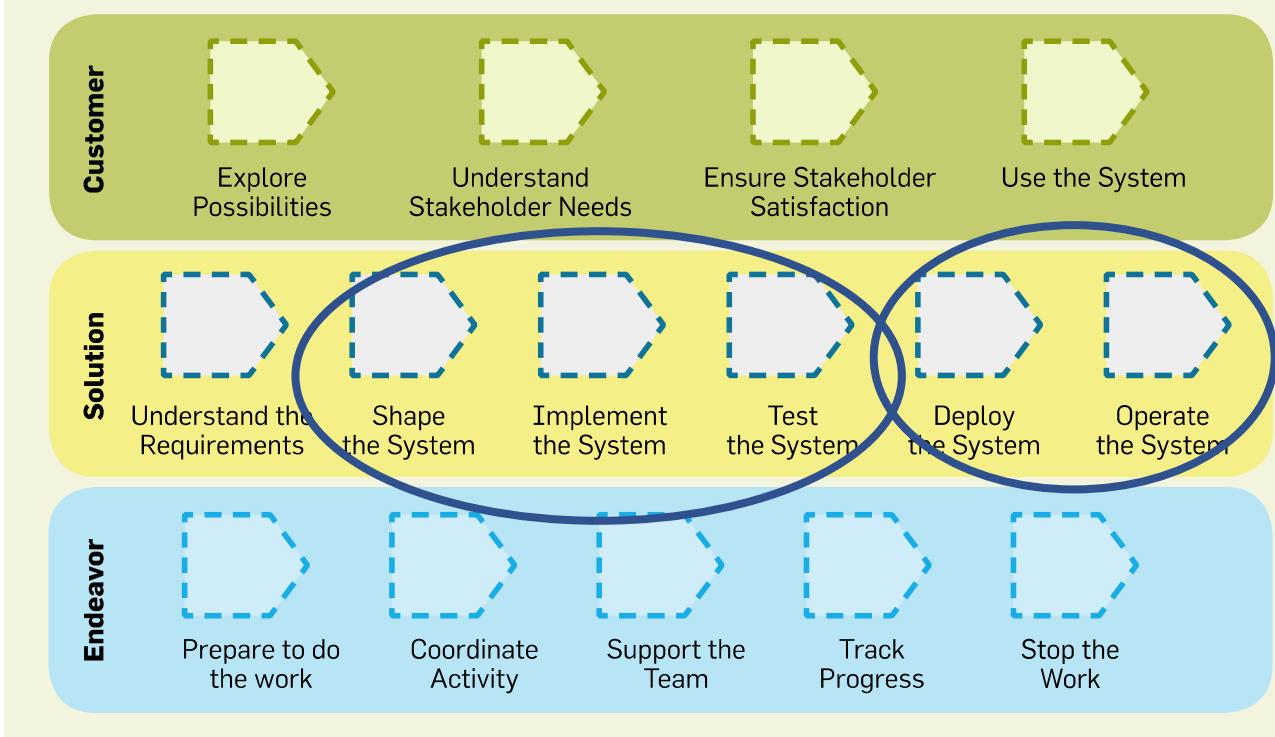
User stories

User Story Maps of product

Maintained list of what user requirements still needs to be done
-Have Iteration goals

List of user requirements for the next Iteration

Product Design incrementally



Overall architecture and tech stack

Web App
Layered architecture

React.js/NEXTjs for the UI (front-end)
MongoDB for the Document Database
Nest/Express.js and Node.js for back end

Iterative and incremental

Git and GitHub for version control and sharing and merging code

Continuous Integration (GitHub)

Continuous Deployment

Test-driven development

Pair and mob programming

Unit tests

Acceptance tests (BDD)

Other tests?

Cloud based deploy - Heroku

Let's talk to the client (Product Owner PO) Peter

Goal – get a vision of what the problem is and why it is a problem and what needs to change if our product is used.

End with a Product Goal statement

Start to break the problem into EPIC User stories

Principles

Ask OPEN question

Active Listening

Stay in Problem space

Write things down – these will become user stories

Engaging with the client (Product Owner PO) Peter

Active Listening

Reimold, C. (1991). 'Hey, don't listen to me like that!' How the way you listen can make or break communication. IPCC 91 Proceedings The Engineered Communication, <https://ieeexplore.ieee.org/abstract/document/172722>

Active listening—

Showing you have thoroughly understood the information imparted

Active listening has two parts to it:

1. You listen closely to get the essence of what the speaker is saying.
2. You restate what you think he said to make sure you've understood. ("Let me see if I've understood up to here." "Do you mean...?")

Creative listening—

Building on the speaker's ideas to form something new: a new idea, a new application, a new solution to a problem

Supportive listening—

Giving the speaker emotional sustenance by showing you empathize with his or her feelings

Continuous Integration in a nutshell

Local development machine with Git.

Local files with code, tests, configurations etc

Modify

(Use VS Code to change Code and Test)

Commit

(Snapshot with description of change and why and add to local repo)

Modify

Commit

Etc

(can roll back to previous version of code, create and work in branches etc)

Ready to merge with pre-production branch codebase (*integrate*)

Push to team shared GitHub Repository

(Automatically) run tests on whole code based before merging – CI server?

Automatically run code standard checks with Linter or SonarQube or similar

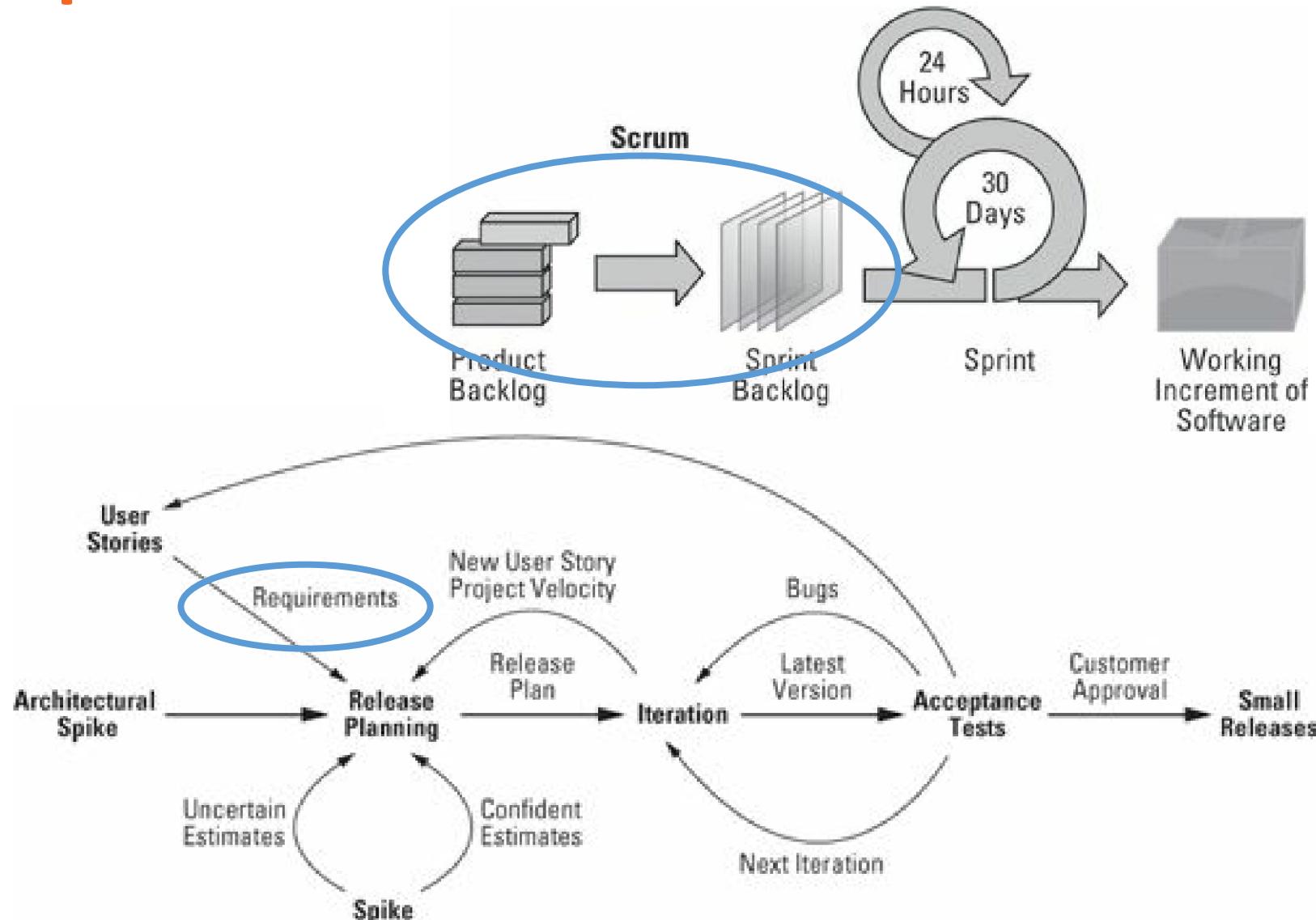
Automatically merge if pass tests

Continuous = every few hours (before feature finished)

GitHub with linked repo

With code from all developers and previously developed code merged and tested.

RE in Agile is iterative – not big upfront



Iterative Enhancement – Not New !

"battle lines between proponents of **agile software development ecosystems** (ASDE's) and **rigorous system development methodologies** (RSM's), based upon fundamentally different assumptions about how the world and organizations work".

WHAT HAS BEEN WILL BE AGAIN, AND WHAT HAS BEEN DONE WILL BE DONE AGAIN; THERE IS NOTHING NEW UNDER THE SUN.

- ECCLESIASTES 1:9

BUT

As agile methods become more popular, some view

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*(4), 390-396.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6), 47-56. doi:10.1109/MC.2003.1204375

Iterative Enhancement – 1975 and earlier !

iterative, evolutionary, and incremental software development—a cornerstone of these {agile} methods — as the “modern” replacement of the waterfall model,

but its practiced and published roots **go back decades.**[Larman & Basili]

The **first step** in the application of the iterative enhancement technique to a software development project consists of a **simple initial implementation of a skeletal subproblem** of the project. (Basili & Turner)

WHAT HAS BEEN WILL BE AGAIN, AND WHAT HAS BEEN DONE WILL BE DONE AGAIN; THERE IS NOTHING NEW UNDER THE SUN.
- ECCLESIASTES 1:9

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*(4), 390-396.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6), 47-56. doi:10.1109/MC.2003.1204375

Iterative Enhancement – Early Backlogs?

This skeletal implementation **acts as an initial guess** in the process of developing a final implementation which meets the complete set of project specifications.

A *project control list* is created that **contains all the tasks that need to be performed** in order to achieve the desired final implementation.

At (lny given point in the process, the project control list acts as a measure of the "distance" between the current and final implementations.

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*(4), 390-396.

Iterative Enhancement – Updating the control list?

In the **remaining steps** of the technique the **current implementation is iteratively enhanced** until the final implementation is achieved.

Each **iterative step** consists of selecting and removing **the next task from the list**, **designing** the implementation for the selected task (the *design phase*), **coding and debugging** the implementation of the task (the *implementation phase*), performing an **analysis of the existing partial implementation** developed at this step of the iteration (the *analysis phase*), and **updating the project control list** as a result of this analysis.

The **process is iterated** until the project control list is empty, i.e., until a final implementation is developed that meets the project specifications.

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*(4), 390-396.

A Requirements Refinery for Agile Software Product management

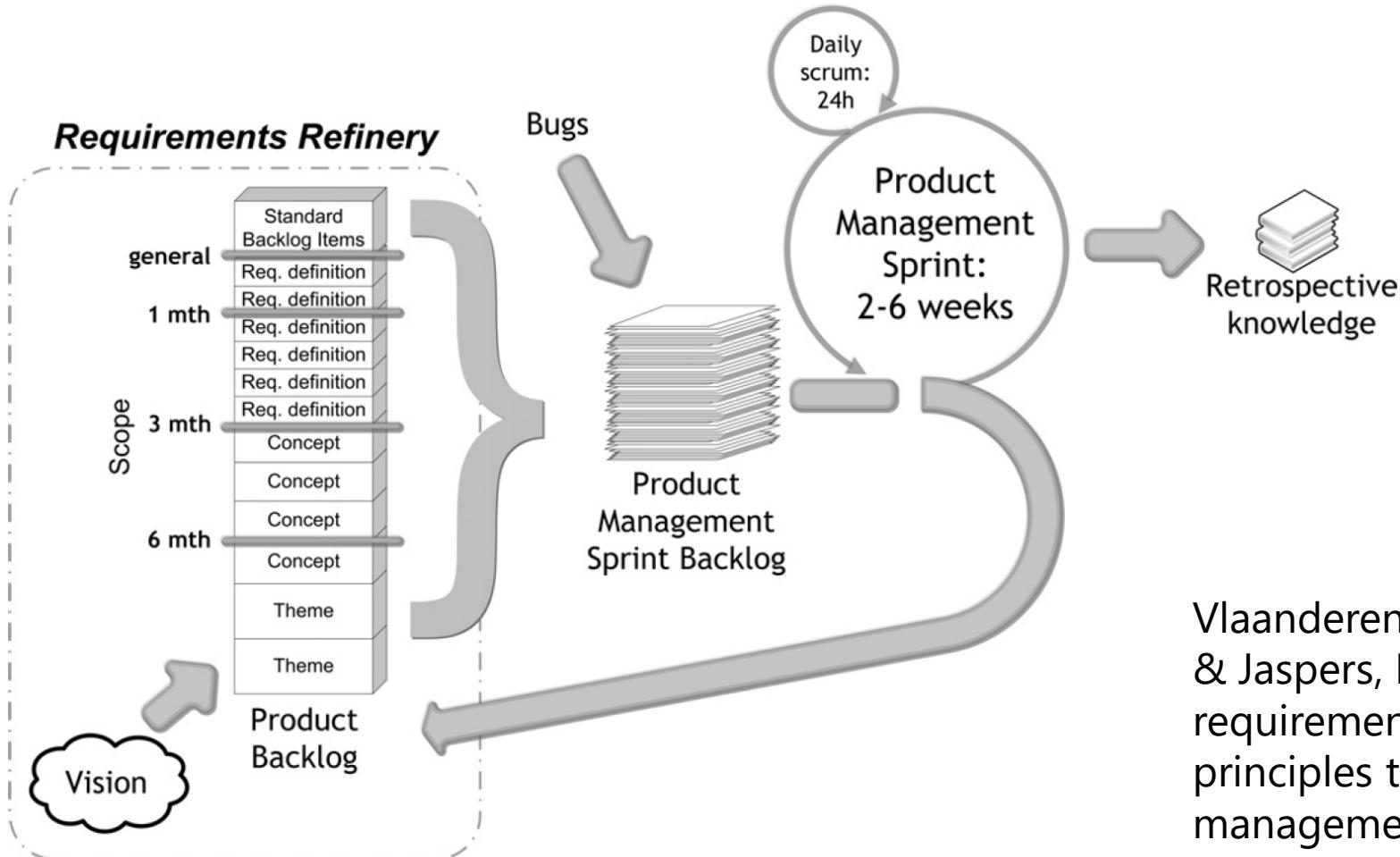


Fig. 1. Agile SPM knowledge flow.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58-70.
doi:10.1016/j.infsof.2010.08.004

Software Product management and Software Development

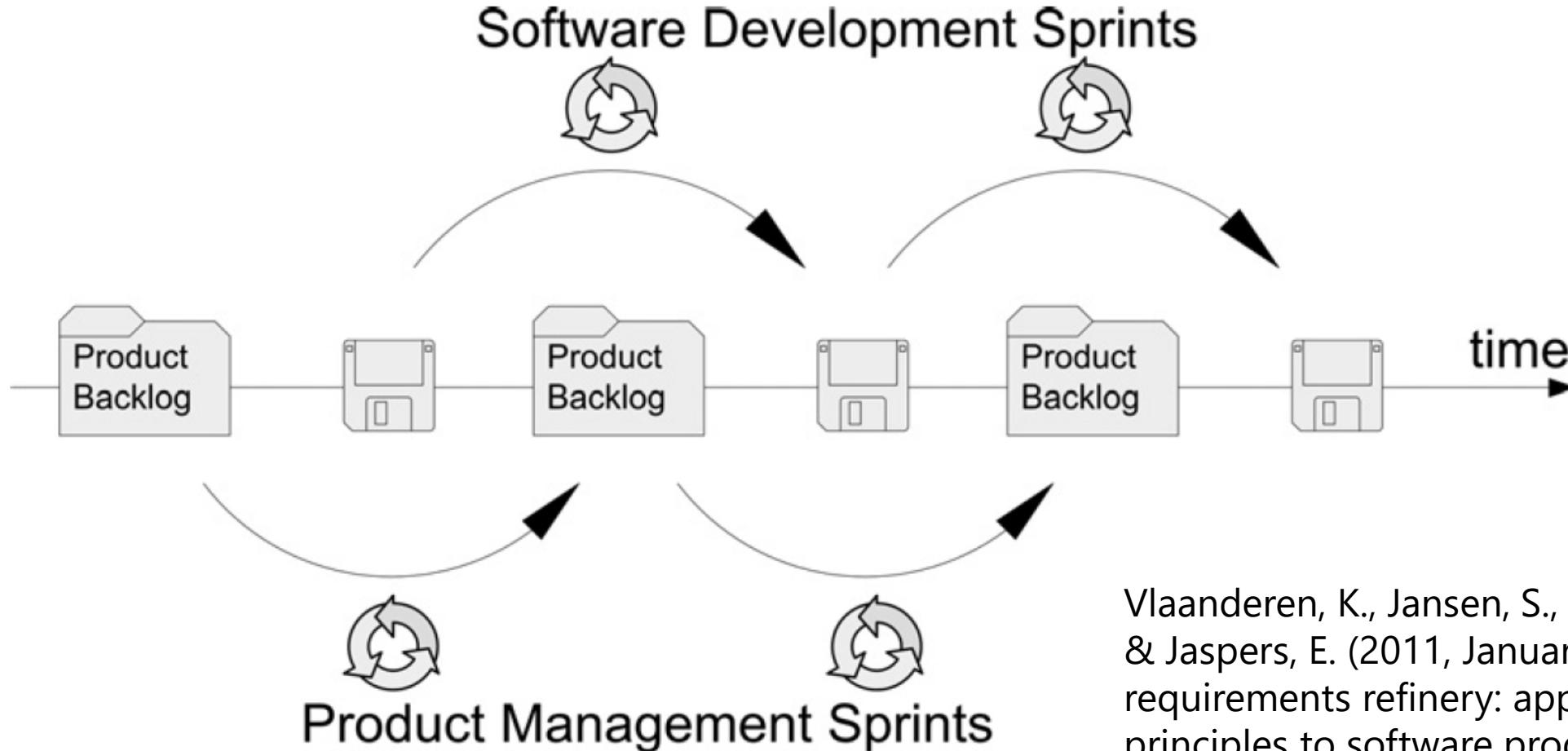
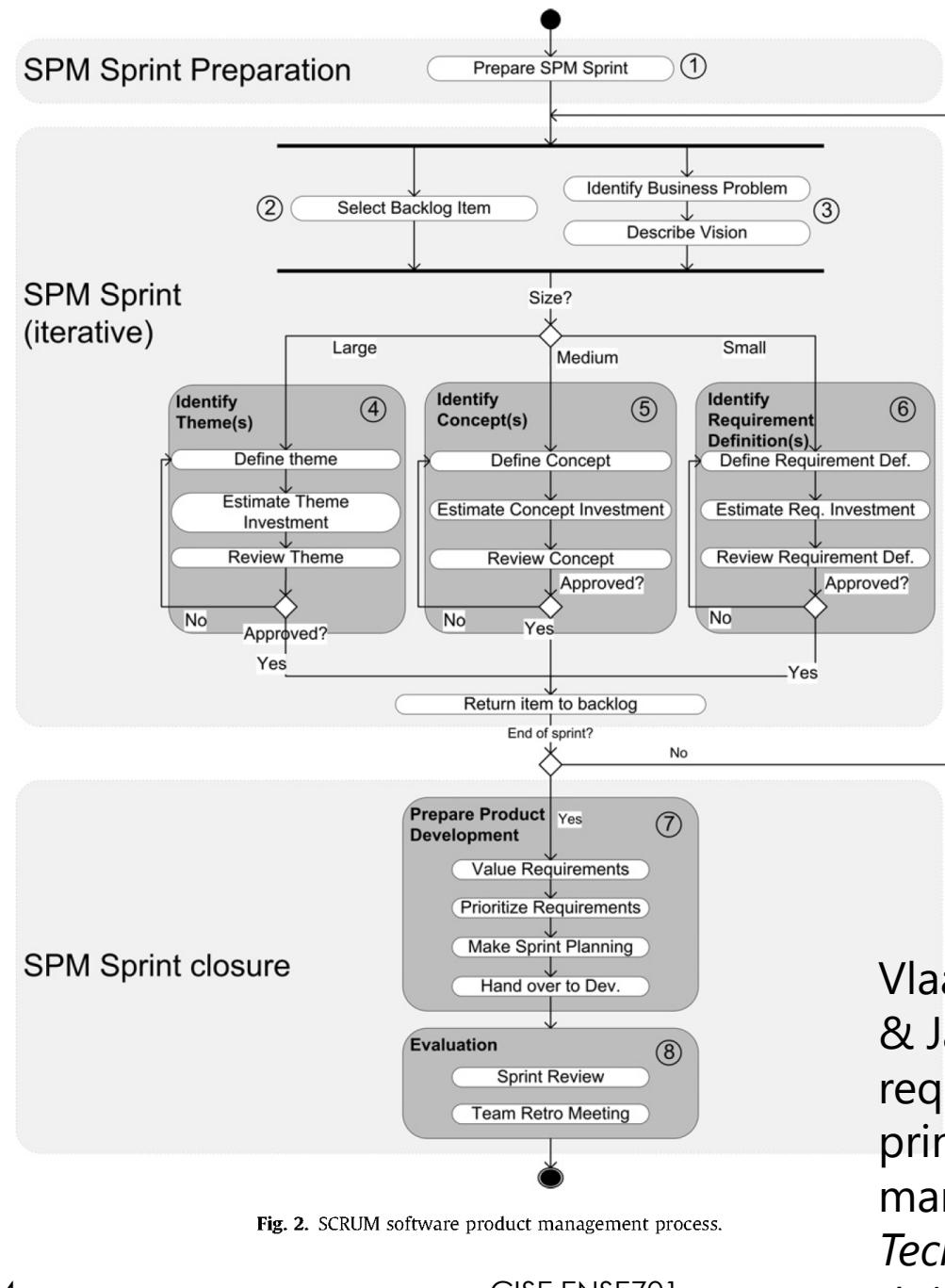


Fig. 3. Alternating sprints.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58-70.
doi:10.1016/j.infsof.2010.08.004

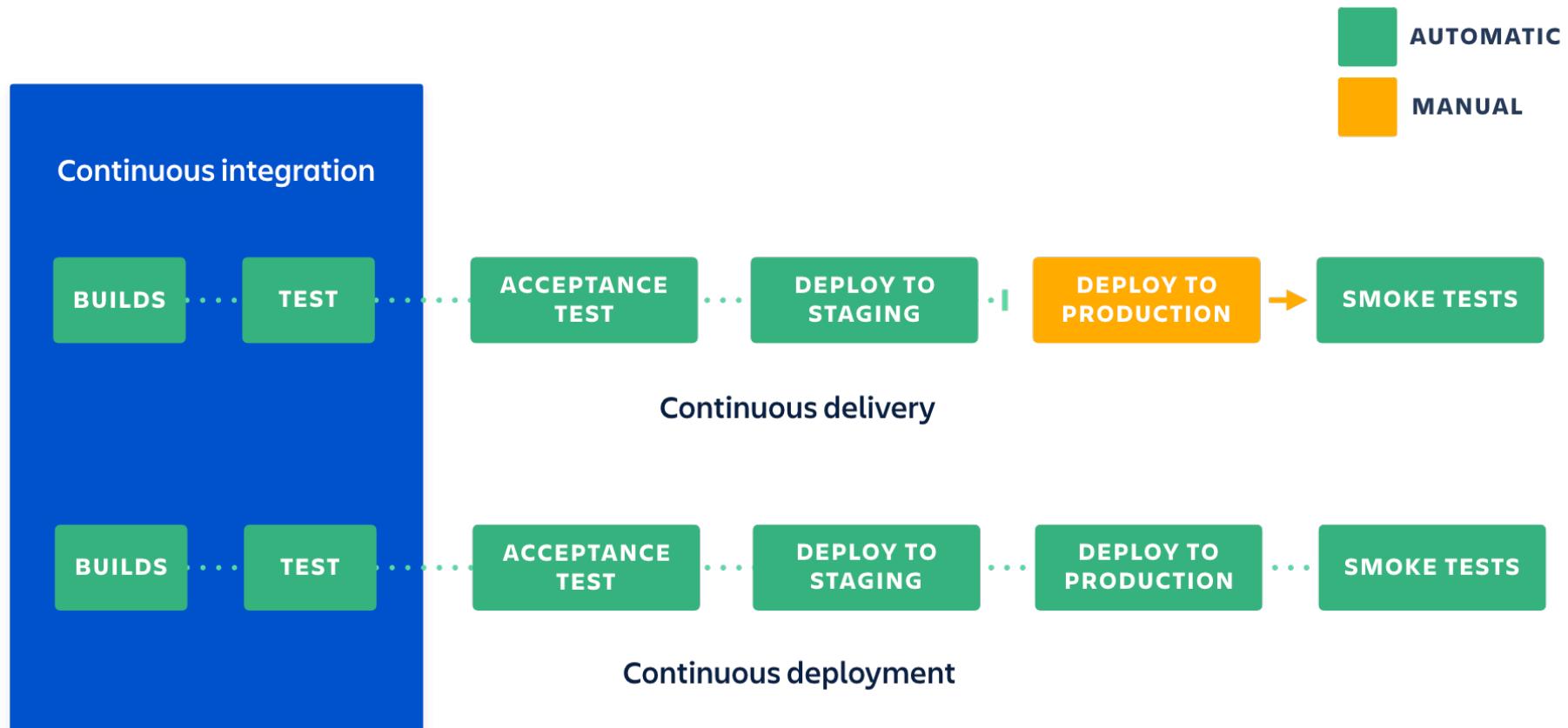
Software Product management Process



Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58-70.
doi:10.1016/j.infsof.2010.08.004

CI/Continuous Deployment (or Delivery) Overview

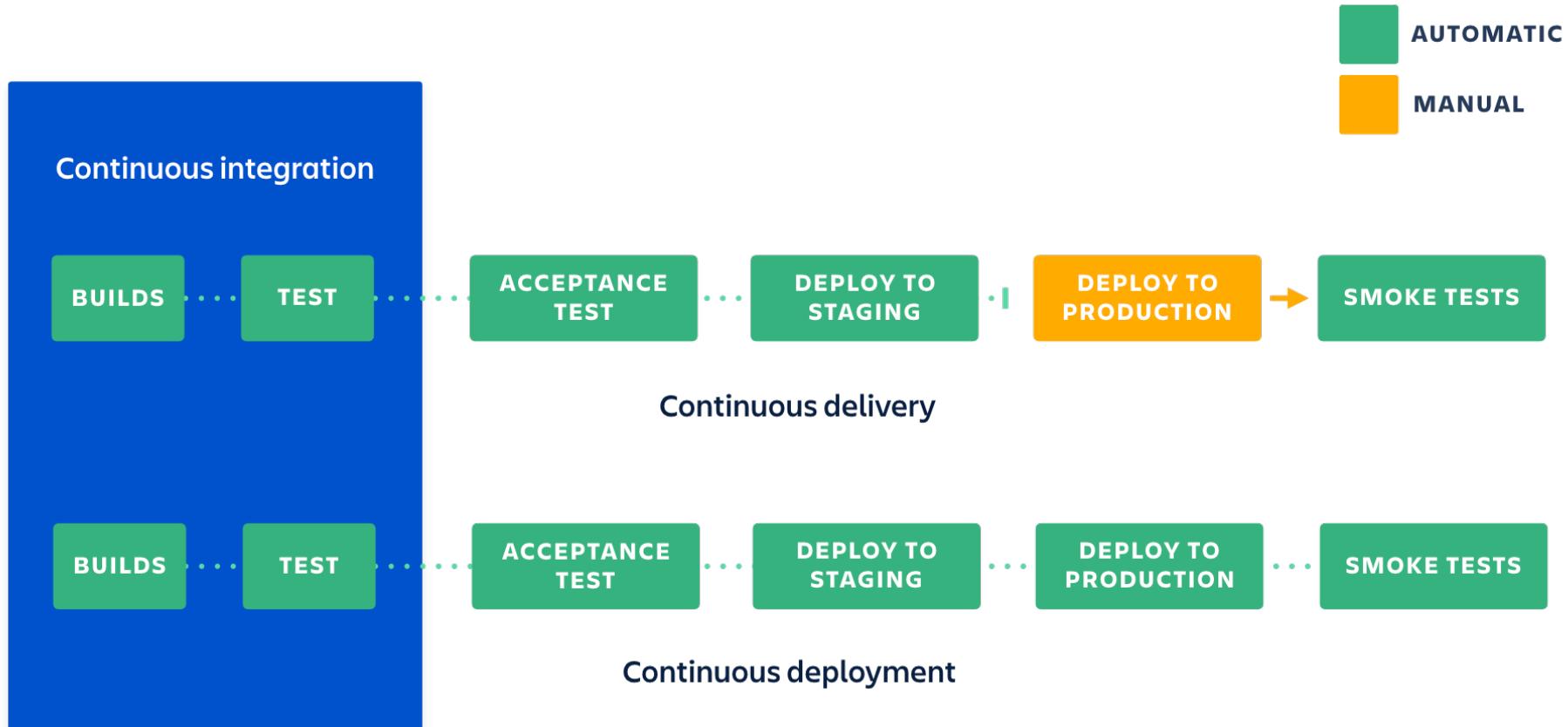
<https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>



CI/Continuous Integration in a Nutshell

<https://www.youtube.com/watch?v=1er2cjUq1UI>

Eric Minick IBM Cloud

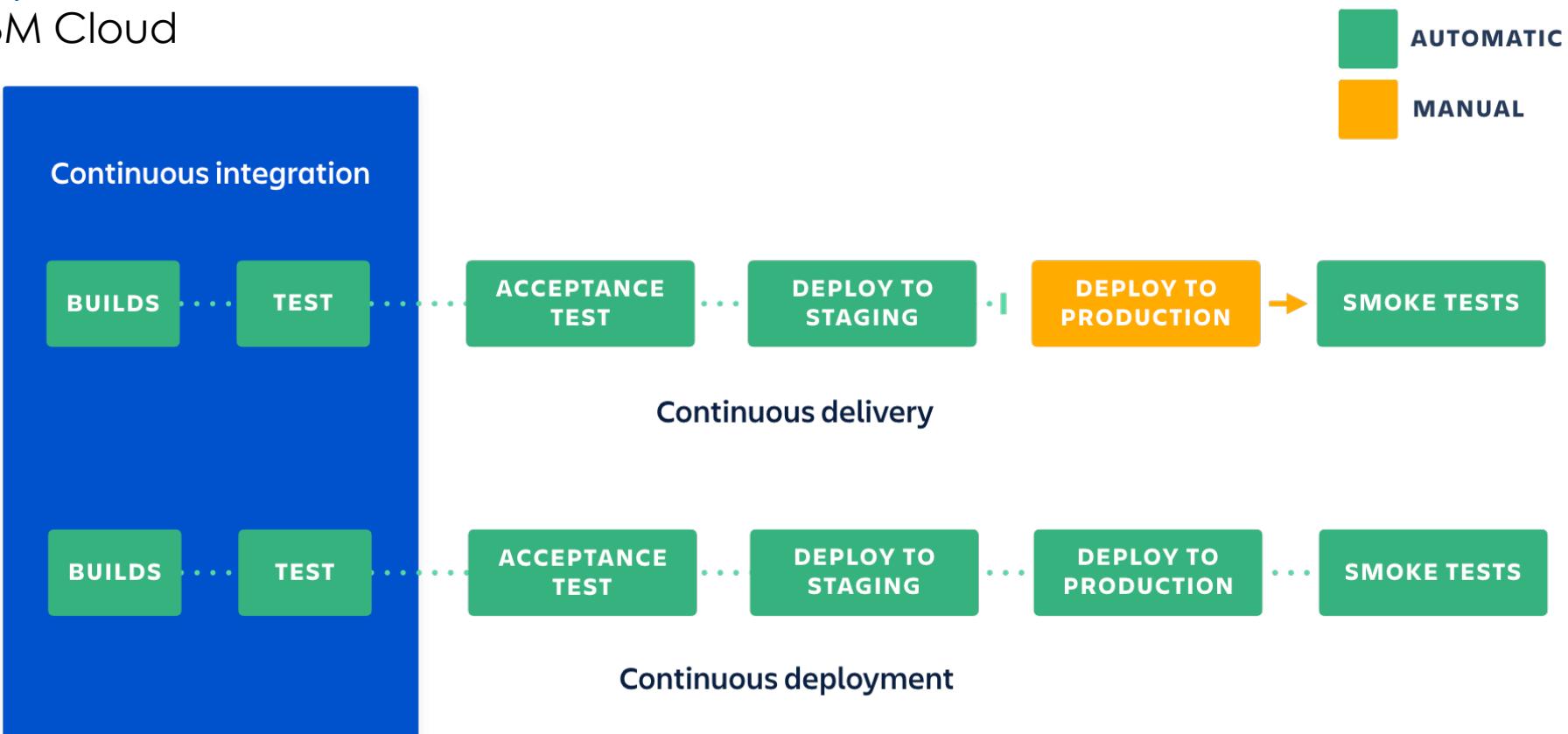


CI/CD Continuous Deployment (or Delivery) in a Nutshell

<https://www.youtube.com/watch?v=2TTU5BB-k9U>

<https://www.youtube.com/watch?v=LNLKZ4Rvk8w>

Eric Minick IBM Cloud



DevOps as a Wow and values – we will come back to this

Values

- Deploy/release frequently (several time a day?)
- Own the product
- Team accountability extends to deployment/release and post-release

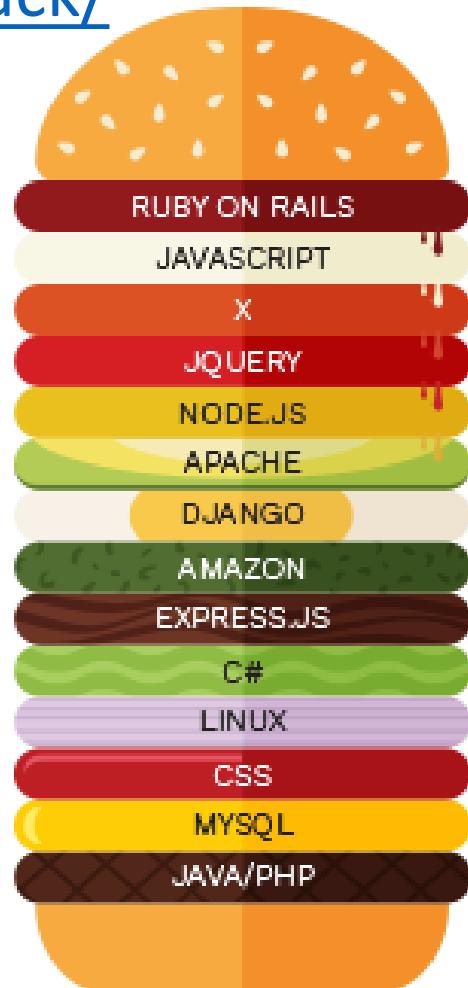
Enablers

- test automation, (CI/CD automation)
- deploy automation,
- infrastructure as code
- rollback/fix forward,
- A/B and canary testing,
- feature flags,
- microservices,
- increased observability -monitoring, alerting and logging post release
- Psychological safety



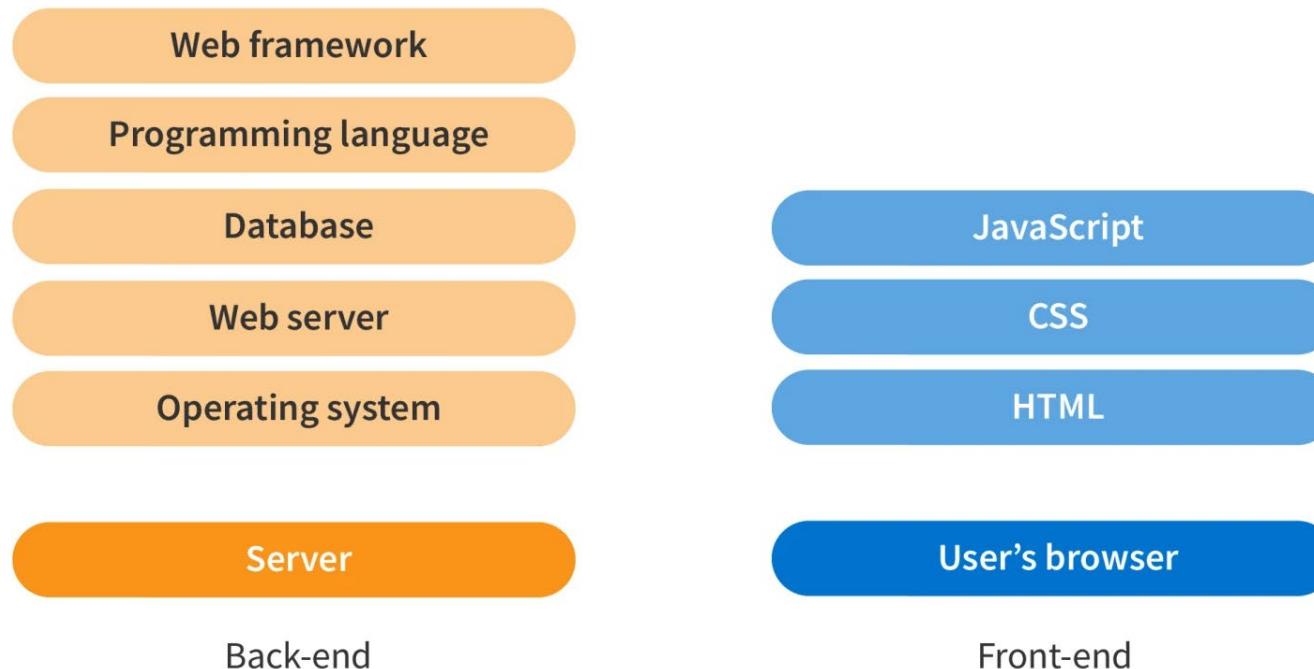
Technology stacks and full-stack developers

<https://www.thesoftwareguild.com/blog/build-your-own-technology-stack/>



Technology stacks and strategy

<https://www.aha.io/roadmapping/guide/it-strategy/technology-stack>



© 2021 Aha! Labs Inc.

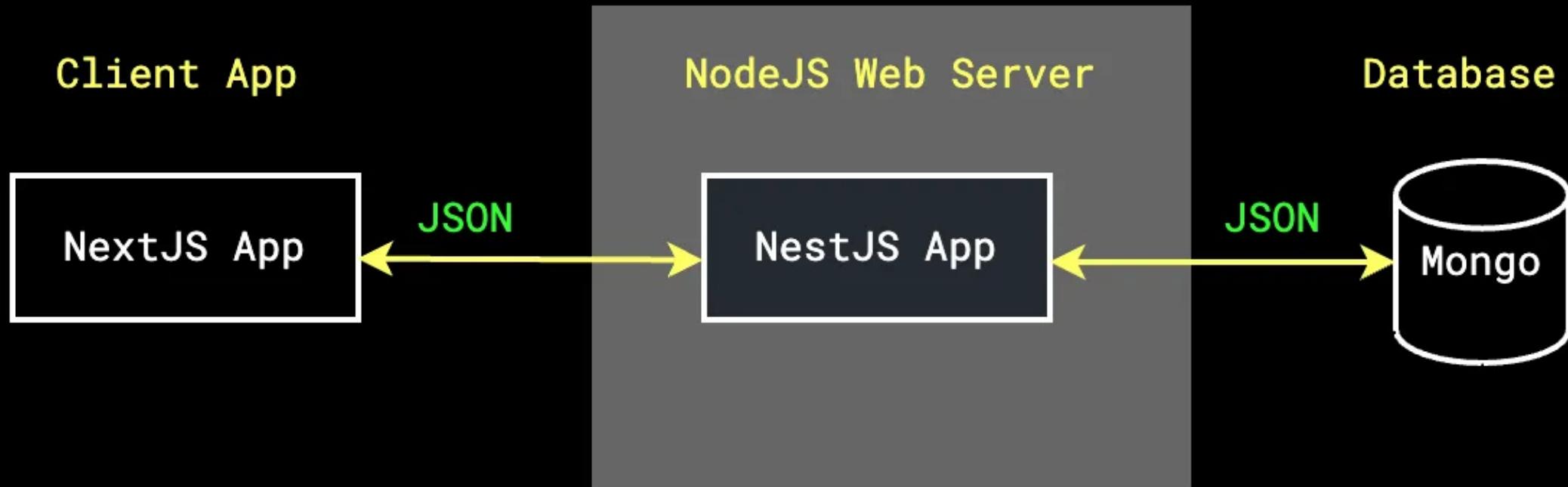
MNNN Stack in a nutshell



Readings

- <https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9>
- <https://medium.com/aws-tip/understanding-nestjs-architecture-f257d054211d>

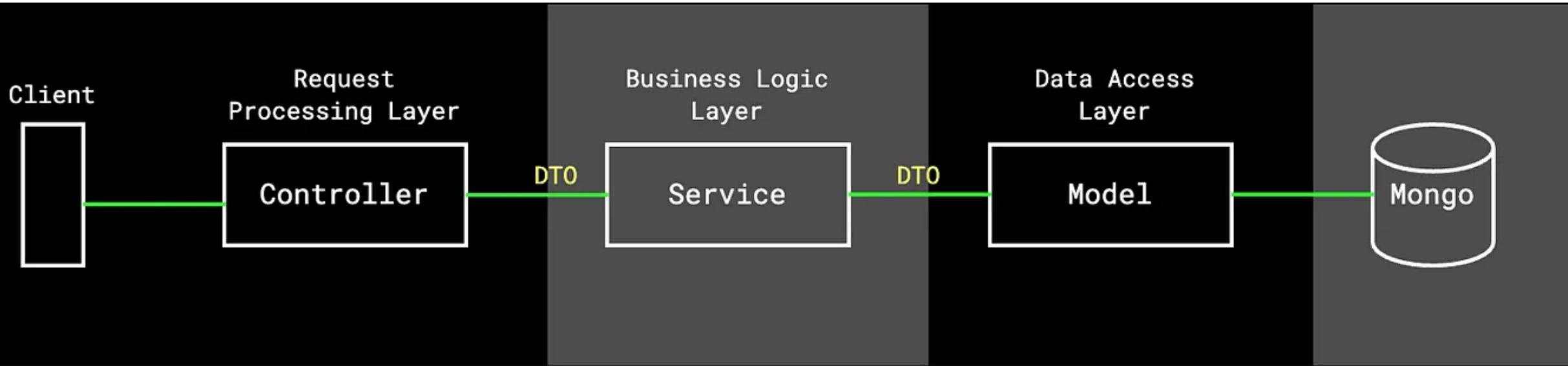
MNNN Components



Read this

<https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9>

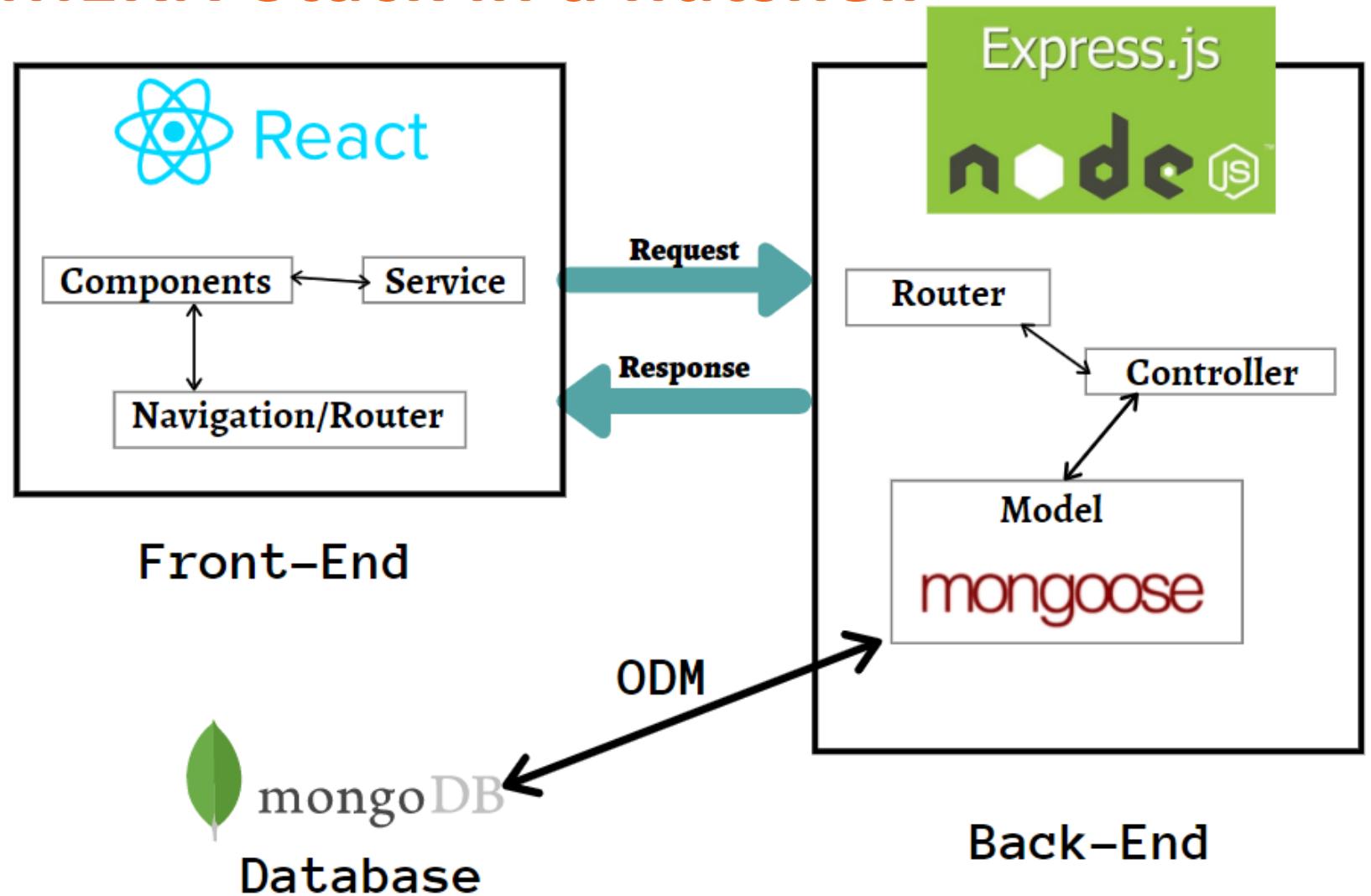
MNNN Layered Architecture



Read this

<https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9>

An Alternative - MERN Stack in a nutshell



Read this

<https://medium.com/techiepedia/what-exactly-a-mern-stack-is-60c304bffbe4>

MongoDB

Document based noSQL database
Cloud based service – Mongo Atlas

Mongodb.com

```
json

{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```

Express.js and Node.js

Express - routing and server (middleware)

Node.js is a javascript runtime so javascript applications can be run outside the browser

[Expressjs.com](https://expressjs.com)

[Nodejs.org](https://nodejs.org)

Nest.js



Nest.js Crash Course 2023: A Comprehensive Step-by-Step Tutorial

Sakura Dev

But like all random resources off the internet they need to be viewed critically :-)

Video

<https://www.youtube.com/watch?v=Hv70fn8xTL4>

Agile values

Agile Manifesto – 4 key Values

12 key Principles

Scrum WoW

Guidelines for working based on empiricism– iterative and incremental workflow. Values and 3 pillars.

DevOps WoW

Continuous Integration

Continuous Delivery and Deployment

Automation of testing, deploy, infrastructure

Team Takes responsibility of Deploying

Team takes responsibility post deployment

Kanban WoW

Lean WoW

Focus on – visibility of work, limit WIP in work queues, reduce waste, improve cycle time

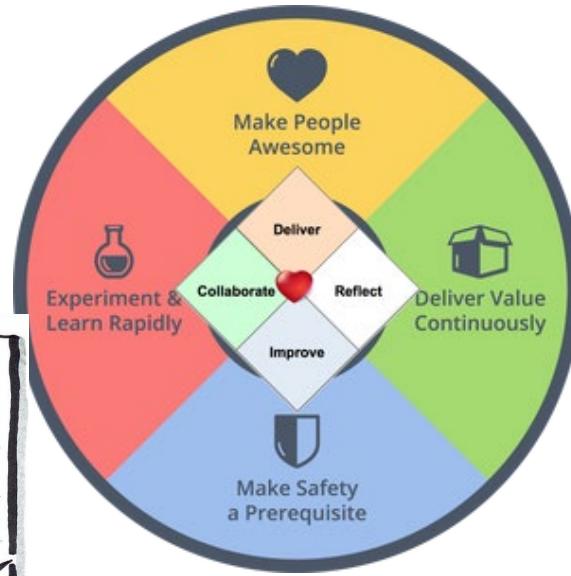
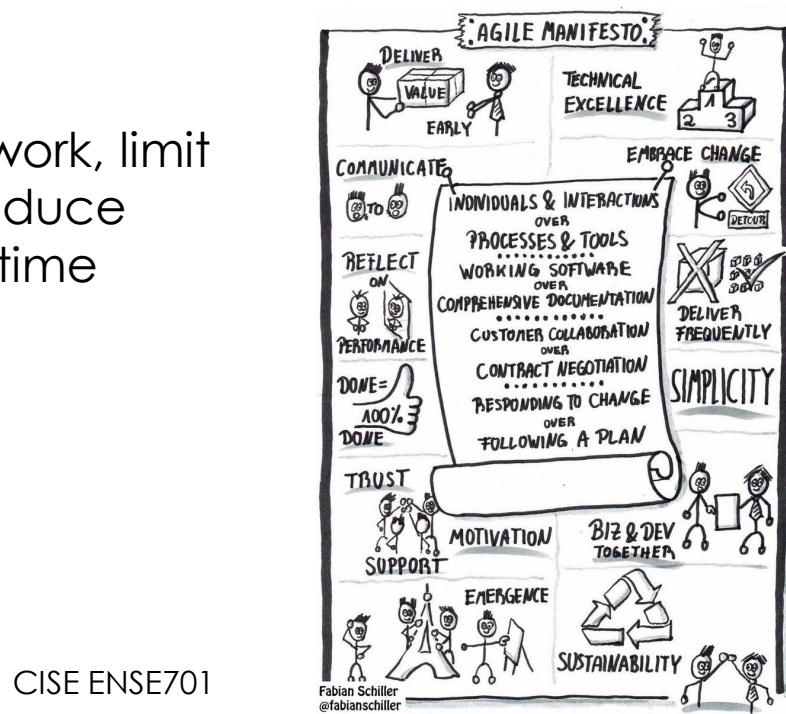
Continuous = frequent & small

Modern Agile – 4 key values

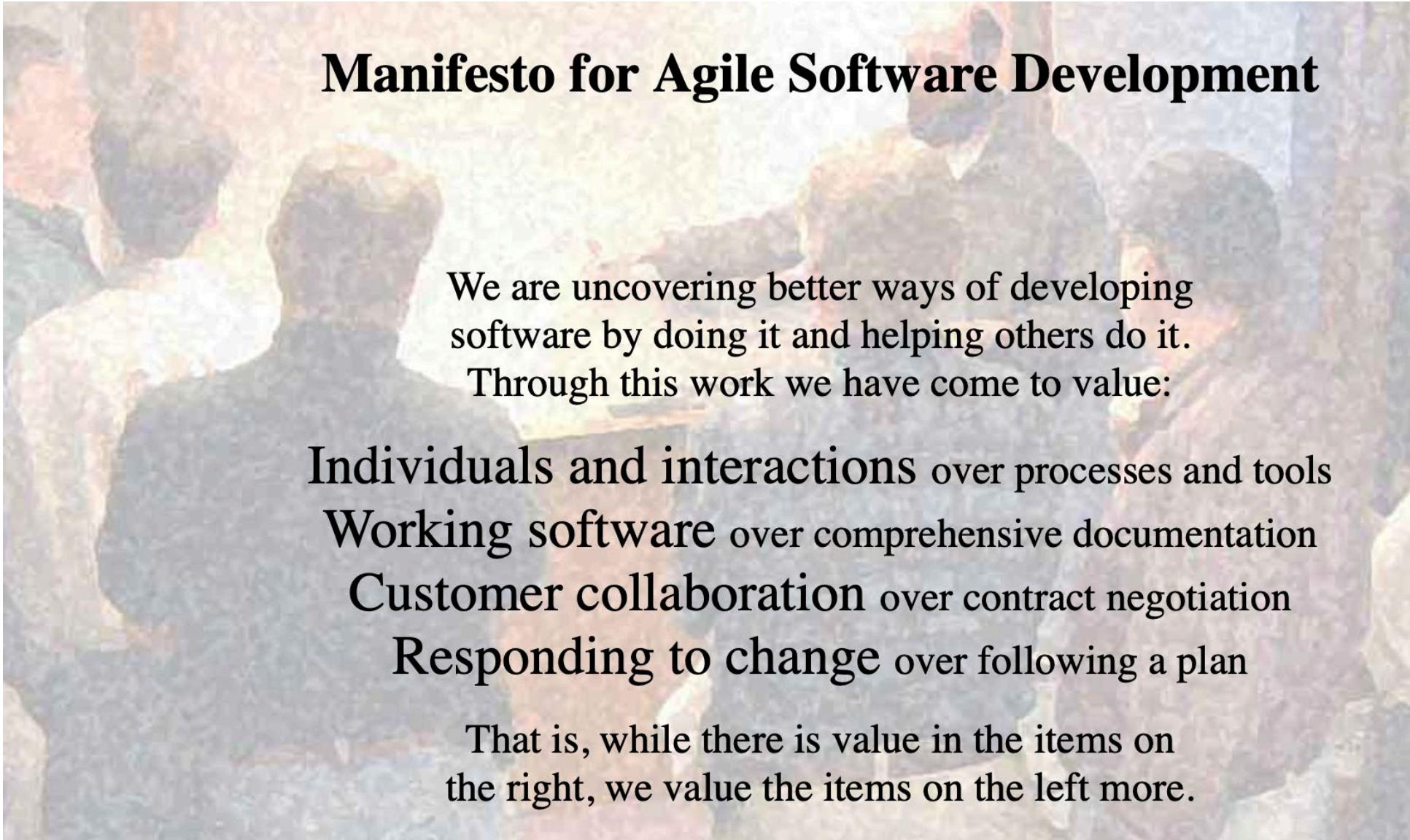
Heart of Agile – 4 key actions

Transparency

Inspect and Adapt



Scrum is Agile....what does that mean?



2001

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

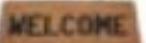
That is, while there is value in the items on the right, we value the items on the left more.

The Agile Manifesto – guides decisions, behaviour



Early and continuous delivery of valuable software

1



Welcome changing requirements even late in development

2



Deliver working software frequently

3



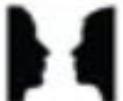
Business people and developers working together daily

4



Build projects around motivated individuals and trust them to get the job done

5



The most effective method of conveying information is face-to-face conversation

6



Working software is the primary measure of progress

7



Sustainable development: maintain a constant pace indefinitely

8



Continuous attention to technical excellence

9



KISS
keep it simple...
Simplicity: maximize the amount of work not done

10



Teams self-organize

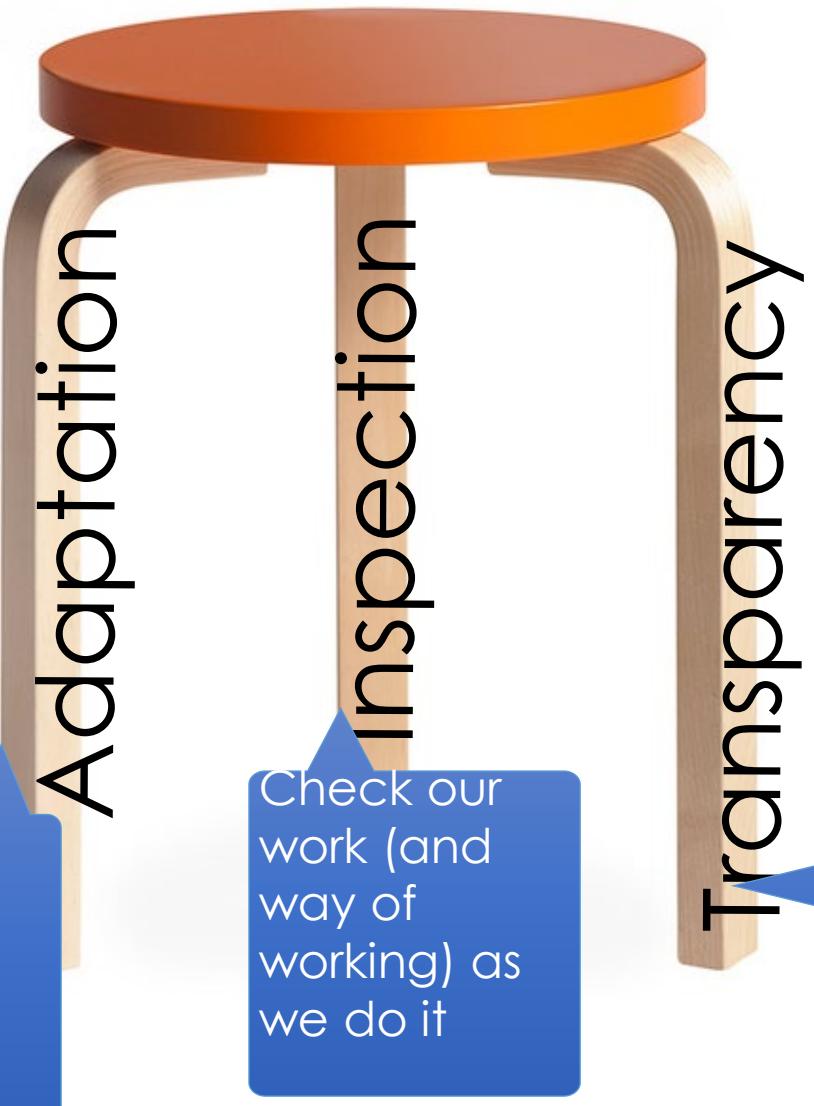
11



Teams regularly reflect and adjust behaviour

12

The 3 foundational ideas of Agile



- Adaptation
 - willingness to change if it makes sense
 - understand implications of change and adapt
 - willingness to experiment to check alternatives
 - Desire to learn and improve

- Inspection
 - willingness to reflect on what has happened
 - curious and questioning
 - Proactive about improvement (not content with status quo)
 - Plan action (adaptation) based on evidence
 - Evaluation of action based on evidence
 - willing to put in effort

Transparency

Workflows and progress are very visible to everyone

Decisions are collaborative where it makes sense

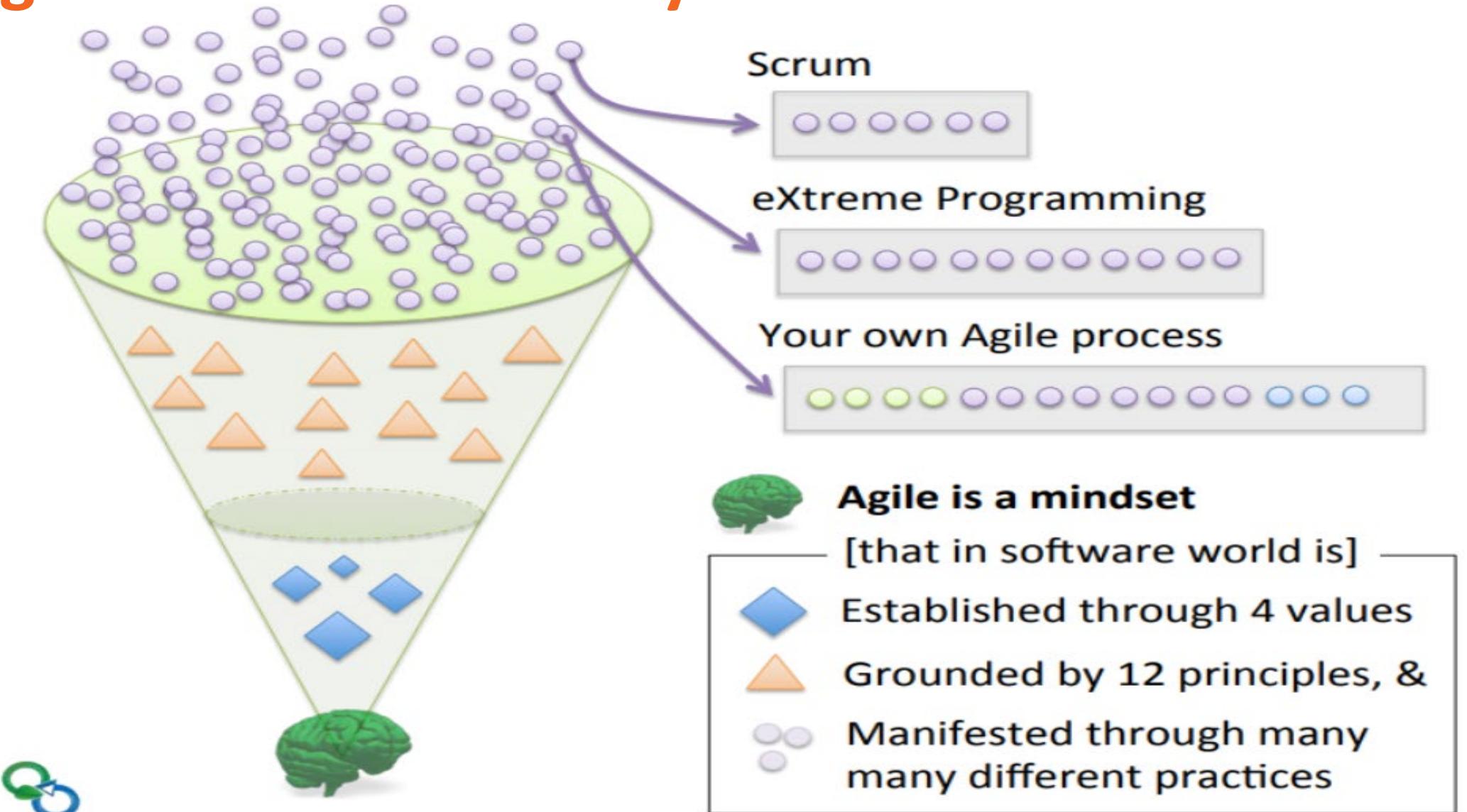
Different points of view are sought proactively

Honesty and openness are normal (tempered with empathy)

The truth is visible

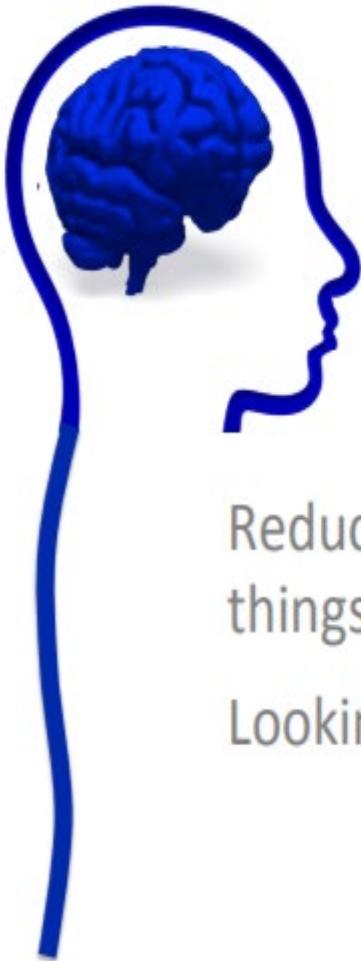
Mistakes and questions are ok

The Agile Manifesto – many WoW



Webinar by Ahmed Sidky – CEO of ICAgile course
<https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf>

Approach to uncertainty with an Agile mindset



Fixed Mindset
approach to
managing
uncertainty

Reducing uncertainty by “nailing things down.”

Looking to fix and confirm things.



Agile Mindset
approach to
managing
uncertainty

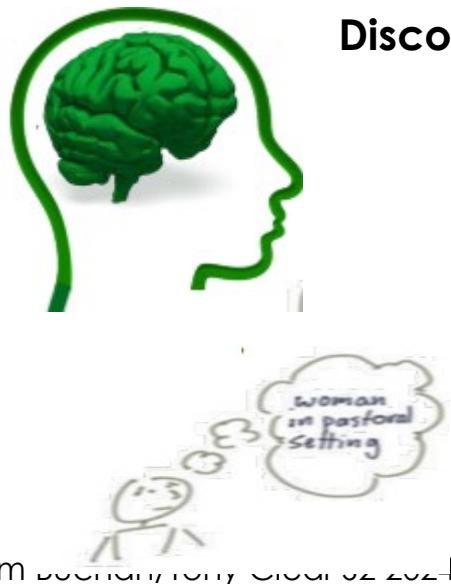
Reducing uncertainty by discovering and learning.

Looking to learn and discover in the most efficient way possible.

Incremental delivery with Agile mindset



Must “nail down” the output in order to start delivery (Linear Thinking)

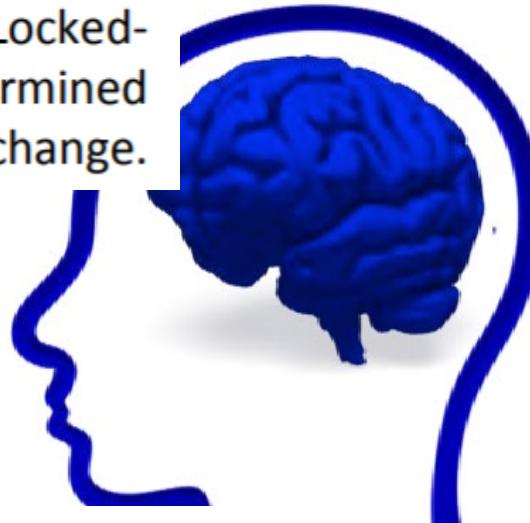


Discover and learn through valuable output and welcoming change (Circular Thinking)

The Growth mindset and the Agile Mindset

Carol Dweck

I believe that my **[Intelligence, Personality, Character]** is inherent and static. Locked-down or fixed. My potential is determined at birth. It doesn't change.



Avoid failure

Desire to Look smart

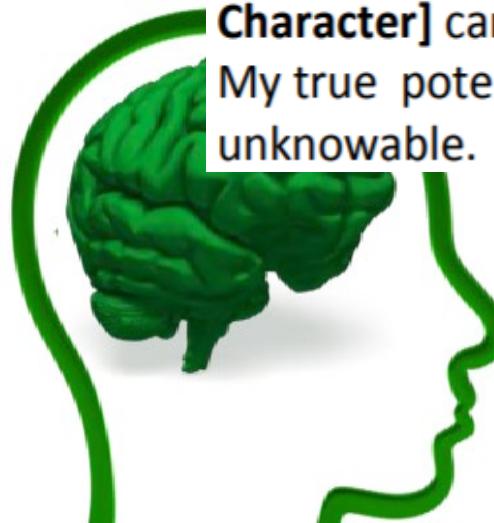
Avoids challenges

Stick to what they know

Feedback and criticism is personal

They don't change or improve

I believe that my **[Intelligence, Personality, Character]** can be continuously developed. My true potential is unknown and unknowable.



Desire continuous learning

Confront uncertainties.

Embracing challenges

Not afraid to fail

Put lots of effort to learn

Feedback is about current capabilities

These practices are often associated with Agile WoW

- Product visioning
- Project chartering
- Affinity (relative) estimation
- Size-based (point) estimation
- Planning poker
- Group estimation
- Value-based documentation
- Prioritized product backlog
- User stories
- Progressive elaboration
- Personas
- Story maps / MMF
- Story slicing
- Acceptance tests as requirements
- Short iterations
- WIP Limits
- Early and frequent releases
- Roadmapping
- Velocity-based planning and commitment
- Iteration planning / Iteration backlog
- Release planning / Release backlog
- Time boxed iterations
- Adaptive (multi-level) planning
- Risk backlog
- Team structure of VT / DT
- Pull-based systems
- Slack
- Sustainable pace

- Frequent face-to-face
- Team chartering
- Cross-silo collaborative teams
- Self-organizing teams
- Cross-functional teams
- Servant leadership
- Task volunteering
- Generalizing specialist
- Tracking progress via velocity
- Burn-up/burn-down charts
- Refactoring
- Automated unit tests
- Coding standards
- Incremental/evolutionary design
- Automated builds
- Ten-minute build
- Monitoring technical debt
- Version control
- Configuration management
- Test driven development
- Pair programming
- Spike solutions
- Continuous integration
- Incremental deployment
- Simple design
- End-of-iteration hands-on UAT
- Automated functional tests
- Automated developer tests (unit tests)
- Exploratory testing
- Software metrics

Modern Agile

Joshua Kerievsky
Keynote Agile conference 2016

Rather than “customer collaboration over contract negotiation,” Modern Agile’s “**Make people awesome**” is more important for our focus. We want our entire ecosystem to be awesome.

Though “working software over comprehensive documentation” was great in 2001, today **we want to “deliver value continuously,”** he says. “The bar has been raised.”

“Responding to change over following a plan” was incredibly important when we wanted to defeat waterfall with agile, he explains. Now **we want to “experiment and learn rapidly”** because “sometimes we don’t even know what the problem is.”

“Individuals and interactions” could be replaced by **“make safety a prerequisite.”** We want to provide psychological safety in our interactions.



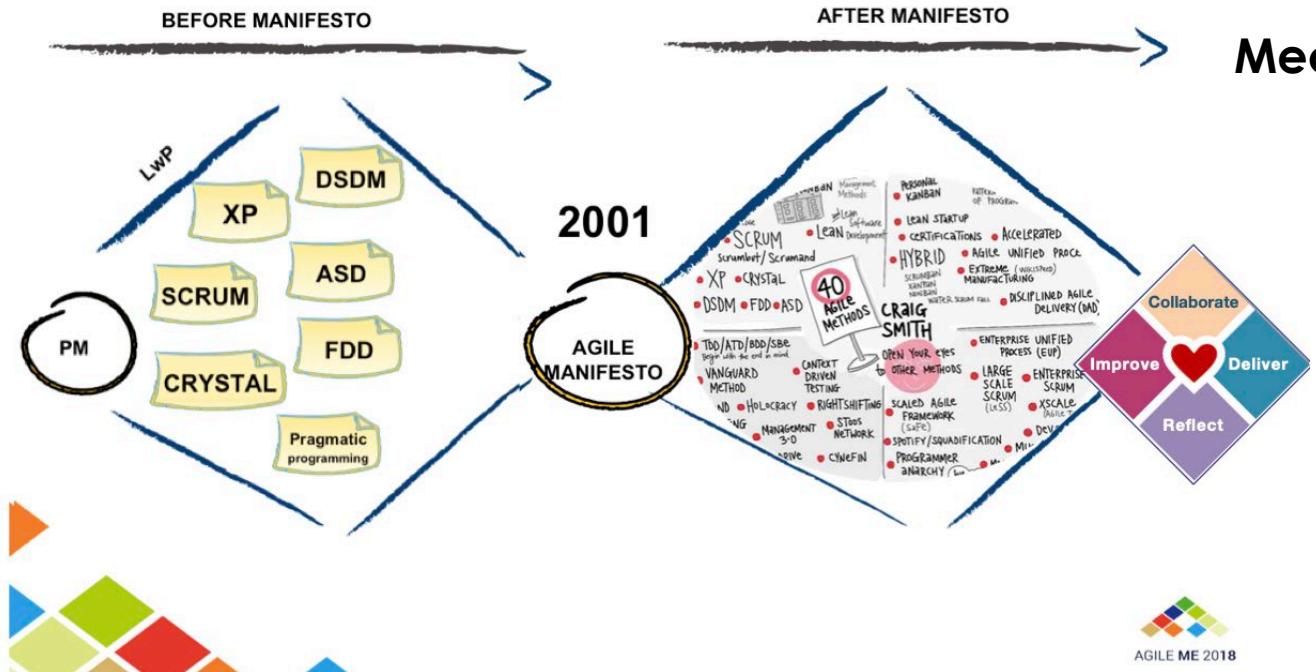
Heart of Agile

Oversimplified:

<https://heartofagile.com/lets-begin/>

144 LOs in Scrum training

Pierre Hervouet's recap of history:



\$29 - Cheap Certification, Free Books

Ad www.scrum-institute.org/

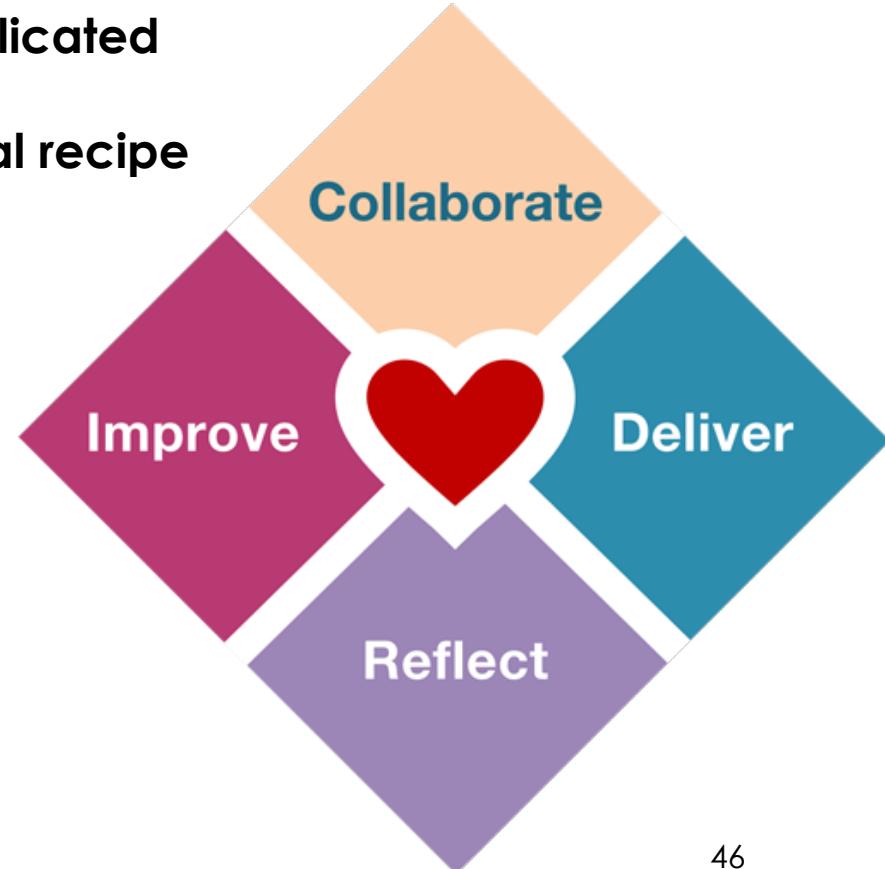
Online SCRUM Master Certification & Be SCRUM Certified Online in 1 Hour

100% Money Back Guarantee · 100% Pass Rate or Refund · Low Cost Scrum Development

Highlights: Online Scrum Training Materials, Multiple-Choice Test Questions...

Overcomplicated

Mechanical recipe



“Heart of Agile” - Talk

<https://heartofagile.com/video-of-the-latest-talk-on-heart-of-agile-by-alistair-cockburn-denmark-october-2018/>

<https://www.itu.dk/om-itu/presse/nyheder/2018/video-agil-ophavsmand-besoegte-danmark>

Last October [2018] Dr. Alistair Cockburn talked to more than 700 people at the IT University of Copenhagen about the lastest ideas and experiments on Heart of Agile.

- “**Heart of Agile**” Article from Crosstalk

<https://heartofagile.com/the-heart-of-agile-crosstalk-magazine/>

Well Before “Heart of Agile”

<https://alistair.cockburn.us/coming-soon/>

And reading for the really dedicated...

Cockburn, A. (2003). *People and Methodologies in Software Development* [Doctoral Dissertation, University of Oslo]. Oslo.

Retrieved 8/03/2022 from

https://www.researchgate.net/profile/Alistair-Cockburn/publication/253582591_People_and_Methodologies_in_Software_Development/links/56d434b208ae2ea08cf8e076/People-and-Methodologies-in-Software-Development.pdf

Or just read the abstract!!

Agile Modelling and Agile Ways of Working

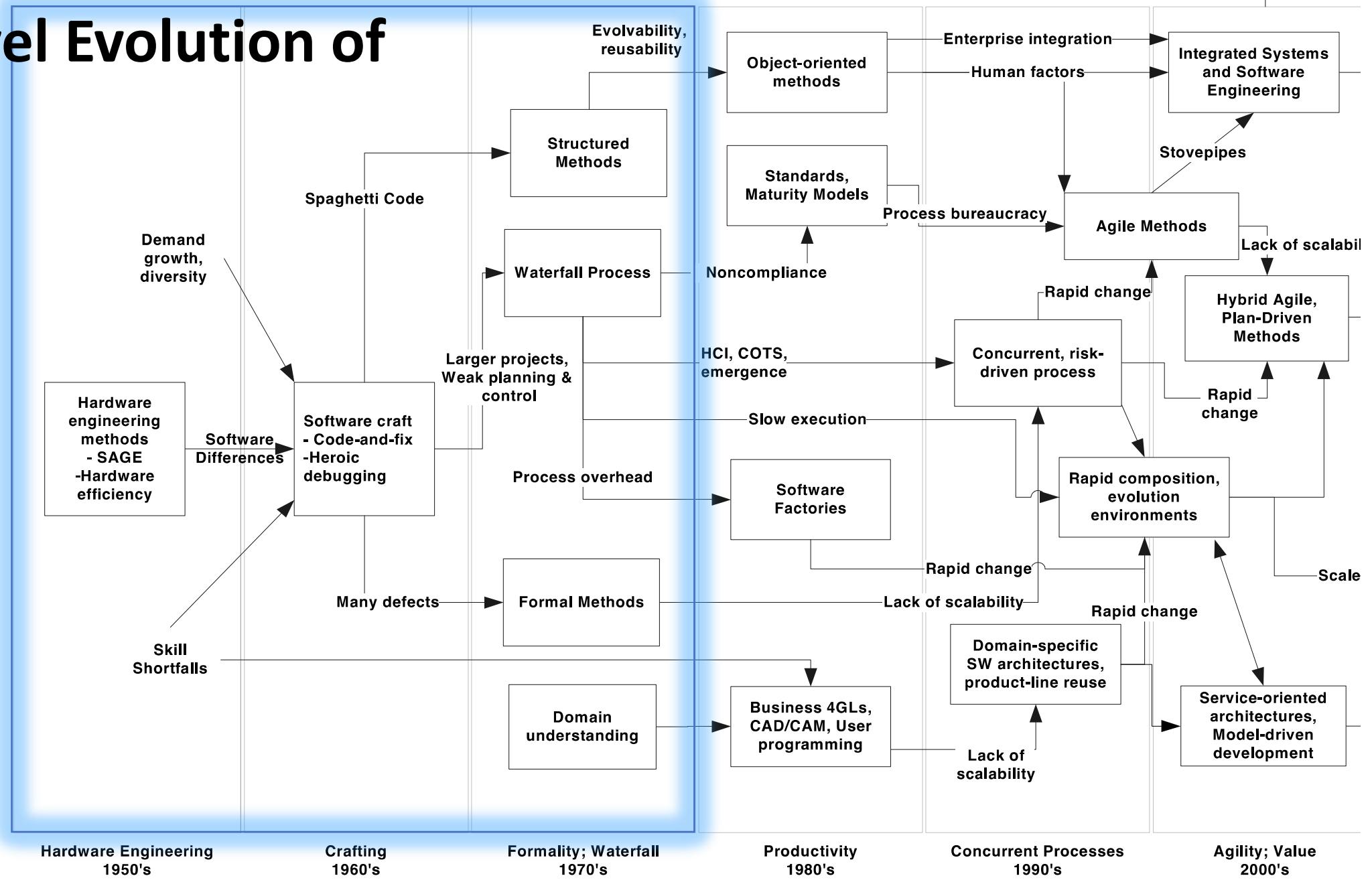
“The Criteria for Determining Whether a Team is Agile”

Scott Ambler

<https://agilemodeling.com/essays/agilecriteria.htm>

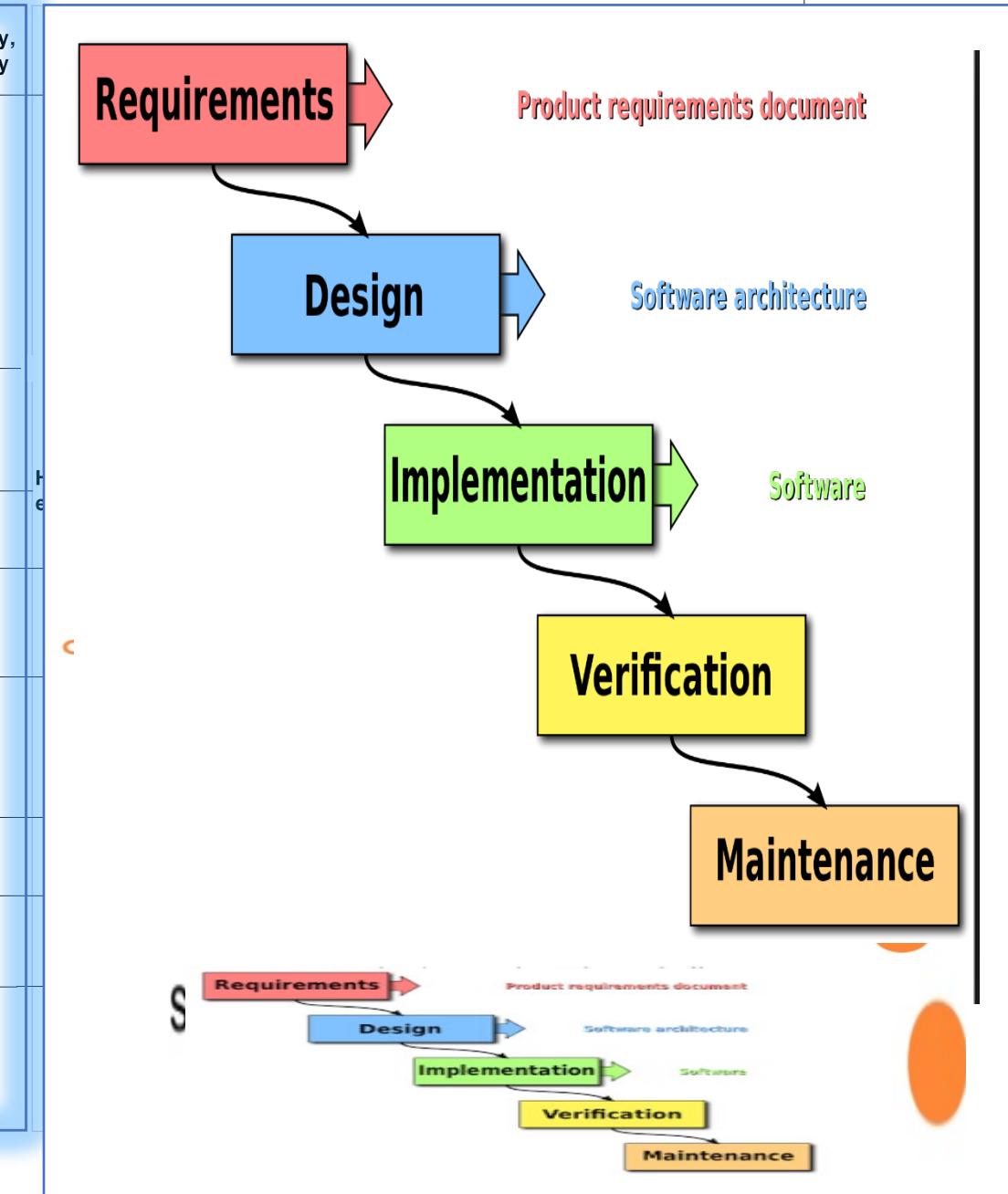
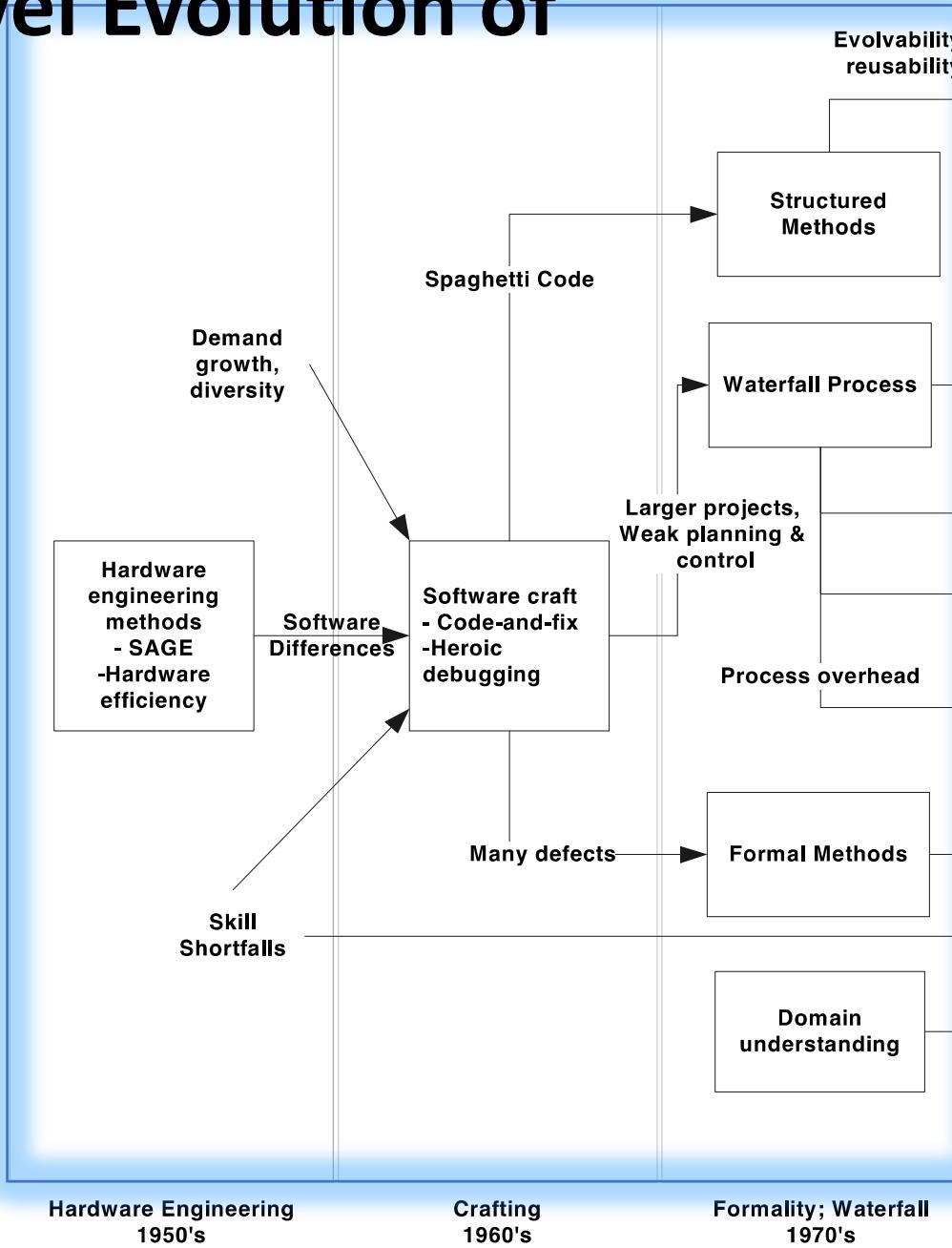
<https://agilemodeling.com/essays/introductiontoam.htm>

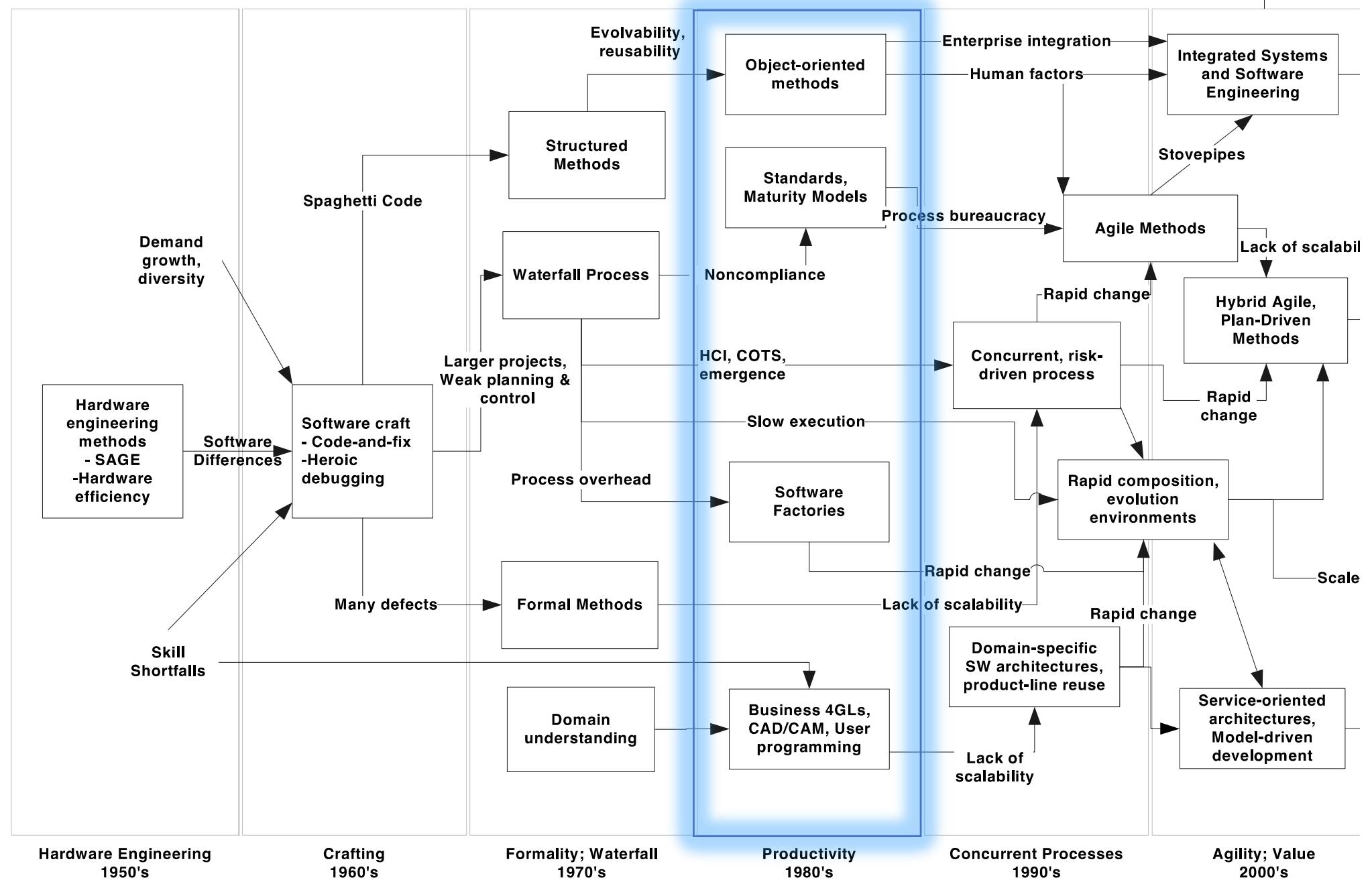
High-level Evolution of SE

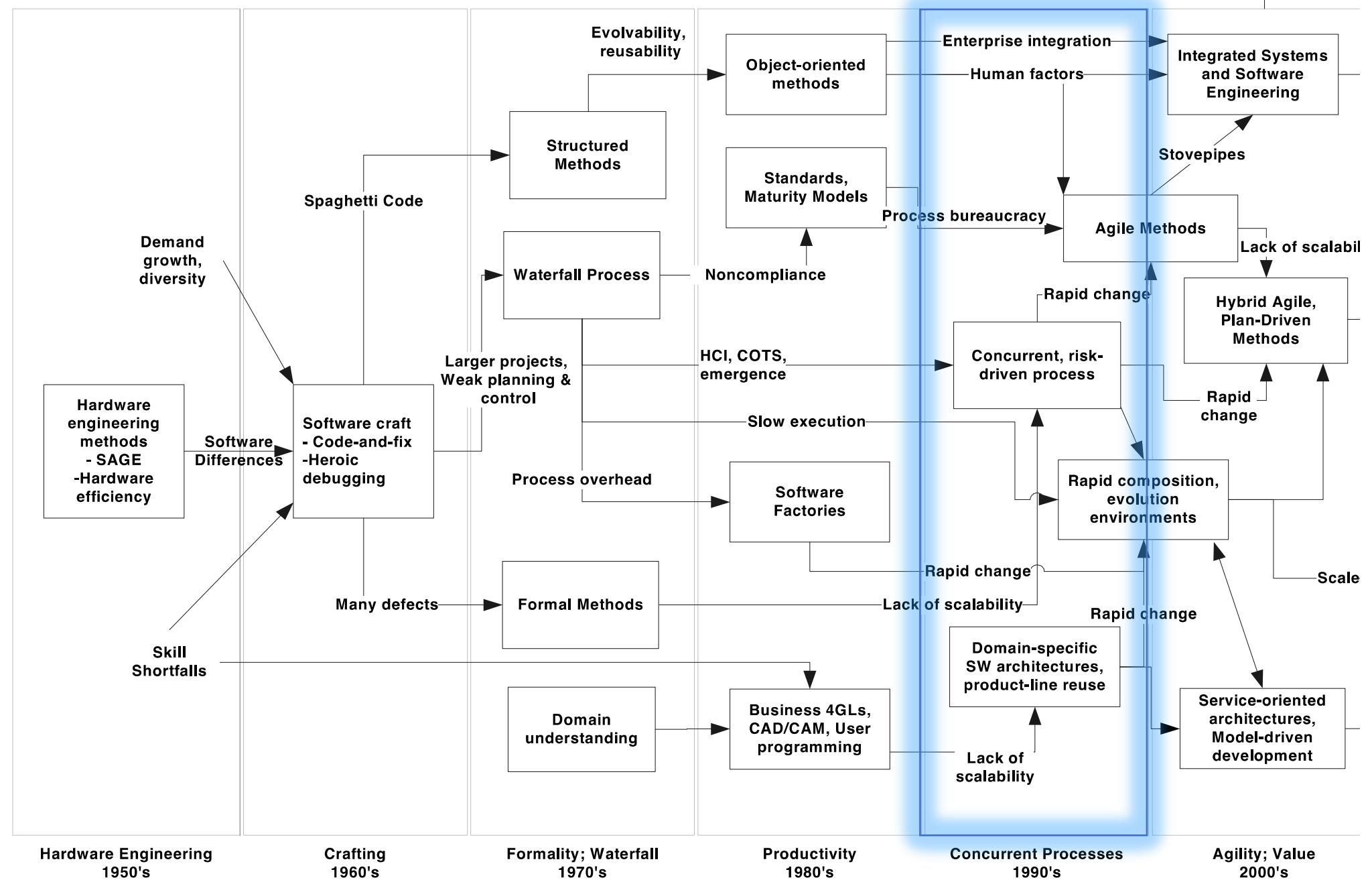


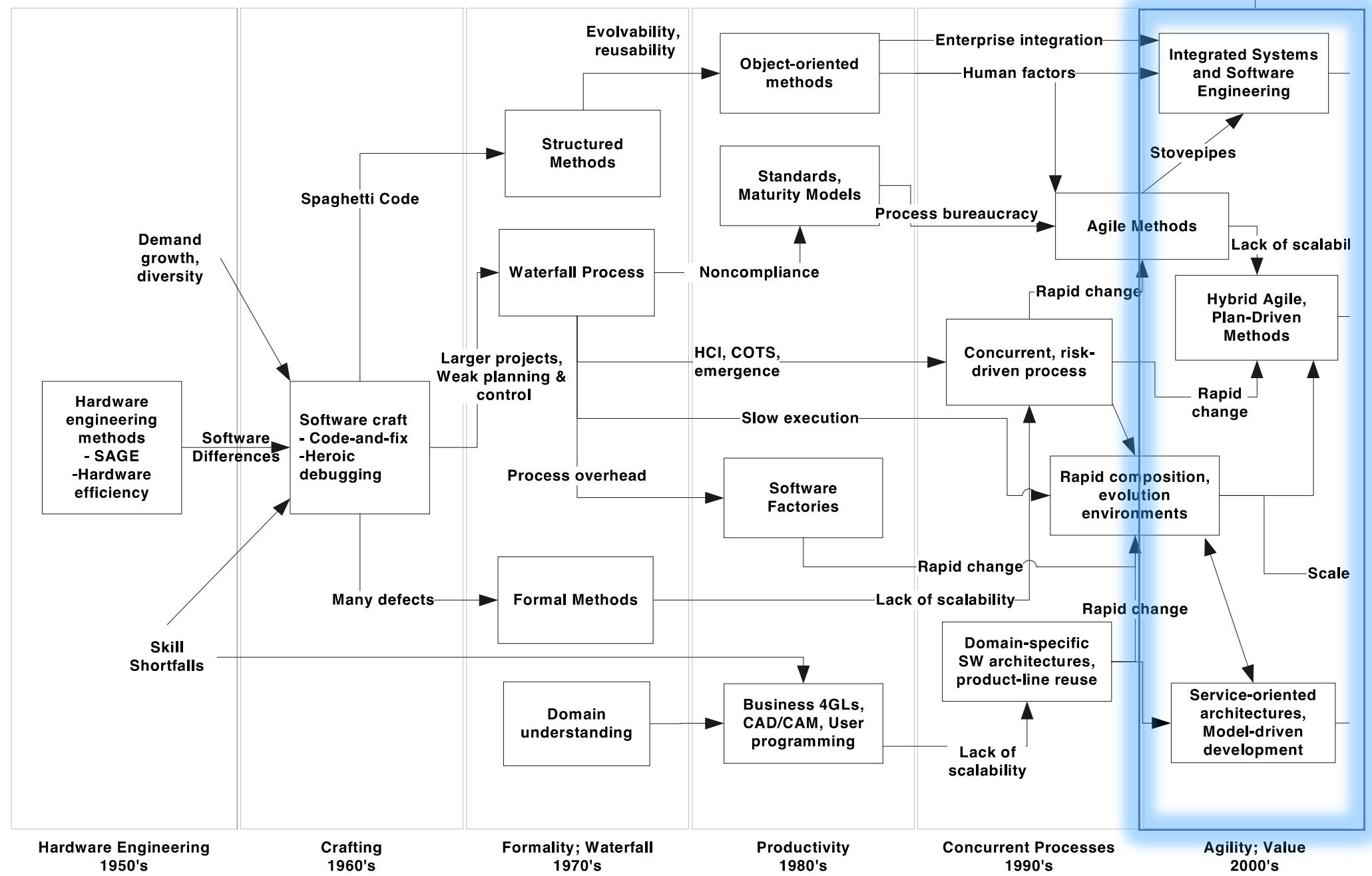
Boehm, B. (2006) "A view of 20th and 21st century software engineering," presented at the Proceeding of the 28th international conference on Software engineering - ICSE '06, New York, New York, USA, p. 12.

High-level Evolution of SE









Diverse sources of knowledge/learning....

- **Online Tutorials** - Freecode.com, YouTube
- **Online documentation** - tools, libraries eg Docker, GitHub
- **Podcasts** Engineering Culture by InfoQ
- **Newsletters** - InfoQ
- **Meetup groups** Agile Auckland, DevOps, Ministry of testing,
- **Blogs** - Medium, Freecode.com,
- **GitHub Repositories** - Open source projects,
- **Company websites** -Google, Xero, Microsoft, Facebook, Netflix, Basecamp
- **Published research**- ACM, IEEEExplore, SpringerLink, ScienceDirect
- **Online Q&A** -Stackoverflow
- **Guest speakers from industry**
- **Work in Industry or Open source projects**
- **Each other** – teaching and listening
- **GenAI services? More later...**
- **Your lecturer**

Takeaways from today...



Iterative and incremental ways of working established and valuable



Scrum is not prescriptive about many aspects of SE INTENTIONALLY



CI/CD focus on automating the integration and then deployment (release)
So small features can be released frequently



CI is a workflow for building, integrating and QA of shared code when
working in a team



Practice and experiment with the WoW and techniques and tools!!



Modern Agile and The Heart of Agile are modernizing and simplifying the values
behind the goal of Agile



#171678965



Questions and Comments....



Jim Buchan/Tony Clear S2 2024

CISE ENSE701

I has a question...

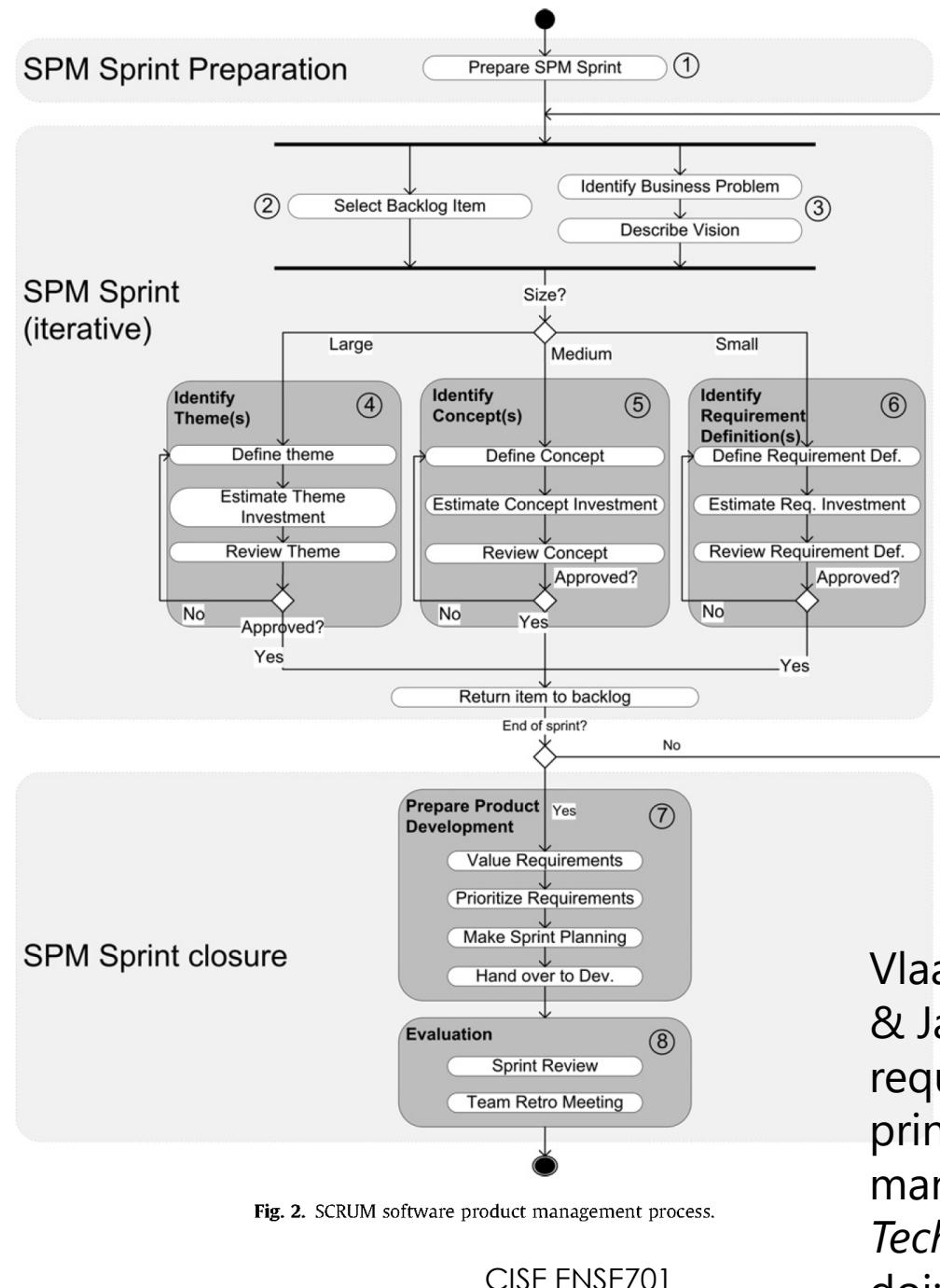


User Stories & Story Maps

Week 3

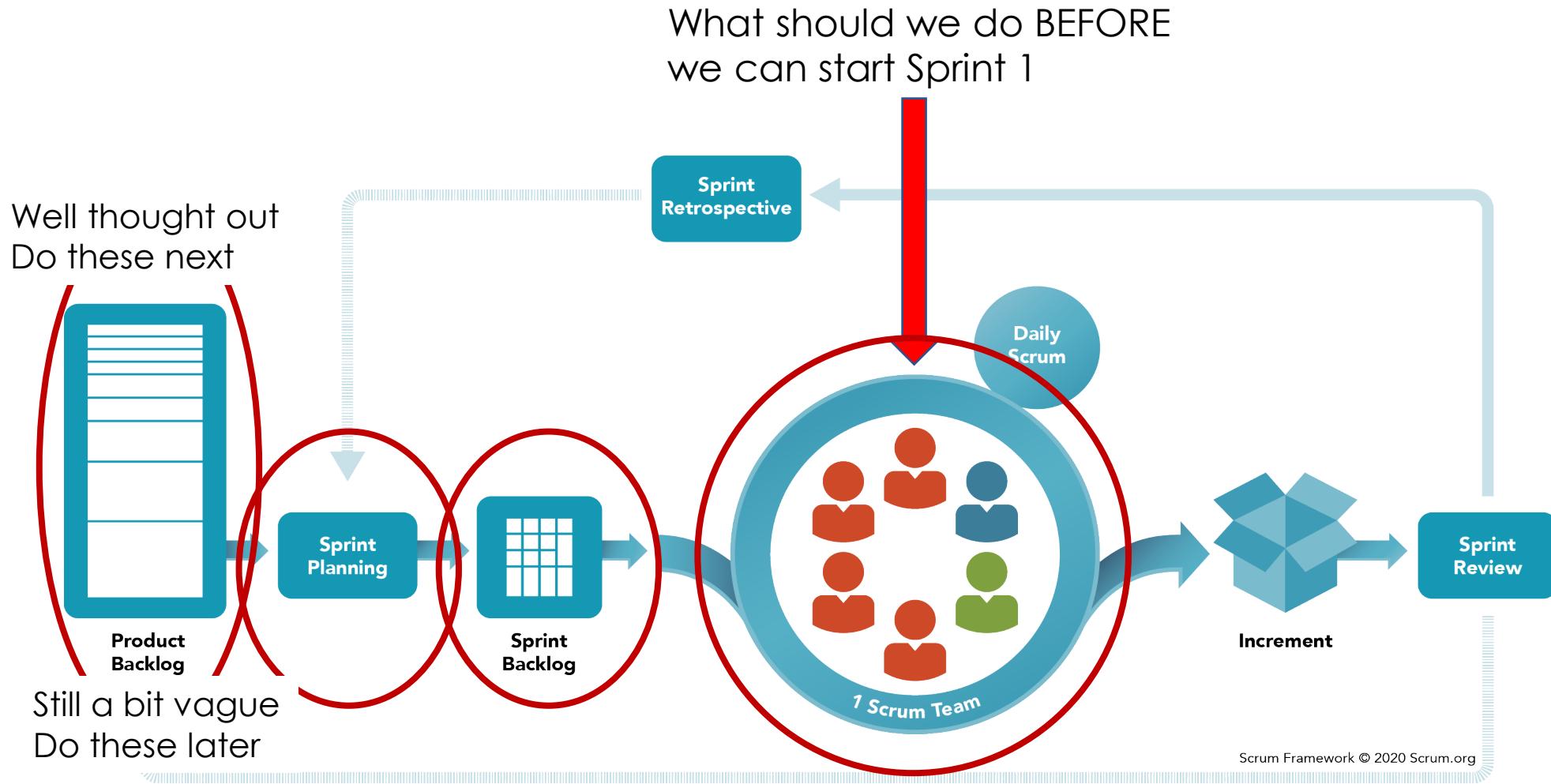


Software Product management Process



Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology*, 53(1), 58-70.
doi:10.1016/j.infsof.2010.08.004

Scrum/Iterative workflow



Introduction on how to write User Stories

There are many perspectives and purposes

<https://www.youtube.com/watch?v=Pn-QMvDTuEY>
FEMKE Design



Pre-sprint – User Requirements_v1

What is the “business” **problem or opportunity** (What needs to change? Why is this product worth doing?)

What is the **product goal**? (Outcome wanted). – MAKE FRONT OF MIND FOR TEAM

What do users want to be able to do to achieve the product goal?

Significant user type with characteristics that distinguish from other user types that may affect the design. NOT “User”

User stories

As a <???> I want to be able to <?????????> so that <??????????>

What criteria for the order?

Who puts them into order?

When are they put into order?

Why are they put into order?

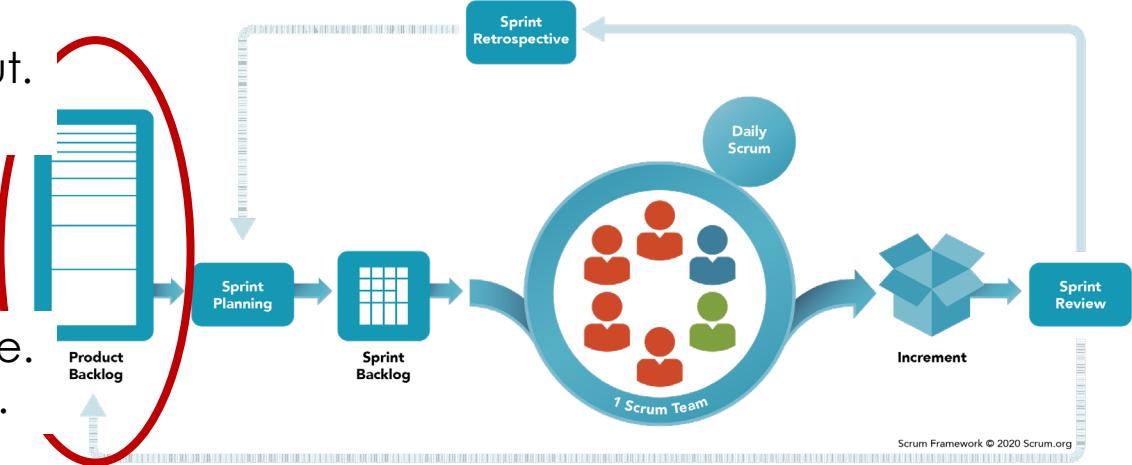
The new capability the user wants so they can reach smaller goal (outcome) In “business” language & NO solution details

What is the goal (desired outcome) of this new user capability?

Problem Space

Placeholders for conversations.

Still a bit vague.
Do these later.



Pre-sprint – User Requirements_v1 User Stories part 1

Significant user type with characteristics that distinguish from other user types that may affect the design. NOT "User"

The new capability the user wants so they can reach smaller goal (outcome) In "business" language & NO solution details

What is the goal (desired outcome) of this new user capability?

As a <???> I want to be able to <?????????> so that <??????????>

As the PO I want to have lots of articles in the evidence repository

As a practitioner, I want lots of evidence to be available so the evidence is convincing

I want some way to be able to keep adding articles with evidence for analysis to add evidence to the repository

I want people to be able to add new evidence so the repo gets bigger and leverage crowd sourcing

As a **researcher** I want to be able to **recommend articles to include in the evidence repository** so that **the evidence available keeps expanding**

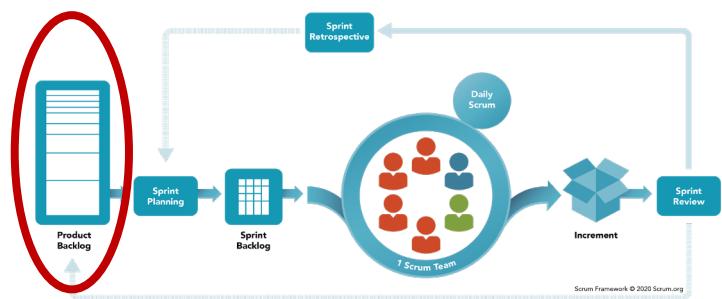
Placeholders for conversations.

We should check the article is about SE with evidence (relevance)

We should check the article is not already in the repo (avoid duplicates)

We should check the quality of the evidence in the article (quality)

The submitter should be informed/thanked if the article they submitted is accepted or not (and why not)



Discovering Product Goal, User Needs

Knowledge about the problem, user needs and possible solutions is distributed unevenly - at the start especially.

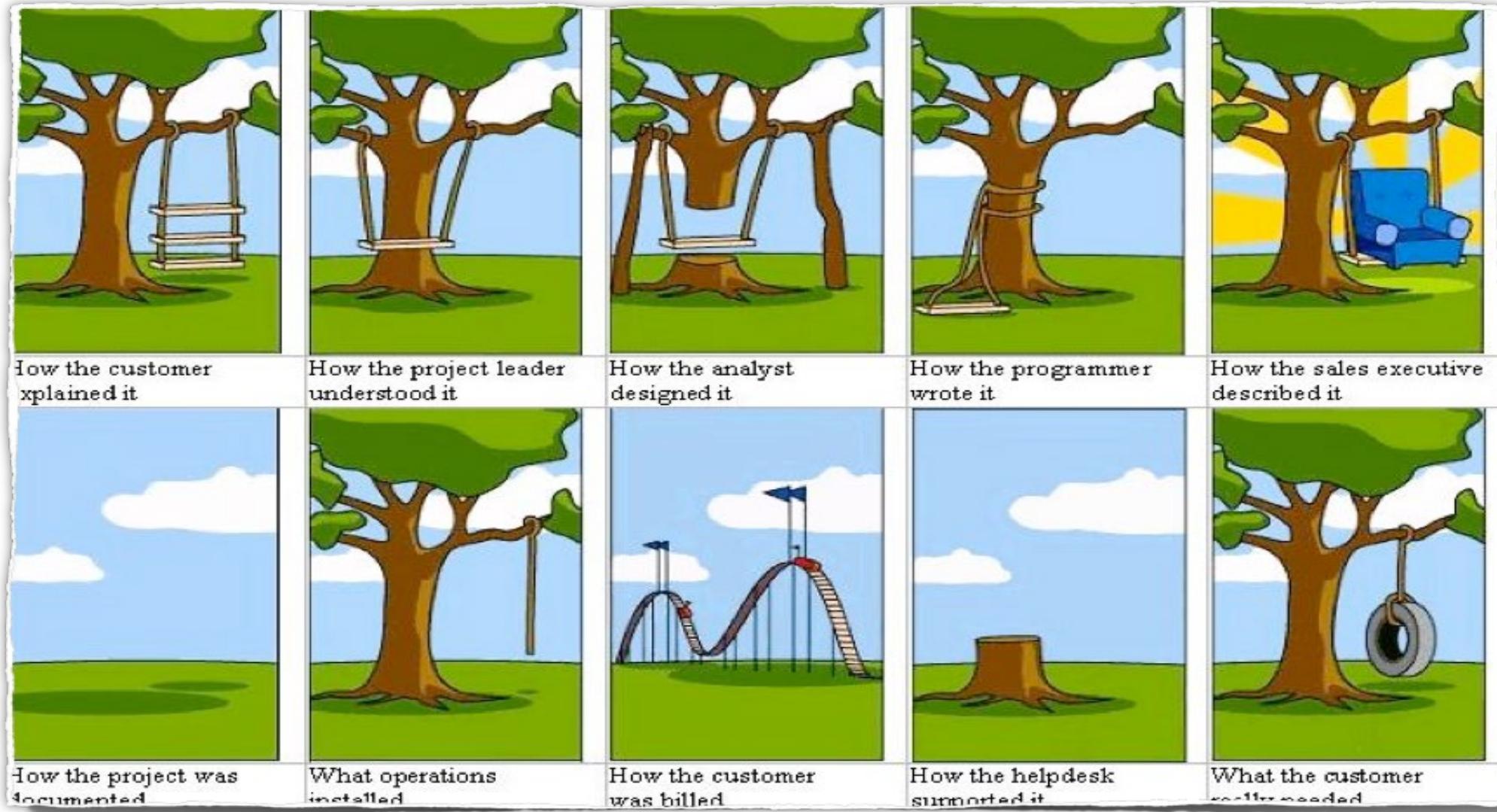
There are many perspectives on the problem and user needs

There are many ways to design a software product to address these needs

Different world-views and language



If you just imagine the user needs from your perspective!



Documenting User Needs/Requirements - User Stories



Physical small cards

Electronic cards/tickets/table

More detailed user stories with at least one acceptance criteria

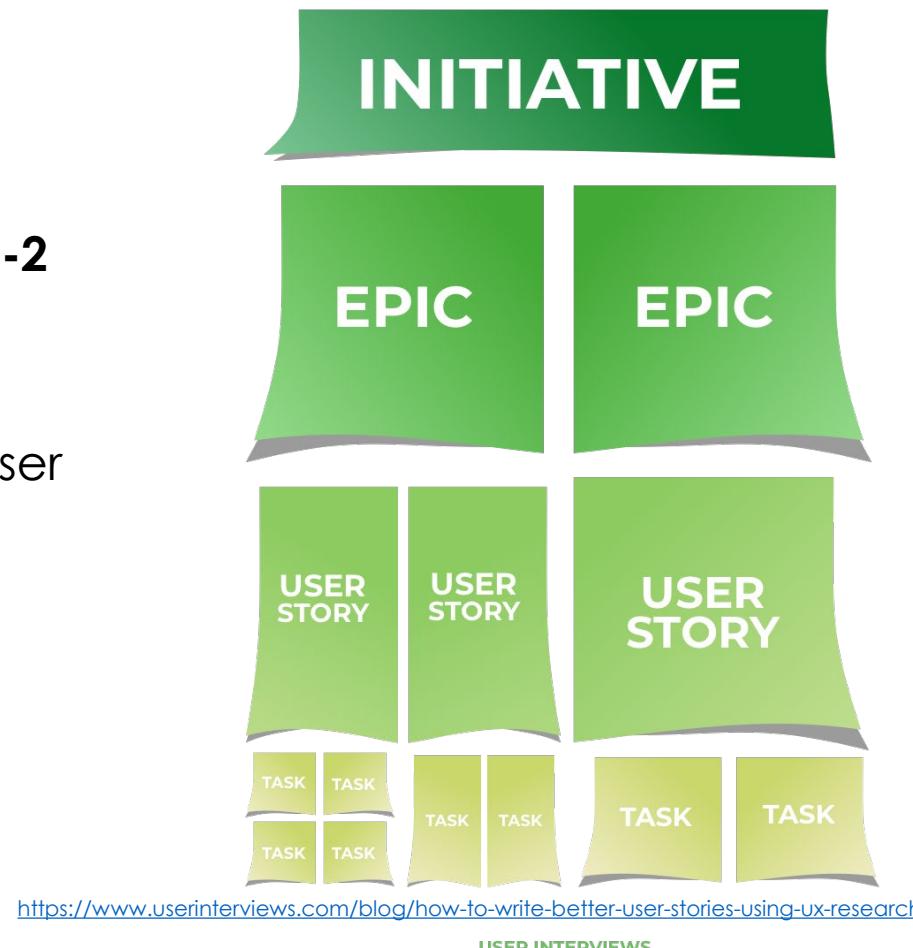
Keep breaking down until you expect that the user story will take **1-2 ideal days** and definitely **< 1 sprint**)

Work out **technical tasks** that need to be done on the next set of user stories to be worked on

Often tasks are associated with User Stories

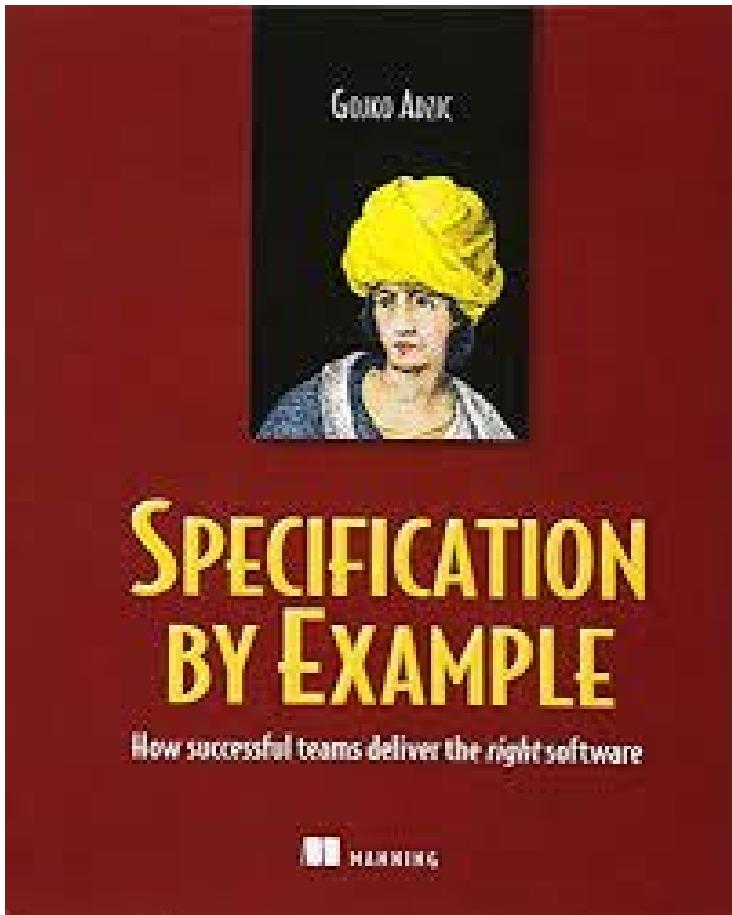
Avoid technical “user stories”

PUT IN LINK HERE



Specification by Example

We will come back to this



Order amount	VIP Member	Number of items in order	Free freight	Order Discount
<\$100	Y	≥ 10	n	y
$\geq \$100$	Y	< 10	y	n
<\$100	Y	≥ 10	n	y
$\geq \$100$	Y	< 10	y	n
<\$100	Y	≥ 10	n	y
$\geq \$100$	Y	< 10	y	n
<\$100	N	≥ 10	n	n
$= \$100$	N	< 10	n	n
<\$100	N	≥ 10	n	n
<\$100	N	< 10	n	n
$= \$100$	N	≥ 10	n	n
<\$100	N	< 10	n	n

Discovering User Stories (user requirements)

Big upfront effort

plan-driven control based on high-confidence (certain), **long-term** predictions

Iterative and incremental effort

Frequent opportunity for changes based on empiricism and short learning loops and **short-term** certainty and long-term uncertainty.

User Story Workshops – product stakeholders, PO, BA, Dev, Tester etc
Stakeholders write them and group (and agree on order or in/out)?

Interviewing users or the PO – get placeholders for future conversations about needs, changes to the current situation, and some detail for user needs that are of most value to be worked on first

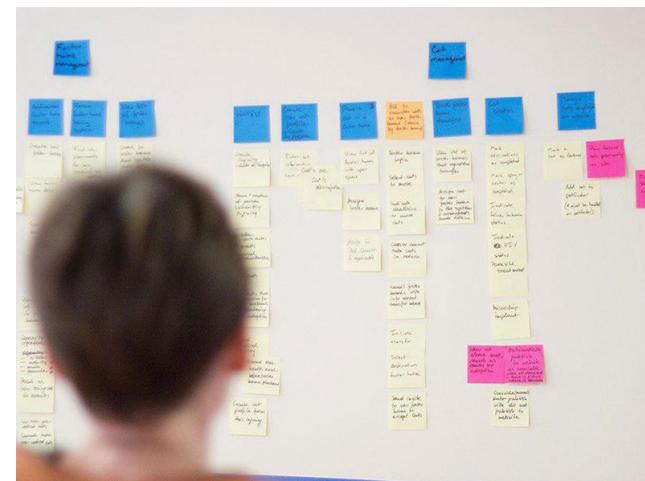
Observation, surveys, impact mapping, customer journey mapping.....

Discovering user Stories – Mapping techniques

Mapping = **visualize** a lot of **information** showing **relationships** and some **structure**
Helps to make sense of something – and is also an information radiator

You can explore these yourself

Customer Journey Maps
Impact Mapping
Value stream mapping



“A successful customer journey map will give you real insight into what your customers want and any parts of your product, brand or process that aren’t delivering.”

David Weaver (co-founder of Vintage Cash Cow)



A customer journey map is a **visual** representation of the **important steps** a customer goes through to achieve a **goal with your company**.

You can get a sense of your customers'

- motivations
- needs
- pain points.

What are the characteristics of a good user story and how to write them so they meet these criteria

Independent of other user stories – minimize inter-team/inter team member dependencies that need a coordination effort

Not too much detail – this will be obtained close to the time of development of it – and stored in team members minds and maybe some acceptance criteria and some design diagrams and end up in the code functionality.

Written in the language of the user – non-technical, avoiding technical solution, can be checked/discussed by PO

Can be “tested” against acceptance criteria – otherwise we are never sure if solution meets the user story needs

We will come back to this after Sprint 1 – after we have tried writing a few user stories

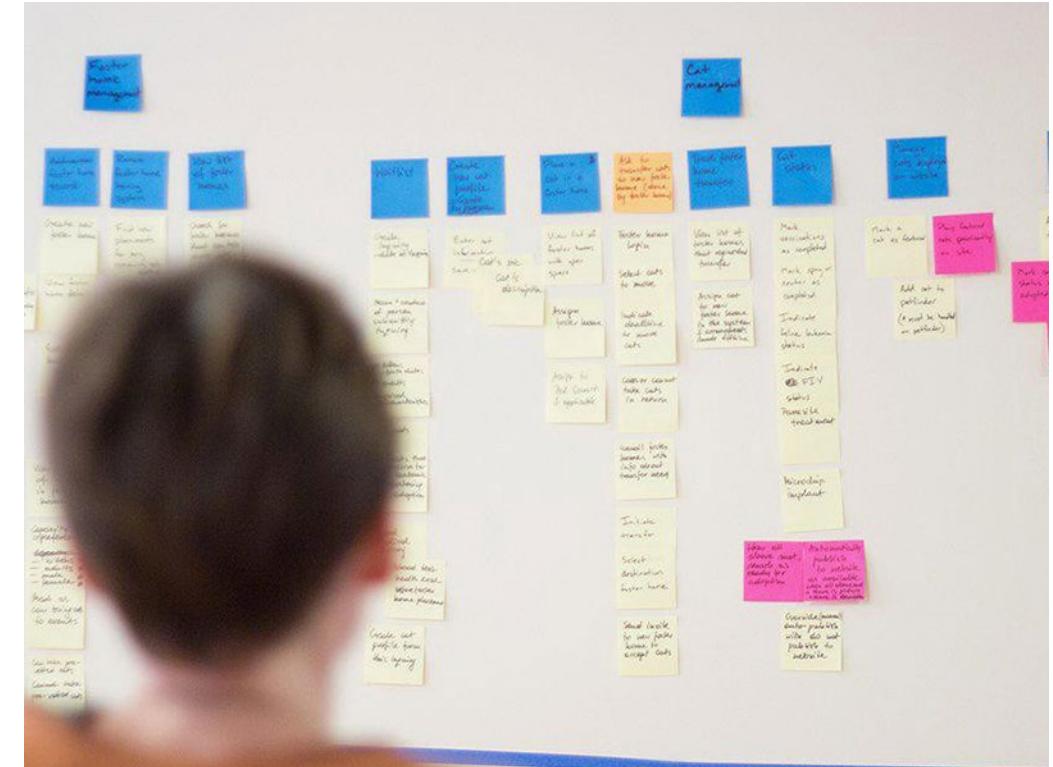
Story Maps are the new Product Backlog

Story mapping is a way of structuring product features (as User Stories) to show a model of the entire product being developed

It avoids some shortcomings of a Product Backlog which just shows a prioritised list of features still be be worked on

Story Maps can be useful for:

- planning delivery product increments
- plan sprint product increments
- Support Up front product design
- Support work prioritisation
- Share and develop understanding



They provide a context for features to understand relationships between features, user types, and their purpose

Story Maps address a need...

The Problems with Flat Backlogs

Traditional Product Backlogs are flat; a prioritized list.

Great for answering “**what do we do next?**”

Not so great for:

- Collaborative building & inspection
- Seeing how everything fits together
- Balancing a view of user-valued features with the need for iteration-size stories
- Planning coherent value-based releases

Risk of becoming a
Feature Factory

Using a story map to slice out a delivery plan or sprint plan



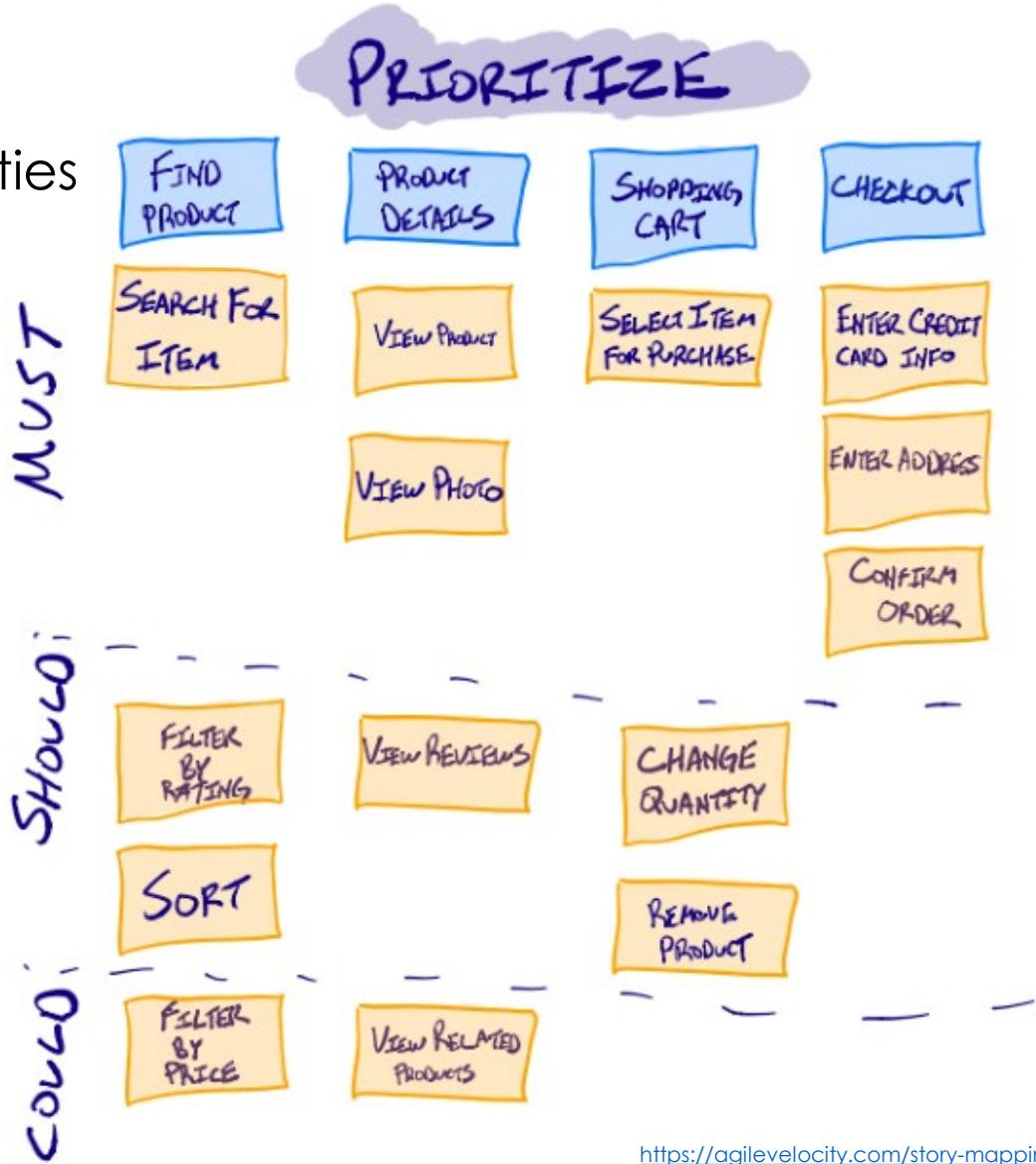
Jeff Patton & Associates, jeff@jpattonassociates.com, [twitter@jeffpatton](https://twitter.com/jeffpatton)



80

The anatomy of a Story Map

Columns relate to user activities

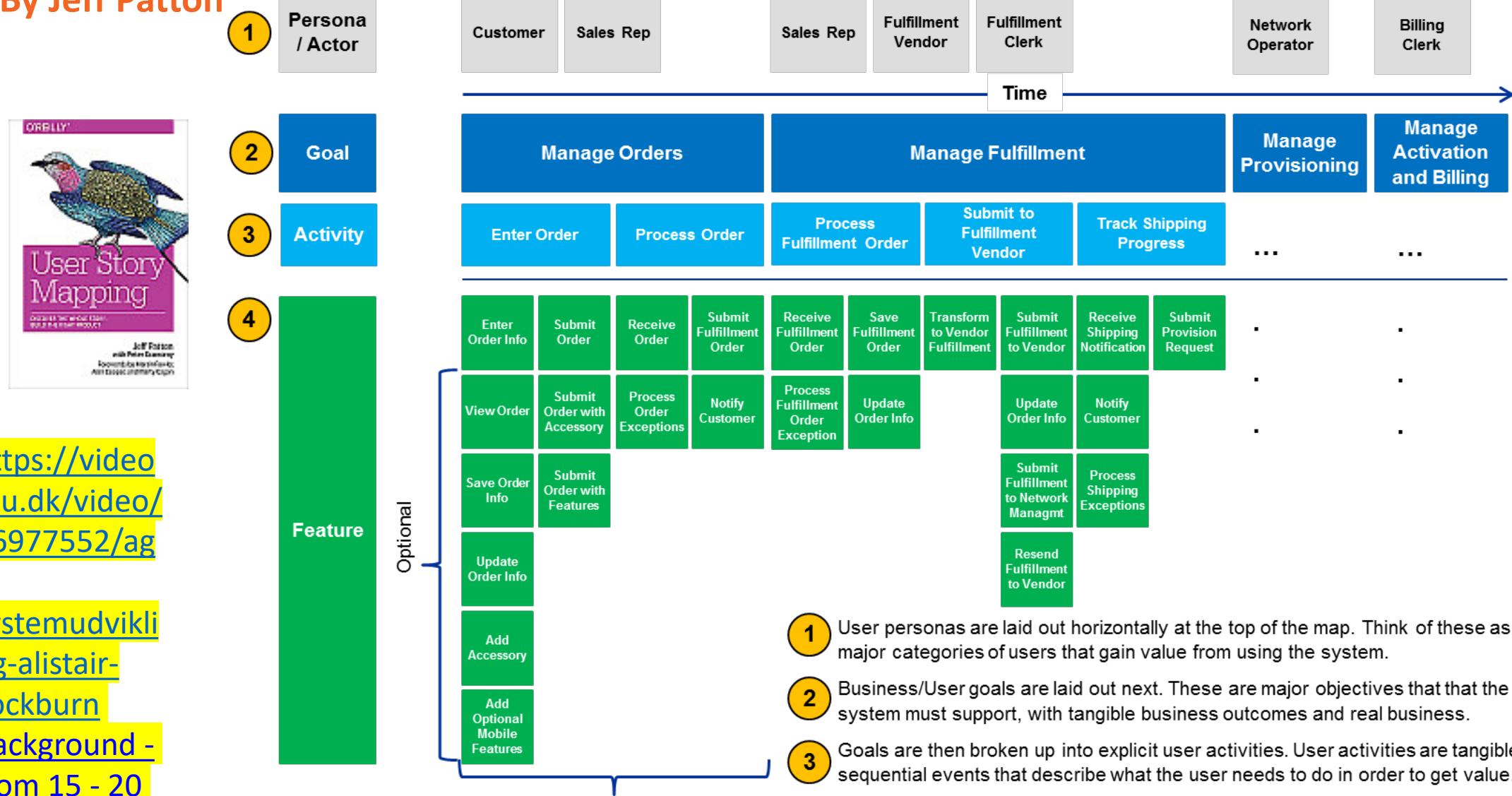


Each column has the system features related to the user activity for that column

Can create other horizontal slices
Of features to represent the scope of delivery cycles or MVP or sprint cycles

User Story Mapping

By Jeff Patton



These are user types

Example

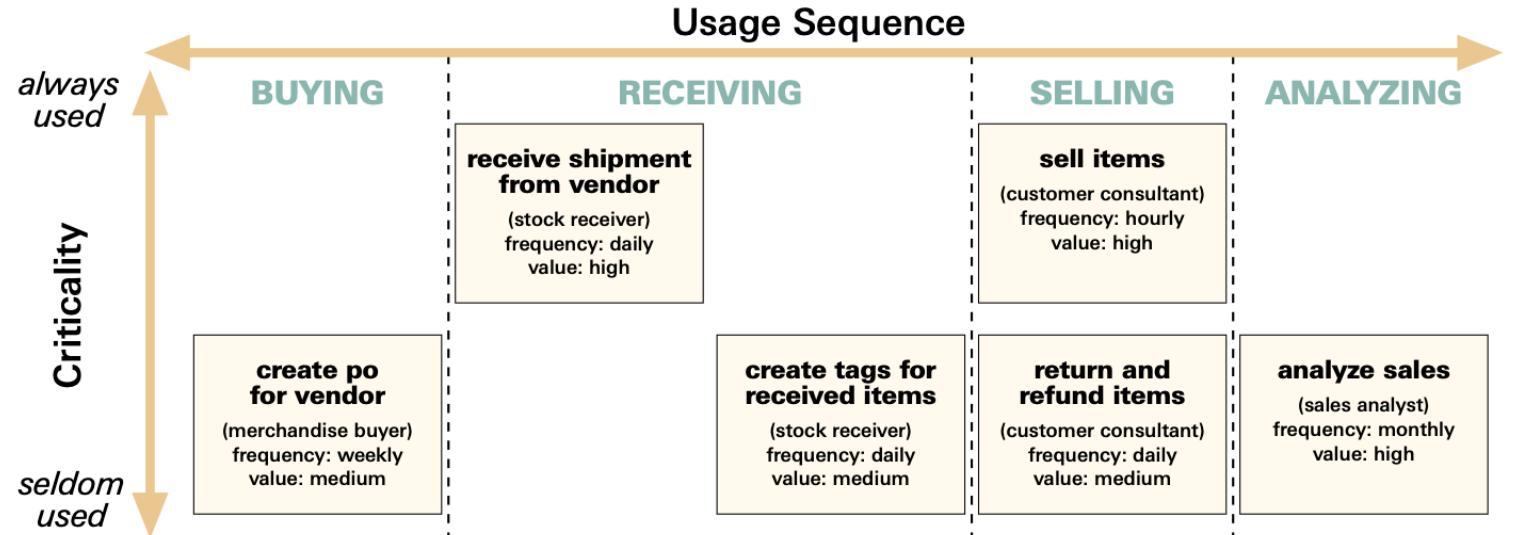
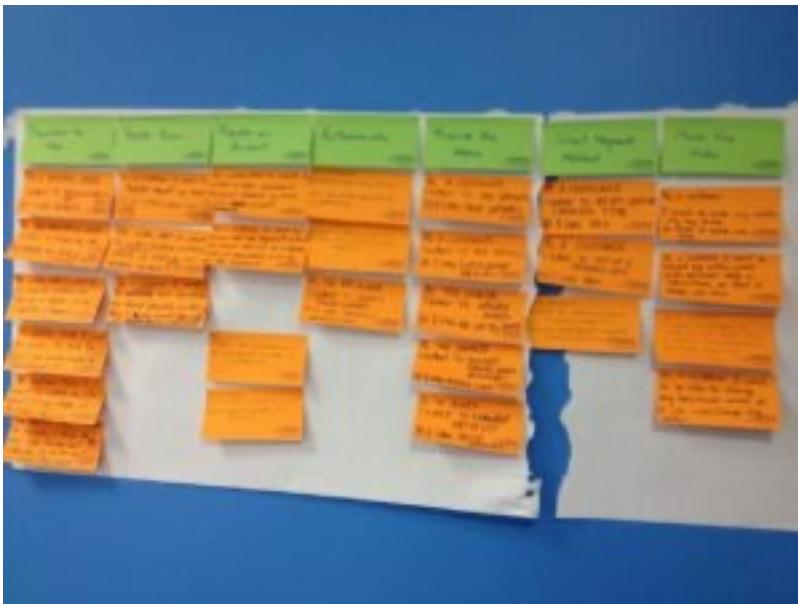


Figure 5: The model is vertically divided into business processes.

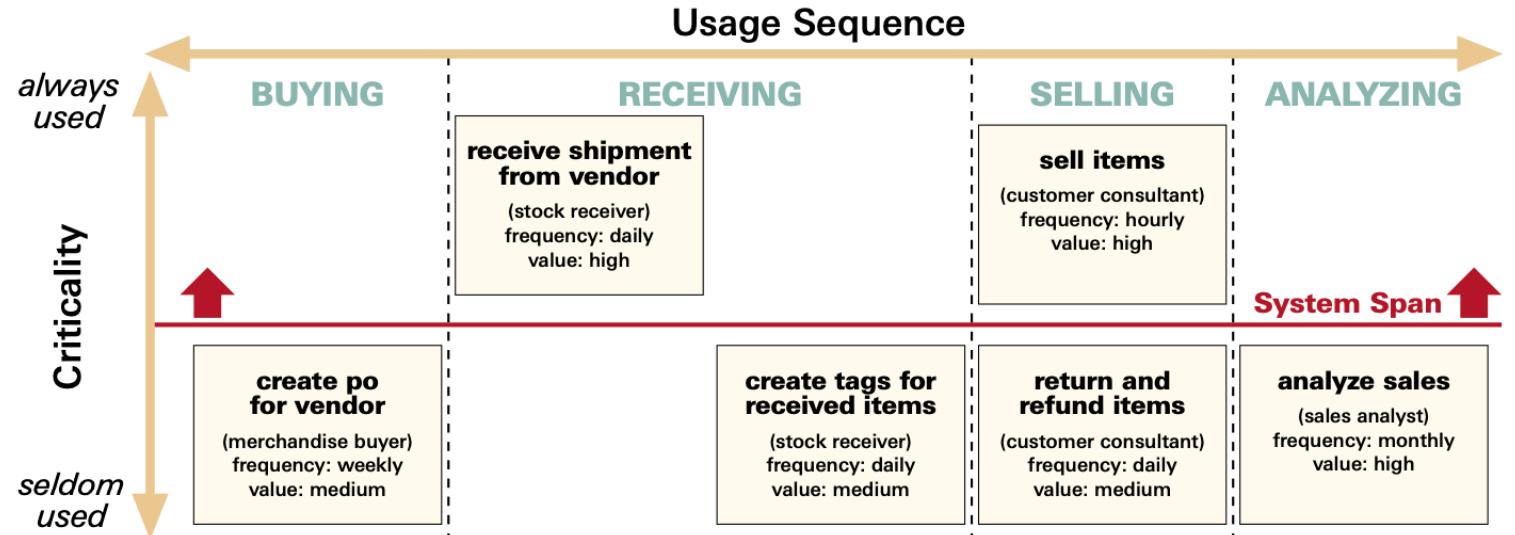
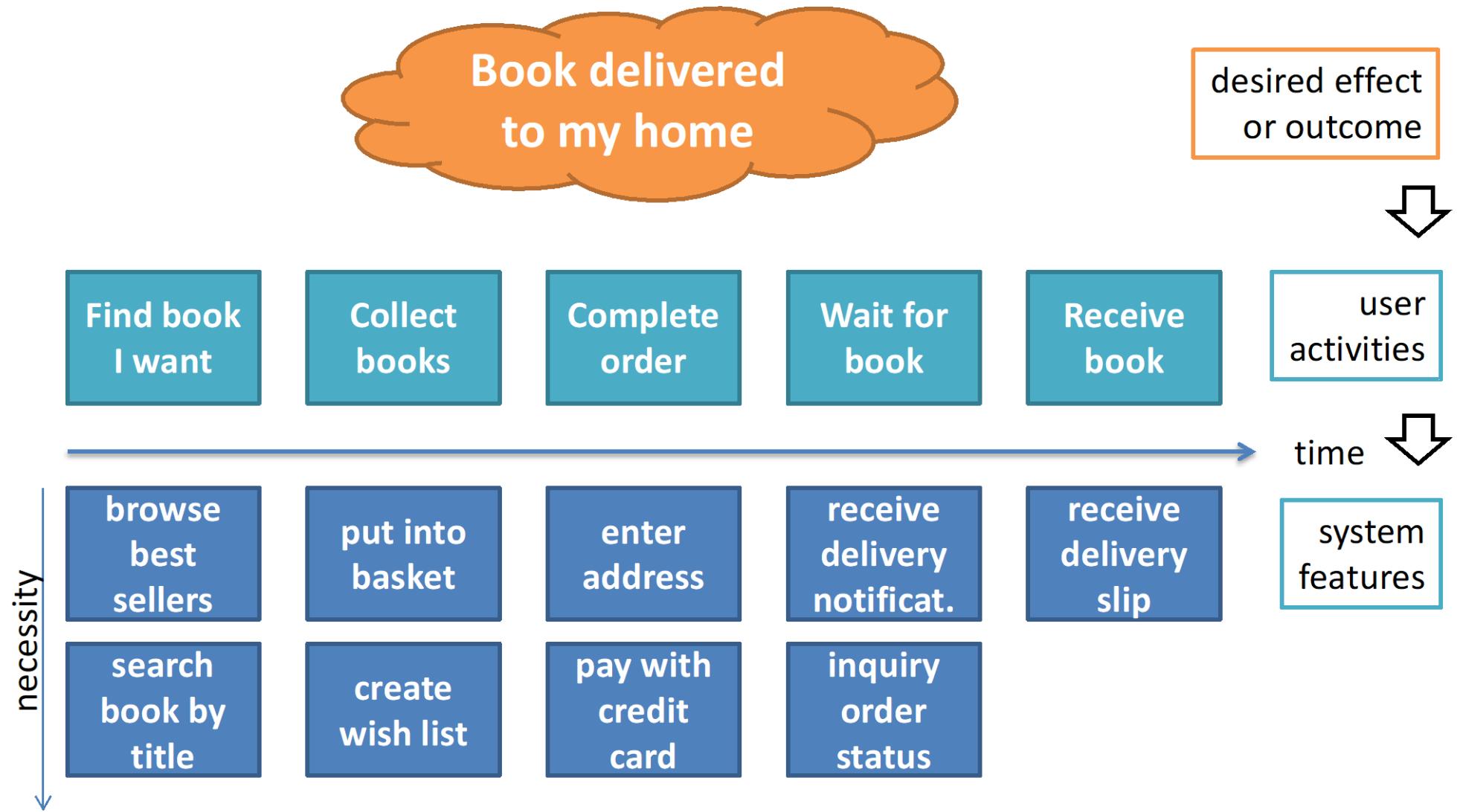


Figure 6: The first system span represents the smallest set of features necessary to be minimally useful in a business context.

Example



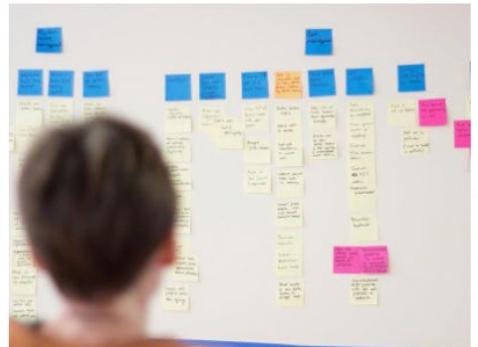
https://www.scrumalliance.org/system/slides/118/original/christianhossa_specificationbyexamplewithgherkin.pdf?1349824954

Good resource

The latest articles about user story mapping

February 19, 2019 by Gergo Matyas

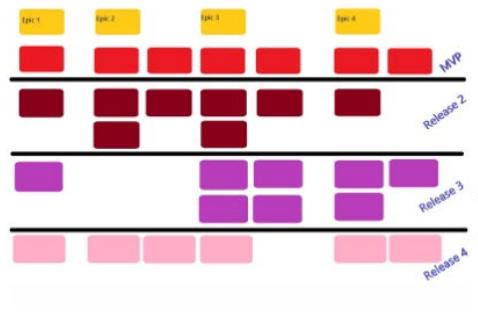
StoriesOnBoard staff pays a lot of attention to collect the latest, most inspiring articles from the internet. Read further to discover fresh new materials and open the previously published collection of user story mapping content.



User Story Mapping as delivery planning

by Paddy Corry

"User Story Mapping adds a narrative element to a larger unit of value by taking a vertical backlog such as Jira or VSTS, and adding a horizontal axis across the top to represent the entire narrative of the larger unit of value. Let's call it the 'unit of value' a feature in this post. If you're not familiar with the technique, it is straight-forward, but also deceptively sophisticated." [Read more...](#)



User Story Mapping and probabilistic forecasting

by Willem-Jan Ageling

"Probabilistic Forecasting summarizes what is known about future events. You will not work with single-valued forecasts but you assign a probability to a number of different outcomes. Probabilistic Forecasting based on historical data is an alternative proposed by #NoEstimates. You don't make assumptions, you use actual data. This makes it a powerful way of



User Story Mapping Games

by Kateřina Mňuková

"User story mapping is a great technique, if everybody knows what to do, how and what to expect (more in my previous articles about story mapping). If you are worried that the workshop itself could be messy due to lack of knowledge about the user story mapping, start it with quick game. It takes maximum 30 min and it explains the purpose of story mapping more than enough and plus it's fun!" [Read more...](#)



7 Reasons to try user story mapping

by Kate Hopkins

"Typically, participants tell the story of a particular user's experience, documenting the various steps and options with sticky notes as they go. Story maps ensure that the focus stays on the user and what he or she is trying to accomplish, rather than on features or development requirements. They're an incredibly versatile tool that helps with design, prioritization, and communication." [Read more...](#)



“User Story Mapping is Big Design Upfront!”

by Willem-Jan Ageling

"Some people I know dislike User Story Mapping. They find it is Big Design Upfront (BDUF). You'd basically do upfront planning of multiple Sprints. And even if you would not do this, if you would allow new insights to be added, you waste a lot of time on the exercise that probably turns out to be largely incorrect when time passes." [Read more...](#)

<https://storiesonboard.com/blog/user-story-mapping-articles>

Resources by the guy who had the idea in 2005!

[http://www.jpattonassociates.com/wp-content/uploads/2015/01/how you slice it.pdf](http://www.jpattonassociates.com/wp-content/uploads/2015/01/how_you_slice_it.pdf)

<https://www.jpattonassociates.com/the-new-backlog/>

Article headings

FLAT BACKLOGS DON'T WORK FOR ME

THE FLAT BACKLOG IS POOR EXPLANATION OF WHAT A SYSTEM DOES

FOR A NEW SYSTEM, THE FLAT BACKLOG IS DISMAL AT HELPING ME DETERMINE IF I'VE IDENTIFIED ALL THE STORIES

RELEASE PLANNING WITH A FLAT BACKLOG IS DIFFICULT

BUILD A STORY MAP

KEEP YOUR EPICS – BUT STOP CALLING THEM THAT BECAUSE IT BOTHERS ME

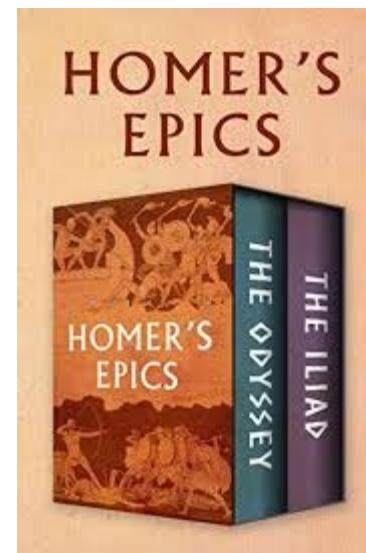
WALK YOUR MAP TO TEST IT – TO GET THE BIG PICTURE

YOUR SOFTWARE HAS A BACKBONE AND A SKELETON – AND YOUR MAP SHOWS IT

KEEP YOUR MAP DISPLAYED TO COMMUNICATE THE BIG PICTURE

A DIFFERENT BACKBONE MAY BE IN ORDER FOR ADDING FEATURES TO AN EXISTING PRODUCT

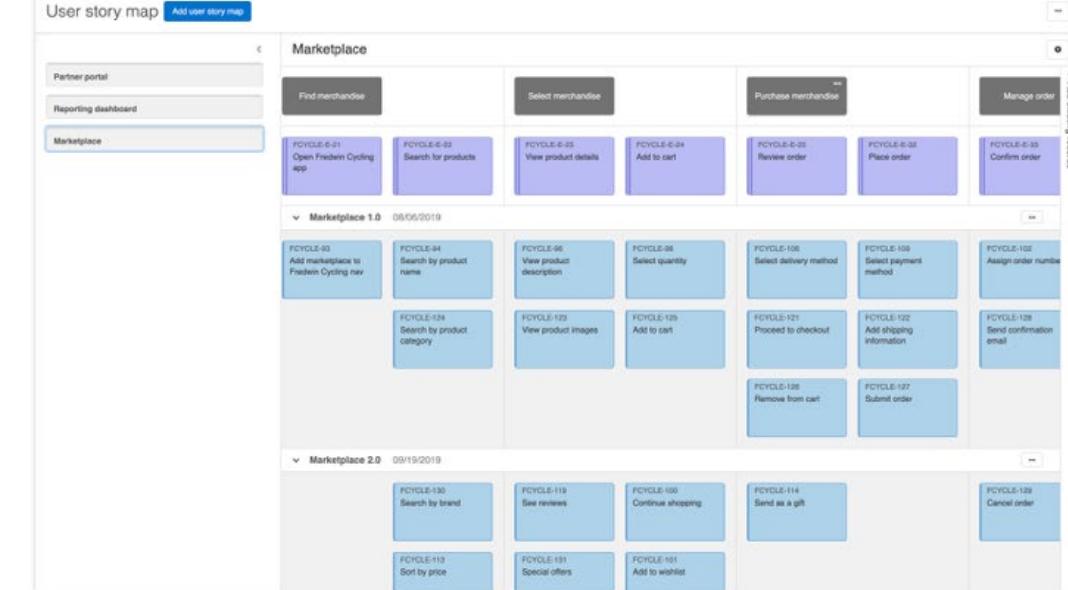
IT'S A PATTERN – NOT AN INNOVATION



<https://media.simplecast.com/episodes/audio/14099/TCPJeffPatton.mp3>

A podcast by Jeff Patton on Talking Code

Tools ...



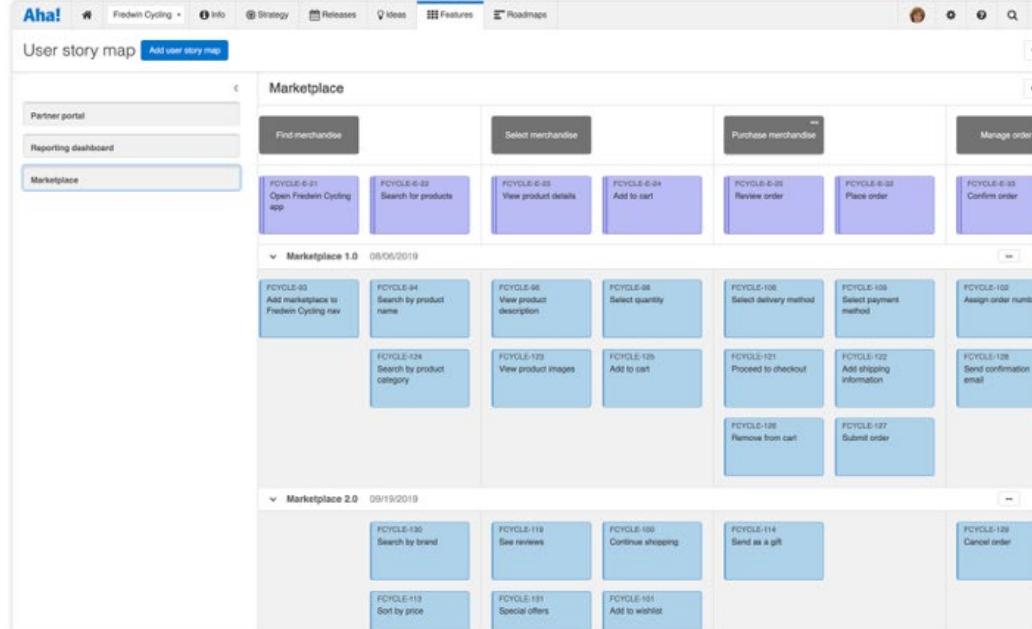
The screenshot shows the Aha! software interface. At the top, there's a navigation bar with links for PRODUCT, USE CASES, SERVICES, PRICING, RESOURCES, and COMPANY. On the right side of the header, there are links for Blog, Support, Careers, Log in, Free trial, and Join demo. Below the header, the main content area has a title "Just Launched! — New User Story Mapping Tool in Aha!" dated July 11, 2019. The author is Claire George. There are social sharing buttons for Twitter, Facebook, and LinkedIn. The main content displays a user story map for a Marketplace feature, showing various user stories and their dependencies. The map includes sections for Marketplace 1.0 and Marketplace 2.0, each with multiple user stories represented by cards.

July 11, 2019

Just Launched! — New User Story Mapping Tool in Aha!

by Claire George 

[Tweet](#) [Share 94](#) [Share](#)



Product and marketing teams love user story maps. How do we know? Because it is the second most popular item in our [ideas portal](#). Many of you told us you wanted a better way to align your roadmap with the user and buyer journey. We agreed. And after quite a bit of journey mapping on our end, we are excited to introduce this important functionality so you can do just that.

 You can now create user story maps in Aha! to align your roadmap with your customer's journey. 



CATEGORIES

Aha! Updates

New and fresh

Company Building

Grow, grow, grow

Product Management

Build products that matter

Marketing

Create breakthrough plans

Enterprise Transformation

All lasting companies change

Career Advice

Be happy

Remote Work

Be great anywhere

Earlier Approaches from FDD and Jeff De Luca!

<http://www.featuredrivendev.com/node/630>

User Story Success Criteria

*acceptance criteria, Acceptance

A list of “rules” or criteria or tests or behaviours that should be met if the story is to be successful

Behaviour Driven Development (BDD)

Acceptance Test Driven Development (ATDD)

As a purchaser I want to be able to pay online by credit card for convenience

Possible Success criteria?

Common template

Initial situation

Event

New situation (output)

Given <??????>

When <??????>

Then <??????>

<https://resources.scrumalliance.org/Article/need-know-acceptance-criteria>

Exercise - write down 2 user stories about SPEED

https://padlet.com/tony_clear/speed-user-stories-8paqocdeblzf3ro4

In pairs select the best 2

In teams of 4 select the best 1

A user story is written from the perspective of what the end user wants in the language of the end user – WHY?

In the Agile way of working the focus is on creating user value –user stories capture this user value and keeps the focus on it during all development

They focus on understanding the business value from the user's perspective (as explained by the PO) so that the **right thing** is built.

Delays the temptation for developers to jump to the solution before understanding the problem and the benefits of the solution.

Helps user (PO) clarify what they want

Makes it easy for the user to verify the requirement and when this will be done and everyone can share this understanding if a common language is used

In a nutshell

Skills, mind set needed

Writing user stories – what info, how much detail, what “size”

Defer detail

Asking questions to get detail

Write acceptance tests (conditions of satisfaction)

Split user stories

Forecast effort for short frequent planning

Write personas

Understand user value

Create and Use story maps

Manage changes

Short feedback cycles
Dev team learns from doing
PO learns from seeing
-> requirements emerge/change

Defer detail until the dev team need it

Act as a reminder to get the detail just in time

Split stories

Write conditions of satisfaction

Act as a reminder that dev team promises to have conversations and ask questions

User stories

Point to requirements

Conversations
Diagrams
Prototypes
User interfaces

Be sensitive to your user task's “altitude”



Too abstract

Activity or “Kite level”

Longer term goals often with no precise ending. I'll perform several functional tasks in the context of an activity

Think about user experience at this level

Functional or “Sea level”

I'd reasonably expect to complete this in a single sitting



Sub-Functional or “Fish level”

Small tasks that by themselves don't mean much. I'll do several of these before I reach a functional level goal



Too detailed



* from Cockburn's Writing Effective Use Cases

Three pieces of information are useful to the development team *knowing these can influence the developers' design of the solution*

As a...

What is the primary actor, user role or **persona** that the requirement is for?

I want...

What is the new action/feature/goal the user wants to be able to happen?

So that...

What is the user's main reason for wanting the new requirement
– the expected benefit?

This is just a thinking tool – you do not always have to use this format

Examples

As a user, I want
to order prints of
my photos

As a photographer, I
want to see a preview
of the photo book
before I order

As a user, I
want to cancel
my order

As a frequent flyer, I
want to rebook a past
trip so that I save
time booking trips I
take often

Software requirements are a communication challenge

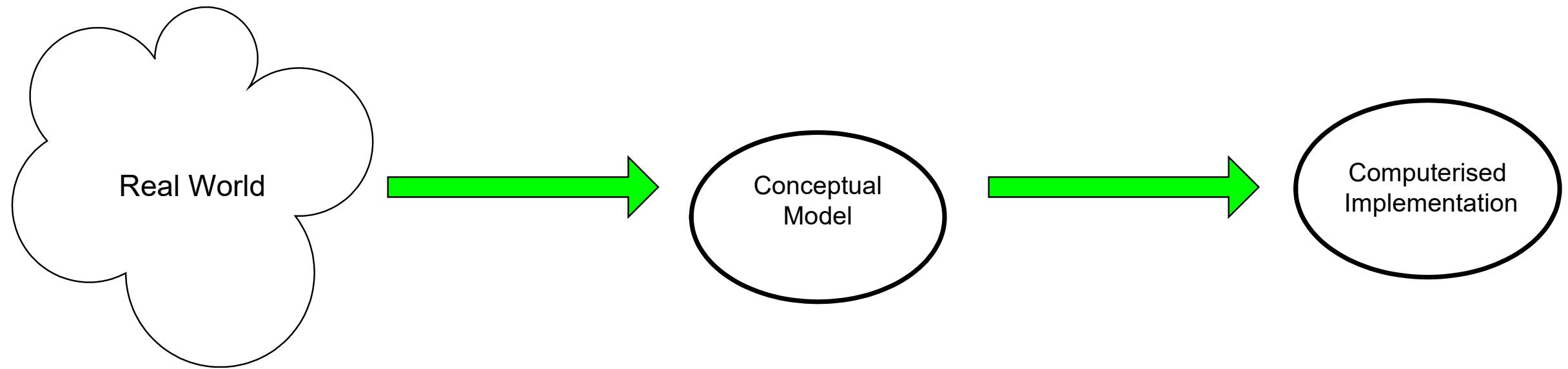
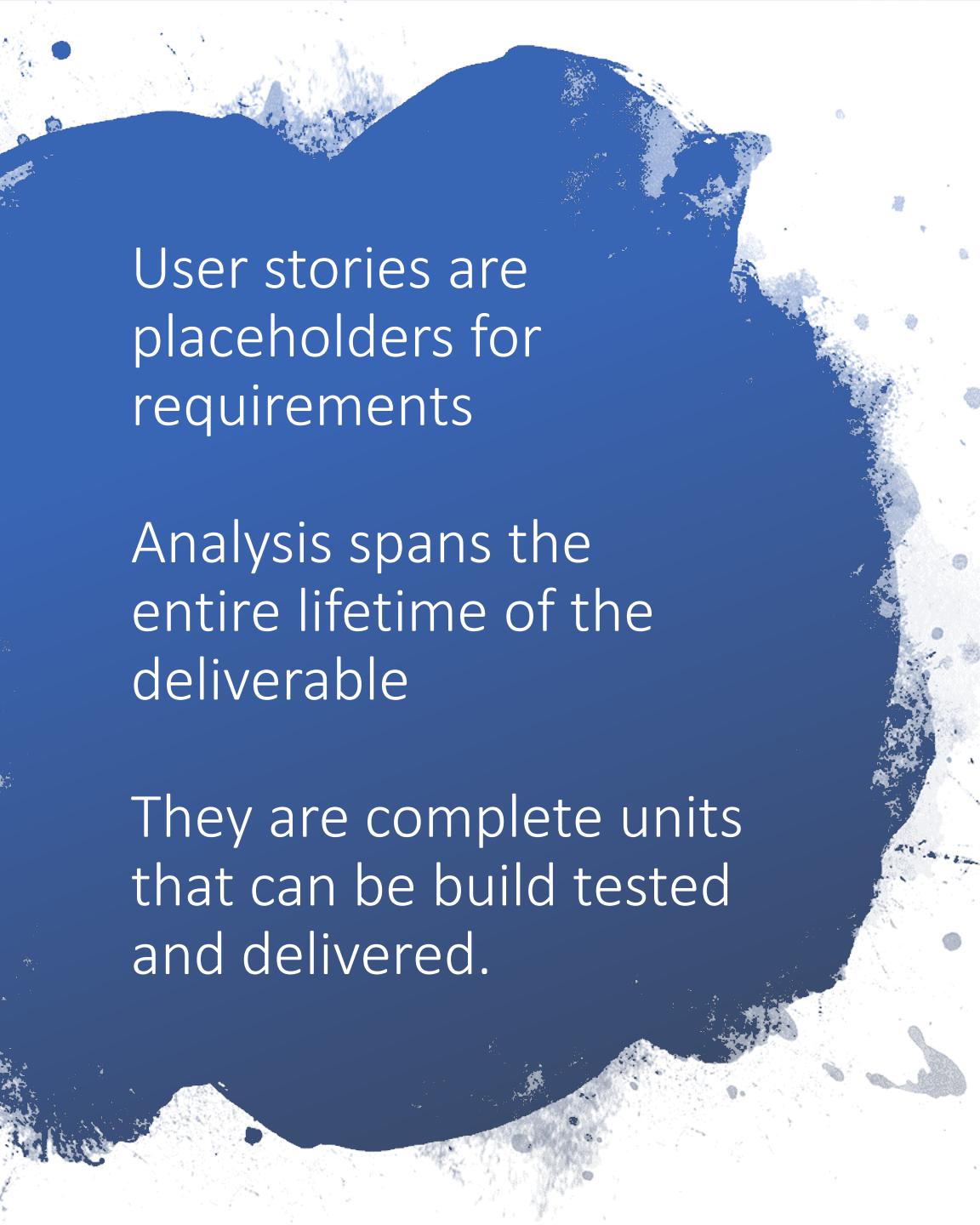


Figure 1: Design as a Mapping Process



User stories are placeholders for requirements

Analysis spans the entire lifetime of the deliverable

They are complete units that can be build tested and delivered.

User stories are worth getting right ...

- ✓ Set the boundaries for each work item – the unit of analysis/discussion
- ✓ Are the unit for planning sprints (*estimation* for sprint backlog)
- ✓ Are the unit for *keeping track* of what is still to be done overall and in what order (ordered product Backlog)
- ✓ Are the unit for *testing* (acceptance tests, DoD)
- ✓ Are the unit for monitoring sprint *progress* (on the story board, burndown charts)
- ✓ Are the basis for *change management* (add/subtract/modify->re-order)
- ✓ Are the unit for *division of labour*

DEVELOPER Task vs story

Create a database to store job seekers CVs in

As a user I want to be able to save my CV to a database

As a **job seeker** I want **my CV to be easily available to as many prospective employers as possible anytime they want** so that I **maximise my chances of being offered a job**

<https://www.kununu.com/de/jobs?q=business%20analyst>

Break into smaller user stories

As a **job seeker** I want **my CV to be easily available to as many prospective employers as possible anytime they want** so that I **maximise my chances of being offered a job**

WHY?

Planning, estimation
work splitting (independent)
testing

HOW?

Workflow Steps

Business rule variation eg sort

Major effort extracted - eg pay by credit card – most effort in first one

Simple vs complex – pick simple version first – American Express is harder to implement

Think about Developer tasks to create features for these

Why?

Maybe to estimate

To have the right people involved – technical specialty?

To plan development in the sprint

Create database

Hook up to node.js

Create front end in react.js

etc

How much of this should I document?



Self-organizing
teams are central
to the Agile way
of working....

So do we need a team leader?

The team agrees on the..

- acceptance criteria (condition of satisfaction)
- proposed solution approach and
- estimate of effort

to complete each story

Creating user stories

Anyone (the team including PO and other stakeholders) can write user stories – user story workshop?

PO is responsible for making sure a PB exists and the order

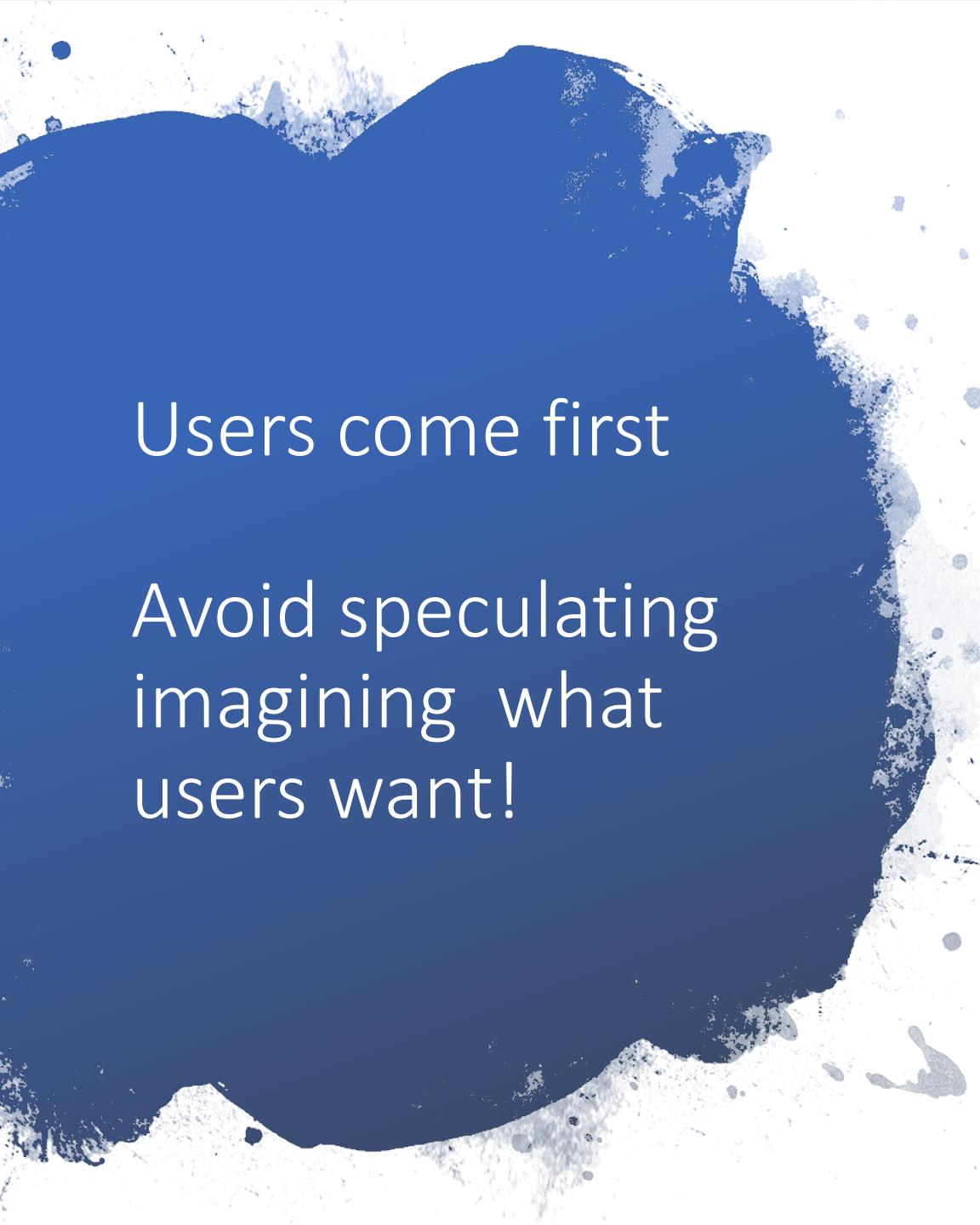
Everyone should be involved in discussions about the user stories

They should represent value/benefit that the user wants

I have also seen developer story cards, non-functional cards, bug cards,

Acceptance criteria should be written as part of the user story
(this will be done when.....)

The initial user stories and acceptance criteria DO NOT need to be detailed or perfect or complete



Users come first

Avoid speculating
imagining what
users want!

If you don't know who the users and customers are and why they would want to use the product, then you should **not** write any user stories.

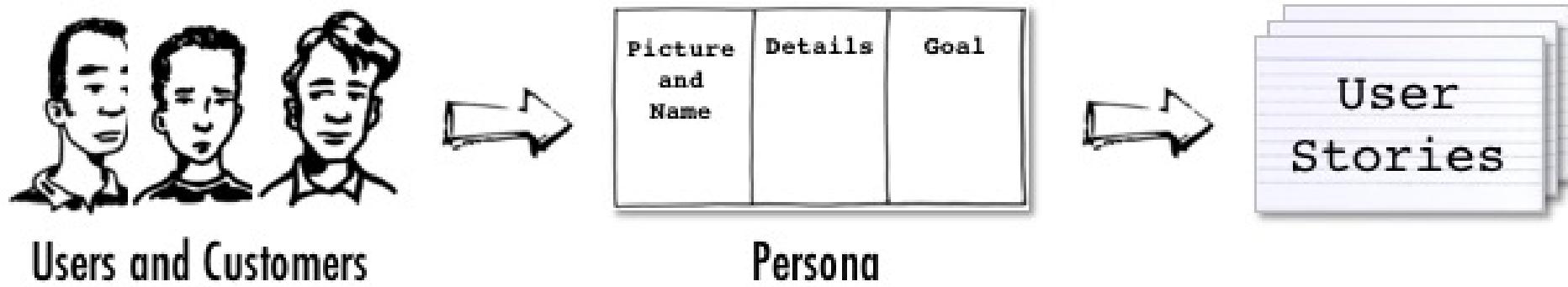
Carry out the necessary user research first, for example, by observing and interviewing users.

Create stories collaboratively



<https://www.romanpichler.com/blog/10-tips-writing-good-user-stories/>

Personas lead to the right stories



Ask yourself what functionality the product should provide to meet the goals of the personas

<https://www.romanpichler.com/blog/personas-epics-user-stories/>

How is detail added to a user story and when?

When – when it is near the top of the PB

Often teams had a reasonably clear idea of what would be in the next 2-3 sprints ahead (order and estimate).

Detail is added by
breaking stories down

Adding acceptance tests (acceptance criteria, success criteria, conditions of satisfaction)

Where are the details...

As a registered user I want to be able to cancel my order so that I can change my mind

- Does the user get a full or partial refund?
 - Is the refund to user's credit card or is it site credit?
- How far ahead must the order be cancelled?
 - Is that the same for all orders?
 - For all customers? Different requirements by market?
- Is a confirmation provided to the user?
 - On-screen? Email? Letter?

Conditions of satisfaction as details (Acceptance criteria Success criteria, Acceptance Tests)

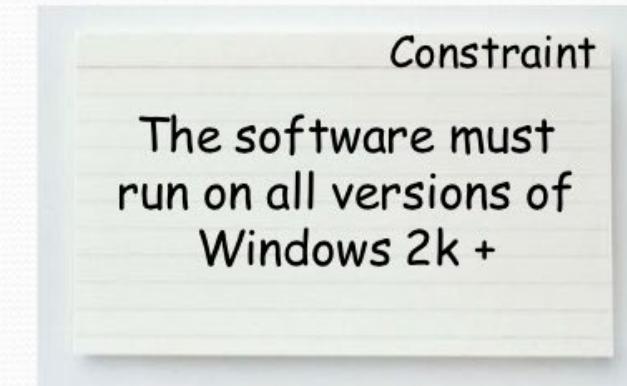
- The product owner's conditions of satisfaction can be added to a story
- These are essentially tests / acceptance criteria

As a user, I
want to cancel
my order

- Verify that a VIP member can cancel the same day without a fee
- Verify that a non-VIP member is charged 10% for a same-day cancellation
- Verify that any refunds are in site credits only
- Verify that an email confirmation is sent
- Verify that the factory is notified of any cancellation

Non-functional requirements

- For some requirements it's inefficient to be captured in user stories because they apply to all of them
 - Example 1: "The system must support peak usage of up to 50 concurrent users"
 - Example 2: "Do not make it hard to i18n the software later if needed"
- Write **constraint cards** and keep them **visible**



User stories about UI

- Keep stories with UI elements until a later sprint when stories shift from being new functionality to being modification of existing functionality
- Example: “A user can select dates from a date widget on the search screen”

Breaking user stories up

1. Workflow Steps

Identify specific steps that a user takes to accomplish a specific workflow, and then implement the workflow in incremental stages.

As a utility, I want to update and publish pricing programs to my customer.

...I can publish pricing programs to the customer's in-home display.

...I can send a message to the customer's web portal.

...I can publish the pricing table to a customer's smart thermostat.

Breaking user stories up

2. Business Rule Variations

At first glance, some stories seem fairly simple. However, sometimes the business rules are more complex or extensive than the first glance revealed. In this case, it might be useful to break the story into several stories to handle the business rule complexity.

As a utility I can sort customers by different ... sort by ZIP code.
demographics.
... sort by home demographics.
... sort by energy consumption.

Breaking user stories up

4. Simple/Complex

When the team is discussing a story and the story seems to be getting larger and larger ("What about x? Have you considered y?"), stop and ask, "What's the simplest version that can possibly work?" Capture that simple version as its own story, and then break out all the variations and complexities into their own stories.

*As a user, I basically want a fixed price, but I also want
to be notified of critical-peak pricing events.*

*...respond to the time and the duration of the critical-
peak pricing event.*

...respond to emergency events.

Definition of Ready – meets quality criteria (ready to be put on the PB)

The below principles of good agile stories come from the post [How To Write Meaningful Agile User Stories](#) by Isaac Sacolick

- Key stakeholders must achieve a shared understanding of the deliverable
- Stories should convey the opportunity, issue, need or value that it will deliver
- The story title should be short and convey the deliverable without reading the details
- Good stories deliver an atomic increment in business
- Stories need to be completed in a Sprint
- Good stories have sufficient acceptance criteria
- The team should be able to estimate the story

INVEST – a quick checklist to check the usefulness (quality) of user stories

The INVEST checklist comes from [INVEST in Good Stories](#) by Bill Wake. It's an acronym with six important quality characteristics for user stories:

Independent — Can the story stand alone by itself ?

Negotiable — Can this story be changed or removed without impact to everything else?

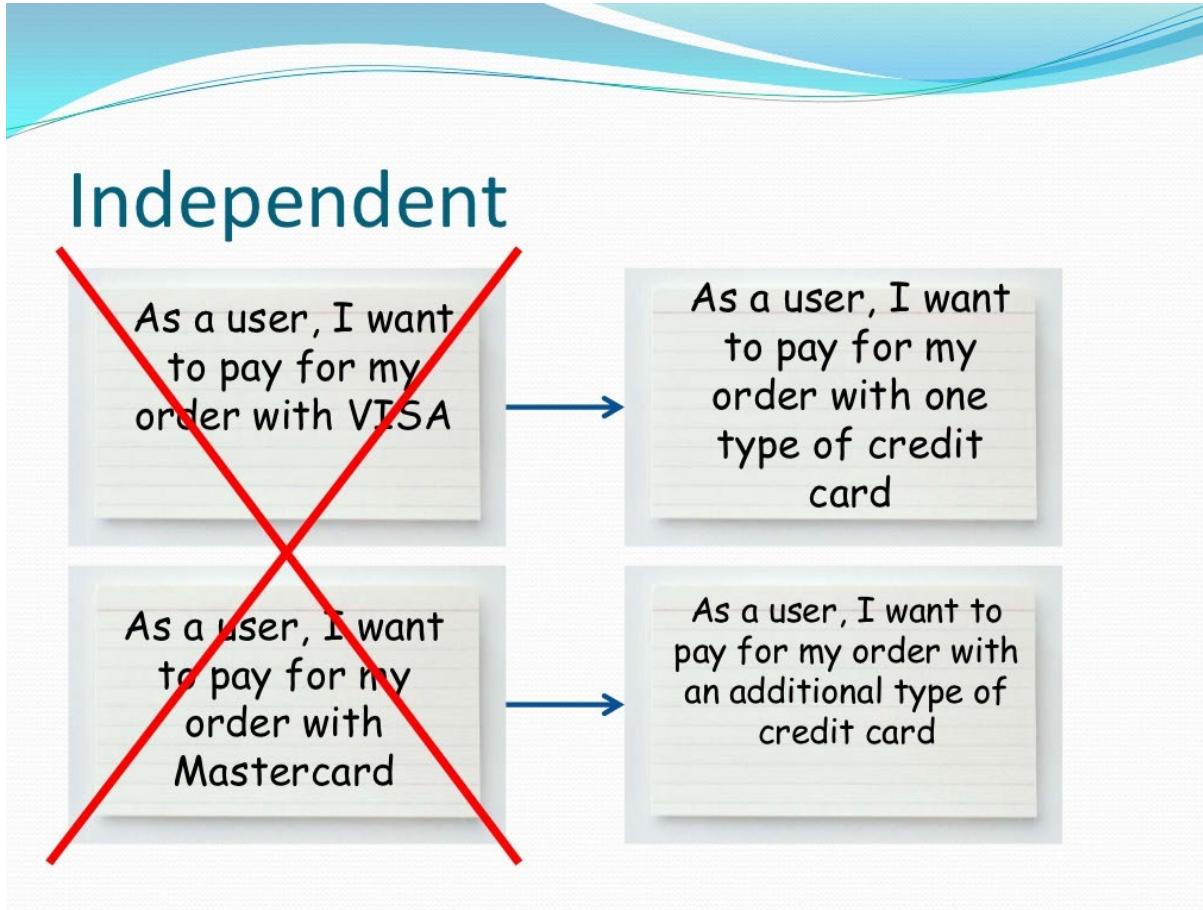
Valuable — Does this story have value to the end user?

Estimable — Can you estimate the size of the story?

Small — Is it small enough?

Testable — Can this story be tested and verified?

INVEST



Negotiable – not a contract!

As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express.
Consider Discover



As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express. Consider Discover. On purchases over £100, ask for card ID number from back of card. The system can tell what type of card it is from the 1st two digits of the card number. The system can store a card number for future use. Collect the expiration month and date of the card



Estimable

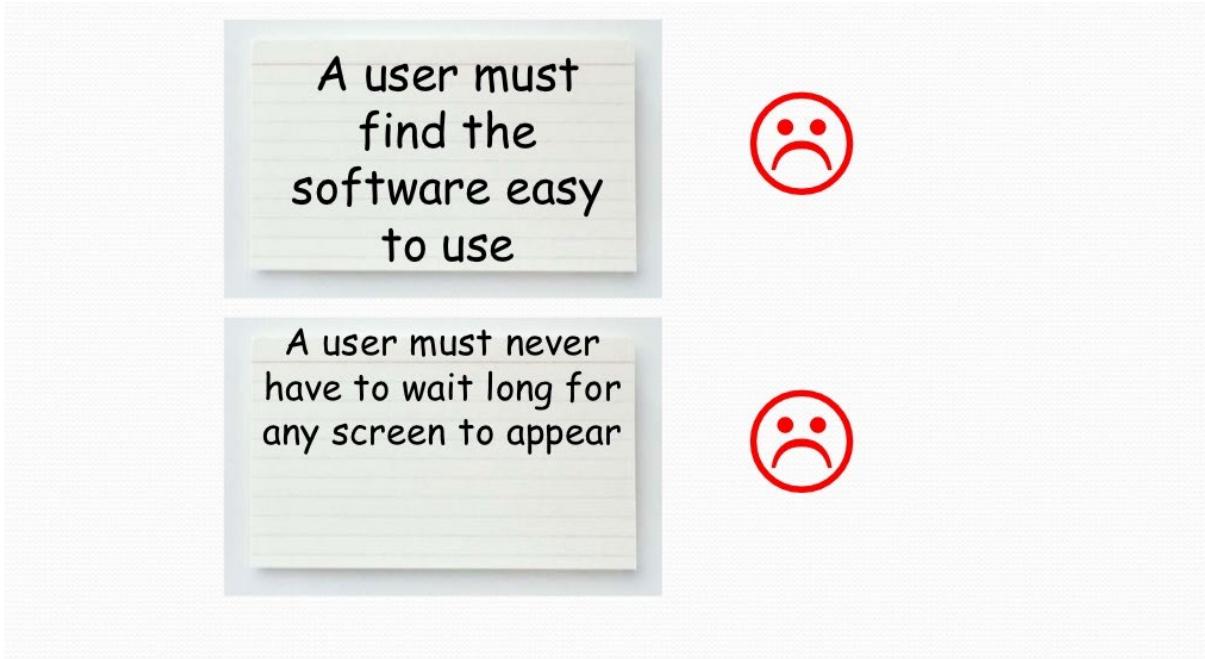
A job seeker can modify a CV already uploaded

- It is important for developers to be able to estimate or at least take a guess at the size of a story

3 common failures

1. Developers lack technical knowledge
2. Developers lack domain knowledge
3. Story is too big

Testable



Definition of Done

What satisfaction criteria apply to every user story so that the story is “out of my life”

May be different for different teams/sprints

IDEAS?

All unit, integration and acceptance tests are passed

Code is submitted to the repository and reviewed

The feature has been deployed on the test/staging/UAT/production environment

Documentation is completed and uploaded to the wiki

UAT is completed

CAB has signed it off

Smells...

- Too many interdependent stories -> make stories larger or slice them differently
- Goldplating -> increase visibility of tasks & workload
- Too many details -> if you run out of room, use smaller cards
- Thinking too far ahead -> stop
- Splitting too many stories
- PO can't prioritise -> close look at value

<https://www.youtube.com/watch?v=0HMsh459h5c>

5 Common Mistakes in User Stories

Dave Farley

Main takeaways



What are user stories?

Card

- Stories are traditionally written on note cards
- Cards may be annotated with estimates, notes, etc.

Conversation

- Details behind the story come out during conversations with the product owner

Confirmation

- Acceptance tests confirm whether the story was coded correctly



#171678565



Questions and Comments....



Tony Clear S2 2024

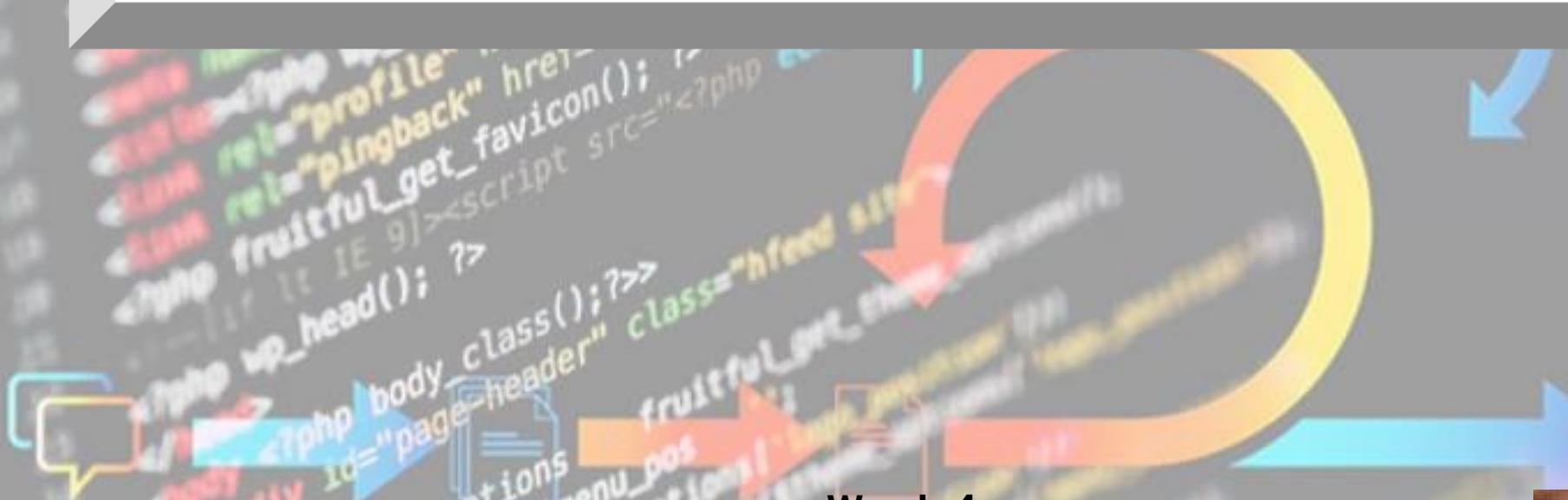
CISE ENSE701

I has a question...



60

Preparing to Iterate/Sprint



Week 4





Taking Stock

The schedule for the course

Where are we now?

The Assessment Schedule

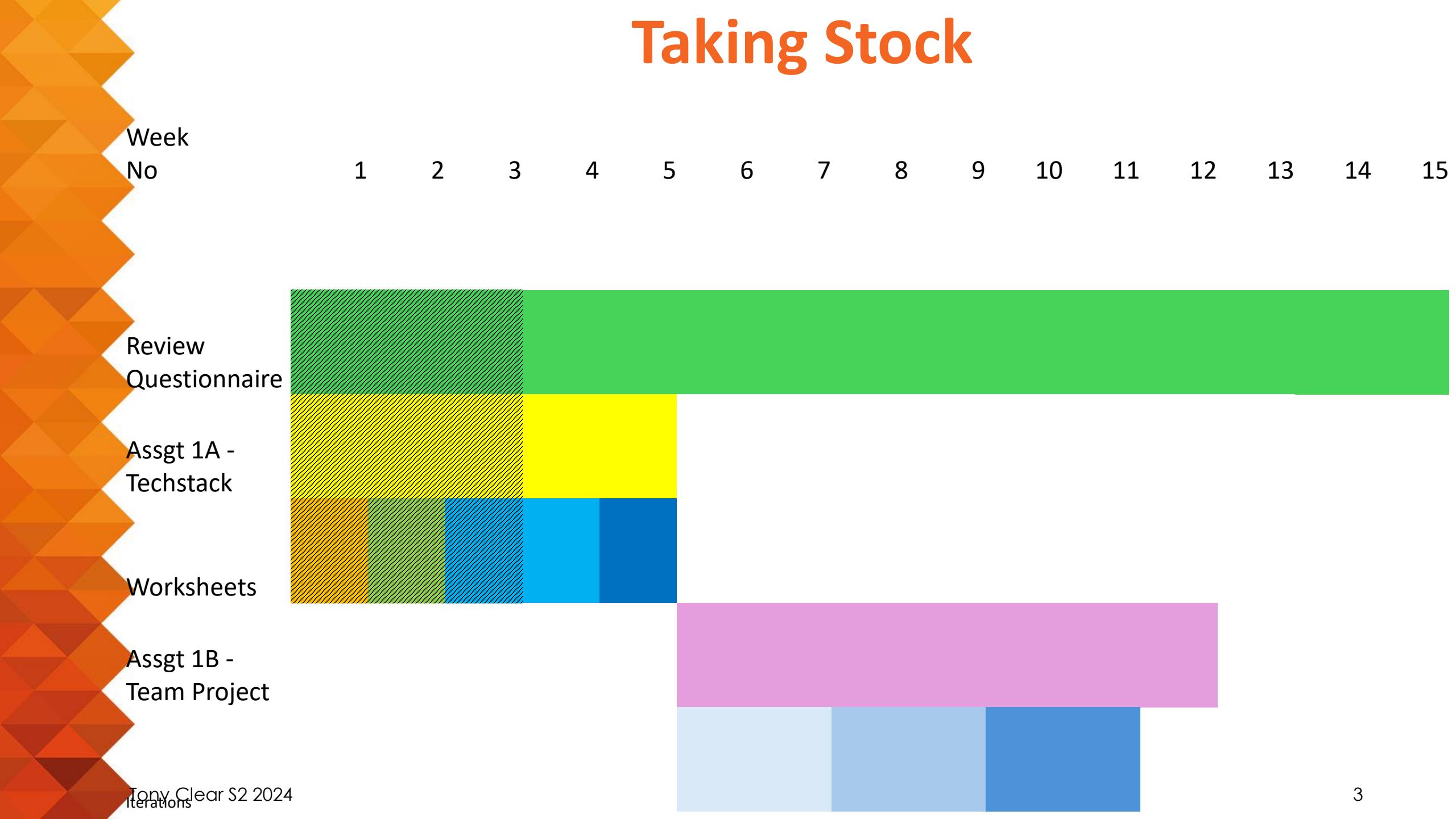
Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

Taking Stock



Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)

(Individual)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,

Setup the tools needed to work with the selected

Tech Stack (front-end/backend)

Set up tools to assure quality of product

Set up tools to deploy the product to the cloud

Set up tools to monitor and alert issues post deploy

Learn how to use the tools

Learn how to use the Tech Stack

Understand the product goals -> Product Backlog

Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)

Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)

(small team - 4 Including QA)

Capability building by Developing a Product in a small team

Apply a new tech stack and tool set

Practice DevOps and Scrum WoW

Collaborate with a Product Owner and team

Three sprints to learn fast – fast feedback

Submit – reviews weeks 7,9,11 (tutorials)

Capability and learning Portfolio with Evidence

Product increments

Sprint 1 weeks 5 and 6

Sprint 2 weeks 7 and 8

Sprint 3 weeks 9 and 10

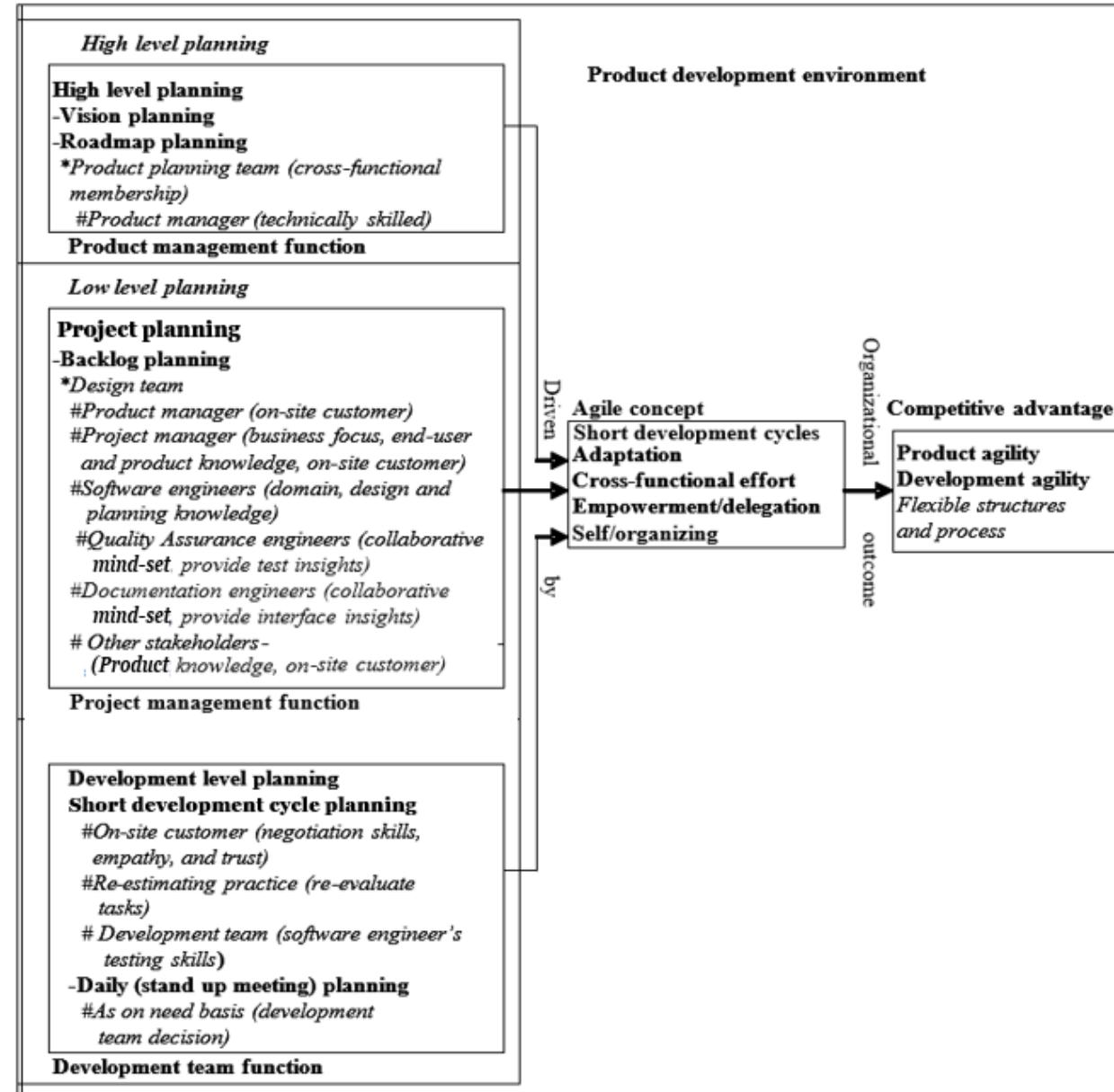
Ass 2 Knowledge Check (30%)

(Individual, online questions)

A set of questions about scenarios to confirm you have understood main language and principles

Sometime in Revision weeks (Faculty schedules)

Agile Planning - Levels



Lal, R., & Clear, T. (2021). Three Levels of Agile Planning in a Software Vendor Environment. In *Australasian Conference on Information Systems* (pp. 1-12).

<https://aisel.aisnet.org/acis/2021/48/>

Quick Recap on the CISE custom process for developing software

R3. Every project needs a slightly different methodology, based on those people characteristics, the project's specific priorities, and the technologies being used. This result indicates that **a team's methodology should be personalized to the team during the project** and may even change during the project.

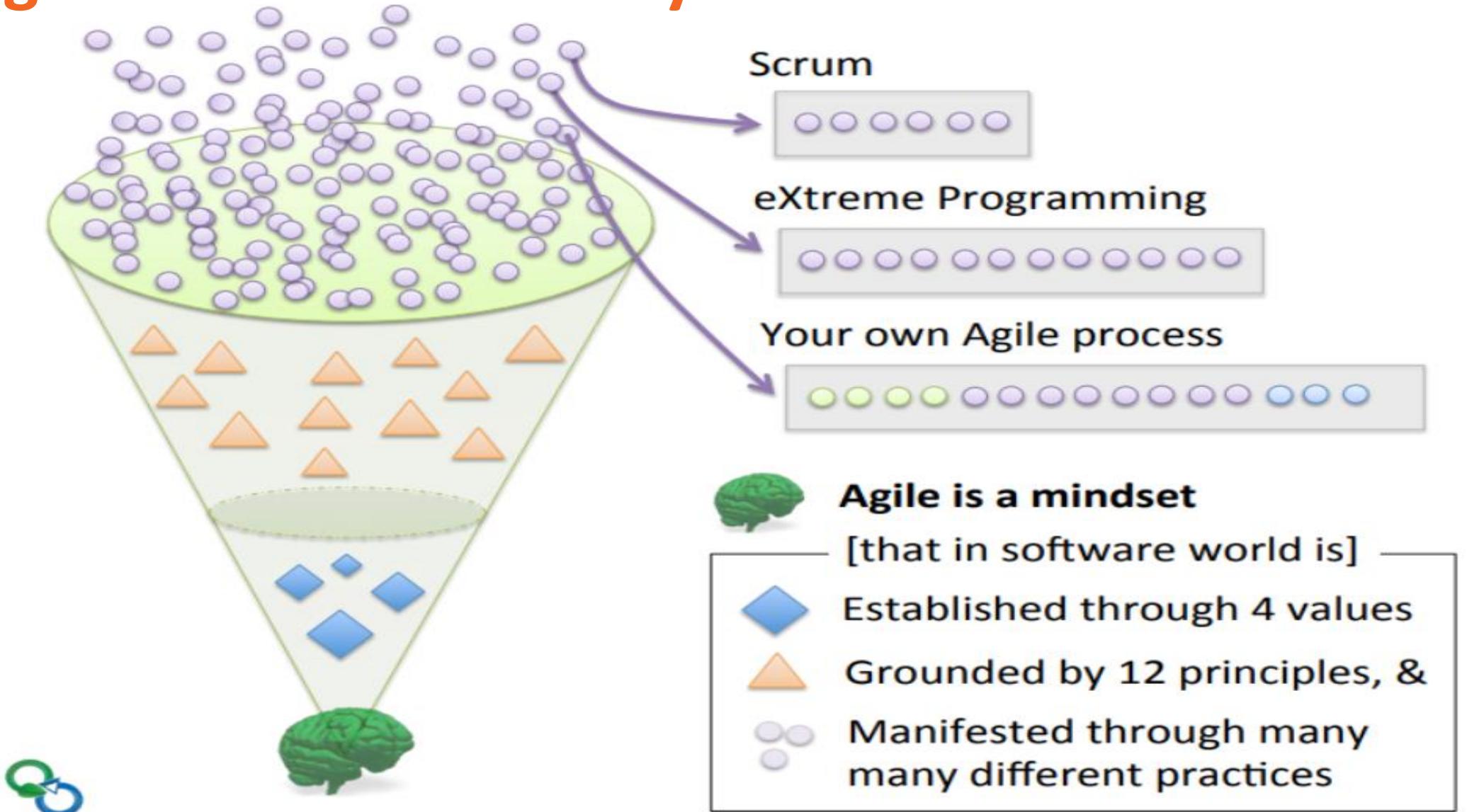
R6. All the above suggests a repeating cycle of behavior to use on projects.

1. The members establish conventions for their interactions — a base methodology — at the start of the project. This can be likened to them "programming" themselves.
2. They then perform their jobs in the normal scurry of project life, often getting too caught up to reflect on how they are doing.
3. They schedule regular periods of reflection in which they reconsider and adjust their working conventions.

Cockburn, A. (2003). *People and Methodologies in Software Development* [Doctoral Dissertation, University of Oslo]. Oslo. Retrieved 8/03/2022 from

https://www.researchgate.net/profile/AlistairCockburn/publication/253582591_People_and_Methodologies_in_Software_Development/links/56d434b208ae2ea08cf8e076/People-and-Methodologies-in-Software-Development.pdf

The Agile Manifesto – many WoW



Webinar by Ahmed Sidky – CEO of ICAgile course
<https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf>

How will we get feedback on product from users/client?

Regular review of product increment

Iteration of design/code/test Prod Increment

Iteration of design/code/test More Prod Increment

Iteration of design/code/test Final Prod Increment

What do we need to do before we start coding?

- Initial product backlog and story map (some uncertainty)
- User stories with Acceptance criteria
- Detail understanding and design of features for next iteration only
- Architecture and tech stack and deployment
- Dev Environment set up
- Plan for iteration 1

Goal and Iteration Backlog

How will we keep improving our process

Regular review of team process

CI/CD
PAIR and MOB
TDD

Iteration Planning meeting

How will we decide what is in each Iteration?

Regular team meeting during iteration...
Is there anything stopping us from reaching the goal?

How will we coordinate work with each other and keep on the same page?

How will we manage changes to requirements?

How will we manage risks?

How will we assure quality?

User stories will be how we document user requirements

WHY?

What about system requirements and features?

User stories will be the unit of work for

Understanding user needs

Splitting up work

Designing product features

Testing product features

Organising the order of doing work

Planning iterations

- Estimation
- -hypothesis

Monitoring work

Lifecycle of User Stories

Discover user needs

Product Goal

Many sources, techniques

IDENTIFY USER TYPES

Write high level user Stories (EPIC Stories)

Keep these on Monitoring board – Product Backlog

Break into smaller user stories and include
Acceptance criteria, success criteria, DOD

Keep these on User Story Map – product overview

Decide on the order to work on
First product hypothesis to test

<https://www.youtube.com/watch?v=Hq9O7mnUNM4>

Two sets – ordered near the top, unordered below them
Capture in Product Backlog and User Story Map

Decide on which will be in the next iteration
(Starting from top of PB)

Get detailed SHARED understanding from PO

Translate into design and code

Estimate how many user stories team can do in
one iteration (team capacity)

Monitor progress and adjust

Estimate the size of each user story and add
user stories until team capacity is reached

How to manage change

Skills we need?

Discovering User Stories (user requirements)

Big upfront effort

plan-driven control based on high-confidence (certain), **long-term** predictions

Iterative and incremental effort

Frequent opportunity for changes based on empiricism and short learning loops and **short-term** certainty and long-term uncertainty.

User Story Workshops – product stakeholders, PO, BA, Dev, Tester etc
Stakeholders write them and group (and agree on order or in/out)?

Interviewing users or the PO – get placeholders for future conversations about needs, changes to the current situation, and some detail for user needs that are of most value to be worked on first

Observation, surveys, impact mapping, customer journey mapping.....

INVEST – a quick checklist to check the usefulness (quality) of user stories

The INVEST checklist comes from [INVEST in Good Stories](#) by Bill Wake. It's an acronym with six important quality characteristics for user stories:

Independent — Can the story stand alone by itself ?

Negotiable — Can this story be changed or removed without impact to everything else?

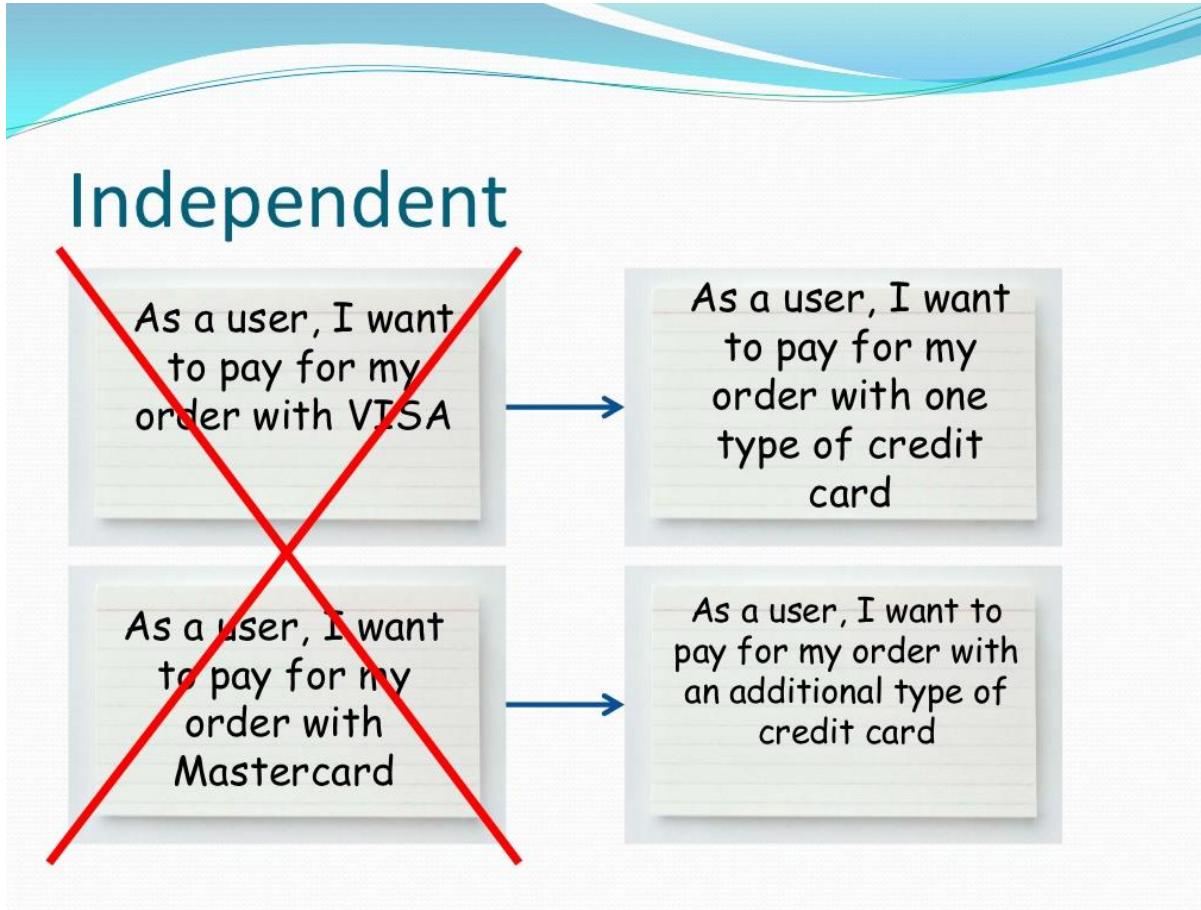
Valuable — Does this story have value to the end user?

Estimable — Can you estimate the size of the story?

Small — Is it small enough?

Testable — Can this story be tested and verified?

INVEST



Negotiable – not a contract!

As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express.
Consider Discover



As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express. Consider Discover. On purchases over £100, ask for card ID number from back of card. The system can tell what type of card it is from the 1st two digits of the card number. The system can store a card number for future use. Collect the expiration month and date of the card



Estimable

A job seeker can modify a CV already uploaded

- It is important for developers to be able to estimate or at least take a guess at the size of a story

3 common failures

1. Developers lack technical knowledge
2. Developers lack domain knowledge
3. Story is too big

Testable

A user must
find the
software easy
to use



A user must never
have to wait long for
any screen to appear



Definition of Done

What satisfaction criteria apply to every user story so that the story is “out of my life”

May be different for different teams/sprints

IDEAS?

All unit, integration and acceptance tests are passed

Code is submitted to the repository and reviewed

The feature has been deployed on the test/staging/UAT/production environment

Documentation is completed and uploaded to the wiki

UAT is completed

CAB has signed it off

Using a story map to slice out a delivery plan or sprint plan



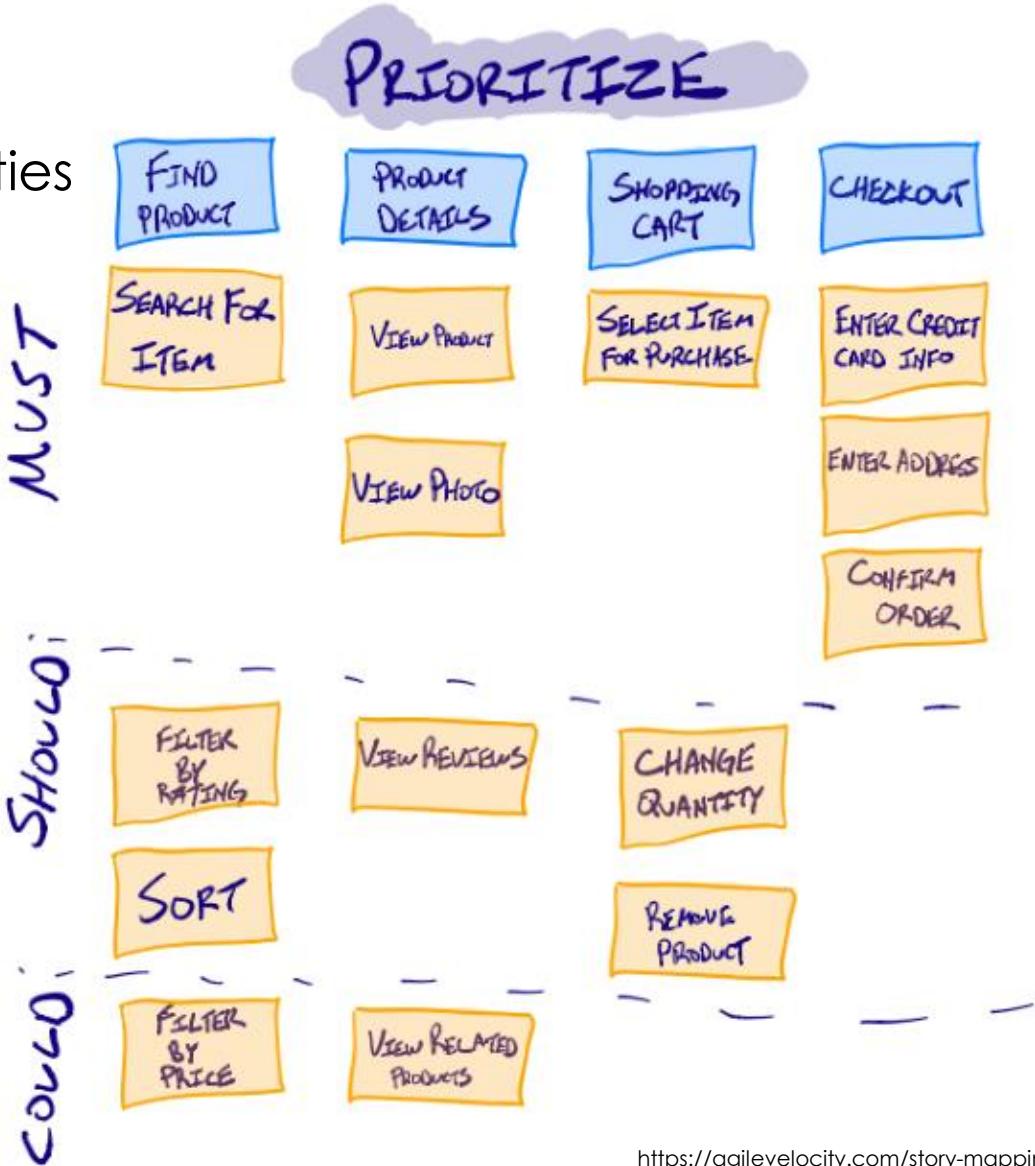
Jeff Patton & Associates, jeff@jpattonassociates.com, [twitter@jeffpatton](https://twitter.com/jeffpatton)



80

The anatomy of a Story Map

Columns relate to user activities



Each column has the system features related to the user activity for that column

Can create other horizontal slices
Of features to represent the scope of delivery cycles or MVP or sprint cycles

Story maps and sprint plans – Miro a useful tool

https://miro.com/index/?utm_source=public_board

The screenshot shows a Miro board titled "Frame 1" with the sub-section "User Story Map". The board contains several story cards:

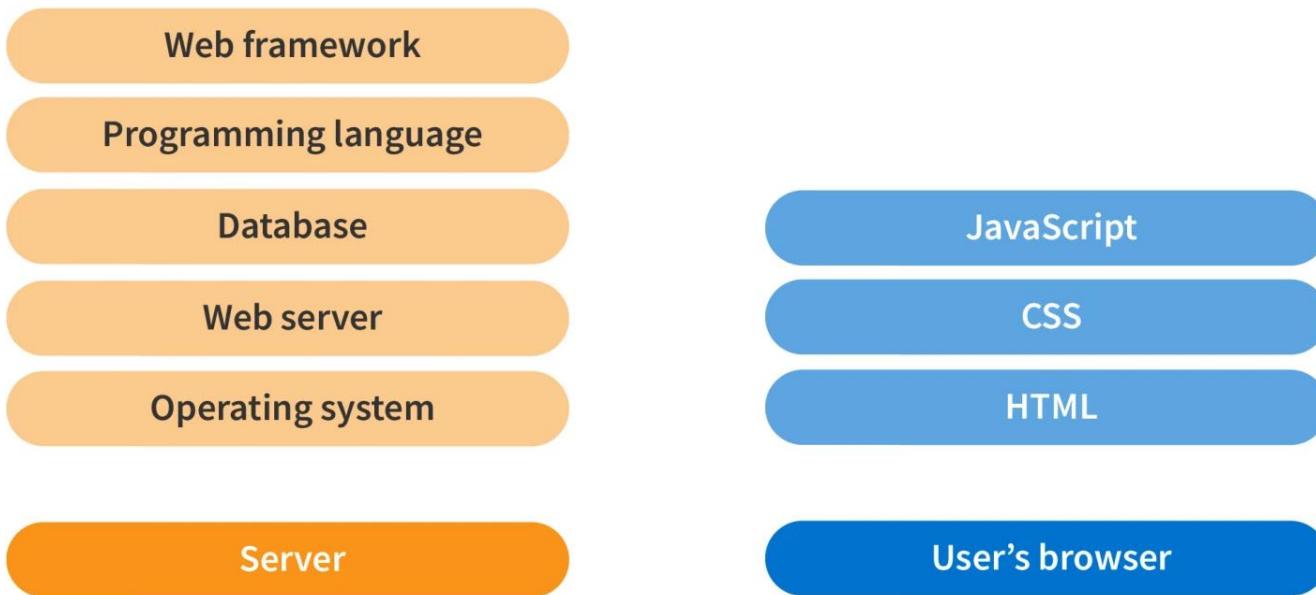
- User Story Map:
 - Searching Database
 - Changing Database
 - Find intended articles
 - Submitting article for addition to site
 - Alter information
- Sprint 1 | 8:
 - Filter search by practice type
 - Have submission portal to submit relevant article information
 - Select what columns of information to display
 - Select year range of articles
- Sprint 2 | 5:
 - Save search queries
 - Notify submitter if article is approved or declined
 - Allow access to modify database information to admins
 - Notify when article is ready for moderation

On the left side of the board, there is a vertical sidebar with icons for a hand, a double arrow, and a list.

Technology stacks and Roadmapping strategy

<https://www.aha.io/roadmapping/guide/it-strategy/technology-stack>

<https://www.aha.io/roadmaps/overview>



Back-end

Front-end

© 2021 Aha! Labs Inc.

User Story Success Criteria

*acceptance criteria, Acceptance

A list of “rules” or criteria or tests or behaviours that should be met if the story is to be successful

Behaviour Driven Development (BDD)

Acceptance Test Driven Development (ATDD)

As a purchaser I want to be able to pay online by credit card for convenience

Possible Success criteria?

Common template

Initial situation

Event

New situation (output)

Given <??????>

When <??????>

Then <??????>

Pre-sprint – Big picture planning schedule when we know the least

3-month – Big room planning

6-week blocks – ShapeUp

Roadmap

Release plan

Big focus on delivering **Value** to users through
Agreeing on and refining

Product goal(s) and
Sprint goals

We value responding to change over following a plan

In our case – 3 short sprints

Roadmap = 3 x 9 day sprints (dates) with 3 sprint goals

Release plan = deploy increment every sprint

Pre-sprint – Selecting what to work on for first sprint

Some estimate of the **size** of a user story

Some estimate of how many user stories the team can do in a sprint (**team's capacity** or velocity)

Keep selecting sized user stories from the top of the PB until the team's capacity is reached (or almost)

The sum of all user story sizes = or < team's capacity

Need to measure story size and team capacity in the **same units**

Size based on complexity, familiarity/novelty, effort

Function Point Analysis. Cocomo

Absolute estimation

Hours/time

Relative estimation – need a baseline user story with defined size to compare to

Story Points

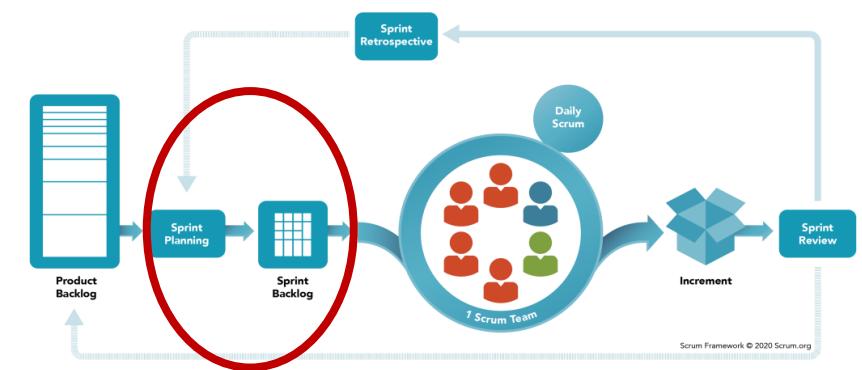
T-shirt sizes

NO NEED TO BE PRECISE!

Make all user stories around the same small size

Measure team sprint velocity in user stories

If deliver small changes and deploy quickly then no need to estimate size (read about #noestimates movement)



Using team diversity to get a good estimate of a user story “size”

Software Project Estimation - Issues

Being able to predict is a hallmark of any meaningful engineering discipline and software engineering is no exception.

Researchers have been exploring prediction systems for areas such as cost, schedule and defect-proneness for more than 40 years. ...**empirical evaluation has not led to consistent or easy to interpret results.**

This matters because it is hard to know what advice to offer practitioners who are — or who ought to be — the major beneficiaries of software engineering research.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8), 820-827.

Software Project Estimation - Examples

Three examples of inconsistent systematic review findings are:

- Jorgensen [13] reviewed 15 studies **comparing model-based to expert-based estimation.**
 - Five of those studies found in favour of expert-based methods,
 - five found no difference, and
 - five found in favour of model-based estimation.
- Mair and Shepperd [25] **compared regression to analogy methods for effort estimation** and similarly found conflicting evidence. From a total of 20 empirical studies,
 - seven favoured regression,
 - four were indifferent and
 - nine favoured analogy.
- Kitchenham et al. [21] found seven relevant empirical studies for the question **is it better to predict using local, as opposed to cross-company, data.**
 - Three studies reported it made no significant difference,
 - whilst four found it was better.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8), 820-827.

Software Project Estimation in ASD?

How big is the software to be developed?

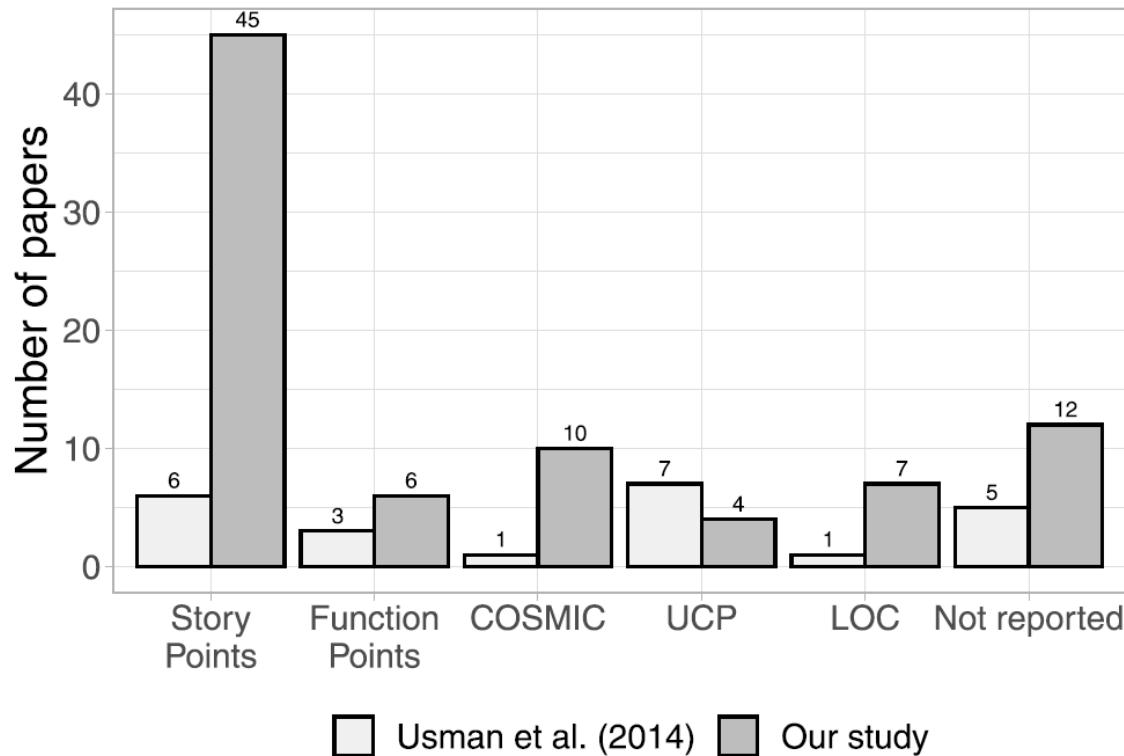


FIGURE 5. Size metrics: Comparison of our results with those of Usman *et al.* (2014).

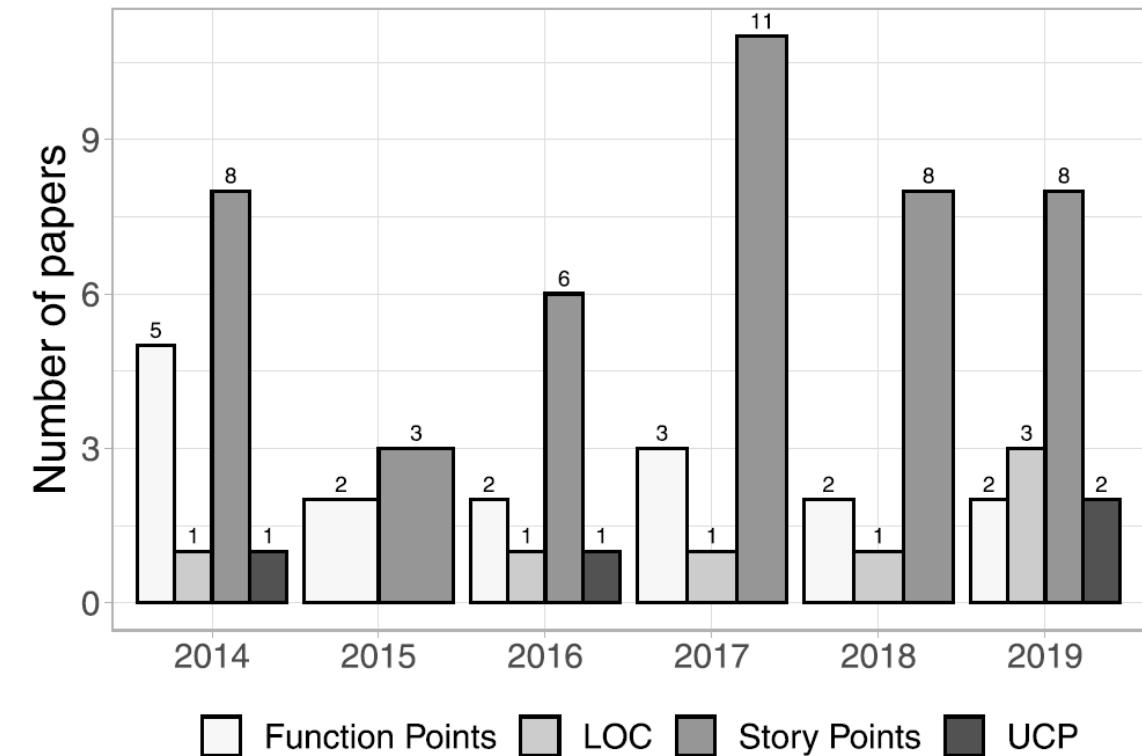


FIGURE 6. Size metrics used in the papers per year of publication.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*, 8, 166768–166800. <https://doi.org/10.1109/ACCESS.2020.3021664>

Software Size Estimation – Function Points as one technique?

1 A SHORT INTRODUCTION TO FUNCTION POINT ANALYSIS

FP (Function Points) is the most widespread functional type metrics which is suitable for quantifying a software application. It is based on 5 user identifiable logical "functions" which are divided into 2 data function types and 3 transactional function types (Table 1). For a given software application, each of these elements is quantified and weighted, counting its characteristic elements, like file references or logical fields.

	Low	Average	High
ILF (Internal Logical File)	7	10	15
EIF (External Interface File)	5	7	10
EI (External Input)	3	4	6
EO (External Output)	4	5	7
EQ (External Inquiry)	3	4	6

Table 1. Complexity weights with corresponding number of UFP.

The resulting numbers (Unadjusted FP) are grouped into Added, Changed, or Deleted functions sets, and combined with the Value Adjustment Factor (VAF) to obtain the final number of FP. A distinct final formula is used for each count type: Application, Development Project, or Enhancement Project.

Meli, R., & Santillo, L. (1999). Function point estimation methods: A comparative overview. FESMA,

Software Project Estimation in ASD - Mobile Apps?

How can we determine how big is the software to be developed – mobile apps?
...the techniques most commonly used for mobile apps were

- **Function Size Measurement and**
- **Expert Judgment.**

planning and development of **mobile applications differs from other traditional software applications** characteristics of the mobile environment...;

- high autonomy requirements,
- market competition,
- and many other constraints.

With regard to the **size metrics and cost drivers**, the results showed that

- the **number of screens**
- and **type of supported platform for smartphones**

most common factors used to measure the estimation prediction.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*, 8, 166768-166800. <https://doi.org/10.1109/ACCESS.2020.3021664>

Software Project Estimation in ASD – some conclusions?

...a combination of data-based methods (using project historical data and context-specific methods) with expert-based methods may be promising in improving accuracy levels.

.. been a decrease in the use of general-purpose size metrics (e.g., UCP) and increase in the use of size metrics that take agile methods into account.

In this respect, **Story Points was the metric most frequently used to determine the size of the software.**

Usman *et al.* [60] also found that **Story Points is the size metric most frequently used by agile teams.**

However, some authors suggest that:

- Story Points should be estimated collectively ... to reach a team consensus so as to
 - alleviate the chances of over-optimism and
 - reduce the issues
 - of anchoring and
 - strong personalities

Working with the Client: Useful Questions to Ask

- ➊ What business problem are you trying to solve?
- ➋ What's the motivation for solving this problem?
- ➌ What would a highly successful solution do for you?
- ➍ How can we judge the success of the solution?
- ➎ Which business activities and events should be included in the solution? Which should not?
- ➏ Can you think of any unexpected or adverse consequences that the new system could cause?
- ➐ What's a successful solution worth
- ➑ Who are the individuals or groups that could influence this project or be influenced by it?
- ➒ Are there any related projects or systems that could influence this one or that this project could affect?

Pre-sprint – User Requirements_v1 User Stories part 1

Significant user type with characteristics that distinguish from other user types that may affect the design. NOT "User"

The new capability the user wants so they can reach smaller goal (outcome) In "business" language & NO solution details

What is the goal (desired outcome) of this new user capability?

As a <????> I want to be able to <?????????> so that <??????????>

As the PO I want to have lots of articles in the evidence repository

As a practitioner, I want lots of evidence to be available so the evidence is convincing

I want some way to be able to keep adding articles with evidence for analysis to add evidence to the repository

I want people to be able to add new evidence so the repo gets bigger and I leverage crowd sourcing

As a **researcher** I want to be able to **recommend articles to include in the evidence repository** so that **the evidence available keeps expanding**

We should check the article is about SE with evidence (relevance)

We should check the article is not already in the repo (avoid duplicates)

We should check the quality of the evidence in the article (quality)

The submitter should be informed/thanked if the article they submitted is accepted or not (and why not)

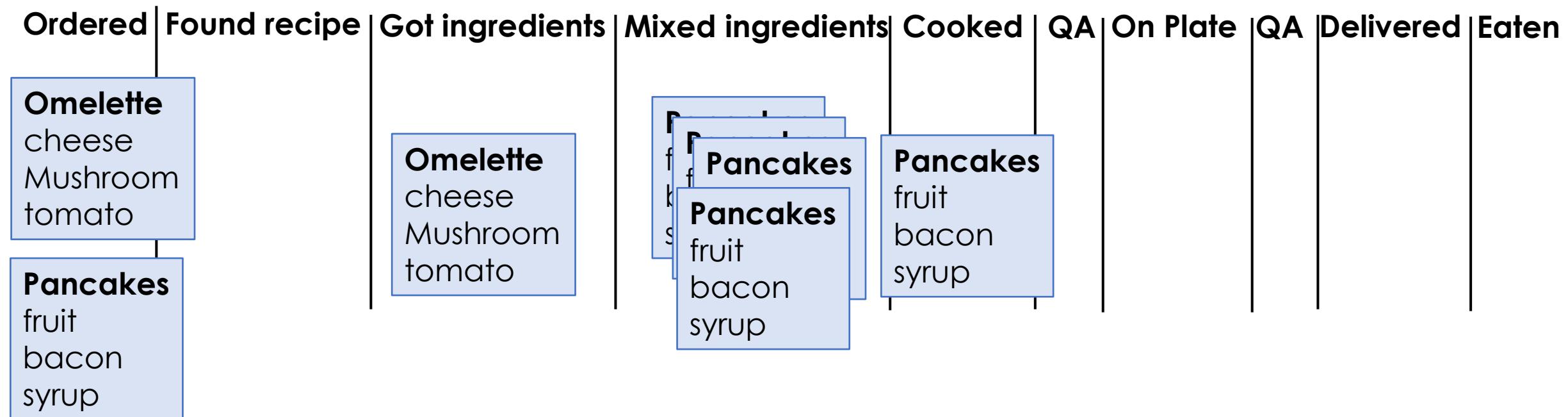
Placeholders for conversations.

Estimating story points??

Monitoring progress during a Sprint and overall

Decide on a set of status labels for user stories- these are the columns in a work board

Target is 10 minutes after order



User Story Board

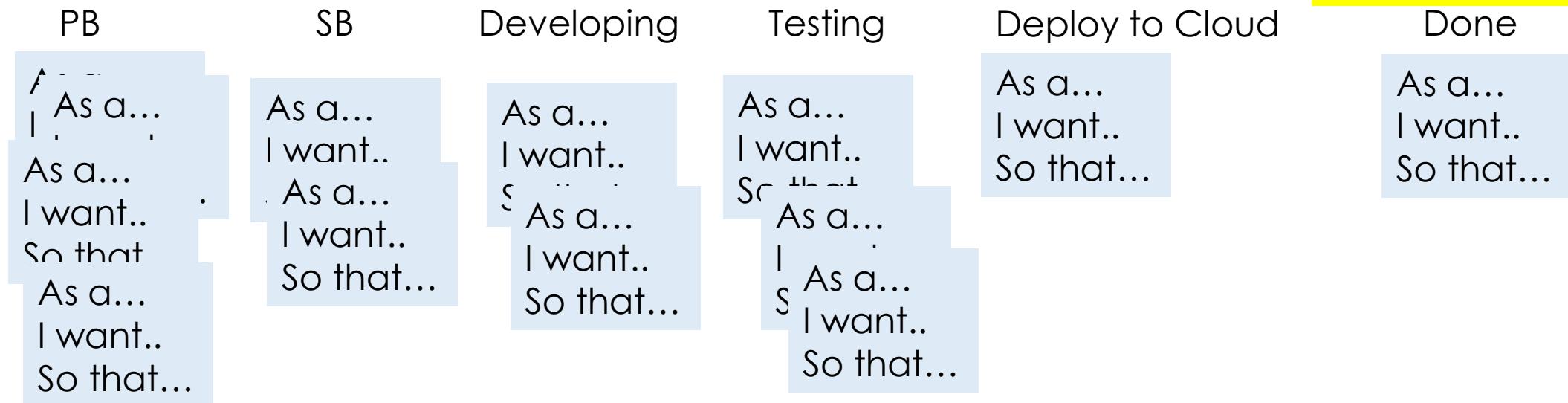
Physical Boards

Electronic Boards

- Trello
- Asana
- Jira
- Azure DevOps

Advantages/Disadvantages?

States of User Stories



Definition of Done?

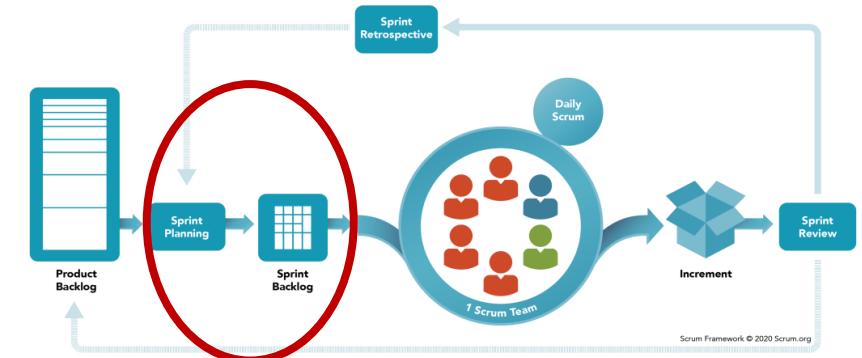
Pre-sprint – Design, tech stack, architecture Non-functional (quality) Requirements

Layered architecture

Front-end NEXT.js

Back-end Nest.js on Node.js

Database. MOngoDB



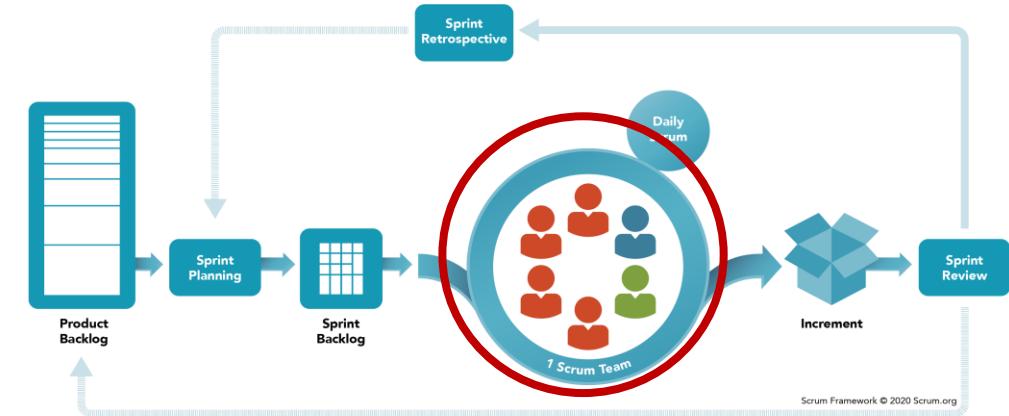
Working as an Individual in a team - workflow

Continuous Integration workflow

Coding standards, code craft, code quality

Non-functional requirements

NOT this should be easy to use and perform well



Team Collaboration

How to be a high Performing team (or at least a team that does not want to kill each other!) (revisit later)

Build trust and share expectations and values

What would happen if NOT true?

Coordinate work, expectations and goals frequently

Scrum meeting

Reflect on how the team works together and learn from previous work to improve

Feel safe together

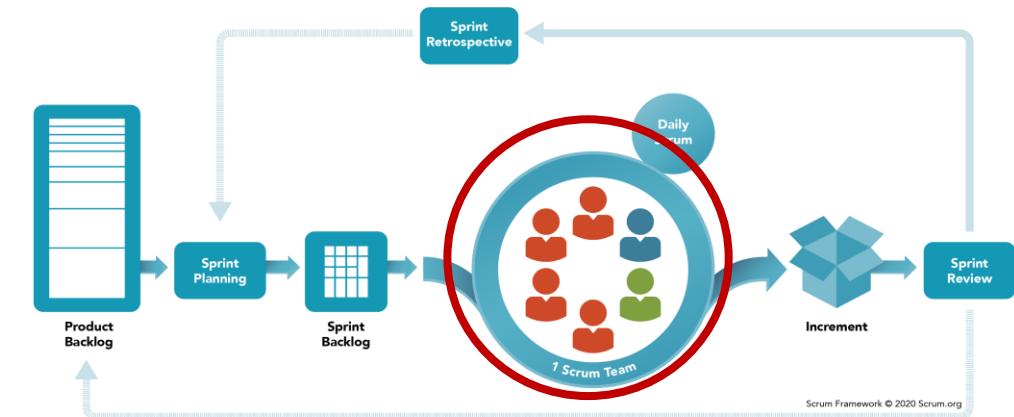
Team ownership/success trumps individual

Break work up – integrate work Cont. Integration

Check work and goals frequently with Users/PO

Work together on the same work – **mob programming**

Solve problems together, share understanding, learn from each other



Risk planning (revisit this later)

Risk – chance of not meeting expected quality or goals

Describe quality and goals!

HIGH IMPACT?

Risk of making something that does not meet expectations of PO

Risk of making something that does not get used

Risk of learning new tech taking longer than predicted

What else?

Risk that the product is not robust

Risk code is not easy to change

Risk that someone with critical knowledge is not available to team

Expectations for Sprint 1

Dev environment and tool pipeline set up for each team member including single repo in GitHub

Product Backlog

Epic stories, detail for priority stories

User Story Map

Sprint Backlog

Based on some way of estimating story sizes and team velocity
Try Planning poker

Continuous integration

Frequent build (every few days)

Testing automated to some extent

Mob programming tried

Releasable Product Increment

Deployed to cloud

Comparative Agility Assessment

At the highest level, the CA approach assesses agility on seven *dimensions*:

- Teamwork;
- Requirements;
- Planning;
- Technical Practices;
- Quality;
- Culture; and
- Knowledge Creating.

Each dimension is made up of three to six *characteristics* for a total of 32 characteristics.

Williams, L., Rubin, K., & Cohn, M. (2010). Driving Process Improvement via Comparative Agility Assessment. In *2010 Agile Conference* (pp. 3-10).
<https://doi.org/10.1109/AGILE.2010.12>



#171678945



Questions and Comments....



Tony Clear S2 2024

CISE ENSE701

I has a question...





ENSE701

Contemporary Issues in Software Engineering

Week 5 Lecture : Requirements Prioritization in Scaled Agile
Distributed Software Development

What is requirement?

- It is about What not How
- It is about need
- IEEE – A condition or capability needed by user to solve a problem or achieve an objective.

What is Requirements Engineering (RE)

- RE, which is a branch of Software Engineering (SE), deals with discovering, specifying, analysing, and documenting the requirements of a system.
- Each software development process goes through the phase of requirements engineering

Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements
- However, there are a number of generic activities common to all processes

Requirements elicitation, analysis and prioritization

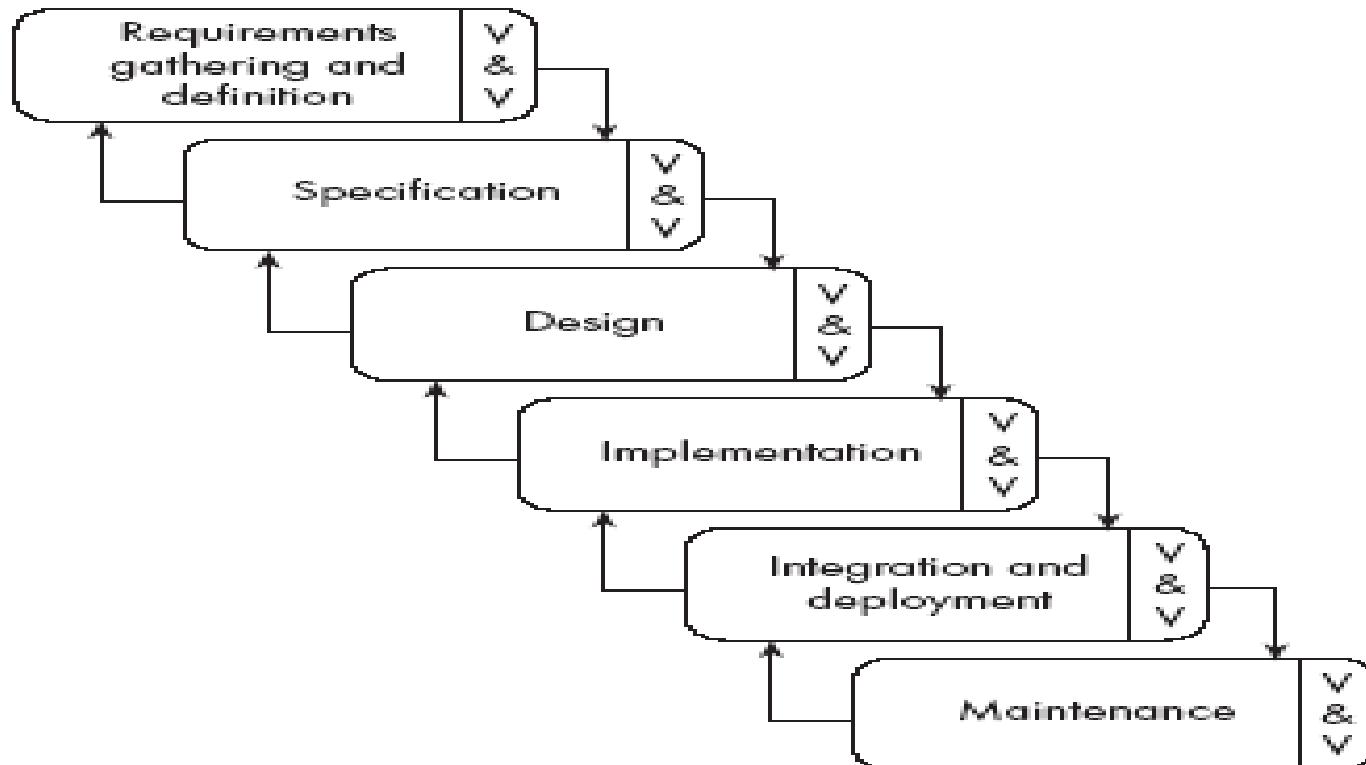
Requirements specification

Requirements validation

Requirements management

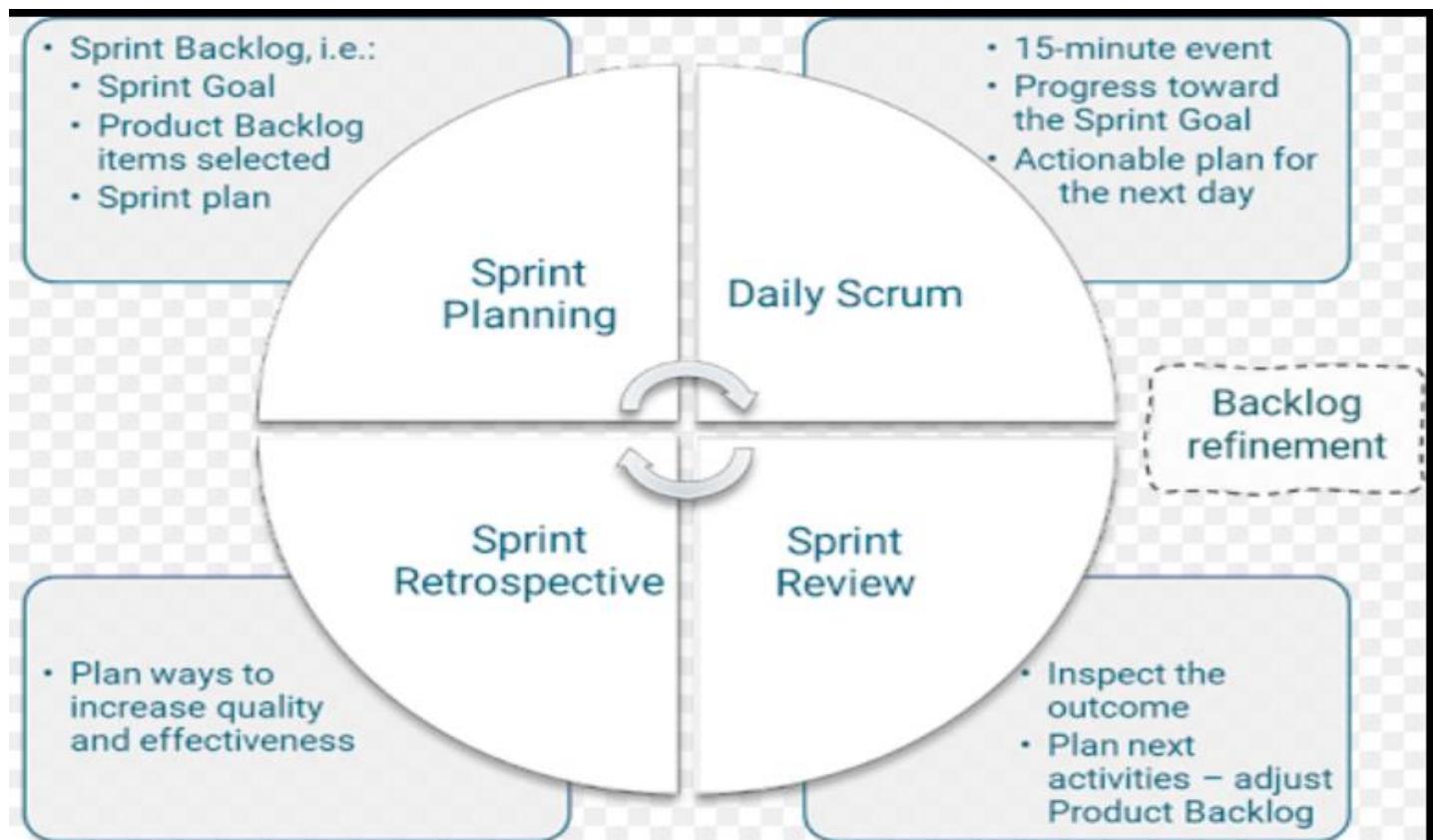
RE and software development methods

- RE in classic development methods (e.g., Waterfall)



RE and Software development processes

RE in Agile development (e.g., Scrum)



RE and Software development processes

- RE in Agile development (e.g., Scrum)

RE activity	Scrum implementation
Requirements Elicitation	<ul style="list-style-type: none">• Product Owner formulates the Product Backlog.• Any stakeholders can participate in the Product Backlog.
Requirements Analysis	<ul style="list-style-type: none">• Backlog Refinement Meeting.• Product Owner prioritizes the Product Backlog.• Product Owner analyzes the feasibility of requirements.
Requirements Documentation	<ul style="list-style-type: none">• Face-to-face communication.
Requirements Validation	<ul style="list-style-type: none">• Review meetings.
Requirements Management	<ul style="list-style-type: none">• Sprint Planning Meeting.• Items in Product Backlog for tracking.• Change requirements are added/deleted to/from Product Backlog.



Main topic of today's lecture: Requirements prioritization in Scaled Agile Distributed Development

- Adoption of basic Agile methods at the enterprise level
- Scaling Agile frameworks (e.g., DAD, SAFe) to support enterprise-wide agility and scale agile practices

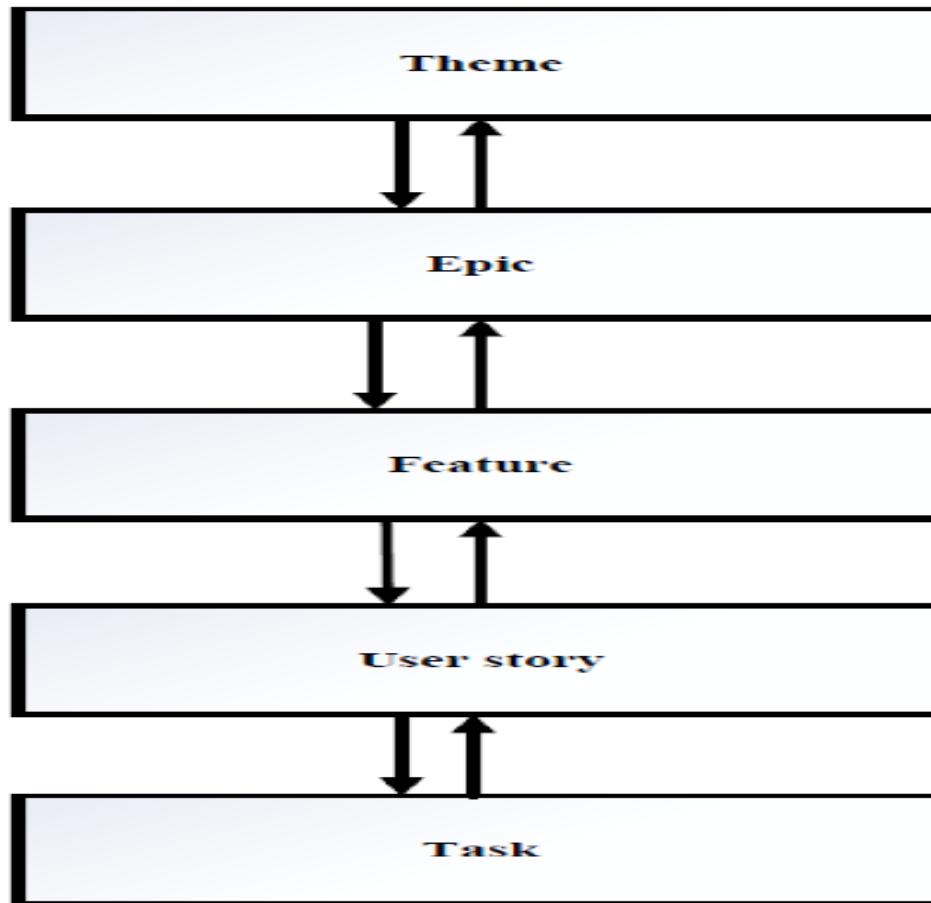
Requirements prioritization in Scaled Agile Distributed Development

- Prioritization of requirements as decision making process
- Prioritization is beyond the team level in scaled Agile development

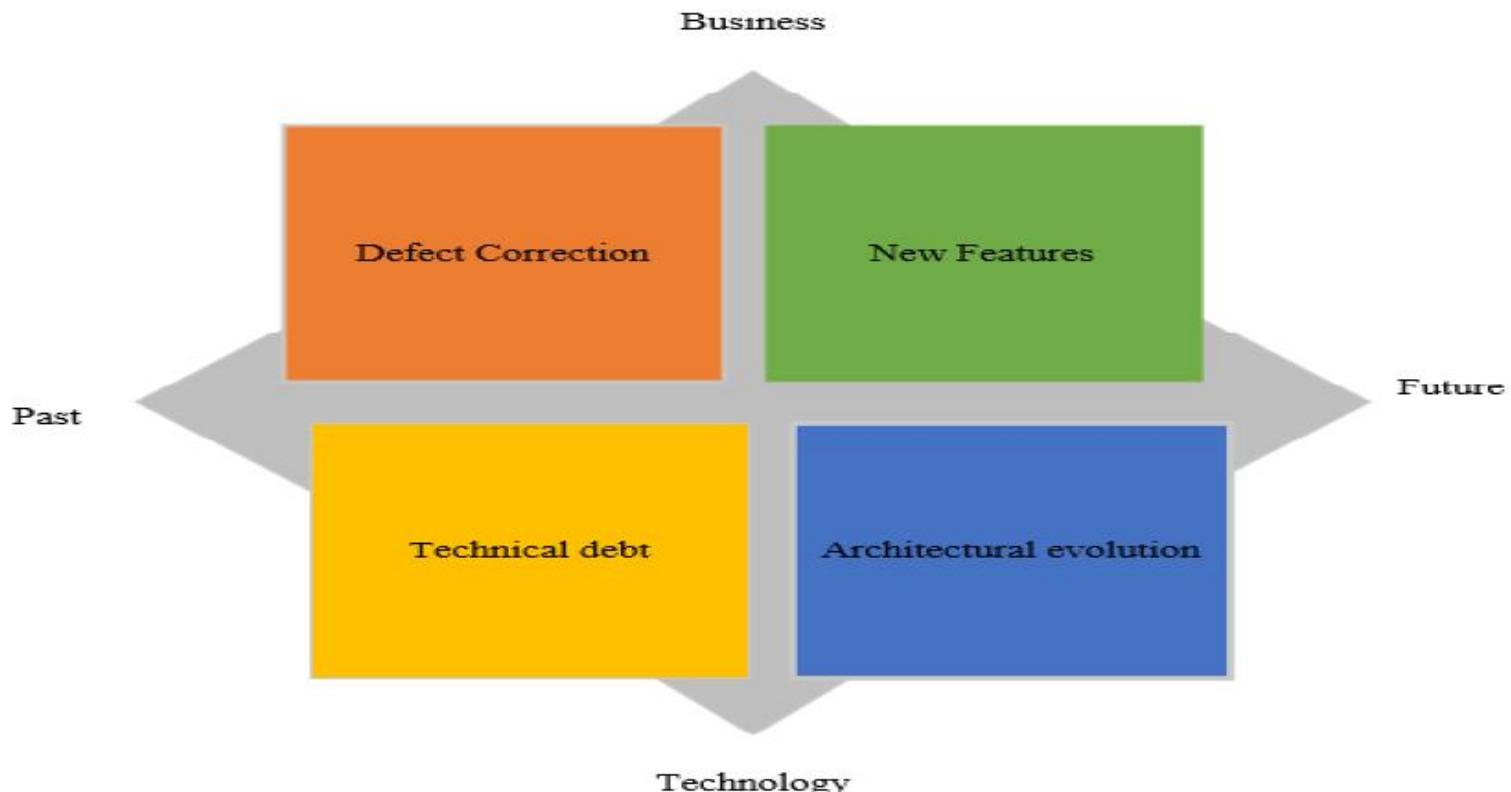
Key decision-making levels

- Portfolio level
- Domain level/Program level
- Team level

Flow of requirements across scaling agile decision-making levels



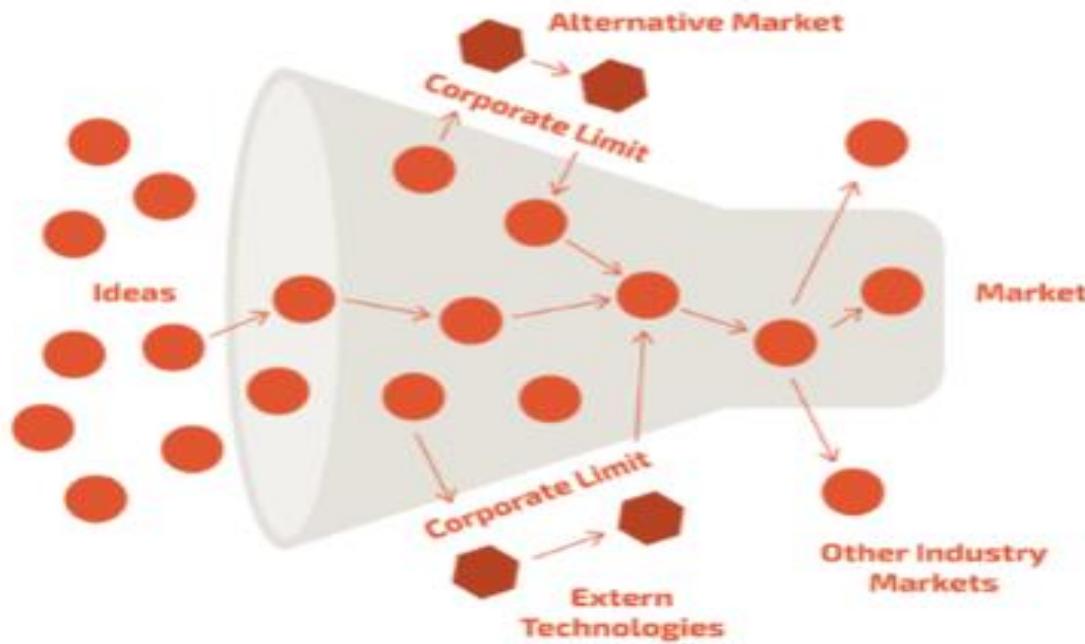
Requirements classification across scaling agile decision-making levels



Open innovation approach for requirements discovery across scaling agile decision-making levels

- Internal sources (e.g., portfolio management, product management, developers)
- External sources (e.g., customers, industry analysts, market research)

Open Innovation Model



Decision-making at Portfolio level

Two main objectives at the portfolio level in terms of decision making:

- Business goals that connect with organization's overall business strategy are decided and prioritized
- HLRs towards achieving business goals are formally signed off, communicated, and allocated to the engineering for implementation

Key practices for Decision-making at Portfolio level

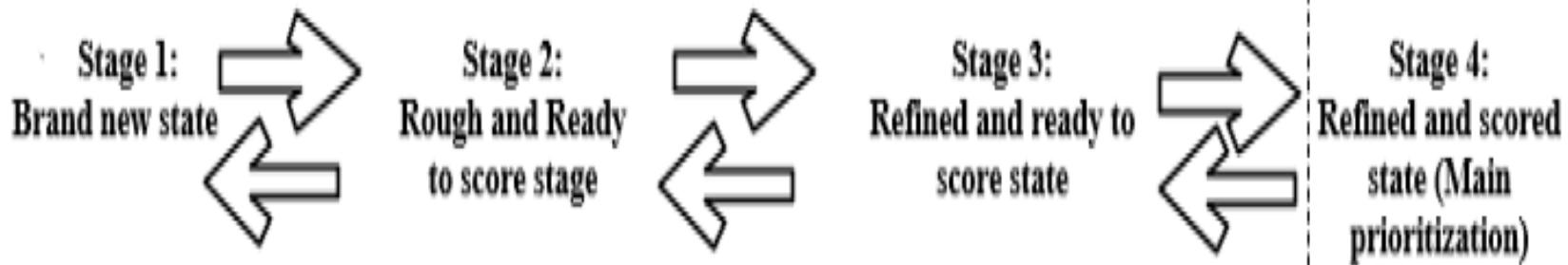
- Portfolio level – supported via inter-iteration prioritization
- Portfolio management – cross-functional decision-making team
- Continuous decision-making – typically quarterly, monthly/fortnightly check-ins, ad-hoc meeting if needed
- Combination of quantitative as well as qualitative practices

Domain level/Program level

Plays two-fold role during decision-making

First part of domain level:

- Initial decision-making on HLRs that emerge from each of the domains towards achieving strategic themes and/or contributes to the formation of strategic themes.
- Typically supported via intra-iteration prioritization



Domain level/Program level

Second part of domain level:

- Inter-iteration prioritization across teams (project specific)
- Decomposition of HLRs to implement them via short development cycles

Team Level

- Typically form intra-iteration prioritization
- Generally employed basic Agile methods (e.g., Scrum) for decision-making on the priority of requirements
- Detailed requirements provide required information to the delivery teams (e.g., user story)
- Priority decisions within the team

Boundary spanning mechanisms across scaling agile decision-making levels

- Boundary spanning is the act of bringing together two or more groups of people who are typically separated by location, that is, locally distributed and/or globally distributed., or separated by hierarchy, or a function
- Boundary spanning yields knowledge acquisition, negotiation, consensus building, and conflict resolution which are the key activities typically needed while making decision on the priority of requirements

Boundary spanning mechanisms

Three categories of boundary spanning mechanisms

- Boundary spanning event(s),
- Boundary spanning requirement artefact(s),
- Boundary spanner role(s)



Collaborative technologies (CTs) across scaling agile decision-making levels

- CTs enable collaboration with distributed members of a decision-making team
- Enable shared understanding of priority of requirements across scaling agile decision-making levels
- CTs- as synchronous and asynchronous mechanisms for decision-making across scaling agile levels
- Spreadsheet, Jira, AHA, ADO, Confluence, PowerPoint, MSTeams, Email

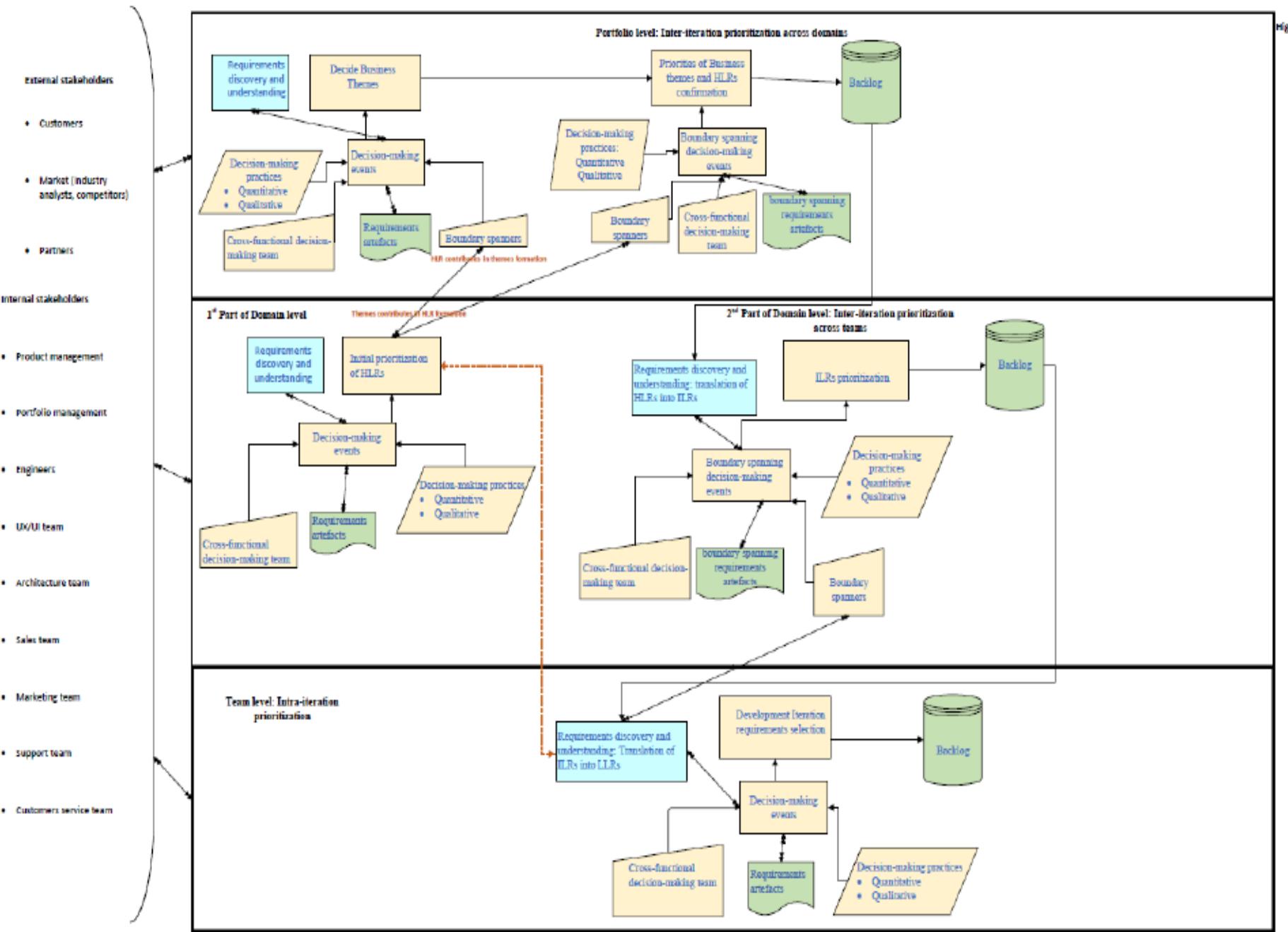
Usefulness of CTs during decision-making

CTs	Role of CTs for requirements prioritization across scaling agile levels				
	Requirements discovery and understanding - elicitation of requirements - knowledge sharing- shared understanding of requirements	Requirements artefacts	Decision making practices- priority score matrix	Decision making events: collaborate with cross site members over requirements	Scaling agile levels
Spreadsheet	For knowledge sharing- <i>shared understanding of requirements</i>	Spreadsheet as an informal Portfolio backlog		decision making event(s) performed at the portfolio level	Primarily employed at the Portfolio level, Can be used at the other levels as well (i.e., domain level, team level)
PowerPoint	For knowledge sharing- <i>shared understanding of requirements</i>			decision making event(s) performed at the portfolio level	Primarily employed at the Portfolio level, Can be used at the other levels as well (i.e., domain level, team level)
AHA	Requirements elicitation- <i>Online customer portal configured via AHA</i> Knowledge sharing- <i>shared understanding of requirements</i>	Backlog (i.e., Portfolio backlog), template configured for HLRs, template configured for strategic themes,	Automated scoring matrix for HLRs configured in AHA	decision making event(s) performed at the portfolio level, 1st part of domain level	Portfolio level, 1st part of domain level

Usefulness of CTs during decision-making

CTs	Role of CTs for requirements prioritization across scaling agile levels				
	Requirements discovery and understanding - elicitation of requirements - knowledge sharing- shared understanding of requirements	Requirements artefacts	Decision making practices- priority score matrix	Decision making events: collaborate with cross site members over requirements	Scaling agile levels
MSTeams Features of MSTeams <i>Screen sharing, Instant messaging, Team specific channel, Tag to notify people, recording of decision-making events, Offline chat audio/video call, web conference</i>	Elicitation of requirements, knowledge sharing- <i>shared understanding of requirements</i>			decision making events at all the levels for synchronous and asynchronous communication	Portfolio level, Domain level, Team level
E-mail	For knowledge sharing- <i>shared understanding of requirements</i>				Least commonly used asynchronous CTs. Employed at Portfolio level, domain level, team level

Conceptual framework of requirements prioritization in SADSD



Ethics & Professionalism

Ethics and Professionalism in Software Engineering

Week 6



Why become aware of Ethical and Professional Responsibilities as a Software Engineer?

Surveillance Technology As a Case

Task for an AI expert and software developer to write a software program

A User story

As a security analyst I want to be able to identify unauthorised items on the desk surface so that I can take appropriate action

Unauthorised Items

Task for an AI expert and software developer to write a software program

A User story

As a security analyst I want to be able to identify unauthorised items on a target surface so that I can take appropriate action

How might we develop this software??

Do you know of some relevant technologies?

Would Yolo be helpful?

<https://pjreddie.com/darknet/yolo/>

“You only look once (YOLO) is a state-of-the-art, real-time object detection system”.

Programming Sensitivities

More specific task for an AI expert and software developer to write a software program

A User story

As a security analyst I want to be able to identify unauthorised items on the desk surface so that I can take appropriate action



Big Tech call center workers face pressure to accept home surveillance

Link to padlet

https://padlet.com/tony_clear/1s4siiuqqcu6fei6

link to NBC news article

<https://www.nbcnews.com/tech/tech-news/big-tech-call-center-workers-face-pressure-accept-home-surveillance-n1276227>

Software Engineering Graduates

In the 2004 software engineering curriculum (Lethbridge et al., 2006), the expectations for a software engineering graduate (as opposed more generally to those from a computer science curriculum) were stated as:

“a software engineering graduate must be able to do the following:

1. Show mastery of the software engineering knowledge and skills necessary to begin practice.
2. Work individually or in a team to develop quality software.
3. Make appropriate trade-offs within the limitations imposed by “cost, time, knowledge, existing systems, and organizations.”
4. **Perform design in one or more domains using software engineering approaches integrating “ethical, social, legal, and economic concerns.”**
5. Demonstrate understanding of and apply current theories, models, and techniques necessary for software engineering.
6. Demonstrate skills such as interpersonal negotiation, effective work habits, leadership, and communication.
7. Learn new models, techniques, and technologies as they emerge.”

Lethbridge, T. C., R. J. Leblanc, J., Sobel, A. E. K., Hilburn, T. B., & Diaz-herrera, J. L. (2006). SE2004: Recommendations for Undergraduate Software Engineering Curricula. *IEEE Software*, 23(6), 19-25. <https://doi.org/10.1109/MS.2006.171>

Intellectual Property and Copyright with SPEED

Finding articles

Anyone can suggest an article to include in the SPEED database.

A Submitter will submit the bibliographic details only (NOT the pdf) of a published study they want to suggest should be included in SPEED (e.g. Title, authors, journal name, year of publication, volume, number, pages, DOI).

It needs to be enough information so the Moderator and Analyst can use AUT library to find the original article. The SPEED app will have a submission form for a Submitter to fill in.

The bibliographic details can be uploaded into the SPEED form ideally in a standard-format file (eg Bibtex format) or the information can be typed in manually, or a combination of these.

The pdf version of the article can NOT be included, for copyright reasons. There can be no link to the article online either, apart from the DOI (Document Object Identifier – a URL) of the article.

Users may have to pay the publisher to get the full article or pay the fees to join a commercial online database such as ACM or IEEEXplore. This is outside the scope of SPEED, however.

Lecture Outline

1. Power and Technology
2. Professional Responsibility
3. Ethics - What & Why
4. Philosophical Ethics
5. Professional Ethics
6. Codes of Ethics
7. Conflict of Professional Responsibility
8. Project Risk & Software Development Impact Statements
9. AI and ethics
10. Further



Computer Ethics



Information Technology – Multiple Discipline Perspectives

- Information Technology (IT) sometimes an umbrella term for the computing disciplines, and a descriptor of the industry in which practitioners work.
- Yet there is little agreement on what the term means.
- Orlowski [7], distinguishes the “IT artifact” under four broad conceptual categories: the *computational* view, the *tool* view, the *proxy* view and the *ensemble* view of IT.
- The *computational* view, of “*technology as algorithm*” underpins the *computer science* discipline.
- [7] Orlowski, W., & Iacono C., Research Commentary: Desperately seeking the “IT” in IT Research – a Call to Theorizing the IT Artifact. *Information Systems Research*, 12:2 (2001), 121-134.



Information Technology – The ‘Ensemble’ View

- The *tool* view of “*technology as labor substitution tool*” and “*technology as productivity tool*”
 - – underpins the **commercial perspective** on IT, and the business rationale for IT industry research and development activities.
- The *proxy* view with “*technology as perception*” or “*technology as diffusion*” is taken up by the **information systems** discipline.
 - – Explorations are conducted into motivations of users, new technology acceptance within organizations, and barriers to the spread of new technologies.
- [7] Orlikowski, W., & Iacono C., Research Commentary: Desperately seeking the “IT” in IT Research – a Call to Theorizing the IT Artifact. *Information Systems Research*, 12:2 (2001), 121-134.



Information Technology – The ‘Ensemble’ View

- The *ensemble* view [7] presents **four** views of Technology
 - It regards “*technology as development project*”
 - – this model in combination with “*technology as algorithm*” could be said to underpin the **software engineering discipline**.
- It also views Technology as ‘*Production Network*’ – a supply side perspective on technology
 - – a global industrial system of technology creation viewed at levels of industry and nation state and collaborative alliances
- [7] Orlikowski, W., & Iacono C., Research Commentary: Desperately seeking the “IT” in IT Research – a Call to Theorizing the IT Artifact. *Information Systems Research*, 12:2 (2001), 121-134.

Information Technology – The ‘Ensemble’ View

- The Technology as “**Embedded System**” perspective is one of an evolving system embedded in a complex and dynamic social context.
- This influenced its introduction and how different user groups engaged with that technology, sometimes called the “**web model**” of Technology
- ‘**Technology as Structure**’ focused on the ways in which technology is enmeshed in its conditions of its use.
- It draws on the theory of ‘structuration’ and the idea that technologies embody set of rules and resources built into the technology during its development and how they are then appropriated by users as they interact with the technology.
- Particular systems and how they evolve into distinctive patterns of use have been studied.
- The “technology in practice” or “technology in use” is observed through interactions in which these patterns are established and reinforced.
- [7] Orlikowski, W., & Iacono C., Research Commentary: Desperately seeking the “IT” in IT Research – a Call to Theorizing the IT Artifact. *Information Systems Research*, 12:2 (2001), 121-134.



Search (Ctrl+E)



Apps

< All teams

E2

ENSE701_S2_2024



Home page

Class Notebook

Classwork

Assignments

Grades

Reflect

Insights

▼ Main Channels

General

Interact with Product Owner

Interact with teaching team

Team Membership

Tutorial

E2

Interact with Prod...

Posts

Files

Notes



+ New

Upload

Edit in grid view



All Documents



Interact with Product Owner > Assgt 1B Iteration 1 Review

	Name	Modified	Modified By
	Collaboration_Worksheets_Iteration_One	6 days ago	Tony Clear
	Timeslots_Iteration_One	6 days ago	Tony Clear
	TimeSlots_Stream_W201.xlsx	July 29	Tony Clear

Appropriation



11:18 am

20/08/2024





Power and Technology

- “A designer of systems, who has the de facto prerogative to specify the range of phenomena that [his] system will distinguish clearly is in possession of enormous degrees of power....
- It is in this sense that computer programmers, the designers of computer equipment, and the developers of computer languages possess power.”
- (Boguslaw, “*The New Utopians*” 1965, p.190 quoted in (Winograd, T. & Flores, F. 1986)).



Power and Technology

- “To the extent that **decisions** made by each of these participants in the design process serve to
 - **reduce,**
 - **limit** or
 - **totally eliminate action alternatives,**
- they are **applying force** and **wielding power** in the precise sociological meaning of these terms”
- (Boguslaw, “*The New Utopians*” 1965, p.190 quoted in (Winograd, T. & Flores, F. 1986)).

Power and Systems

- Systems development an extension of managerial power by the use of an administered system?
- The system as a tool to exert control within the wider instrumentality of the organisation itself, the organisation's goals and the objectives of the project.
- Systems developed as instruments of control often struggle in bridging the gap between design and execution,
- May fail in implementation because of their unacceptability to the intended user community.
- Remember conceptions of IT – algorithm, tool, proxy or ensemble
- *Tool* for control or *Ensemble* reflecting unpredictability in the ways that IT may be appropriated??

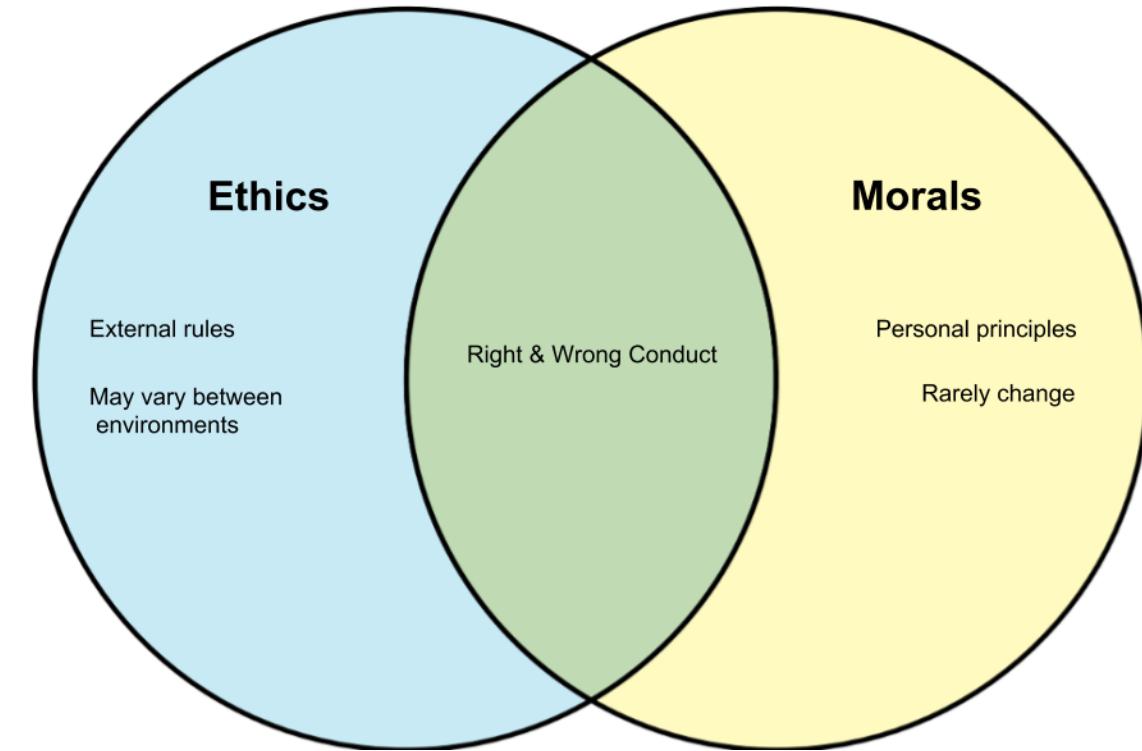
Computing Professionals - Responsibilities

- Computing professionals' actions **change the world**.
- To act responsibly, they should **reflect upon the wider impacts of their work, consistently supporting the public good**.
- The ACM Code of Ethics and Professional Conduct (“the Code”) expresses the **conscience of the profession**.
- <https://ethics.acm.org/>
- <https://www.acm.org/special-interest-groups/sigs/sigcas>

What is ethics?

Defining Terms

- Society:
 - The group of people organized under ordered community
- Morality
 - Division between right and wrong action
- Ethics
 - Is based on standards of what is believed to be right and what is wrong



Teaching Social and Ethical Impact Of Computing

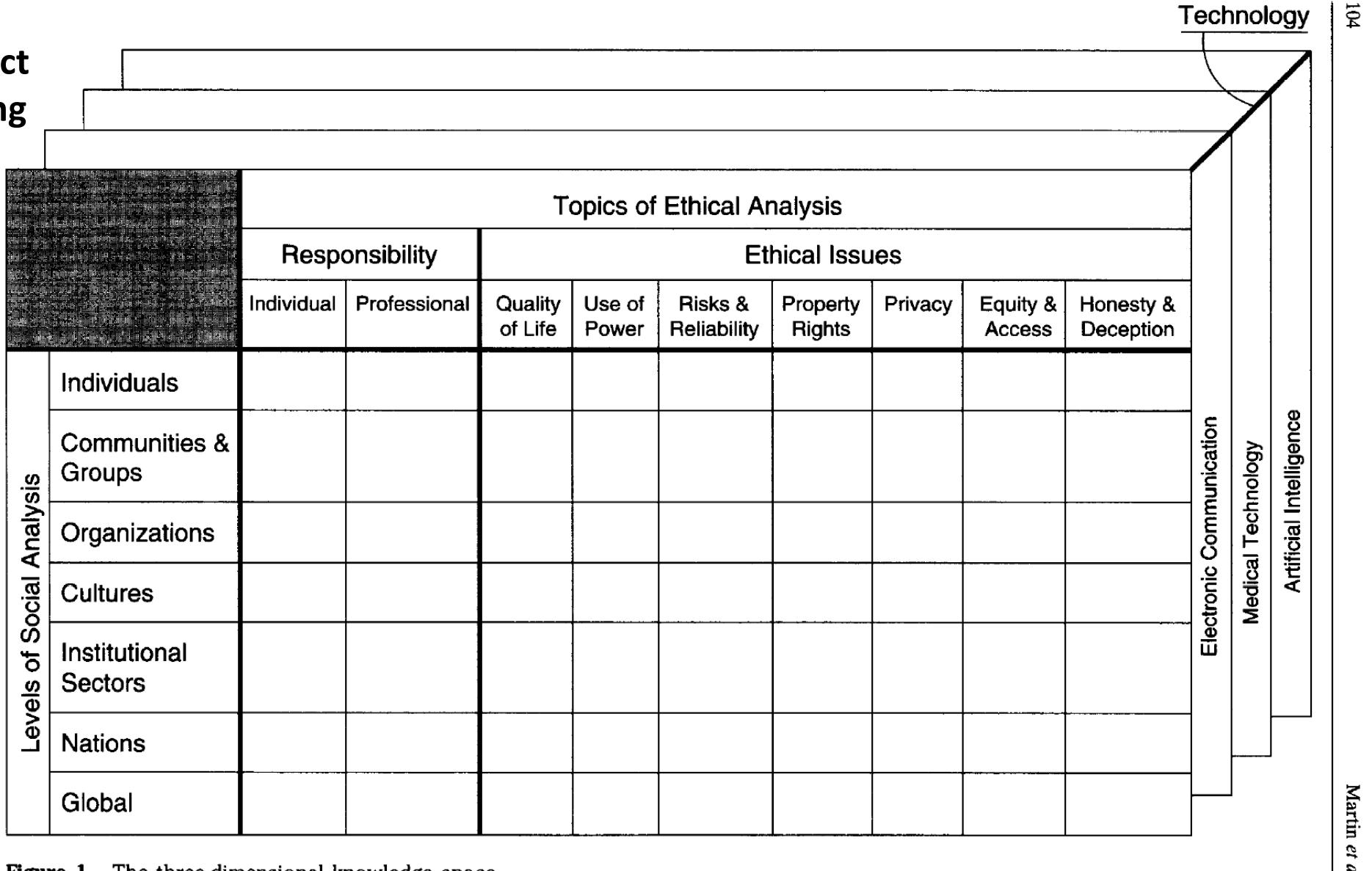


Figure 1. The three-dimensional knowledge space

Martin, C. D., Huff, C., Gotterbarn, D., & Miller, K. (1996). A framework for implementing and teaching the social and ethical impact of computing. *Education and Information Technologies*, 1(2), 101-122.



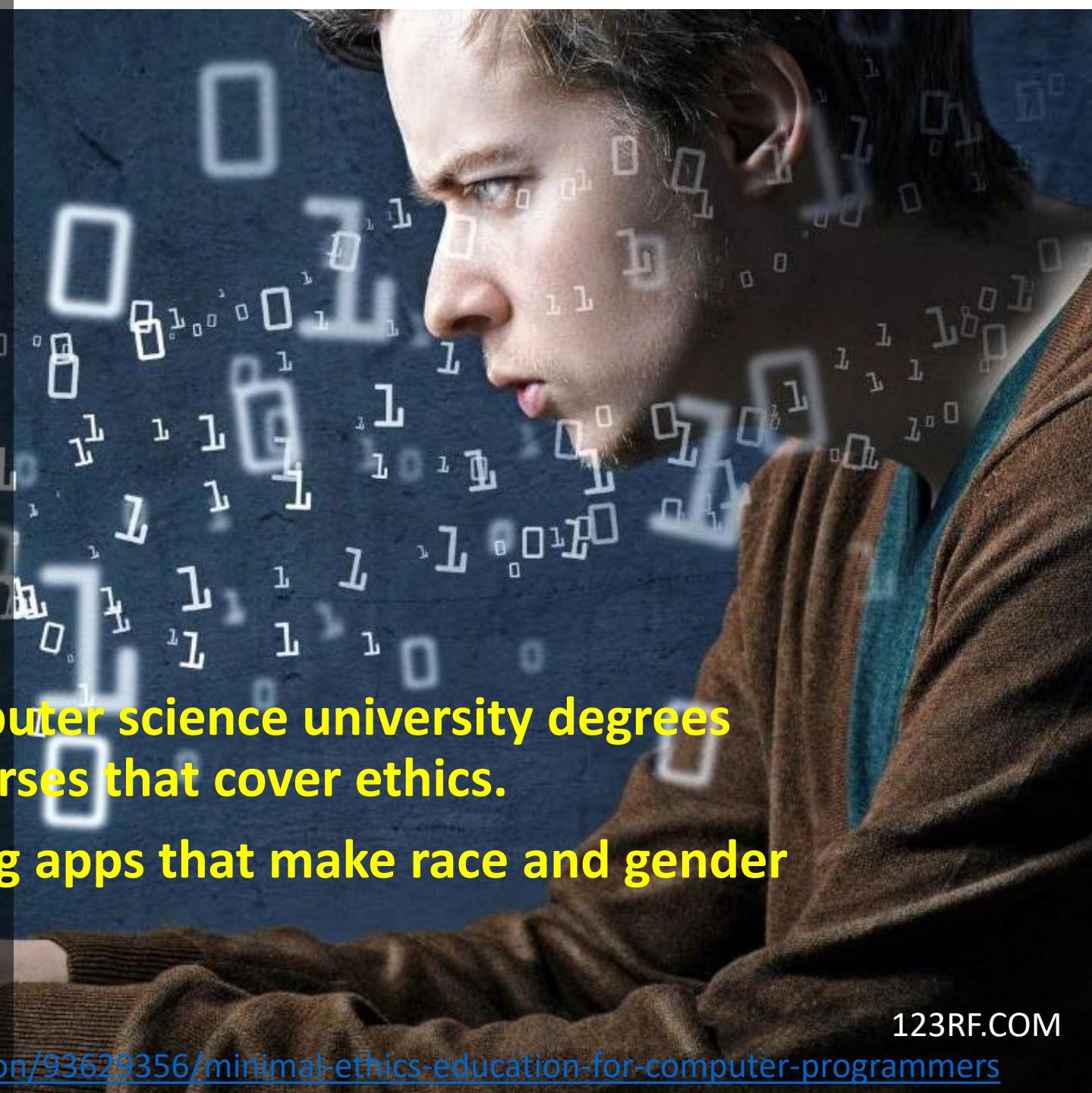
What is ethics?

- *The set of principles about what is right and wrong that individuals use to make choices to guide their decisions* (Stair et al., 2020, p. 68)



Why study ethics?

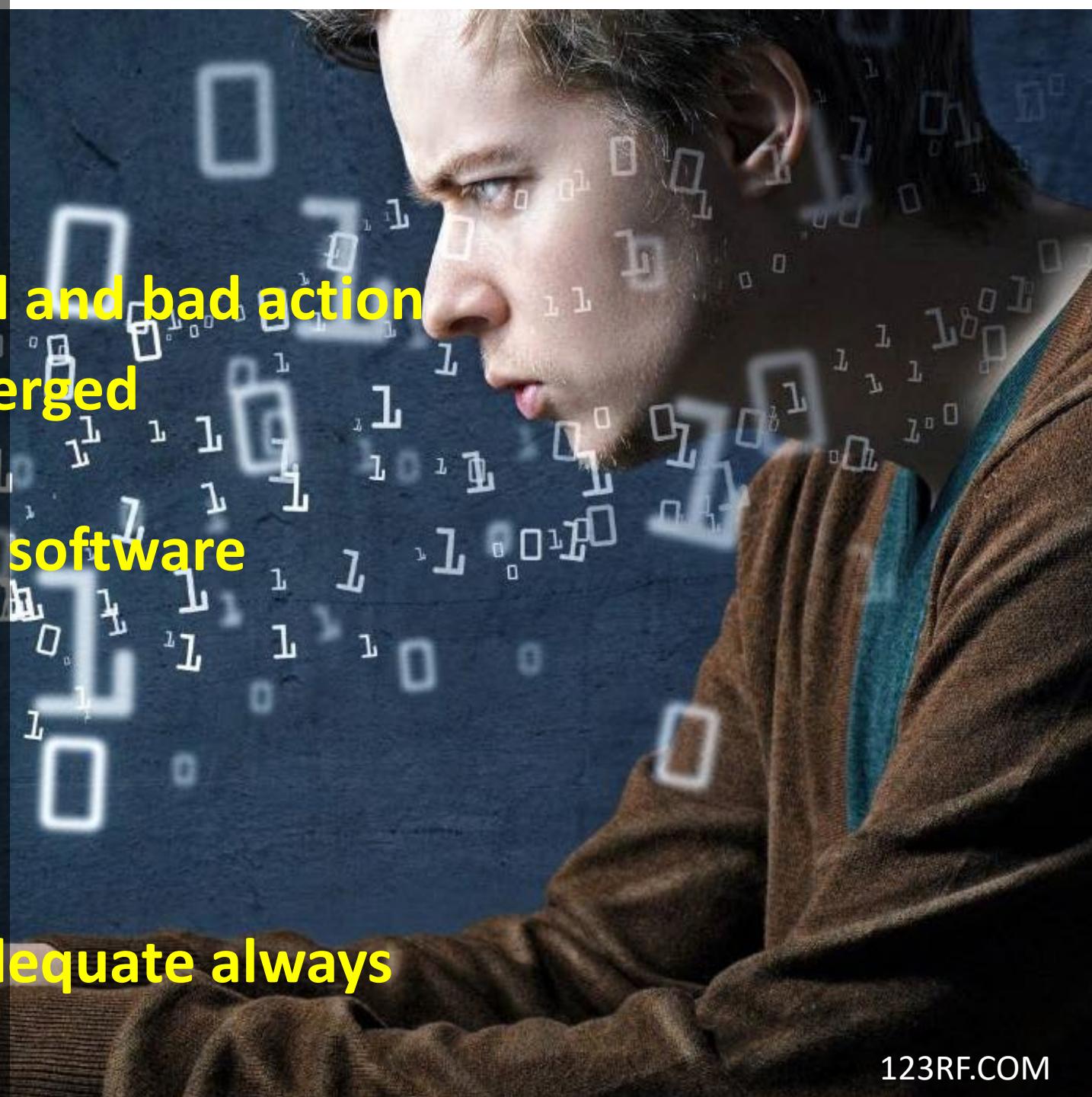
- **Most of New Zealand's computer science university degrees do not include papers or courses that cover ethics.**
- **Some developers are creating apps that make race and gender divides worse*.**





Why study ethics?

- To decide what is a good and bad action
- New problems have emerged
 - Cyberbullying
 - Proprietary nature of software
 - Privacy
 - Spam
 - Identity theft
 - etc.
- Common sense is not adequate always



Why study ethics (cont.)

Scenario 1*

- Megan Meier, a 13-year-old resident of Dardenne Prairie, Missouri, had an account on MySpace where she received a “friend” request from a user named Josh Evans
- Evans, who claimed to be a 16-year-old boy, told Meier that he lived near her and was being home-schooled by his parents
- At first, Evans sent flattering e-mails to Meier, which also suggested that he might be romantically interested in her
- Soon, however, Evans’s remarks turned from compliments to insults, and Evans informed Meier that he was no longer sure that he wanted to be friends with her because he heard that she “wasn’t very nice to her friends”
- Next, Meier noticed that some highly derogatory posts about her—e.g., “Megan Meier is a slut” and “Megan Meier is fat”—began to appear on MySpace
- Meier, who was reported to have suffered from low self-esteem and depression, became increasingly distressed by the online harassment (cyberbullying) being directed at her
- On October 17, 2006, Meier decided to end her life by hanging herself in her bedroom
- An investigation of this incident, following Meier’s death, revealed that Josh Evans was not a teenage boy; she was Lori Drew, the 49-year-old mother of a former friend of Meier’s

*From (Tavani, 2013) and online at <https://abcnews.go.com/GMA/story?id=3882520&page=1>
or: <https://www.youtube.com/watch?v=HFsfDLCKfQU>

Why study ethics (cont.)

Cyberethics as a Unique Kind of Ethics

- Some say yes based on:
 - Computers are malleable
 - Computer malleability creates “Policy Vacuums”
 - Accordingly computer ethics is a way to
 - analyze these policy vacuums
 - and to formulate appropriate policies
- Some say no:
 - The principles of ethics are relatively constant
 - The principles of medical ethics, legal ethics, computer ethics, etc. are the same
 - There does not seem to be sufficient evidence to substantiate the claim that one or more new ethical issues have been introduced

Policy Vacuums: absence of policies for dealing with new possibilities in the information technology

The distinction between law and ethics

- An assumption is always made that what is ethical is also what is legal
- What is unethical is illegal
- However, it is possible for an act to be ethical but illegal
- Or legal but unethical
- Technology opportunities can outpace the legal framework
- Uber drivers - contractors or employees? - 12/02/2021 – **Contractor in NZ**
- <https://www.heskethhenry.co.nz/insights-opinion/employment-court-deems-uber-driver-a-contractor/>
- The New Zealand Employment Court has dealt Uber a significant blow in a case taken by four of its drivers, **ruling that they are employees – not contractors.** 27/08/2022
- <https://chapmantripp.com/trends-insights/uber-drivers-found-to-be-employees/>
- Uber drivers - contractors or employees? - 16/03/2021 – **Employee in UK**
- <https://www.heskethhenry.co.nz/insights-opinion/uk-supreme-court-delivers-decision-on-uber-driver-employment-status/>
- <https://medium.com/swlh/uber-and-lyfts-business-model-is-not-a-business-model-13c1433dd00c>

Ethics and law are **not the same thing**

	Legal	Illegal
Ethical	?	?
Unethical	?	?

Development of computing ethics

Cyberethics evolution can be divided into four phases

Time period	Technological Features	Associated Issues
1950s-1960s	Stand alone machine	Artificial Intelligence (AI), database privacy
1970s-1980s	Minicomputers & the ARPANET; desktop computer interconnected via privately owned networks	Intellectual property and software piracy, computer crime, and communications privacy
1990s-present	Internet, World Wide Web, and early “Web.2.0” applications, environments, and forums	Free speech, anonymity, legal jurisdiction, behavioral norms in virtual communities
Present to near future	Convergence of information and communication technologies with nanotechnology and biotechnology; increasing use of autonomous systems	Artificial intelligence electronic agents (“bots”) with decision-making capabilities, and developments in nanocomputing, bioinformatics, and ambient intelligence

Philosophical Ethics (cont.)

Ethical Theories

- There are many ethical theories
- However, we consider only a few that are most widely discussed and used
 - Consequentialism
 - Egoism
 - Utilitarianism
 - Altruism
 - Deontology
 - Human nature
 - Relativism
 - Hedonism
 - Eudaimonism <https://www.youtube.com/watch?v=VFPBf1AZOQg>
 - Emotivism

Philosophical Ethics (cont.)

Ethical Mindset

- Plato and the Good Life
- https://www.youtube.com/watch?v=-oJs5u_GAYA&list=RDLVVFPBf1AZOQg&index=3
- *“The unexamined life is not worth living”*

Eudaimonism



Cajander, Å. (2010). *Usability - Who Cares?: The Introduction of User-Centred Systems Design in Organisations*. Unpublished PhD Thesis Uppsala University, Uppsala, Sweden <http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-122387>

Central image - employee, exuding contented, harmonious and positive work experience.

Smaller images, values affecting work as experienced today.

The clock and money images represent values overshadowing all others when it came to forces governing the work of the employees in our research.

The scales indicate the necessity to strive for a better balance between these important aspects of time efficiency and financial results in an organisation, and other essential aspects and conditions that will affect how well the individual, as well as the organisation as a whole, will perform.

Among these latter aspects are experiencing joy, a sense of belonging, and a meaningful, rewarding, healthy and balanced work situation. [p.13-14]

Philosophical Ethics (cont.)

Professor Casey – should Batman kill the joker?

-
- <https://www.instagram.com/tv/CT0epClhQRO/>

Philosophical Ethics (cont.)

Consequentialism

- Consequentialism: actions are judged either good or bad, right or wrong
 - Egoism: puts an individual's interests and happiness above everything else
 - Utilitarianism: an action is right if it results in the greatest benefit of the greatest number of people
 - Altruism: an action is right if the consequences of that action are favourable to all except the actor

Philosophical Ethics (cont.)

Deontology

- Does not concern itself with the consequences of the action
- It is concerned with the will of the action.
- An action is good or bad depending on the will inherent in it.

Moral dilemmas and “crowd ethics”

Ask an audience to decide the ethics for intelligent autonomous machines

- Dilemma: decision making problem between two moral choices, neither of which is ideal
- <http://moralmachine.mit.edu>

Professional Ethics

What is a Profession?

- Before we delve into professional ethics, it is useful first to understand what is meant by “profession” and “professional”
- A profession is an area in which one professes
- Requirement of a profession*
 - Expert knowledge: special technical knowledge that is certified by some authority
 - Autonomy: a professional has power over how the service is provided
 - Observance of a code of conduct
- **Professionals often make decisions that can have significant social effects**

To profess is to make a public declaration, a claim of something.

*From (Tavani, 2013)

Professional Ethics (cont.)

Code of Ethics and Professional Conduct

- Many professionals have established a professional society
- A professional society adopts codes of conduct
- Examples
 - Medical profession → New Zealand Medical Association (NZMA)
 - Engineering profession → Registered Engineering Associates (REA)
 - Legal profession → New Zealand Law Society (NZLS)
- All the above professions have formal codes of ethics/conduct
- Computer/IT professionals in New Zealand also established
 - IT Professionals New Zealand
- Computer/IT professionals not in New Zealand – Global Societies
 - Association for computing machinery (ACM)
 - Institute for Electrical and Electronics Engineers-Computer Society (IEEE-CS)

Professional Ethics – 8 Tenets as Essence of ITP Code of Ethics



Good faith



Integrity



Community Focus



Skills



Continuous
Development



Informed Consent



Managed Conflicts
of Interest



Competence

IEEE-CS Software Engineering Code of Ethics

- “Each principle of this code addresses **three levels of ethical obligations** owed by professional software engineers in each of these relationships.
- The **first level** identified is a **set of ethical values**, which professional **software engineers share** with all other human beings **by virtue of their humanity**.
- The **second level** obliges software engineering professionals to **more challenging obligations** than those required at Level One. Level Two **obligations** are required because **professionals owe special care to people who may be affected by their work**.
- The **third and deeper level** comprises several **obligations** that derive directly from **elements unique to the professional practice of software engineering**.”
- Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. *Communications of the ACM*, 40(11), 110-118.

IEEE-CS SE Code of Ethics (cont.)

- ***Level One: Aspire (to be human).*** Statements of aspiration provide vision and objectives, and are intended to direct professional behavior. These directives require significant ethical judgment.
- ***Level Two: Expect (to be professional).*** Statements of expectation express the obligations of all professionals and professional attitudes. Again, they do not describe the specific behavior details, but they clearly indicate professional responsibilities in computing.
- ***Level Three: Demand (to use good practices).*** Statements of demand assert more specific behavioral responsibilities within software engineering, which are more closely related to the current state of the art. The range of statements is from the more general aspirational statement to specific measurable requirements.

IEEE-CS SE Code of Ethics at a Glance

CODE OF ETHICS AT A GLANCE

The short version of the Code summarizes aspirations at a high level of abstraction. The clauses that are included in the full version give examples and detail of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive Code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing, and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety, and welfare of the public, software engineers shall adhere to the following Principles:

1. **Public:** Software engineers shall act consistently with the public interest.
2. **Client and Employer:** Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. **Product:** Software engineers shall ensure their products and related modifications meet the highest professional standards possible.
4. **Judgment:** Software engineers shall maintain integrity and independence in their professional judgment.
5. **Management:** Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. **Profession:** Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. **Colleagues:** Software engineers shall be fair to and supportive of their colleagues.
8. **Self:** Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

IEEE-CS SE Code of Ethics - Preamble

- “The Preamble to the Code was significantly revised.
- It includes specific ethical standards to help the professional make ethical decisions.
- The Code emphasizes the professional’s **obligations to the public at large**.
- This obligation is the final arbiter in all decisions. “In all these judgements, **concern for the health, safety, and welfare of the public is primary; that is, the ‘Public Interest’ is central to this Code.**”
- The **primacy of well being and quality of life of the public**, in all decisions related to software engineering, is **emphasized throughout the Code**”.

Gotterbarn, D., Miller, K., & Rogerson, S. (1999). Computer society and ACM approve software engineering code of ethics. *Computer*, 32(10), 84-88.

IEEE-CS SE Code of Ethics – Ethical Tensions

- “**Ethical tensions** can best be addressed by **thoughtful consideration of fundamental Principles**, rather than blind reliance on detailed regulations.
- These Principles should influence software engineers to consider broadly **who is affected by their work**;
- to examine if they and their colleagues are **treating other human beings with due respect**;
- to consider **how the public**, if reasonably well informed, **would view their decisions**;
- to analyze **how the least empowered will be affected** by their decisions;
- and to consider whether their **acts** would be **judged worthy of the ideal professional working as a software engineer**.
- **In all these judgments concern for the health, safety, and welfare of the public is primary; that is, the *public interest* is central to this Code”.**

Gotterbarn, D., Miller, K., & Rogerson, S. (1999). Computer society and ACM approve software engineering code of ethics. *Computer*, 32(10), 84-88.

ACM Code of Ethics – General Ethical Principles

- 1. GENERAL ETHICAL PRINCIPLES.
- *A computing professional should...*
- 1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
- 1.2 Avoid harm.
- 1.3 Be honest and trustworthy.
- 1.4 Be fair and take action not to discriminate.
- 1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
- 1.6 Respect privacy.
- 1.7 Honor confidentiality.

ACM Code of Ethics – Professional Responsibilities

- 2. PROFESSIONAL RESPONSIBILITIES.
- *A computing professional should...*
- 2.1 Strive to achieve high quality in both the processes and products of professional work.
- 2.2 Maintain high standards of professional competence, conduct, and ethical practice.
- 2.3 Know and respect existing rules pertaining to professional work.
- 2.4 Accept and provide appropriate professional review.
- 2.5 Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.
- 2.6 Perform work only in areas of competence.
- 2.7 Foster public awareness and understanding of computing, related technologies, and their consequences.
- 2.8 Access computing and communication resources only when authorized or when compelled by the public good.
- 2.9 Design and implement systems that are robustly and usably secure.



ACM Code of Ethics - Leadership

- 3. PROFESSIONAL LEADERSHIP PRINCIPLES.
- Leadership may either be a formal designation or arise informally from influence over others. In this section, “leader” means any member of an organization or group who has influence, educational responsibilities, or managerial responsibilities. While these principles apply to all computing professionals, leaders bear a heightened responsibility to uphold and promote them, both within and through their organizations.

A computing professional, especially one acting as a leader, should...

- 3.1 Ensure that the public good is the central concern during all professional computing work.
- 3.2 Articulate, encourage acceptance of, and evaluate fulfillment of social responsibilities by members of the organization or group
- 3.3 Manage personnel and resources to enhance the quality of working life.
- 3.4 Articulate, apply, and support policies and processes that reflect the principles of the Code.
- 3.5 Create opportunities for members of the organization or group to grow as professionals.
- 3.6 Use care when modifying or retiring systems.
- 3.7 Recognize and take special care of systems that become integrated into the infrastructure of society.

Professional Ethics (cont.)

Strength and Weaknesses of Professional Codes (Tavani 2013)

Strengths	Weaknesses
Codes inspire the members of a profession to behave ethically.	Codes include directives that tend to be too general and too vague.
Codes guide the members of a profession in ethical choices.	Codes are not always helpful when two or more directives conflict.
Codes educate the members about their professional obligations.	Codes comprise directives that are neither complete nor exhaustive.
Codes discipline members when they violate one or more directives.	Codes are ineffective (have no “teeth”) in disciplinary matters.
Codes inform the public about the nature and roles of the profession.	Codes sometimes include directives that are inconsistent with one another.
Codes “sensitize” members of a profession to ethical issues and alert them to ethical aspects they otherwise might overlook.	Codes do not always distinguish between microethics issues and macroethics issues.
Codes enhance the profession in the eyes of the public.	Codes can be self-serving for the profession.

Conflict of Professional Responsibility

- Whistle-blower: is a person who exposes any kind of information or activity that is deemed illegal, unethical, or not correct within an organization that is either private or public
- Here we will discuss the following questions
 - Do employee and employers have a special obligation of loyalty to each other?
 - Should loyalty to one's employer ever preclude an employee's "blowing the whistle" in critical situations?



Source*

* Whistleblowing law keeps media out of the loop <https://www.rnz.co.nz/national/programmes/mediawatch/audio/2018735702/whistleblowing-law-keeps-media-out-of-the-loop>

Conflict of Professional Responsibility

- Consider Challenger disaster in January 1986, which resulted in the deaths of the seven crew members
- Engineers who designed the space shuttle were aware of the safety risks in launching the shuttle in cooler temperatures
- Some engineers, when learning the Challenger was scheduled for launch on a cool January morning, went to their supervisors to express their concerns
- However, a decision was made to stick with the original launch date
- Having received no support from their supervisors, should those engineers have gone directly to the press?

Would whistle-blowing at that level have saved the lives of the Challenger's crew?

NZ issue and CAA: <https://www.nzherald.co.nz/nz/whistleblowers-warn-caas-new-approach-could-lead-to-more-aviation-accidents-in-new-zealand/HAXQ74ITMNBHLCLGGBKGN2OI/>

Conflict of Professional Responsibility (cont.)

- Can professional codes guide engineers in whistle-blowing decisions?
 - Does the IT Professional NZ code of ethics answer the question?
- Section 6.12 and 6.13 of SECEPP state
 - express concerns to the people involved when significant violations of this Code are detected unless this is impossible, counterproductive, or dangerous;
 - report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counterproductive, or dangerous.
- They are still too vague ?

Extending Codes of Ethics to Risk Assessment? A Bi-Cultural Project Context

- In mid-2002, a number of students at Auckland University of Technology (AUT) began work on a **project to extend the existing IT systems of a Maori tribal authority, Te Runanga a Iwi o Ngapuhi (TRAION)**, the statutory body representing the Ngapuhi tribe, or *iwi* (Clear et al., 2004).
- Proposed changes in a broadly conceived project included the following:
 - Online registration of tribal members
 - Linking members to several groupings of significance to Maori • Extended family (*whanau*) • Subtribe (*hapu*)
 - *Marae* (a meeting-house complex used for several cultural purposes and serving the Maori community centered in that location)
 - Creating a database of genealogical (*whakapapa*) information
 - Creating a database of interests in communally owned tribal land

Gotterbarn, D., Clear, T., Gray, W., & Houlston, B. (2006). Developing Software in a Bicultural Context: The Role of a SoDIS® Inspection. *International Journal of Technology and Human Interaction*, 2(2), 1-21.



Extending Codes of Ethics to Risk Assessment? A Bi-Cultural Project Context (Cont'd)

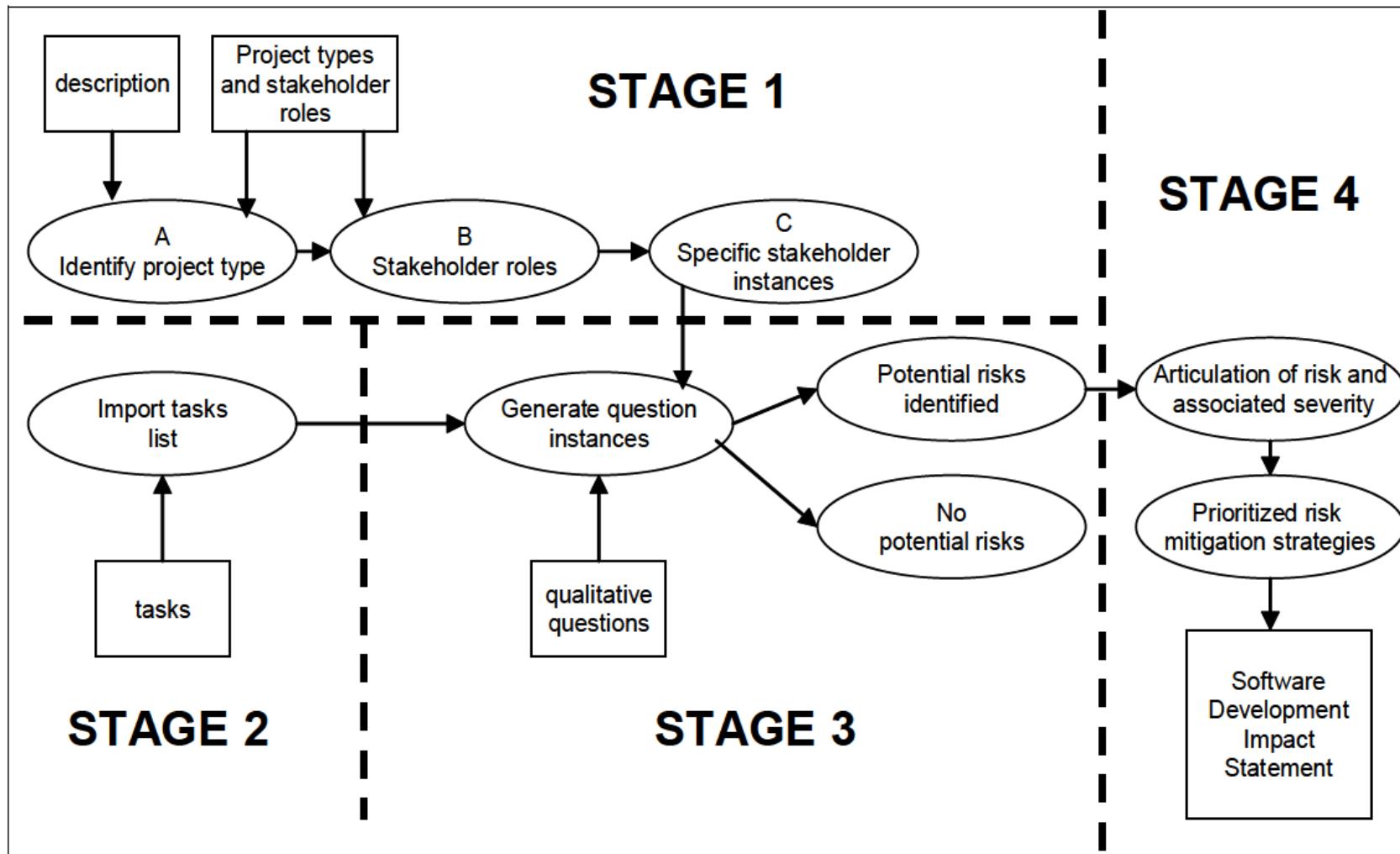
- The TRAION project, then, was entering into sensitive areas. For Maori, “**Identity and worth were found in family and tribal connectedness** [and] ...identity was linked to both ancestry and place” (King, 2003, p. 77).
- As a consequence, Maori people have **known sensitivities about research related to *whakapapa*** (genealogical and land-based information), which is considered a *taonga* (treasured possession) particular to
 - the groupings (*whanau*, *hapu*, and *iwi*) who have interests in this information.
- To better articulate the risks and to investigate the issues inherent in computerizing such sensitive information,
- a **Software Development Impact Statement (SoDIS) analysis**
- was undertaken.



Gotterbarn, D., Clear, T., Gray, W., & Houlston, B. (2006). Developing Software in a Bicultural Context: The Role of a SoDIS® Inspection. *International Journal of Technology and Human Interaction*, 2(2), 1-21.

SoDIS Project TRAION Risk Assessment?

Figure 1. SoDIS® process (Gotterbarn & Rogerson, 2005)



Gotterbarn, D., Clear, T., Gray, W., & Houlston, B. (2006). Developing Software in a Bicultural Context: The Role of a SoDIS® Inspection. *International Journal of Technology and Human Interaction*, 2(2), 1-21.

TRAION SoDIS Inspection: Outcomes

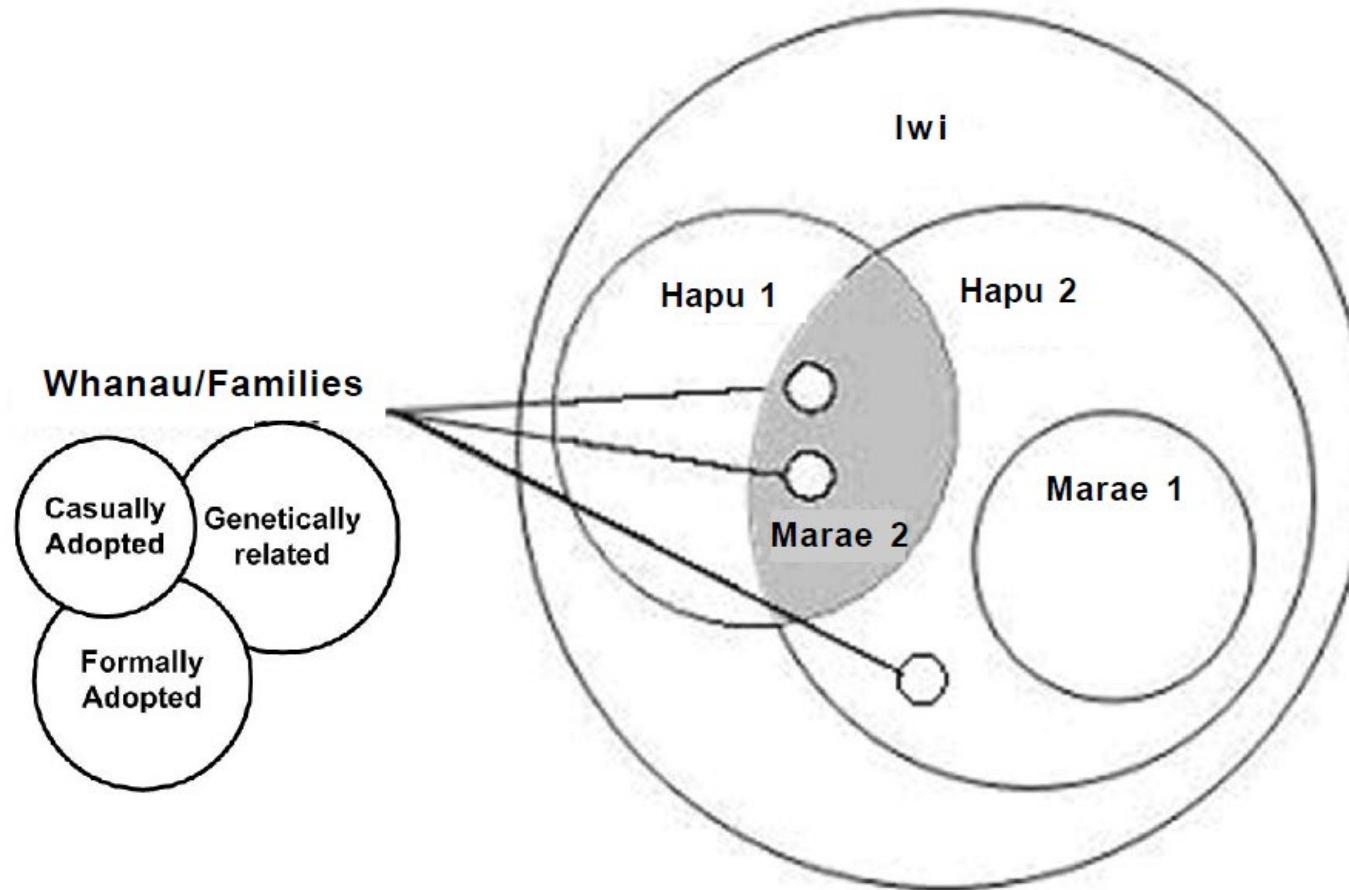
- In the **first cluster of *client developer communication***, the key Maori ethical questions related to **who could legitimately represent the *iwi* as the project client**. ...historical situations in which the authority of the chiefs had been subverted and the authority structures of Maori society undermined ... when engaging in research and development with Maori (Bishop, 1996), **the processes of initiation** and the **need to work through due tribal and group decision making and authority structures** are critical.
- In the **second cluster of *sensitive data***, several Maori ethical questions arose. **Privacy concerns** and mechanisms for **obtaining consent** for provision of data for genealogical research purposes raised complex questions of **who could legitimately view what data**.
- The collective ownership of *whakapapa* at different levels meant that **group and individual access rights had to be negotiated**. Individual data were personal, but ***whanau* data were the property of the family group to decide**, and *hapu* and *iwi* had their own interests and group decision-making processes in order to determine these rights.
- For instance, ***what rights would system administrators, data entry clerks, and Runanga management have to access or restrict access to this data?*** These policies and authorization protocols would need to be developed and agreed upon through accepted tribal decision-making processes.

TRAION SoDIS Inspection: Outcomes (Cont'd.)

- Similarly, **protocols concerning display of cultural artifacts over the Web or use of whakapapa information for commercial purposes** (e.g., to defray expenses of the site or to support storage and research costs) would need to be agreed upon at the tribal level in order to offset concerns over commercialization and inappropriate use of treasured information and sacred objects.
- **Data integrity and the need to preserve the very authenticity of whakapapa as a stakeholder in its own right** has been noted as a key Maori concern.
- In the **third cluster, user experience**, several more Maori ethical questions arose. Again, **questions over authority in disputed circumstances would need to be settled** (i.e., who could determine official groupings and their standing?). For instance, a particular Northern grouping, Ngati Hine, claims *iwi* status but has been deemed by Te Ohu Kai Moana as a *hapu*. **How do such determinations hold standing, who decides official lists of hapu and iwi, and how are dissenting voices to be registered?** Likewise, under what criteria are membership applications to be refused registration and what is the impact for those refused? What authority will systems administrators and Runanga clerks possess, and **what controls will be in place to ensure the integrity of data entered and stored?** How is the integrity of *whakapapa* to be maintained in each of these circumstances?

Who is Legitimately a Member?

Figure 2. Iwi relationship structures (adapted from Clear et al., 2004)



Extending Codes of Ethics to Risk Assessment? - Baptist Action Trust

- The qualitative risk analysis (SoDIS inspection) was commissioned in early December 2004 by Stuart Simpson of Eagle Technology, on behalf of their client Baptist Action Trust (BAT).
- The SoDIS inspection was conducted by AUT researchers, in conjunction with Stuart Simpson (a co-author of this paper), in a form of “collaborative practice research” (Mathiassen, 2002).
- The system was to automate Homecare rosters,
- time and travel payments for Homecare workers,
- the billing of Homecare clients,
- rostering of Healthcare workers at two Hospitals and a Rest Home,
- wage payments to staff at these sites by timesheet entry and electronically transmitted to the Datacom payroll processing Bureau,
- other client billing,
- and the transfer of General Ledger transactions to the BAT corporate system.

Baptist Action Trust - SoDIS Inspection?

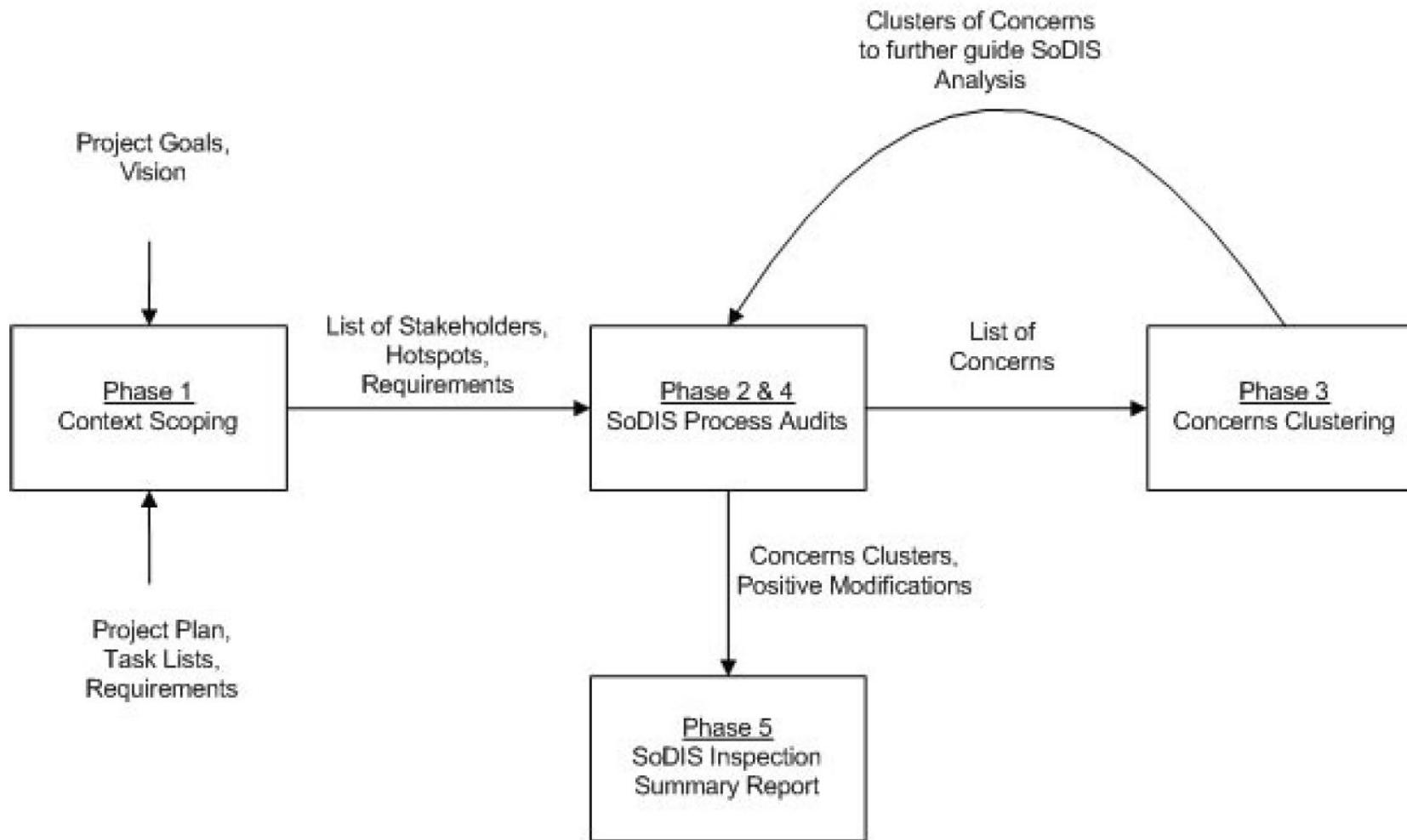


Figure 1. The SoDIS Inspection Process

Kwan, C., Hitchcock, L., Clear, T., Gotterbarn, D., & Simpson, S. (2005). Refining the SoDIS® Process in the Field: A COTS Project as a Context for Risk Analysis. In S. Mann & T. Clear (Eds.), *18th Annual NACCQ Conference* (pp. 25-32). Tauranga, New Zealand: NACCQ.

BAT SoDIS Inspection: Outcomes

- The team had identified **16 critical concerns, 106 significant concerns, and 2 minor concerns** for the project to take into account. A few of these concerns were more in the nature of questions where the team lacked local knowledge to make a judgment as a site visit was not undertaken.
- The team categorised the concerns into four main clusters as follows:
 - **Overall project cluster.** For example; issues concerning the overall project implementation and the needs of all stakeholders; issues that may result in Supplier, Consultant, or Developer intervention; issues to do with clarity of the scope of project goals; issues that may cause confusion to all stakeholders; and issues that may cause support service intervention, additional installation and processing costs, or additional work to stabilise business processes in the event of project breakdown.
 - **Administrative, legal or regulatory cluster.** Such as; issues concerning administrative processes, legal requirements, and conformance to regulations and professional standards; issues that may cause potential loss of control of operation and service; issues that may cause significant overtime and expenditure; and issues that may cause conflicts and inaccuracies in time rosters and time payment errors.
 - **Data security, privacy and accuracy cluster.** Issues concerning security, privacy, and accuracy of data within both data storage and data transmission, and issues concerning data integrity and reconciliation and the possible resultant downstream effects.
 - **Quality of end user service delivery cluster.** Issues relating to interruptions to or degradation of service delivery caused by possible conflicts and contradictions within the proposed solution and its implementation, and issues relating to user dynamics, professional responsibility, and the critical nature of service delivery to BAT clients.
- <http://citrenz.ac.nz/conferences/2005/papers/choon.pdf>

BAT SoDIS Inspection: Outcomes

- **5.3 SoDIS Inspection Outcome - Recommendations**
- Where the AUT risk assessment team were able to derive solutions for concerns, or positive modification suggestions, these were presented in the report. For example:
 - Ensuring a managed data conversion process with careful plans for checking data accuracy and completeness
 - Ensuring that adequate security protocols are in place and that the technology supports BAT policies and procedures
 - A clear procedure for off-line adjustments to the automated business processes and ensuring total accuracy within both automated and off-line processes
 - Confirmation that the application meets regulatory constraints and will detect clashes in the rostering
 - **Ensuring that business processes are designed to complement automated systems (and vice-versa), and the change process is adequately managed**

AI ethics

Asimov's Three Laws

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

The Three Laws, quoted as being from the "Handbook of Robotics, 56th Edition, 2058 A.D. introduced in Asimov's 1942 short story "Runaround" in the March 1942 issue of *Astounding Science Fiction*.





IMPLICATIONS FOR SOFTWARE PRACTITIONERS:

- Good reviews of Generative AI technology and its implications:
 - Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. *IEEE Software*, 40(4), 30-38.
 - Ozkaya, I. (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, 40(3), 4-8. <https://doi.org/10.1109/MS.2023.3248401>

RESPONSIBILITIES FOR SOFTWARE PRACTITIONERS:

What will determine if the next phase includes innovations beyond our imagination or another AI winter is largely dependent on **not our ability to continue technical innovations**, but on our ability to practice software engineering and computer science through the highest level of ethics and responsible practices. We need to be bold in experimenting with the potential of LLMs in improving software development, and we need to be cautious and not forget fundamentals of engineering ethics and rigor.

Ozkaya, I. (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, 40(3), 4-8.
<https://doi.org/10.1109/MS.2023.3248401>

AI ethics

Intelligent Autonomous Machines (IAMs)

A major problem is that, currently, **it is not known how best to ensure that IAMs make the right decisions.**

What is to be the basis of their moral or ethical fabric?





Main problem with programming or hardwiring ethics into IAMs

64

- Hardwired or programmed IAMs mean they can only do what they are programmed or hardwired or told to do
- But intelligent human ethical behaviour seems to invoke free will and choice
- **Result: we no longer have an IAM**
- In any case, such hardwiring or programming does not help IAMs when faced with a dilemma
- **Result: we no longer have an IAM**



(Autonomous Car n.d.)

How do we give IAMs moral principles and make them ethical?

- Use Machine Learning (ML) algorithms to teach moral concepts to intelligent autonomous machines by using human moral behaviour
- In 2016, the Microsoft Twitter chatbot *Tay* produced racist and genocidal speech in less than 24 hours
[https://www.theguardian.com/world/2016/mar/29/microsoft-tay-tweets-antisemitic-racism](https://www.theguardian.com/world/2016/mar/29/microsoft-tay-tweetsantisemitic-racism)
- ML has other examples of discrimination against people on the basis of race and gender*
- Teaching ethics to IAMs can lead to learning the worst of human ethical behaviour

*Caliskan-Islam A, Bryson JJ, Narayanan, A (2016) Semantics derived automatically from language corpora necessarily contains human biases. https://www.princeton.edu/~aylinc/papers/caliskan-islam_semantics.pdf

The Need for New Ethical Principles for Autonomous Systems?

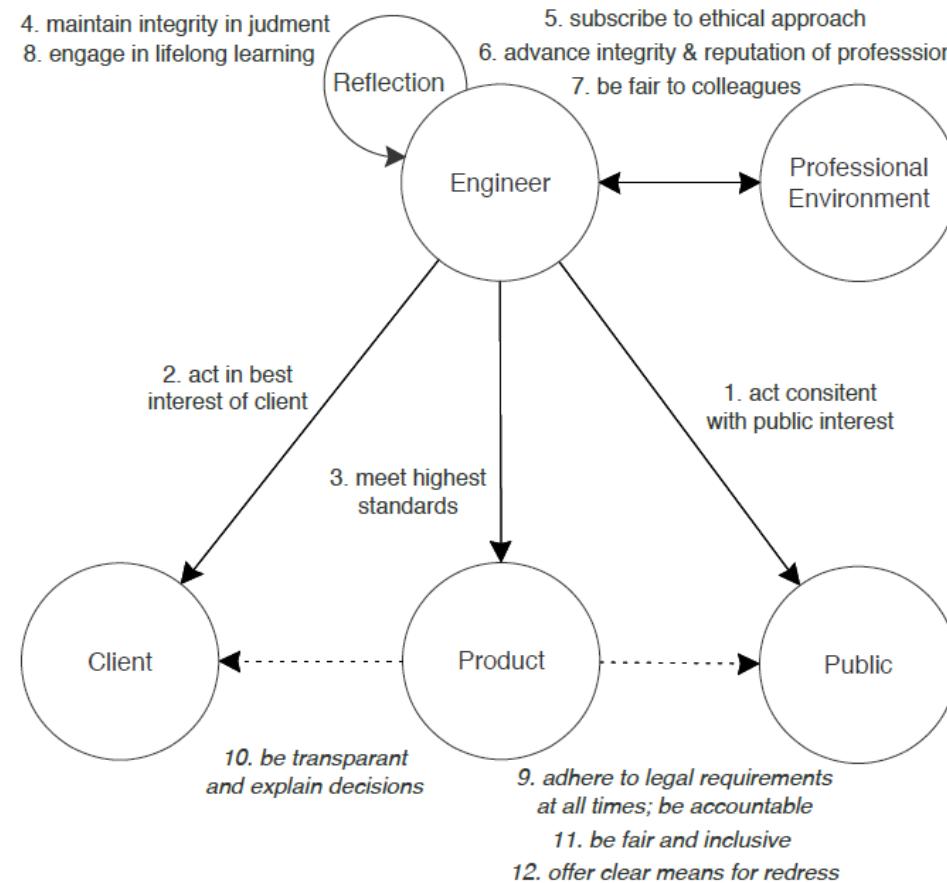


Figure 1: Schematic overview of the Code of Ethics of IEEE/ACM; circles represent actors, arrows represent ethical principles, and dotted arrows suggest new ethical principles for autonomous systems, such as self-adaptive systems.

Weyns, D. (2020). Towards a code of ethics for autonomous and self-adaptive systems. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 163-165)

The Software
Engineering
Code of Ethics



Ethics and Professionalism in Software Engineering

Associate Professor Tony Clear

Department of Computer Science and Software
Engineering,

Auckland University of Technology

Tony.clear@aut.ac.nz



References

- Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. *Communications of the ACM*, 40(11), 110-118.
- Gotterbarn, D., Miller, K., & Rogerson, S. (1999). Computer society and ACM approve software engineering code of ethics. *Computer*, 32(10), 84-88.
- Herman T. Tavani. "Ethics & Technology, controversies, questions, and strategies for ethics in computing" Fourth Edition 2013 , ISBN 9781118281727
- Joseph Migga Kizza. "Ethical and Social Issues in the Information Age" Sixth Edition, ISBN 9783319707129
- Justin, M. (2017, September 21). *Ethical Considerations in Autonomous-System Design*. Retrieved from <https://www.electronicdesign.com/automotive/ethical-considerations-autonomous-system-design>
- Martin, C. D., Huff, C., Gotterbarn, D., & Miller, K. (1996). A framework for implementing and teaching the social and ethical impact of computing. *Education and Information Technologies*, 1(2), 101-122.
- Autonomous Car [Digital Image]. (n.d.). Retrieved March 3, 2019 from <https://www.electronicdesign.com/automotive/ethical-considerations-autonomous-system-design>
- Autonomous Drone [Digital Image]. (n.d.). Retrieved March 3, 2019 from <https://www.electronicdesign.com/automotive/ethical-considerations-autonomous-system-design>
- Fig. 2.2. (2017) *An analogy explaining the difference between ethics and morality*. Imagine society as a town, [Figure]. From Ethics for the Information Age (pg. 51), by Michael J. Quinn., 2017, New York: Harry N. Abrams.
- Stair, R., Reynolds, G., Bryant, J., Frydenberg, M., Greenberg, H., & Schell, G. (2020). *Principles of information systems*. Boston, USA: Cengage Learning.

Further Questions for Consideration

Associate Professor Tony Clear

Department of Computer Science and Software Engineering,

Auckland University of Technology

Tony.clear@aut.ac.nz





Do Some Computer Corporations Have Special Moral Obligations?

Search Engine Companies

- Search engines should shoulder social responsibility because they
 - provide access to information that is crucial for responsible citizenship
 - are now central to education
 - are owned by private corporations—i.e., by businesses that are mostly interested in making a profit
- A small case from education...
- Clear, T. (2006). Google™ - "Do No Evil" - Yeah Right! *SIGCSE Bulletin*, 38(4), 8-10.

thesis - Google Search - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Mail Print Stop Refresh Address http://www.google.com/search?q=thesis&hl=en&sa=N&tab=iw

Sign in

Google Web Images Video News Maps more »

thesis Search Advanced Search Preferences

Results 1 - 10 of about 87,600,000 for thesis [definition]. (0.16 seconds)

Web

Want Thesis Help?
www.originalthesiswriting.com 100% PhD writers. APA, MLA & other formats. Installment payment option Sponsored Link

Thesis Statements
How to Tell a Strong Thesis Statement from a Weak One. How to Generate a Thesis Statement if the Topic is Assigned. Almost all assignments, no matter how ...
www.indiana.edu/~wts/pamphlets/thesis_statement.shtml - 18k - Cached - Similar pages

LEO Thesis Statement
A thesis statement in an essay is a sentence that explicitly identifies ... A thesis statement is an assertion, not a statement of fact or an observation. ...
leo.stcloudstate.edu/acadwrite/thesistatement.html - 5k - Cached - Similar pages

What is a thesis?
A thesis statement declares what you believe and what you intend to prove. A good thesis statement makes the difference between a thoughtful research ...
mciu.org/~spj/web/thesis.html - 17k - Cached - Similar pages

Education news & resources at the Times Higher Education Supplement
Higher education and university news, features, statistics, research funding opportunities and academic jobs.
www.thes.co.uk/ - 50k - Cached - Similar pages

How to Organize Your Thesis
A guide to what is needed in a graduate research thesis.
www.sce.carleton.ca/faculty/chinneck/thesis.html - 24k - Cached - Similar pages

Welcome to Thesis Builder
Finally, click the "Make an Online Outline" button to generate an outline you can use to write a draft of a persuasive essay based on your thesis statement. ...
www.ozline.com/electraguide/thesis.html - 9k - Cached - Similar pages

Dissertation/Thesis Guide
A practical Guide to assist in the crafting, implementing and defending of a graduate school thesis or dissertation. Authored by S. Joseph Levine, ...
www.learnerassociates.net/dissthes/ - 63k - Cached - Similar pages

Thesis - Wikipedia, the free encyclopedia
In academia, a thesis or dissertation is a document that presents the author's ... At UK universities, the term thesis is usually associated with a Ph.D. ...
en.wikipedia.org/wiki/Thesis - 42k - Cached - Similar pages

Premium Thesis Writing
Professional Thesis Assistance
Your satisfaction is guaranteed
www.PrivateWriting.com Sponsored Links

Thesis or Dissertation
Instant access. 50,000 examples.
Download anytime. Low Prices!
www.dissertationsandtheses.com

Thesis
Discount code to use: "mp32"
Money back & satisfaction guarantee
MasterPapers.com

Thesis-PhD/Master Level
No Plagiarism Guarantee, Secured
\$9/page Affordable and Quality
www.codemanagers.com

Custom Thesis, \$17/page
Any topic. Over 200 writers.
1-866-707-2737
www.phd-dissertations.com

Custom Thesis and Essays
US\$8. No Plagiarism. On-time.
Quality Papers-Money Back Guarantee
www.academicblueprint.com

Thesis
Thousands of A+ papers available
for instant download
www.TermPapers2000.com

Thesis
Visualize, Organize, Outline, Write
Download Free Trial Software!
www.WritersBlocks.com

More Sponsored Links »

Blackboard Academic Suite - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Favorites Media Home Help Logout

Address http://autonline.aut.ac.nz/webapps/portal/frameset.jsp?tab=courses&url=/bin/common/course.pl?course_id=_8854_1 Go Links

AUT online

Home All My Courses My Old Courses Community Course Admin

ALL MY COURSES > SOFTWARE ENGINEERING (S2 2006) > CONTROL PANEL > TURNITIN ASSIGNMENTS > VIEW TURNITIN ASSIGNMENT

This is your assignment inbox. To view a paper, click the paper's title. To view an Originality Report, click the paper's Originality Report icon in the report column. A ghosted icon indicates that the Originality Report has not yet been generated.

assignment inbox edit assignment libraries class stats preferences help

Inbox for: Turnitin for Methodology Assignment

show: new

show: low % ↔ high %

submit Roster Sync

page: [1]

author	title	report	grade	gm	file	paper id	date
Wang, Ke	Methodology Evaluation	11%	--	--		27776856	08-23-06
Fires, Olivia	Methodology Evaluation	5%	--	--		27774652	08-23-06
Hill, Jonathan	0593662 Jonathan, 0590945 Phillip - Meth...	5%	--	--		27781711	08-24-06
Kun, Dirang	Methodology	5%	--	--		27772241	08-23-06
Humpherson, David	Method evaluation V0pt2	2%	--	--		27789307	08-27-06
Quinlan, Phillip	-- no submission --	--	--	--	--	--	late
Ru, Ray	-- no submission --	--	--	--	--	--	late

Copyright © 1998-2006 iParadigms, LLC. All Rights Reserved.

usage policy | privacy pledge | helpdesk | research resources

Powered by Blackboard

Internet

Colonising the lifeworld?

- The ‘lifeworld’ of CS Educators
 - We want to see students rise to challenges set in their courses
 - build knowledge, develop and grow
 - develop professionally and ethically
 - Be fairly rewarded for their own efforts



•An example

- Google as a search engine - a technical system steering CS educational experiences consistent with lifeworld of educators and students
- Google as an advertising site - economic system primary steering CS educational experiences in a manner inconsistent with lifeworld of educators
 - But maybe consistent with lifeworld of some students?
- Turnitin.com as a response (all students viewed as cheats)
 - All examples of distorted communication

Do Some Computer Corporations Have Special Moral Obligations?

Bionics

<https://www.nationalgeographic.com/magazine/2010/01/bionics-robotics-medical-technology/>

<https://www.ceoinstitute.com/resources/bionics-institute-tinnitus-research>

<http://www.medicalbionics.com/rd.php>

The Christchurch Call <https://www.christchurchcall.com/>

Conflict of Professional Responsibility (cont.)

Scenario and Questions

- ❑ In the early 1980s, a U.S. military proposal called the Strategic Defense Initiative (SDI) was introduced and debated
- ❑ It was a national missile defense (NMD) system that would provide a “defense shield” against incoming ballistic missiles
- ❑ The SDI proposal, which was vigorously supported by the Reagan administration, soon became very controversial
- ❑ While SDI’s supporters argued that the missile system was essential for America’s national defense, critics argued that the system’s software was unreliable

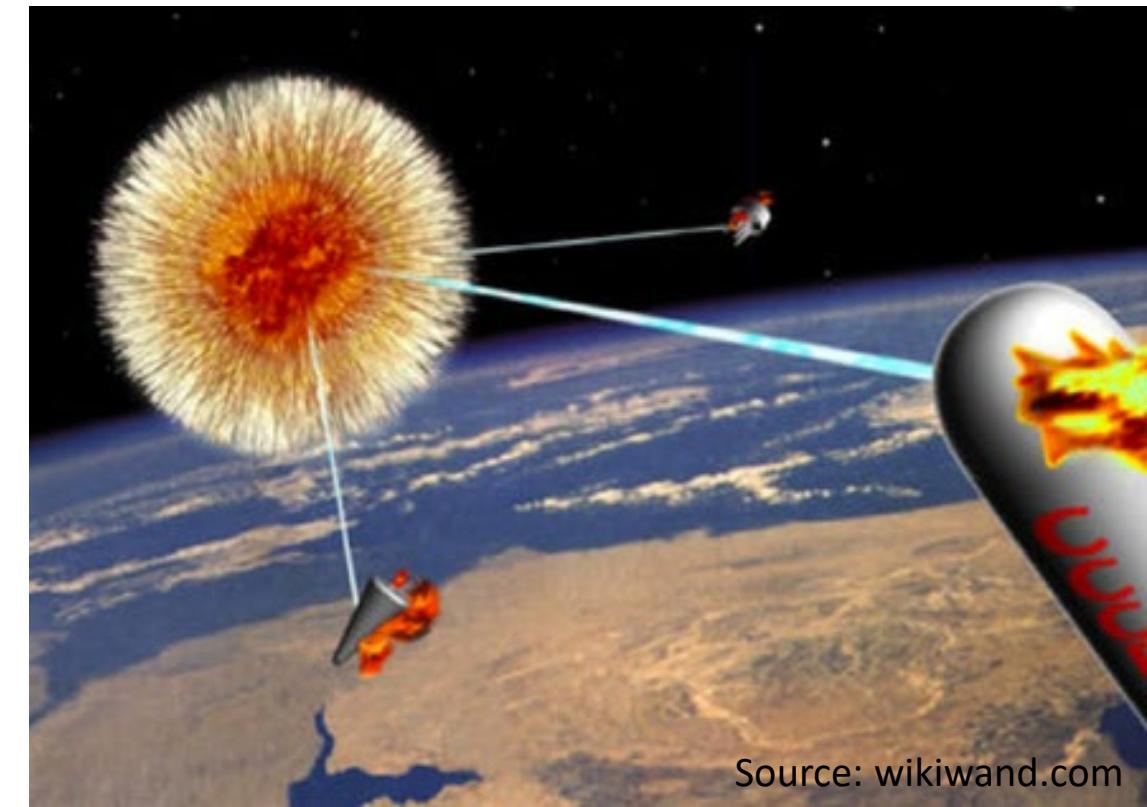
One critic, David Parnas, decided to go public with his concerns about SDI.



Source: wikiwand.com

Conflict of Professional Responsibility (cont.)

- When Parnas went public with his position, some of SDI's supporters accused him of disloyalty and of acting out of his own self-interest.
- Many of Parnas's defenders, pointed out that Parnas walked away from a lucrative consulting contract.
- *Did Parnas do the right thing?*



Conflict of Professional Responsibility (cont.)

- Richard De George criteria has an answer
- Two situations
 - a. morally permitted to blow the whistle, and
 - b. morally obligated to do so
- You are morally permitted to go public with information about the safety of a product if
 - 1. The product will do serious and considerable harm to the public
 - 2. The engineer(s) have reported the serious threat to their immediate supervisor
 - 3. The engineer(s) have exhausted the internal procedures and possibilities
- You are morally required to go public if
 - 4. The engineer(s) have accessible, documented evidence that would convince a reasonable, impartial, observer that one's view of the situation is correct
 - 5. The engineer(s) have good reasons to believe that by going public the necessary changes will be brought about



Conflict of Professional Responsibility (cont.)

- Discussion (back to SDI and David Parnas case)

**Was Parnas morally
permitted, morally
required (obligated) or
both to blow the whistle?**

Appendix: Professional Decision Making

- A Three-step Strategy for Approaching Computer Ethics Issues (*)
 - Step 1: Identify a practice involving information and communications technology, or a feature of that technology, that is controversial from a moral perspective.
 - 1a. Disclose any hidden (or opaque) features or issues that have moral implications
 - 1b. If the ethical issue is descriptive, assess the sociological implications for relevant social institutions and socio-demographic and populations.
 - 1c. If the ethical issue is also normative, determine whether there are any specific guidelines, that is, professional codes that can help you resolve the issue.
 - Step 2. Analyze the ethical issue by clarifying concepts and situating it in a context.
 - 2a. If a policy vacuums exists, go to Step 2b; otherwise, go to Step 3.
 - 2b. Clear up any conceptual muddles involving the policy vacuum and go to Step 3.
 - Step 3. Deliberate on the ethical issue. The deliberation process requires two stages.
 - 3a. Apply one or more ethical theories (see Chapter 2) to the analysis of the moral issue, and then go to Step 3b.
 - 3b. Justify the position you reached by evaluating it via the standards and criteria for successful logic argumentation (see Chapter 3).

* Adopted from (Tavani 2013)

Quality Assurance, Testing and TDD

Week 7



Taking Stock

The schedule for the course

Where are we now?

The Assessment Schedule

Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

Taking Stock

Week

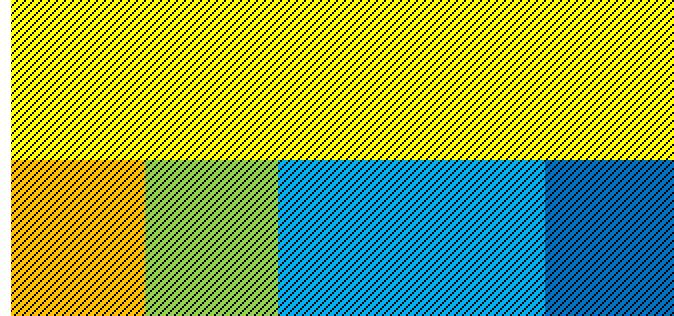
No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Review
Questionnaire



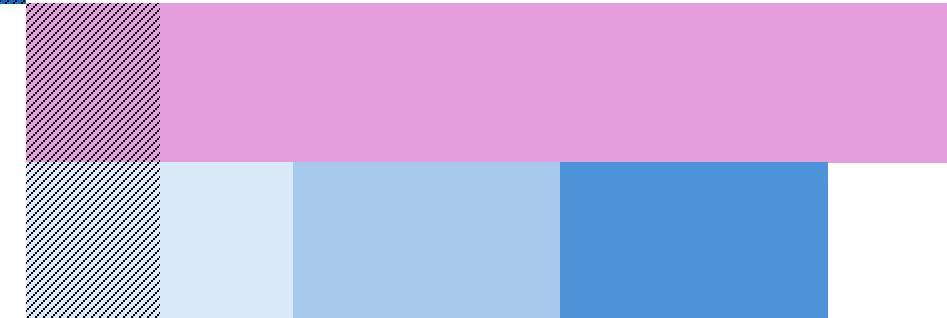
Assgt 1A -
Techstack



Worksheets

Assgt 1B -
Team Project

Iterati
ons



Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)

(Individual)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,

Setup the tools needed to work with the selected

Tech Stack (front-end/backend)

Set up tools to assure quality of product

Set up tools to deploy the product to the cloud

Set up tools to monitor and alert issues post deploy

Learn how to use the tools

Learn how to use the Tech Stack

Understand the product goals -> Product Backlog

Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)

Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)

(small team - 4 Including QA)

Capability building by Developing a Product in a small team

Apply a new tech stack and tool set

Practice DevOps and Scrum WoW

Collaborate with a Product Owner and team

Three sprints to learn fast – fast feedback

Submit – reviews weeks 8,10,12 (tutorials)

Capability and learning Portfolio with Evidence

Product increments

Sprint 1 weeks 6 and 7

Sprint 2 weeks 8 and 9

Sprint 3 weeks 10 and 11

Ass 2 Knowledge Check (30%)

(Individual, online questions)

A set of questions about scenarios to confirm you have understood main language and principles

Sometime in Revision weeks (Faculty schedules)

Team Contract and Commitments

Additions:

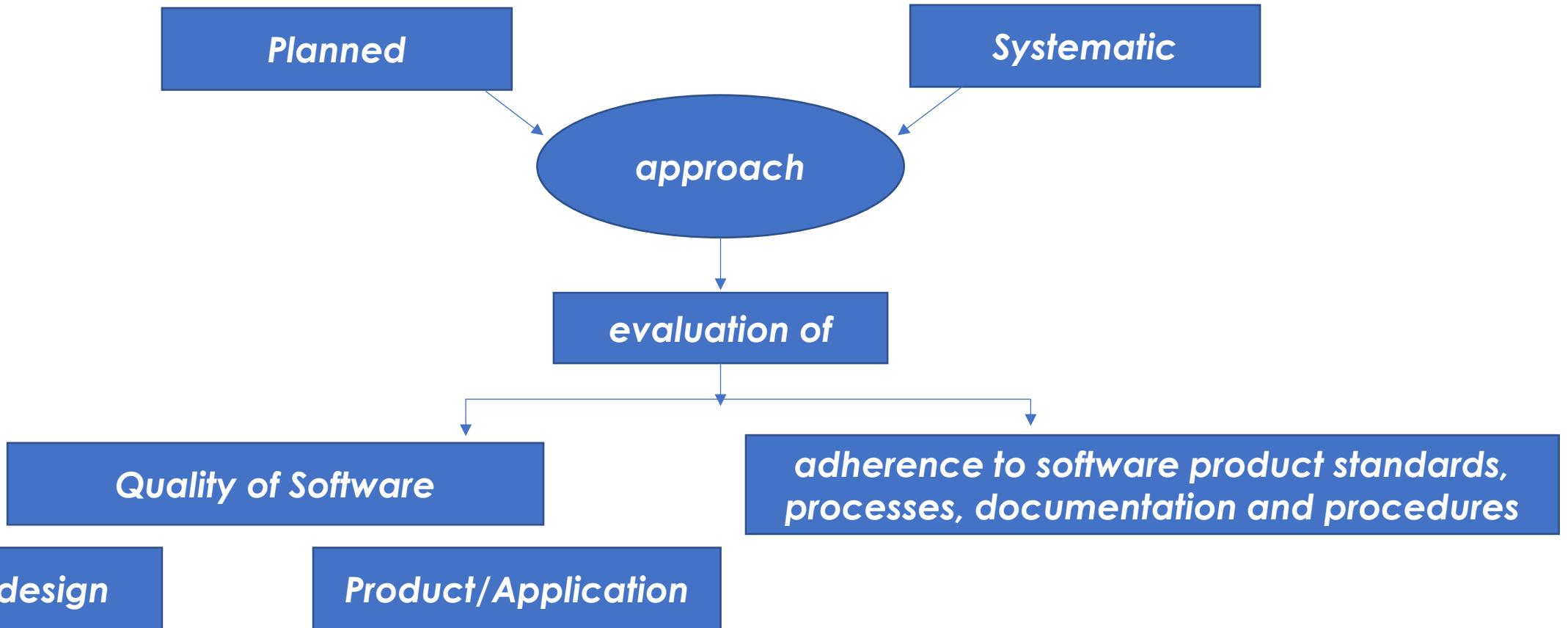
Revision History

Date	Version	Description	Author
<dd/mm/yyyy>	<x.x>	<details>	<name>

Team Issue Report (*date and revision version*):

- Explanation of Adverse situation
- Strategy for making up for deficiencies to the team such as: skills lack or lack of team contribution
- Strategy for demonstrating team commitment
- Author and signature

What is Quality Assurance?



Key Aspects of Quality?

...software quality has been termed “the elusive target” [9], which can be viewed from five different perspectives:

- The *transcendental view* sees quality as something that can be recognized but not defined.
- The *user view* sees quality as fitness for purpose.
- The *manufacturing view* sees quality as conformance to specification.
- The *product view* sees quality as tied to inherent characteristics of the product.
- The *value-based view* sees quality as dependent on the amount a customer is willing to pay for it.

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects.
ACM Inroads, 2(2), 14-15. <https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality & Quality Assurance - Process ?

Quality in Learning

there are differing models for educational quality. The one I prefer is that ...whereby learning ... as a '*transformative process*' for the student – like the '**'transcendental view'** of quality.

Differing ways of producing quality outcomes depending upon the goals

the need for a ***quality process*** to ensure predictable and high quality outcomes. This is where selection of a methodology to fit the needs of the project is necessary.



Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality – Roles & Measurement ?

A further important element of a **quality process** - the identification and **allocation of roles and responsibilities** to appropriately skilled team members

an alternative to process, **measurement** is a classic approach to quality, whether that addresses process or product dimensions.

For instance, the ISO9126 **standard** specifies a set of software quality attributes, including: functionality; reliability; efficiency; usability; maintainability and portability

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality –Metrics and Testing?

A set of **metrics** accompany standards,

- a yardstick by which conformance to the standard can be demonstrated or a lack of conformance can be highlighted.
- For students, **standards for coding and document formatting, or for recording meeting minutes** may be relevant examples, where **compliance** with the quality standard **can be objectively demonstrated**.

testing, while part of QA - more properly classified as a **quality control** activity (QC) rather than QA.... it is inherently **part of the production function, but as a control check added on at the end**.

- BUT a **well framed and multi layered testing strategy** (including unit tests, integration tests, usability tests, performance and stress tests, acceptance tests etc.) is **a key element supporting a QA framework** for systems related projects.

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality - Reviews?

more in-line activities of *quality review* have much to offer.

For instance Robert Glass when asked for the three best software engineering practices came up with “*inspections, inspections, inspections*” arguing that they “do a better job of error-removal than any competing technology”

Reviews in turn can be *periodic* and *formalised* through mechanisms such as “a walkthrough and a formal technical review” [3], “design and code inspections” [7] or other forms of audit.

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality - Continuous Processes?

more *continuous processes* such as **pair programming**, or the shared workshop models e.g. (JAD) [4].

Test Driven Development (TDD) [13], in which design of tests leads development work, can also be thought of as a *continuous review process*, whereby **quality is “built in” from the outset**.

specific practices may be applied e.g. ongoing practices of **continuous integration, regular (e.g. daily) code builds, refactoring** etc.

or more control oriented practices such as **change control, configuration and version management** [2].

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality - Continuous Improvement?

Finally **at a meta-level** - the notion of *continual process improvement*, or **software process and practice improvement (SPPI)**,

- “aims to build an infrastructure and culture that support effective methods, practices, and procedures and integrate into the ongoing way of doing business”

meta-level thinking ... as students reflect upon the effectiveness of the processes and practices they have applied in their projects. Ideally, they would **adapt and refine them** as they proceed.

- At a minimum, to **reflect upon the processes and practices they have applied** during their projects and **demonstrate awareness of how they could have done things differently** and what those improvements might look like in future.

Clear, T. (2011, June). THINKING ISSUES: A 'potted guide' to quality assurance for computing capstone projects. *ACM Inroads*, 2(2), 14-15.
<https://doi.org/DOI: 10.1145/1963533.1963536>

Key Aspects of Quality – the QA Accountability?

A document (pdf) of a practical **Team agreement** including

- commitments to one another,
- how the team will communicate,
- team roles/accountabilities (including QA for each iteration) and an explanation of why the roles/accountabilities are needed.

1. A description of the Quality Assurance accountability and how it has been exercised within the team

- a. An explanation of your team's emphasis on QA for that iteration
- b. a reflection on how successful you think it was and why?
- c. a reflection on what you would emphasise or do differently for the next iteration

Evolving Views of Software Quality

Over the years, there has been debate about what constitutes software quality and how it should be measured.

This controversy has caused uncertainty across the software engineering community, affecting levels of commitment to the many potential determinants of quality among developers.

An up-to-date catalogue of software quality views could provide developers with contemporary guidelines and templates

Ndukwe, I. G., Licorish, S. A., Tahir, A., & MacDonell, S. G. (2023). How have views on software quality differed over time? Research and practice viewpoints. *Journal of Systems and Software*, 195, 111524.

Software Quality Models

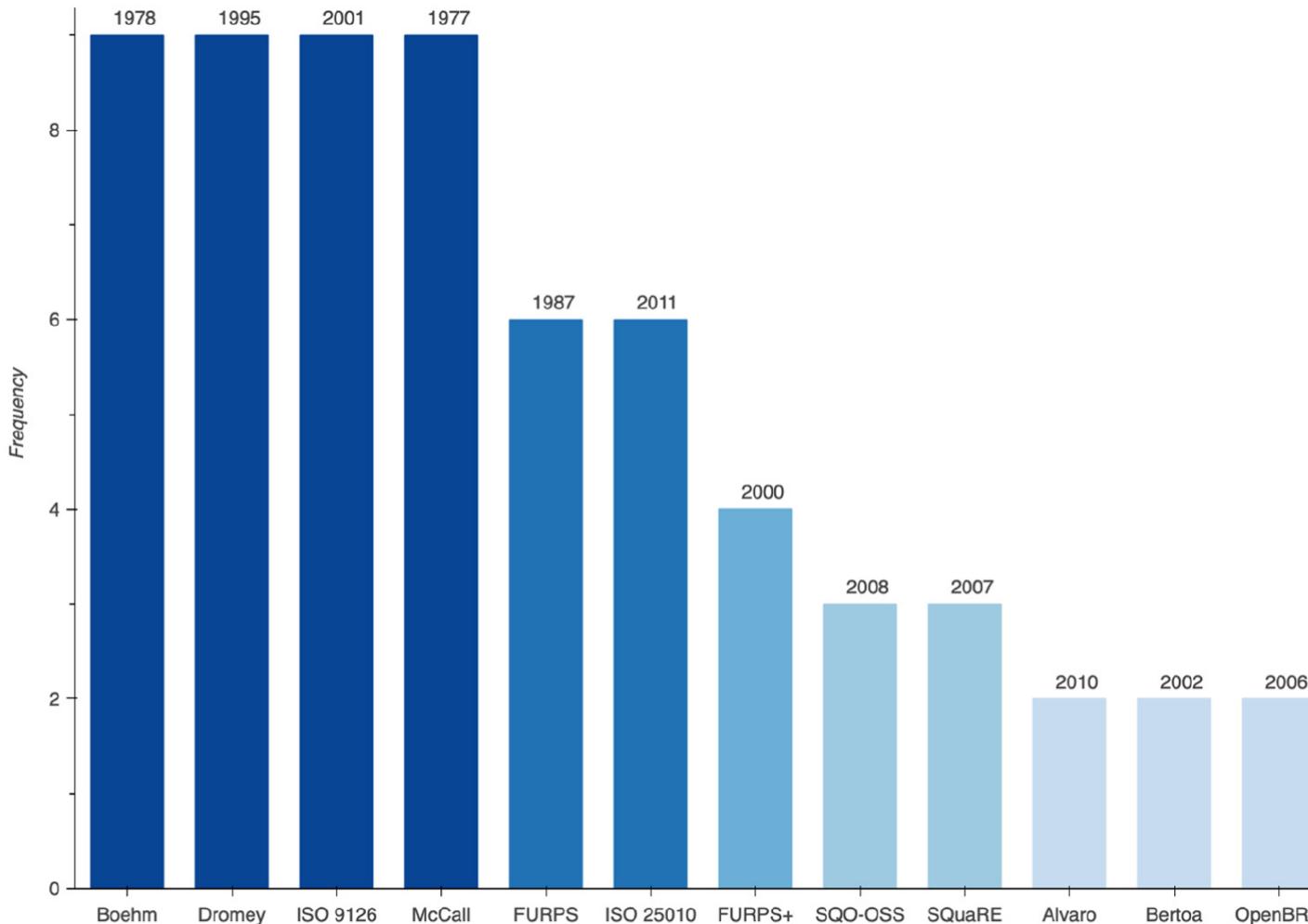


Fig. 1. Frequencies of software quality models with the year each model was proposed.

Ndukwe, I. G., Licorish, S. A., Tahir, A., & MacDonell, S. G. (2023). How have views on software quality differed over time? Research and practice viewpoints. *Journal of Systems and Software*, 195, 111524.

Top 20 Software Quality Characteristics

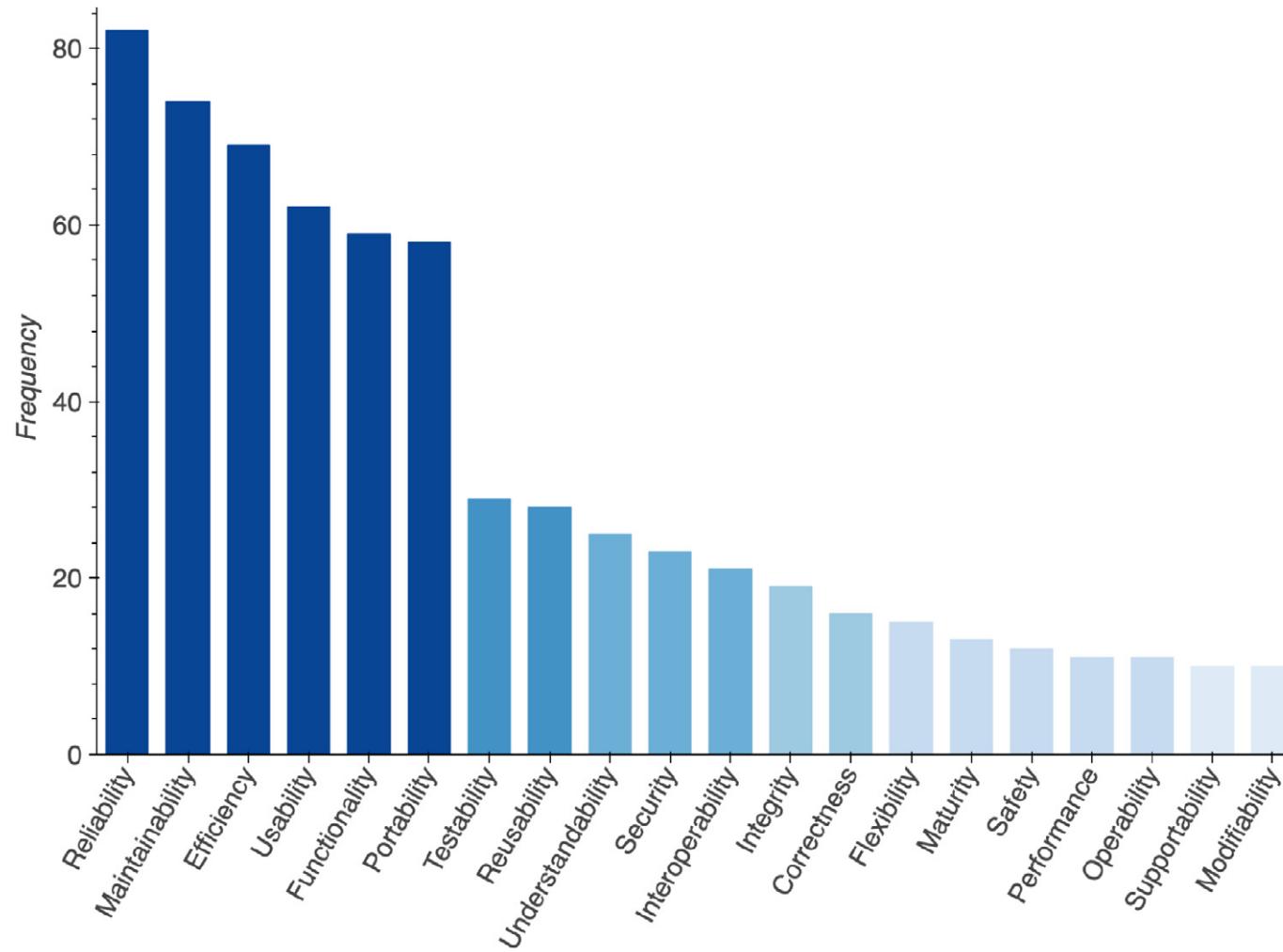


Fig. 2. Frequencies of top twenty software quality characteristics.

Ndukwe, I. G., Licorish, S. A., Tahir, A., & MacDonell, S. G. (2023). How have views on software quality differed over time? Research and practice viewpoints. *Journal of Systems and Software*, 195, 111524.

Questions about quality and testing?

What are the quality criteria to test the quality of code against

What is the quality testing practices to run the tests against these criteria?

How will we know if it passes or fails the quality test?

What should be done if the quality test is failed?

What practices will help to ensure quality test fails do not happen?

Catch low quality

Prevent low quality

How many tests will it take to prove something is correct ?

You can never be 100% sure there are no defects – you can only prove there is a defect!

Then... What is Testing?

Testing is verifying the behaviour of your application is what is specified/expected

Maybe writing code or to do the test or using the application

**Code that passes all the tests we can think of is good quality
(ready to be released)**

Testing needs a set of criteria to test against and the ability to recognize pass or fail

Expected **output** of a function/class/component for a given (type of) **input(s)**

The expected behaviour of an application or system while being used

A given set of coding standards – naming conventions, design principles

A set of standards for UI components and good practice

Good test? “The product should be user friendly”

Why write tests? (4:00)

<https://youtu.be/ZmVBCpefQe8>



Why Write Tests?

Documentation

Tests are specifications of how our code should work

Consistency

Verify that developers are following good practice and the conventions of our team

Comfort and Confidence

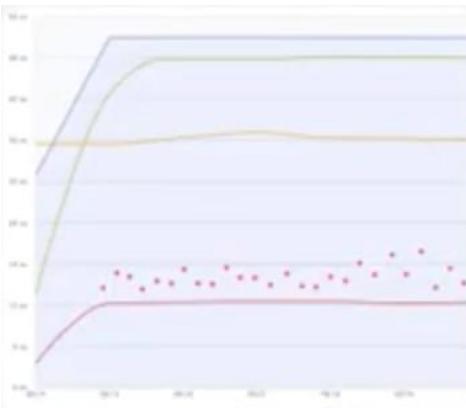
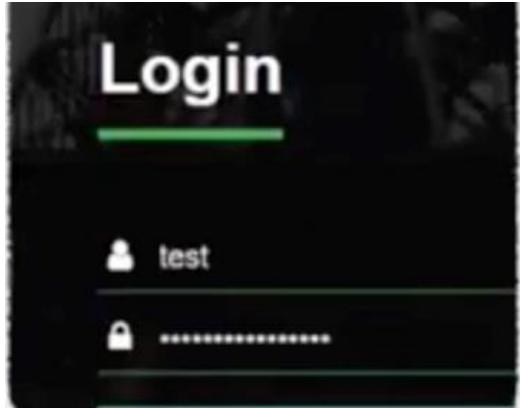
A strong test suite is like a warm blanket – feel ok to release, experiment

Productivity

We write tests because it allows us to ship code faster

Types of software testing

Black Box



Functional

Regulatory Compliance

Usability



Exploratory

Developers do these

White Box

```
'a valid user r  
rRegRequest = n  
.build()  
.with {  
    status = A  
    it
```

```

    .withName("registration-request")
    .withEmail("user@user.com")
    .withPassword("password123")
    .withStatus("ACTIVE")
    .build();
}

UserRegistration registration =
    userRegistrationService.register(new UserBuilder().withEmail("user@user.com").withName("user").withPassword("password123").withStatus("ACTIVE").build());

```

Unit Tests

Integration

System

Performance
Stress
Load/stability
Security

Static tests

Linters
SonarQube
Load/stability
79 security

Regression

Static tests

Code analysis while you are coding or a large code base

- Visual Studio Code squiggly lines
- Linters (ESLint)
- Other tools - Sonarqube

The risk of no integration testing



Integration of code developed bit by bit is high risk

DEPENDENCIES

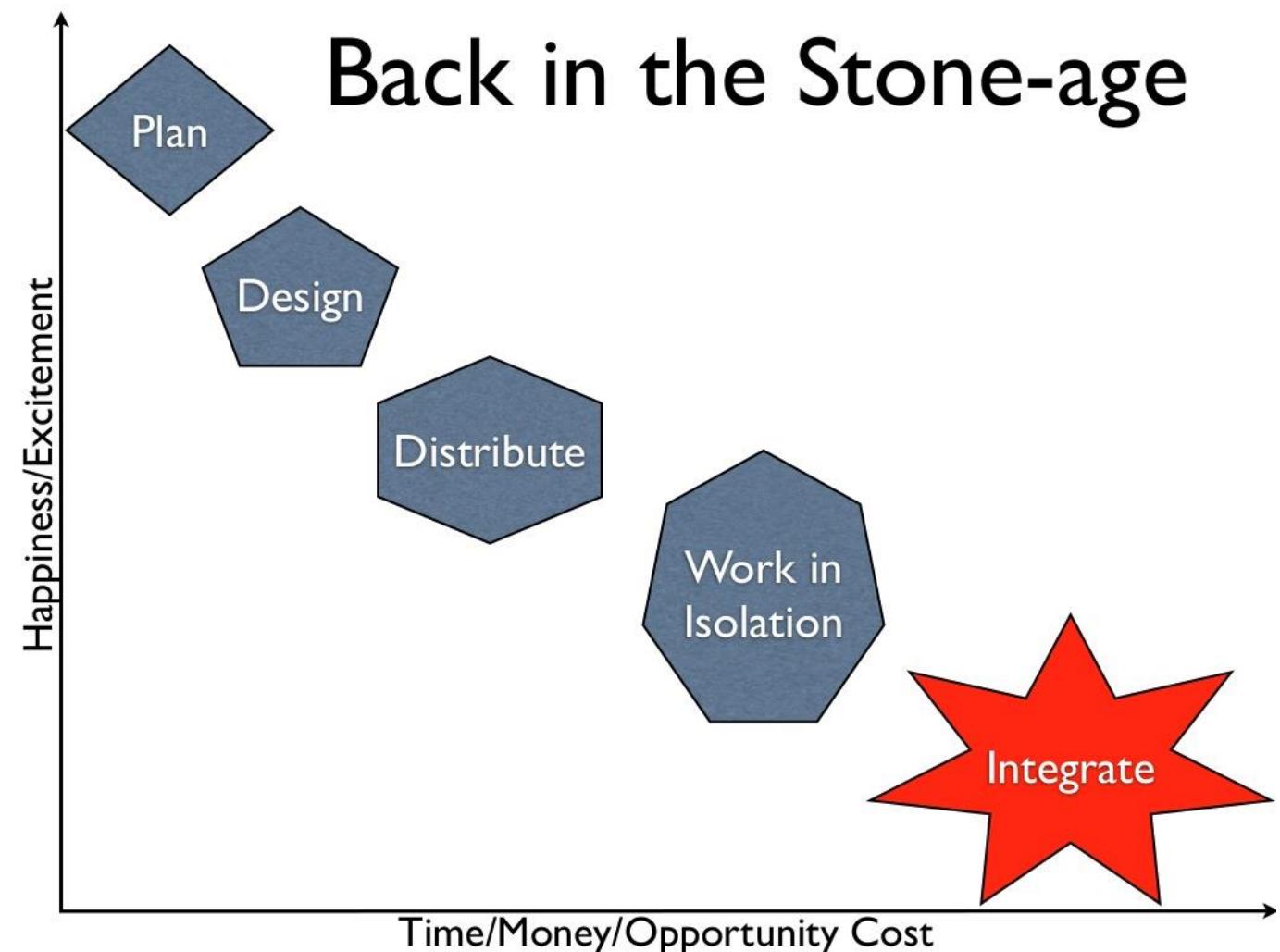


COORDINATION

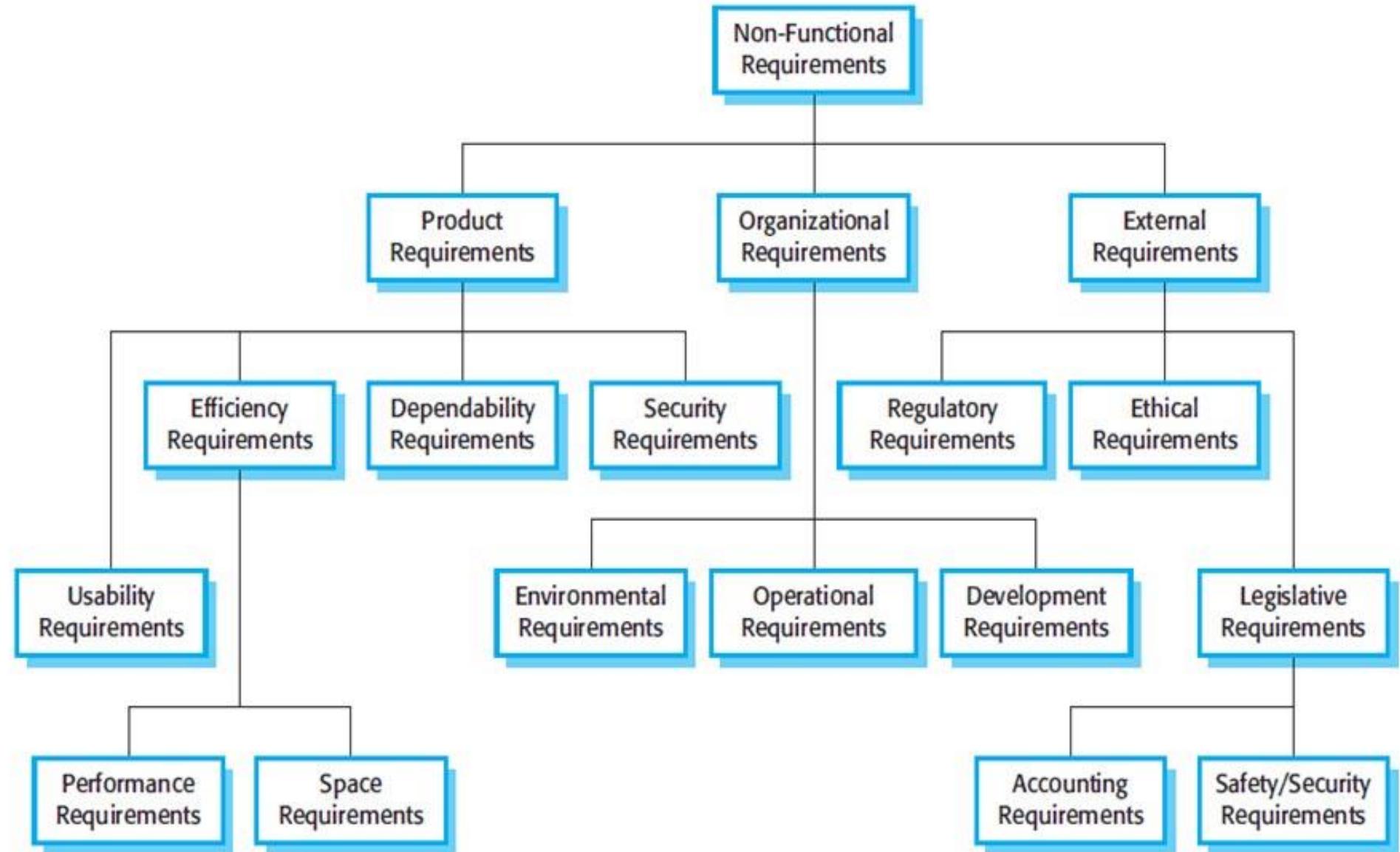
It works on my machine

I didn't realise you were
doing it that way

I didn't understand what
you were doing



Non-Functional Requirements or Quality Requirements

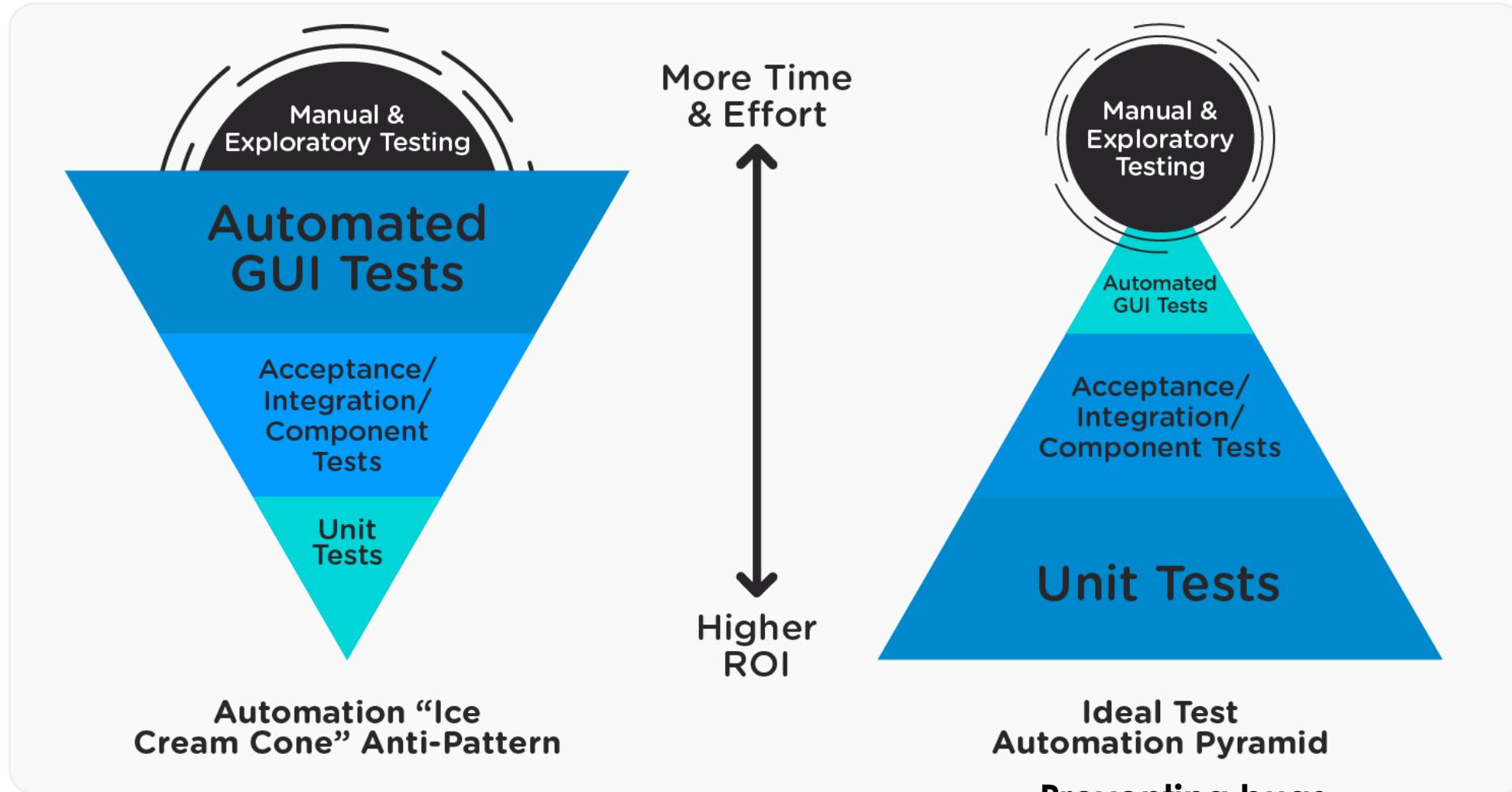


Non-Functional Requirements or Quality Requirements

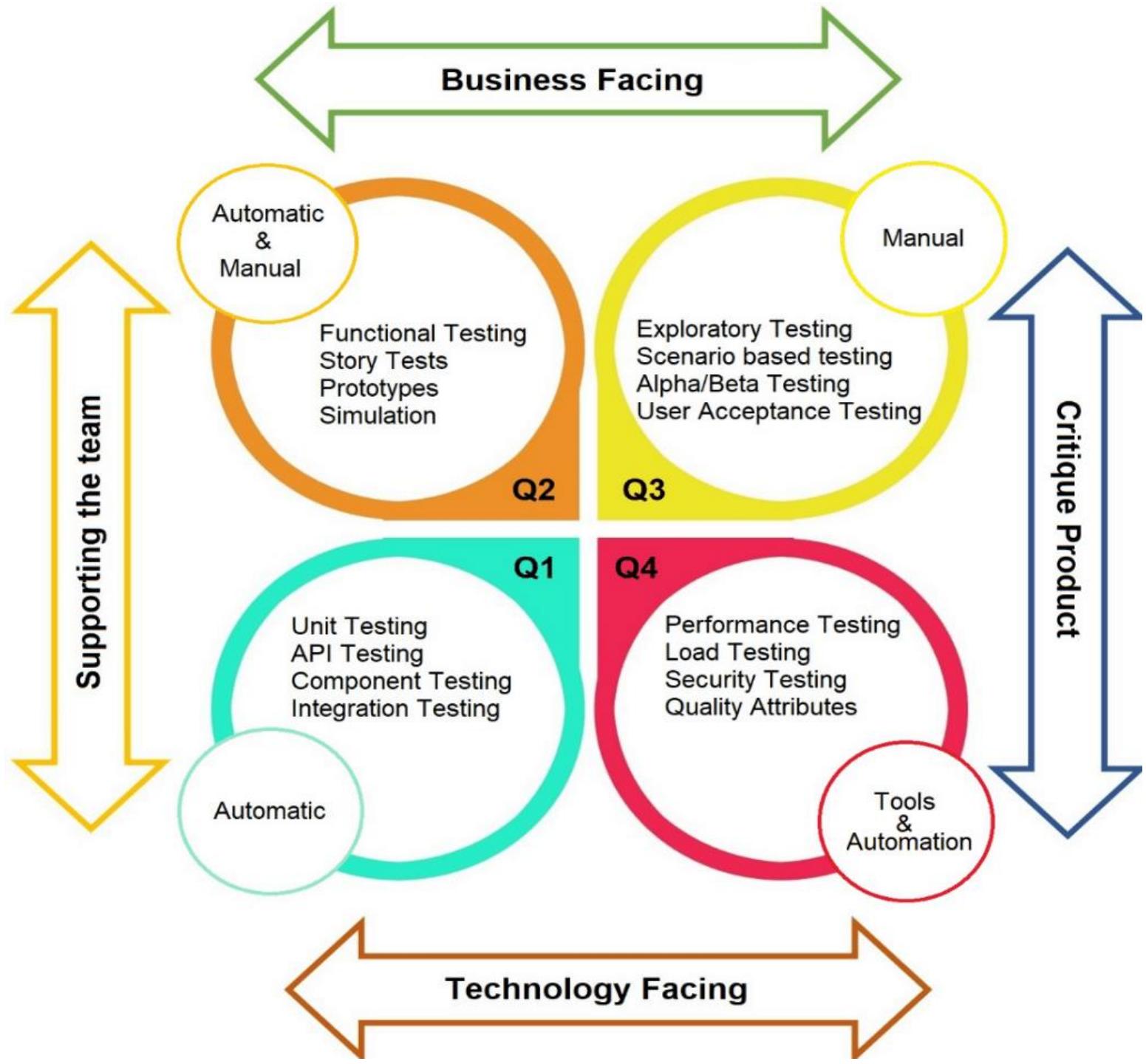
Functional Requirements	Non Functional Requirements
<ul style="list-style-type: none">• Product features	<ul style="list-style-type: none">• Product property
<ul style="list-style-type: none">• Describe the actions with which the user work is concerned	<ul style="list-style-type: none">• Describe the experience of the user while doing the work
<ul style="list-style-type: none">• A functions that can be captured in use cases	<ul style="list-style-type: none">• Non-functional requirements are global constraints on a software system that results in development costs, operational costs
<ul style="list-style-type: none">• A behaviors that can be analyzed by drawing sequence diagrams, state charts, etc	<ul style="list-style-type: none">• Often known as software qualities
<ul style="list-style-type: none">• Can be traced to individual set of a program	<ul style="list-style-type: none">• Usually cannot be implemented in a single module of a program



Test Pyramid – which types of tests to spend the most effort on



Crispin's Agile testing Quadrants



Unit Tests with React (10:38)

<https://youtu.be/ZmVBCpefQe8>



A Basic Test

```
const expected = true;
const actual = false;

if (actual !== expected) {
  throw new Error(` ${actual} is not ${expected}`);
}
```

Unhelpful Output

```
node tests/basic.test.js
/Users/chrisschmitz/code/presentation-react-testing/tests/basic.test.js:5
throw new Error(` ${actual} is not ${expected}`);
^
          ^
Error: true is not false
    at Object.<anonymous> (/Users/chrisschmitz/code/presentation-react-testing/tests/basic.test.js:5:9)
    at Module._compile (internal/modules/cjs/loader.js:776:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:787:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:829:12)
```

A Jest Test

```
const expected = true;
const actual = false;

test("it works", () => {
  expect(actual).toBe(expected);
});
```

Test assertion Library
Test Runner
Mocking features

```
→ presentation-react-testing yarn jest tests/basic-jest.test.js
yarn run v1.19.0
$ /Users/chrisschmitz/code/presentation-react-testing/node_modules/.bin/jest tests/basic-jest.test.js
FAIL tests/basic-jest.test.js
  ● it works

    expect(received).toBe(expected) // Object.is equality

      Expected: false
      Received: true

      3
      4
      > 5   test("it works", () => {
      6     expect(actual).toBe(expected);
      7   });

      at Object.<anonymous>.test (tests/basic-jest.test.js:5:18)
```

React testing – does it render correctly

The screenshot shows a browser window with two tabs: "App.test.tsx" and "App.tsx". The "App.tsx" tab is active, displaying the following code:

```
1 import * as React from "react";
2 import * as ReactDOM from "react-dom";
3 import { getQueriesForElement } from "@testing-library/dom";
4
5 import { App } from "./App";
6
7 test("renders the correct content", () => {
8   const root = document.createElement("div");
9   ReactDOM.render(<App />, root);
10
11   const { getByText, getByLabelText } = getQueriesForElement(root);
12
13   expect(getByText("Todos")).not.toBeNull();
14   expect(getByLabelText("What needs to be done?")).not.toBeNull();
15   // expect(root.querySelector("button").textContent).toBe("Add #1");
16 });
17
```

The browser's left sidebar displays a "Todos" application with a text input field containing "What needs to be done?" and a button labeled "Add #1".

Simulating user interaction

The screenshot shows a code editor interface with a dark theme. On the left, an `App.test.tsx` file is open, containing Jest test code for a `App` component. The code includes two tests: one for rendering correct content and another for allowing users to add items to their list. On the right, a browser window displays a "Todos" application with a list of items: "TODOs", "What needs to be done?", and "Add #1". A context menu is open over the "Add #1" item, showing options: "RTL Presentation Slides", "Testing", and "RTL Slides". The "RTL Presentation Slides" option is highlighted with a blue border. At the bottom, the browser's developer tools show a "Console" tab with "0" entries and a "Problems" tab with "3" entries.

```
File Edit Selection View Go Help  
Intro to Testing / 4. Simulating User Interation  
App.test.tsx ● App.finished-test.tsx  
1 import * as React from "react";  
2 import { render, fireEvent } from "@testing-library/react";  
3  
4 import { App } from "./App";  
5  
6 test("renders the correct content", () => {  
7   const { getByText, getByLabelText } = render(<App />);  
8  
9   getByText("TODOs");  
10  getByLabelText("What needs to be done?");  
11  getByText("Add #1");  
12});  
13  
14 test("allows users to add items to their list", () => {  
15   const { getByText, getByLabelText } = render(<App />);  
16  
17   const input = getByLabelText("What needs to be done?");  
18   fireEvent.change(input, { target: { value: "New Todo" } });  
19   fireEvent.click(getByText("Add #1"));  
20   expect(getByText("New Todo")).toBeInTheDocument();  
21  
22});  
23});
```

https://vu1lx.csb.app/

TODOS

What needs to be done?

- RTL Presentation Slides Add #1
- RTL Presentation Slides
- Testing
- RTL Slides

Console 0 Problems 3 R
Console was cleared

Unit testing Frameworks

<https://www.youtube.com/watch?v=3e1GHCA3GP0>

Unit Testing with Jest and React testing Library (24 mins 2020)



<https://www.youtube.com/watch?v=ZmVBCpefQe8&t=13s>

Getting started with React Testing (51 mins (2020))



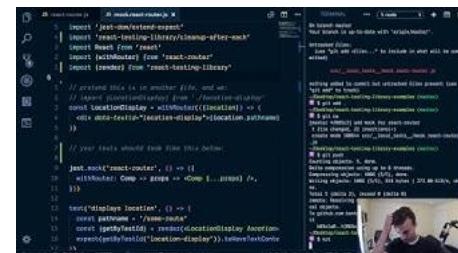
<https://www.youtube.com/watch?v=7r4xVDI2vho&t=30s>

Jest Crash Course



<https://www.youtube.com/watch?v=XDkSaCgR8g4&t=24s>

Component Unit Testing and mocking with react testing library (14 mins 2018)



Testing Resources

<https://www.youtube.com/watch?v=3e1GHCA3GP0&t=7s>

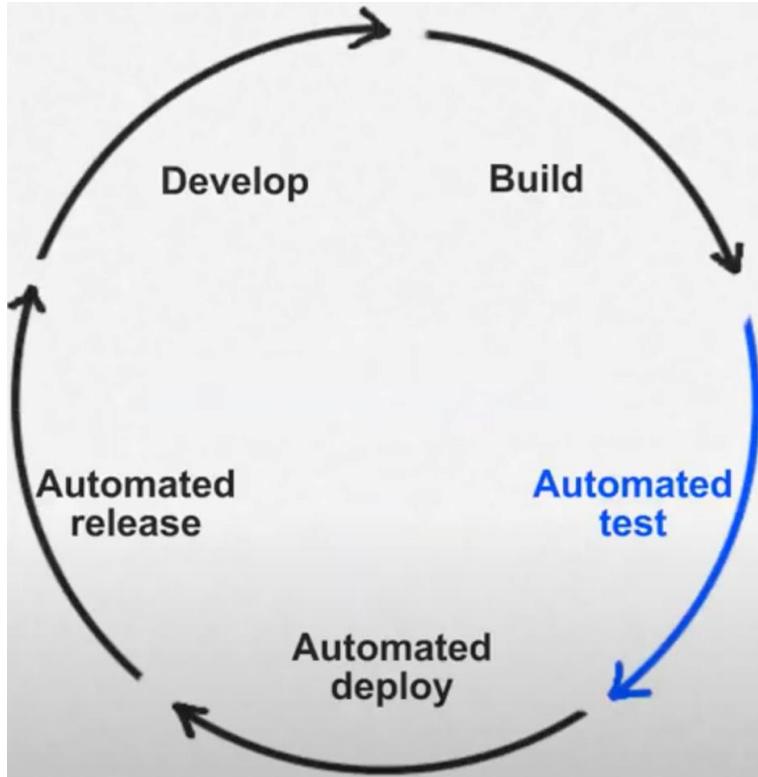
Complete Guide to Component testing with Jest for beginners. Crash course on Jest and mocking

<https://www.youtube.com/watch?v=XDkSaCgR8g4&t=26s>

Component Unit Testing (and mocking) with react-testing-library

Test Automation

Regression testing – run ALL tests in a code base (or selected, prioritised ones)



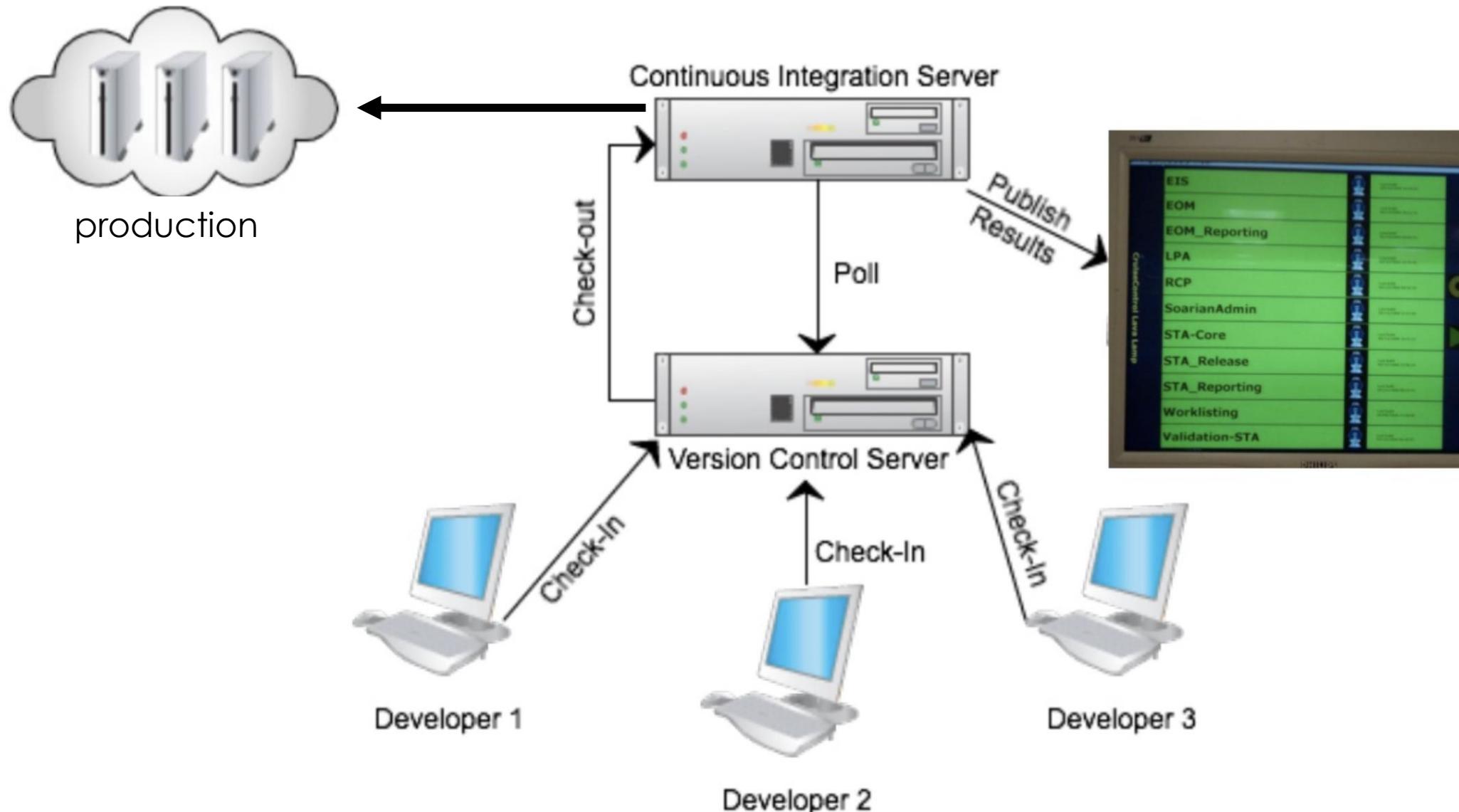
Continuous Delivery and DevOps ways of working rely on test automation to make rapid deployment possible

Test automation benefits

- guarantees tests are run at the right time and everything stops if they fail (still need to write and maintain the tests!)
- Frees up developers and testers for more analytical testing (exploratory, code reviews)

See the work of Michael Bolton and Linda Crispin!

Automating Integration Testing Continuously



What is Automated Testing? (7 mins, 2019)

<https://youtu.be/Nd31XiSGJLw>

Test cases for a unit test design

Test case ID	Test Case Description	Test setup Pre-conditions	Test Steps Instructions	Test data for each step (inputs)	Expected results Outputs
--------------	-----------------------	---------------------------	-------------------------	----------------------------------	--------------------------

Write two test cases for this requirement for an online store:

If a user is a VIP customer and they order more than 10 items then they should not pay freight

Would this be a good user story?

As a VIP customer I want to be rewarded for my status so I can save money

Acceptance criteria (user stories)

Success criteria, Acceptance Tests

GIVEN [an initial context]

WHEN [some event happens]

THEN [an expected state linked to the functionality]

Given the person making an order is logged on and is a VIP customer and they order 11 items, **when** the order is confirmed **then** delivery of the order will be free

Backend API testing with Postman or Insomnia

Postman also covered in Worksheet 2

<https://www.postman.com/product/what-is-postman/>

<https://learning.postman.com/docs/writing-scripts/script-references/test-examples/#getting-started-with-tests>

Test Driven Development

Developers are responsible for writing unit tests

These are a check that their code is behaving as expected – choosing expected input and output values – particularly EDGE cases.

jUnit

pHpUnit

nUnit

<https://medium.com/codeclan/testing-react-with-jest-and-enzyme-20505fec4675>

<https://medium.com/javascript-in-plain-english/i-tested-a-react-app-with-jest-testing-library-and-cypress-here-are-the-differences-3192eae03850>

9 excuses why developers don't test their code

Can you think of any?.....

1. "My Code Works Fine — Why Should I Even Bother Testing It?"
2. "This Piece of Code Is Untestable"
3. "I Don't Know What to Test"
4. "Testing Increases the Development Time, and We're Running Out of Time"
5. "The Requirements Are No Good"
6. "This Piece of Code Doesn't Change"
7. "I Can Test This Way Faster If I do It Manually"
8. "The Client Only Wants to Pay for Deliverables"
9. "This Piece of Code Is So Small ... It Won't Break Anything"

<https://medium.com/better-programming/9-excuses-why-programmers-dont-test-their-code-8860a616b1b5>



What is TDD and the workflow? What are the benefits and why? What is the red green cycle What is refactoring?

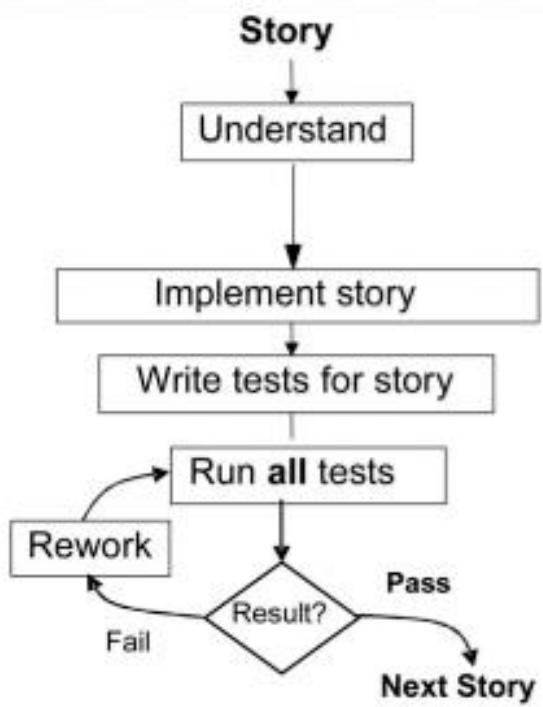
When Test Driven Development Goes Wrong

Test Driven Development is one of the best ways that we have to amplify our talent as software developers, maybe software engineers. This Software Engineering practice is one of the best ways to improve the quality of your code, but it is difficult to do it well, and it often goes wrong.

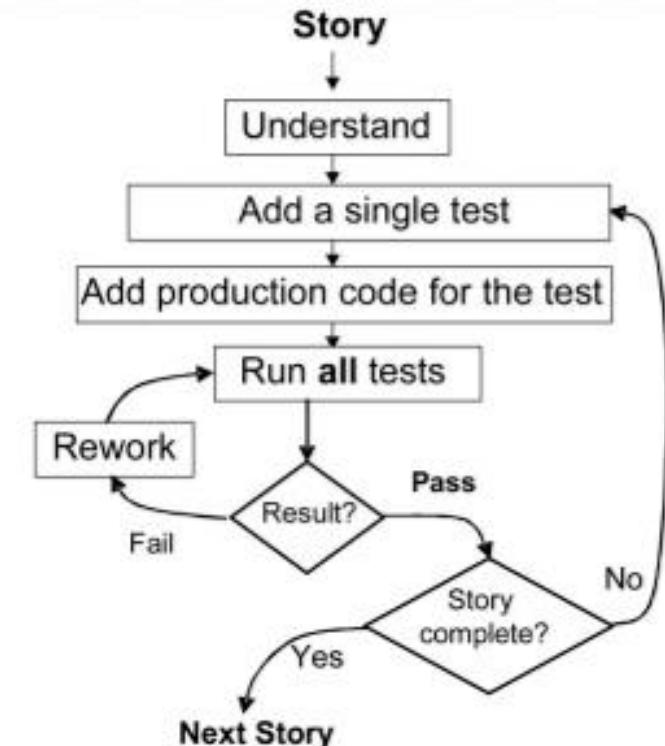
The interesting thing is that when it goes wrong, it may be at its most valuable. TDD is a cornerstone of Continuous Delivery, BDD and DevOps. Using it to give us valuable, efficient feedback on the quality of our designs is at the heart of its value but is often missed by people who are new to it. Dave explores these ideas with some real code examples to demonstrate the value of TDD. In this episode, Dave Farley explains 5 common ways that TDD goes wrong, how to fix them, and what we can learn from them.

https://www.youtube.com/watch?v=UWtEVKVPBQ0&list=FLAw_BzmvV1FBxEPeV4pDqug&index=7

Test Driven Development



Test - Last



Test - First

H.Erdogmus, et al., "On the effectiveness of the test-first approach to programming," *Software Engineering, IEEE Transactions on*, vol. 31, pp. 226-237, 2005.

TDD = TFD + Refactoring

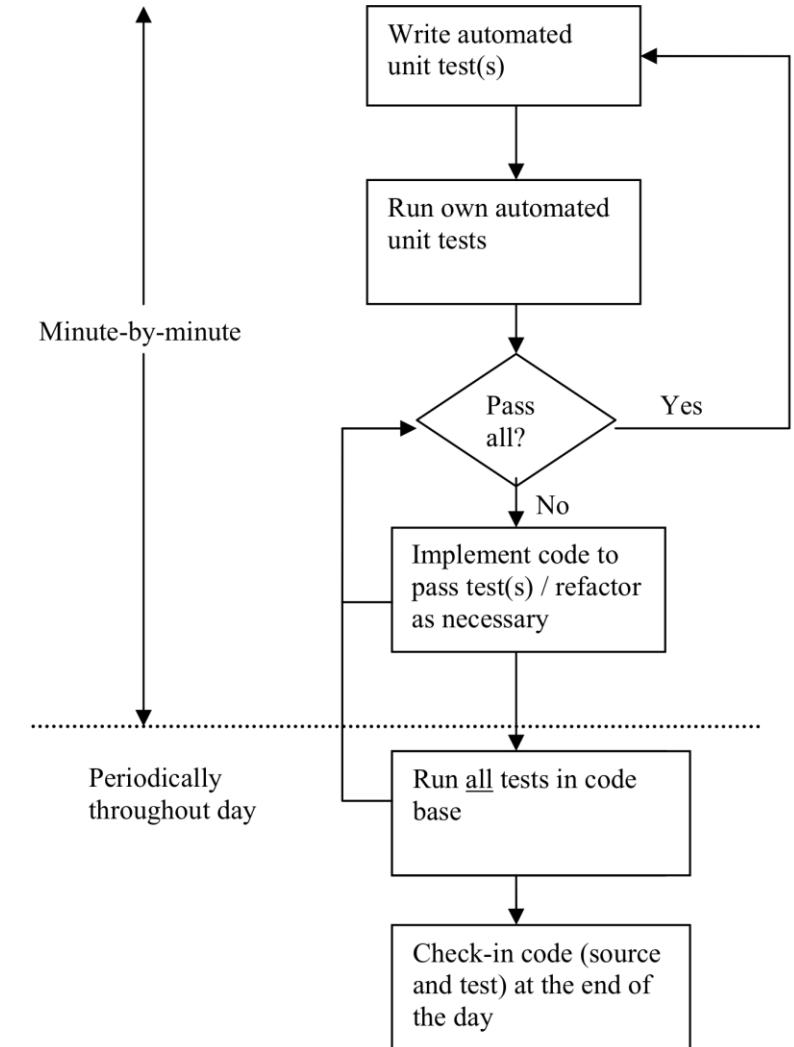
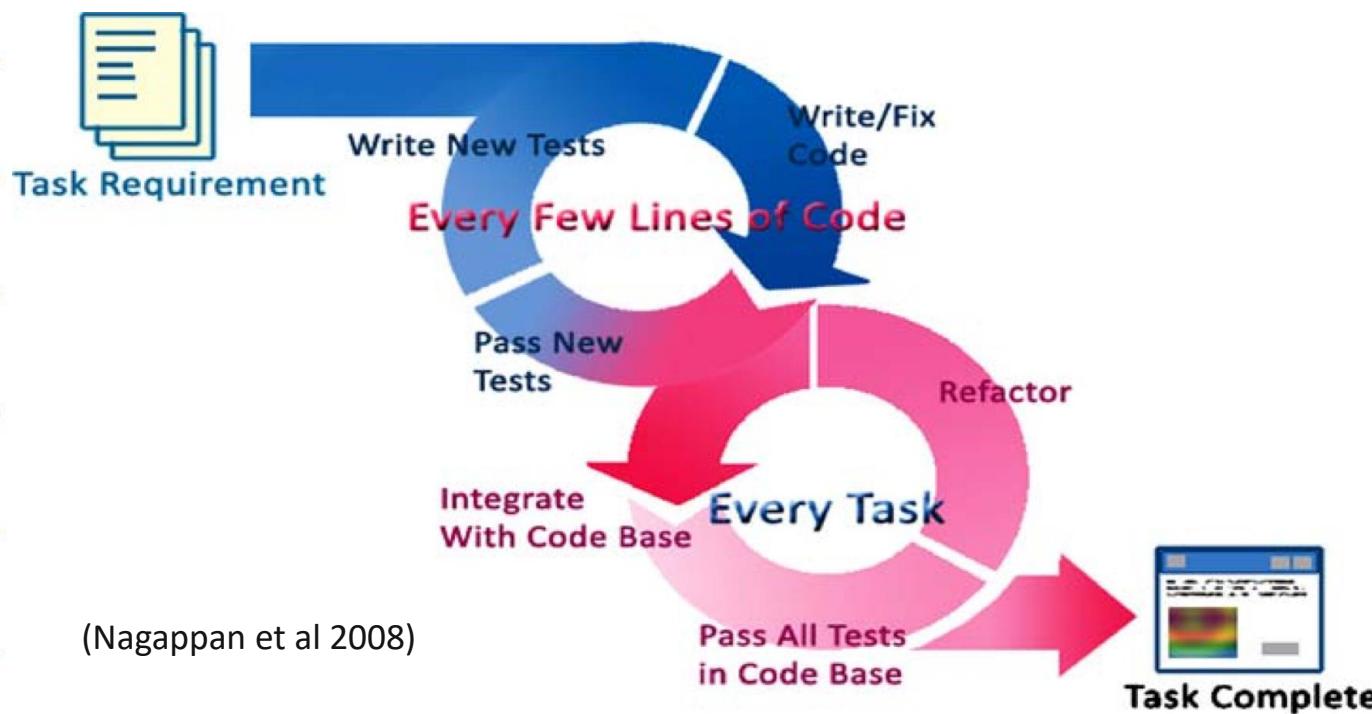


Figure 1: Test-Driven Development

(Bhat & Nagappan, 2006)



My client wants the dev to make a slight change to the number of books for a VIP discount to 10 or more.

Is the code change I make and example of Refactoring the code?

NO, a functional change is NOT refactoring
Refactoring just changes the code structure

Examples of TDD - for further study

Intro to TDD and where it came from and why it is needed. Good content – delivery SLOW

<https://www.youtube.com/watch?v=dWayn0QsJr8>

“Toy” Example of TDD code writing in Java and jUnit – look at others’ comments

https://www.youtube.com/watch?v=O-ZT_dtIrR0

Another step-by step “toy” example in Java and jUnit– red and green refactoring

Lynda.com

Programming Foundations: Test-Driven Development. (EXCELLENT!)

Spring: Test-Driven Development with Junit

Exploring test-driven development with the assert keyword

From [Advanced Java Programming](#) course

Testing setup

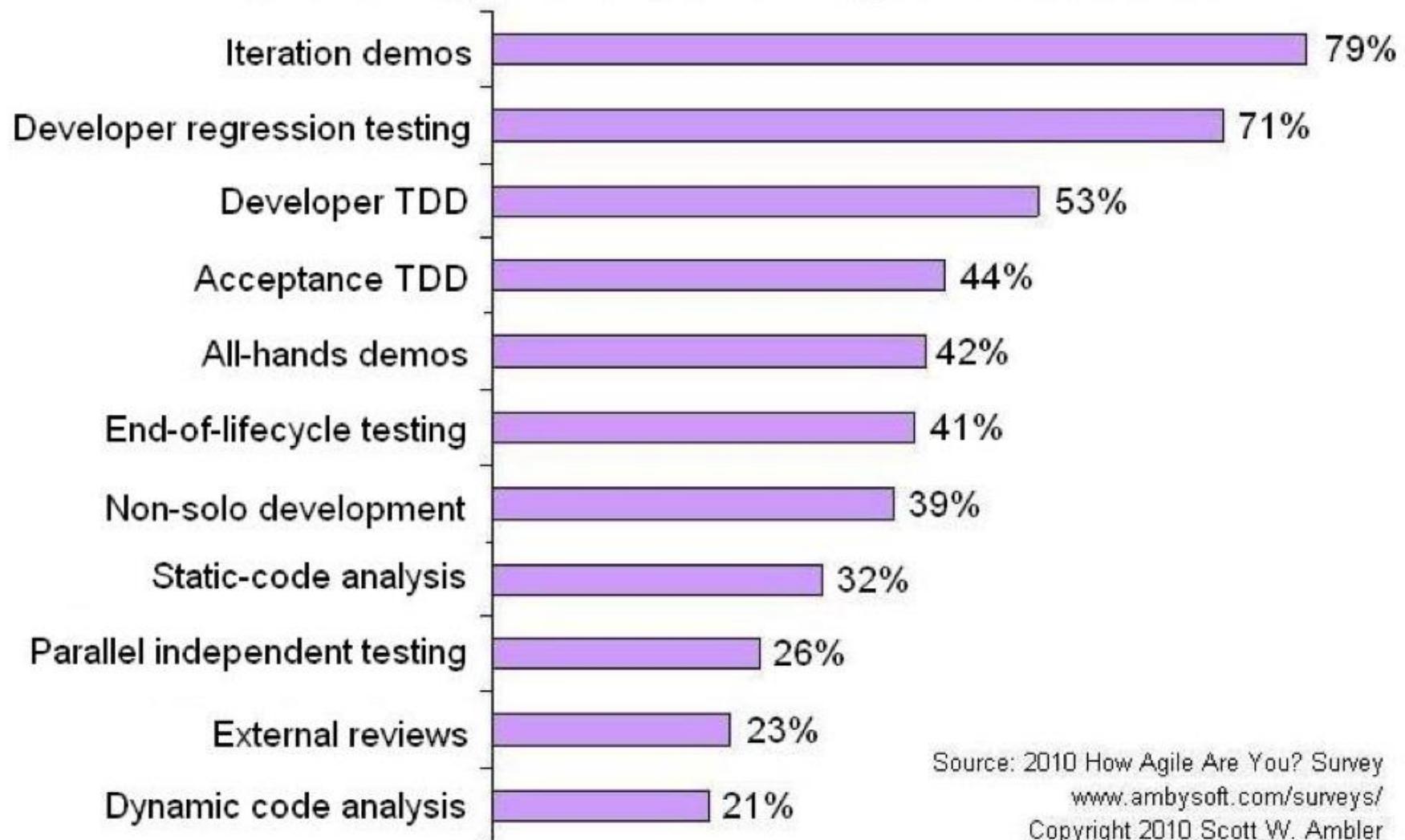
Sometimes we need a test database or **staging database** so we can do testing with data that isn't in the production database, and set up certain initial database conditions

Sometimes to test our object we rely on another class/object not yet coded – we can write a **mock object** for it that behaves (accepts/returns whatever) like the real object would with no actual processing. Our test can work with the mock object.

Don't forget about data – a test data base can be vital, a demo without a solid set of test data will be a failure ☹

Cf. test data for worksheet 4

How are Agile Teams Validating their own Work?



Source: 2010 How Agile Are You? Survey
www.ambysoft.com/surveys/
Copyright 2010 Scott W. Ambler

Test-Driven Development: Concepts, Taxonomy, and Future Direction



Test-driven development creates software in very short iterations with minimal upfront design. Poised for widespread adoption, TDD has become the focus of an increasing number of researchers and developers.

David Janzen
Simex LLC

Hossein Saeidian
University of Kansas

The *test-driven development* strategy requires writing automated tests prior to developing functional code in small, rapid iterations. Although developers have been applying TDD in various forms for several decades,¹ this software development strategy has continued to gain increased attention as one of the core extreme programming practices.

XP is an *agile method* that develops object-oriented software in very short iterations with little upfront design. Although not originally given this name, TDD was described as an integral XP practice necessary for analysis, design, and testing that also enables design through refactoring, collective ownership, continuous integration, and programmer courage.

Along with pair programming and refactoring, TDD has received considerable individual attention since XP's introduction. Developers have created tools specifically to support TDD across a range of languages, and they have written numerous books explaining how to apply TDD concepts. Researchers have begun to examine TDD's effects on defect reduction and quality improvements in academic and professional practitioner environments, and educators have started to examine how to integrate TDD into computer science and software engineering pedagogy. Some of these efforts have been implemented in the context of XP projects, while others are independent of them.

TEST-DRIVEN DEVELOPMENT DEFINED

Although its name implies that TDD is a testing

method, a close examination of the term reveals a more complex picture.

The test aspect

In addition to testing, TDD involves writing automated tests of a program's individual units. A unit is the smallest possible testable software component. There is some debate about what exactly constitutes a unit in software. Even within the realm of object-oriented programming, both the class and method have been suggested as the appropriate unit. Generally, however, the method or procedure is the smallest possible testable software component.

Developers frequently implement test drivers and function stubs to support the execution of unit tests. Test execution can be either a manual or automated process and can be performed by developers or designated testers. Automated testing involves writing unit tests as code and placing this code in a test harness or framework such as JUnit. Automated unit testing frameworks minimize the effort of testing, reducing a large number of tests to a click of a button. In contrast, during manual test execution developers and testers must expend effort proportional to the number of tests executed.

Traditionally, unit testing occurred after developers coded the unit. This can take anywhere from a few minutes to a few months. The unit tests might be written by the same programmer or by a designated tester. With TDD, the programmer writes the unit tests prior to the code under test. As a result, the programmer can immediately execute the tests after they are written.

feature software metrics

Does Test-Driven Development Really Improve Software Design Quality?

David S. Janzen, California Polytechnic State University, San Luis Obispo

Hossein Saeidian, University of Kansas

TDD is first and foremost a design practice. The question is, how good are the resulting designs? Empirical studies help clarify the practice and answer this question.

Software developers are known for adopting new technologies and practices on the basis of their novelty or anecdotal evidence of their promise. Who can blame them? With constant pressure to produce more with less, we often can't wait for evidence before jumping in. We become convinced that competition won't let us wait.

Advocates for test-driven development claim that TDD produces code that's simpler, more cohesive, and less coupled than code developed in a more traditional test-last way. Support for TDD is growing in many development contexts beyond its common association with Extreme Programming. Examples such as Robert C. Martin's bowling game demonstrate the clean and sometimes surprising designs that can emerge with TDD,¹ and the buzz has proven sufficient for many software developers to try it. Positive personal experiences have led many to add TDD to their list of "best practices," but for others, the jury is still out. And although the literature includes many publications that teach us how to do TDD, it includes less empirical evaluation of the results.

In 2004, we began a study to collect evidence that would substantiate or question the claims regarding TDD's influence on software. Unfortunately, these perspectives miss TDD's primary purpose, which is design. Granted, the tests are important, and automated test suites that can run at the click of a button are great. However,

- Misconception #1: TDD equals automated testing. Some developers we met placed a heavy emphasis on automated testing. Because TDD has helped propel automated testing to the forefront, many seem to think that TDD is only about writing automated tests.

- Misconception #2: TDD means write all tests first. Some developers thought that TDD involved writing the tests (all the tests) first, rather than using the short, rapid test-code iterations of TDD.

TDD misconceptions

As we learned, professional development teams

Does Test-Driven Development Improve the Program Code? Alarming Results from a Comparative Case Study

Maria Sjöaalto¹ and Pekka Abrahamsson²

¹ F-Secure Oyj,
Elektrotölkätie 3, FIN-90570 Oulu, Finland

Maria.Sinisa@f-secure.com

² VTT Technical Research Centre of Finland

Pekka.Abrahamsson@vtt.fi

Abstract. It is suggested that test-driven development (TDD) is one of the most fundamental practices in agile software development, which produces loosely coupled and highly cohesive code. However, how the TDD impacts on the structure of the program code have not been widely studied. This paper presents the results from a comparative case study of five small scale software development projects where the effect of TDD on program design was studied using both traditional and package level metrics. The empirical results reveal that an unwanted side effect can be that some parts of the code may deteriorate. In addition, the differences in the program code, between TDD and the iterative test-last development, were not as clear as expected. This raises the question as to whether the possible benefits of TDD are greater than the possible downsides. Moreover, it additionally questions whether the same benefits could be achieved just by emphasizing unit-level testing activities.

Keywords: Test-Driven Development, Test-first Programming, Test-first Development, Agile Software Development, Software Quality.

1 Introduction

Test-driven development (TDD) is one of the core elements of Extreme Programming (XP) method [1]. The use of the TDD is said to yield several benefits. It is claimed to improve test coverage [2] and to produce loosely coupled and highly cohesive systems [3]. It is also believed to encourage the implementation code to be more explicit [3] and to

What Makes Testing Work: Nine Case Studies of Software Development Teams

Christopher D Thomson*

Business School

University of Hull

Hull, UK

c.thomson@hull.ac.uk

Mike Holcombe, Anthony J H Simons

Department of Computer Science

University of Sheffield

Sheffield, UK

{m.holcombe,a.simons}@dcs.shef.ac.uk

accurately. We found that the teams had a high degree of variation in external quality that cannot be easily explained by the teams' testing practice alone or the practices of XP alone. Instead we find that testing must become part of the culture of the team, and can do so in at least two different ways.

II. LITERATURE REVIEW

Testing; test first; test driven development; extreme programming; empirical; qualitative; testing culture.

1. INTRODUCTION

Extreme programming (XP) [1] presents what is, on the surface, a simple but effective testing practice known as *test first*, or *test driven development*. The idea is reassuringly simple, that unit tests should be defined and run before any implementation is present. Several studies have attempted to measure the effect of the *test first* practice on the quality of the software produced and time taken, but the results presented are inconclusive.

The testing practice of XP encompassed more than simply *test first*. System testing is automated, incremental, regular, and early. User acceptance testing is similar although often less or not at all automated [1]. But these features can be used independently of *test first* and perhaps with some success. This raises our research question:

How does the practice of testing affect a team following XP – or if test first is not followed then do the other practices still influence the way testing is performed and the external quality?

Most Common Mistakes in Test-Driven Development Practice: Results from an Online Survey with Developers

Mauricio Finavarro Aniche, Marco Aurélio Gerosa

Department of Computer Science - University of São Paulo (USP) - Brazil
{aniche,gerosa}@ime.usp.br

Abstract

Test-driven development (TDD) is a software development practice that supposedly leads to better quality and fewer defects in code. TDD is a simple practice, but developers sometimes do not apply all the required steps correctly. This article presents some of the most common mistakes that programmers make when practicing TDD, identified by an online survey with 218 volunteer programmers. Some mistakes identified were: to forget the refactoring step, building complex test scenarios, and refactor another piece of code while working on a test. Some mistakes are frequently made by around 25% of programmers.

1. Introduction

Test-driven development (TDD) is an important practice in Extreme Programming (XP) [1]. As agile practices suggest, software design emerges as software grows. In order to respond very quickly to changes, a constant feedback is needed and TDD gives it by making programmers constantly write a small test that fails and then make it pass. TDD is considered an essential strategy in emergent design because when writing a test prior to code, programmers contemplate and decide not only the software interface (e.g. class/method names, parameters, return types, and exceptions thrown), but also on the software behavior (e.g. expected results given certain inputs) [13].

TDD is not only about test. It is about helping the team to understand the features that the users need and to deliver those features reliably and predictably. TDD turns testing into a design activity as programmers use

Kent Beck sums up TDD as follows: 1) quickly add a test; 2) run all tests and see the new one fail; 3) make a little change; 4) run all tests and see them all succeed; 5) refactor to remove duplication [18]. In order to get all benefits from TDD, programmers should follow each step. As an example, the second step states that programmers should watch the new test fail and the fifth step states to refactor the code to remove duplication. Sometimes programmers just do not perform all steps of Beck's description. Thus, the value TDD aggregates to software development process might be reduced.

This article presents some of the most common mistakes that programmers make during their TDD sessions, based on an online survey conducted during two weeks in January, 2010, with 218 volunteer programmers. The survey and its data can be found at <http://www.ime.usp.br/~aniche/tdd-survey/>.

This article is structured as follows: Section 2 presents some studies about the effects of TDD on software quality; Section 3 shows the most common mistakes programmers make based on the survey; Section 4 discusses about the mistakes and ideas on how to sort them out; Section 5 presents threats to validity on the results of this article; Section 6 concludes and provides suggestions for future works.

2. The effects of TDD on software quality

Empirical experiments about the effects of TDD have been conducted generally with two different groups: graduate students at universities and professional developers at the industry. Most of them show that TDD increases code quality, reduces the defect density, and provides better maintainability.

In practice, few organizations apply the strict TDD process in the form of the repetition of the sequence of steps described above. The real insight has been test-first development and, more specifically, the idea that any new code must be accompanied by new tests. It is not even critical that the code should come only after the test (the "F" of TFD): what counts is that you never produce one without the other.

This idea has come to be widely adopted —and should be adopted universally. It is one of the major contributions of agile methods.

(Meyer 2014)

Full citation: Buchan, J., Li, L., & MacDonell, S.G. (2011) Causal Factors, Benefits and Challenges of Test-Driven Development: Practitioner Perceptions, in Proceedings of the 18th Asia-Pacific Software Engineering Conference (APSEC 2011). Hochiminh City, Vietnam, IEEE Computer Society Press, pp.405-413.
[doi: 10.1109/APSEC.2011.44](https://doi.org/10.1109/APSEC.2011.44)

Causal Factors, Benefits and Challenges of Test-Driven Development: Practitioner Perceptions

Jim Buchan, Ling Li, Stephen G. MacDonell

SERL, School of Computing and Mathematical Sciences
AUT University, Private Bag 92006
Auckland 1142, New Zealand

jim.buchan@aut.ac.nz, angela.linz@gmail.com, stephen.macdonell@aut.ac.nz

Abstract

This report describes the experiences of one organization's adoption of Test Driven Development (TDD) practices as part of a medium-term software project employing Extreme Programming as a methodology. Three years into this project the team's TDD experiences are compared with their non-TDD experiences on other ongoing projects. The perceptions of the benefits and challenges of using TDD in this context are gathered through five semi-structured interviews with key team members. Their experiences indicate that use of TDD has generally been positive and the reasons for this are explored to deepen the understanding of TDD practice and its effects on code quality, application quality and development productivity. Lessons learned are identified to aid others with the adoption and implementation of TDD practices, and some potential further research areas are suggested.

Keywords: test-driven development (TDD), TDD benefits, TDD challenges, causal network

I. INTRODUCTION

Test-driven development (TDD), which emphasizes a mind-set that functional code should be changed only in response to a failed test, is considered “proven practice” by many contemporary software development practitioners and textbook writers. Although it is a technique that has been practiced for decades [1], it has recently gained more visibility with the rise in use of Agile methodologies such as Extreme Programming (XP), where it is a core practice [2].

Proponents of TDD have reasoned that its use should result in improvements to code quality [3], testing quality [4], and application quality [5], compared to the traditional Test-Last (TL) approach. It has also been claimed to improve overall development productivity, encourage early understanding of the scope of requirements (user stories), as well as potentially leading to enhanced developer job satisfaction and confidence [3].

In contrast, critics claim that the frequent changes to tests in TDD are more likely (than in TL) to cause test breakages, leading to costly rework and loss of productivity [6]. Boehm and Turner [6] also note that with TDD the consequences of developers having inadequate testing skills may be amplified, compared to the consequences for a TL approach. Other critics note that TDD may not be appropriate for all application domains [7].

While these claims and criticisms provide some basis for evaluating the possible utility of TDD, practitioners and researchers have recognized the need for stronger evidence as a basis for investing – or not – in the effort to adopt and implement this set of practices. Recently, empirical researchers have been investigating the claimed benefits, constraints, and applicability of TDD in a variety of industrial and academic settings to build up a body of evidence. This empirical evidence is mixed in its results regarding the benefits of TDD (covered in more detail in Section VI).

This report adds further evidence regarding TDD in practice by describing the experiences of a software development team that has used TDD for three years in a specific project. This project (referred to as the “TDD project”) adopted Extreme Programming (XP) as a development methodology. This was the first project to adopt TDD in this organization

TDD Empirical Study at SERL in AUT

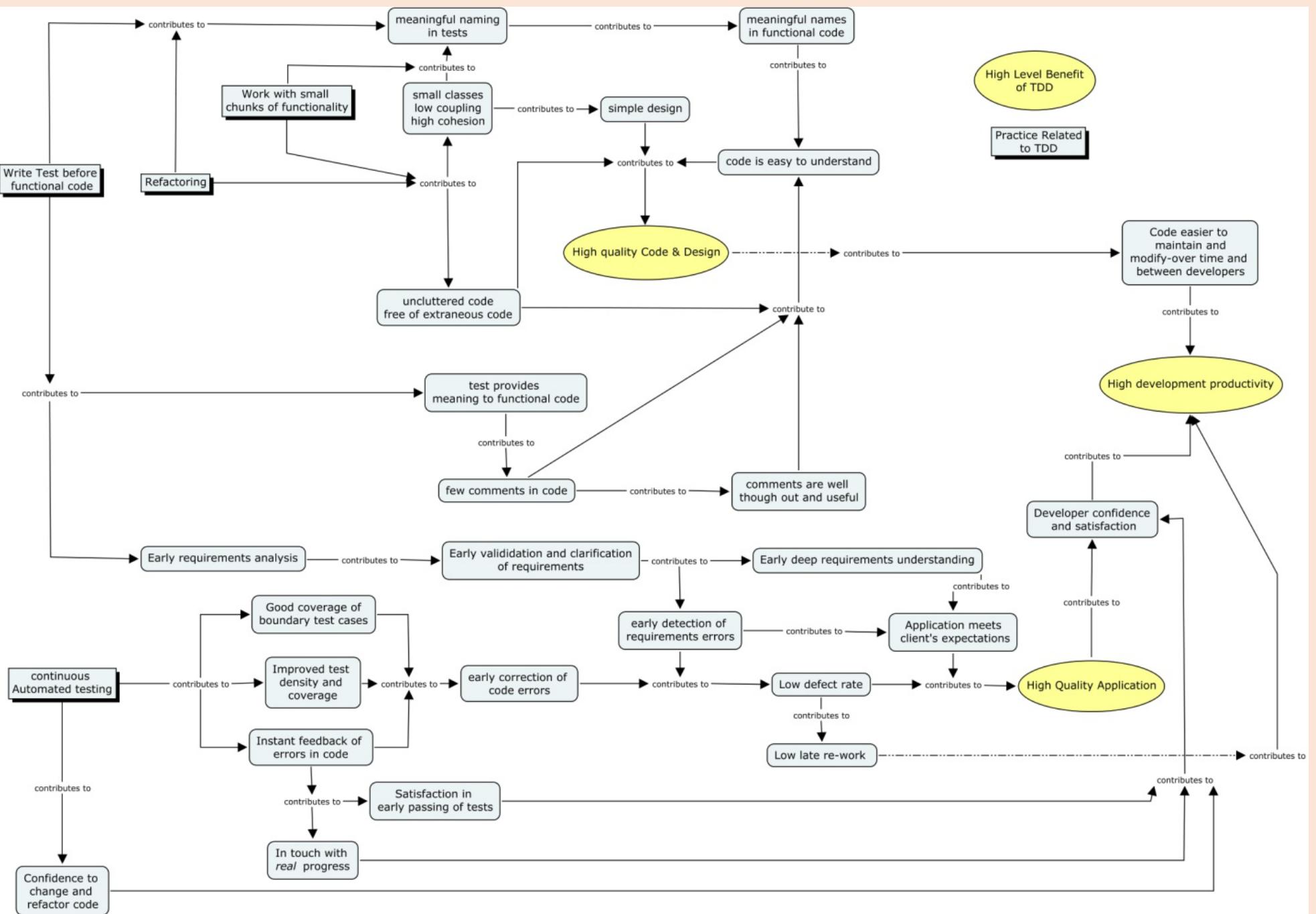


Figure 2. Causal network of TDD benefits and contributing factors.

TDD Research

Table 1. Summary of TDD research in industry.

Study	Type	Number of companies	Number of programmers	Quality effects	Productivity effects
George ⁸	Controlled experiment	3	24	TDD passed 18% more tests	TDD took 16% longer
Maximilien ⁹	Case study	1	9	50% reduction in defect density	Minimal impact
Williams ¹⁰	Case study	1	9	40% reduction in defect density	No change

Table 2. Summary of TDD research in academia.

Controlled experiment	Number of programmers	Quality effects	Productivity effects
Kaufmann ¹¹	8	Improved information flow	50% improvement
Edwards ¹²	59	54% fewer defects	n/a
Erdogmus ¹³	35	No change	Improved productivity
Müller ¹⁴	19	No change, but better reuse	No change
Pančur ¹⁵	38	No change	No change

(Hossein 2005)

How software engineering research aligns with design science: a review

Emelie Engström¹ · Margaret-Anne Storey² · Per Runeson¹ · Martin Höst¹ · Maria Teresa Baldassarre³

Published online: 18 April 2020
© The Author(s) 2020

Abstract

Background Assessing and communicating software engineering research can be challenging. Design science is recognized as an appropriate research paradigm for applied research, but is rarely explicitly used as a way to present planned or achieved research contributions in software engineering. Applying the design science lens to software engineering research may improve the assessment and communication of research contributions.

Aim The aim of this study is 1) to understand whether the design science lens helps summarize and assess software engineering research contributions, and 2) to characterize different types of design science contributions in the software engineering literature.

Method In previous research, we developed a visual abstract template, summarizing the core constructs of the design science paradigm. In this study, we use this template in a review of a set of 38 award winning software engineering publications to extract, analyze and characterize their design science contributions.

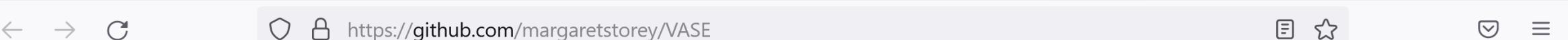
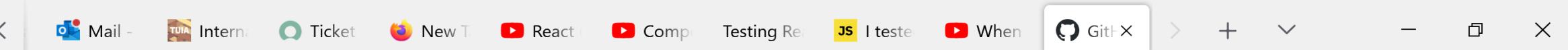
Results We identified five clusters of papers, classifying them according to their different types of design science contributions.

Conclusions The design science lens helps emphasize the theoretical contribution of research output—in terms of technological rules—and reflect on the practical relevance, novelty and rigor of the rules proposed by the research.

Keywords Design science · Research review · Empirical software engineering

Empirical Studies of SE and How to communicate findings with practitioners?

Engström, E., Storey, M.-A., Runeson, P., Höst, M., & Baldassarre, M. T. (2020, 2020/07/01). How software engineering research aligns with design science: a review. *Empirical Software Engineering*, 25(4), 2630–2660. <https://doi.org/10.1007/s10664-020-09818-7>



VASE: Visual Abstracts for Software Engineering

Slides and materials from a tutorial given at CBSoft 2019 (held in Brazil) describe how to use the design science template in action are posted here: <https://github.com/margaretstorey/cbsoft2019tutorial/> (the tutorial was also about research methods, see slides 64 of the slides deck <https://github.com/margaretstorey/cbsoft2019tutorial/blob/master/CBSoft%20Tutorial%202019%20Slides.pdf>).

Slides from talk at ESEM are available here: <https://www.slideshare.net/mastorey/using-a-visual-abstract-as-a-lens-for-communicating-and-promoting-design-science-research-in-software-engineering>

Related blog post is here: <http://margaretstorey.com/blog/2017/11/09/visual-abstracts/>

Empirical software engineering research aims to generate prescriptive knowledge that can help software engineers improve their work and overcome their challenges, but deriving these insights from real-world problems can be challenging. We promote design science as an effective way to produce and communicate prescriptive knowledge while we propose using a visual abstract template to communicate design science contributions and highlight the main problem/solution constructs of this area of research, as well as to present the validity aspects of design knowledge.

We have posted a template of this visual abstract, and we welcome comments as well as examples of how this abstract can be applied to your research!

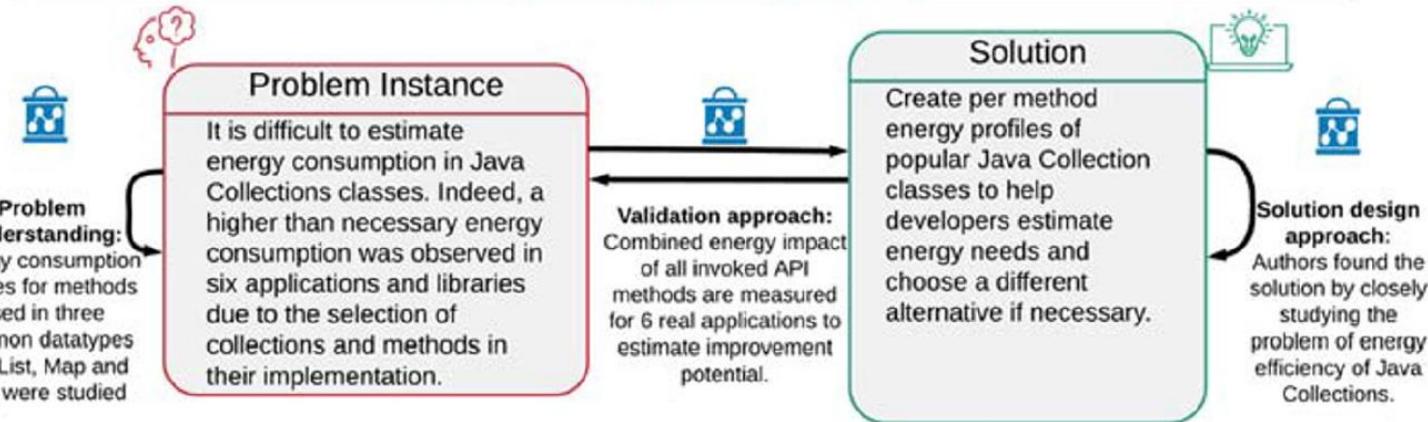


dfucci Davide Fucci

Sharing findings with
practitioners
DSSE.org
<https://github.com/margaretstorey/VASE>



Technological rule: To optimize the energy efficiency of software developed in Java use per-method energy profiles



Relevance: The technological rule is relevant for developers wishing to use green programming techniques (in particular to select among methods in Java Collections)

Rigor: Authors study if what they find also applies to other cases by checking if using the profiles actually helps improve the energy consumption of various applications (e.g., Google Gson, XStream and K-9 Mail).

Novelty: A new approach for profiling Java Collections in terms of energy consumption.

Paper Title: *Energy Profiles of Java Collections Classes*

Sharing findings with practitioners

DSSE.org

<https://github.com/margaretstorey/VASE>



#171678945



Questions and Comments....

I has a question...



Tony Clear S2 2024

CISE ENSE701



63

Review Techniques

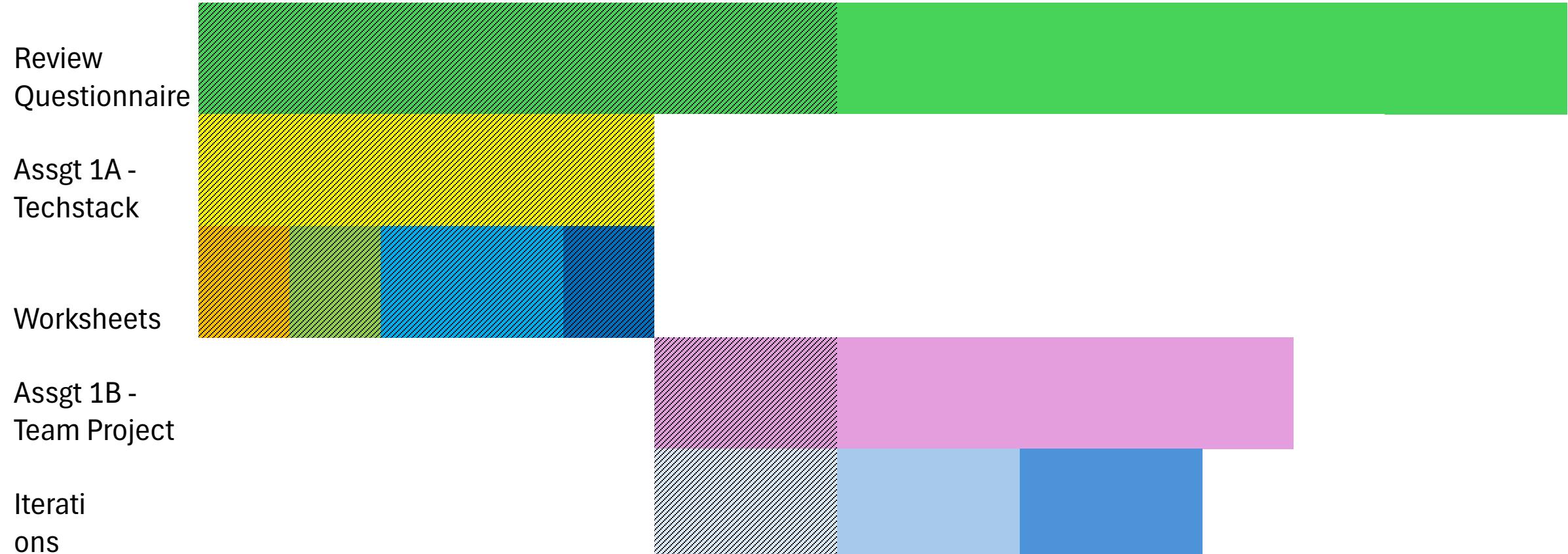
Week 8



Taking Stock

Week
No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



Code Quality

What are the quality criteria to test the quality of code against

What are the code quality testing practices to run the tests against these criteria?

What practices will help to ensure test fails do not happen?

Code intentionality is clear – good naming conventions

This means it is easy to understand (read) how it works, what it is supposed to do, and easier to change, review, test, debug,

Code structure is high quality – code units are **loosely coupled and highly cohesive**
Object oriented – S.O.L.I.D principles

Do not repeat code – multiple places to change (**DRY Principle**)

Code Reviews will improve code quality

Test Driven Design will improve code quality

Test Automation will improve code quality

Code reviewsSo many benefits!

<https://betterprogramming.pub/5-ways-code-reviews-helped-my-career-8d72aa1d2474>

<https://medium.com/inside-league/how-one-code-review-rule-turned-my-team-into-a-dream-team-fdb172799d11>

DEV community analysis [8,15]

RQ3: Applying empathy in software engineering activities	Communication and Collaboration - practitioners consider empathy useful or important when communicating with colleagues, clients, and users.	software developers play different roles in their organizations...would involve talking to people and wherever you need to deal with people, empathy can play a key role
	Coding - practitioners discuss the need for empathy when they are coding or maintaining the code of other developers,	Something that I learned as the time passes was to have empathy with another developer's code."
	Management and Leadership - practitioners, view the need for empathy to successfully coordinate, communicate, motivate, and work with their teams and colleagues.	"To make an impact, our SRE leaders need to lead with empathy and help the rest of the organization engineer with empathy."
	Code review - practitioners consider empathy necessary in the code review process	Empathy for other engineers - ... Be mindful that...asking for their input is essentially asking them for their time

Contemporary Issues in SE	ENSE701	Assignment 1B
16.→ Screenshots of your team GitHub repository showing any branches and all users (including all tutors) and the initial code for the MNNN stack setup	<input type="checkbox"/>	<input type="checkbox"/>
17.→ Screenshots of your team GitHub Actions files showing your Integration automation pipeline	<input type="checkbox"/>	<input type="checkbox"/>
18.→ Screenshots of ONE local development environment using VS Code, showing the initial folder structure - including .gitignore, .env, the frontend folder (packages.json) and the backend (packages.json) file.	<input type="checkbox"/>	<input type="checkbox"/>
19.→ Screenshots of your team MongoDB Atlas setup	<input type="checkbox"/>	
20.→ A diagram with explanations (can be hand drawn) of your developer process for each developer to follow to get their code deployed. It should include: a. → Your team standards for feature branches -- where and naming conventions b. → Your team standards for commits -- how often and format for commit messages c. → Your team expectations about how often to push to GitHub and when to merge feature branches with the production code d. → Your team process for automation of unit testing, linting, manual code reviews BEFORE merging with production (i.e. continuous integration) and the use of pull requests. e. → Your team process for deployment to Vercel	<input type="checkbox"/>	

How to have high quality code reviews

<https://medium.com/better-programming/how-to-review-code-in-7-steps-98298003b7ec>

<https://betterprogramming.pub/5-rules-for-every-code-review-98bf60dd5dbe>

<https://curtiseinsmann.medium.com/ive-coded-reviewed-over-750-pull-requests-at-amazon-here-s-my-exact-thought-process-cec7c942a3a4>

<https://medium.com/swlh/3-problems-to-stop-looking-for-in-code-reviews-981bb169ba8b>

Inspections

Fagan, M. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 3, 182-211.

<https://www.proquest.com/openview/dd282e91ad39c894cc36b0ec56c23c7f/1?pq-origsite=gscholar&cbl=35072>

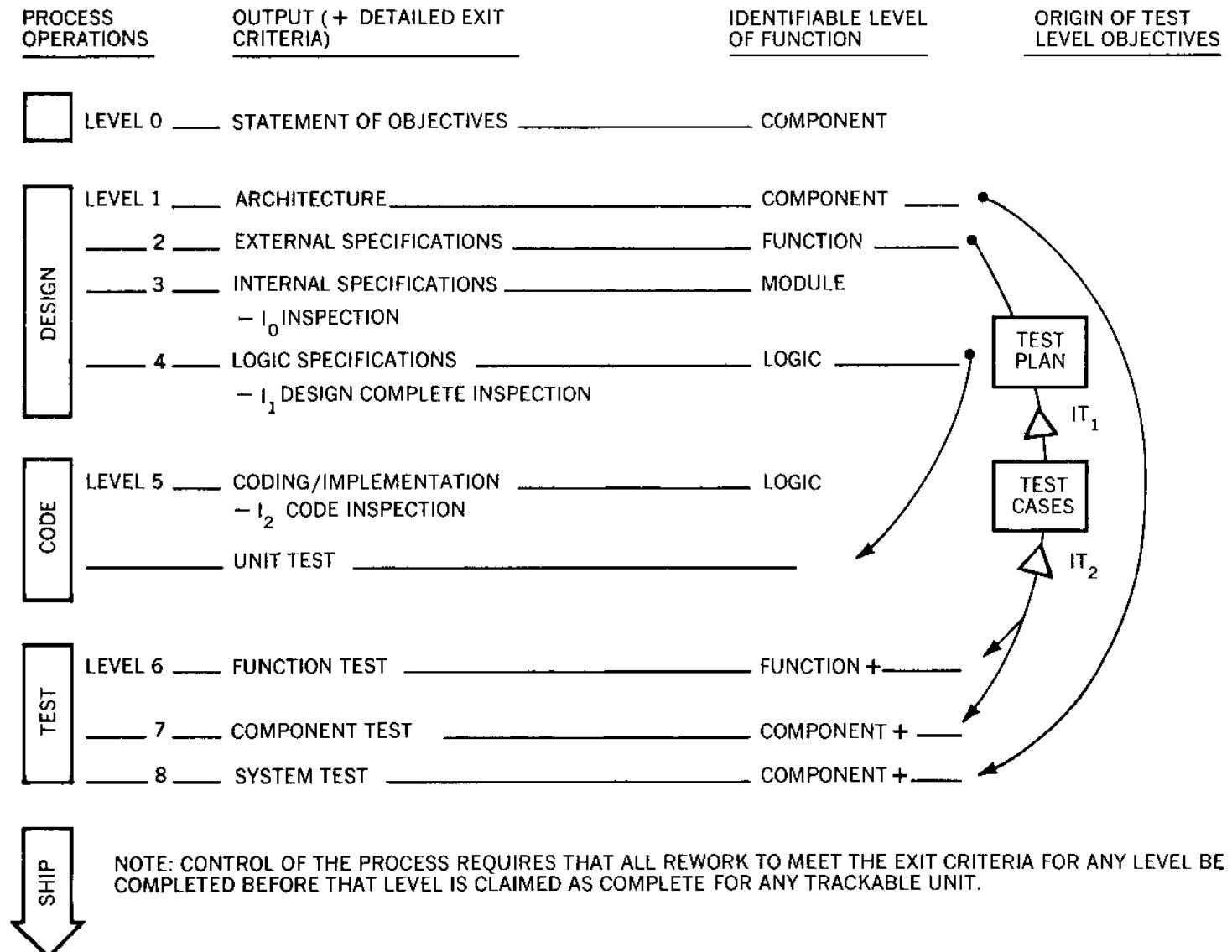
https://link.springer.com/chapter/10.1007/978-3-642-59412-0_35

Glass, R. (1999). Inspections - Some Surprising Findings. *Communications of the ACM*, 42(4), 17-19.

<https://dl-acm-org.ezproxy.aut.ac.nz/doi/pdf/10.1145/299157.299161>

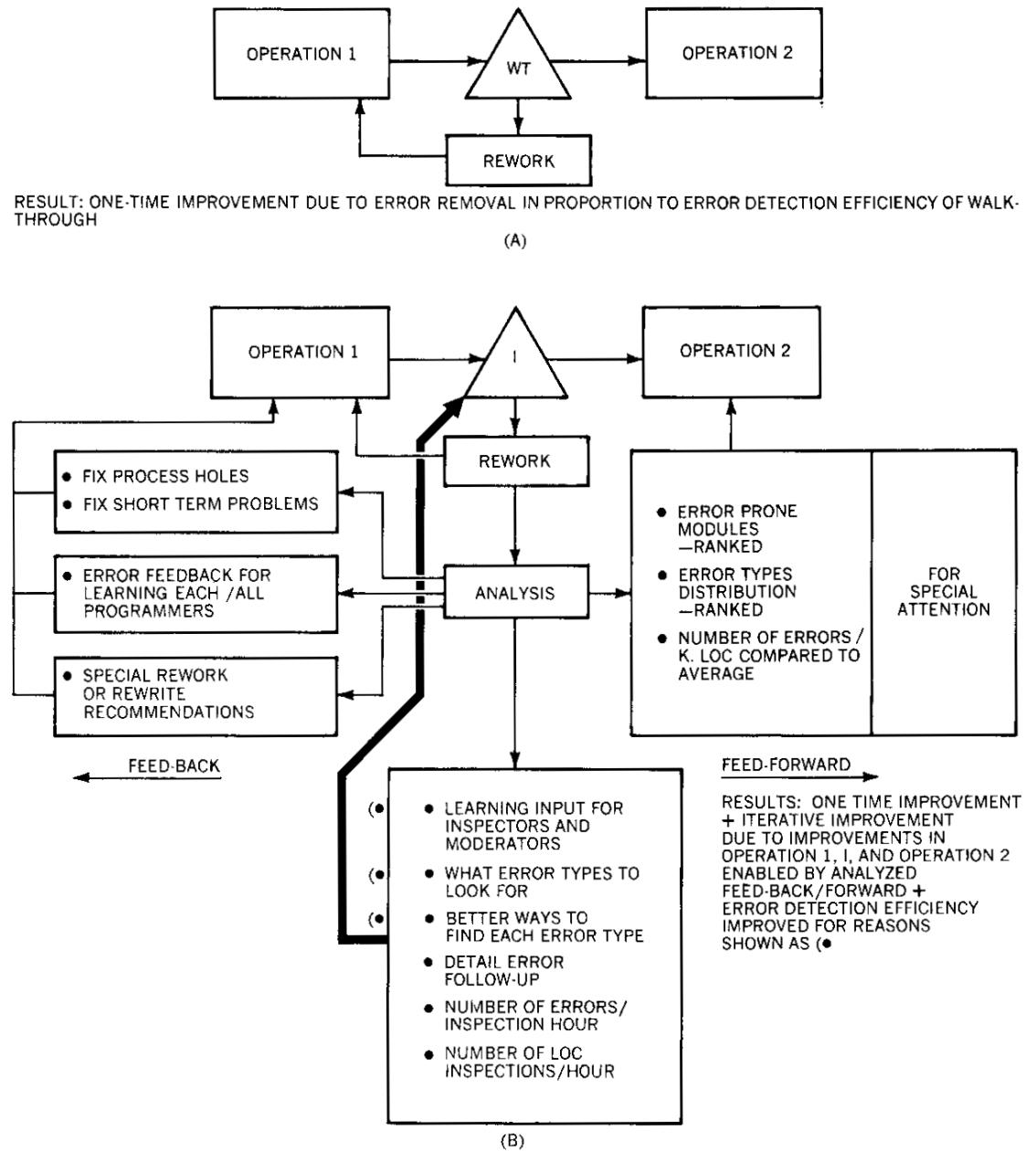
Fagan Inspections

Figure 1 Programming process



Fagan Inspections vs. Walkthroughs

Figure 10 (A) Walk-through process, (B) Inspection process



Why it is important to have well-crafted clean code?

Quality software is developed in teams

Other people will need to read and understand how your code works to extend it, debug it, change it or remove it.

You may need to do the same a day later, two weeks later, 6 months later

THINK ABOUT WHO WILL COME NEXT!
BE A GOOD TEAM MATE!

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. — Martin Golding

So how can I craft my code so it is easier for me and others to understand how it works?

Pull Requests and Documentation

<https://betterprogramming.pub/why-every-git-commit-message-must-include-its-commit-context-1171c0b2f710>

<https://dev.to/holderburato/patterns-for-writing-better-git-commit-messages-4ba0>

Integration of Code - workflow

What are the steps during a day for a developer working in a team of developers with a shared code base in GitHub to work on code integrate their code

What does “Continuous” integration mean?

Pull latest version of working branch to local repo

Work on it writing test and functional code, running tests

Commit frequently with informative commit messages to local repo

Push code to working branch in GitHub and merge after checking all tests pass locally

~~Run integration tests on GitHub for merged code~~

Pull request merge with Develop or Main branches

Collaborator reviews, discusses, runs integration tests, if passes – merge to Develop/Master

Reflective Practice and Reviews

The role of reflective practice in increasing your professional effectiveness – putting reviews in context

<https://youtu.be/M9hyWVEG2x0?si=VAAT65bmY5rPzCjB>

Forms of action and reflection – not just doing, but thinking on what and how you are doing

<https://www.youtube.com/watch?v=x2MfNE91jLk>

Schön, D. (1987). *Educating the Reflective Practitioner*. Jossey Bass.

Automating Continuous Integration (and Deployment) Embedding Review

Based on some **trigger** (e.g. Pull request to merge into Main or Develop)

Take some steps **automatically** to **check the code will integrate** with the code to be merged into

Build the code (compile?)

Check the code meets some rules (linter)

Run all the **unit tests** for the entire code branch as if it is merged

Do the Merge

CD

If the merge is to the Main – deploy – **release it-** to the users

CI Servers

They will follow watch for the trigger specified

Follow the commands to run automatically

Usually stored in a YAML file

Github actions workflow

<https://github.com/actions/starter-workflows/blob/main/automation/manual.yml>

4 ways we use GitHub Actions to build GitHub - The GitHub Blog

<https://images.app.goo.gl/QVxMe427dviFDncX7>

Collaborative Programming Practices

The Development Cycle

In pair programming, two programmers jointly produce one artifact (design, algorithm, code). The two programmers are like a unified, intelligent organism working with one mind, responsible for every aspect of this artifact. One partner, the driver, controls the pencil, mouse, or keyboard and writes the code. The other partner continuously and actively observes the driver's work, watching for defects, thinking of alternatives, looking up resources, and considering strategic implications. The partners deliberately switch roles periodically. Both are equal, active participants in the process.

L. Williams, R. R. Kessler, W. Cunningham and R. Jeffries,
"Strengthening the case for pair programming," in *IEEE Software*, vol. 17, no. 4, pp. 19-25, July-Aug. 2000, doi:
[10.1109/52.854064](https://doi.org/10.1109/52.854064)

What Is eXtreme Programming?

eXtreme Programming is a software development method that favors informal and immediate communication over the detailed and specific work products required by many traditional design methods. Pair programming fits well within XP for reasons ranging from quality and productivity to vocabulary development and cross training. XP relies on pair programming so heavily that it insists all production code be written by pairs.

XP consists of a dozen practices appropriate for small to midsize teams developing software with vague or changing requirements. The methodology copes with change by delivering software early and often and by absorbing feedback into the development culture and ultimately into the code.

Several XP practices involve pair programming:

- Developers work on only one requirement at a time, usually the one with the greatest business value as established by the customer. Pairs form to interpret requirements or to place their implementation within the code base.
- Developers create unit tests for the code's expected behavior and then write the simplest, most straightforward implementations that pass their tests. Pairs help each other maintain

the discipline of writing tests first and the complementary, though quite distinct, discipline of writing simple solutions.

- Developers expect their intentions to show clearly in the code they write and refactor their code and others' if necessary to achieve this result. A partner who has been tracking the programmer's intention is well equipped to judge the program's expressiveness.
- Developers continuously integrate their work into a single development thread, testing its health by running comprehensive unit tests. With each integration, the pair releases ownership of their work to the whole team. At this point, different pairings can form if another combination of talent is more appropriate for the next piece of work.

To learn more, see Kent Beck's book,¹ or consult the eXtreme Programming Roadmap at xp.c2.com, where a lively community debates each XP practice.

Reference

1. K. Beck, *eXtreme Programming Explained: Embrace Change*, Addison Wesley Longman, Reading, Mass., 2000.

Mob Programming

First coined in the Extreme Programming (XP) community in **2003** by Moses Hohman to describe their practice of code refactoring in a group of more than two.



The team took “Mob Programming” to the next level.



Mobbing just extends the benefits of pair programming with more people working on a coding problem?

Agile leadership in mob programm

Mob Programming



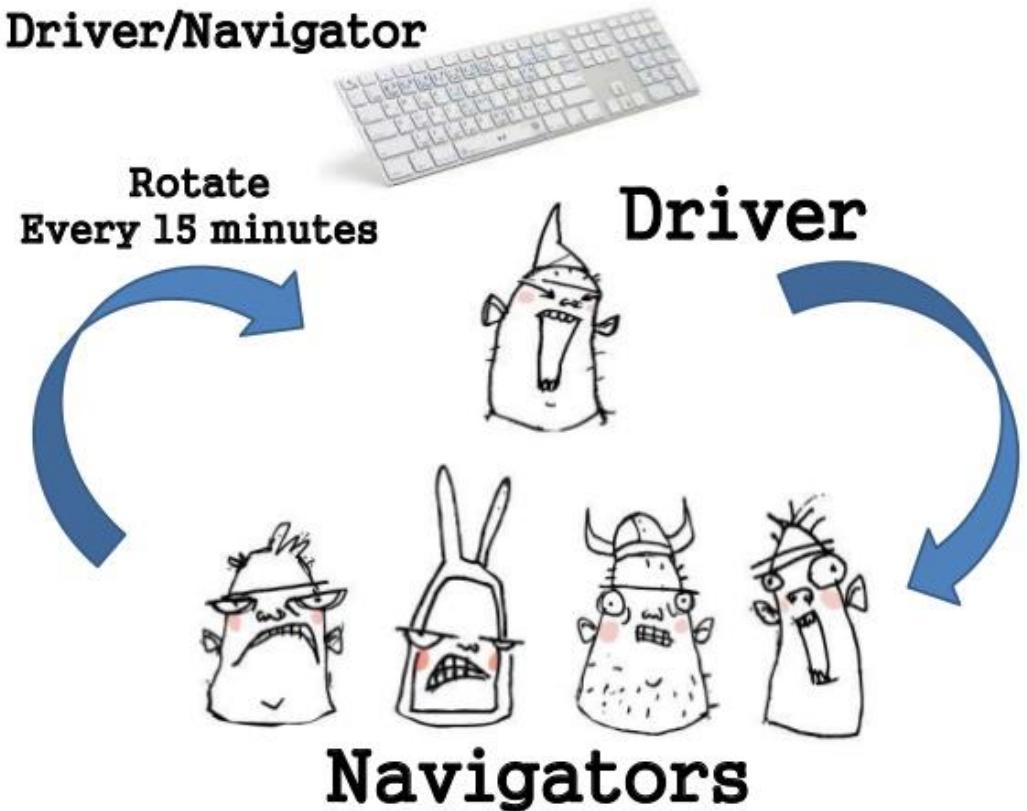
Woody Zuill began popularizing it again from **2013**

Mob Programming

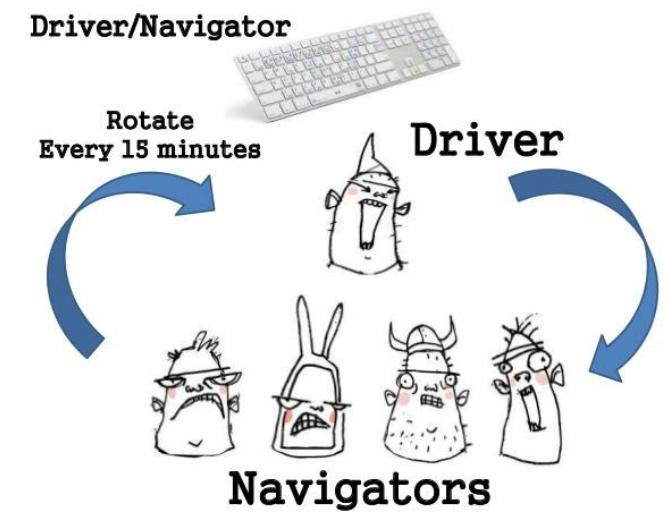
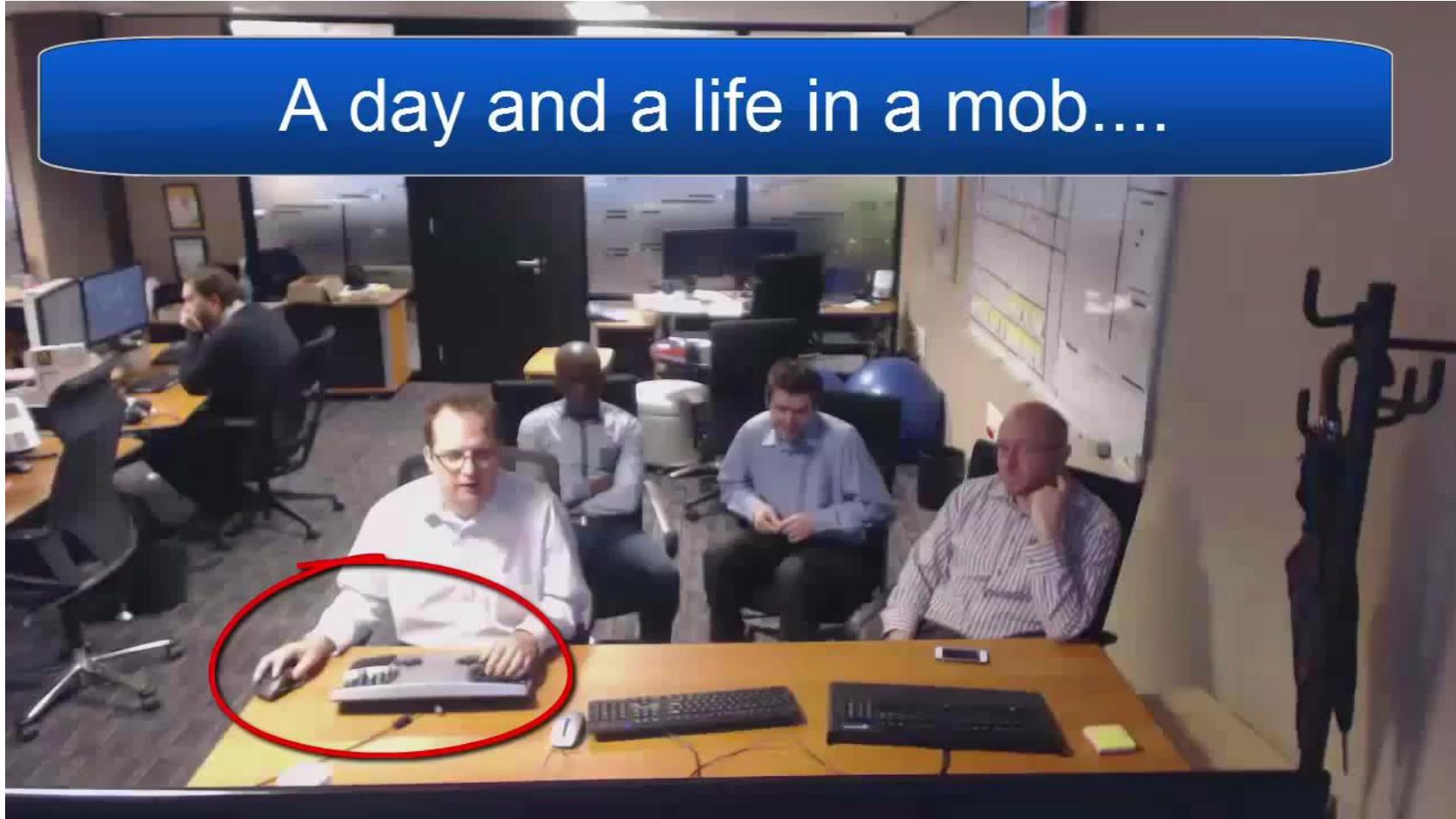
All the brilliant minds working on
the same thing...
at the same time...
in the same space...
on the same computer...

Just like a real mob.

Driver/Navigator



Mob Programming



Seemed to have lots of side benefits and became the usual way of working for some teams

The Observed Benefits



Team code ownership emerged naturally



Individual have broader knowledge of the code base and front-end/back-end

- work shared more easily
- design decisions better informed
- critical knowledge loss by absence of individual less likely-



Increased confidence in quality of code

- improved team morale

The Observed Benefits



Consistent use of tools used



Onboarding new team member
quicker



Higher confidence in the predictability
of work effort

Increased productivity longer term



Reduction in multi-tasking



A higher level of code craft



Reduction of work in progress

Increased productivity longer term



Less technical debt

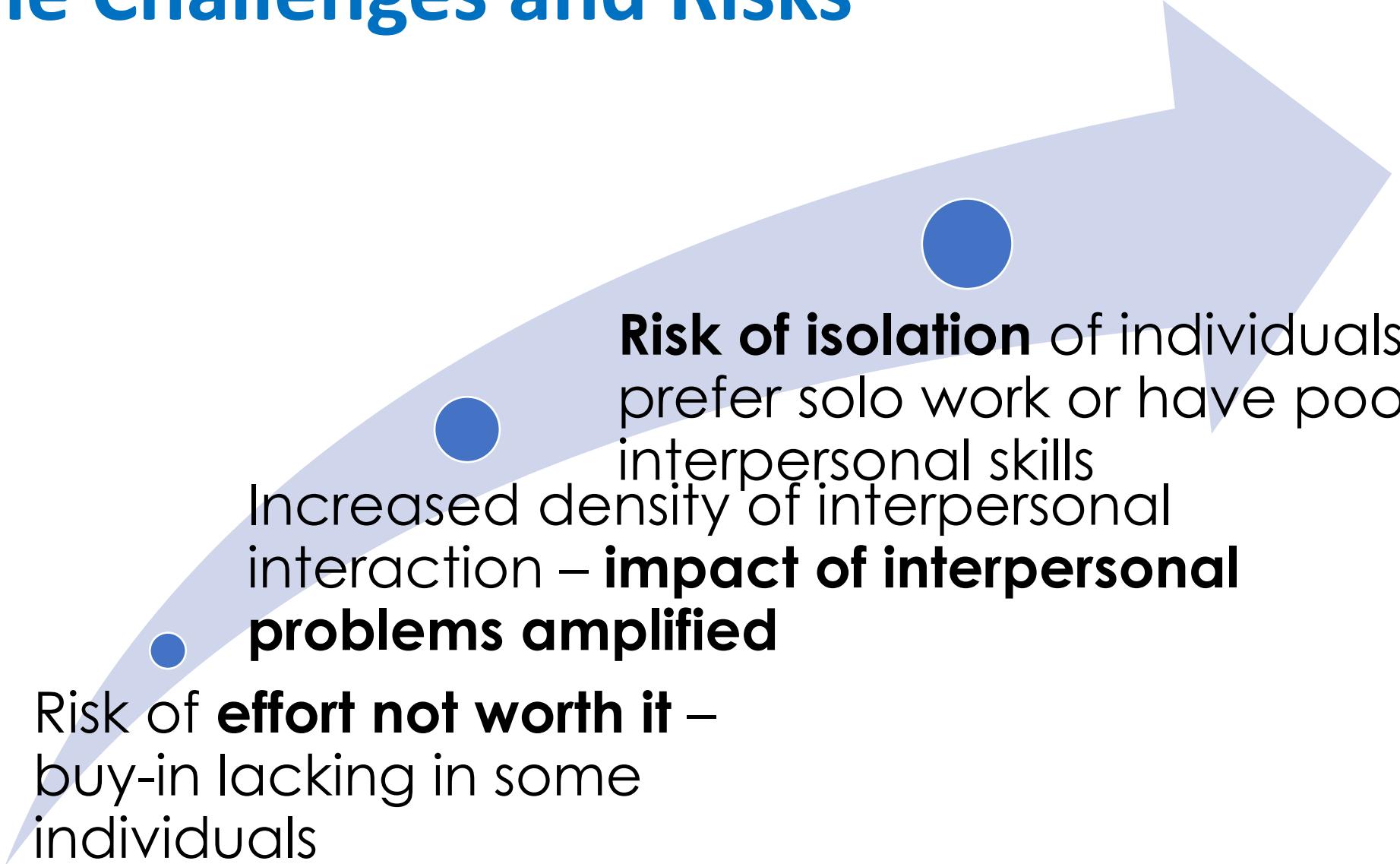


Fewer interruptions



Fewer delays because of unavailable information

The Challenges and Risks

- 
- **Risk of isolation** of individuals who prefer solo work or have poor interpersonal skills
 - Increased density of interpersonal interaction – **impact of interpersonal problems amplified**
 - Risk of **effort not worth it** – buy-in lacking in some individuals

The Challenges and Risks



Suitable Workplace

Finding a suitable consistent work place with a dedicated machine was a challenge



Role of tester in mob

Understanding and stabilizing the role of the QA in mobbing was challenging – QA morale low initially



Slower pace of coding

Pace of code generation slowed – can interrupt mobbing if time pressure dominates short term

Retrospectives

The high-level purpose is to keep learning as a team by reflecting on the last sprint

What can we learn from this that suggests something new to try for the next sprint

Need a process! There are many – find what suits the team

Happiness histogram
Sailboat
Answer questions

Need team members to

All participate

Be honest/candid

FOCUS ON MAKING THE TEAM STRONGER

<http://scrummastertoolbox.libsyn.com/bonus-the-top-3-challenges-to-better-retrospectives-with-david-horowitz>

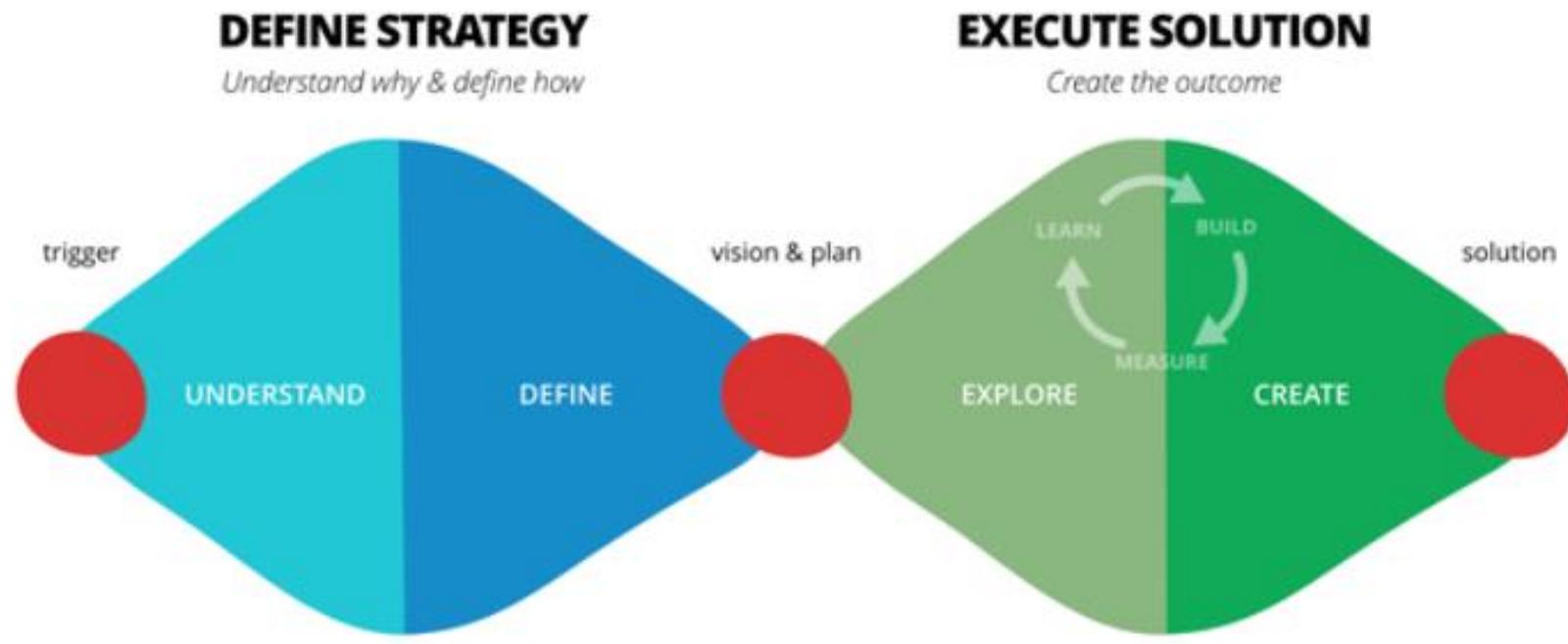
<http://scrummastertoolbox.libsyn.com/how-to-find-what-agile-retrospective-format-works-for-your-team-justin-chapman>

https://www.ponolabs.com/labs/wp-content/uploads/2019/03/Team_Canvas_v5.pdf

Retrospectives

Double diamond decision making

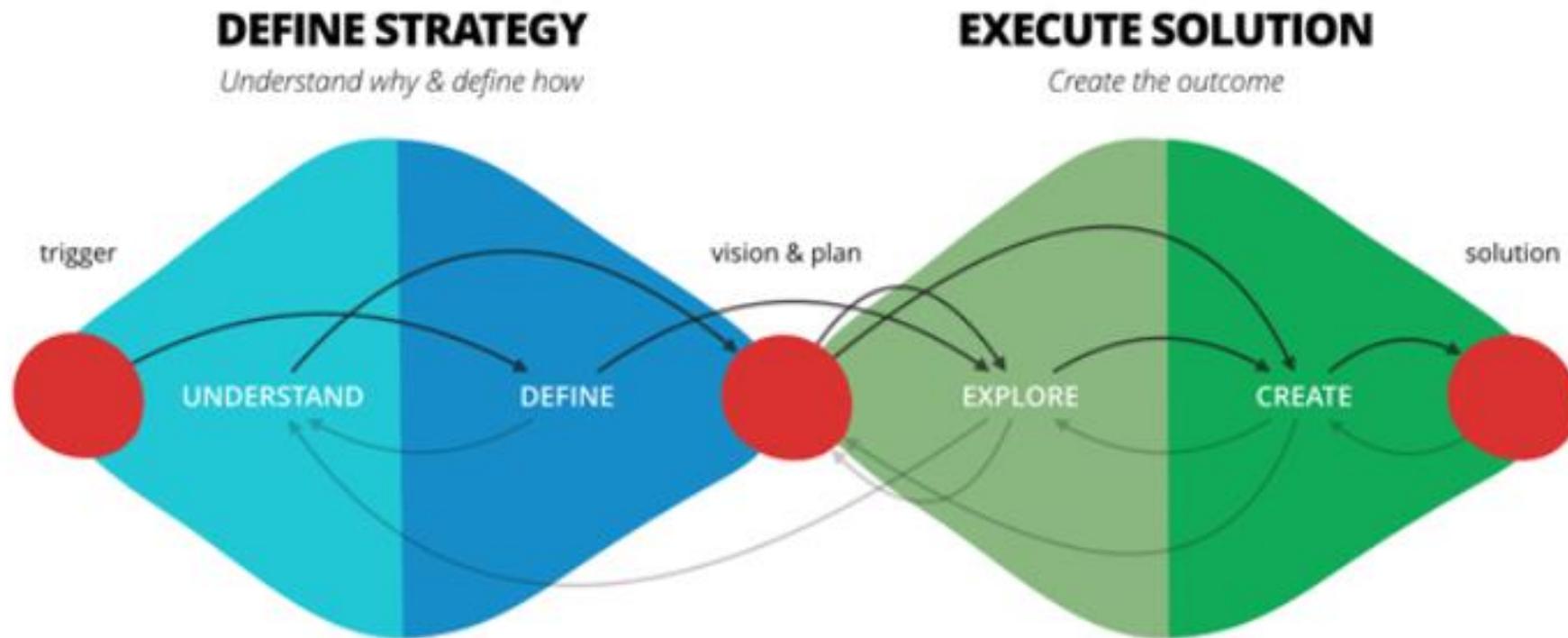
<https://www.thoughtworks.com/insights/blog/double-diamond#:~:text=The%20'Double%20Diamond'%20process%20maps,of%20thinking%20that%20designers%20use.&text=It%20describes%20significant%20up%2Dfront,to%20produce%20a%20final%20solution>



Double Diamond Thinking

Create

As we gain confidence in the solution, exploration gives way to engineering. Now we're creating and optimising working software. The opportunity here is two-fold. First, a working solution delivered to market. Second, we gather real market feedback. As a result, our understanding deepens, and new discoveries influence an ever-changing strategy. Software engineering is not merely execution of a plan, it also defines strategy.



Retrospectives

Divergent thinking

What can we learn (impediment)

ICE analysis (Impactful Confidence Effort)

Rank based on this – T-shirt sizes

Lean Coffee
(dot vote)

What are some options to try

Groan Zone

Convergent thinking

Some high value learning

One or more things to change for the next sprint

<https://miro.com/templates/mad-sad-glad-retrospective/>

Retrospectives

Top Issue

To get the team to converge on practical and impactful action items from the retro

Practical actions that team will take ownership for and implement in the next sprint and will make a difference

A retro that leads to no change is worse than a waste of time
No improvement and no value to the retro meeting

The retro needs follow through -> people get engaged->solves problems!

<https://www.mountaingoatsoftware.com/blog/a-simple-way-to-run-a-sprint-retrospective>

Tools to help online Retro

padlet

retrotool.io

<https://retrotool.io/>

retrium

https://www.infoq.com/articles/remote-retrospective-engage/?utm_source=notification_email&utm_campaign=notifications&utm_medium=link&utm_content=content_in_followed_topic&utm_term=daily

Software Process Improvement

Fundamental differences between traditional and agile software development regarding SPI[5].

First,

while SPI in the plan driven perspective **prescribes norms** for how the individual, team and organization should operate,

agile software development address the improvement and management of software development **practices within individual teams** [2].

In agile development, **processes are not products**,

but rather practices that evolve dynamically with the team as it adapts to the particular circumstances [21].

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

Software Process Improvement (2)

Second,

plan-driven methods, such as the waterfall model, usually adopt a **top-down approach** for improving the software development process [5],

while the agile view has a **bottom-up approach**.

Third, SPI in plan-driven development often emphasizes the **continuous improvement** of the organizational software process for future projects,

while the principles of agile software development focus on **iterative adaption and improvement in the on-going projects**.

Short development cycles provide **continuous and rapid loops to iterative learning**, to enhance the process and to pilot the improvement.

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

Software Process Improvement (3)

When doing agile development, there are typically two meetings where the team focuses on improving the process.

- 1) Daily meetings. In the daily meeting the team members are supposed to coordinate their work and focuses on solving problems that stop the team from working effectively.

In Scrum, the Scrum-master is supposed to facilitate this meeting and making sure impediments to the process are removed

- 1) Retrospective [22]. At the end of each iteration, a retrospective is held. In this meeting the team focuses on what was working well and what needs to be improved. Measures are then taken.

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

SPPI – Diagnosis & Planning

Table 3. Factors in the team radar diagnosis instrument

Factor	Description
Shared leadership	Leadership is rotated to the person with key knowledge, there is jointly shared decision authority.
Team orientation	Priority is given to team goals more than individual goals, team members respect other members' behaviour.
Redundancy	Members have multiple skills so that they can perform (parts of) each others tasks.
Learning	The team develops shared mental models, and a capacity for learning to allow operating norms and rules to change.
Autonomy	The ability to regulate the boundary conditions of the team, the influence on management (and other externals) on activity.

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

SPPI – Study Context

Table 1. Properties of the maintenance and development team

Context	“Maintenance”	“Development”
Type of system	Web-based	Back-end of large system
Technology	Primarily Java	C and C++
Project size	140.000 lines of code, and several, open-source modules	3.000.000 lines of code
Project phase	Maintenance and adding new functionality	New development
Project length	Started in 2008, handed over to customer fall of 2009.	Started in early 1990's, still ongoing.
Team size	Five: One senior and four junior developers	Eight senior developers
Team composition	Almost eight months	Almost four months

Table 2. Agile practices in the two teams

Agile practice	“Maintenance”	“Development”
Iterative development	Yes	Yes
Continuous integration	Yes	No
Sprint planning	No	Yes
Sprint demo	No	Yes
Sprint retrospective	No	Yes
Daily standup	No	Yes
Self-managing team	Yes	Yes
Refactoring	Yes	Yes
Co-location	Yes	Yes
Pair-programming	2 people	No

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

SPPI – Diagnosis & Planning

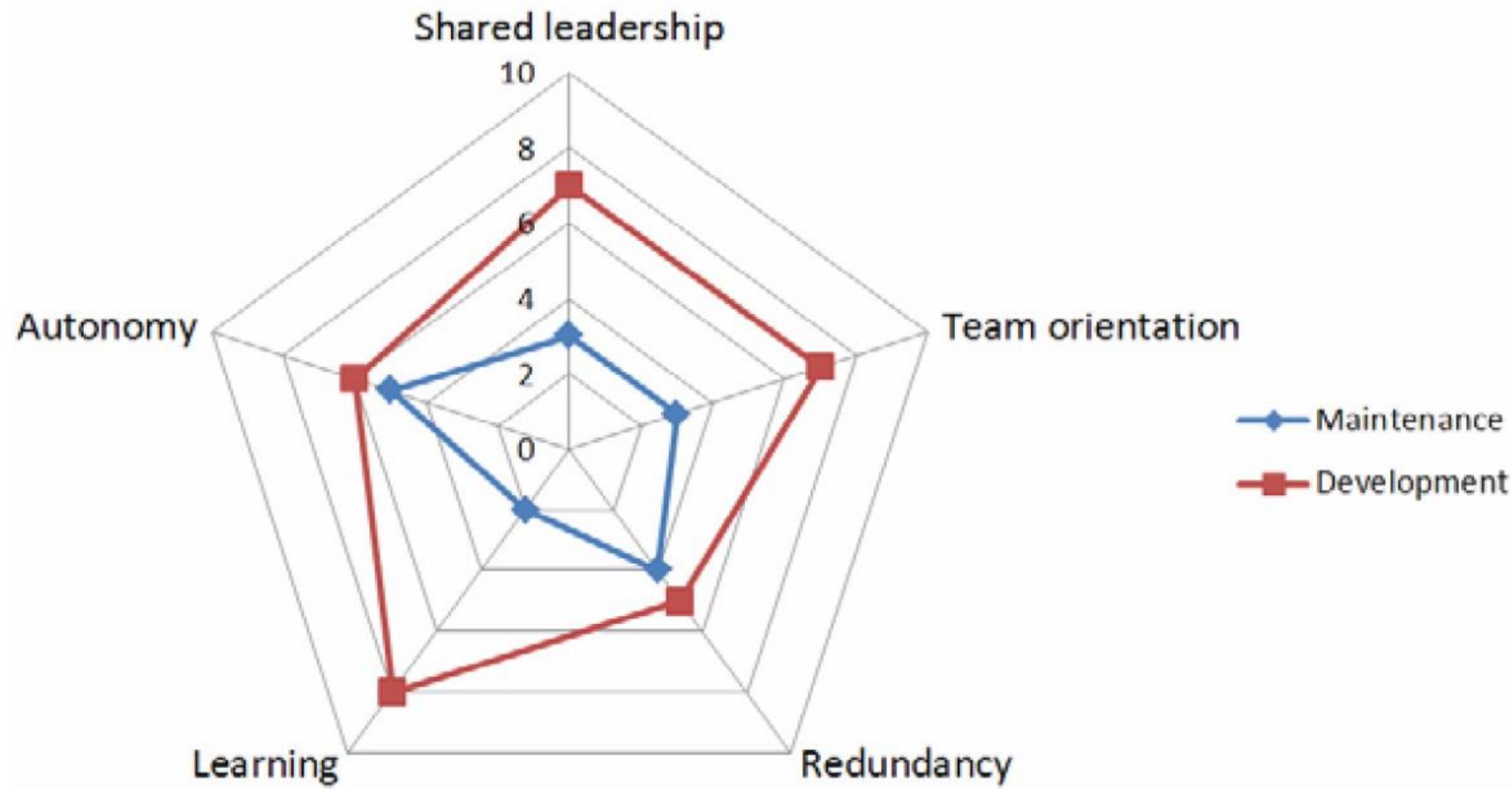


Fig. 1. A plot of teamwork characteristics of the two teams

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

SPPI –Planning: Mtce Team

To improve teamwork in the two teams,

- presented results of diagnosis phase,
- discussed priority on teamwork factors together with the teams.

As a result concrete measures to improve development processes and teamwork were suggested.

For the maintenance team we observed challenges related to *shared leadership, team orientation, and learning*.

- As for leadership, team dominated by junior developers, little involvement of the team in leadership and little process in place.
- Team heavily specialized, - team members working on independent modules, again lowered team orientation.
- Finally, team had no arenas for learning except for being in the same room, but observation showed little discussion and feedback on the actual work tasks the team members were involved in.

SPPI –Planning: Mtce Team

In a workshop, we presented the scores, problems and consequences to the team.

Team decided to reintroduce important agile practices they had stopped doing. In prioritized order:

- Sprint retrospective to improve learning. Team members would be able to give feedback and improve both the development process as well as the product.
- Daily stand-up meetings to improve coordination of tasks, team communicating, and solve problems daily. The meeting was expected to have an effect on shared leadership, team orientation and learning.
- Code review to improve software quality, learning and increase redundancy.

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.

SPPI –Planning: Dev Team

The development team got higher scores on all factors compared to the maintenance team. The team prioritized to improve the problems with the highest potential for the team: *inefficient sprint planning, variable ownership to project goals, and not solving process related problems in the retrospective.* The following actions were suggested:

- Open space sprint planning, to conduct sprint planning more efficiently. The sprint planning meetings in the team were dominated by specialists and long lasting. Using the open space process, the team members would suggest topics to discuss and then several discussions could happen in parallel in the same room. Team members are encouraged to walk between discussions. This action was expected to improve shared leadership and team orientation.
- Pair programming to improve team orientation. Making people to work closely together constantly giving feedback could also improve shared decision-making and improve learning.
- Collocating the team in the same room, would improve communication and oversight, and improving team orientation.

SPPI in Agile – A Technique for Diagnosis & Planning?

Now we return to our research question, “*how to efficiently improve teamwork in agile software development?*”

We have shown results from using diagnosis with the team radar and action planning in a small and immature team and in a large and more mature team.

Both the teams perceived the diagnosing and the outcome as something they learned from, because it illuminated issues they had seen individually but not discussed within the team.

It is not enough to do retrospectives if the team is not able to discuss the cause of the problems they are experiencing.

SPPI in Agile – A Technique for Diagnosis & Planning?

This study indicates that

process improvement, although a central concept in agile development is still hard to achieve.

The main implication for practice is that

this study with two teams reveals that *process improvement does not happen by itself even in agile methods*, there needs to be effort invested to actively experiment with solutions.

Ringstad, M. A., Dingsøyr, T., & Moe, N. B. (2011). Agile process improvement: diagnosis and planning to improve teamwork. In *European Conference on Software Process Improvement* (pp. 167-178): Springer.



#171678945



Questions and Comments....



Tony Clear S2 2024

I has a question...

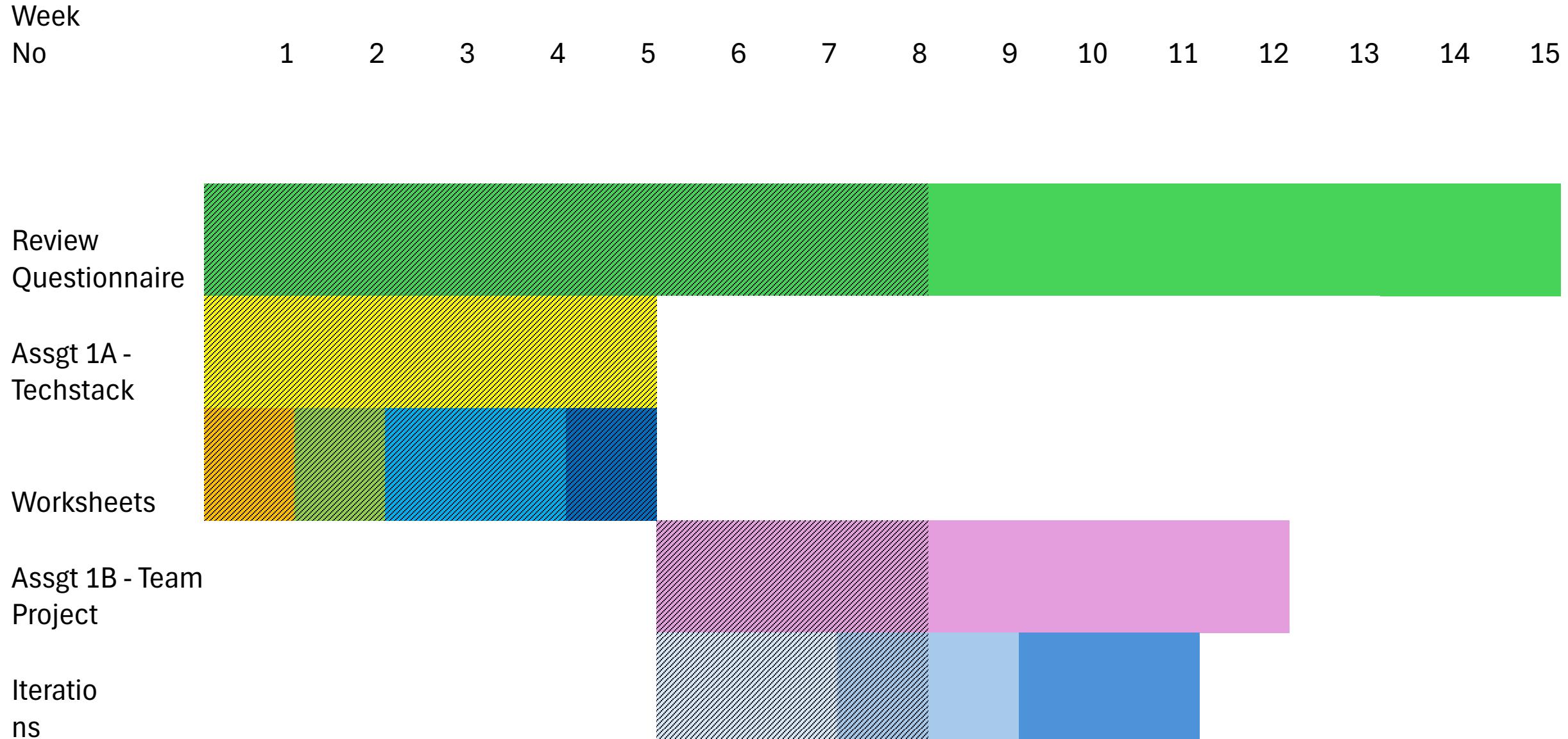


Code Craft and Code Quality

Week 9



Taking Stock



Quality of the code

Behaves as expected – no bugs **Unit tests as a “contract”**

Easy to change

Easy to understand how it works and change

The intention of the code is clear

Naming and structure

Change is predictable – impact is limited

Small code structures

Techniques with strong empirical and anecdotal evidence of improving code quality

Test Driven Development

CI/CD

Pair/mob programming

Code Reviews

Explicit Coding standards –Static code checkers - Linters, SonarQube etc

Quality of the product

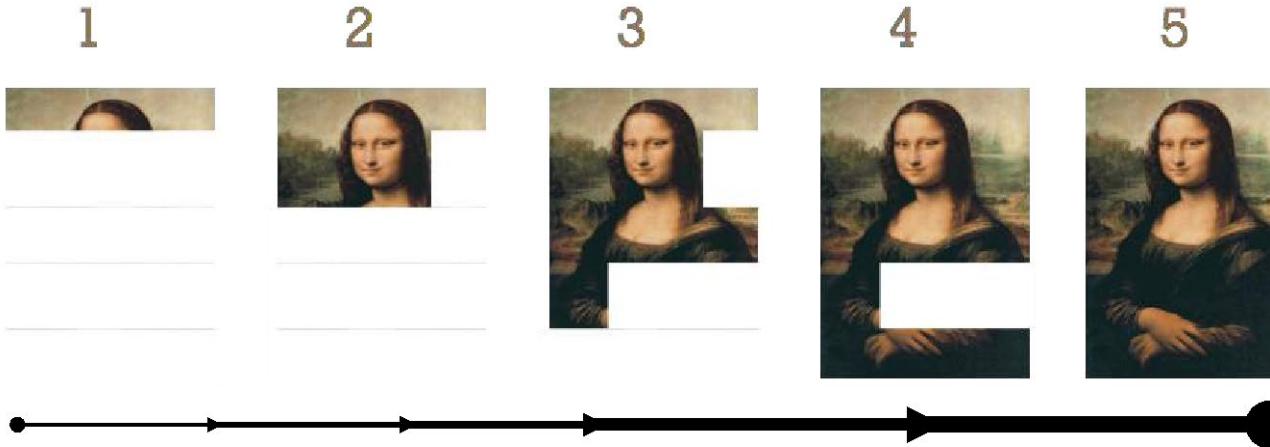
Is useful to users – solves a problem

Behaves as expected

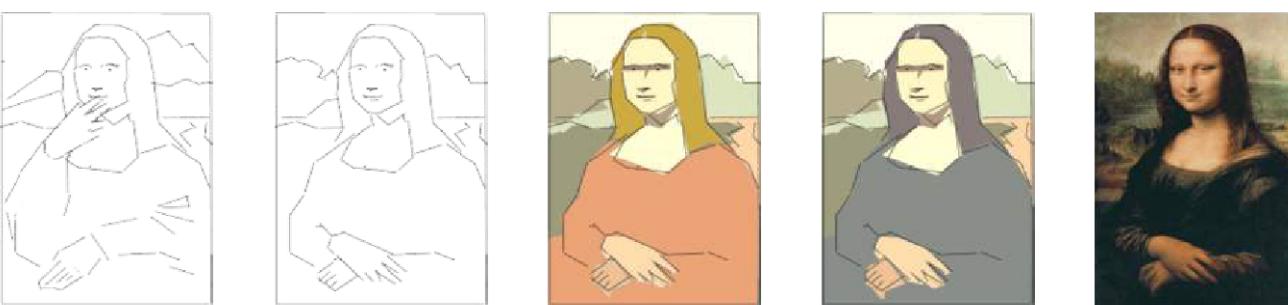
Specifications and Scenarios Based on Acceptance criteria

Coding is a craft

Crafting code is different to just writing it!

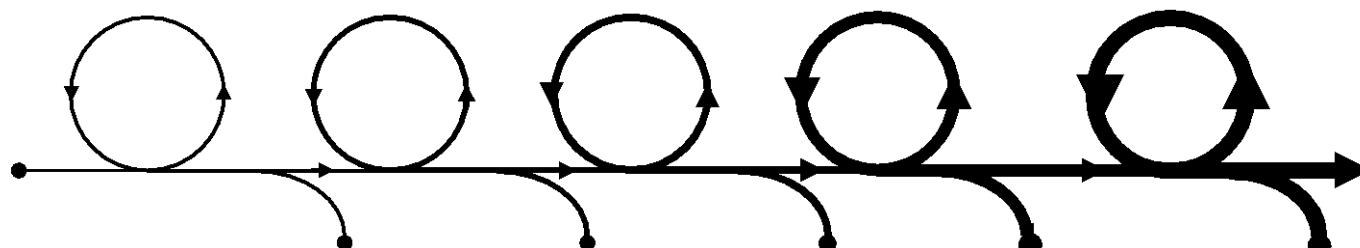


We do NOT create a full design then build from the ground up until we have the finished product.



We start with a sketch, iteratively adding detail.

We revise, extend and refine - working at different levels of abstraction until the software meets someone's needs.



Software is never really finished

Why it is important to have well-crafted clean code?

Quality software is developed in teams

CODE is read more often than it is written

Other people will need to read and understand how your code works to extend it, debug it, change it or remove it.

You may need to do the same a day later, two weeks later, 6 months later

THINK ABOUT WHO WILL COME NEXT!
BE A GOOD TEAM MATE!

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. — Martin Golding

So how can I craft my code so it is easier for me and others to understand how it works?

Code (and think) small!

"The first rule of functions is that they should be small. The second rule of functions is that they should be smaller than that." — Robert C. Martin

Function bodies should rarely be more than 20 line long and mostly less than 10 lines

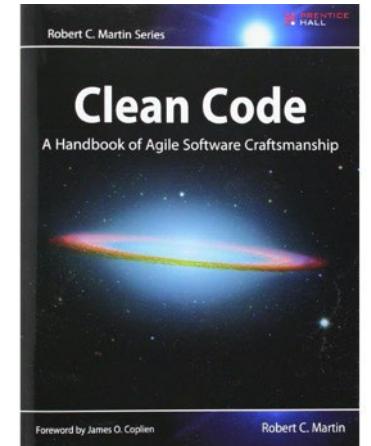
Functions should take as few arguments as possible, preferably none

Functions should do one thing — and do it well

Classes should be sized so they are responsible for one thing only

The Single Responsibility Principle (SRP) – the "S" in SOLID principles

Easier to follow and understand
– low cognitive load



e.g. a function that fetches, manipulates and stores data should be split into three smaller functions

WARNING: Too many tiny classes can be difficult to understand and change

Make code Self-documenting (readable, its intention is clear, understandable)

“Clear and expressive code with few comments is far superior to cluttered and complex code with lots of comments.” — Robert C. Martin

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) &&
(employee.age > 65))
```

Do NOT use magic numbers
65 should be replaced with
Const minAgeForBenefits = 65

Gets refactored to :

```
if (employee.isEligibleForFullBenefits())
```

- The comment is removed
- The conditional logic is encapsulated into a method
- Because a method is used and not a free-standing function, instance variables can be used, creating a zero-argument method call
- The method is given a descriptive name, making its responsibility super clear

<https://medium.com/better-programming/clean-code-5-essential-takeaways-2a0b17ccd05c>

Example of readability of a function

```
const handleSubmit = (event) => {
  event.preventDefault();
  NoteAdapter.update(currentNote)
    .then(() => {
      setCurrentAlert('Saved!')
      setIsAlertVisible(true);
      setTimeout(() => setIsAlertVisible(false), 2000);
    })
    .then(() => {
      if (hasTitleChanged) {
        context.setRefreshTitles(true);
        setHasTitleChanged(false);
      }
    });
};
```

```
const showSaveAlertFor = (milliseconds) => () => {
  setCurrentAlert('Saved!')
  setIsAlertVisible(true);
  setTimeout(
    () => setIsAlertVisible(false),
    milliseconds,
  );
};

const updateTitleIfNew = () => {
  if (hasTitleChanged) {
    context.setRefreshTitles(true);
    setHasTitleChanged(false);
  }
};

const handleSubmit = (event) => {
  event.preventDefault();
  NoteAdapter.update(currentNote)
    .then(showSaveAlertFor(2000))
    .then(updateTitleIfNew);
};
```

<https://itnext.io/tips-for-writing-self-documenting-code-e54a15e9de2>

Single responsibility

```
1 class User
2 {
3     void CreatePost(Database db, string postMessage)
4     {
5         try
6         {
7             db.Add(postMessage);
8         }
9         catch (Exception ex)
10        {
11            File.WriteAllText("LocalErrors.txt", ex.ToString());
12            File.WriteAllText("ServerErrors.txt", ex.ToString());
13        }
14    }
15 }
```

CreatePost() can create a new post, log an error in the database, and log an error in a local file

C# code

```
1 class Post
2 {
3     private ErrorLogger errorLogger = new ErrorLogger();
4
5     void CreatePost(Database db, string postMessage)
6     {
7         try
8         {
9             db.Add(postMessage);
10            errorLogger.log(ex.ToString());
11        }
12        catch (Exception ex)
13        {
14            File.WriteAllText("LocalErrors.txt", ex.ToString());
15            File.WriteAllText("ServerErrors.txt", ex.ToString());
16        }
17    }
18
19    class ErrorLogger
20    {
21        void log(string error)
22        {
23            db.LogError("An error occurred: ", error);
24            File.WriteAllText("\LocalErrors.txt", error);
25        }
26    }
27 }
```

In the article Principles of Object Oriented Design, Robert C. Martin defines a responsibility as a 'reason to change', and concludes that a class or module should have one, and only one, reason to be changed.

It's all in the name...

```
const fStuNms = stus.map(s => s.n)      Whaaaaat?
```

```
const filteredStudentNames = students.map(student => {  
  return student.name;  
});
```

- Use intention-revealing names — e.g., `int elapsedTimeInDays`, not `int days`
- Use pronounceable names — e.g., `Customer`, not `DtaRcrd102`
- Avoid encodings — don't use an `m_` prefix for members and [don't use Hungarian notation](#)
- Pick one word per concept — don't `fetch`, `retrieve`, `get` for the same concept

Common naming conventions

If your value is a boolean, start with **is** or **has**, like **isEnrolled: true**

If your value is storing an array, the name should be plural, eg **students**

Numbers should start with **min** or **max** if possible

For functions, there should be a helpful verb in front,
like **createSchedule** or **updateNickname**

Naming standards for Java

<https://google.github.io/styleguide/javaguide.html#s5-naming>

<https://itnext.io/tips-for-writing-self-documenting-code-e54a15e9de2>

Write (and read) Useful Test Descriptions

```
const getDailySchedule = (student, dayOfWeek) => {
```

It retrieves the daily schedule; if the day of the week is a weekend it returns an empty array; if the student has detention it sticks it onto the end of the schedule; and if the student isn't enrolled in the school, it prints a link to a the school website.

```
describe('getDailySchedule tests', () => {
  it("retrieves the student's full schedule", () => {
    it('returns an empty array if given a weekend day', () => {
      it('adds detention if a student got one that day', () => {
        it('prints a school website link if student not enrolled yet', () => {
```

<https://itnext.io/tips-for-writing-self-documenting-code-e54a15e9de2>

Techniques for crafting clean code...

Refactoring is the process of restructuring existing computer code without changing its external behavior.

Test-driven development is a process where requirements are turned into specific test cases, then the code is added so the tests pass.

The process of crafting software might look something like this:

1. Write failing tests that verify the required but unimplemented behaviour.
2. Write some (potentially bad) code that works and makes those tests pass.
3. Incrementally refactor the code, with the tests continuing to pass, making it more clean with each development iteration.

Design Patterns

Software design patterns provide templates and tricks used to design and solve recurring software problems and tasks. Applying time-tested patterns result in extensible, maintainable and flexible high-quality code, exhibiting superior craftsmanship of a software engineer.

<https://www.educative.io/courses/software-design-patterns-best-practices>

Design Patterns have become an object of some controversy in the programming world in recent times, largely due to their perceived ‘over-use’ leading to code that can be harder to understand and manage.

The Gang of Four and 23 Design Patterns

Creational Patterns

- Builder Pattern
- Singleton Pattern
- 🔒 Prototype Pattern
- 🔒 Factory Method Pattern
- 🔒 Abstract Factory Pattern

Structural Patterns

- 🔒 Adapter Pattern
- 🔒 Bridge Pattern
- 🔒 Composite Pattern
- 🔒 Decorator Pattern
- 🔒 Facade Pattern
- 🔒 Flyweight
- 🔒 Proxy Pattern

Behavioral Patterns

- 🔒 Chain of Responsibility Pattern
- 🔒 Observer Pattern
- 🔒 Interpreter Pattern
- 🔒 Command Pattern
- 🔒 Iterator Pattern
- 🔒 Mediator Pattern
- 🔒 Memento Pattern
- 🔒 State Pattern
- 🔒 Template Method
- 🔒 Strategy Pattern
- Visitor Pattern

Common OOP problem and solution patterns

Singleton

The singleton pattern is used to limit creation of a class to only one object. This is beneficial when one (and only one) object is needed to coordinate actions across the system. There are several examples of where only a single instance of a class should exist, including caches, thread pools, and registries.

Factory Method

A normal factory produces goods; a software factory produces objects. And not just that — it does so without specifying the exact class of the object to be created. To accomplish this, objects are created by calling a factory method instead of calling a constructor.

Strategy

Observer

Builder

Adapter

State

<https://www.geeksforgeeks.org/category/design-pattern/>

<https://www.geeksforgeeks.org/software-design-patterns/>

<https://medium.com/educative/the-7-most-important-software-design-patterns-d60e546afb0e>

Code reviews ... another code crafting enabler

<https://medium.com/better-programming/how-to-review-code-in-7-steps-98298003b7ec>

DRY (WET), YAGNI

YAGNI (You Aren't Gonna Need It)

Do not implement something until you are going to need it

DRY (Don't Repeat Yourself)

A piece of code should be implemented in just one place in the source code

You can create a common function or abstract your code to avoid any repetition in your code.

(WET= Write everything Twice!)

Making OOP with a SOLID Design

Introduced by Robert C. Martin (Uncle Bob), in his 2000 paper [Design Principles and Design Patterns](#).
The actual SOLID acronym was, however, identified later by Michael Feathers (“Working with Legacy Code”).

S — Single responsibility principle

every module or class should have responsibility over a single part of the functionality provided by the software.

O — Open/closed principle

utilize inheritance and/or implement interfaces that enable classes to polymorphically substitute for each other

L — Liskov substitution principle

objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

I — Interface segregation principle

no client should be forced to depend on methods it does not use

D - Dependency inversion principle

High-level modules should not depend on low-level modules. Both should depend on abstractions.
Abstractions should not depend on details. Details should depend on abstractions.

SOLID Design Principles

Introduced by Robert C. Martin (Uncle Bob), in his 2000 paper [Design Principles and Design Patterns](#).
The actual SOLID acronym was, however, identified later by Michael Feathers ("Working with Legacy Code").

<https://en.wikipedia.org/wiki/SOLID>

In software engineering, **SOLID** is a [mnemonic acronym](#) for five design principles intended to make [object-oriented](#) designs more understandable, flexible, and [maintainable](#).

[Sandi Metz](#) (May 2009). "[SOLID Object-Oriented Design](#)". [YouTube](#).

[Archived](#) from the original on 2021-12-21.

Retrieved 2019-08-13. Talk given at the 2009 Gotham [Ruby](#) Conference.

<https://www.youtube.com/watch?v=v-2yFMzxqwU>

<https://www.youtube.com/watch?v=6Bia81dl-JE> (check out 9 mins 30 ff.)

Building on SOLID foundations - Steve Freeman & Nat Pryce

Authors of: *Growing Object-Oriented Software, Guided by Tests*

O – Open/closed principle

We can make sure that our code is compliant with the open/closed principle by utilizing inheritance and/or implementing interfaces that enable classes to polymorphically substitute for each other.

```
1 class Post
2 {
3     void CreatePost(Database db, string postMessage)
4     {
5         if (postMessage.StartsWith("#"))
6         {
7             db.AddAsTag(postMessage);
8         }
9         else
10        {
11            db.Add(postMessage);
12        }
13    }
14 }
```

do something specific whenever a post starts with the character '#'.
If we later wanted to also include mentions starting with '@', we'd have to modify the class with an extra 'else if' in the CreatePost() method

```
1 class Post
2 {
3     void CreatePost(Database db, string postMessage)
4     {
5         if (postMessage.StartsWith("#"))
6         {
7             db.AddAsTag(postMessage);
8         }
9         else
10        {
11            db.Add(postMessage);
12        }
13    }
14 }
```

The evaluation of the first character '#' will now be handled elsewhere

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

<https://12factor.net>

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

The 12 Factor App

Roadmap Resources - Topics

Skill Based

<https://roadmap.sh/react>

<https://roadmap.sh/javascript>

<https://roadmap.sh/typescript>

roadmap.sh is a community effort to create roadmaps, guides and other educational content to help guide the developers in picking up the path and guide their learnings.

<https://roadmap.sh/>

Role Based

<https://roadmap.sh/frontend>

<https://roadmap.sh/backend>

e.g. Architectural Patterns - 12 Factor Apps

<https://www.youtube.com/watch?v=FryJt0Tbt9Q>

I. Codebase

One codebase tracked in revision control, many deploys

A twelve-factor app is always tracked in a version control system, such as Git, Mercurial, or Subversion. A copy of the revision tracking database is known as a *code repository*, often shortened to *code repo* or just *repo*.

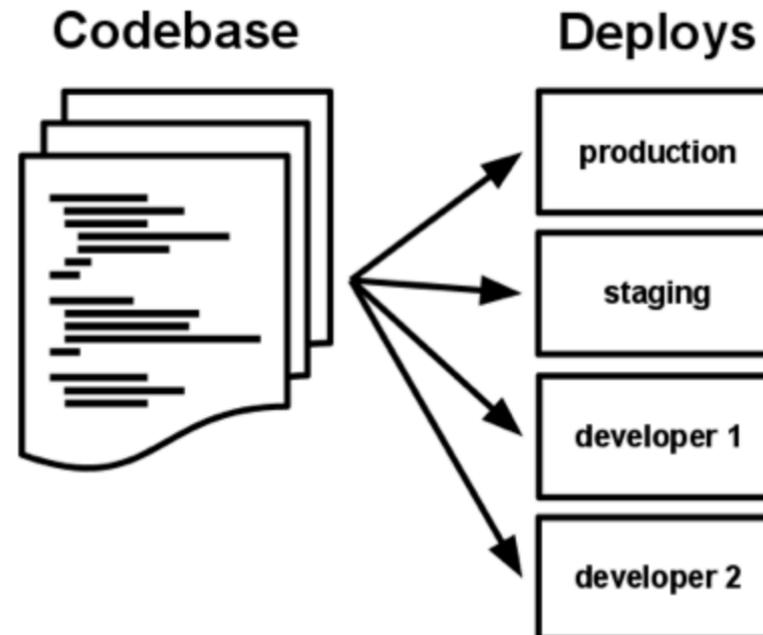
A *codebase* is any single repo (in a centralized revision control system like Subversion), or any set of repos who share a root commit (in a decentralized revision control system like Git).

There is always a one-to-one correlation between the codebase and the app:

- If there are multiple codebases, it's not an app – it's a distributed system. Each component in a distributed system is an app, and each can individually comply with twelve-factor.
- Multiple apps sharing the same code is a violation of twelve-factor. The solution here is to factor shared code into libraries which can be included through the dependency manager.

There is only one codebase per app, but there will be many deploys of the app. A *deploy* is a running instance of the app. This is typically a production site, and one or more staging sites. Additionally, every developer has a copy of the app running in their local development environment, each of which also qualifies as a deploy.

The codebase is the same across all deploys, although different versions may be active in each deploy. For example, a developer has some commits not yet deployed to staging; staging has some commits not yet deployed to production. But they all share the same codebase, thus making them identifiable as different deploys of the same app.



“Clean code is not written by following a set of rules. You don’t become a software craftsman by learning a list of heuristics. Professionalism and craftsmanship come from values that drive disciplines.” — Robert C. Martin

Try to read this sort of stuff every day

<https://medium.com/better-programming/10-must-read-books-for-software-engineers-edfac373821b>

<https://www.makeuseof.com/tag/basic-programming-principles/>

<https://www.geeksforgeeks.org/7-common-programming-principles-that-every-developer-must-follow/>

<https://medium.com/better-programming/clean-code-5-essential-takeaways-2a0b17ccd05c>

<https://medium.com/better-programming/how-to-review-code-in-7-steps-98298003b7ec>

<https://medium.com/young-coder/is-it-time-to-get-over-design-patterns-8851864a6834>



#171678965



Questions and Comments....



Tony Clear 2024 S2

I has a question...



Software Ecosystems

Week 10



Software Ecosystems - Definitions?

an entity where an **ecosystem owner** provides not simply a software product but an underpinning platform.

This platform **offers a set of APIs** through which external developers can connect and build applications.

Common examples can be seen in “App Marketplaces,” such as those provided by:

- Apple through its App Store,
- Google its Play Store and,
- in the New Zealand context, Xero the accounting software company through its APP Marketplace.

The key goal of “software product and platform producing organizations...is to run an innovative continuous software

business with propensity for growth.” [5]

The construction of an ecosystem around a platform aims at achieving what Cusumano has dubbed Staying Power [4], “by minimizing risk, increasing innovation, increasing revenue, and creating a healthy network of partners around the business.” [5]

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: What do we need to know? *ACM Inroads*, 10(2), 18-20. <https://doi.org/10.1145/3395963>

Software Ecosystems - Definitions?

formally a software ecosystem has been defined

“as a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources, and artifacts.” [6]

Clear, T. (2020). THINKING
ISSUES: Software Ecosystems:
what do we need to know?
ACM Inroads, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems –Maturity Models?

a focus area maturity model presents a set of areas of focus,

- which contain capabilities,
- which in turn contain a set of practices,
- within maturity levels and
- result in functional domain capabilities.

These can be implemented as levels of achievement (maturity) and
Institutionalized in an organizational context.

Clear, T. (2020). THINKING
ISSUES: Software Ecosystems:
what do we need to know?
ACM Inroads, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems –Maturity Focus Area?

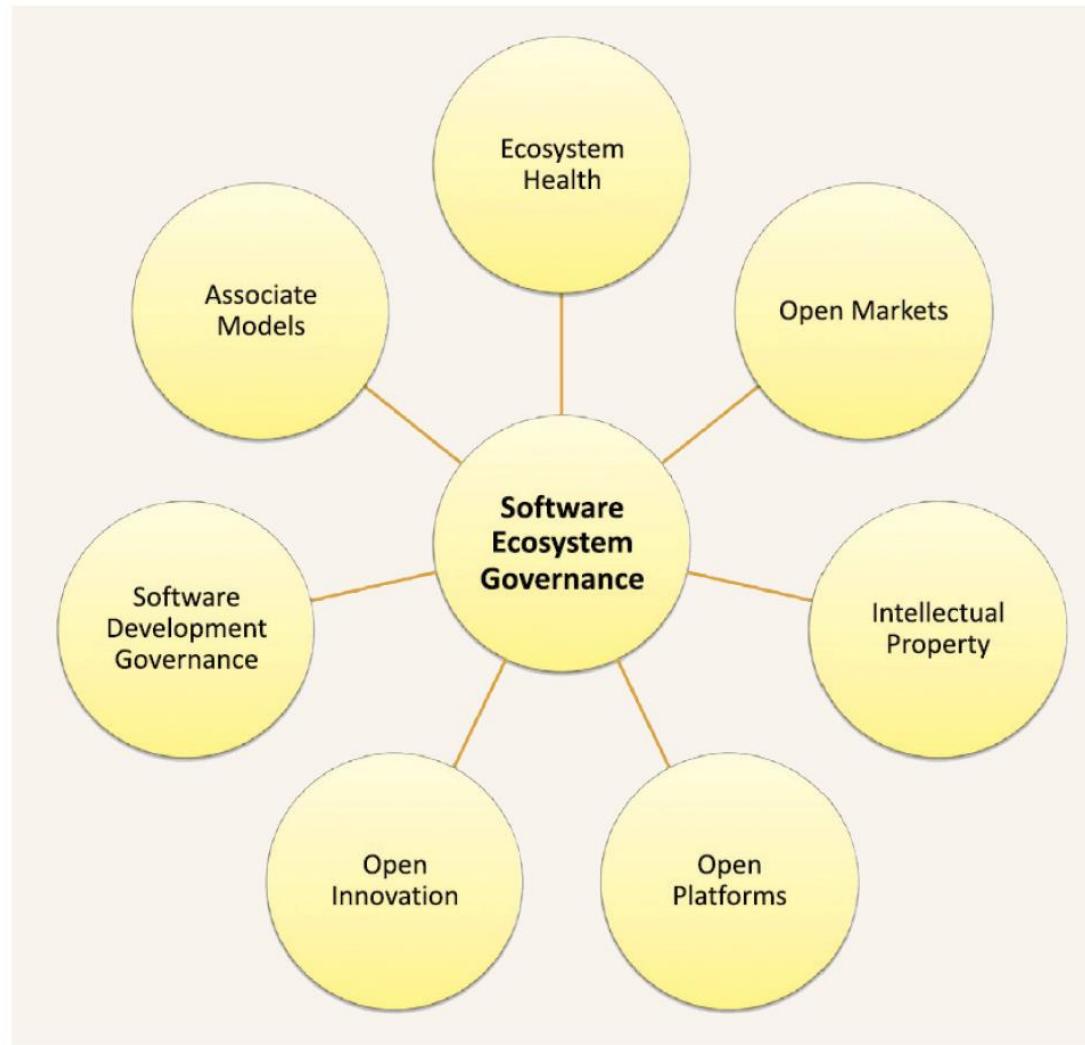


Figure 1: Seven Focus Areas of the SEG-M² [5, fig. 2]

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20. <https://doi.org/10.1145/3395963>

Software Ecosystems –Associate Models

- **Associate Models** – All practices to do with management and coordination of partners. It contains practices such as the creation of partnership models, partner training, and consultancy and sales partner support.

One of the more technical aspects of associate models is the creation of systems that enable partners to communicate with end users, such as approval systems in app stores or SAP's customer partner connection center, that enables partners to share ticketing systems with customers and SAP itself.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems – Ecosystem Health

- **Ecosystem Health** – the practice area that regards the ecosystem as a living ecosystem that can be analyzed as a whole, also contrasting itself with other potentially influencing ecosystems.

The practices in this focus area are concerned with partner health analysis, sharing of market data, and making strategic choices regarding competing ecosystems.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems – Open Markets

- **Open Markets** – the practice area that concerns itself with the creation of an open market for services and applications.

The practices belonging to extension approval, extension marketing, business model innovation, and app delivery are part of the open markets focus area.

The area evenly divides itself across management and technical boundaries.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20. <https://doi.org/10.1145/3395963>

Software Ecosystems – Open Platforms

- **Open Platforms** – All practices related to the creation of a stable solid and open platform belong to the open platforms focus area.

It is concerned with the creation of a platform, the platform's security, its extension capabilities, and documentation.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems – Intellectual Property

- **Intellectual Property** – The practices to do with patent management and intellectual property management within the ecosystem.

At the lowest levels it is concerned with innovation sharing across the ecosystem. At the higher levels it is concerned with patents, licenses, and stimulation of ecosystem health by co-creation.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems – Open Innovation

- **Open Innovation** – the practice area concerned with sharing knowledge across the ecosystem to feed external developers with new possibilities for improvement, also known as niche creation.

At the lowest levels it is concerned with sharing development practices and innovations with partners. At higher levels it is concerned with creating shared innovations and ecosystem standards.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20.
<https://doi.org/10.1145/3395963>

Software Ecosystems – Software Development Governance

- **Software Development Governance** – all practices concerned with observing, supporting, and enabling software developers.

The practices are concerned with domains such as testing, road mapping, shared requirements. At the lowest levels, the focus area is concerned with opening up to developers and enabling them to develop third-party extensions. At higher levels it is concerned with collecting data (software operation knowledge, or SOK) about applications and their developers and about supporting developers in helping each other.

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20. <https://doi.org/10.1145/3395963>

Software Ecosystems Configuration

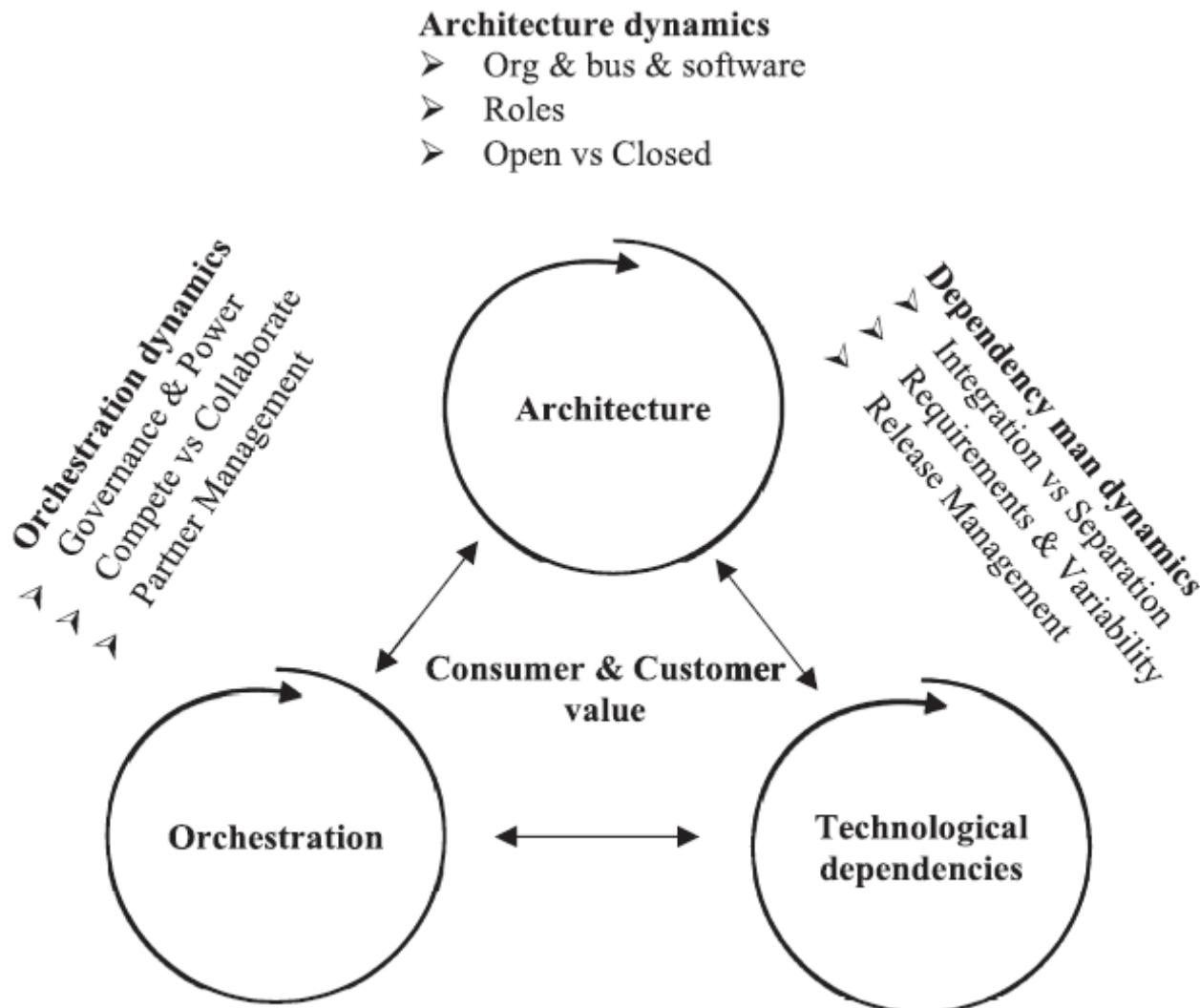


Fig. 1. SECO configuration and contingencies.

Software Ecosystems & Open Innovation

BACKGROUND: SOFTWARE ECOSYSTEMS AND OPEN INNOVATION

A software ecosystem (SECO) consists of a set of actors united under a common vision and aiming to solve a common problem, often through the help of an underpinning technological platform. The actors collaborate and potentially also compete in a shared market for software and services.³ There are many examples of successful SECOs with underpinning platforms, both open⁴ and proprietary.⁵ Examples include operating systems (such as Microsoft Windows and Google's Android), web browsers (for example, Google's Chrome and Mozilla Firefox), and smart home assistants (like Amazon's Alexa and Apple's Siri).⁵ To provide access to their underpinning technology and enable complementary services, keystone organizations typically provide access to an open application programming interface (API). Open APIs allow organizations to share functionality, while allowing their core technologies to remain proprietary, fostering open innovation (OI) within their ecosystems.

OI is an emerging field of research that aims to better understand how organizations "purposively manage knowledge flows across organizational boundaries" for improved organizational innovation.² Chesbrough and Bogers describe three knowledge flows,² modeled in Figure S1:

1. *outside in*, where knowledge flows from external sources to improve internal innovation processes
2. *inside out*, where internal knowledge flows outside the organizational boundaries to external entities for innovation
3. *coupled*, where knowledge flows bidirectionally between the innovating actors.

(Continued)

FIGURE S1. The open innovation model by Chesbrough and Bogers,² where the inside of the funnel represents the inside of the company, and the funnel's borders represent the company's wall to the outside through which the different knowledge flows (outside in, inside out, and coupled).

Software Ecosystems & Requirements Flow

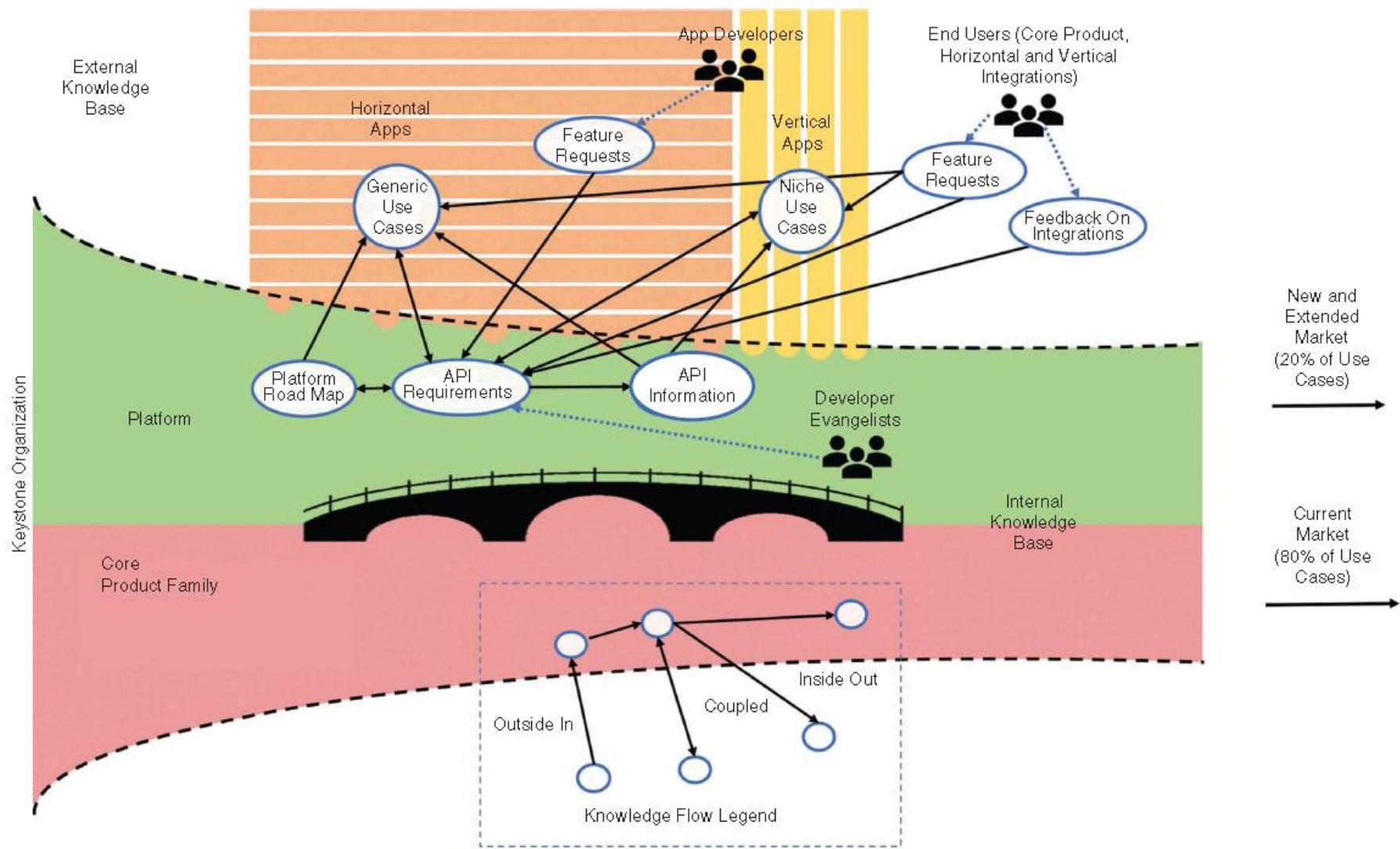


FIGURE 1. Requirements knowledge flows in the SECO, as adapted from the OI model by Chesbrough and Bogers.² The knowledge flow legend indicates the type of knowledge flow represented by the directional arrows.

Damian, D., Linaker, J., Johnson, D., Clear, T., & Blincoe, K. (2021). Challenges and Strategies for Managing Requirements Selection in Software Ecosystems. *IEEE Software*, 38(6), 76-87.
<https://doi.org/10.1109/MS.2021.3105044>

Software Ecosystems –New Competencies Demanded

Table 1: The new skills demanded of developers in software ecosystems

Focus Area	Primary Skill Sets Demanded	Expertise
Associate Models	<ul style="list-style-type: none">management and coordination of partnerscreation of systems that enable partners to communicate with end users	Hybrid - relationship mgt & technical
Ecosystem Health	<ul style="list-style-type: none">partner health analysis, sharing of market data, and making strategic choices regarding competing ecosystems	Hybrid - strategy & technical
Open Markets	<ul style="list-style-type: none">extension approval, extension marketing, business model innovation, and app deliveryevenly divided across management and technical boundaries	Hybrid - mgt and technical
Open Platforms	<ul style="list-style-type: none">creation of a platform, the platforms security, its extension capabilities, and documentation	Technical and documentation
Intellectual Property	<ul style="list-style-type: none">innovation sharing across the ecosystemat higher levels concerned with patents, licenses, and stimulation of ecosystem health by co-creation	Hybrid - relationship mgt, strategy & technical
Open Innovation	<ul style="list-style-type: none">at lowest levels concerned with sharing development practices and innovations with partnersat higher levels concerned with creating shared innovations and ecosystem standards	Hybrid - relationship mgt, strategy & technical
Software Development Governance	<ul style="list-style-type: none">concerned with observing, supporting, and enabling software developersconcerned with domains such as testing, road mapping, shared requirementsopening up to developers and enabling them to develop third-party extensionsat higher levels it concerned with collecting data about applications and developers and about supporting developers in helping each other	Hybrid - relationship and data mgt strategy, technical and documentation

Clear, T. (2020). THINKING ISSUES: Software Ecosystems: what do we need to know? *ACM Inroads*, 10(2), 18-20. <https://doi.org/10.1145/3395963>



#171678945



Questions and Comments....



Tony Clear 2024 S2

I has a question...

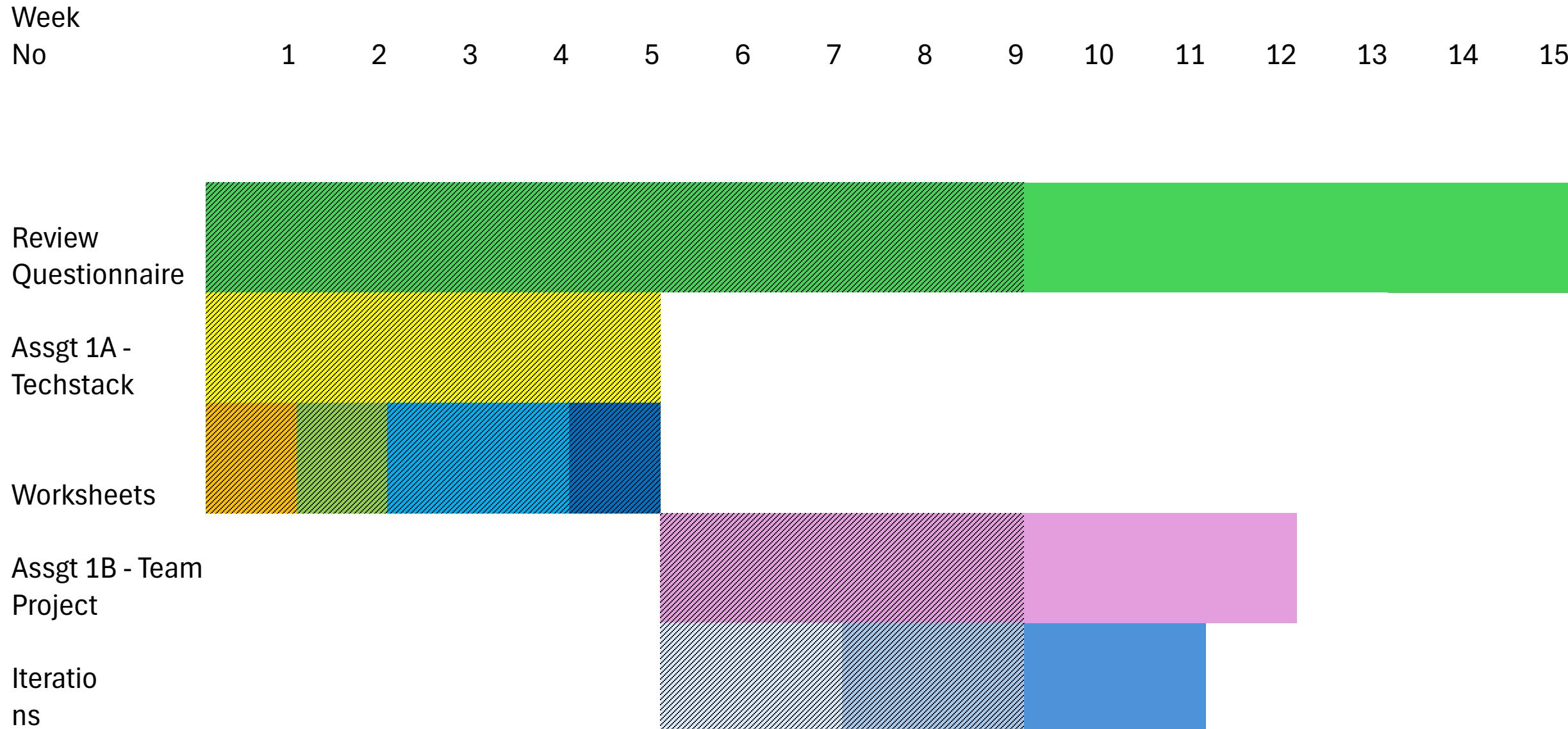


Software Risks and Scale

Week 10



Taking Stock



Software and Risk - Definitions?

In risk analysis and mitigation literature the primary focus is on the project development vision defined in terms of budget and schedule overruns and not on satisfying the customer by meeting technical requirements.

Henry [2004] defines project risk as

“an event, development, or state in a software project that causes damage, loss, or delay.”

Schwalbe [2004] also defines ‘project risk’ as

“...problems that might occur on the project and how they might impede project success.”

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.

Software and Risk – Traffic Control scenario

...traffic control software to direct traffic approaching a multi-lane bridge into the least congested lanes to facilitate a maximum and continuous traffic flow across the bridge, especially in rush hours.

From this description we would identify stakeholders in this software as including:

- vehicle drivers traversing the bridge,
- Bridge maintenance people,
- and the city traffic authority.



It is also straightforward to define success criteria for this software. They might include:

- the system works well in its context;
- it does not promote vehicle accidents;
- the project was delivered on time;
- the project was within budget;
- and the cost/benefit analysis was accurate showing that those developing the system could expect a reasonable return on investment.

The system met all of these conditions and yet it was judged a failure. Why?

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.

Software and Risk - Traffic Control Gone Wrong?

The system needs to manage large amounts of traffic moving through 20 lanes. Cars go over the bridge at two levels. The computer must make continuous interactive rapid and accurate processing decisions about such quantities as lane capacity, average speed of the lane, stopped lanes, taking lanes out of use, changing directions of lanes to account for rush-hour flows.

The system was installed and worked well until the system was required to manage constant heavy traffic loads for 8 hours during **an emergency nuclear disaster evacuation exercise**.

In the eighth hour the software changed lane directions for lanes already filled with cars and the misdirection and accidents clogged the bridge for almost 20 hours.

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.



Software and Risk – Cold Reboot Needed?



In the eighth hour the software changed lane directions for lanes already filled with cars and the misdirection and accidents clogged the bridge for almost 20 hours.

- The crystal clock used for the timing of these decisions would gradually go out of synchronization with 7 or more hours of continuous use.

The developer was fully aware of this problem.

To meet the problem, **the developer specified in the User Manual that the software should be briefly stopped and restarted after 6 hours of continuous heavy traffic loads**. This action would reset the clock and no problem would be encountered.

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.

Software and Risk – Documentation Fix?

To meet schedule and budget constraints, **the bridge software developers opted merely to place a warning in the user manual rather than provide a software solution.**

The primary goals were to deliver the system on time, within budget, and satisfying the customer.

The focus of the risk analysis and mitigation narrowed to those many issues which impact these goals negatively and risks that would derail the project's development.

This narrowing of focus to development risks is canonised in many information systems and software development textbooks and risk management articles

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.

Software and Risk – All Stakeholders?

Software development's shift of project vision contributed to the narrowing of focus on specific types of risks, an **emphasis on quantifiable risk almost to the exclusion of qualitative risk**.

This emphasis on quantitative risk contributes to an underestimating or ignoring of the need to consider risks to extra-project stakeholders in the development of the software.

Schmidt points out that the “[f]ailure to identify all stakeholders: Tunnel vision leads project management to ignore some of the key stakeholders in the project, effecting requirements, and implementation, etc.” [Schmidt et al. 2001 p 15]

Project risk analysis must be expanded beyond the traditional risk analysis to include a broader scope of risks and stakeholders.

Gotterbarn, D., & Rogerson, S. (2005). Responsible Risk Analysis For Software Development: Creating The Software Development Impact Statement. *Communications of the AIS*, 15, 730-750.

Software and Risk Generic Models?

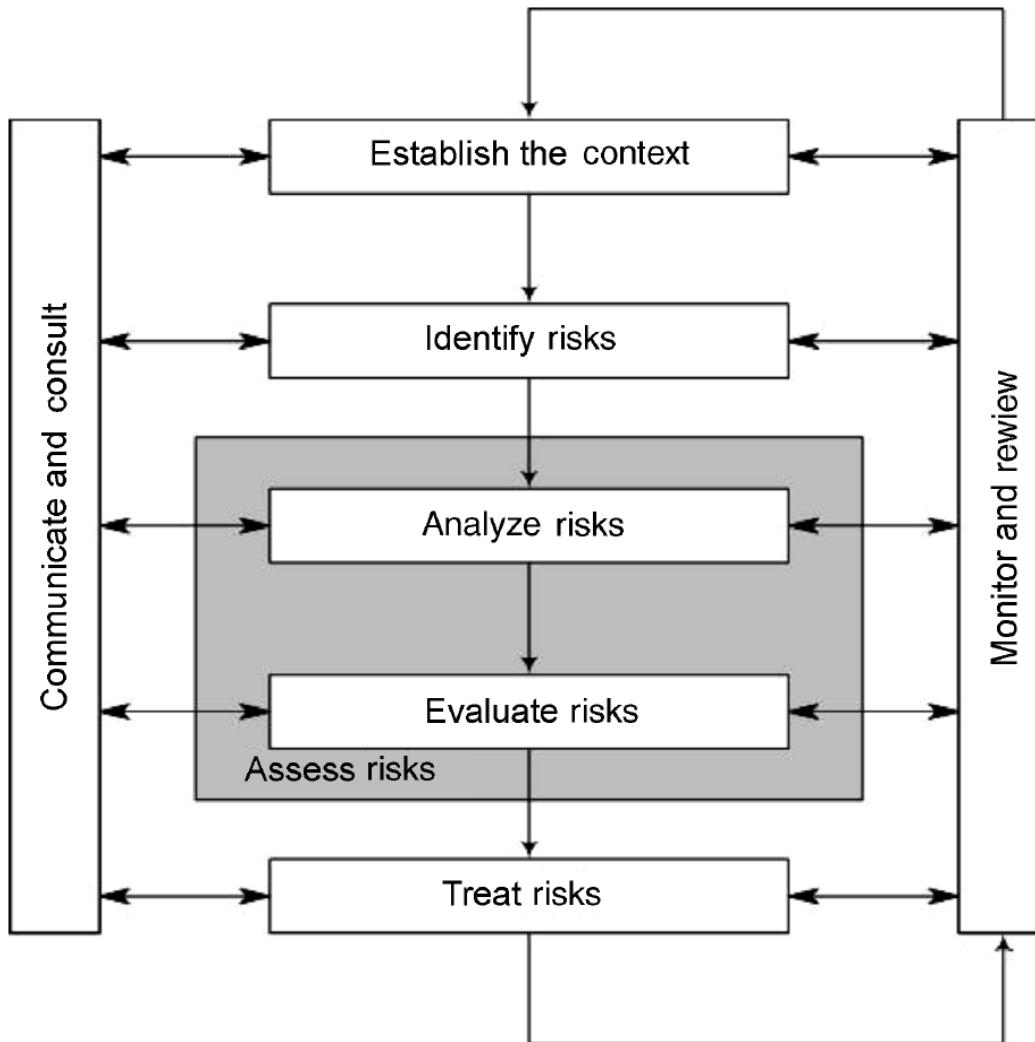


FIGURE 18.1 Risk management (AS/NZS, 1999 p.16).

Gotterbarn, D., Clear, T., & Kwan, C. (2008). A Practical Mechanism for Ethical Risk Assessment - A SoDIS Inspection In K. Himma & H. Tavani (Eds.), *The Handbook of Information and Computer Ethics* (pp. 429-472). John Wiley & Sons.

Software and Risk - Security Risk Lifecycle?

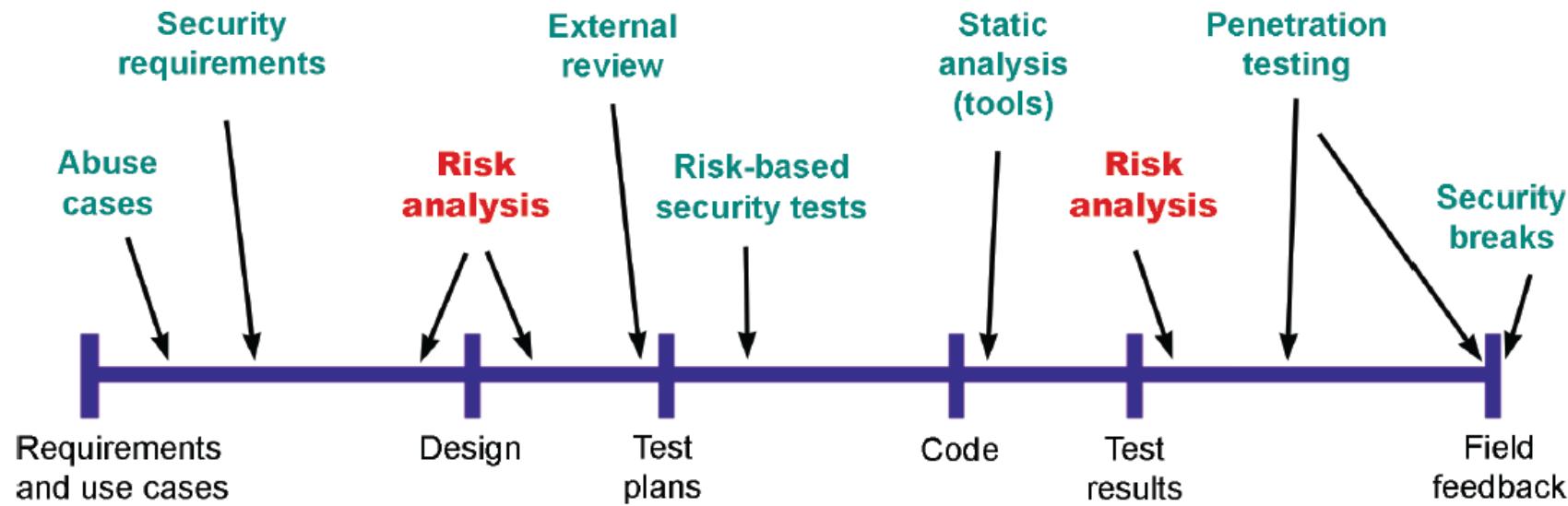


FIGURE 18.2 Hilson's security life cycle.

Gotterbarn, D., Clear, T., & Kwan, C. (2008). A Practical Mechanism for Ethical Risk Assessment - A SoDIS Inspection In K. Himma & H. Tavani (Eds.), *The Handbook of Information and Computer Ethics* (pp. 429-472). John Wiley & Sons.

Software and Risk – Stakeholders and SoDIS?

During the SoDIS Audit process, the SPA forces the analysts to first identify potential stakeholders for this project. The SPA aids the process by providing a partial list of stakeholder types that have been associated with that type of project. Once the stakeholders have been identified, the analysts examine questions that can be either task (hotspot) focused or stakeholder focused. In answering the questions the analysts seek to identify and note potential negative consequences for the identified stakeholders or for the project and, where possible, suggest solutions for the identified items.

http://www.sodis.org/so-dis-audit-process.html#stakeholders

Gotterbarn, D., Clear, T., & Kwan, C. (2008). A Practical Mechanism for Ethical Risk Assessment - A SoDIS Inspection In K. Himma & H. Tavani (Eds.), *The Handbook of Information and Computer Ethics* (pp. 429-472). John Wiley & Sons.

Software and Risk – SoDIS Inspection?

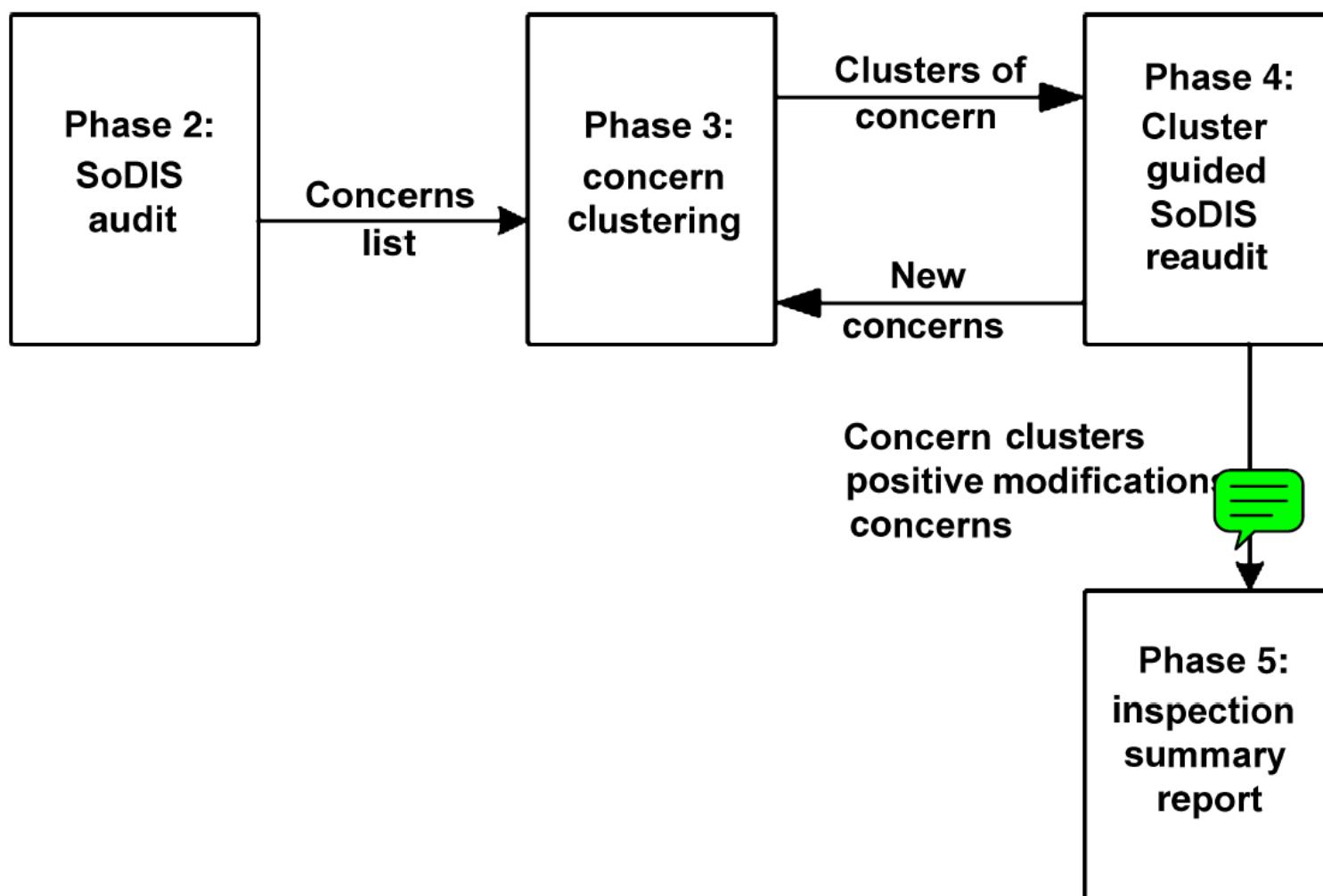


FIGURE 18.8 SoDIS inspection phases 2–5.

Gotterbarn, D., Clear, T., & Kwan, C. (2008). A Practical Mechanism for Ethical Risk Assessment - A SoDIS Inspection In K. Himma & H. Tavani (Eds.), *The Handbook of Information and Computer Ethics* (pp. 429-472). John Wiley & Sons.

Concluding: Software Projects – Student & Risks?

Recent work by New Zealand researchers

- **Student projects and risk management**
- **Derived a set of risk categories**
- **Conclusions include:**

In terms of the Risk Framework's impact on student performance in the course, we found that teams with poor risk management strategies tended to perform worse in the project overall, and had a much higher chance of contacting the course instructor during the semester to report significant issues within the team. (Kirk et al., 2024)

- **Risk categories tabulated next slide...**

Kirk, D., Luxton-Reilly, A., & Tempero, E. (2022). *Refining a Risk Framework for Student Group Projects* Proceedings of the 22nd Koli Calling International Conference on Computing Education Research, Koli, Finland. <https://doi.org/10.1145/3564721.3564730>

Kirk, D., Luxton-Reilly, A., Tempero, E., Crow, T., Denny, P., Fowler, A., Hooper, S., Meads, A., Shakil, A., & Singh, P. (2024). Educator Experiences of Low Overhead Student Project Risk Management. Proceedings of the 26th Australasian Computing Education Conference,

Table 1: Kirk et al. Risk Categories Description [10, 11]

C

Category	Description
Student Contribution	Student contribution to project.
Engagement	Commit to project as a result of mental state, for example, interest, conscientiousness, motivation.
Expertise	Ability to execute tasks to be carried out. Dimensions include knowledge, experience, familiarity with technology.
Personality	Personal attributes that contribute towards team dysfunction or ineffective participation, for example, team members who prefer to work in isolation, are too shy to speak in meetings or are resistant to complying with decisions.
Availability	Degree to which student is available to the project caused by external factors. Examples are students taking several courses with limited time for the project, time differences, family illness causing student to be unable to contribute or communicate.
Wellbeing	Student health issues that affect their contribution.
Team Self-management	Ability of team to self-manage effectively.
Communication	Success of stakeholder communication. Relates to failure to plan communications mechanisms or execute these as planned.
Co-ordination	Success of coordinating project tasks. Examples include version control, responsibilities, task scheduling. Work not well coordinated on large multi-team projects.
Co-operation	How well students agree and treat each other with respect. Factors include differing viewpoints and interpretations due to cultural, background and personality differences.
Process	Success of execution of planned project tasks. Issues include uneven distribution of workload and individuals not completing agreed tasks on time.
Clarity	Degree of clarity around project activities and structure. Examples are not defining activities well, documentation expectations, confusion around role of lecturer.
Resources	Failures with required resources.
Hardware	Unforeseen issues with equipment. Examples are hard disks, internet connections and not having needed equipment.
Software	Unanticipated software issues, for example, using third party components.
Technology	Unforeseen technology issues, for example, technology changing too fast.
Stakeholder Contribution	Clients and lecturers contribution to project.
Commitment	Willingness to commit to project, for example, interest, conscientiousness, motivation.
Expectations	Stakeholder expectations of the project.
Expertise	Capability with respect to the application. Dimensions include knowledge, experience, familiarity with technology.
Availability	Degree to which stakeholder is available to the project

l
g a
t
s
on
;64



#171678945



Questions and Comments....



Tony Clear 2024 S2

I has a question...



Agile, History and Ways of Working

Week 11

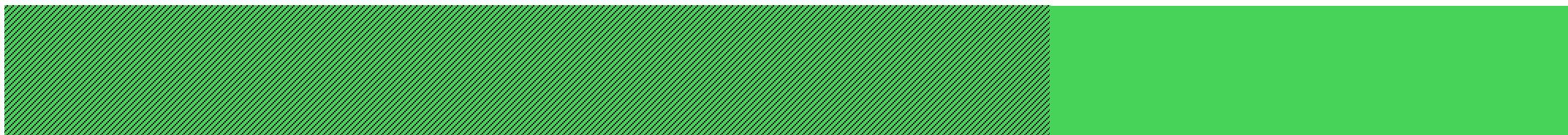


Taking Stock

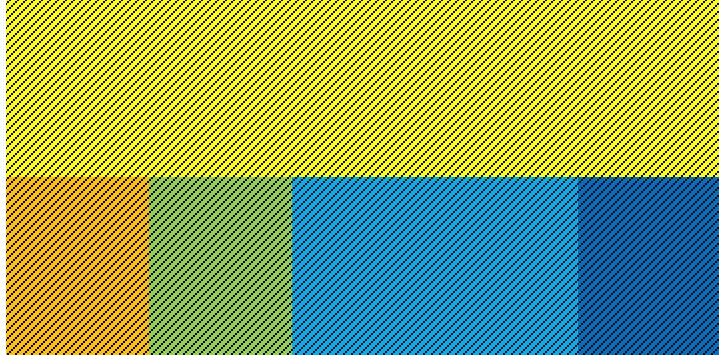
Week
No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Review
Questionnaire

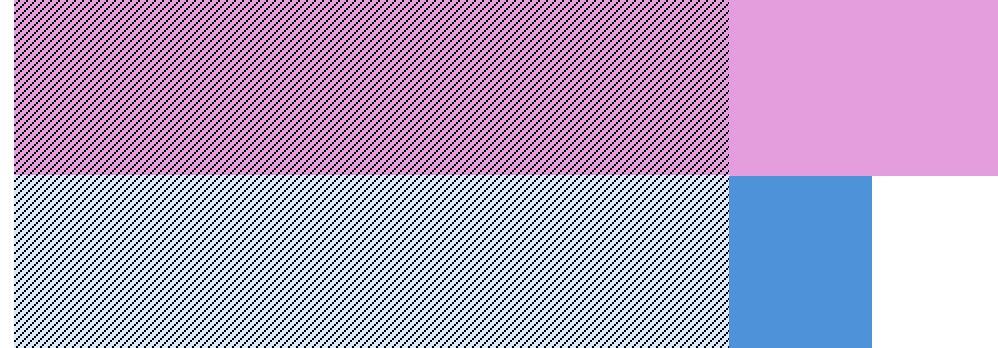


Assgt 1A -
Techstack



Worksheets

Assgt 1B - Team
Project



Iteratio
ns

Software Engineering?

Issues surrounding ‘agile’ software engineering and developments in practice

- **Modern agile etc.**
- **Scrum guide revisions**
 - Jim’s NZ Post Presentation
 - Scott Ambler’s critique

Ways of working and modelling

Other Lifecycles

History of Software Engineering

- **Insights from pioneers and experts**
- **Margaret Hamilton**
- **Fred Brooks**
- **Grady Booch**



#171678945



Questions and Comments....



Tony Clear 2024 S2

CISE ENSE701

I has a question...

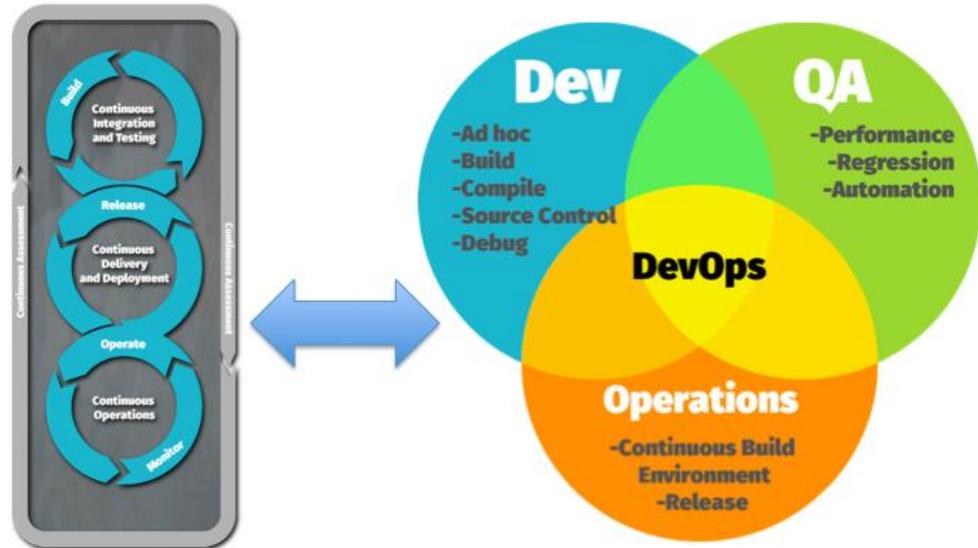


Week 12

Course Review

Empirical Software Engineering

The DevOps Way of Working

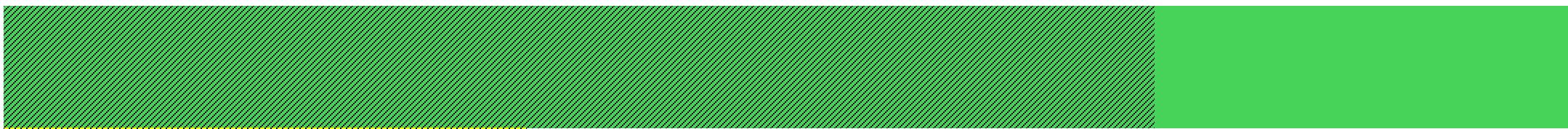


Taking Stock

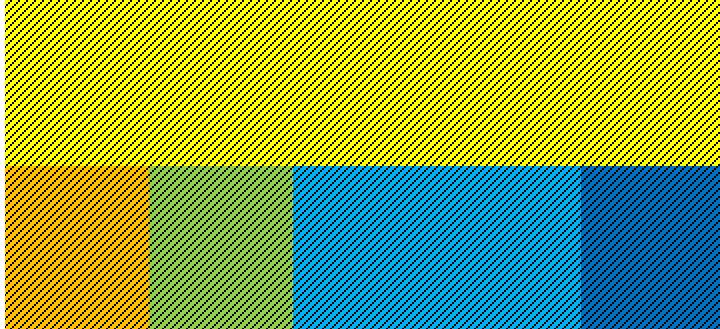
Week
No

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

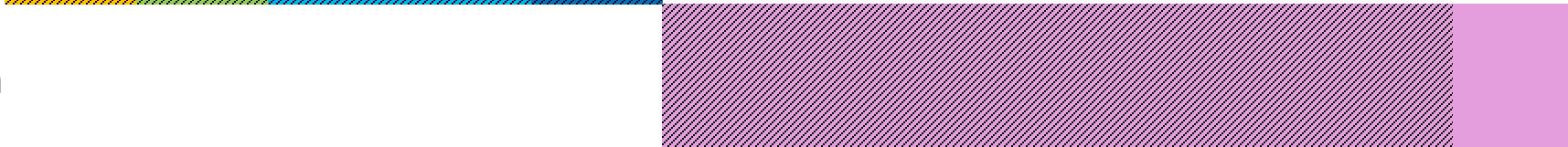
Review
Questionnaire



Assgt 1A -
Techstack



Worksheets



Assgt 1B - Team
Project

Iterati
ons

Briefly Reviewing the Course

- Check the course schedule and modules for topics and activities traversed
- See the next slide for assessment
- Review the example test (under assessments on Canvas) to get some sense of the style of the final test (although topics have evolved since then)
- Review Jim's mind map – but technology elements based on the earlier MERN platform used - and consider the roadmaps for the current tech stacks and elements – two slides on
- Review the key events in the development process and practices adopted
- Consider the insights from your collaborative software engineering experience and lectures and digest them for the final test

Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)

(Individual)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,

Setup the tools needed to work with the selected

Tech Stack (front-end/backend)

Set up tools to assure quality of product

Set up tools to deploy the product to the cloud

Set up tools to monitor and alert issues post deploy

Learn how to use the tools

Learn how to use the Tech Stack

Understand the product goals -> Product Backlog

Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)

Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)

(small team - 4 Including QA)

Capability building by Developing a Product in a small team

Apply a new tech stack and tool set

Practice DevOps and Scrum WoW

Collaborate with a Product Owner and team

Three sprints to learn fast – fast feedback

Submit – reviews weeks 7,9,11 (tutorials)

Capability and learning Portfolio with Evidence

Product increments

Sprint 1 weeks 5 and 6

Sprint 2 weeks 7 and 8

Sprint 3 weeks 9 and 10

Ass 2 Knowledge Check (30%)

(Individual, online questions)

A set of questions about scenarios to confirm you have understood main language and principles

Sometime in Revision weeks (Faculty schedules)

Roadmap Resources - Topics

Skill Based

<https://roadmap.sh/react>

<https://roadmap.sh/javascript>

<https://roadmap.sh/typescript>

roadmap.sh is a community effort to create roadmaps, guides and other educational content to help guide the developers in picking up the path and guide their learnings.

<https://roadmap.sh/>

Role Based

<https://roadmap.sh/frontend>

<https://roadmap.sh/backend>

e.g. Architectural Patterns - 12 Factor Apps

<https://www.youtube.com/watch?v=FryJt0Tbt9Q>

How will we get feedback on product from users/client?

Regular review of product increment

How will we keep improving our process

Regular review of team process

Iteration of design/code/test Prod Increment

Iteration of design/code/test More Prod Increment

Iteration of design/code/test Final Prod Increment

What do we need to do before we start coding?

- Initial product backlog and story map (some uncertainty)
- User stories with Acceptance criteria
- Detail understanding and design of features for next iteration only
- Architecture and tech stack and deployment
- Dev Environment set up
- Plan for iteration 1

Goal and Iteration Backlog

How will we decide what is in each Iteration?

Regular team meeting during iteration...
Is there anything stopping us from reaching the goal?

How will we coordinate work with each other and keep on the same page?

How will we manage changes to requirements?

How will we manage risks?

How will we assure quality?

Some Empirical Research in SERL: What works under what circumstances and why?

When is it better to use mob, pair or solo programming?

In an Agile team, how does shared understanding of requirements occur?

Project or feature triage – how to feed the agile delivery engine?

Organizational structure and Agile – managing organizational interfaces with agile teams

How can programming/SE/SRE/Testing be taught/learned more effectively?

What are the emerging roles in an agile team and how is work divided among them?

Requirements tracing for cyber-physical systems

How can technical debt be measured?

How is a new team member onboarded and how can this be improved?

What can we learn from failed projects such as Novapay? Dilemma analysis.

Behaviour patterns of developers from mining software repositories?

What potential requirements are embedded in user reviews?
Machine learning for text processing.

How can coordination of distributed teams be improved?

How can team “health” be monitored and diagnosed?

What are the expectations of the PO?

What are the benefits and challenges of planning poker for estimation? What can be changed to improve the benefits?

How are requirements changes managed in globally distributed teams?

Which code should be refactored and how? Semi-automated refactoring tool.

How is DevOps implemented and what are the benefits and challenges?

Measuring testability with dynamic metrics?

What are the benefits and challenges of implementing TDD and why?

What are the success factors post adoption of Agile?

Some Empirical Research in SERL: What works under what circumstances and why?

When is it better to use mob, pair or solo programming?

How is a new team member onboarded and how can this be improved? Interview Survey

In an und occ
R f
Organizational structure and Agile – managing organizational interfaces with agile teams

How can programming/SE/SRE/Testing be taught/learned more effectively?

cyber-physical systems

an technical debt measured?

What are the expectations of the PO?

What are the benefits and challenges of planning poker for estimation? What can be changed to improve the benefits?

When is best to do mob, pair or solo programming? Interview and Observation.

Behaviour patterns of developers from mining software repositories?

How is DevOps implemented and what are the benefits and challenges?

Spred and oring tool.

What potential requirements are embedded in user reviews? Machine learning and text processing.

cyber-physical systems

How can team "health" be monitored and diagnosed?

ination of
ns be impr

How is DevOps implemented and what are the benefits and challenges in practice? Interview-based Case Studies

The plan...

There are a lot of claims about the benefits of DevOps and descriptions of how to implement it, but very little empirical evidence?

What are the benefits, enablers and challenges in implementing DevOps *in practice*?

The plan...

There are a lot of claims about the benefits of DevOps and descriptions of how to implement it?

What are the benefits, enablers and challenges in implementing DevOps in practice?

Interviews survey

Multiple organisations

Different stages of

Views of different roles

Deep insights
in real

Content
Analysis of
Transcripts

The plan...

What are the enablers and challenges in implementing DevOps in practice?

Interviews survey

Multiple organisations

Different stages of

Views of different roles

Dev (one of 3 back-end, 2 front-end, two floaters), Tester, Ops, QA release manager, Tech Lead Infrastructure, Training manager

Medium Sized SaaS Product company. Fast growth.

**Agile way of working (scrum)
2 years into the DevOps journey**

Examples of titles of other research papers...

Relationship of DevOps to Agile, Lean and Continuous Deployment A Multivocal Literature Review Study

Lucy Ellen Lwakatare^(✉), Pasi Kuvala, and Markku Oivo

Faculty of Information Technology and Electrical Engineering,
University of Oulu, Oulu, Finland
{lucy.lwakatare,pasi.kuvala,markku.oivo}@oulu.fi

Abstract. In recent years, the DevOps phenomenon has attracted interest amongst practitioners and researchers in software engineering, reflecting the greater emphasis on collaboration between development and IT operations. However, despite this growing interest, DevOps is often conflated with agile and continuous deployment approaches of software development. This study compares DevOps with agile, lean and continuous deployment approaches in software development from four perspectives: origin, adoption, implementation and goals. The study also reports on the claimed effects and on the metrics of DevOps used to assess those effects. The research is based on an interpretative analysis of qualitative data from documents describing DevOps and practitioner's responses in a DevOps workshop. Our findings indicate that the DevOps phenomenon originated from continuous deployment as an evolution of agile software development, informed by a lean principles background. It was also concluded that successful adoption of DevOps requires agile software development.

Literature Review: Promises and Challenges of DevOps

Shubhangi Nagpal
University of Waterloo
s6nagpal@uwaterloo.ca

Anam Shadab
University of Waterloo
a2shadab@uwaterloo.ca

DevOps Adoption Benefits and Challenges in Practice: A Case Study

Leah Riungu-Kalliosaari^{1(✉)}, Simo Mäkinen¹, Lucy Ellen Lwakatare²,
Juha Tiihonen¹, and Tomi Männistö¹

¹ Department of Computer Science, University of Helsinki,
Gustaf Hällströmin katu 2b, P.O. Box 68, 00014 Helsinki, Finland
riungu@cs.helsinki.fi

² Department of Information Processing Science,
University of Oulu, P.O. Box 3000, 90014 Oulu, Finland

Abstract. DevOps is an approach in which traditional software engineering roles are merged and communication is enhanced to improve the production release frequency and maintain software quality. There seem to be benefits in adopting DevOps but practical industry experiences have seldom been reported. We conducted a qualitative multiple-case study and interviewed the representatives of three software development organizations in Finland. The responses indicate that with DevOps, practitioners can increase the frequency of releases and improve test automation practices. DevOps was seen to encourage collaboration between departments which boosts communication and employee welfare. Continuous releases enable a more experimental approach and rapid feedback collection. The challenges include communication structures that hinder cross-department collaboration and having to address the cultural shift. Dissimilar development and production environments were mentioned as some of the technical barriers. DevOps might not also be suitable for all industries. Ambiguity in the definition of DevOps makes adoption difficult since organizations might not know which practices they should implement for DevOps.

Examples of titles of other research papers...

DevOps Capabilities, Practices, and Challenges: Insights from a Case Study

Mali Senapathi
Auckland University of Technology
Auckland, New Zealand
mali.senapathi@aut.ac.nz

Jim Buchan
Auckland University of Technology
Auckland, New Zealand
jim.buchan@aut.ac.nz

Hady Osman
Auckland, New Zealand
hadyos@gmail.com

ABSTRACT

DevOps is a set of principles and practices to improve collaboration between development and IT Operations. Against the backdrop of the growing adoption of DevOps in a variety of software development domains, this paper describes empirical research into factors influencing its implementation. It presents findings of an in-depth exploratory case study that explored DevOps implementation in a New Zealand product development organisation. The study involved interviewing six experienced software engineers who continuously monitored and reflected on the gradual implementation of DevOps principles and practices. For this case study the use of DevOps practices led to significant benefits, including increase in deployment frequency from about 30 releases a month to an average of 120 releases per month, as well as improved natural communication and collaboration between IT development and operations personnel. We found that the support of a number of technological enablers, such as implementing an automation pipeline and cross functional organisational structures, were critical to delivering the expected benefits of DevOps.

1 INTRODUCTION

The DevOps concept [1] emerged to bridge the disconnect between the development of software and the deployment of that software into production within large software companies [2]. The main purpose of DevOps is to employ continuous software development processes such as continuous delivery, continuous deployment, and microservices to support an agile software development lifecycle. Other trends in this context are that software is increasingly delivered through the internet, either server-side (e.g. Software-as-a-Service) or as a channel to deliver directly to the customer, and the increasingly pervasive mobile platforms and technologies on which this software runs [3]. These emerging trends support fast and short delivery cycles of delivering software in the fast-paced dynamic world of the Internet. As such DevOps has been well received in the software engineering community and has received significant attention particularly in the practitioner literature [4]. Annual 'State of DevOps' reports show that the number of DevOps teams has increased from 19% in 2015 to 22% in 2016 to 27% in 2017 [5].

Examples of titles of other research papers...

Emerging Trends for Global DevOps: A New Zealand Perspective

Waqr Hussain*, Tony Clear*, Stephen MacDonell*

*Software Engineering Research Lab (SERL)

School of Engineering, Computer and Mathematical Sciences (SECMS), Auckland University of Technology (AUT)

Auckland, New Zealand

whussain,tclear,smacdonell@aut.ac.nz

Abstract—The DevOps phenomenon is gaining popularity through its ability to support continuous value delivery and ready accommodation of change. However, given the relative immaturity and general confusion about DevOps, a common view of expectations from a DevOps role is lacking. Through investigation of online job advertisements, combined with interviews, we identified key Knowledge Areas, Skills and Capabilities for a DevOps role and their relative importance in New Zealand's job market. Our analysis also revealed the global dimensions and the emerging nature of the DevOps role in GSE projects. This research adds a small advanced economy (New Zealand) perspective to the literature on DevOps job advertisements and should be of value to employers, job seekers, researchers as well as educators and policy makers.

Keywords— *GSD; GSE; DevOps; Continuous Integration; Continuous Deployment; Education; Empirical Analysis; Online Job Postings Analysis; Content Analysis; Cloud; AWS.*

I. INTRODUCTION

DevOps has recently gained popularity as a philosophy that synergizes the operational silos of Software Development (Dev) and IT Operations (Ops) [1]. The three main catalysts that propel its rapid adoption include: a) higher quality expectations from software as it is increasingly offered as a service in the cloud b) demands for rapid delivery of change with growing acceptance of agile and its change embracing attitude, and c) the availability of on-demand powerful and plentiful hardware on the cloud [2]. The uptake of the DevOps trend has been global [1]. Some large organizations claim to have successfully applied its practices in their distributed teams and achieved smooth team collaboration, shortened feedback loops and better customer collaboration [3].

A DevOps strategy supports a globally scalable, rapid and incremental service delivery strategy within a cloud computing infrastructure. Thus it offers potential for software and services companies to operate and compete successfully beyond the traditional centres of technology innovation. For many 'Small Advanced Economies' (SAEs) such as New Zealand [4], this

service model has attractions. Governments and the IT sector see opportunities to move themselves up the global value chain through delivery of high value products and services. Underpinned by a skilled population base, they believe this will result in more high paying jobs and greater export income [5].

Many believe that DevOps is here to stay, at least in many IT sectors to help organizations deliver quality service with efficiency [2]. However, given the relative recency and emergent nature of the DevOps phenomenon, an inadequate body of knowledge, and general confusion surround DevOps concepts and definitions [6]. The software industry therefore, does not share a common view of its meaning. This lack of clarity fuels several misperceptions. Employers are unable to set and describe the right expectations (Knowledge Skills and Capabilities (KSCs)) for DevOps roles. Job seekers are thus unsure of the commonly assigned responsibilities and expectations [7]. Educators on the other hand, find it challenging to train students and impart the desired skillsets and capabilities for their smooth transition into these roles [8].

A recent study has compared responsibilities of a DevOps engineer role in three countries (USA, UK and Canada) and has shown that the responsibilities and skills expected from DevOps roles significantly vary from country to country [8]. Motivated by country specific differences in their study, this paper analyzes what New Zealand employers actually state they require in DevOps job listings and the extent to which Global Software Engineering (GSE) aspects are included, whether explicitly or implied. The aim is to better understand the growing phenomenon of DevOps in the New Zealand job market, (as one example of an SAE), identify major KSCs and understand how some of those relate to GSE. We draw on interviews and insights from practitioners, to complement the job description data. The questions this research tries to address are:

1. *What knowledge areas, skills and capabilities (KSCs) are in demand for a DevOps role in New Zealand's IT market?*

Case Study: The Software Development process

Cross functional

Scrum teams

Product Owner

Teams organised by product functional module

Sprints (2-3 weeks)

Product Backlog

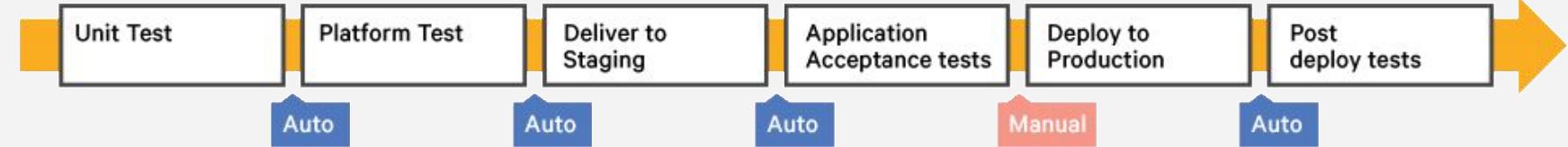
Daily standups

Sprint planning

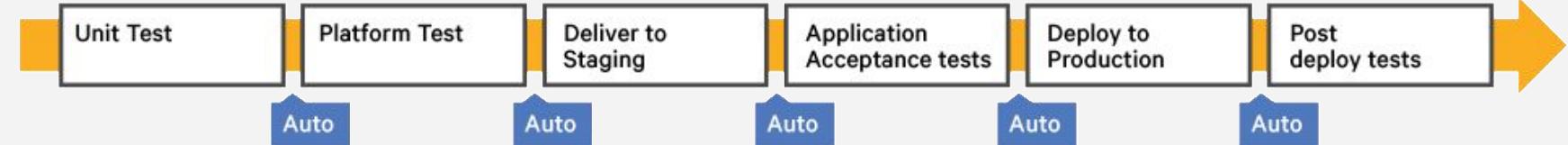
Sprint review

Retrospectives

Continuous Delivery



Continuous Deployment



it is continuous delivery of small features or feature sets
May not suit an iterative approach like Scrum
May suit more of a Kanban Approach

Diagram Used with permission of Hady Osman

Case Study: Team structures

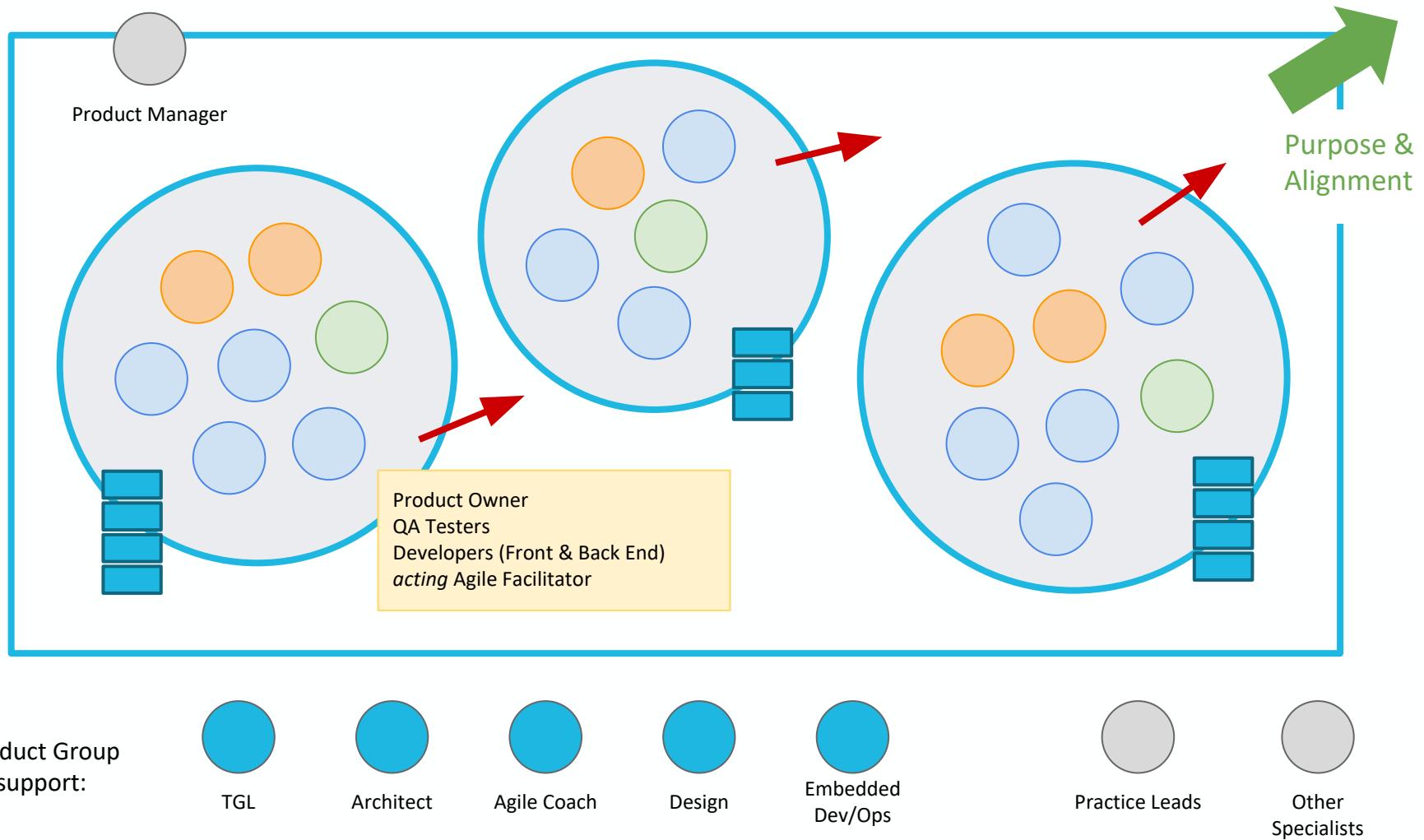
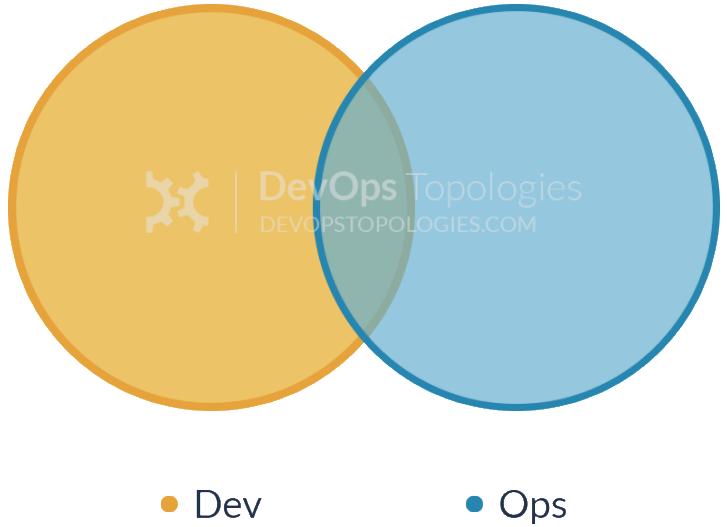


Diagram Used with permission of Hady Osman

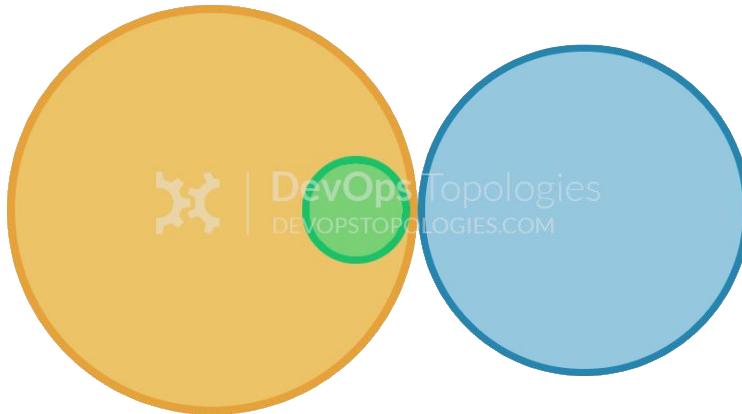
Team structures – patterns and anti-patterns



Type 1 suitability: an organisation with strong technical leadership.

Potential effectiveness: HIGH

Devopstopologies.com



Type 3 suitability: organisations with several different products and services, with a traditional Ops department, or whose applications run entirely in the public cloud.

Potential effectiveness: MEDIUM

The Shared meaning of DevOps

What are the main concepts associated with the meaning of DevOps?



User feedback of new feature in production- try small things out and learn fast!

Is there a shared meaning of DevOps – Case Study

Embedded ops. Ops, Dev, QA work together. Ops is considered from the start and in a longer term (QA Release manager)

It just means you are not relying on other teams to do the infrastructure. You have control over it – choice of tool to use for example (Developer)

Bringing [ops and dev] skill sets togetherand bringing people together... with more natural communication.... Also looking at automation as a rule-of-thumb (Team lead Infrastructure)

Deployment [of a feature] is in the control of the team that develops it.....We write code, review it, test it, merge it and deploy it. (Ops in Team)

I think it's a name for the period of time where software developers transition from just giving their stuff to system admins when it is finished, to actually caring and looking after it themselves. (Training manager)

Every team owns its own infrastructure rather than ...someone else is doing the job when we have problems. (Tester/QA)

Is there shared understanding of the meaning of DevOps? - Literature

- The combination of people, culture, process, tools and methodologies that reduce risk and cost, enable technology to change at the speed of the business and improve overall quality.[13]
- DevOps describes the practices that streamline the software delivery process, emphasizing the learning by streaming feedback from production to development and improving the cycle time. [11]
- DevOps is a movement intended to reduce the time between code complete and code in production as well as share responsibility and increase collaboration between developers and system administrators [2]
- DevOps is a job title

The meaning of DevOps – main concepts

Bringing Dev and Ops closer

collaborate (rather than blaming)– planning, design, code.

Communicate early and frequently and face to face

Extend team **capability**, influence each other's design of code and infrastructure,

Team **responsibility** extended to **deployment**, infrastructure, as well as **post-deployment** monitoring, debugging and fixing issues in deployed features (on-call is shared)

what the team **cares** about is changed

Get **team capability** to deliver features frequently to:

create value for customers quickly and

get fast feedback on usefulness to users

Learn from mistakes

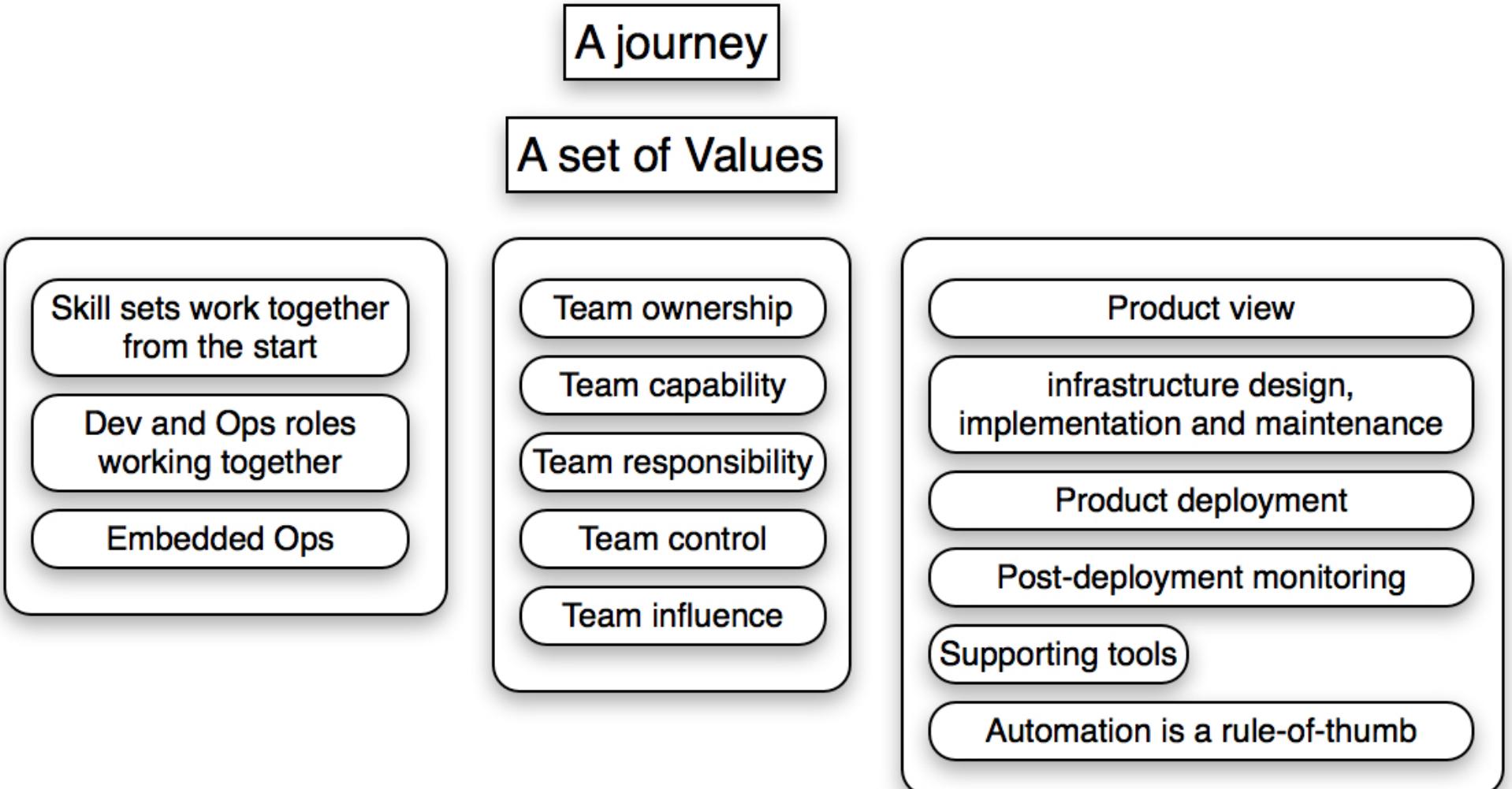
Reduce risk

A movement – **a set of values** – a way of thinking. A way of working - tools and practices that support the realisation of these values and assumptions.

A **journey**...stages of adoption

A **job title** – with Dev and Ops (and QA?) capabilities

What does DevOps mean...



What DevOps is NOT

- It isn't just a job title prefix
- It isn't a team (but “Dev” and “Ops” are)
- It isn't a technology
- It doesn't mean “a system administrator who can code”
- It doesn't mean “automating stuff”
- It doesn't mean “there's no operations team now”

<https://thenextweb.com/syndication/2020/06/05/get-the-fundamentals-of-devops-right-then-worry-about-tools/>

Triggers and Motivation for Adopting DevOps

What events and reasoning trigger/drive an organisation to start/continue) adopting DevOps?

What motivates a team to be willing to adopt DevOps?

What are the expected benefits/value to the customer/organisation/team?

Who (what roles) drive the move to DevOps?



Triggers and goals of adopting DevOps

The end goal is that we can institute change sort of fast but with integrity. We accept more risk for this speed. (*Embedded Ops*)

Cadence and speed. To be able to deliver quality software at speed you need to have fewer points along the journey. (*Training Manager*)

Continuous deployment I think. The ability to make a change and have it reflected in the real world instantly. (*Team lead Infrastructure*)

It just means you are not relying on other teams to do the infrastructure. You have control over it – choice of tool to use for example. To get the feel of small startups in a big organisation. (*Developer*)

The goal is for the production team to own the infrastructure (*Tester*)

Avoid the outages needed for large releases (*Team lead Infrastructure*)

Development team wanted more hands-on with infrastructure (QA Release manager)

As the teams grew there was a bottleneck to get stuff into production because we had to give it to the Ops team. DevOps was to overcome this bottleneck (*Training Manager*)

Avoid the double ups and start-stops in communications between ops and devs dealing with an issue ticket. (*Team lead Infrastructure*)

Triggers and Motivation for Adopting DevOps

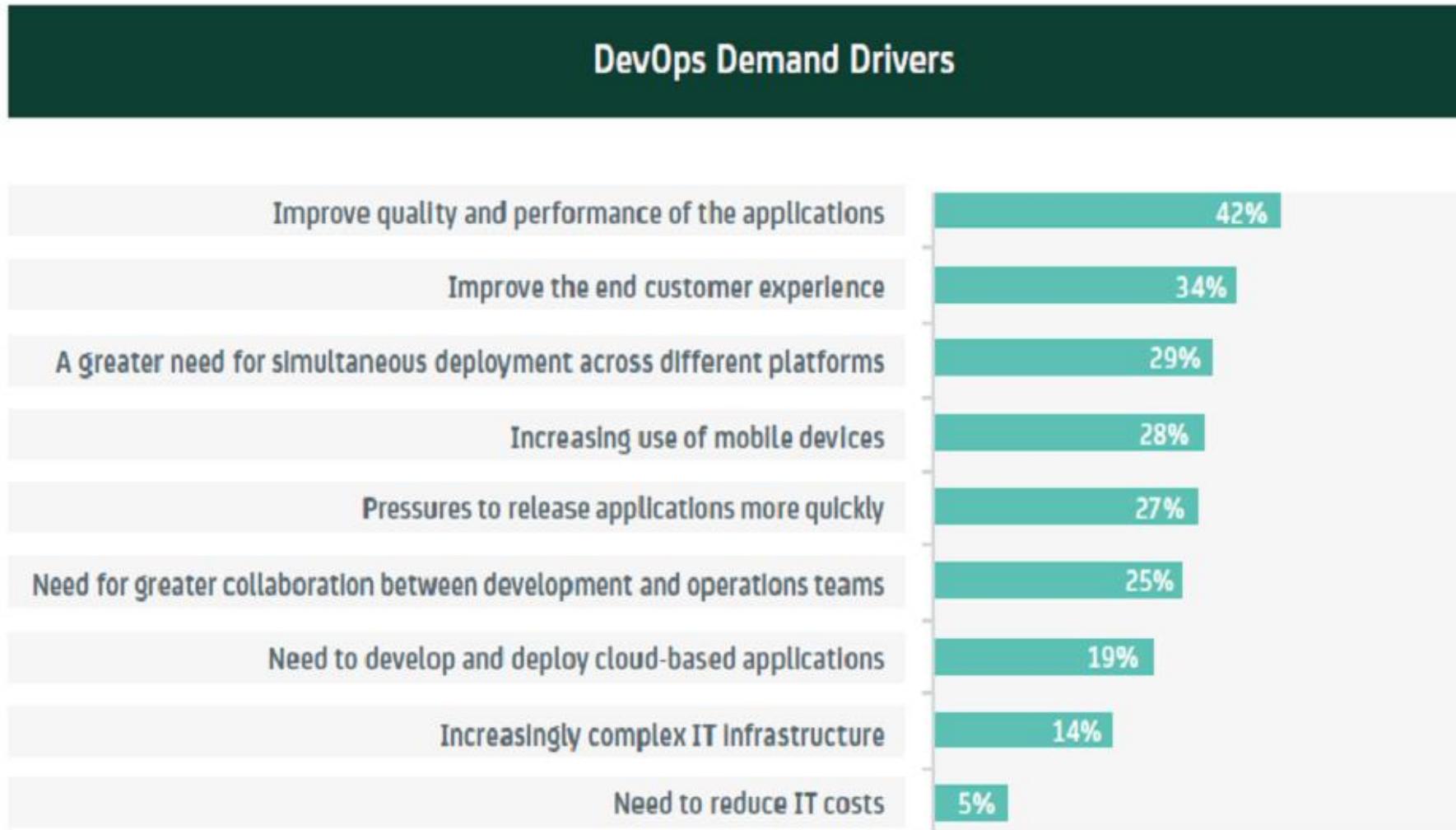


Figure 2 what is driving need for DevOps?

[13] (literature review)

Triggers and Motivation for Adopting DevOps

Developers driving/willing to avoid pain of Ops bottleneck

Managers – driving for faster feature release to respond to users needs quicker. Also fewer outages (planned and unplanned) – improved customer experience.

Ops – increase speed of feature releases.



Drivers for Adopting DevOps

Conversation with pre-DevOps champions Chief Platform Officer and Chief Product Officer

frequent frustration between the company's operation and product teams who have had **competing priorities**

Operation and Product teams operated under what was identified as a **mismatch of incentives and control**.

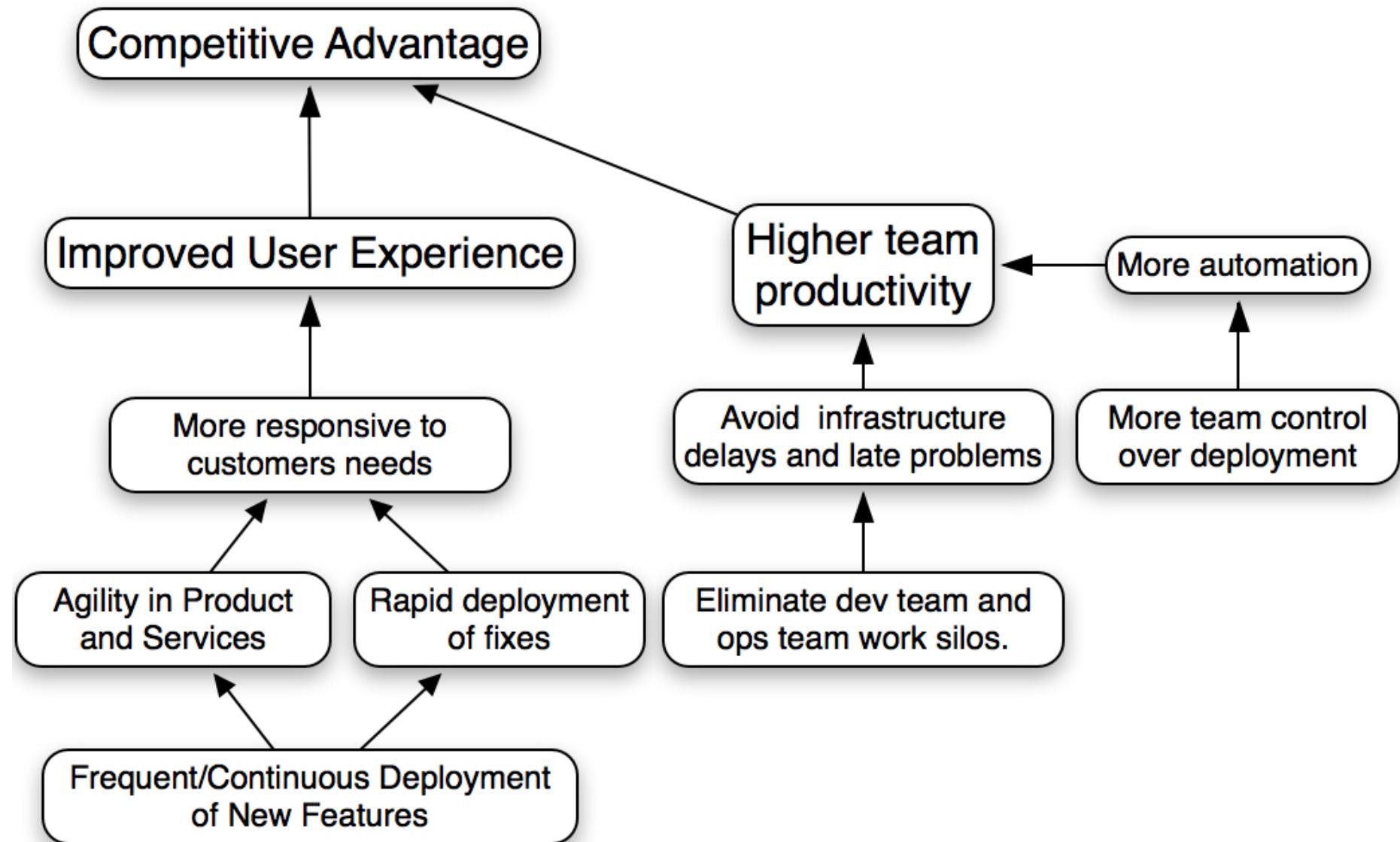
Operation teams were accountable for performance and uptime, development teams were in a better position to improve it

development teams were accountable for shipping product with great agility and velocity, operation teams were in controls of major portions of the SDLC

Strategic tech changes required **infrastructure as code capability**
skill set is dev and ops



The drivers to adopt DevOps



Benefits of DevOps Adoption

What are the expected/claimed benefits of adopting DevOps?

What are the actual organisational/customer/team benefits realised as a result of adopting a DevOps way of working?

What evidence is there that these benefits are realised?



Case Study - Benefits of DevOps Adoption

Team ownership and responsibility is huge, the devs and QAs have loved it. More frequent releases – because more deployers and smaller releases. Easier to contain a release. More features for end users.
(QA Release manager)

You write better code because you know what's going to happen to it.
(Developer)

Reduced the dependencies and reliance on Ops team for infrastructure related tasks needed for a new feature (eg a new DNS entry could take a week – the Dev team wouldn't even know what to ask sometimes).
(Team lead Infrastructure)

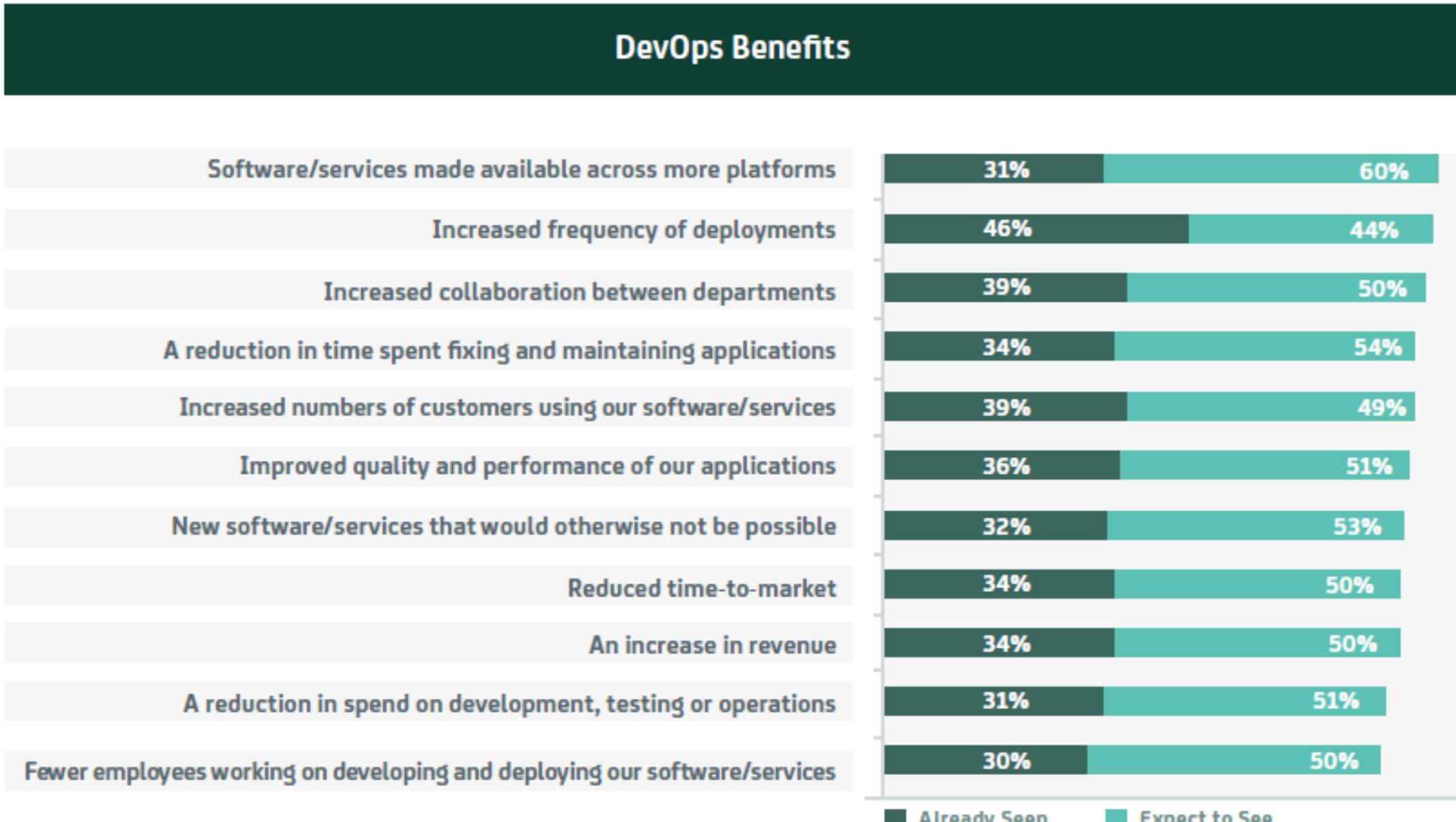
You can build so much better integrity. You build your own solutions that work for your own environment. *(Embedded Ops)*

Thinking about infrastructure before the end – getting more rapid development. Releases smaller and less risky – fewer outages.
(Training manager)

Speed up changes to infrastructure needed for a release (fewer checkpoints, and in our control)
(Tester/QA)

Better shared knowledge – we can diagnose and fix problems quicker.
(Tester/QA)

Benefits of DevOps Adoption – literature [13]

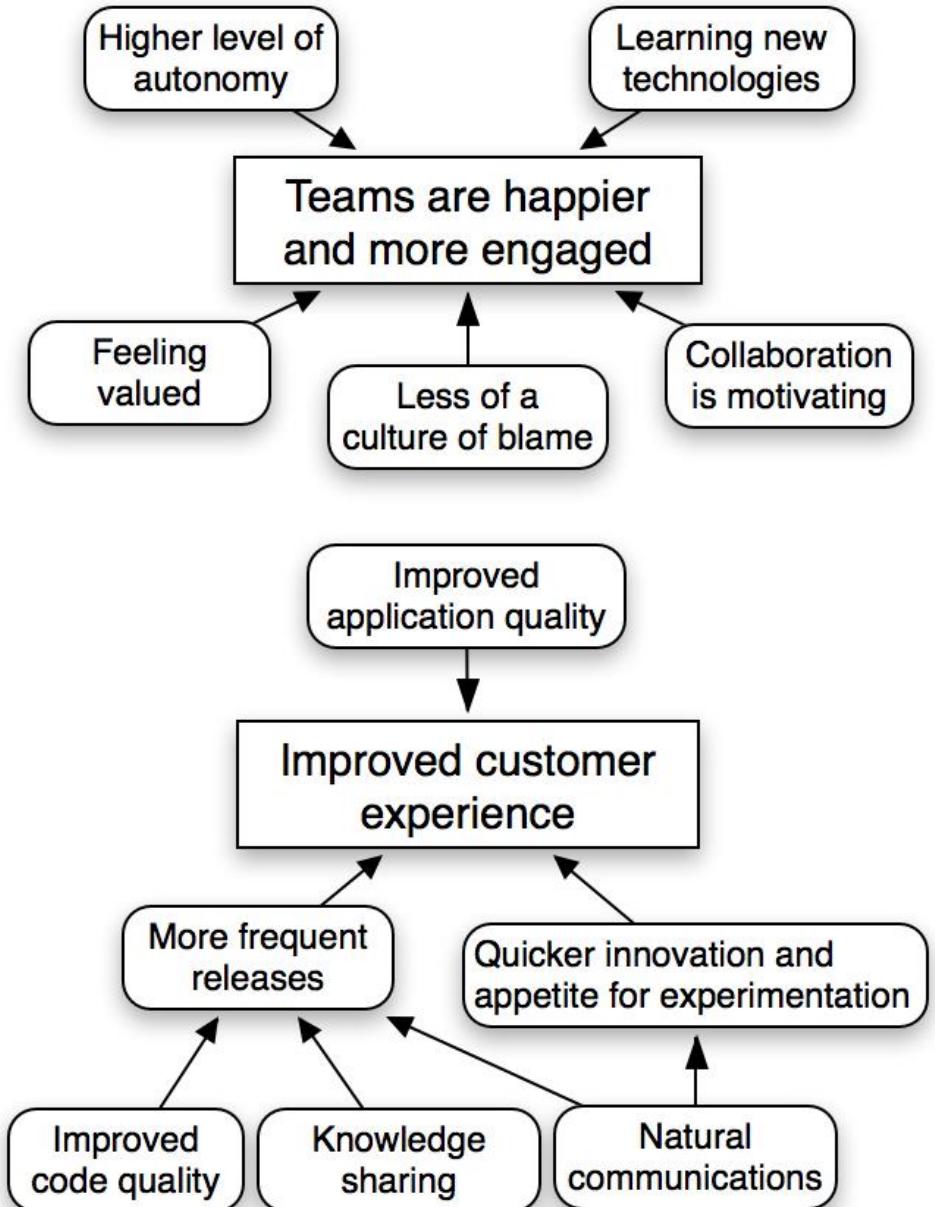


Benefits of DevOps Adoption

More deploys means **faster time-to-market** and an **enhanced feedback loop**. With an enhanced feedback loop, organizations have a better idea of customer reactions to features and consequently, better means of **continuous improvement.**[5]

DevOps also contributes to **stability enhancements**. Stability in this context stands for error-free operation of a system and the ability to keep processing requests. **Enhanced change success rate and (60 x) deployment success rate**[5]

Perceived Benefits of Working the DevOps way



Enablers and success factors of DevOps Adoption

What are some **success** factors of DevOps adoption?

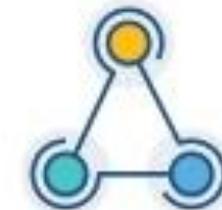
What capabilities, culture, practices, technologies and tools are significant enablers of successful DevOps adoption?



Agile and Lean



Automation



Service Frameworks



Enablers and success factors

Trusting the production team to deploy was difficult (QA Release manager)

The team can set up their own dashboard for monitoring whatever they think is important post-deployment of a feature. (Ops in Team)

Lots of communications – you can never have enough. (Developer)

Cross training the development team on ops and having ops take part in the production team meetings. It's hard just getting the right people, though. (Training manager)

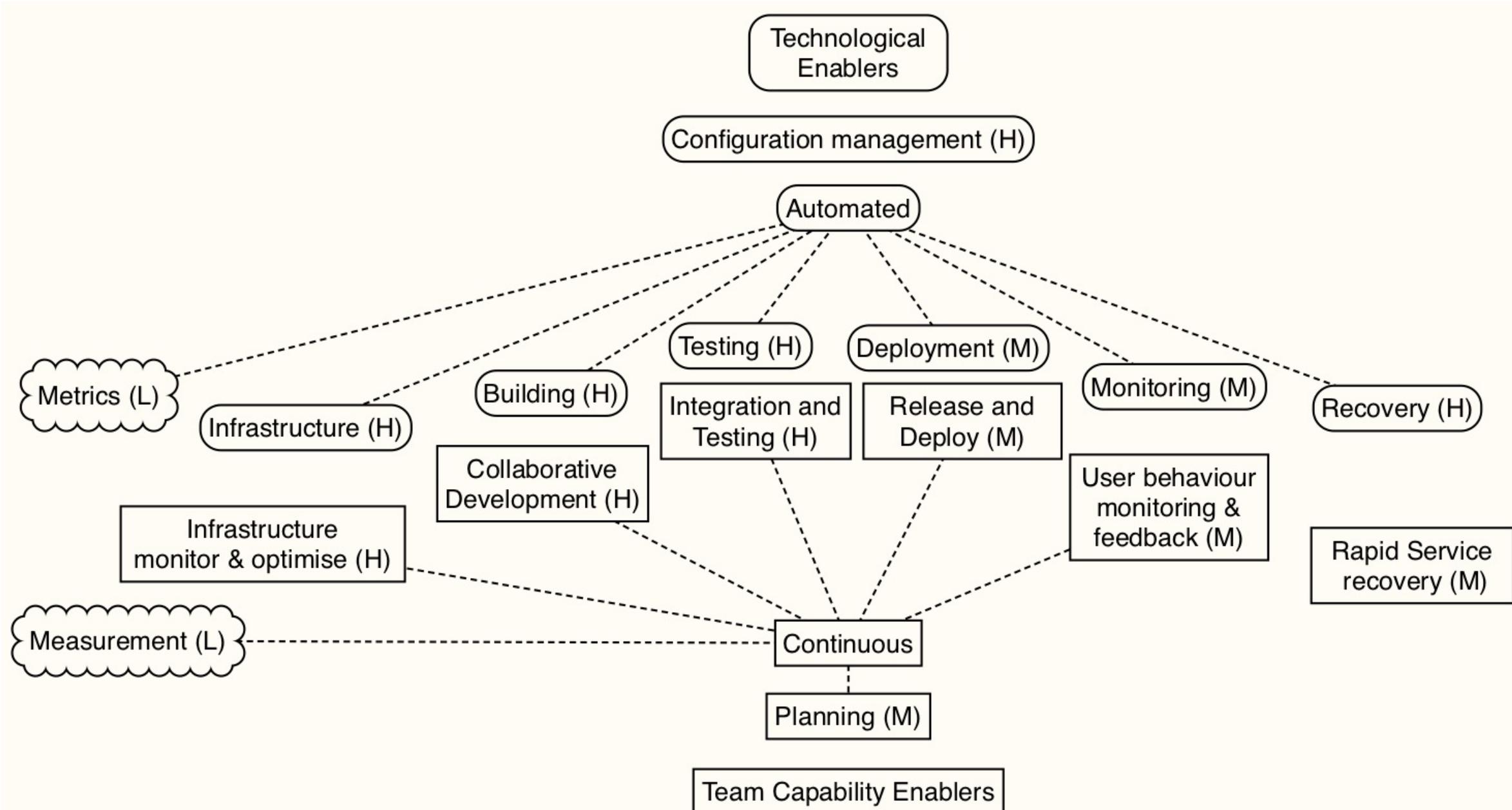
The switch from a monolithic application to microservices was a big thing.(Team lead Infrastructure)

Having a set of tools to automate testing and builds and continuous integration(Tester/QA)

Enablers of Value of DevOps Adoption

Capabilities	Continuous planning Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behavior monitoring and feedback Service failure recovery without delay
Cultural Enablers	Shared goals, definition of success, incentives Shared ways of working, responsibility, collective ownership Shared values, respect and trust Constant, effortless communication Continuous experimentation and learning
Technological Enablers	Build automation Test automation Deployment automation Monitoring automation Recovery automation Infrastructure automation Configuration management for code and infrastructure

Technical and team capability Enablers of success in DevOps



Challenges and Barriers of DevOps

What are some inhibitors/barriers to successfully adopting and continuing with DevOps

What are significant challenges in implementing DevOps in practice?



Challenges?

Staffing is probably our biggest challenge. (QA Release manager)

The big mental shift was huge (Developer)

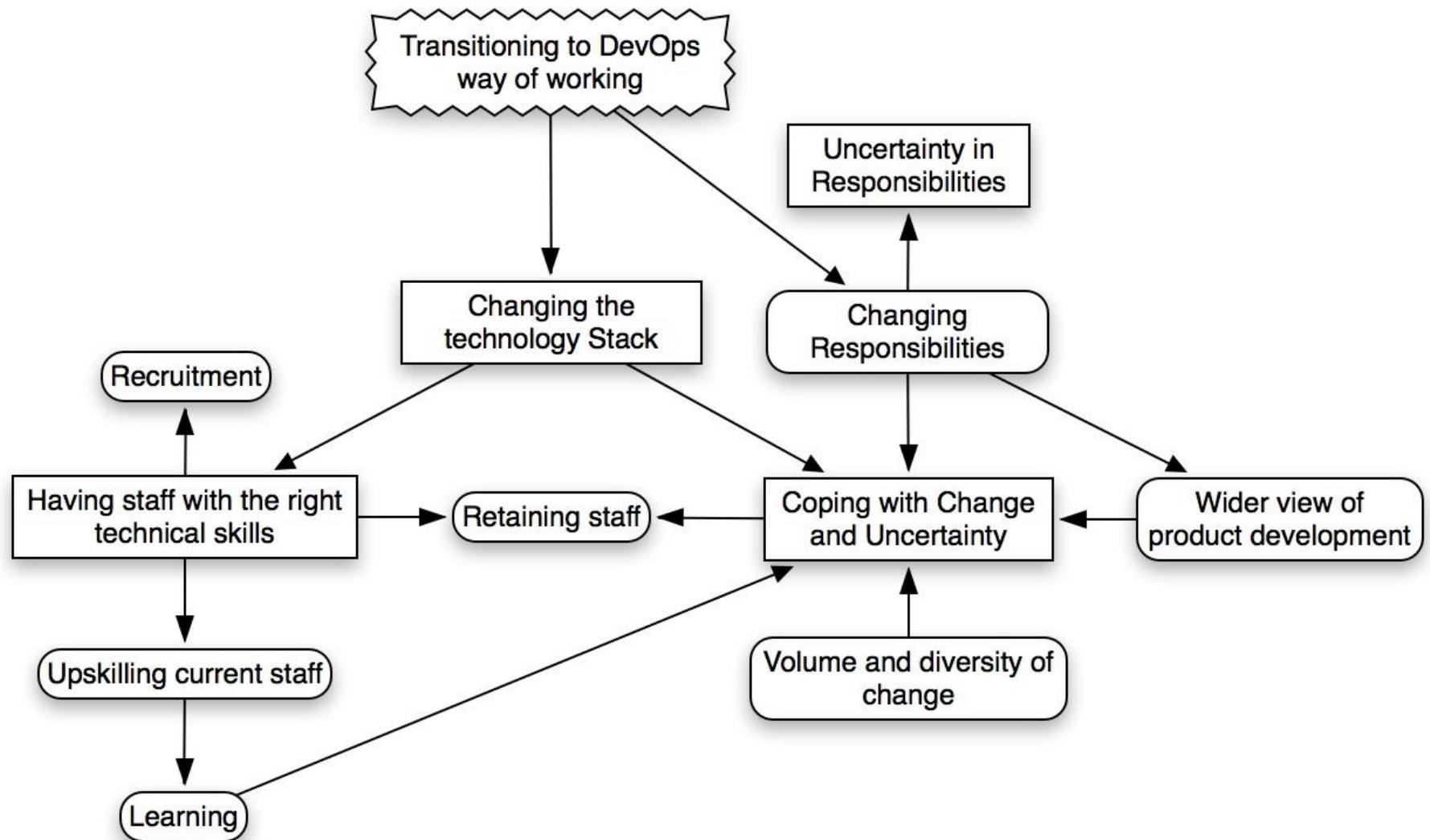
Changing the technology stack to the cloud and micro-services was a big enabler, but it was incredible complex. (Team lead Infrastructure)

Windows is a big challenge. It's hard to get the right tools together for automation.(Ops in Team)

Upskilling the entire team so anyone has the capability to be on-call. (Training manager)

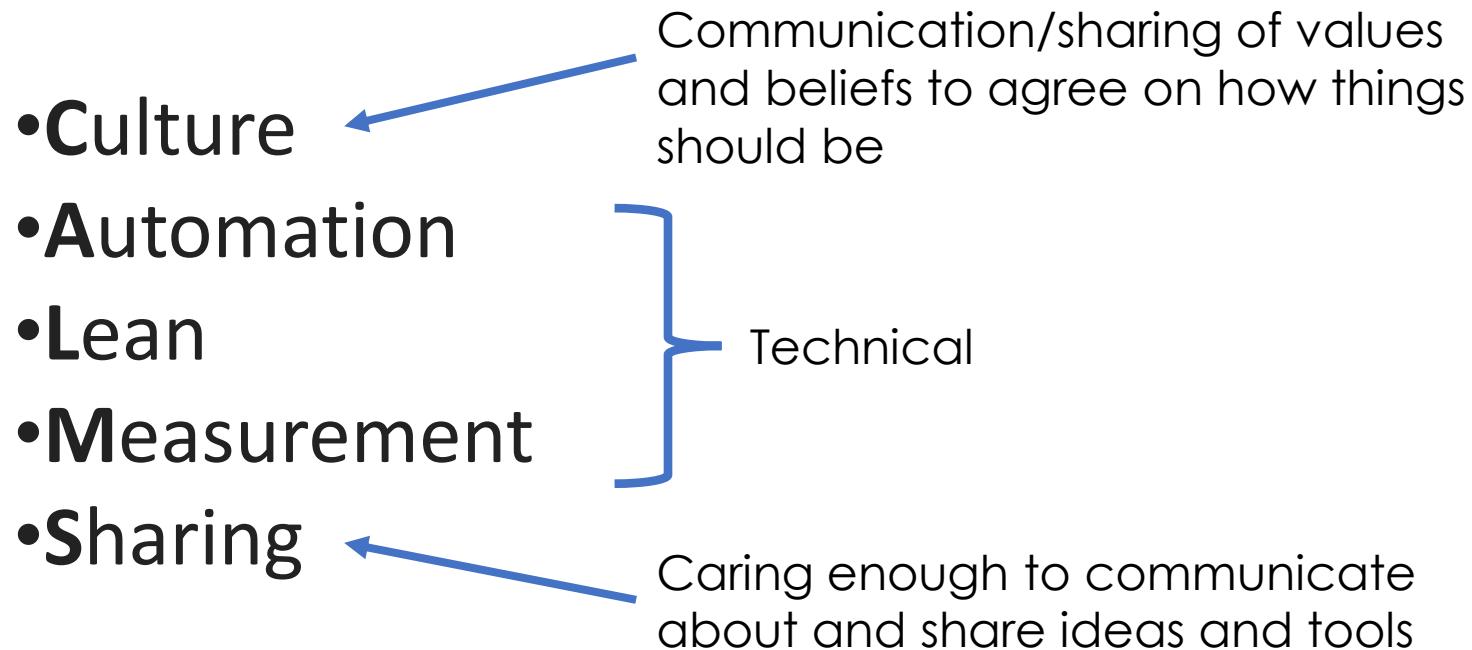
There are so many new tools and ideas, it's hard just keeping up. (Tester/QA)

Challenges in working the DevOps way



DevOps principles: Keep your CALM(S) first (then select some tools to support this...)

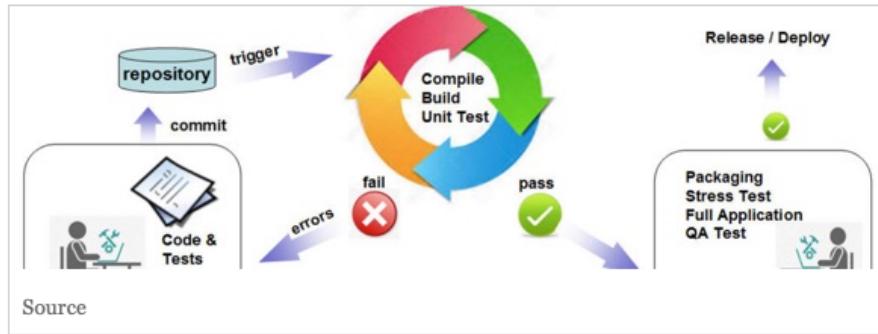
CALMS is a very accurate description of how to practice DevOps:



<https://thenextweb.com/syndication/2020/06/05/get-the-fundamentals-of-devops-right-then-worry-about-tools/>

Technology & DevOps

Automated Testing Tools



- #1 Integration testing tool of 2020: Cucumber
- #1 End-To-End Testing Tool of 2020 — Functional: SoapUI Pro
- #1 End-To-End Testing Tool of 2020 — Load Testing: LoadRunner

We must start an evaluation of automated testing tools by first fitting them into the testing pyramid. The testing pyramid has 4 layers:

- **Unit** — This is your base of all automated testing. As far as volume is concerned, you should have the most unit tests compared to other types. These tests should be written and run by software developers to ensure that a section of an application (known as the “unit”) meets its design and behaves as intended.
- **Component** — The main objective of component testing is to verify the input/output behavior of the test object. This ensures that the test object’s functionality is working correctly, as per the desired specification.
- **Integration** — This is the phase in testing in which individual software modules are combined and tested as a group.
- **End-to-End** — This layer is self-explanatory. We’re looking at the flow of an application, right from the start to the finish, and making that it’s behaving as expected.

- Development and Build Tooling
- **Automated Testing Tools**
- Deployment Tooling
- Runtime DevOps Tooling
- Collaboration DevOps Tooling

<https://medium.com/better-programming/must-learn-devops-tools-for-2020-1a8a2675e88f>

Technology & DevOps

- **The Periodic Table of DevOps Tools**

<https://digital.ai/learn/devsecops-periodic-table/>

Summary

DevOps is a way of working that combines culture and values with practices and tools to provide the development team with capability to continuously develop, deploy and monitor new features

Drivers – some context dependent. Expectation is that more frequent delivery will result in increased agility and better customer experience.

Benefits – some context dependent. Frequent delivery resulted in increased collaboration, communications, agility and better customer experience. Value all round!

Enablers – clear understanding of value, responsibilities. Upskilling support. Supporting automation tools. Architecture.

Challenges – change management and supporting it. Finding staff, upskilling and retaining staff.



DevOps Capabilities, Practices, and Challenges: Insights from a Case Study

M. Senapathi, J. Buchan and H. Osman

In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (EASE'18). ACM, New York, NY, USA, 57-67. DOI: <https://doi.org/10.1145/3210459.3210465>

Emerging Trends for Global DevOps: A New Zealand Perspective

Hussain, W., Clear, T., & MacDonell, S.

In D. Cruzes & A. Sharma (Eds.), *Proceedings 2017 IEEE 12th International Conference on Global Software Engineering* (pp. 21-30). IEEE. <https://doi.org/10.1109/ICGSE.2017.16>

References

- [1] B. Fitzgerald, N. Forsgren, K.-J. Stol, J. Humble, and B. Doody, "Infrastructure Is Software Too!," 2015.
- [2] T. A. Limoncelli and D. Hughes, "LISA'11 Theme—"DevOps: New Challenges, Proven Values"," *USENIX; login: Magazine*, vol. 36, 2011.
- [3] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps Adoption Benefits and Challenges in Practice: A Case Study," in *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings* 17, 2016, pp. 590-597.
- [4] S. Elliot, "DevOps and the cost of downtime: Fortune 1000 best practice metrics quantified," *International Data Corporation (IDC)*, 2014.
- [5] J. Hamunen, "Challenges in adopting a Devops approach to software development and operations," 2016.
- [6] J. Smeds, K. Nybom, and I. Porres, "DevOps: a definition and perceived adoption impediments," in *International Conference on Agile Software Development*, 2015, pp. 166-177.
- [7] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective*: Addison-Wesley Professional, 2015.
- [8] J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang—bridging the gap between development and operations," in *European Conference on Service-Oriented and Cloud Computing*, 2014, pp. 108-122.
- [9] L. Baresi, S. Guinea, A. Leva, and G. Quattrochi, "A discrete-time feedback controller for containerized cloud applications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 217-228.
- [10] K. Nybom, J. Smeds, and I. Porres, "On the Impact of Mixing Responsibilities Between Devs and Ops," in *International Conference on Agile Software Development*, 2016, pp. 131-143.
- [11] M. Hütermann, "Gain Fast Feedback," in *DevOps for Developers*, ed Berkeley, CA: Apress, 2012, pp. 81-94.
- [12] J. Sussna, "Cloud and DevOps: A Marriage Made in Heaven," *Introducing DevOps to the Traditional Enterprise/eMag*, vol. 14, 2014.
- [13] S. Nagpal and A. Shadab, "Literature Review: Promises and Challenges of DevOps."
- [14] V. P. R. Kumar and V. Babu, "Devops-a review," *Image and Video Processing*, p. 32, 2012.
- [15] P. Duvall, "Breaking down barriers and reducing cycle times with DevOps and continuous delivery," Retrieved from GigaOM Pro website: <http://try.newrelic.com/rs/newrelic/images/GigaOm-Pro-Report-Breaking-down-barriers-and-reducing-cycle-times-with-devops-and-continuous-delivery.pdf>, 2012.
- [16] J. Humble and J. Molesky, "Why enterprises must adopt devops to enable continuous delivery," *Cutter IT Journal*, vol. 24, p. 6, 2011.
- [17] S. W. Hussaini, "Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, 2014, pp. 178-183.
- [18] J. Kim, C. Meirosu, I. Papafili, R. Steinert, S. Sharma, F.-J. Westphal, et al., "Service provider DevOps for large scale modern network services," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, 2015, pp. 1391-1397.
- [19] M. Goodwell. (2016, 2016-03-27). DevOps in a Microservices World. Available: <https://dzone.com/articles/devops-microservices-world>
- [20] L. E. Lwakatare, T. Karvonen, T. Sauvola, P. Kuvala, H. H. Olsson, J. Bosch, et al., "Towards DevOps in the Embedded Systems Domain: Why is It So Hard?," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, 2016, pp. 5437-5446.
- [21] J. Greenough, "How the 'Internet of Things' will impact consumers, businesses, and governments in 2016 and beyond," *Business Insider*, 2016.
- [22] A. Storms, "How security can be the next force multiplier in devops," *RSAConference,(San Francisco, USA)*, 2015.
- [23] H. Karl, S. Dräxler, M. Peuster, A. Galis, M. Bredel, A. Ramos, et al., "DevOps for network function virtualisation: an architectural approach," *Transactions on Emerging Telecommunications Technologies*, vol. 27, pp. 1206-1215, 2016.
- [24] ISACA, "Title," unpublished].
- [25] P. Labs, "State of DevOps Report," 2016.
- [26] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, pp. 42-52, 2016.
- [27] L. E. Lwakatare, P. Kuvala, and M. Oivo, "Dimensions of DevOps," in *International Conference on Agile Software Development*, 2015, pp. 212-217.
- [28] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps?: A Systematic Mapping Study on Definitions and Practices," in *Proceedings of the Scientific Workshop Proceedings of XP2016*, 2016, p. 12.
- [29] W. J. Geurts, "Faster is Better and Cheaper," in *INCOSE International Symposium*, 2016, pp. 1002-1015.
- [30] J. Clapham, "DevOps – The Reluctant Change Agent's Guide," *Introducing DevOps to the Traditional Enterprise/eMag*, vol. 14, 2014.
- [31] F. Colavita, "DevOps Movement of Enterprise Agile Breakdown Silos, Create Collaboration, Increase Quality, and Application Speed," in *Proceedings of 4th International Conference in Software Engineering for Defence Applications: SEDA 2015*, P. Ciancarini, A. Sillitti, G. Succi, and A. Messina, Eds., ed Cham: Springer International Publishing, 2016, pp. 203-213.

Later Developments and Framings - References

- Dennehy, D., & Conboy, K. (2017). Going with the flow: An activity theory analysis of flow techniques in software development. *Journal of Systems and Software*, 133, 160-173.
- Zhao, G. (2021-TBD). *Modelling Strategies for DevSecOps in a Global Setting: A Delphi Study* [Doctoral Thesis, Auckland University of Technology]. Auckland.



#171678945



Thank you!

Questions and Comments....



Tony Clear 2024

CISE ENSE701

I has a question...



50