

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

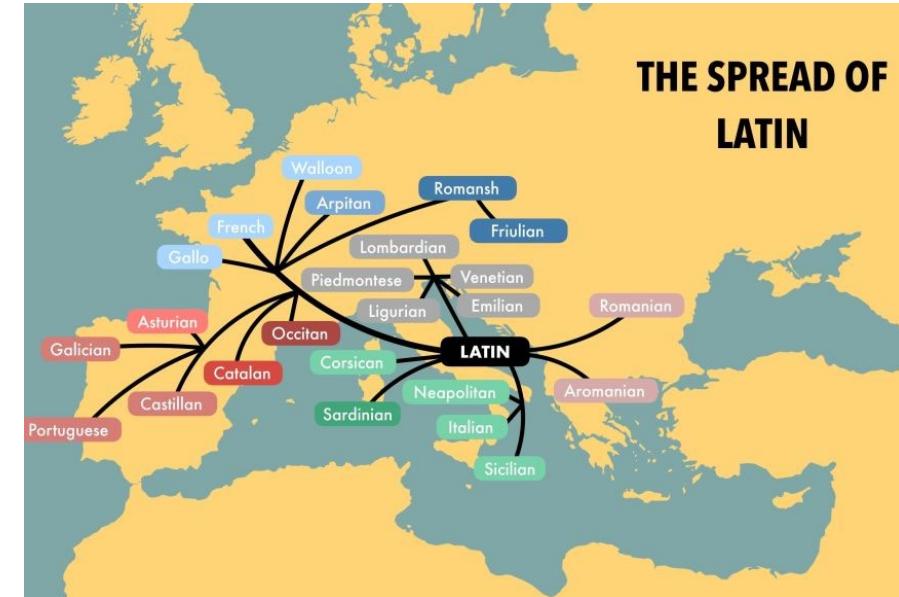
Held roles of discipline group leader, associate head of school, head of school, faculty associate dean research

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

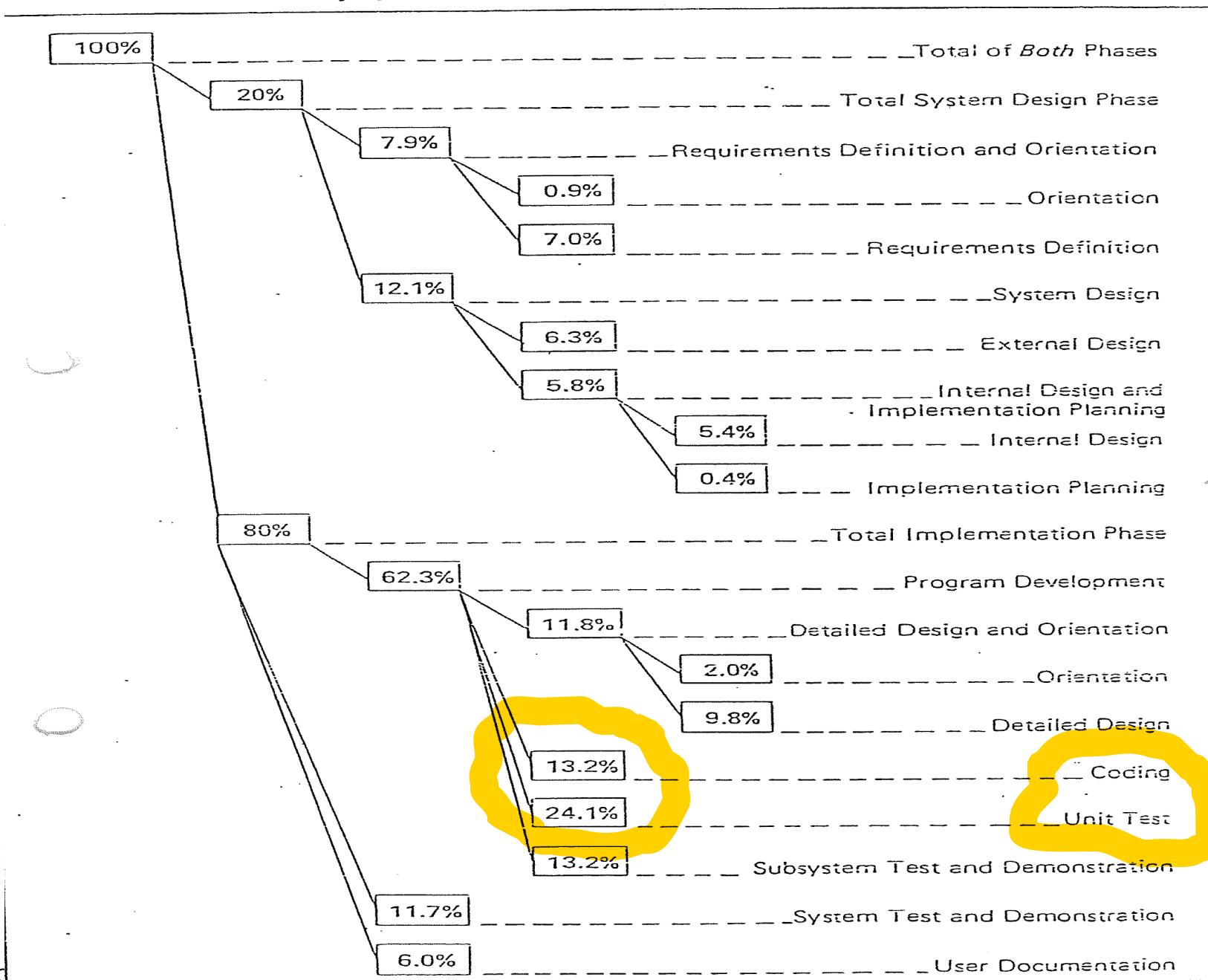
...**delivering functionality constantly** helped them “**better estimate** the cost of each feature, allow for project creep and to therefore have better control over the project as a whole”.

... ‘**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...

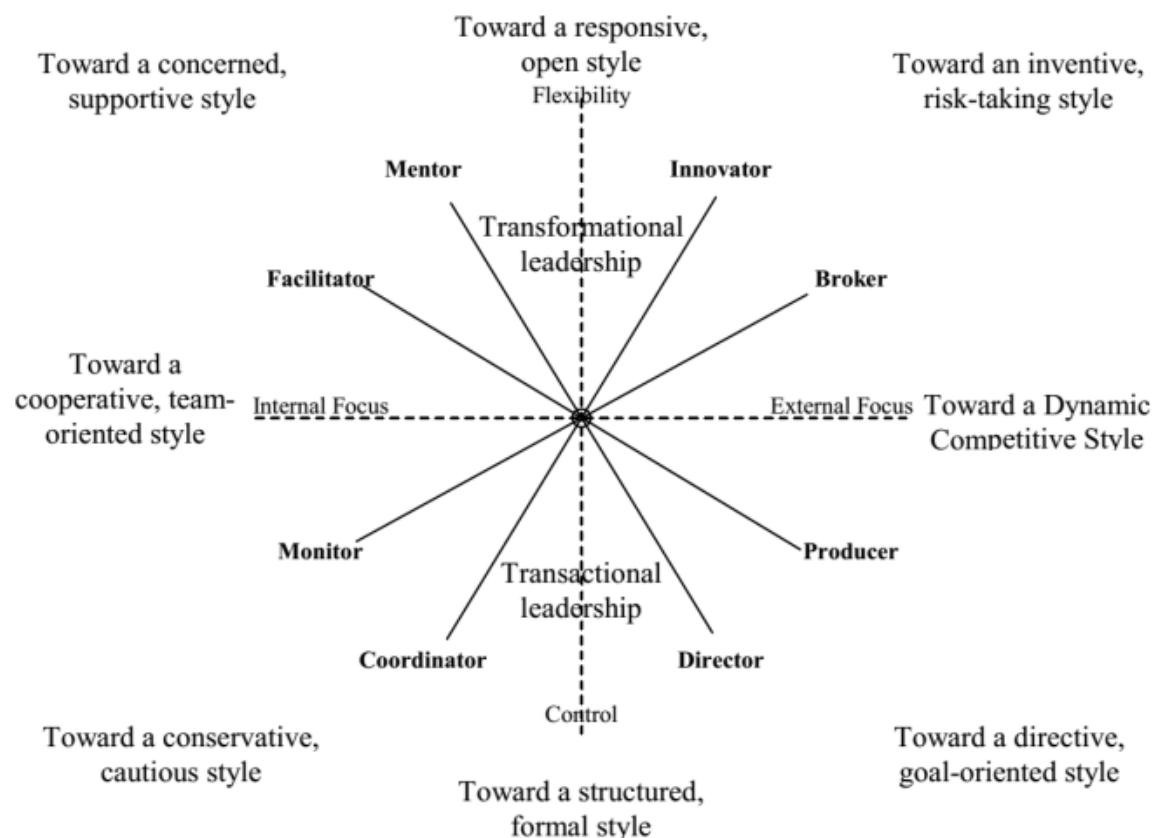


Figure 5: Competing Values Framework of Leadership Roles [43]

...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

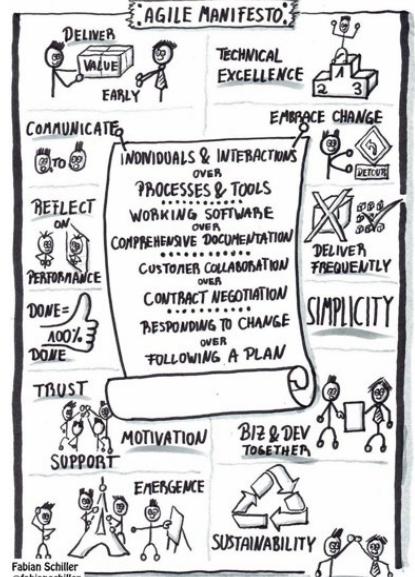
Ass 2 Knowledge Check (30%)

(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

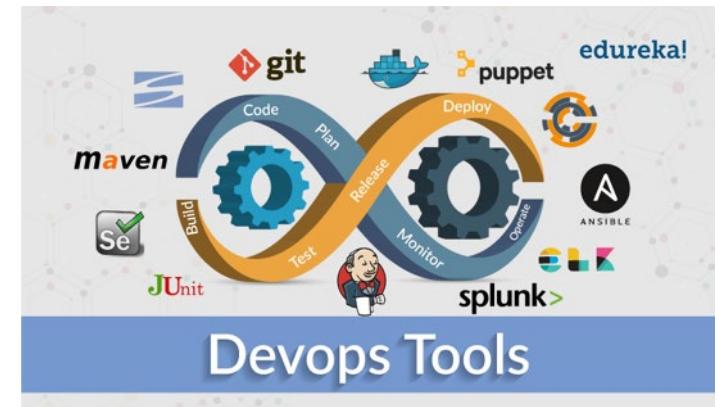
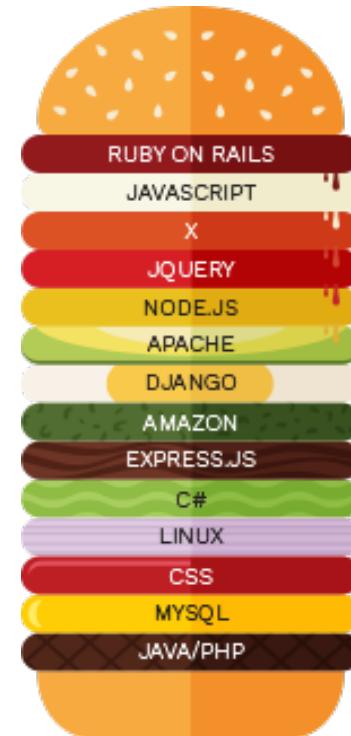
Automated testing/checking from the start – tool pipeline

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

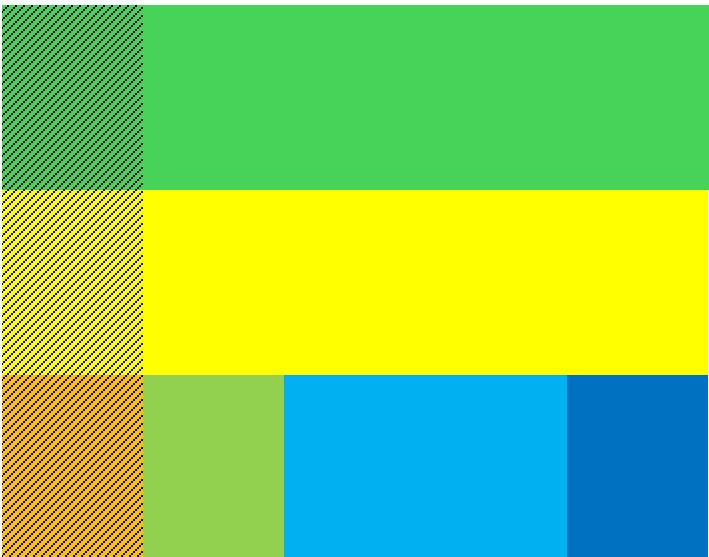
Week
No

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

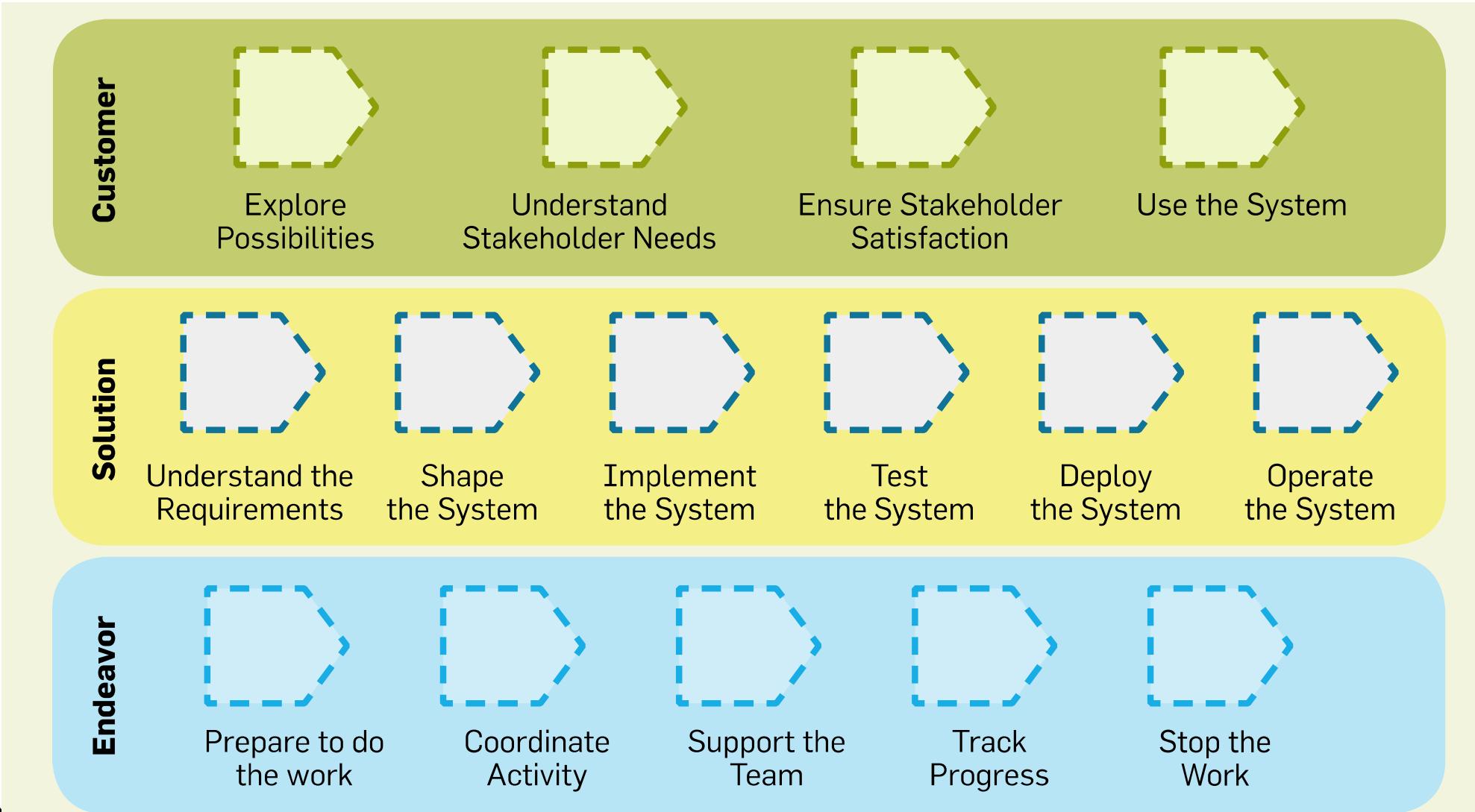
Maintain

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

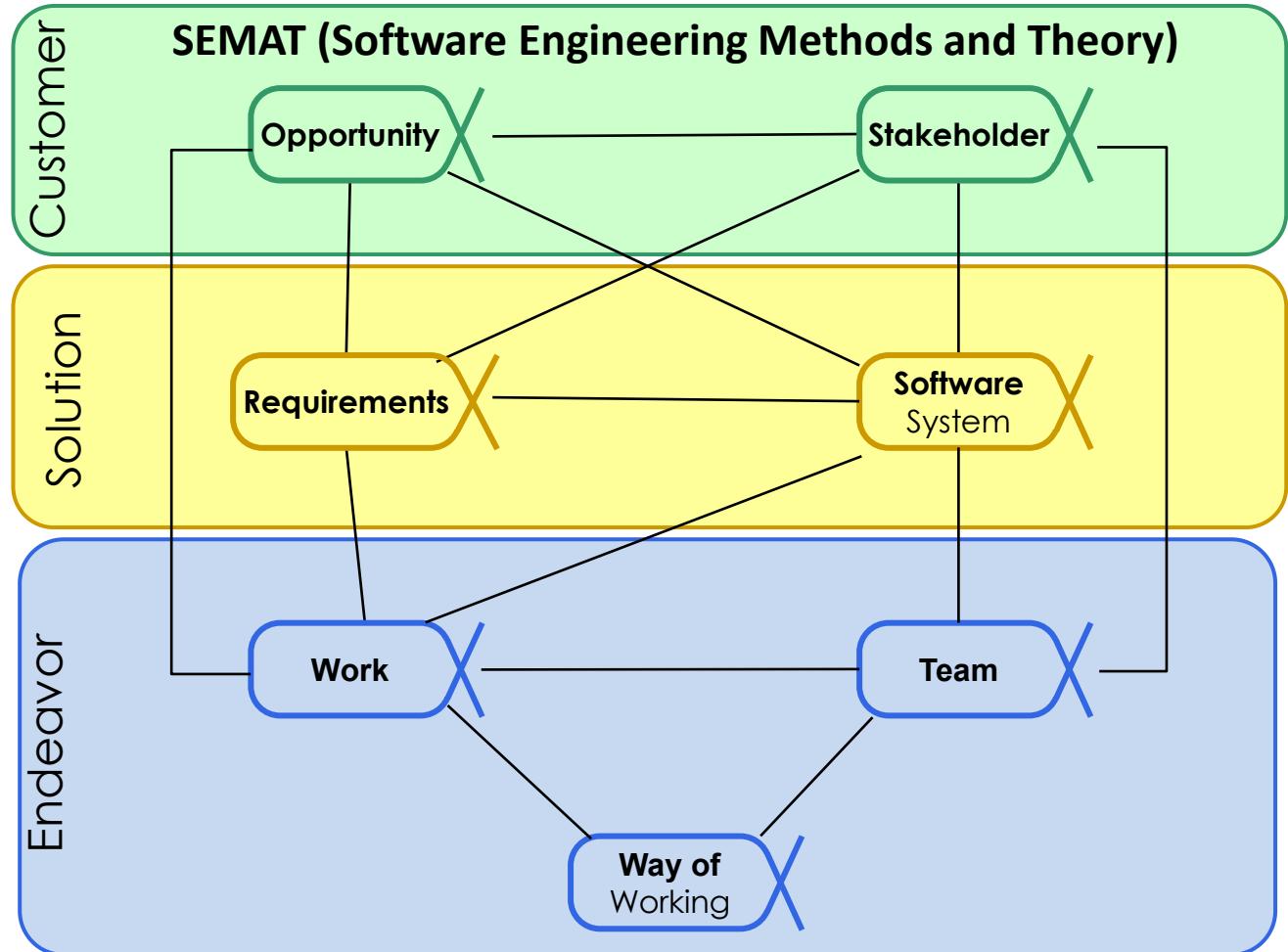
Maintain/update the work

Work = ?????

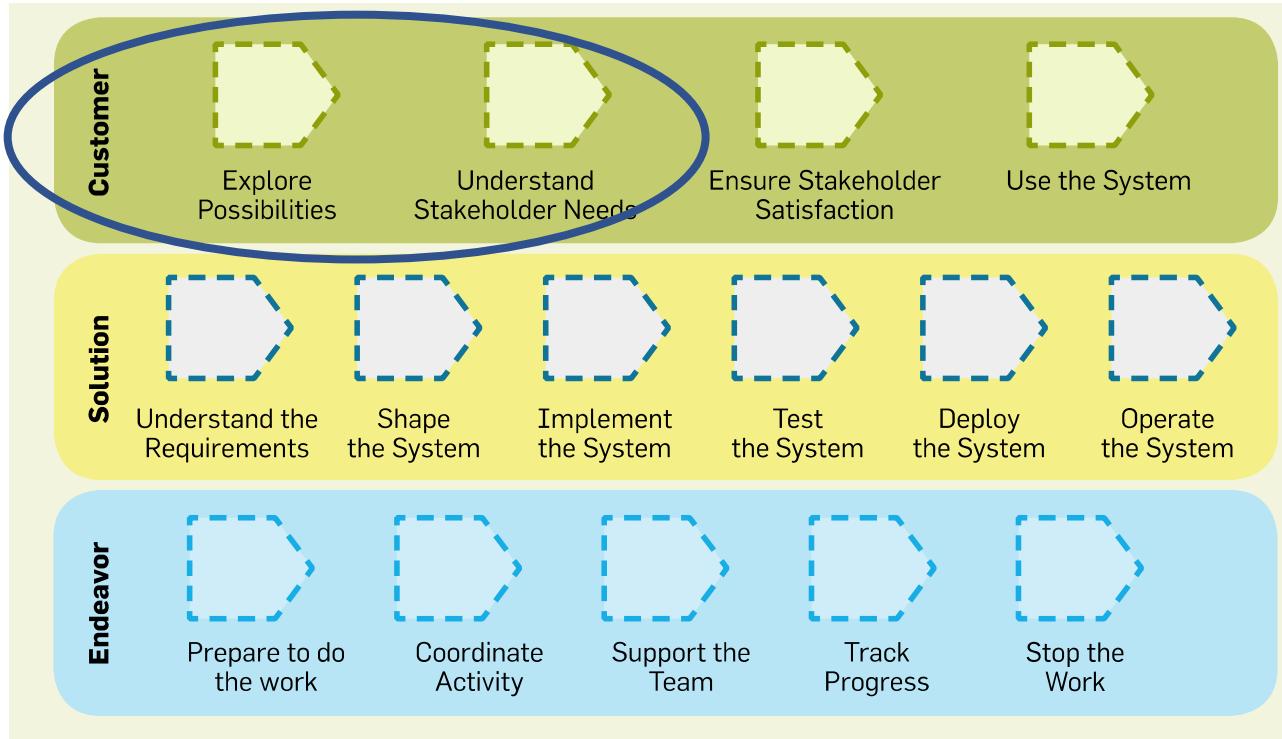
IS IT LINEAR?

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

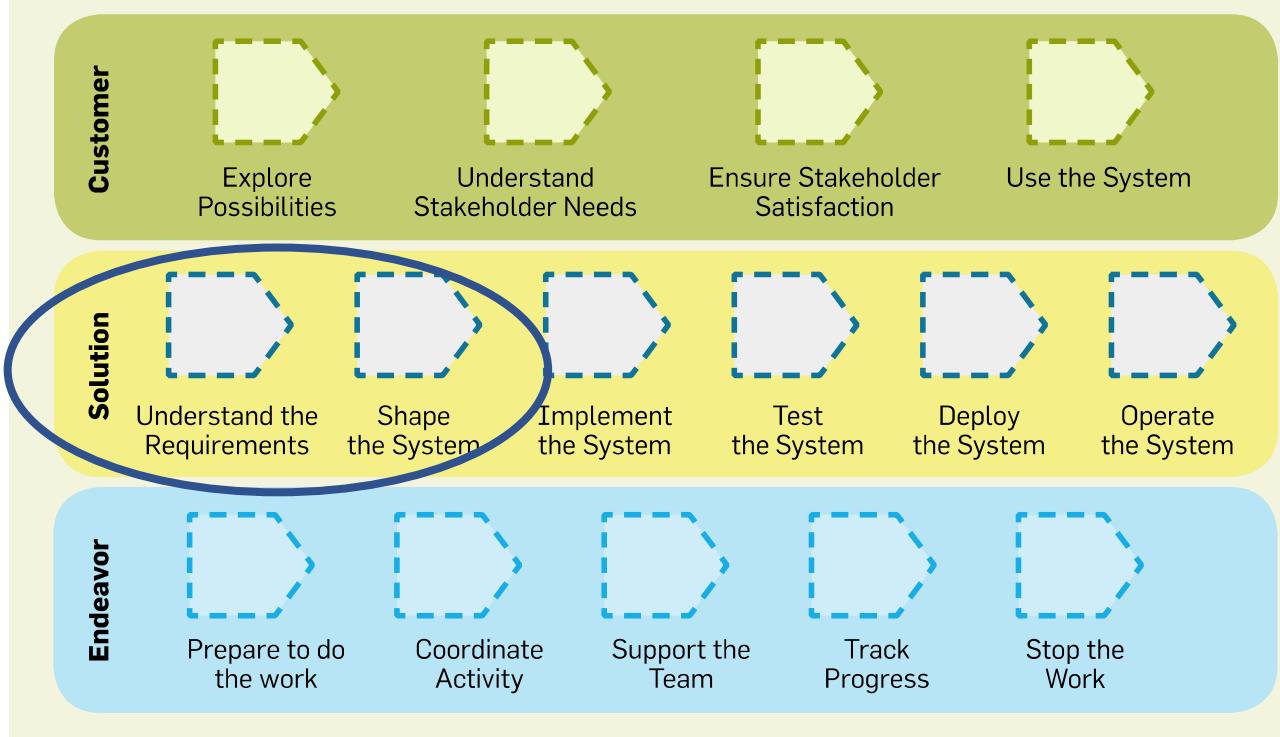


Work closely with the users and client – frequent interactions and coordination and feedback

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



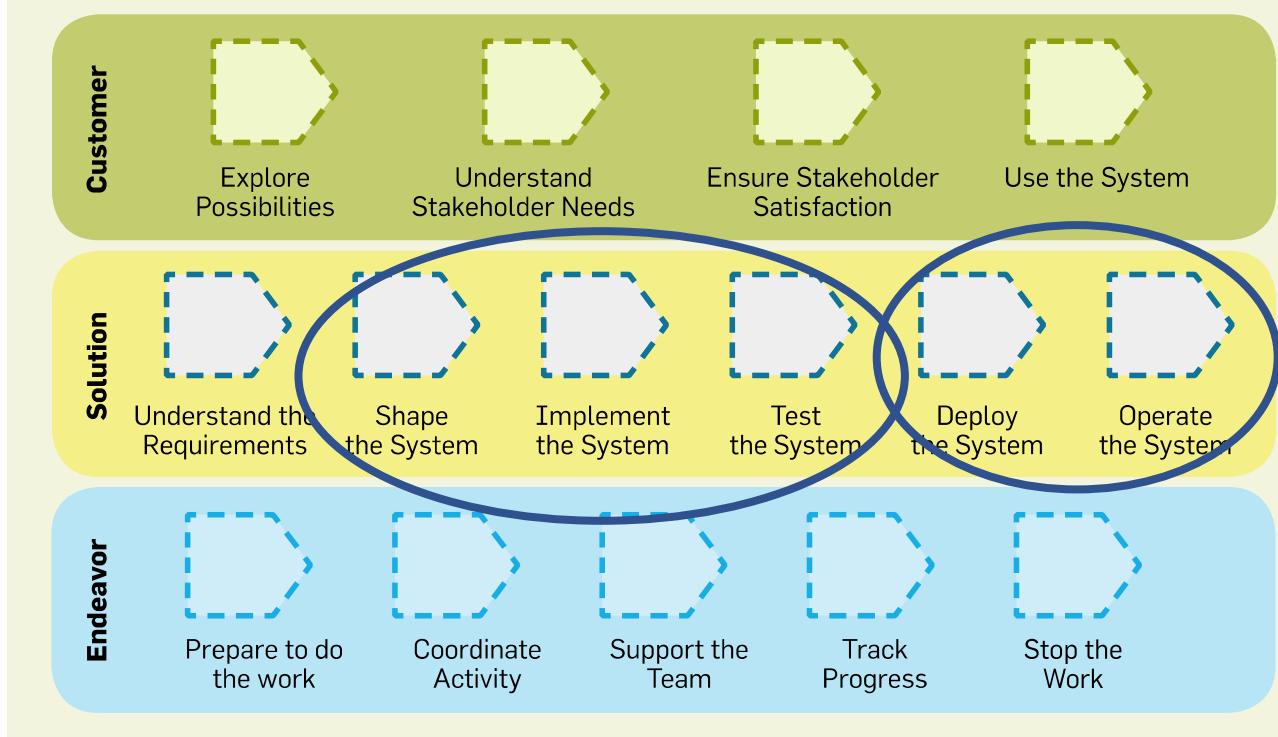
User stories

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

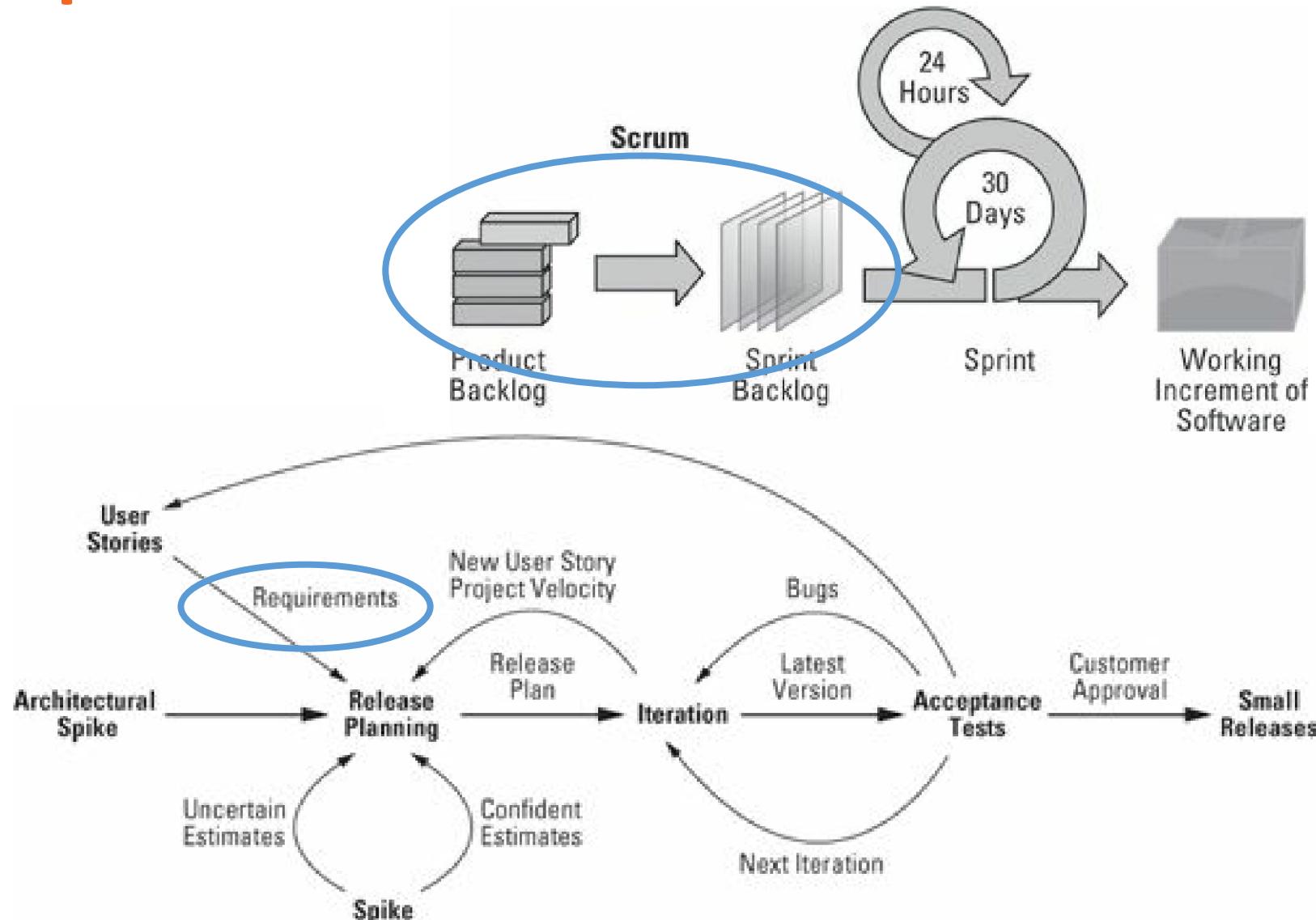
Automatically merge if pass tests

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 (1ny 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

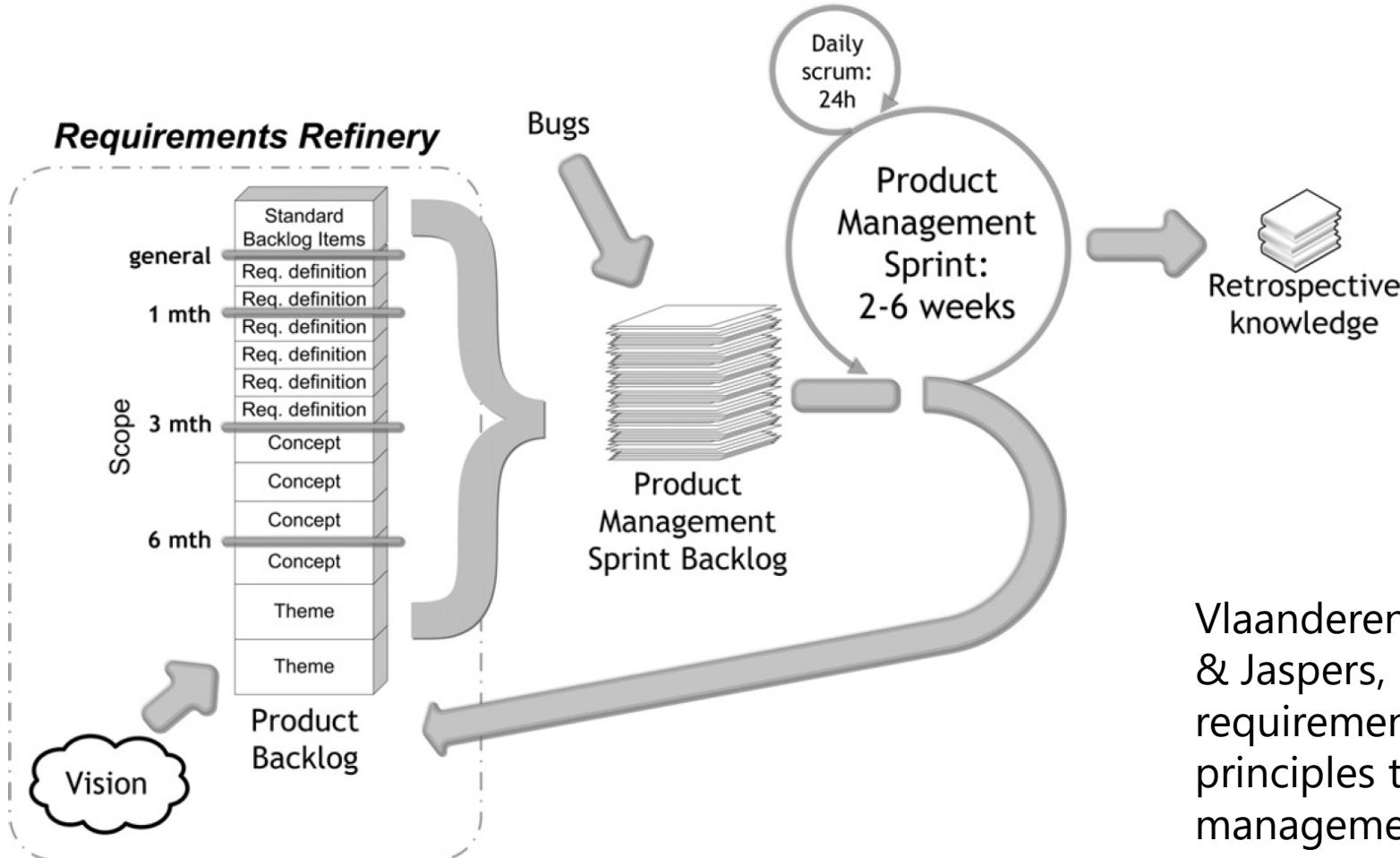


Fig. 1. Agile SPM knowledge flow.

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

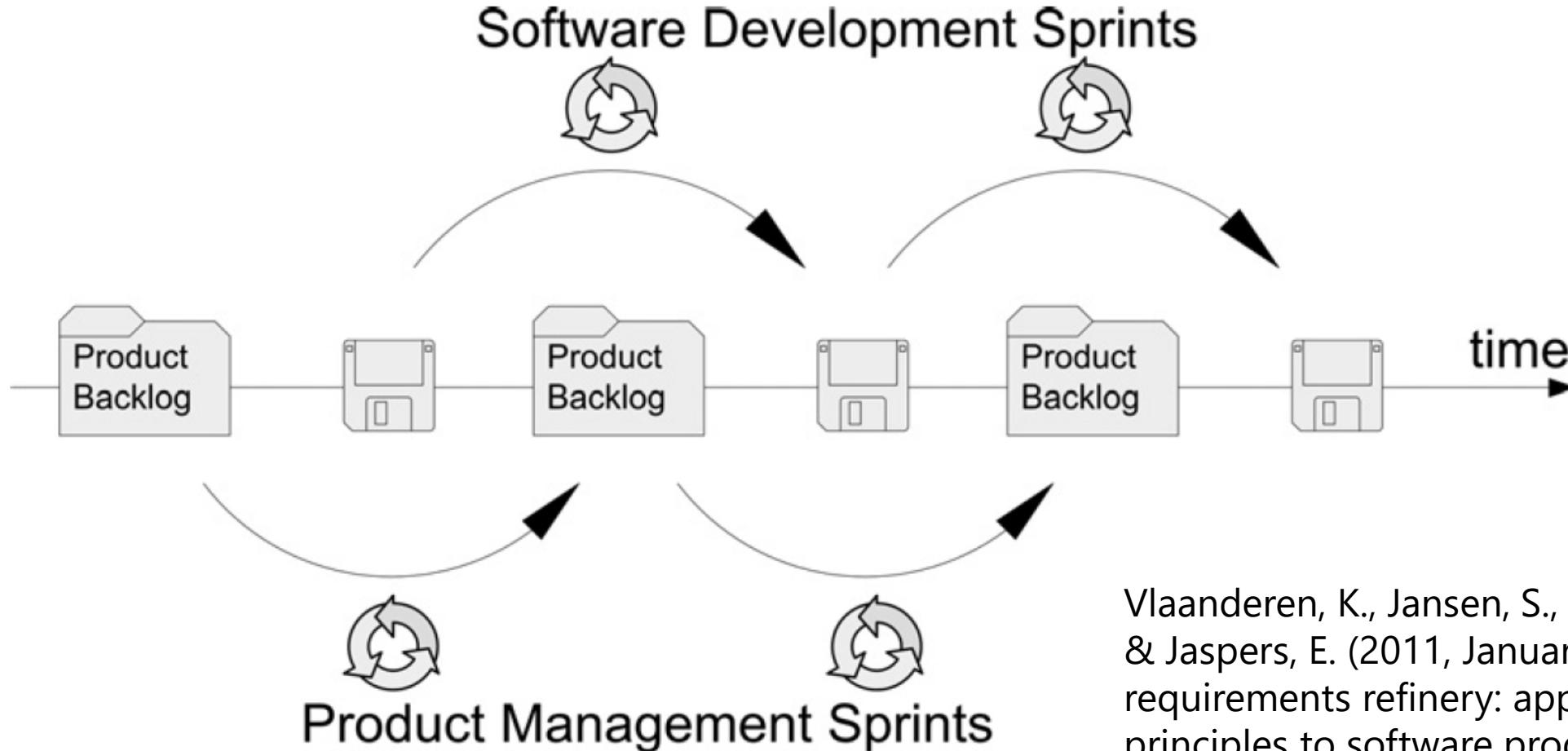
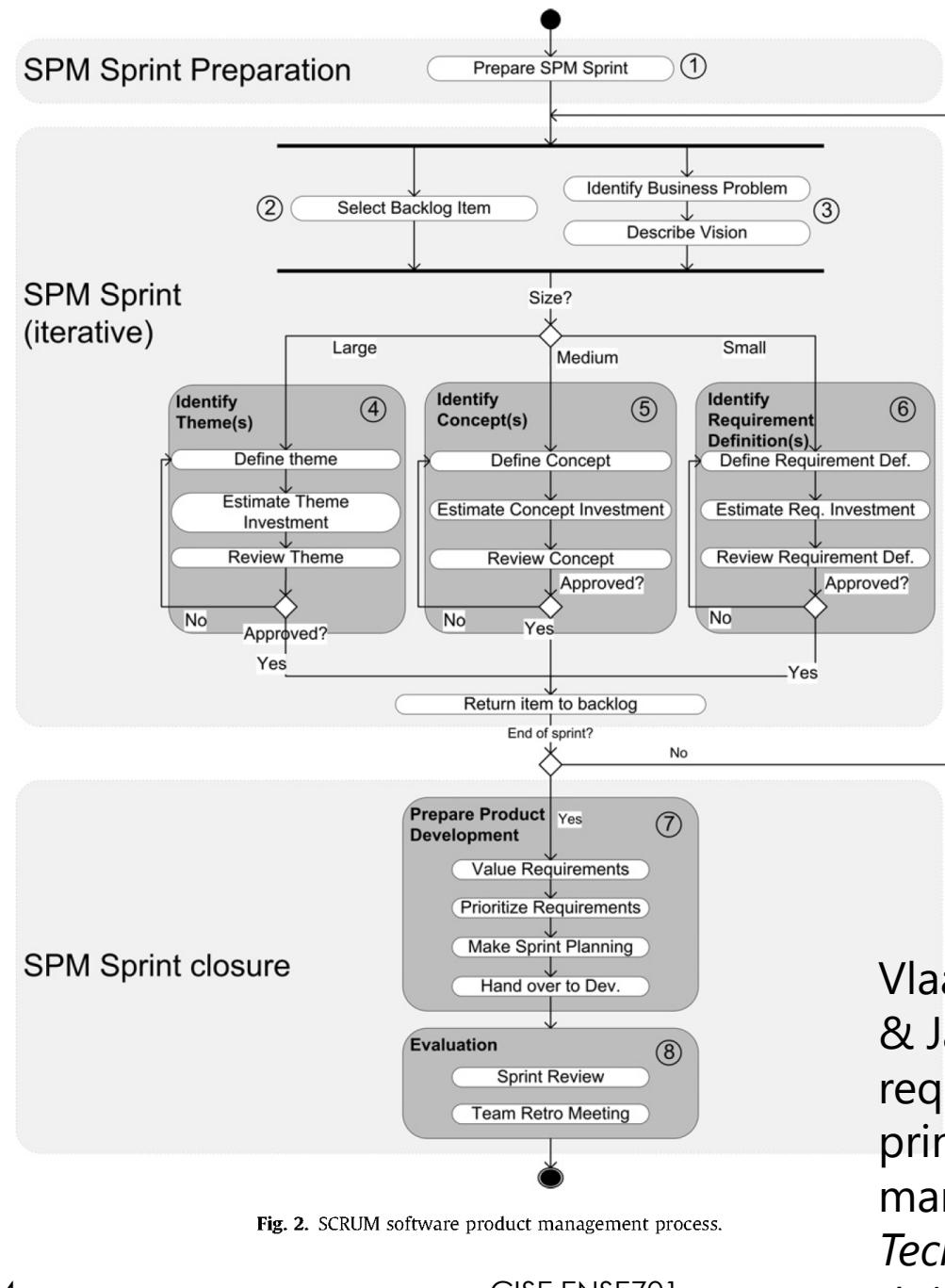


Fig. 3. Alternating sprints.

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

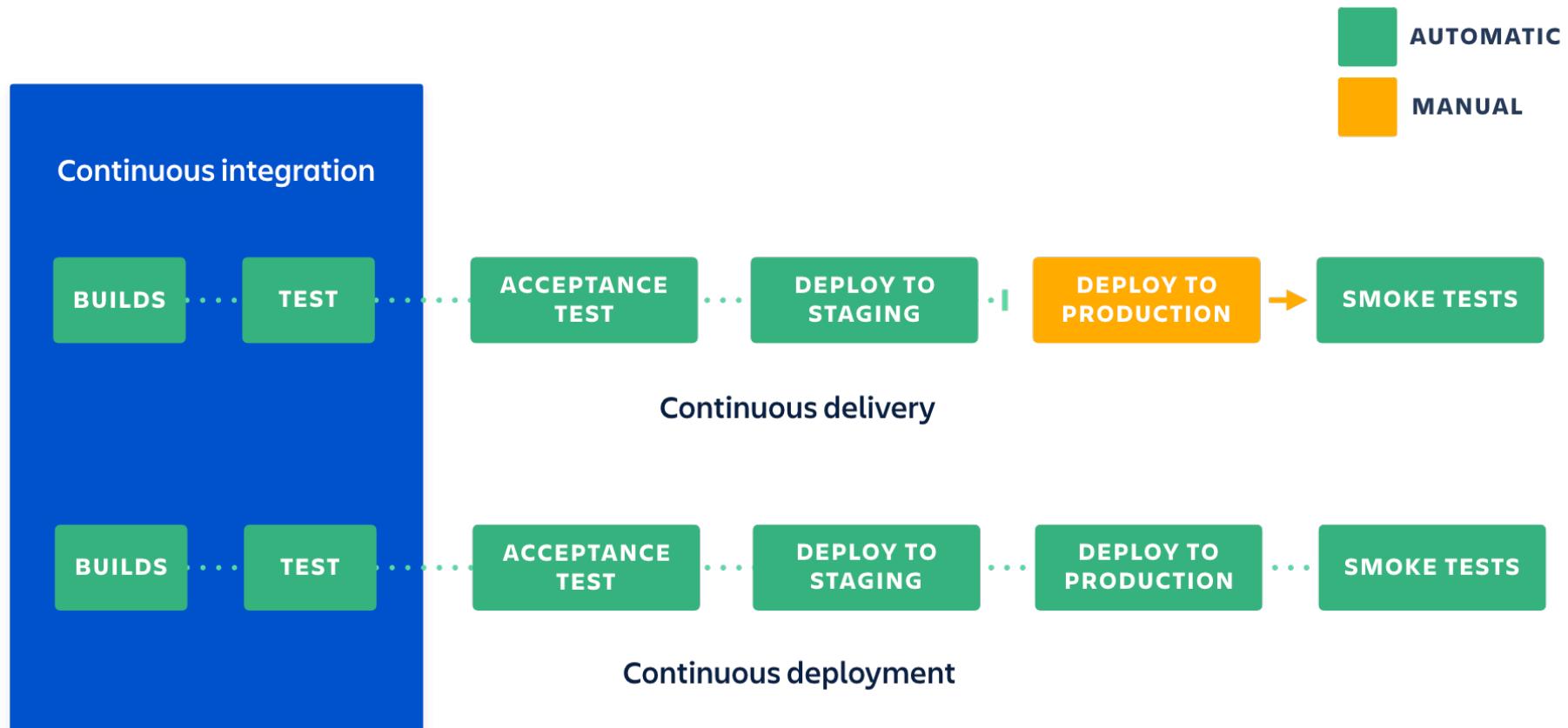
Software Product management Process



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

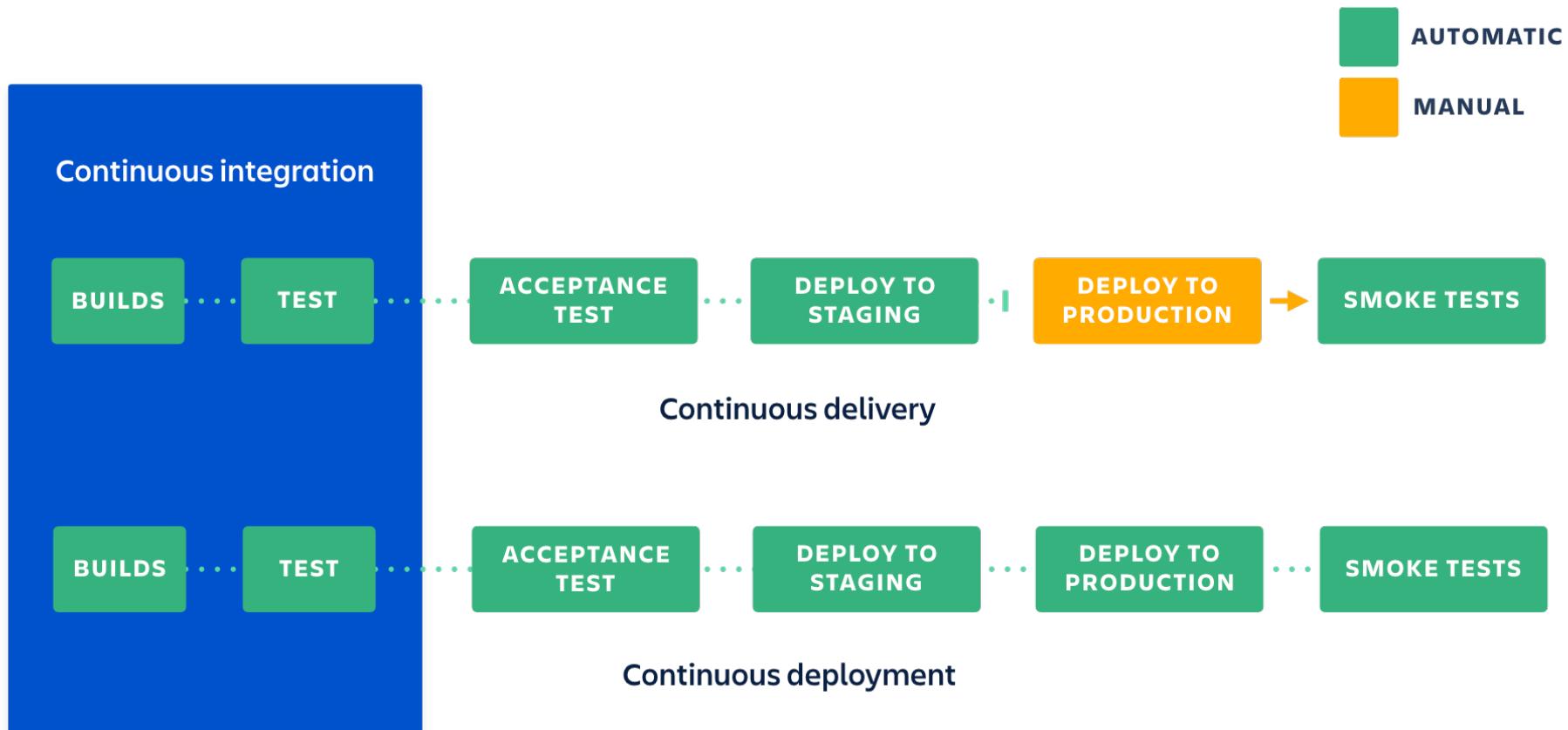
<https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>



CI/Continuous Integration in a Nutshell

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

Eric Minick IBM Cloud

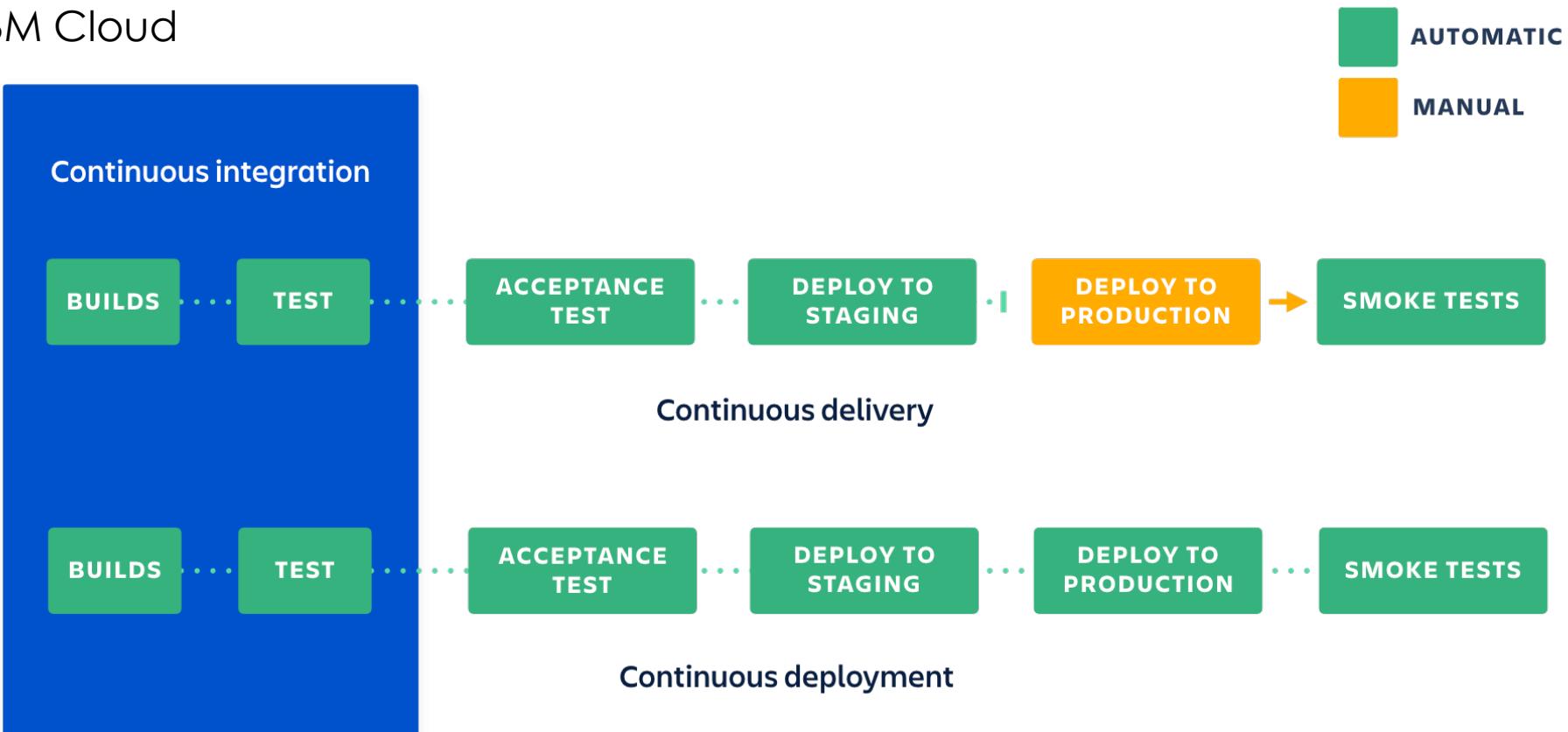


CI/CD Continuous Deployment (or Delivery) in a Nutshell

<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

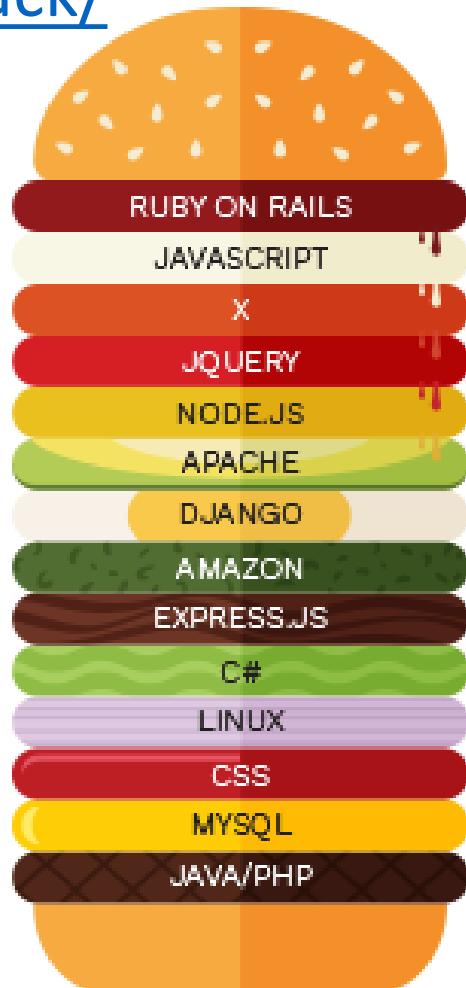
Enablers

- 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



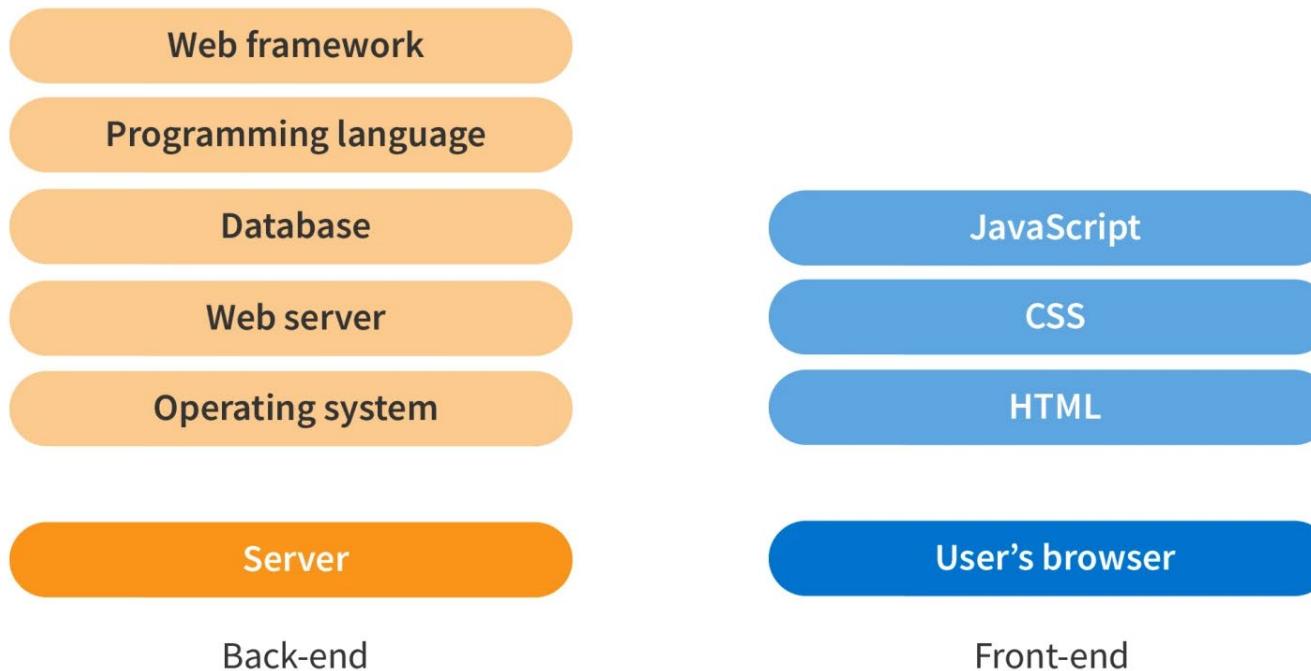
Technology stacks and full-stack developers

<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.

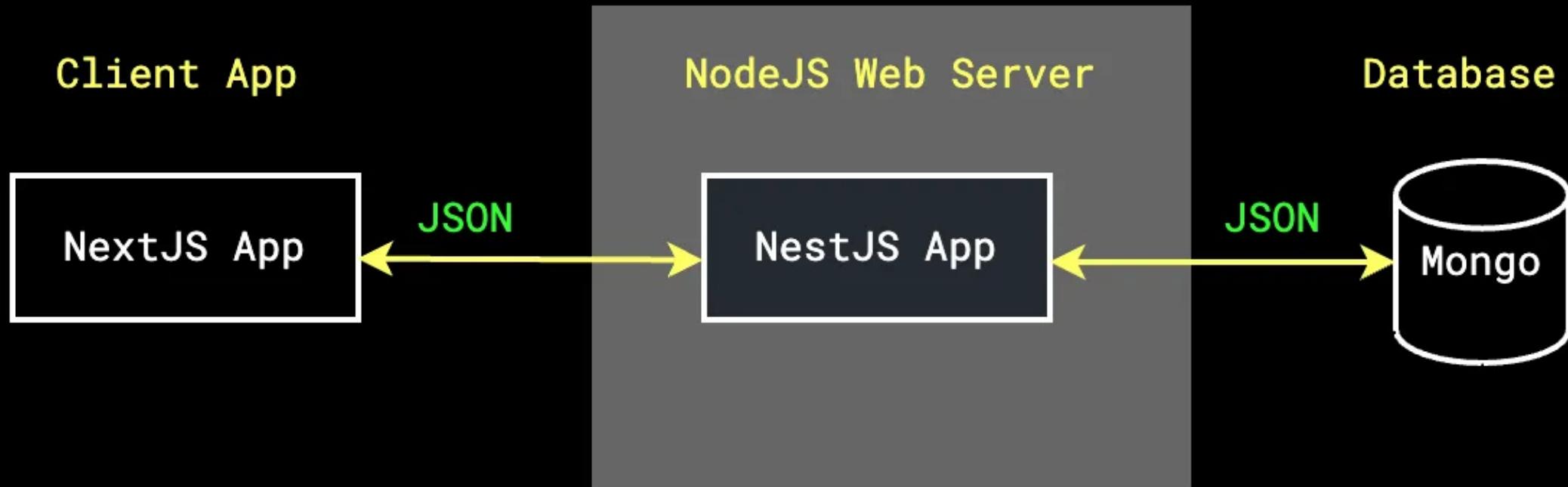
MNNN Stack in a nutshell



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>

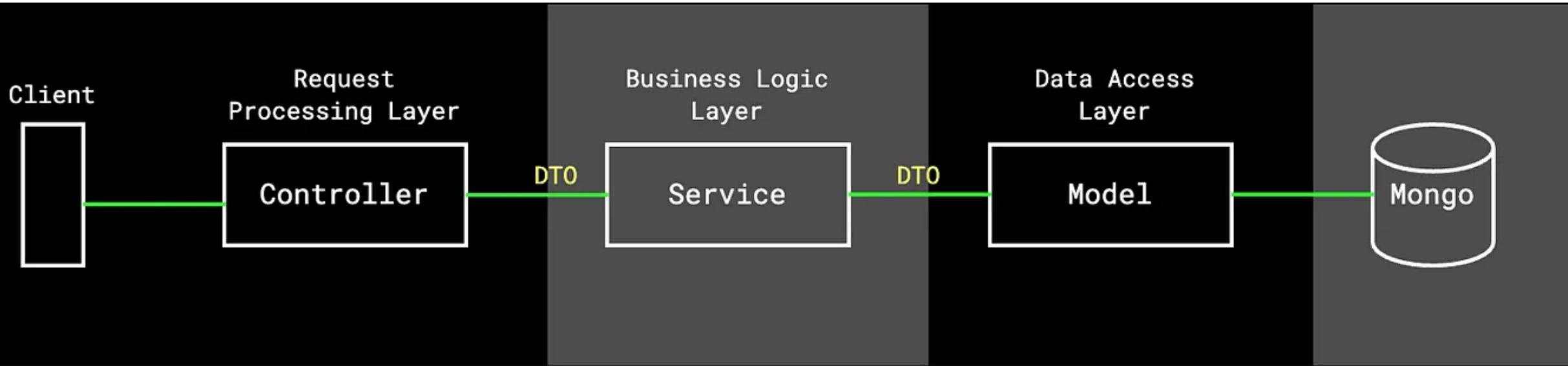
MNNN Components



Read this

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

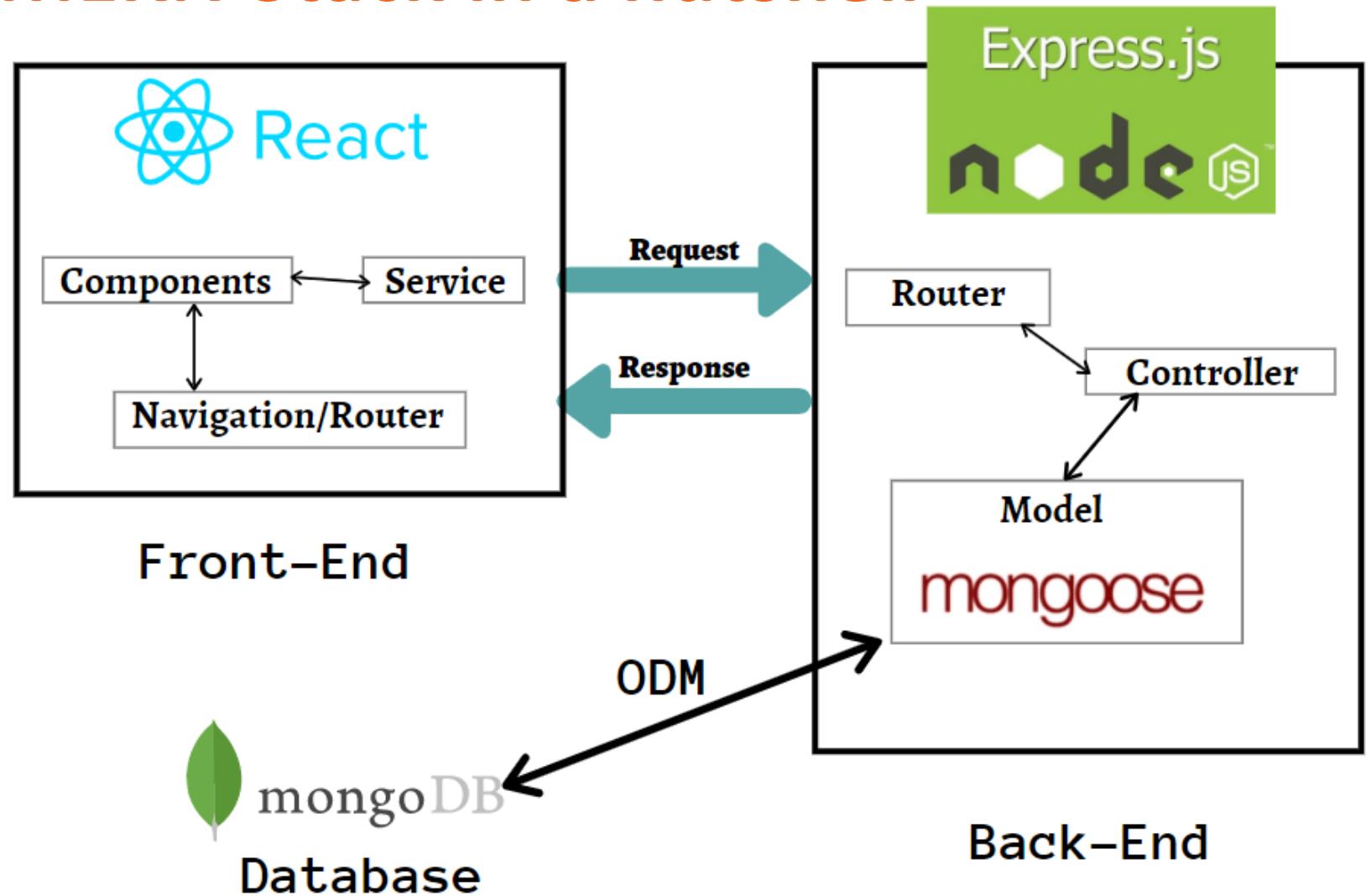
MNNN Layered Architecture



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

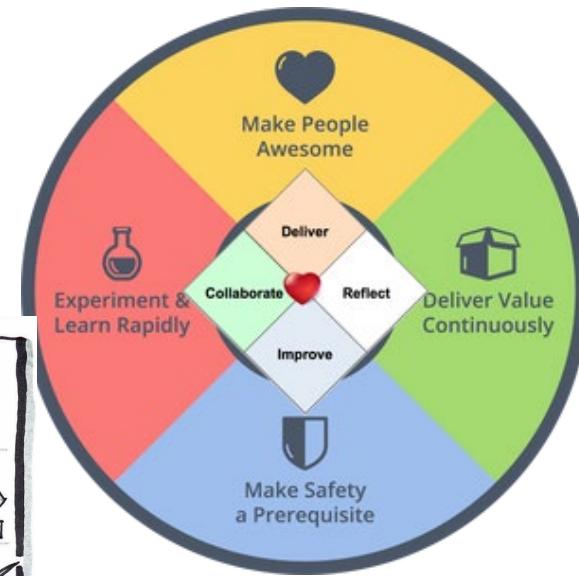
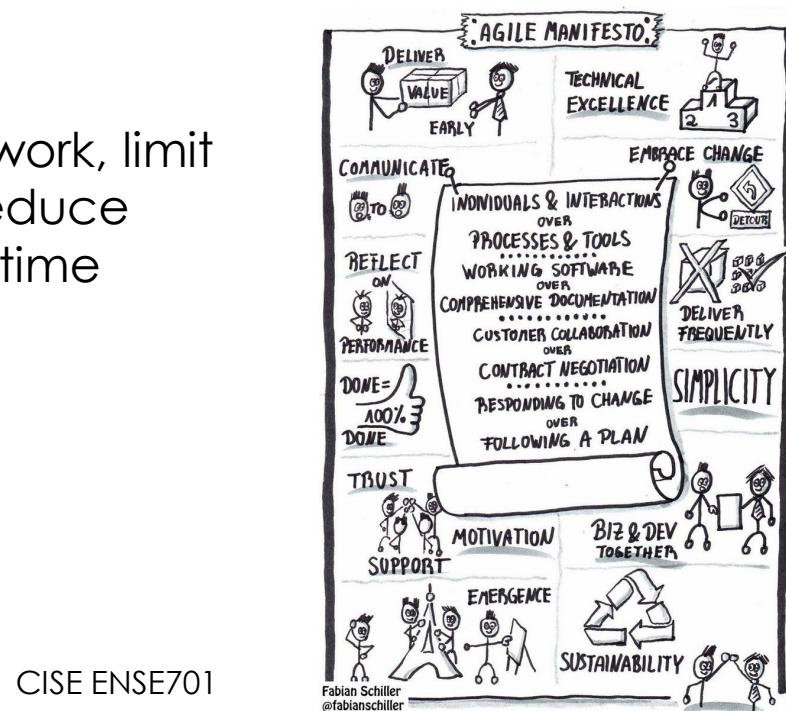
Continuous = frequent & small

Modern Agile – 4 key values

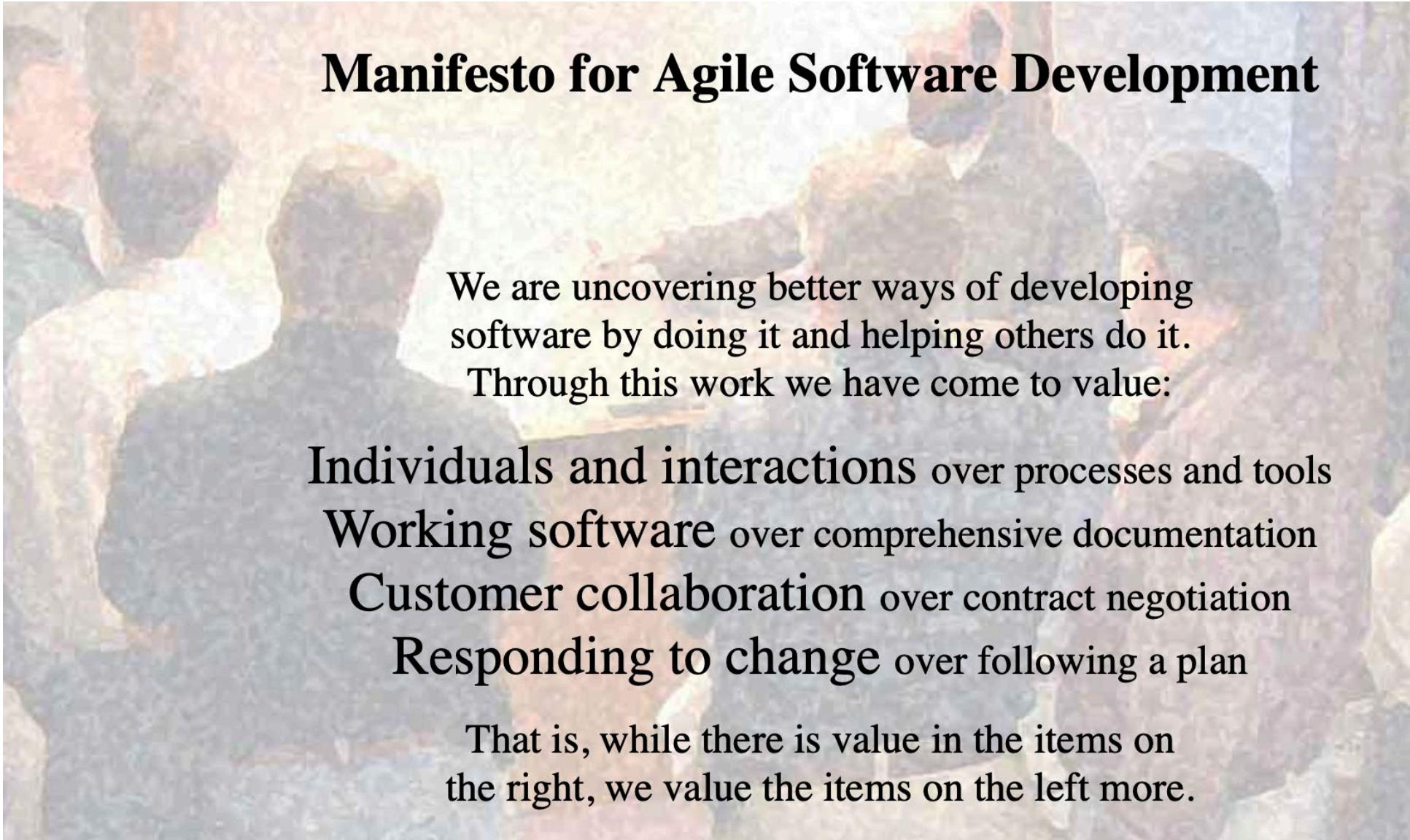
Heart of Agile – 4 key actions

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

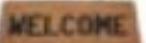
That is, while there is value in the items on the right, we value the items on the left more.

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

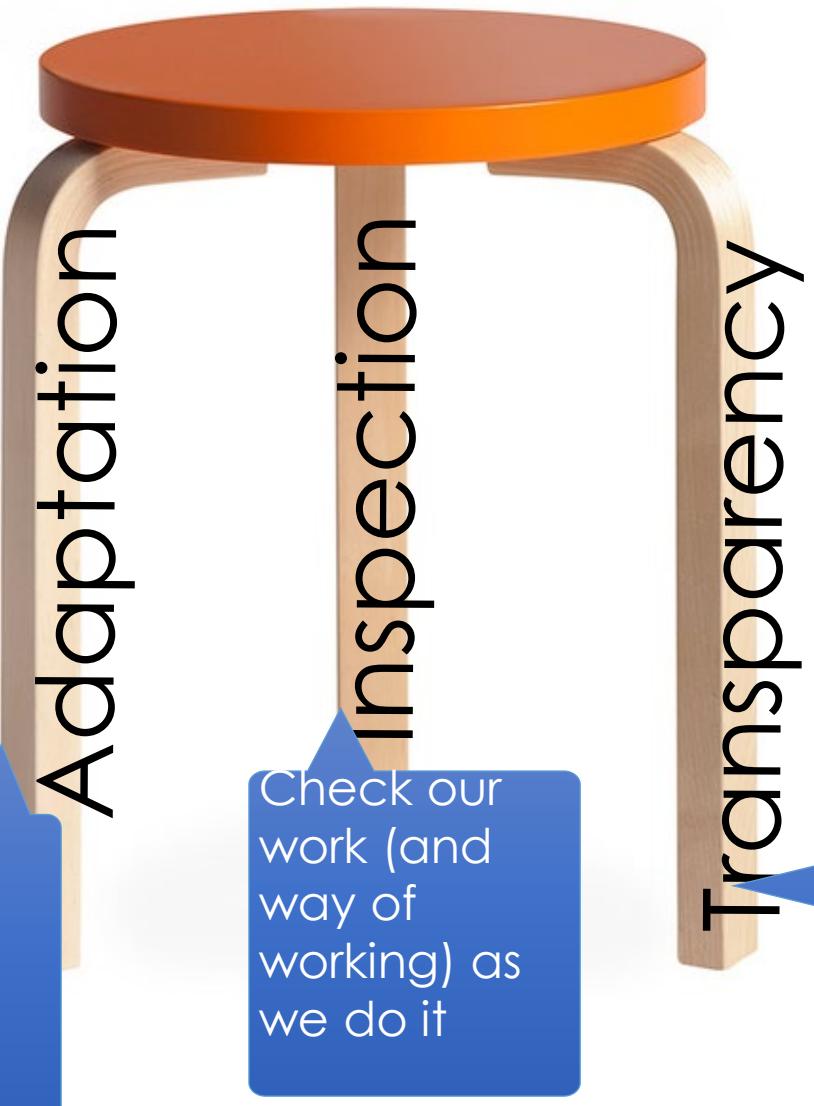
11



Teams regularly reflect and adjust behaviour

12

The 3 foundational ideas of Agile

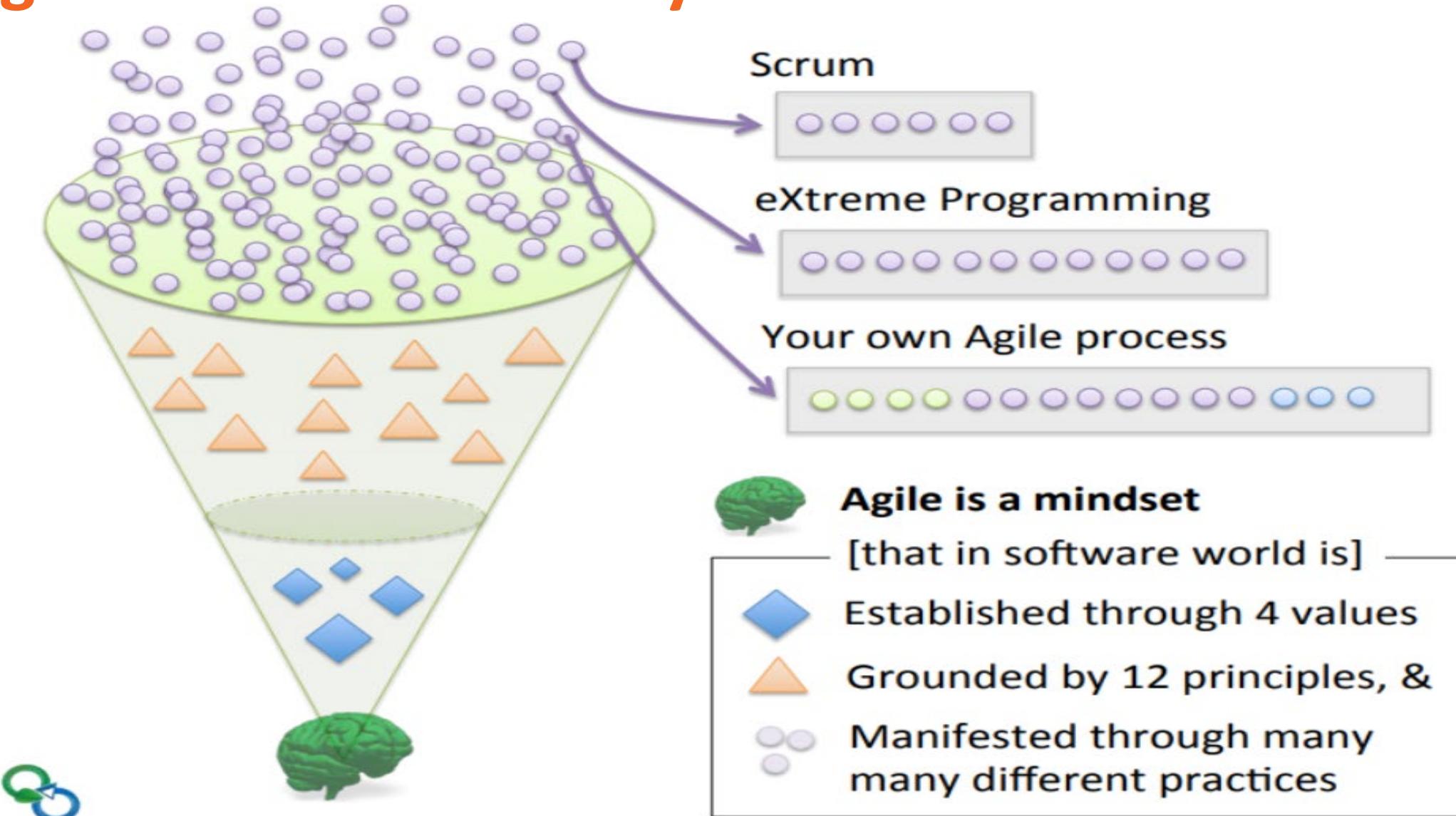


- Adaptation
 - willingness to change if it makes sense
 - understand implications of change and adapt
 - willingness to experiment to check alternatives
 - Desire to learn and improve

- 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

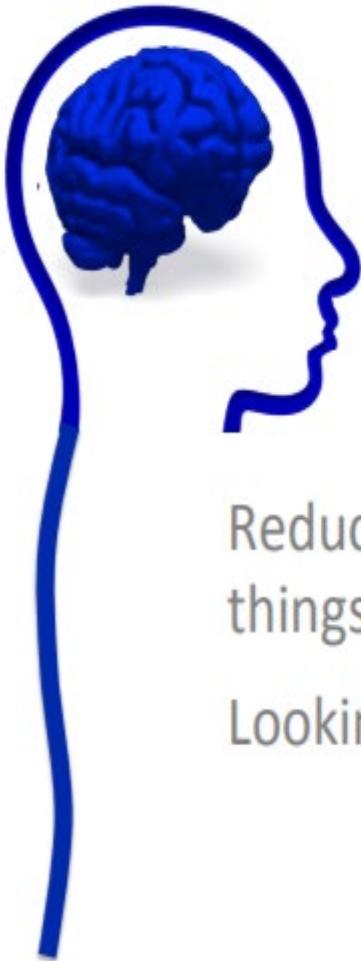
Transparency
Workflows and progress are very visible to everyone
Decisions are collaborative where it makes sense
Different points of view are sought proactively
Honesty and openness are normal (tempered with empathy)
The truth is visible
Mistakes and questions are ok

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)

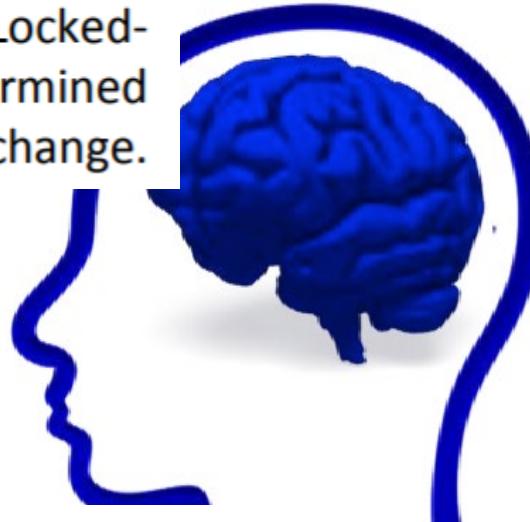


Discover and learn through valuable output and welcoming change (Circular Thinking)

The Growth mindset and the Agile Mindset

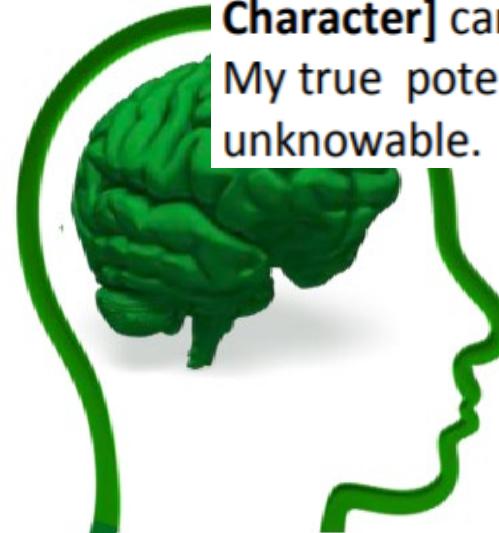
Carol Dweck

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

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

144 LOs in Scrum training

Oversimplified:

SCRUM Certification - \$29 - Cheap Certification, Free Boo

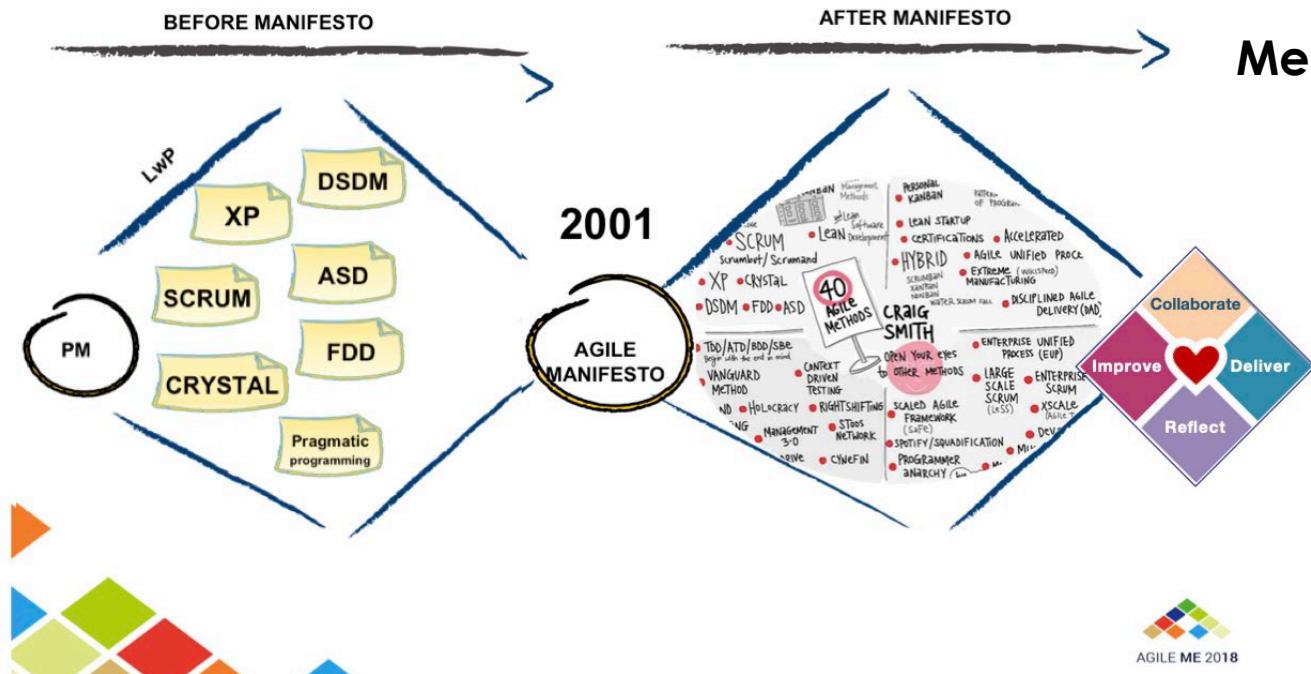
Ad www.scrum-institute.org/ ▾

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

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

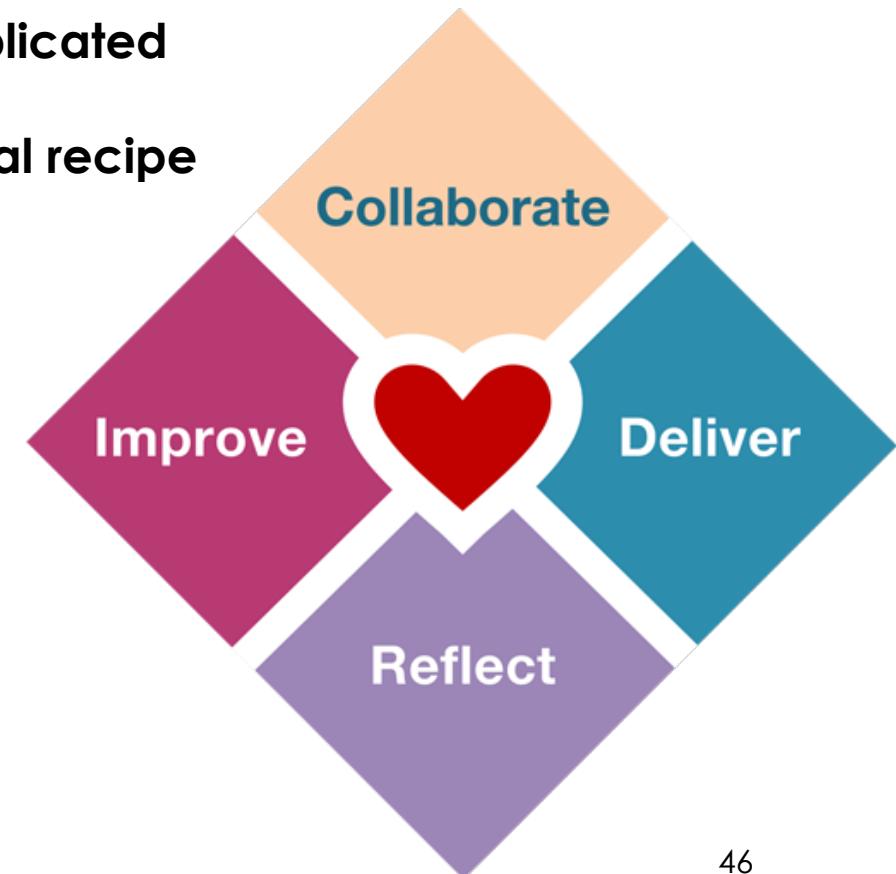
Highlights: Online Scrum Training Materials, Multiple-Choice Test Questions...

Pierre Hervouet's recap of history:



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

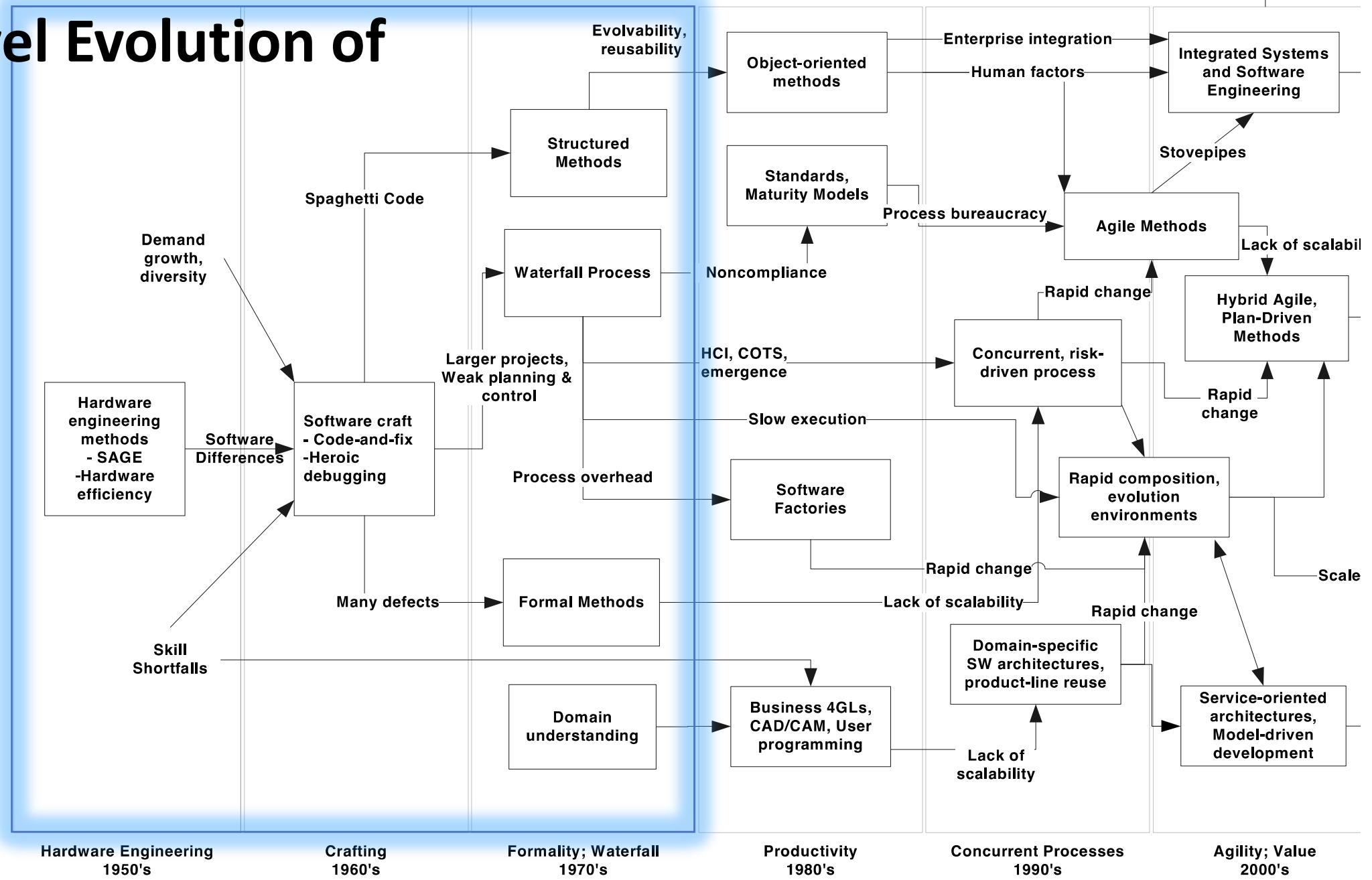
“The Criteria for Determining Whether a Team is Agile”

Scott Ambler

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

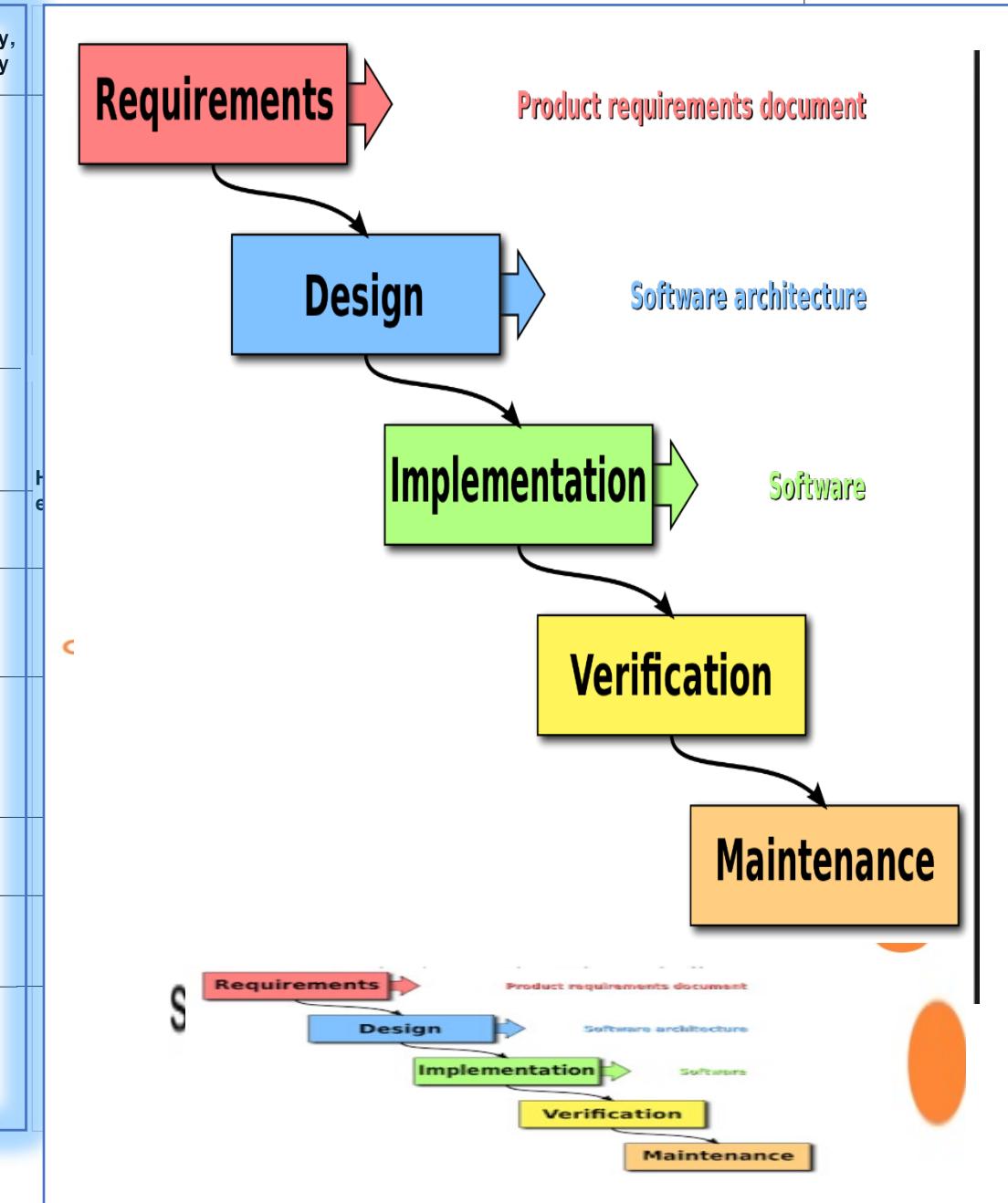
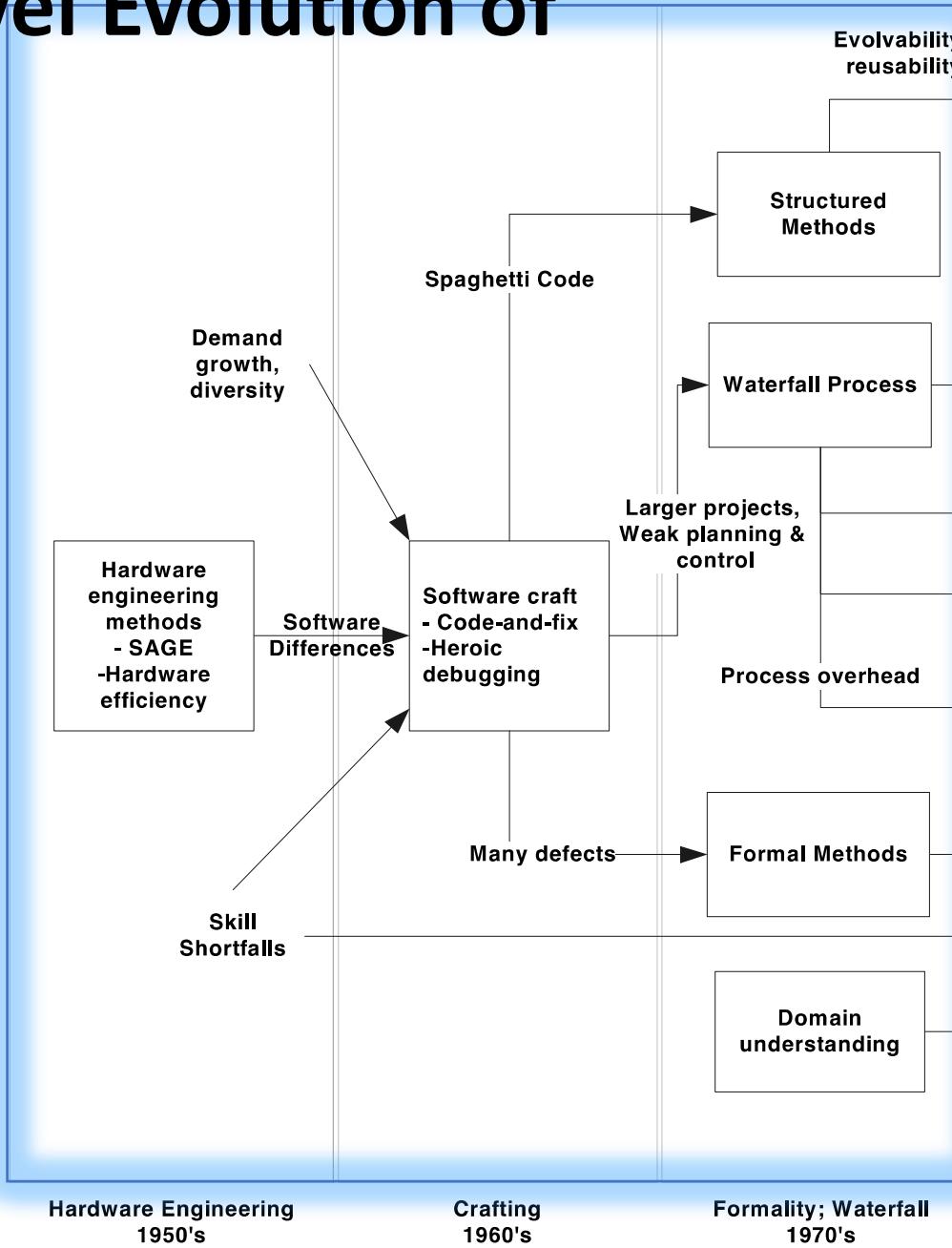
<https://agilemodeling.com/essays/introductiontoam.htm>

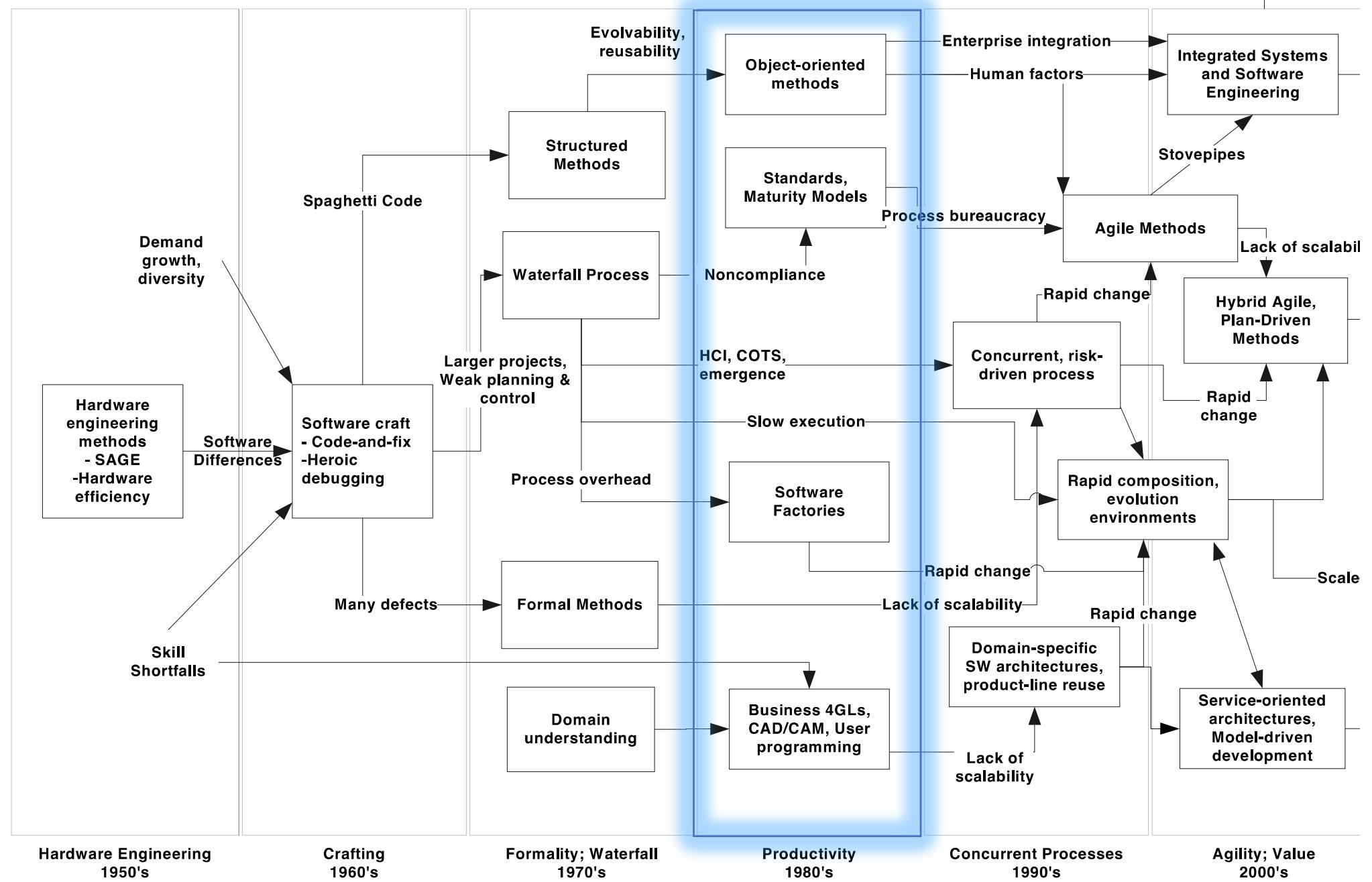
High-level Evolution of SE

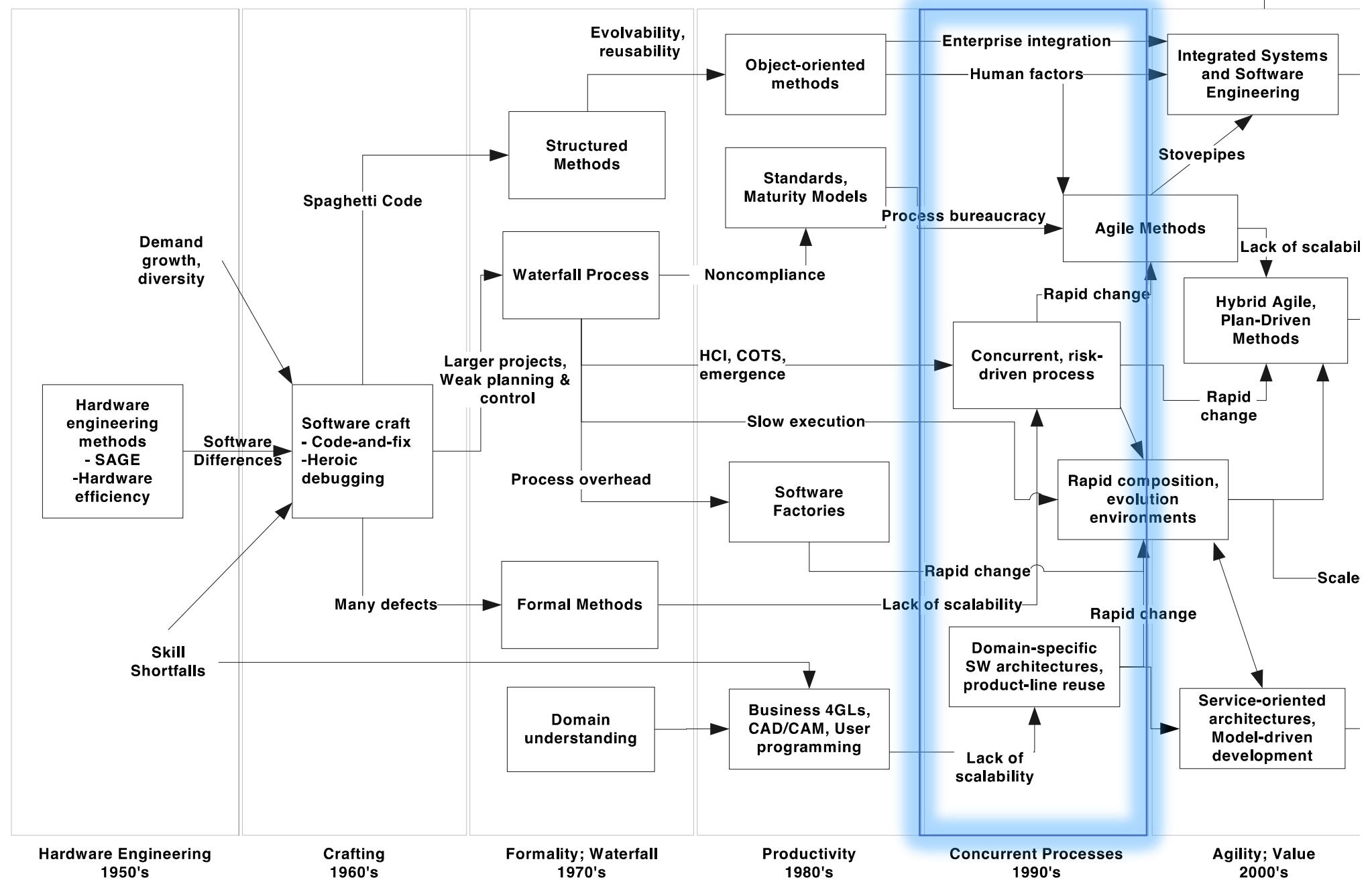


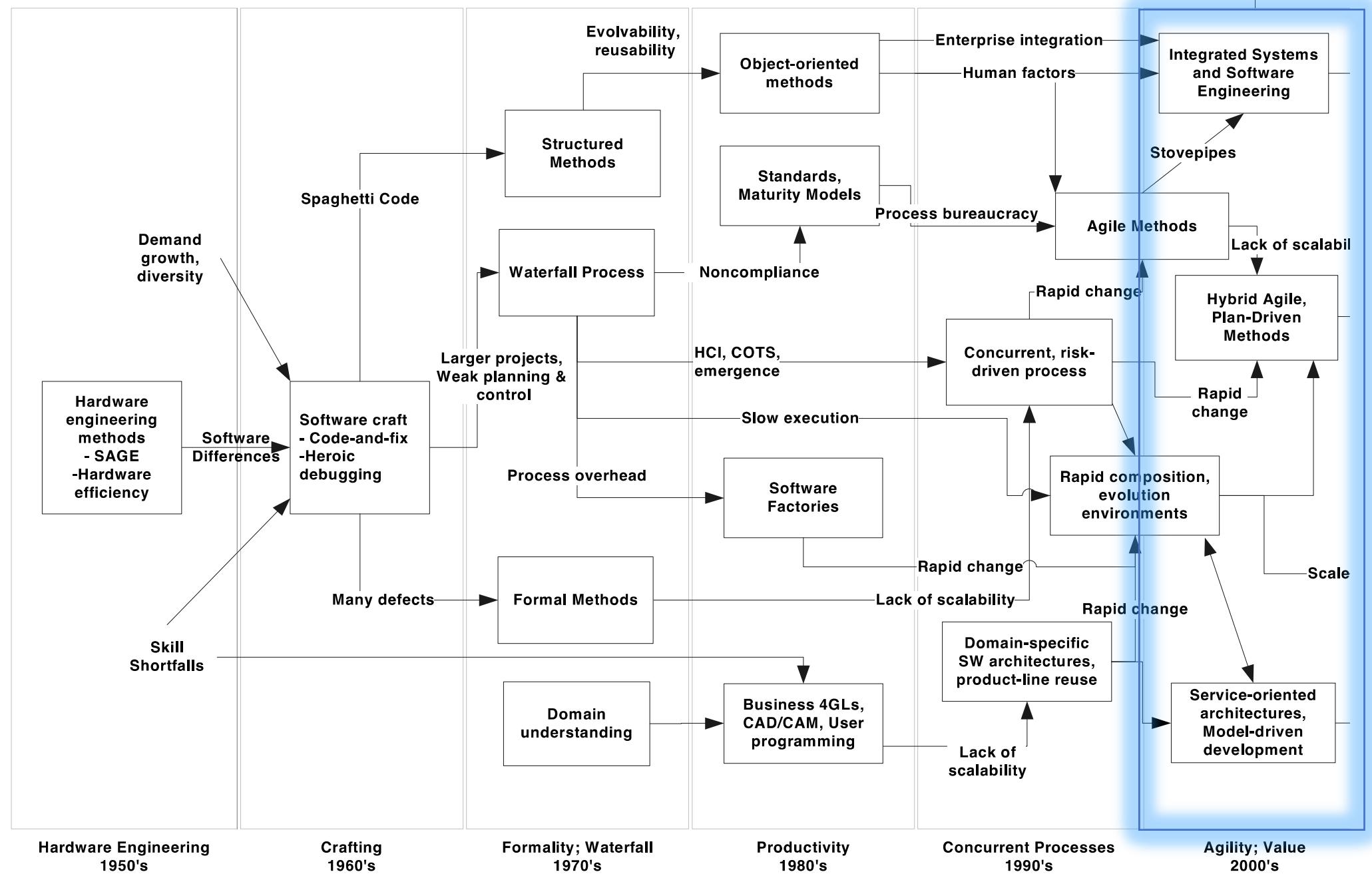
Boehm, B. (2006) "A view of 20th and 21st century software engineering," presented at the Proceeding of the 28th international conference on Software engineering - ICSE '06, New York, New York, USA, p. 12.

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



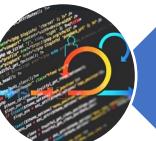
CI/CD focus on automating the integration and then deployment (release)
So small features can be released frequently



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...

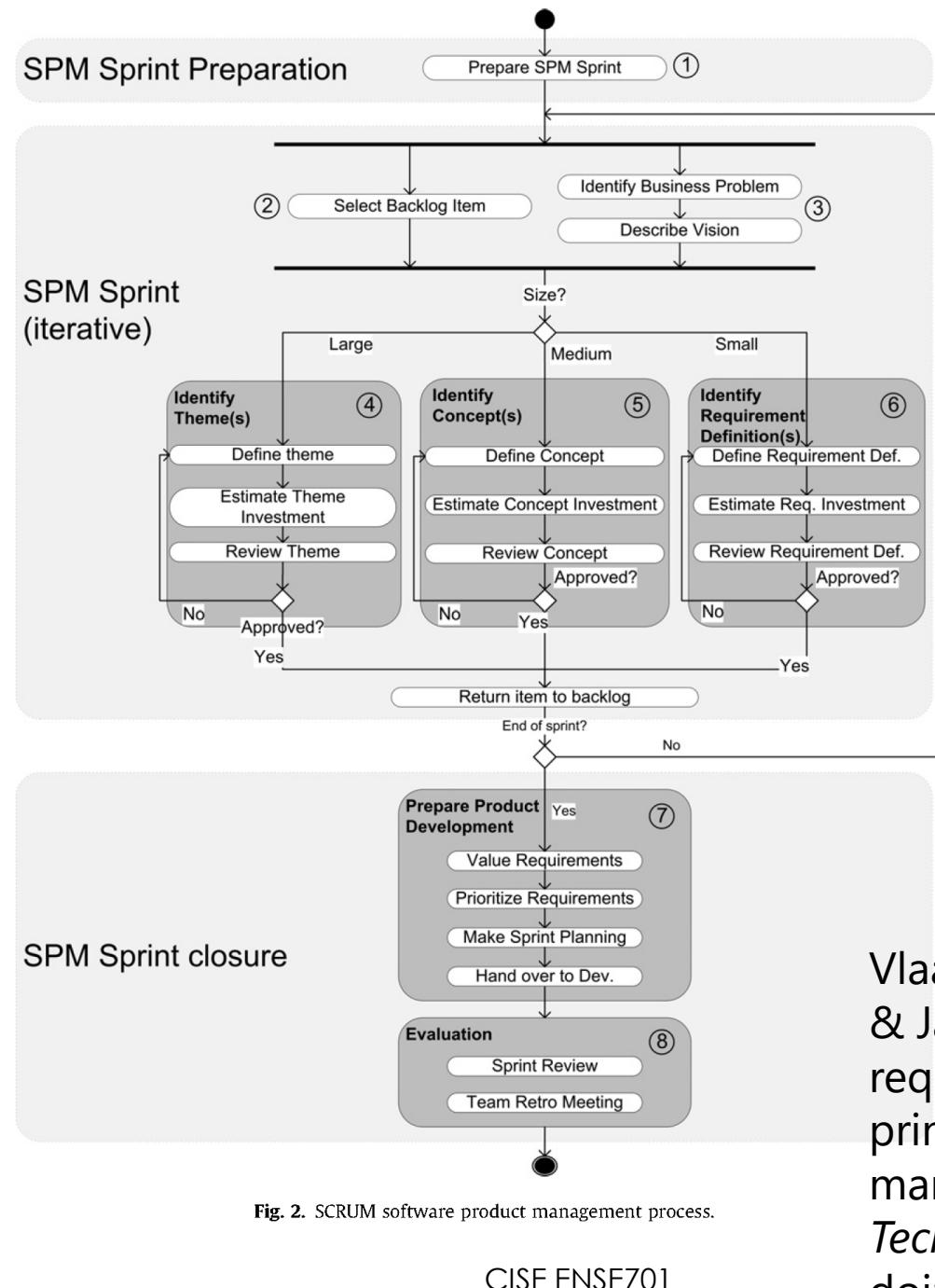


User Stories & Story Maps

Week 3

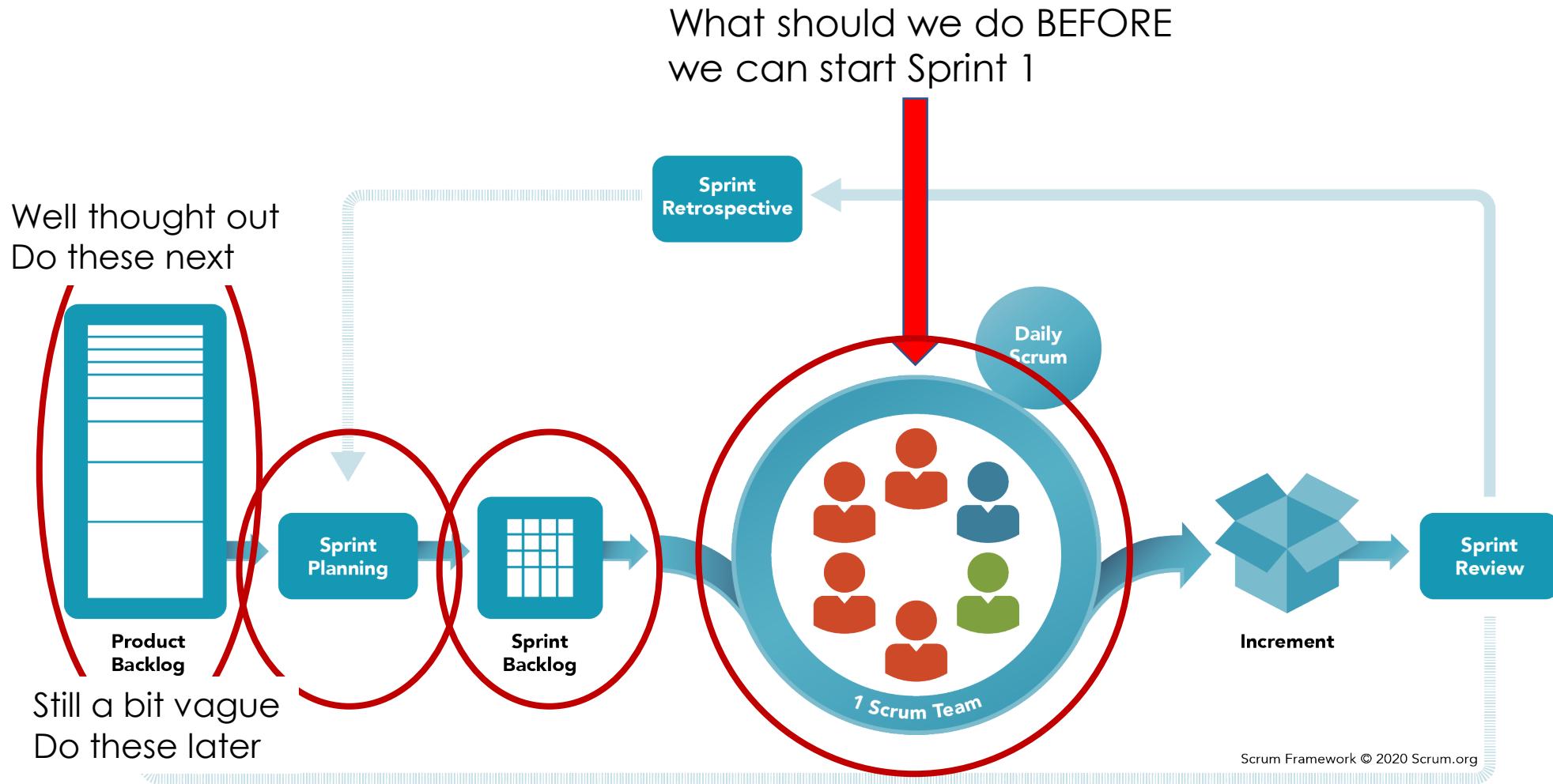


Software Product management Process



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

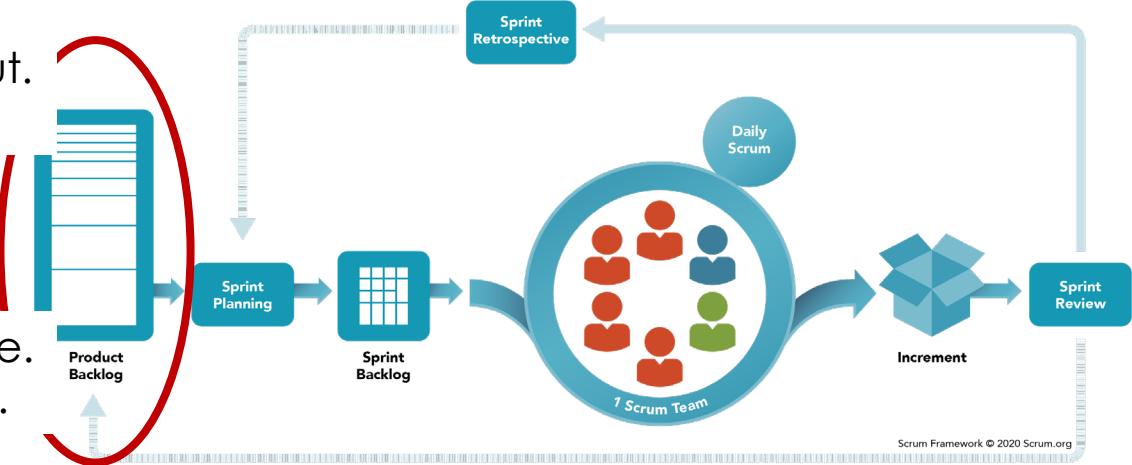
What is the goal (desired outcome) of this new user capability?

Problem Space

Well thought out.
Do these next.

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**

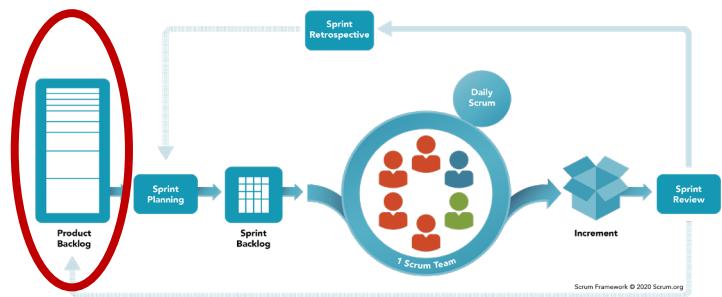
Placeholders for conversations.

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.

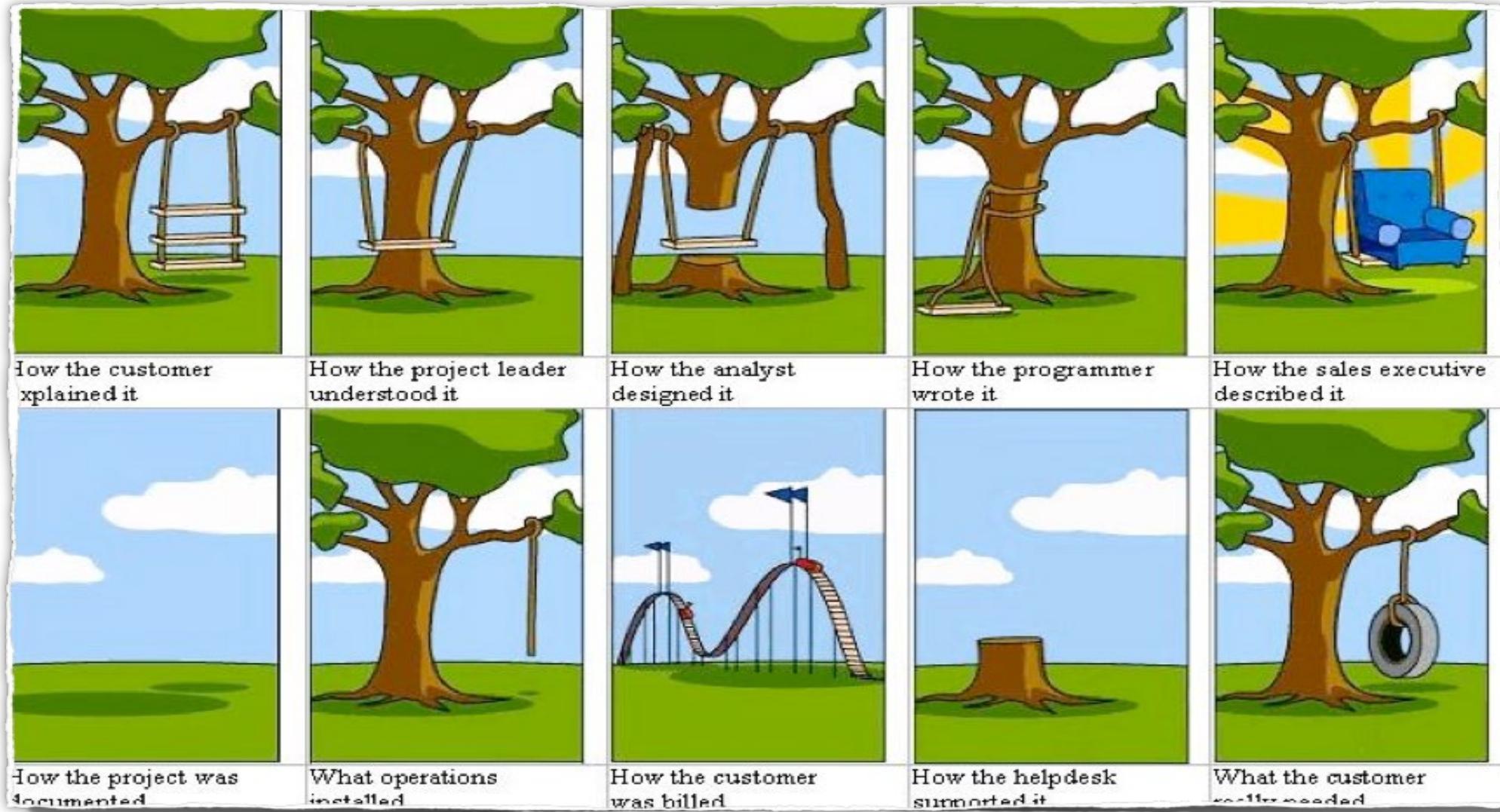
There are many perspectives on the problem and user needs

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

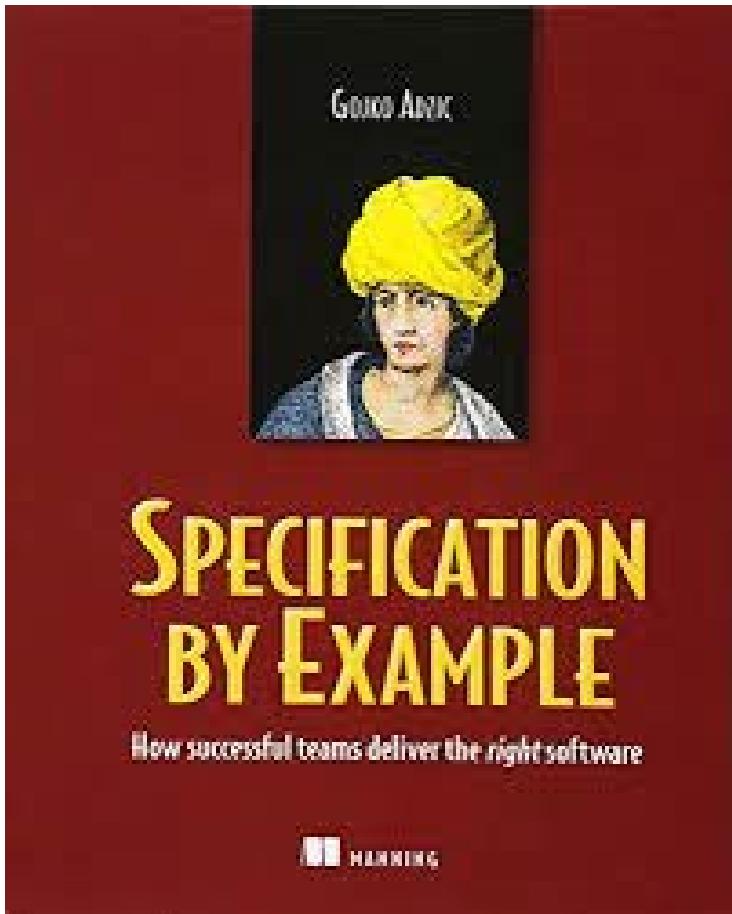


<https://www.userinterviews.com/blog/how-to-write-better-user-stories-using-ux-research>

USER INTERVIEWS

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

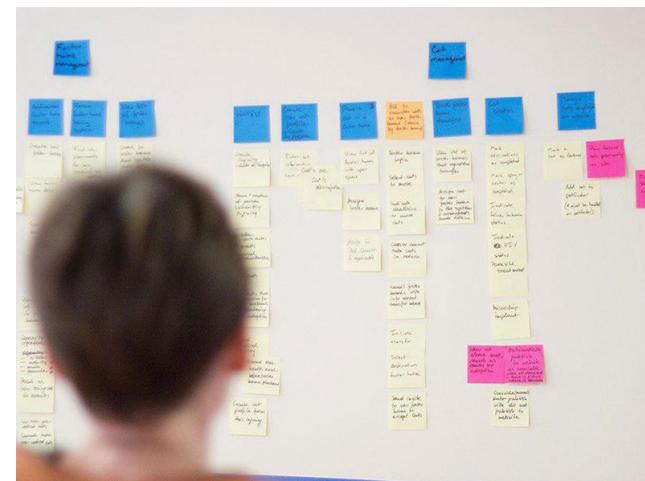
Observation, surveys, impact mapping, customer journey mapping.....

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

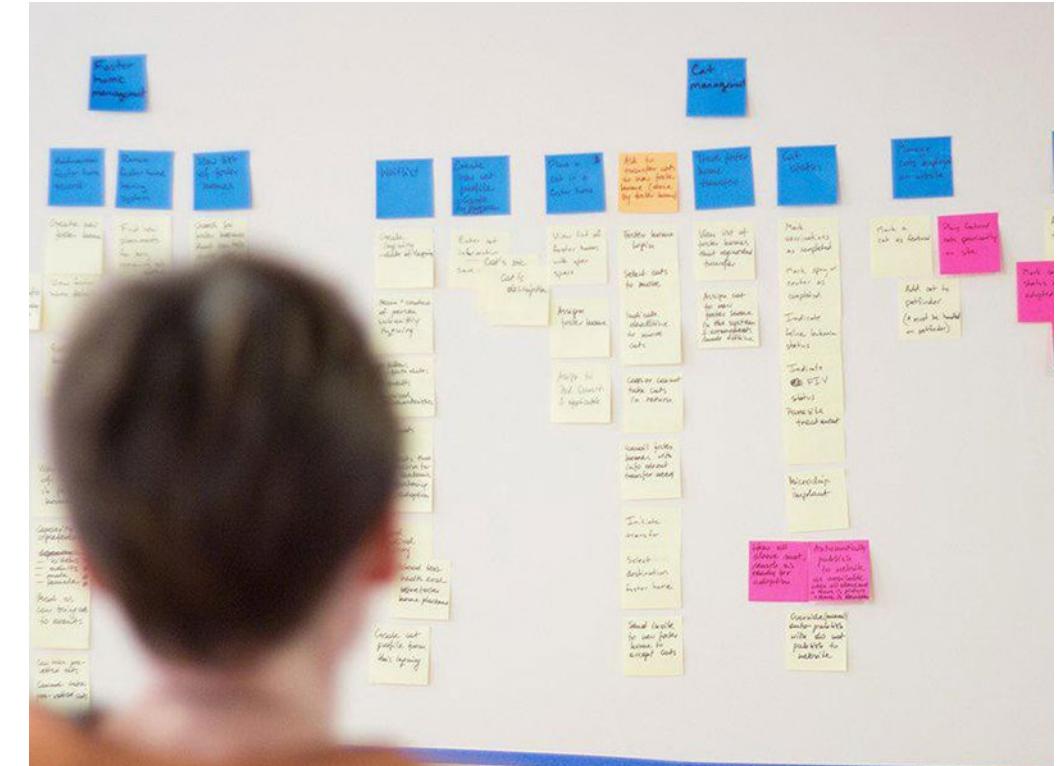
Story Maps are the new Product Backlog

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



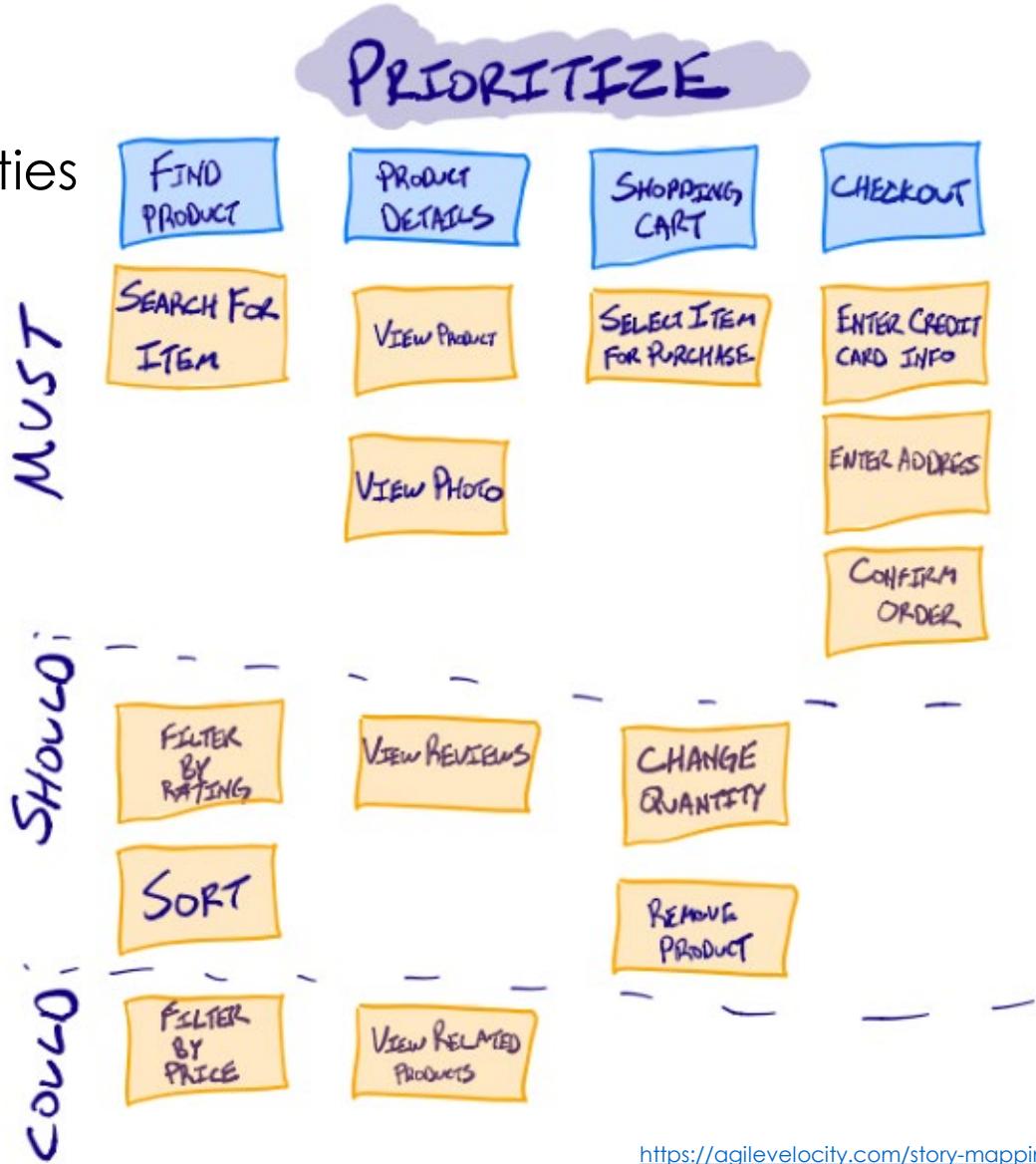
Jeff Patton & Associates, jeff@jpattonassociates.com, [twitter@jeffpatton](https://twitter.com/jeffpatton)



80

The anatomy of a Story Map

Columns relate to user activities

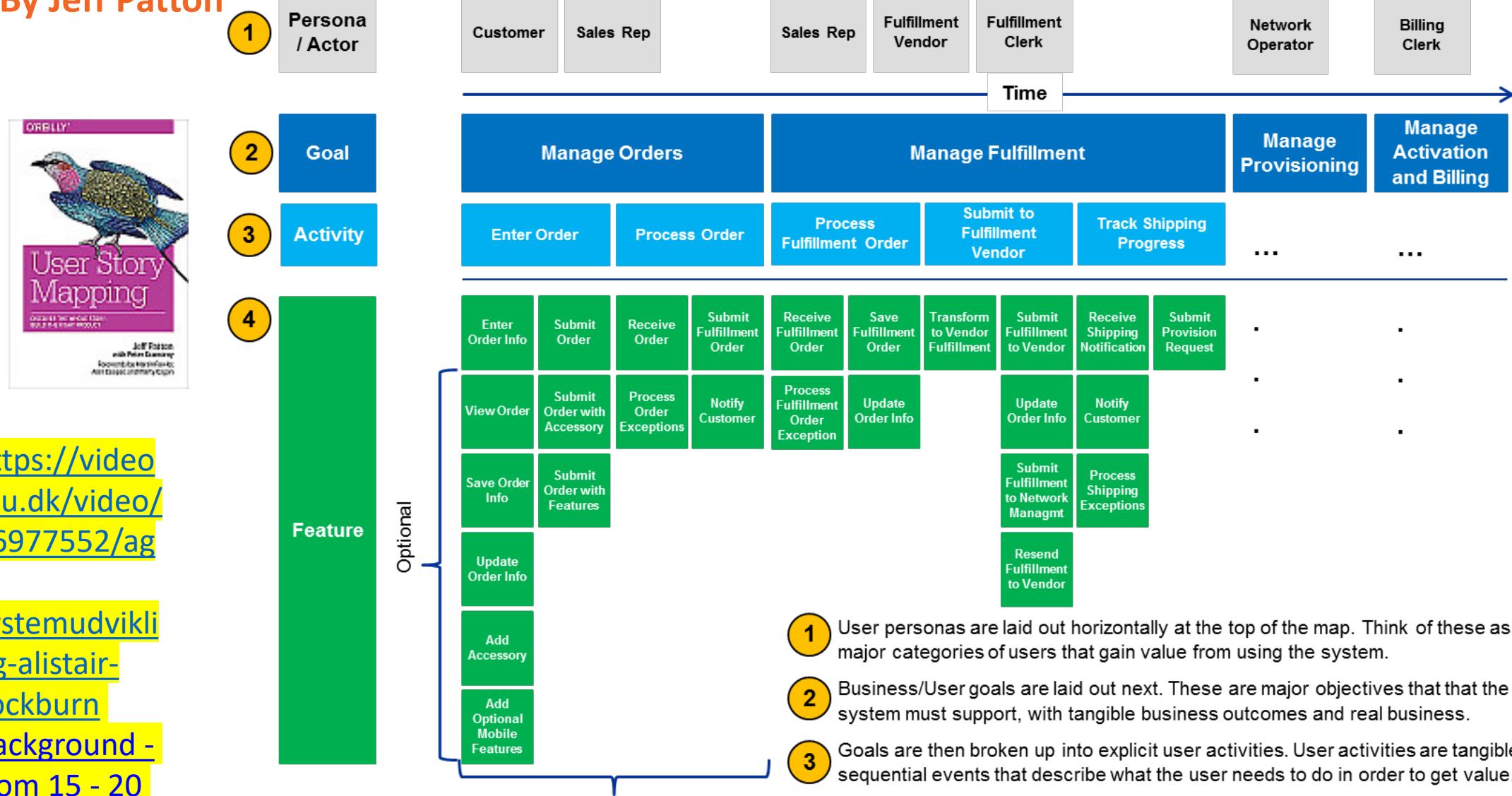


Each column has the system features related to the user activity for that column

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

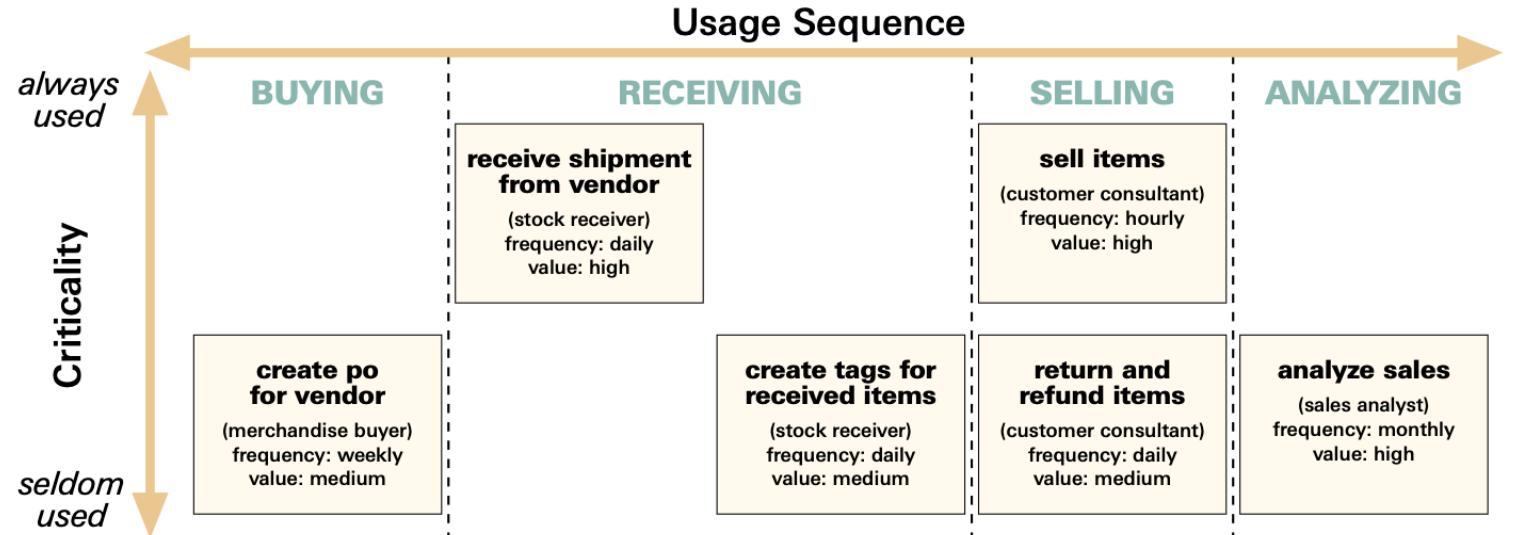
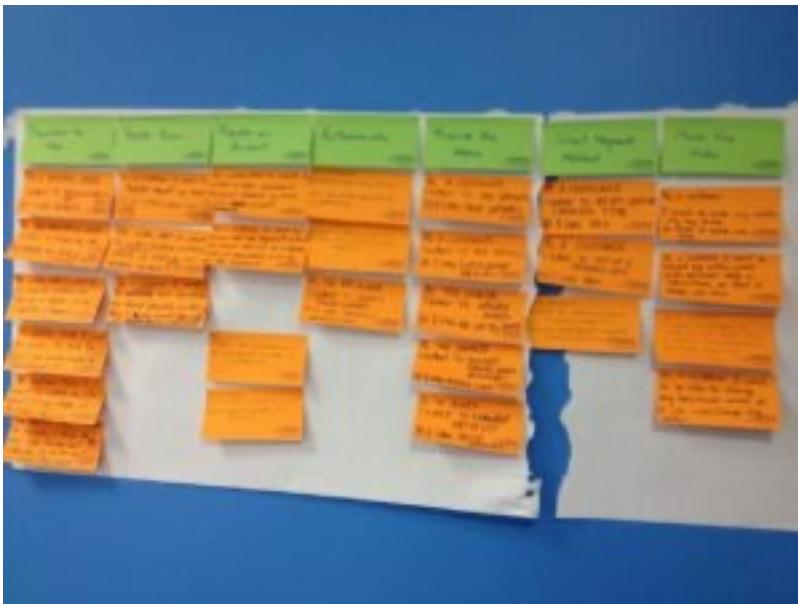


Figure 5: The model is vertically divided into business processes.

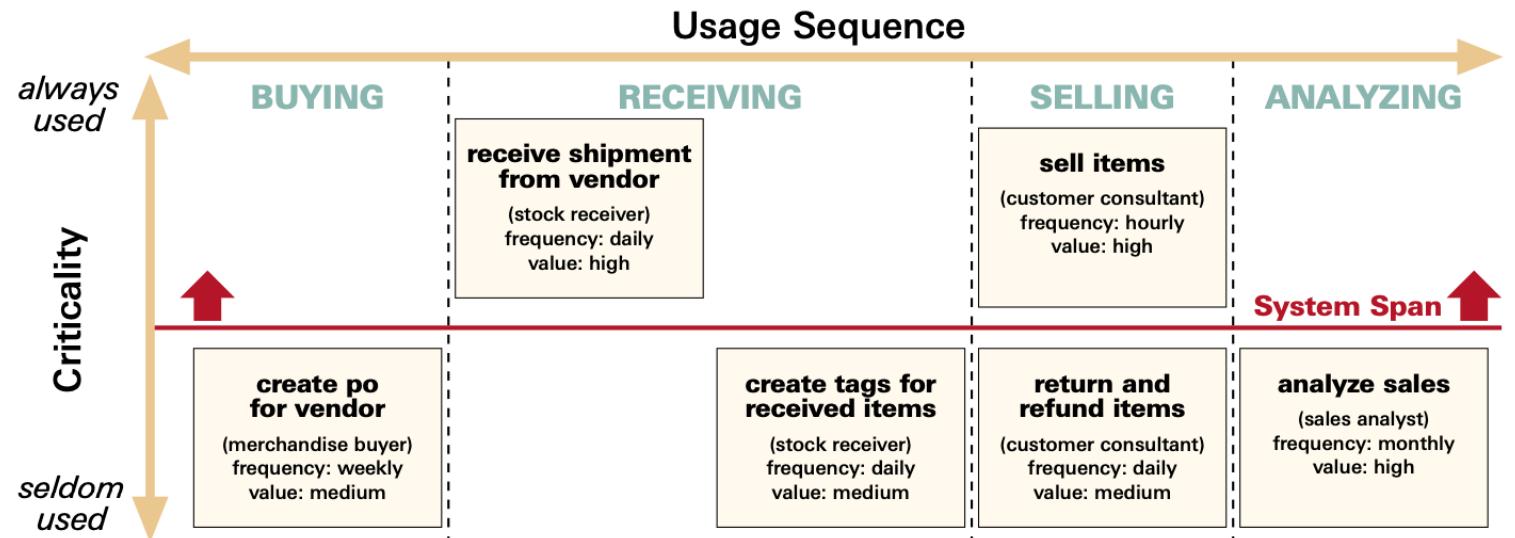
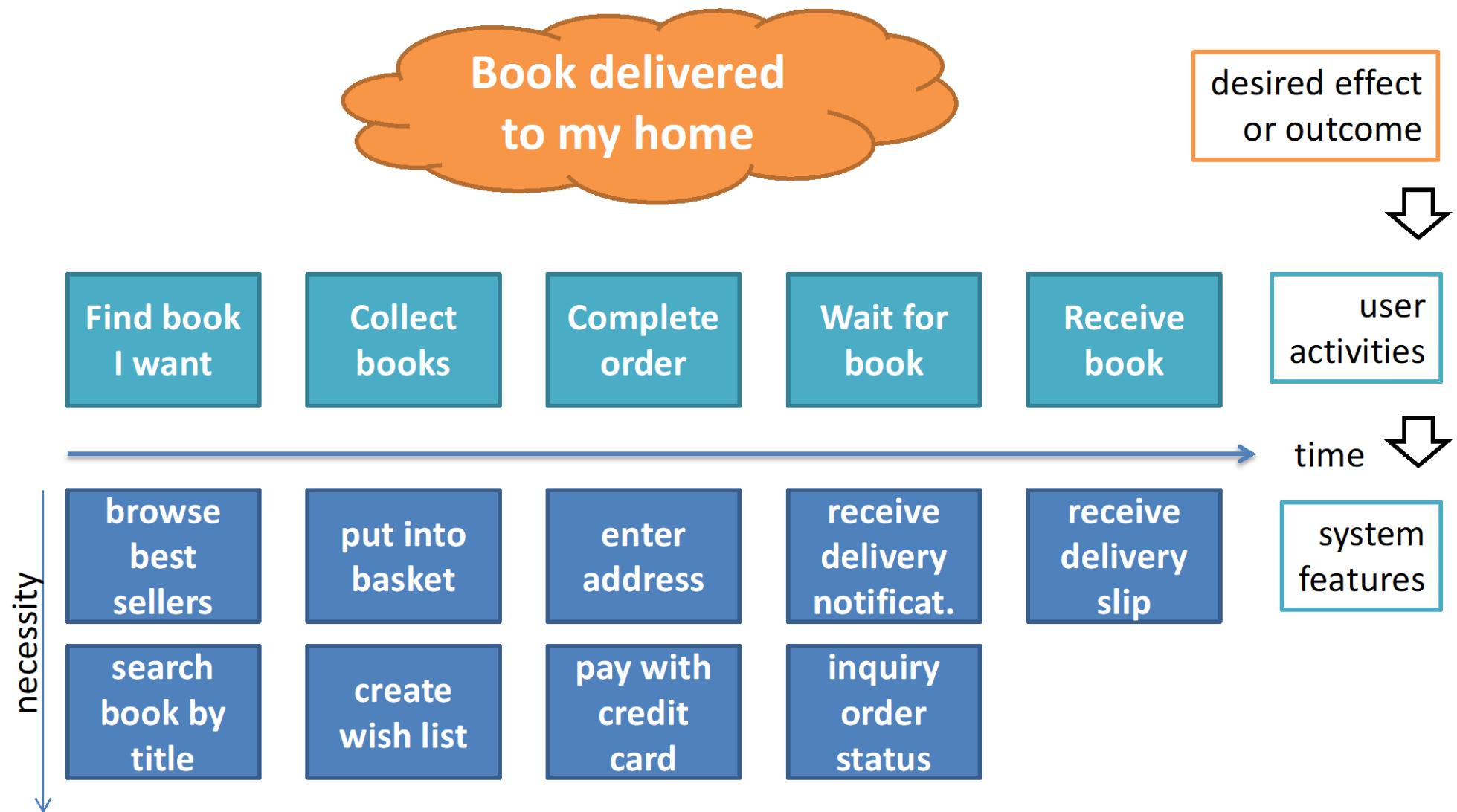


Figure 6: The first system span represents the smallest set of features necessary to be minimally useful in a business context.

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

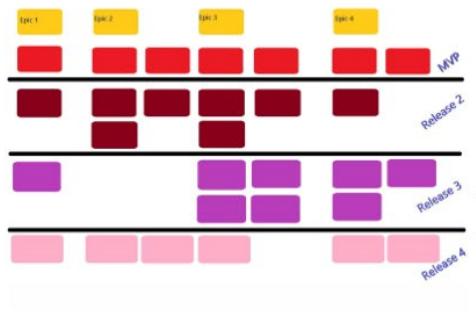
StoriesOnBoard staff pays a lot of attention to collect the latest, most inspiring articles from the internet. Read further to discovery fresh new materials and open the previously published collection of user story mapping content.



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

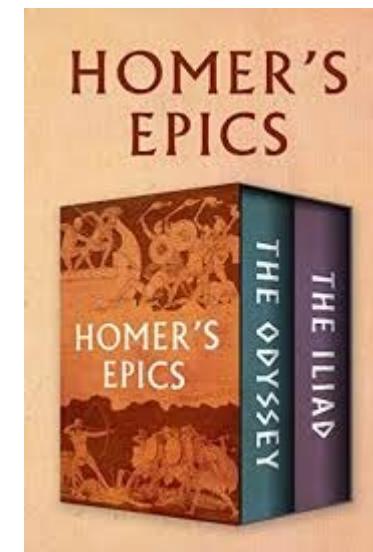
WALK YOUR MAP TO TEST IT – TO GET THE BIG PICTURE

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 ...

Aha! | Blog

PRODUCT USE CASES SERVICES PRICING RESOURCES COMPANY

Blog Support Careers Log in

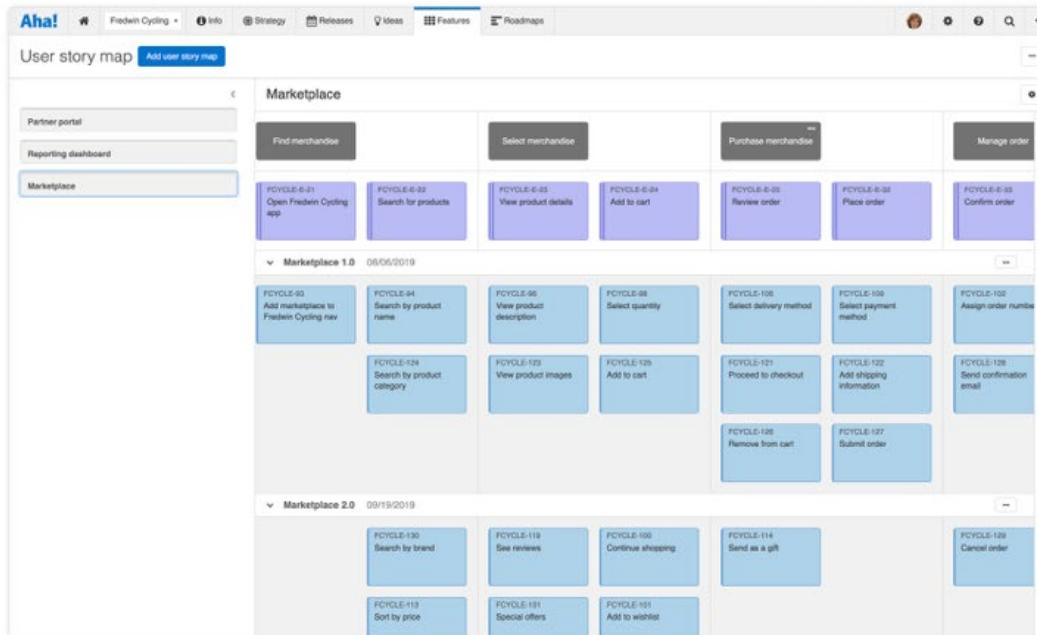
Free trial Join demo

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

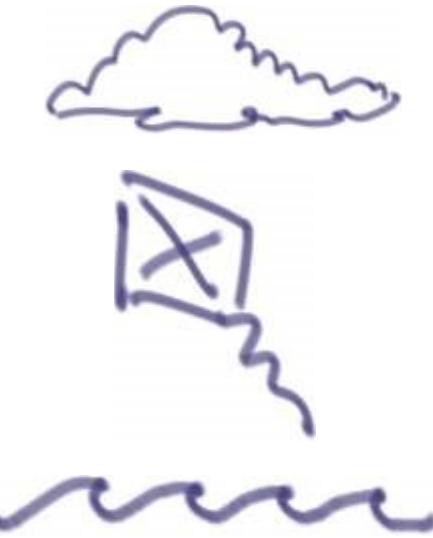
Act as a reminder that dev team promises to have conversations and ask questions

User stories

Point to requirements

Conversations
Diagrams
Prototypes
User interfaces

Be sensitive to your user task's “altitude”



Too abstract

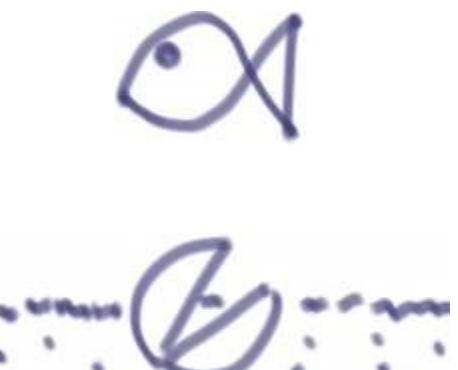
Activity or “Kite level”

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

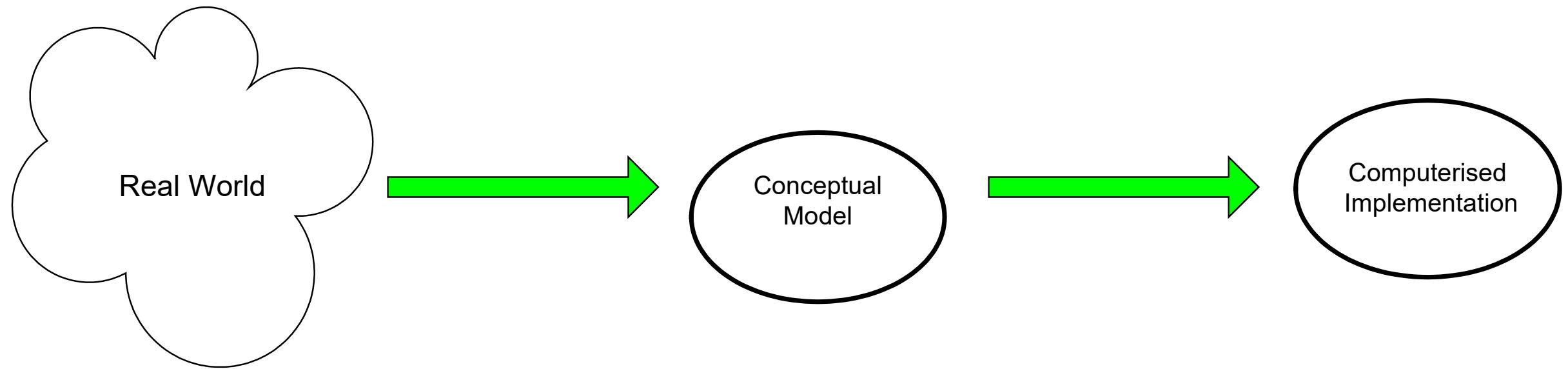
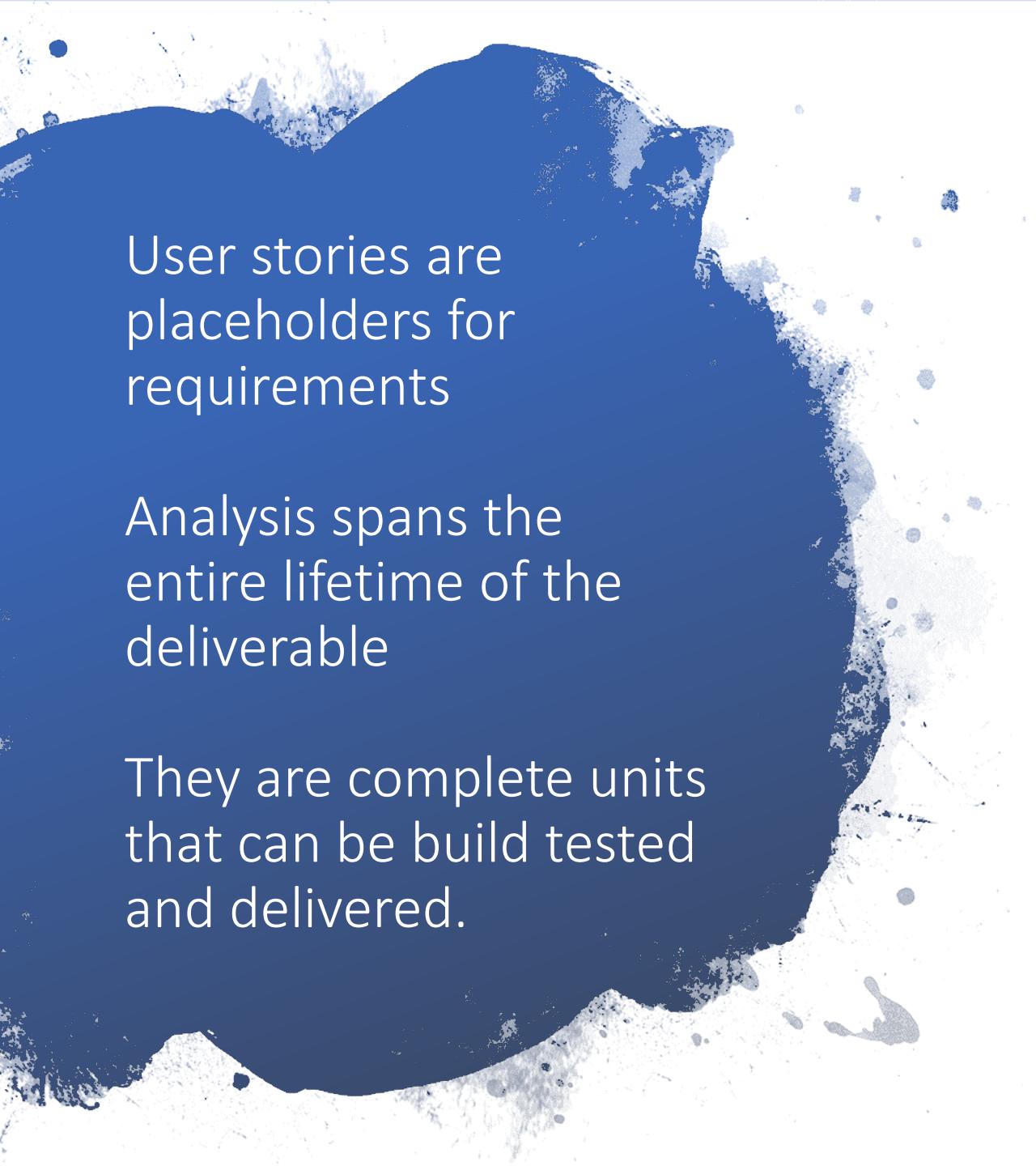


Figure 1: Design as a Mapping Process



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

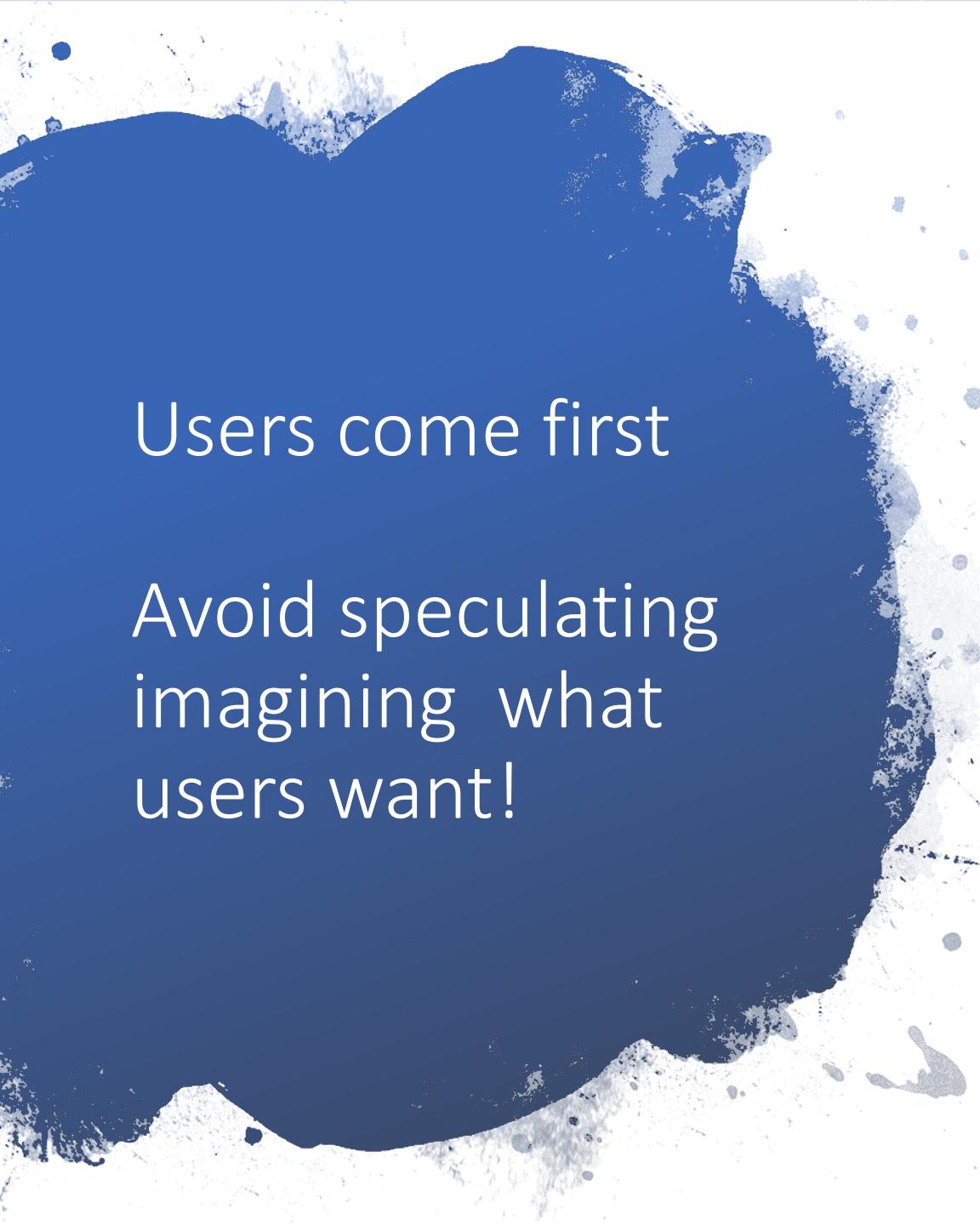
Everyone should be involved in discussions about the user stories

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.

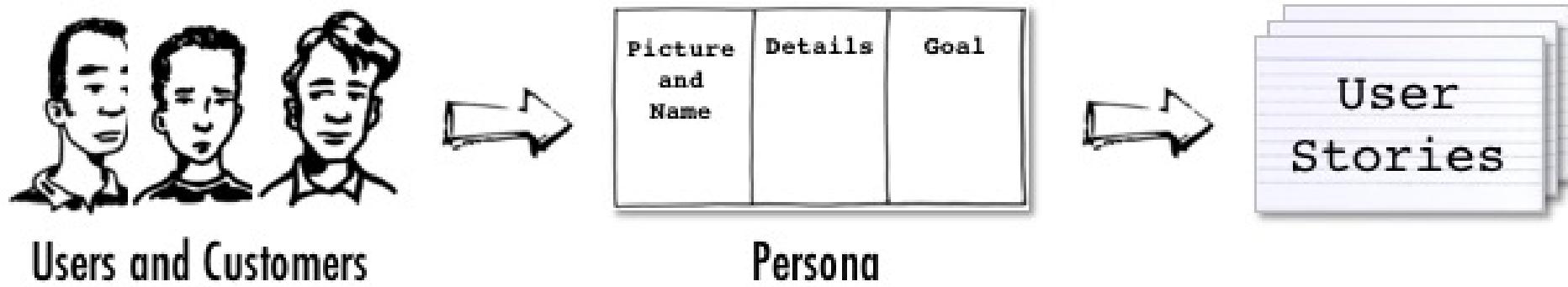
Carry out the necessary user research first, for example, by observing and interviewing users.

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)

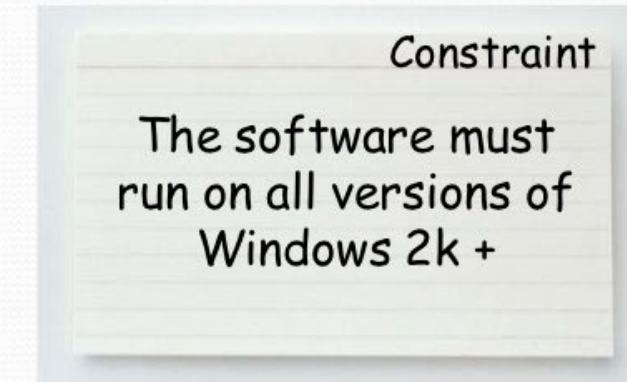
- The product owner's conditions of satisfaction can be added to a story
- These are essentially tests / acceptance criteria

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?

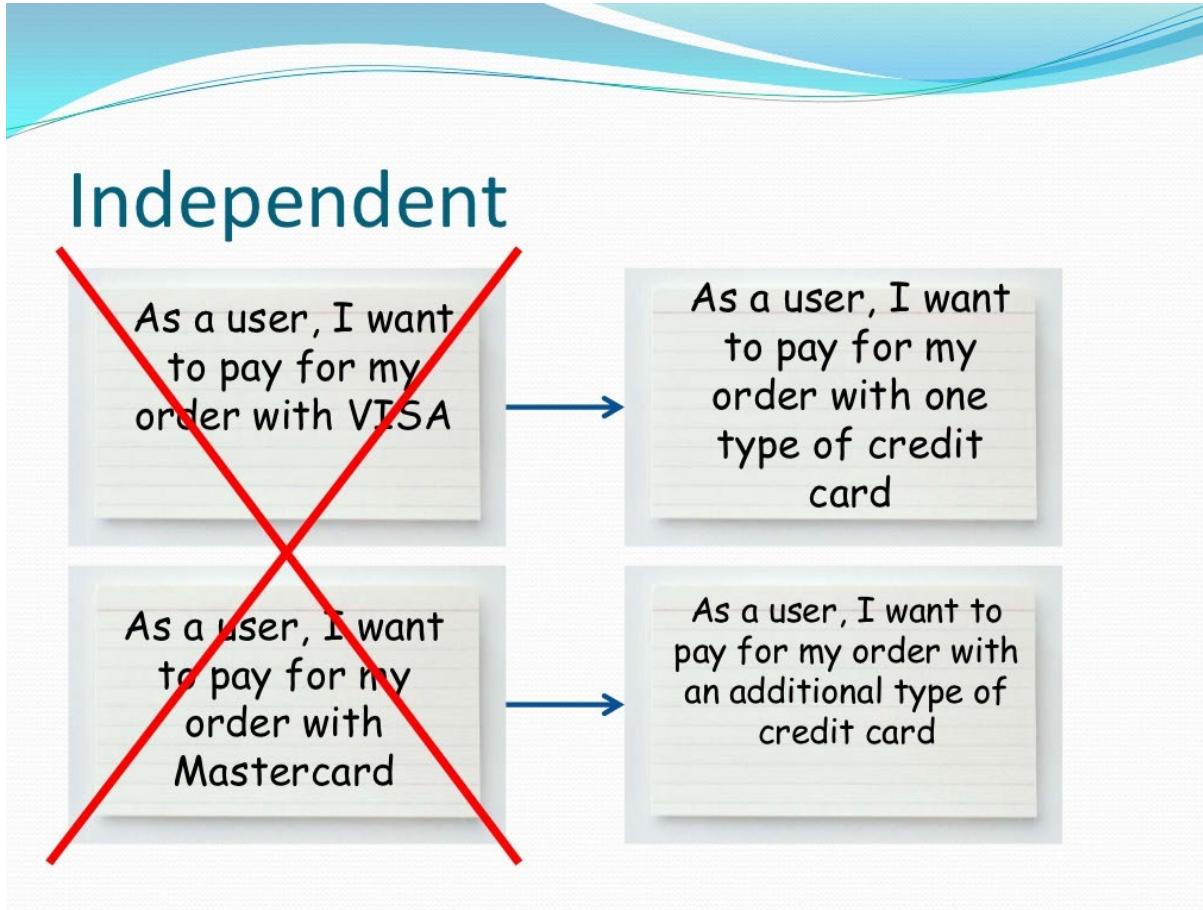
Valuable — Does this story have value to the end user?

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

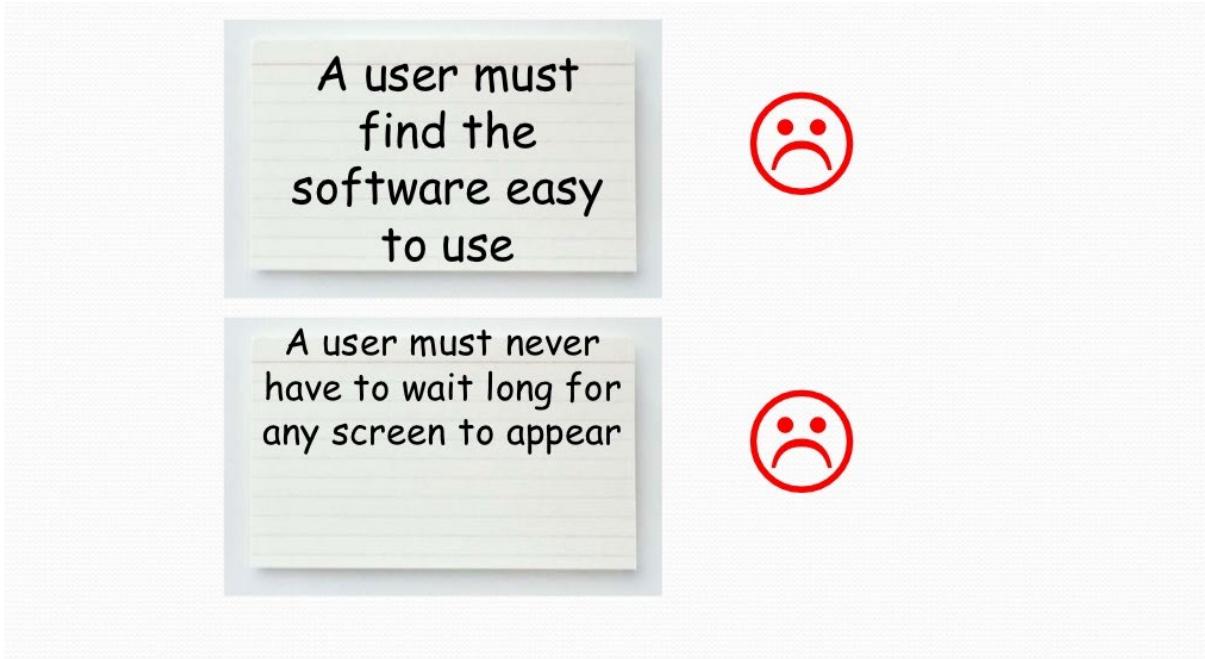
A job seeker can modify a CV already uploaded

- 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





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

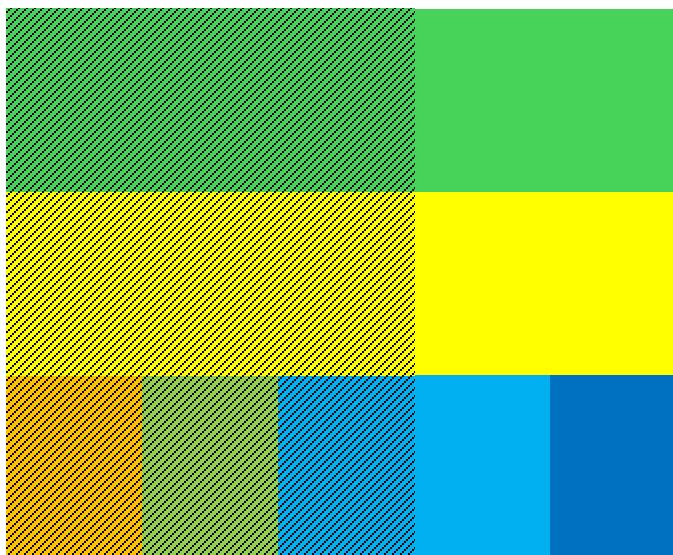
No

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

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

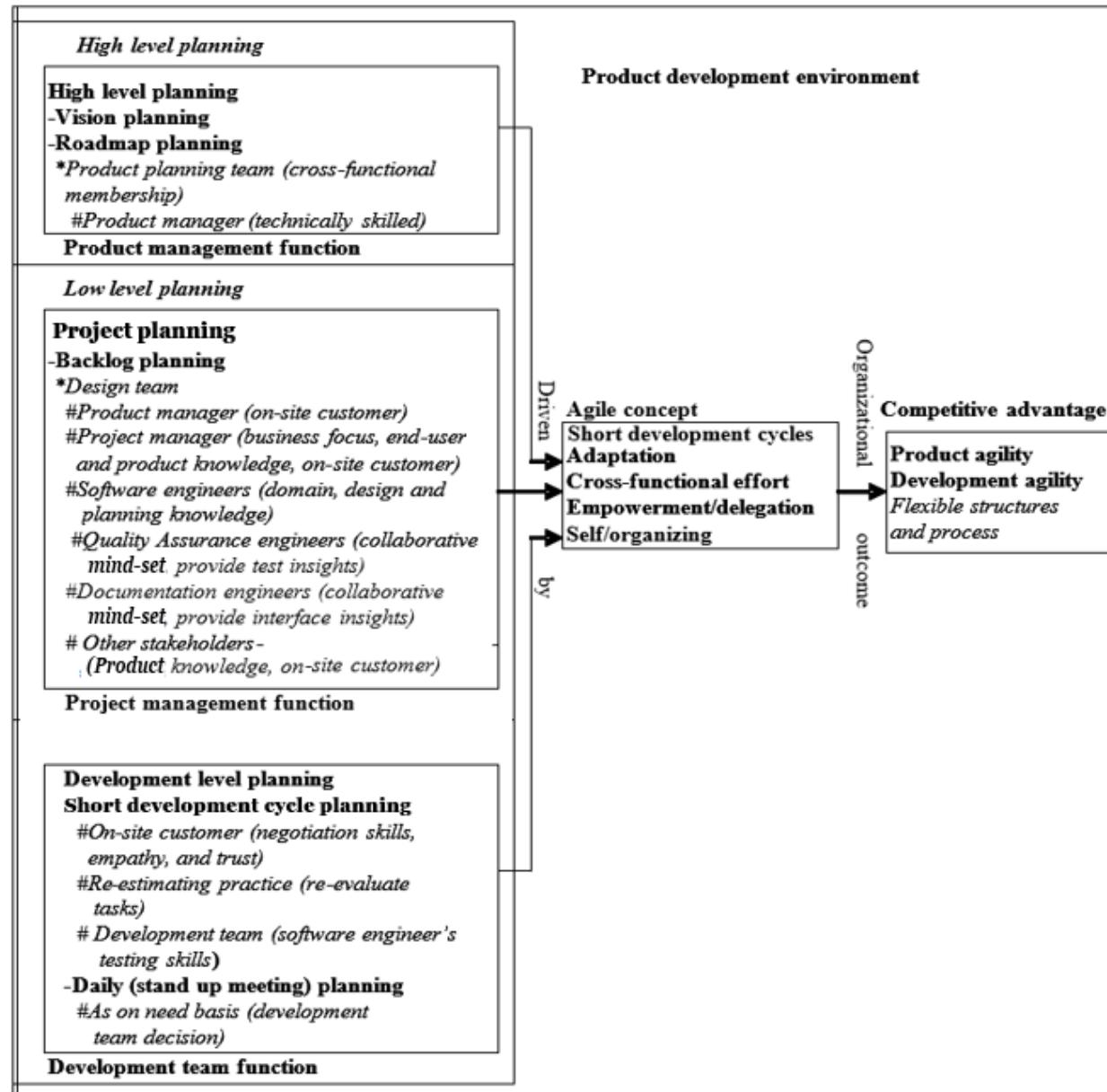
Ass 2 Knowledge Check (30%)

(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.

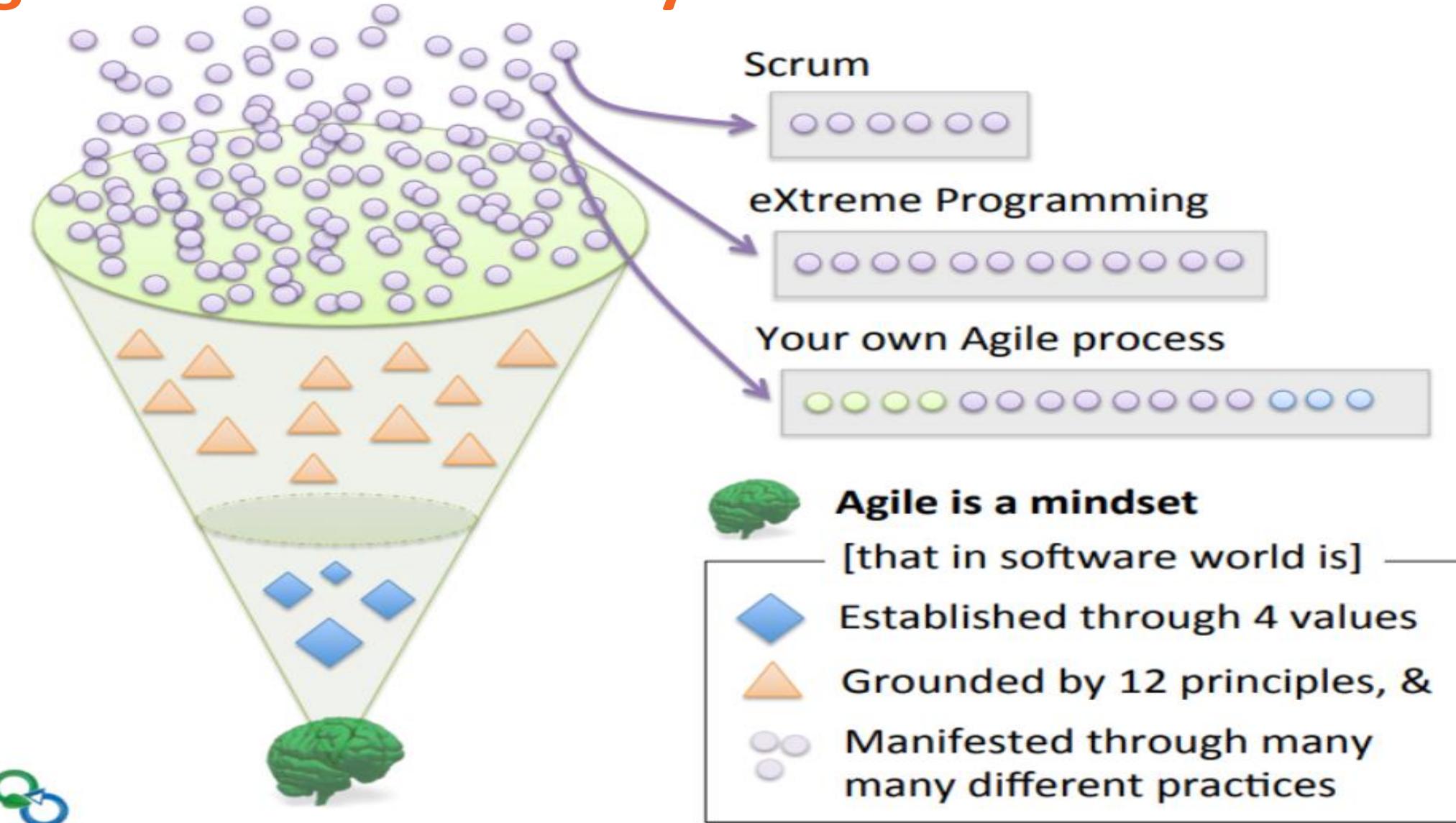
R6. All the above suggests a repeating cycle of behavior to use on projects.

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



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

**CI/CD
PAIR and MOB
TDD**

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?

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?

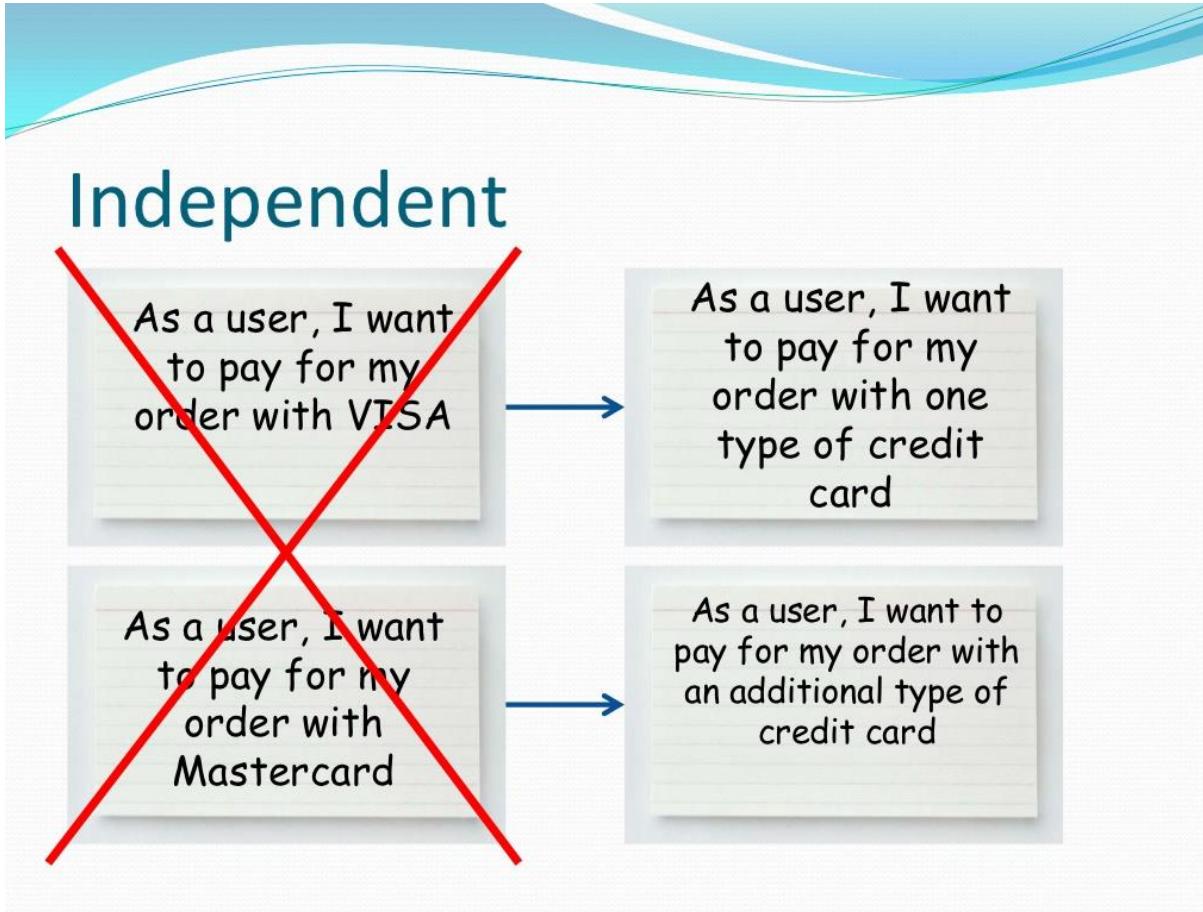
Valuable — Does this story have value to the end user?

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

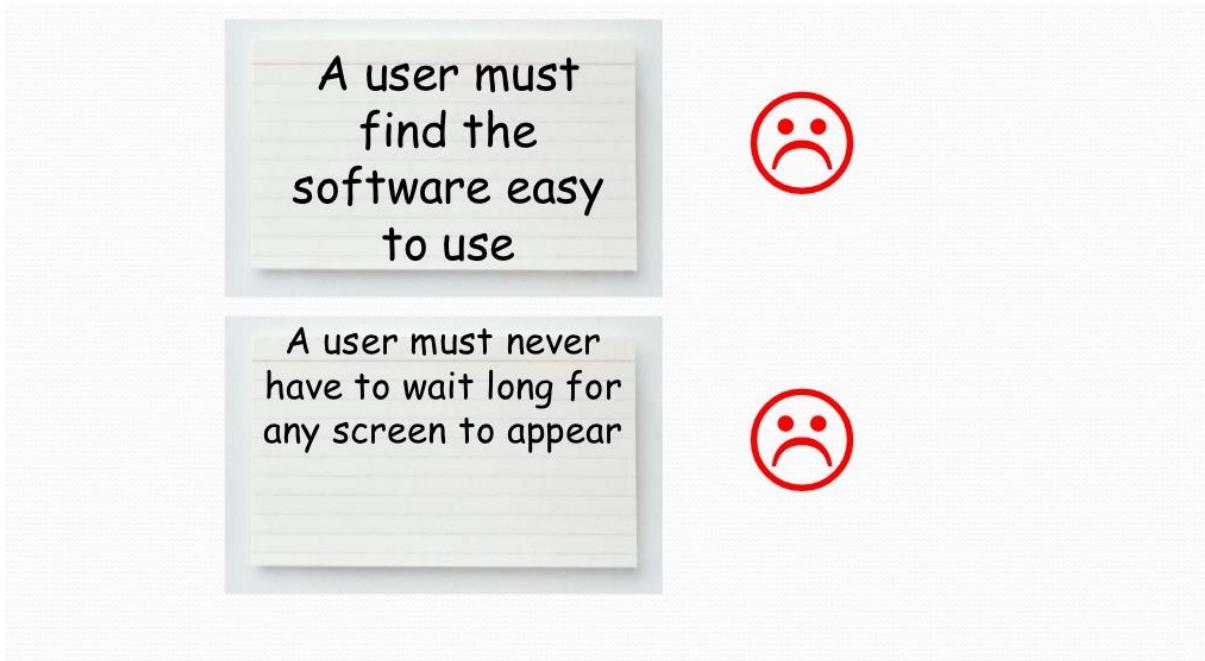
A job seeker can modify a CV already uploaded

- 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



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



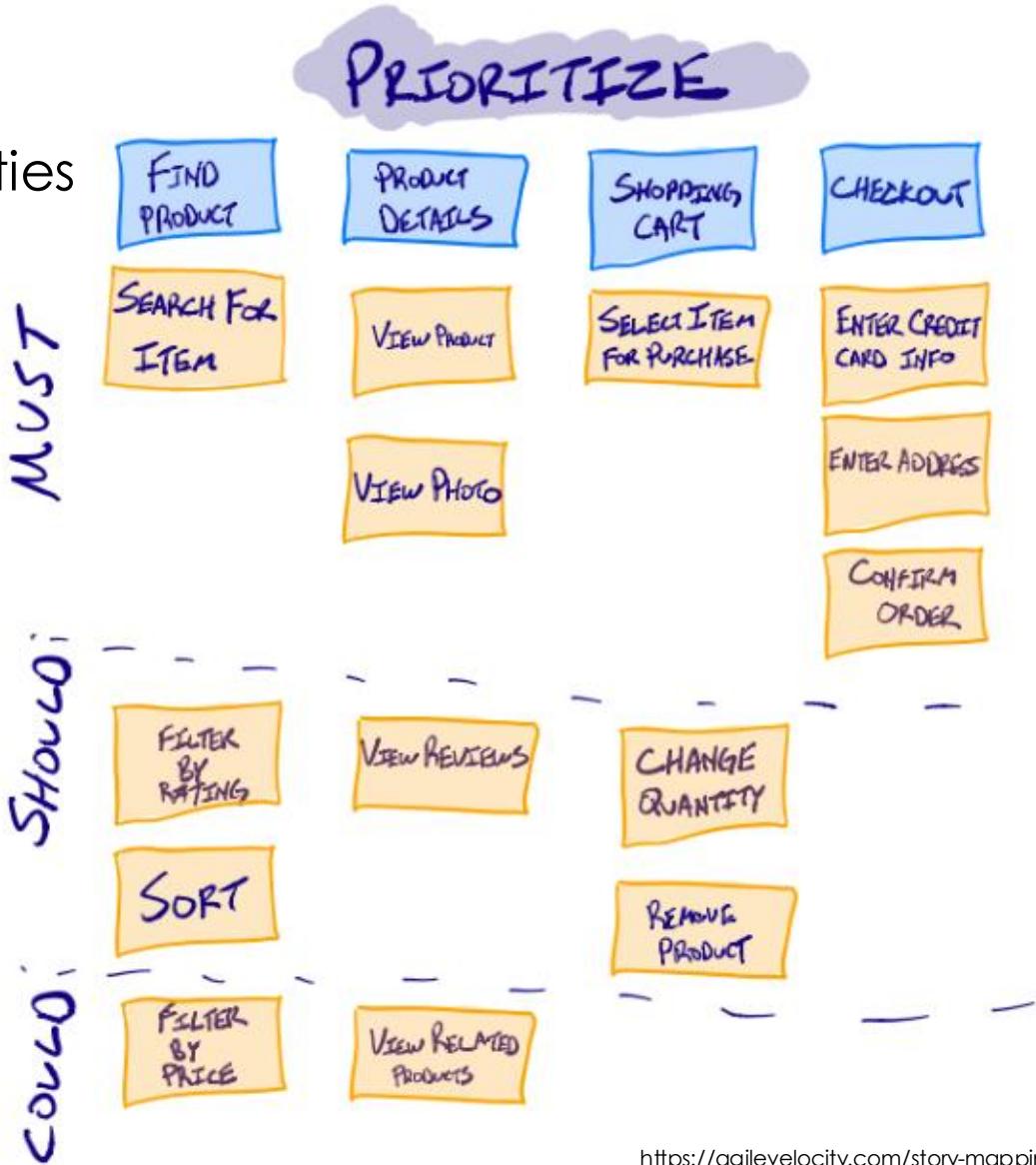
Jeff Patton & Associates, jeff@jpattonassociates.com, [@jeffpatton](https://twitter.com/jeffpatton)

(cc) BY-NC-SA

80

The anatomy of a Story Map

Columns relate to user activities



Horizontal (slices) groups are priority
Importance or frequency of use

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

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

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 a "User Story Map". The board is organized into three main sections: Sprint 1 (8 stories), Sprint 2 (5 stories), and Sprint 3 (4 stories). Each section contains several user stories represented by rounded rectangles. Some stories are highlighted with blue borders, while others are in grey boxes. A sidebar on the left provides navigation and search functions.

Frame 1

User Story Map

Sprint 1 | 8

- Searching Database
- Changing Database
- Find intended articles
- Submitting article for addition to site
- Alter information
- Moderate site
- Moderate incoming information

Sprint 2 | 5

- Filter search by practice type
- Select what columns of information to display
- Select year range of articles
- Have submission portal to submit relevant article information
- 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

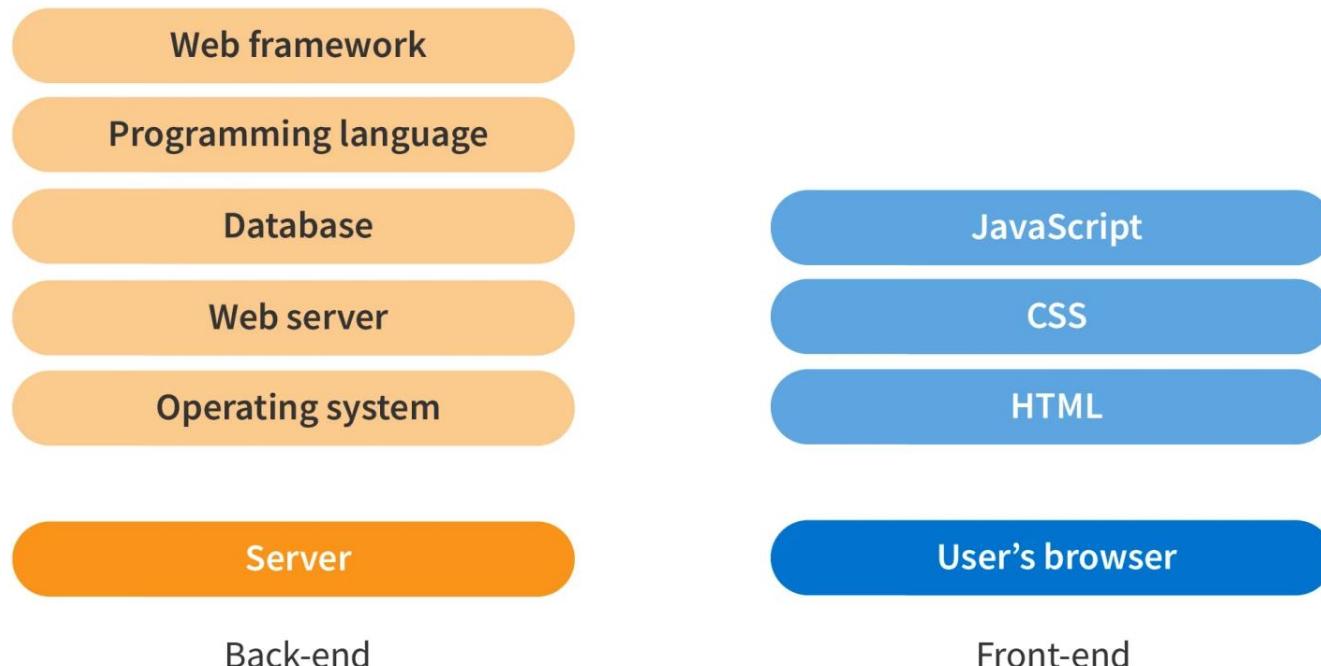
Sprint 3 | 4

- Option to approve or decline article
- Declined articles added to other db
- Compared with existing articles in both normal db and declined to automatically remove repeats

Technology stacks and Roadmapping strategy

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

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



© 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

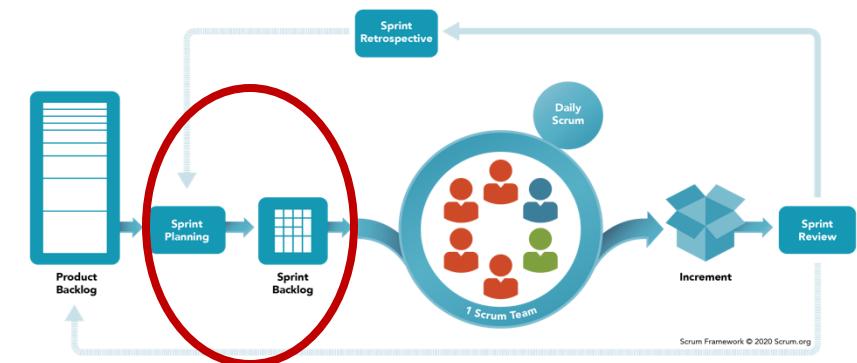
T-shirt sizes

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?

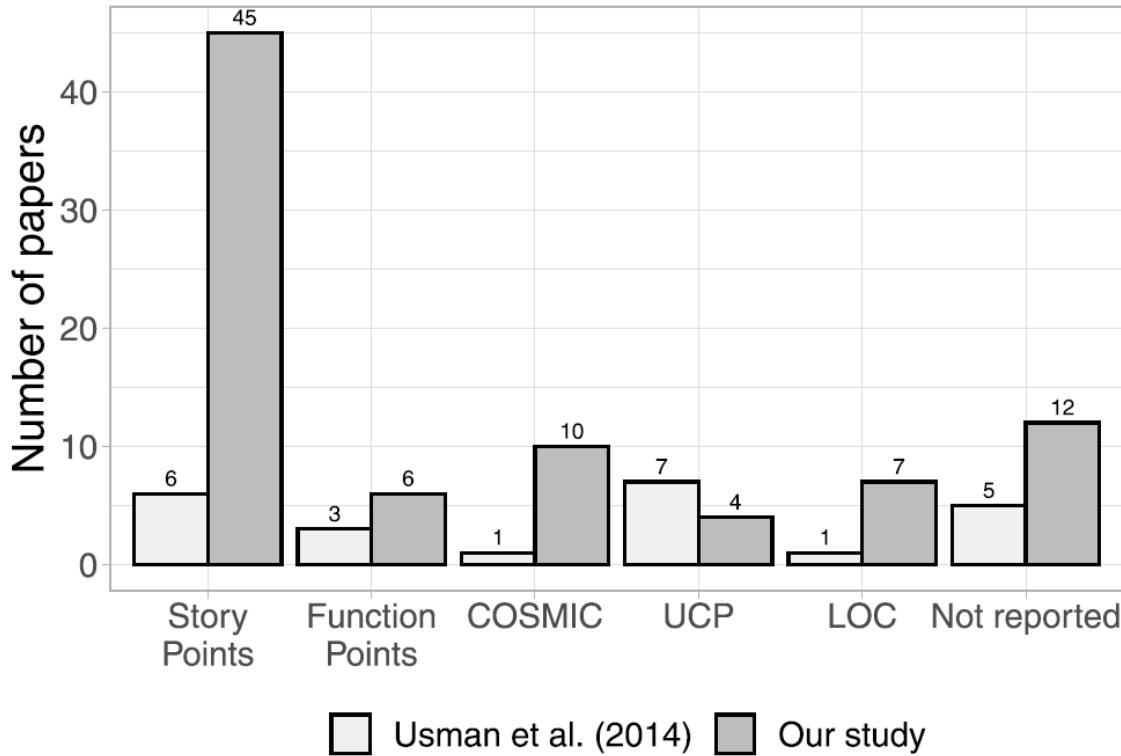


FIGURE 5. Size metrics: Comparison of our results with those of Usman *et al.* (2014).

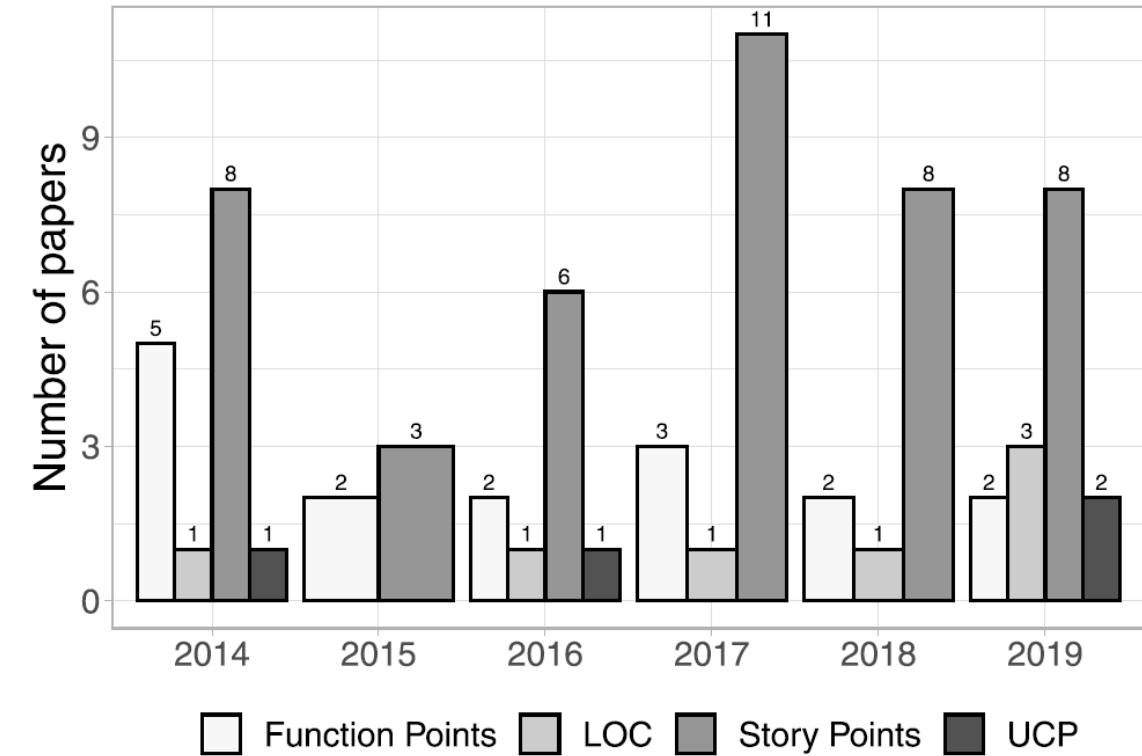


FIGURE 6. Size metrics used in the papers per year of publication.

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

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>

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)

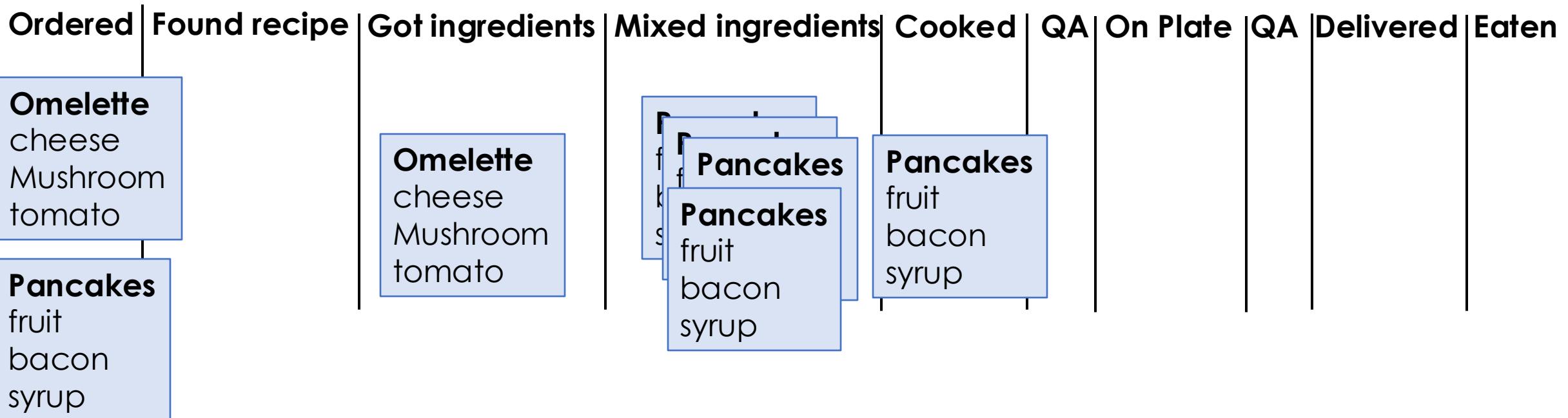
Placeholders for conversations.

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

PB

As a...
I want..
So that...
As a...
I want..
So that...

SB

As a...
I want..
As a...
I want..
So that...

Developing

As a...
I want..
As a...
I want..
So that...

Testing

As a...
I want..
So that...
As a...
I want..
So that...

Deploy to Cloud

As a...
I want..
So that...

Definition of Done?

Done

As a...
I want..
So that...

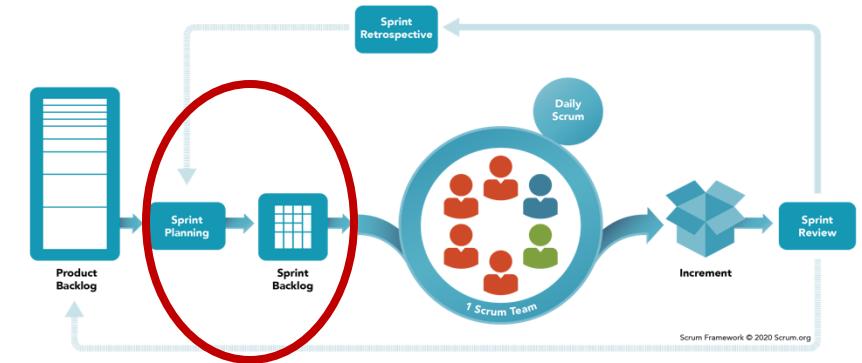
Pre-sprint – Design, tech stack, architecture Non-functional (quality) Requirements

Layered architecture

Front-end NEXT.js

Back-end Nest.js on Node.js

Database. MOngoDB



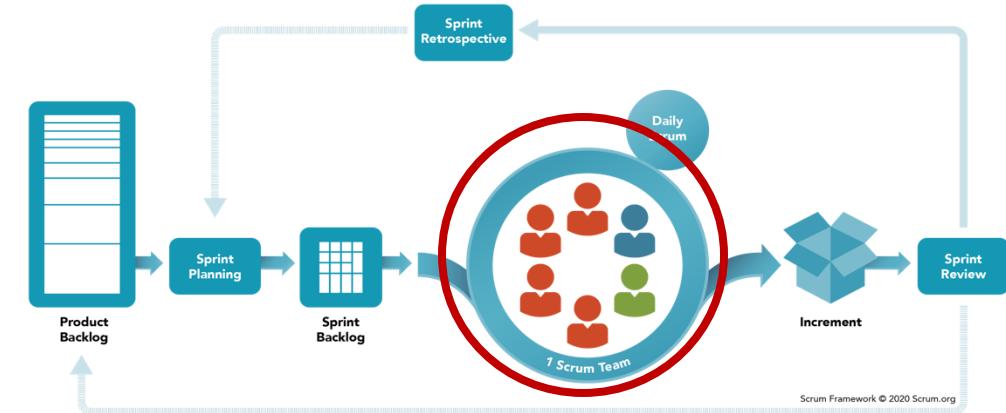
Working as an Individual in a team - workflow

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

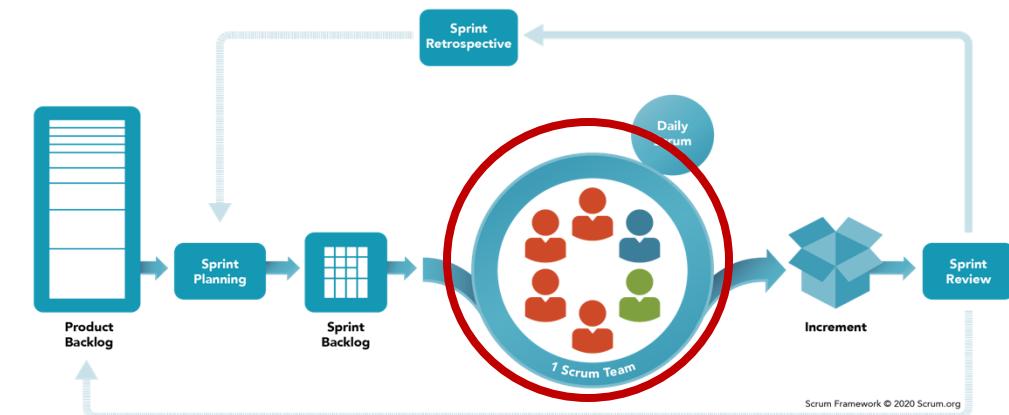
Team ownership/success trumps individual

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>



#171478945



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

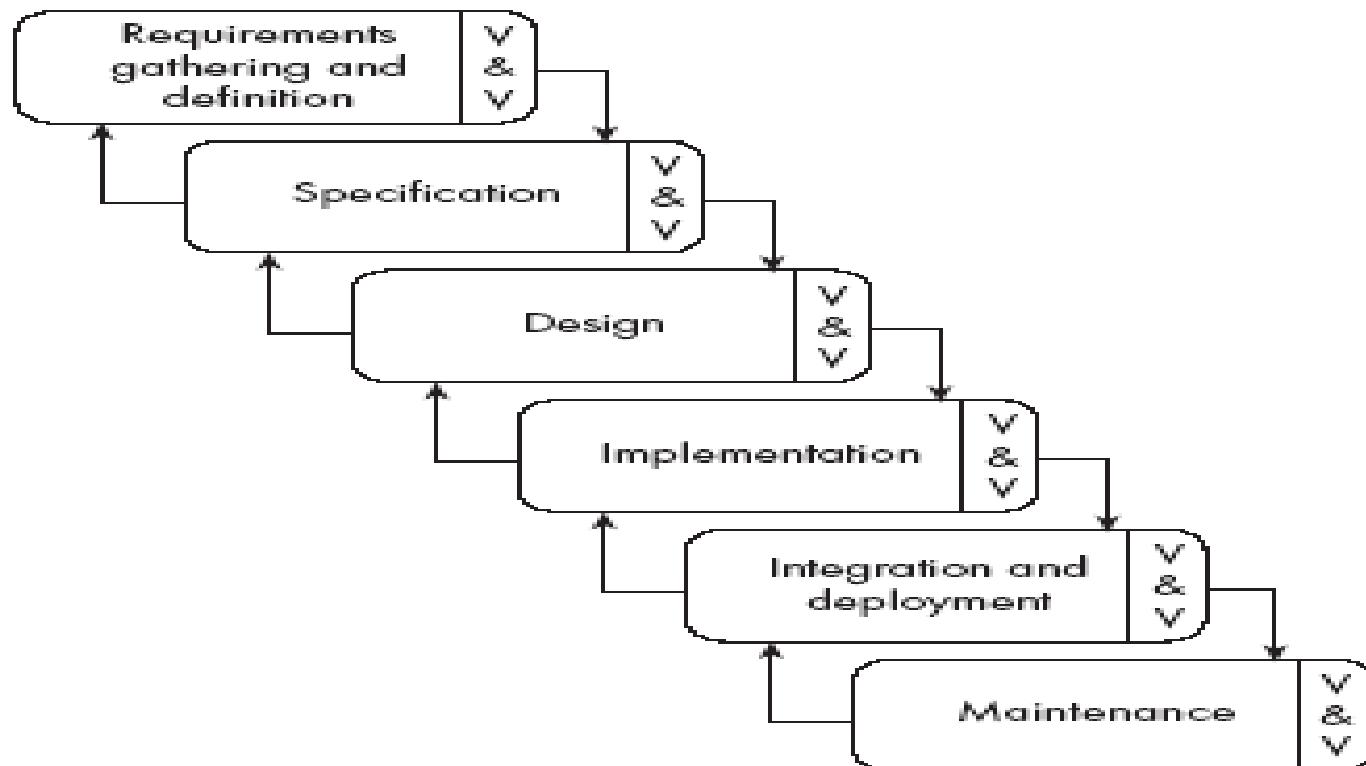
Requirements specification

Requirements validation

Requirements management

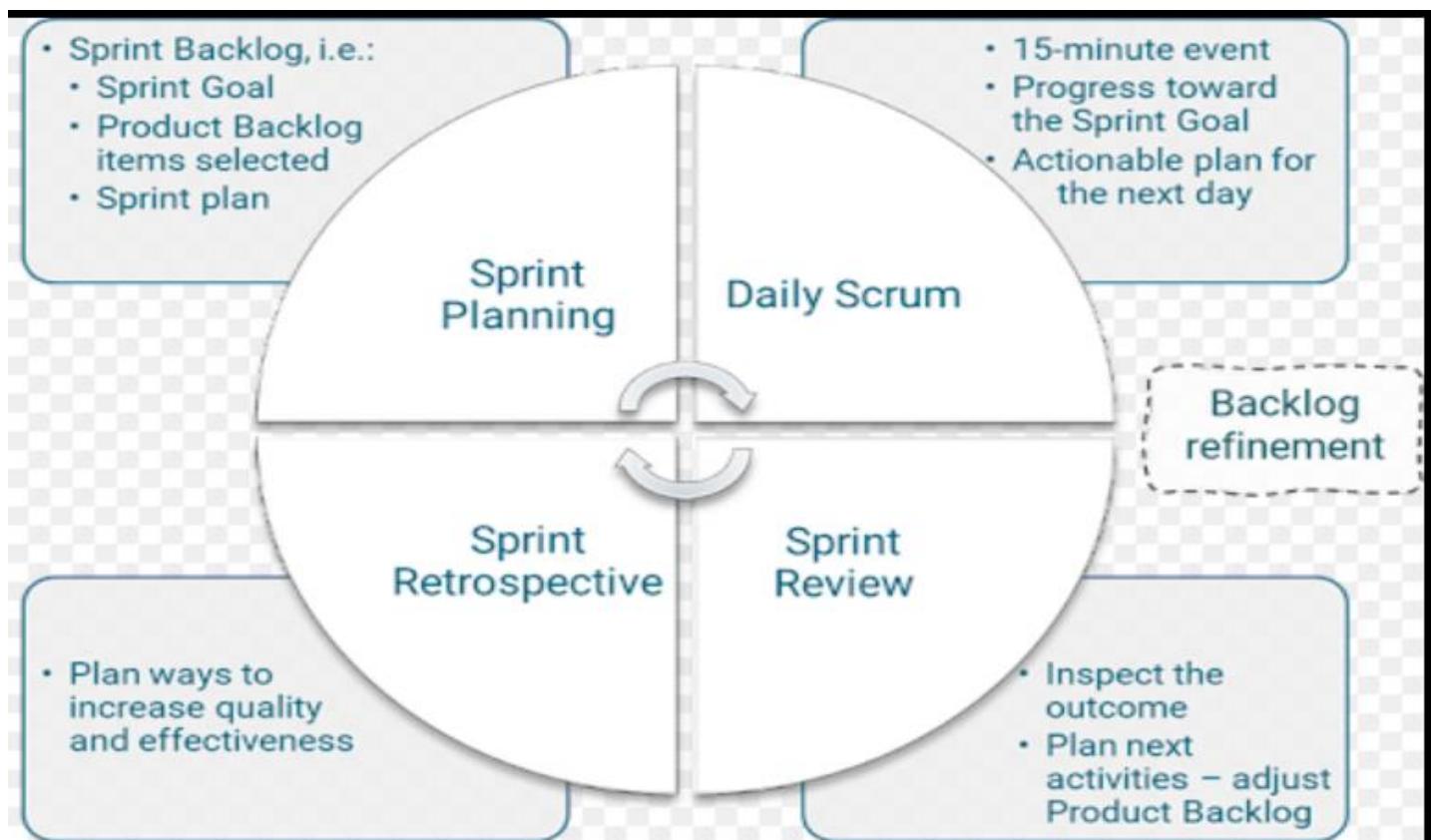
RE and software development methods

- 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

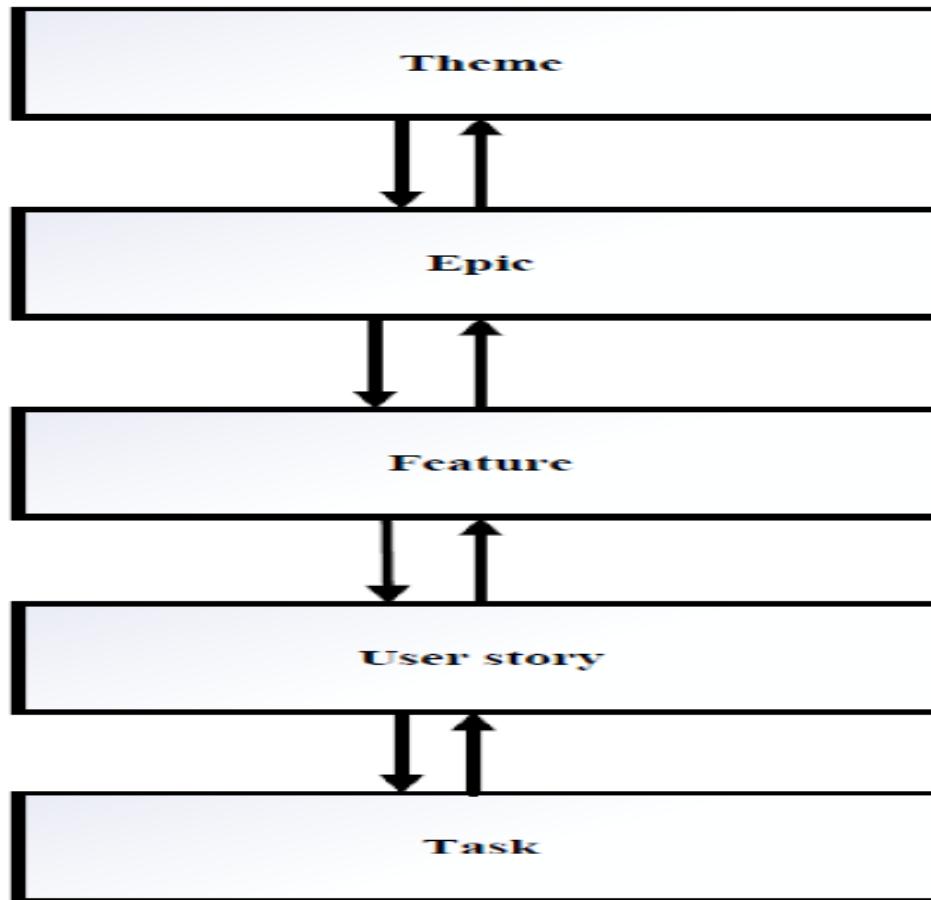
Requirements prioritization in Scaled Agile Distributed Development

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

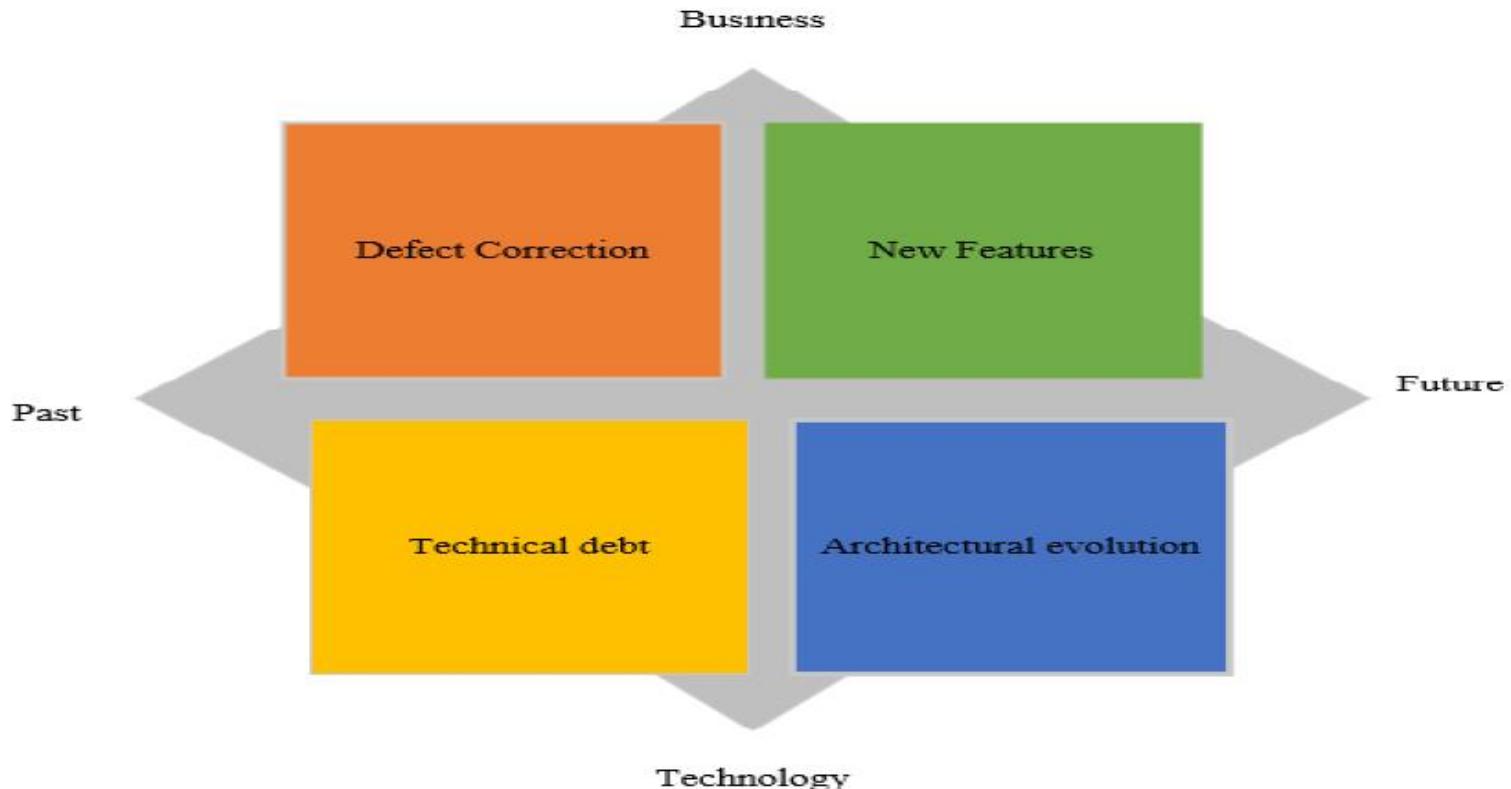
Key decision-making levels

- 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

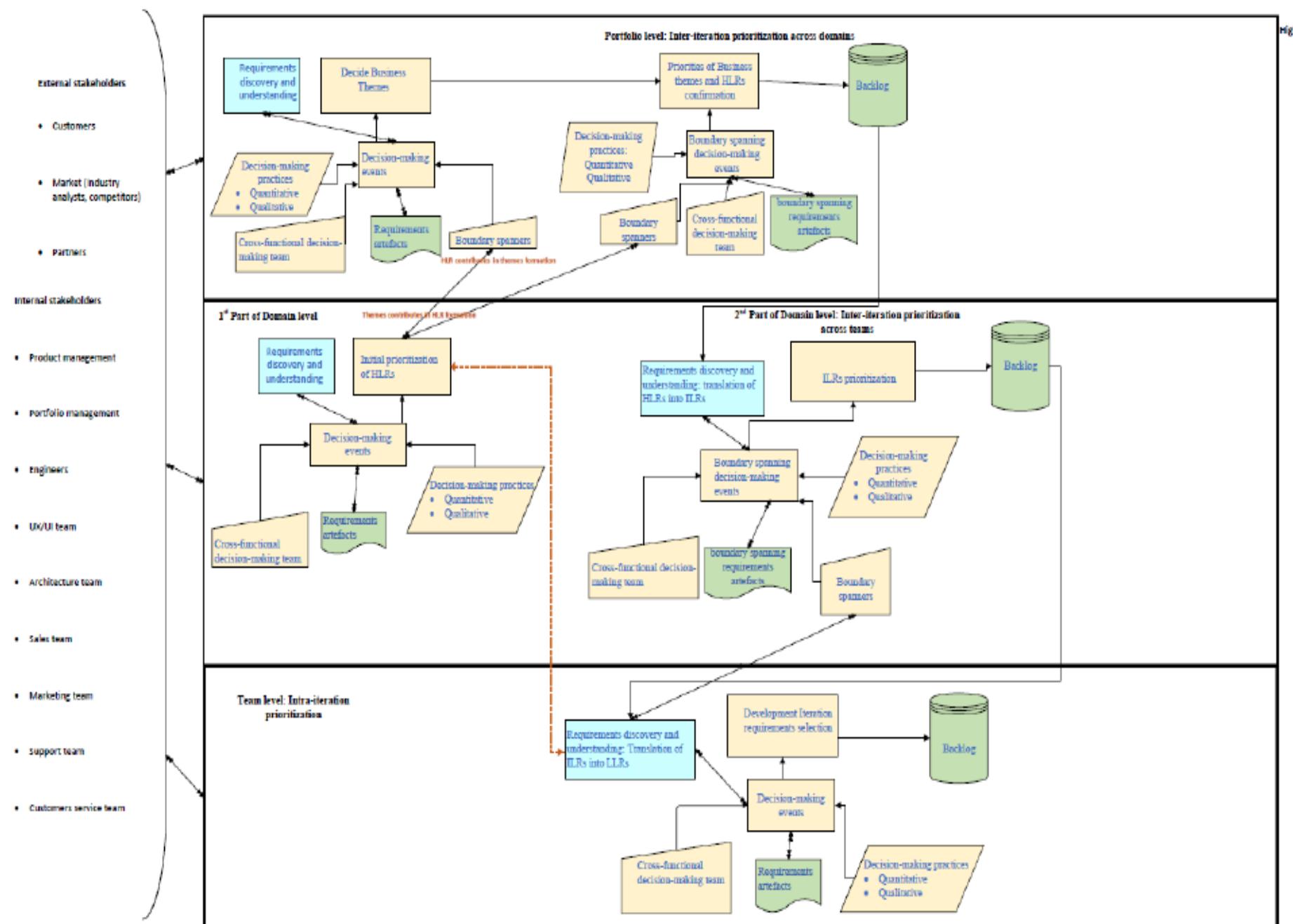
Usefulness of CTs during decision-making

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





AUT SOFTWARE ENGINEERING
RESEARCH LABORATORY

AUT

Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study



Assoc. Prof. Tony Clear
Tony.clear@aut.ac.nz



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.



Profile and Current Activities

1. Co-Director of Software Engineering Research lab at AUT <https://serl.aut.ac.nz/>
2. Global Software Engineering
 1. Scaled Agile
 2. Software Ecosystems – RSNZ Catalyst Leaders with Prof Daniela Damian
 3. GSE Education
 4. Global Collaboration
3. Computing Education
 1. Curriculum & Competencies [CC2020]
 2. Onboarding Software Professionals - SIGCSE Special Project [SERL]
 3. Editorial Roles [*ACM TOCE*, *ACM Inroads*, *Computer Science Education*], PC various conferences - CS Ed and SE
 4. Teaching BCIS Papers *Contemporary Issues in Software Engineering*, *Computing Technology In Society*, *R&D Project*, *B Eng(Hons)Industrial Project*, plus *Masters & PHD Supervision*

Do Scaling Agile Frameworks Address Global Software Development Risks? An Empirical Study

Sarah Beecham, Tony Clear, Ramesh Lal, John Noll
Journal of Systems and Software, Jan/Feb 2021
Journal First

Int'l Conference on Global Software Engineering (ICGSE) 2021

Presentation 19th May 2021
Tony Clear, Sarah Beecham, Ramesh Lal, John Noll



Overview

- Focus of the paper
- Phase one - Developing a GSD Risk Catalog
- Phase Two – Theoretical Mapping
- Phase Three – Empirical Assessment
- Conclusions

Paper Focus: Scaling Agile Frameworks and GSD Risks

- A **three-phase** process (lit review/practice-risk mapping/empirical validation),
- Illustrated how **two scaling agile frameworks**
 - –**DAD** and **SAFe**–
- Largely address **63 software development risks**
- Identified in a ***GSD Risk Catalog***
- But stronger in **some areas** than others

Phase One – Developing a GSD Risk Catalog

First phase:

- Identifying **Global Software Development** risks faced by software development organizations,
- By examining the **literature on risks**
In both **conventional** and **GSD contexts**

Result:

- A **GSD Risk Catalog** of **63 risks**,
- Divided into **four quadrants** following [**Wallace and Keil \(2004\)**](#):
 1. Customer Mandate,
 2. Scope and Requirements,
 3. Execution, and
 4. Environment

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

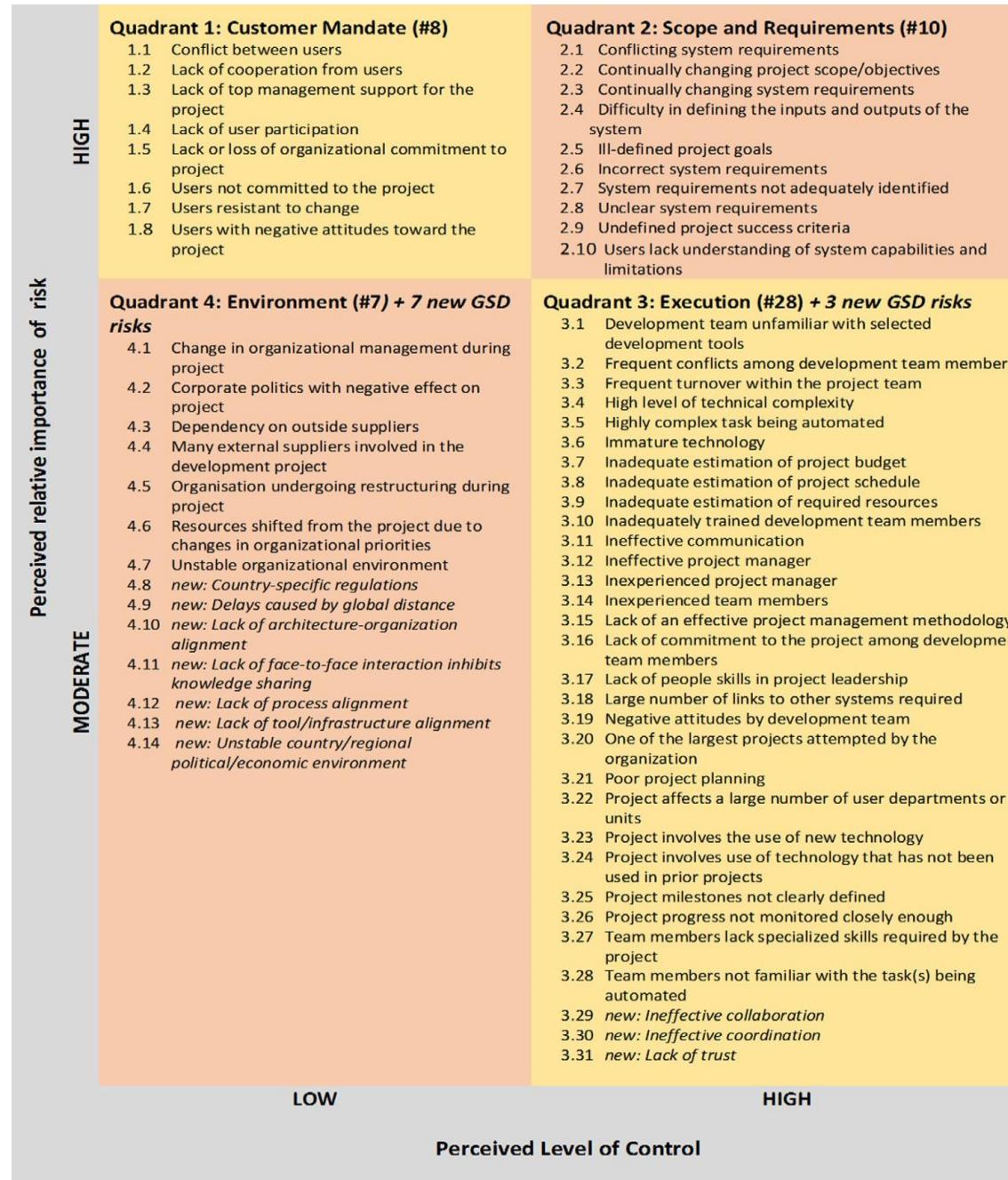


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M.,
Brereton, O. P.,
Kitchenham, B.
A., Turner, M.,
& Niazi, M.
(2014). Risks
and risk
mitigation in
global
software
development:
A tertiary
study.
*Information
and Software
Technology*,
56(1), 54-78.

Phase Two - Theoretical Mapping

- Identified potential **risk mitigation** and **elimination practices** in the **two scaling agile frameworks (DAD and SAFe)**.
- We **compared** these extracted **practices** (from **DAD** and **SAFe**) to **risks** in the ***GSD Risk Catalog*** (developed in Phase one)
- **mapped practices to risks** indicates **how scaling agile practices *might* eliminate or mitigate those risks**

Phase Three – Empirical Assessment

- Assessed the strength of the scaling agile frameworks to mitigate or eliminate risk,
Avoiding criticism that we “*speculated that the strategy would have helped observed problems*” ([Verner et al., 2014](#)),
- Performed an **empirical assessment** of the **theoretical mappings** from Phase 2.
- To determine:
 - **frequency** with which **practices** in each framework **performed** in two companies,
 - and the **risks encountered** by those companies.

2 Global Cases and Locations

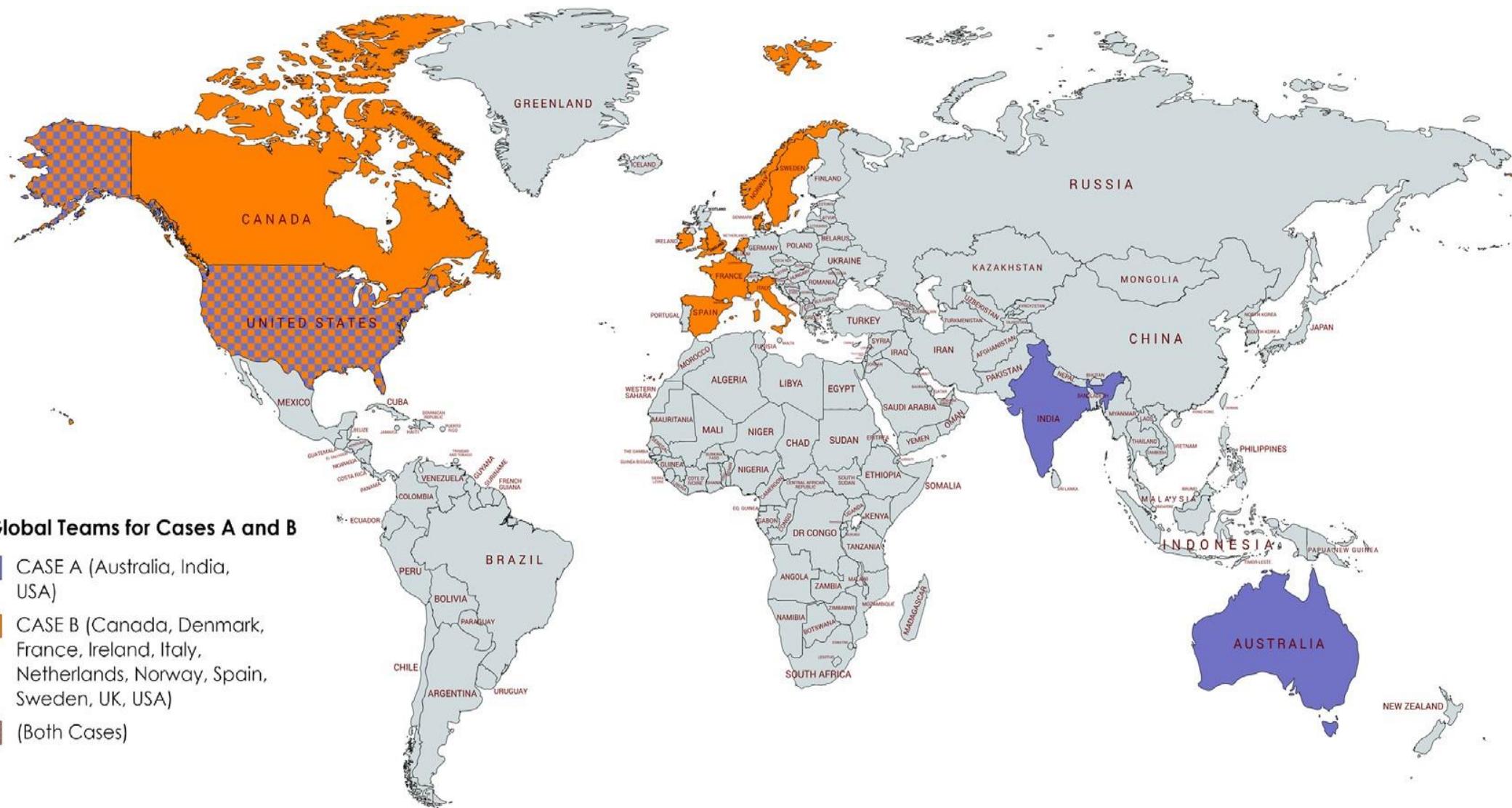


Fig. 5. Case locations.

Phase Three – Empirical Assessment (Cont'd)

- Examined **observation** and **interview notes** and **transcripts**, **self-assessment survey results**,
- In multiple case study of two **global software companies**
- To understand
 - Extent to which DAD and SAFe practices (from theoretical mapping) were implemented in the company
 - Evidence of any of the risks (in the mapping or GSD Risk Catalogue) occurring in either company

If **practice is implemented** in a company and



risk NOT seen = theory supported

risk seen = theory unsupported

Phase Three – Broad Conclusion

Adds to **limited empirical evidence of efficacy of scaling agile frameworks.**

suggesting claims

Scaling Agile Frameworks address risk and are driven by value

have some validity!

Conclusions From Paper

Of the **four quadrants** in the **GSD Risk Catalog**

1. **Customer Mandate risks** quadrant appears to be **better addressed through the SAFe framework than DAD**
2. **Scope and Requirements risks** are **addressed well by both methods**
3. **Execution risks** are **better mitigated by DAD than SAFe**
4. **Environment risks** are **less well addressed by either approach**

Suggests **Environment set of risks** are **less amenable to being addressed by a process framework.**

GSD Risks – not fully addressed by Scaling Agile Practices?

17 (out of 63) risks observed in both cases, eight are GSD risks specific:

1. Ineffective collaboration,
2. Ineffective coordination,
3. Lack of trust,
4. Country-specific regulations,
5. Delays caused by global distance,
6. Lack of architecture-organization alignment,
7. Lack of face-to-face interaction inhibits knowledge sharing,
8. Lack of process alignment.

Implications for all remote and home working?

Further Conclusions From Paper

Creating the **GSD Risk Catalog**

- Found many **Global Software Development risks, not identified** in the Wallace and Keil inventory
- **Ten new risks** GSD risks added to inventory
- Eight of these ten **GSD risks observed** (as shown on previous slide) in both companies, except:
 - Lack of tool/infrastructure alignment and
 - Unstable country/regional political/economic environment

These **new risks** appear to be **endemic** and suggest
a risk tariff in GSD

Final Takeaway

The **result** of a three phased methodology created a **scaling agile risk theoretical mapping** applied in a multiple case study showing how **two** scaling agile frameworks **Disciplined Agile Delivery** and the **Scaled Agile Framework** can potentially **eliminate** or **mitigate** the majority of 'software project' risks

- many **global software development** risks still pervade.

Scaling Agile Frameworks do not support every GSD risk.



Expansion

- GSD Risk Catalog
 - Process of dev't elaborated in Supplementary Technical Report pp. 5 - 7
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2020). *Companion to manuscript: Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study* https://www.lero.ie/sites/default/files/Beecham_2020_TR003.pdf
- Practices by Framework
 - *The result of these three phases is a **scaling agile risk theoretical mapping** that shows how two scaling agile frameworks—Disciplined Agile Delivery and the Scaled Agile Framework—can potentially eliminate or mitigate software project risks in global software development. [p. 21 also process elaborated on p.23 of TR]*
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study. *Journal of Systems and Software*, 171(110823). <https://doi.org/https://doi.org/10.1016/j.jss.2020.110823>
 - Table 11 of paper – DAD p.22
 - Table 12 of paper – SAFE p. 25



Utility and Impact?

- BCIS Current R&D Project
- Security Policy for a Globally Distributed SME [Various Asia/Pacific sites]
 - Paananen, H., Lapke, M., & Siponen, M. (2020). State of the art in information security policy development. *Computers & Security*, 88, 101608.
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study. *Journal of Systems and Software*, 171(110823).
<https://doi.org/https://doi.org/10.1016/j.jss.2020.110823>
- Startup Software Dev't Org
- Markets a Software Solution which helps MSPs (Managed Service Providers) to automate common processes associated with IT support and administration



Case context?

- **Culture of company**
- Freewheeling style
- Historical flow of contract staff
- Inconsistent and Individualistic practices
- Personal use of open-source tools
- Security risks to be analysed and addressed
- Move towards more team-based development
- Desire to grow and scale
- **How to develop and implement a security policy?**
- **Standardization, prioritizing, what to mandate vs. encourage?**



Framing the situation ?

- How to develop and implement a security policy?
- Standardization, prioritizing, what to mandate vs. encourage?

- Possible contribution from our scaled agile risk model?
- Focus on risks to determine priorities?
- Focus on practices to identify areas of concentration and sequence of activities?
- Potential contribution to framing both the problem and solution
- Issues highlighted from *GSD Risk Catalog* below

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

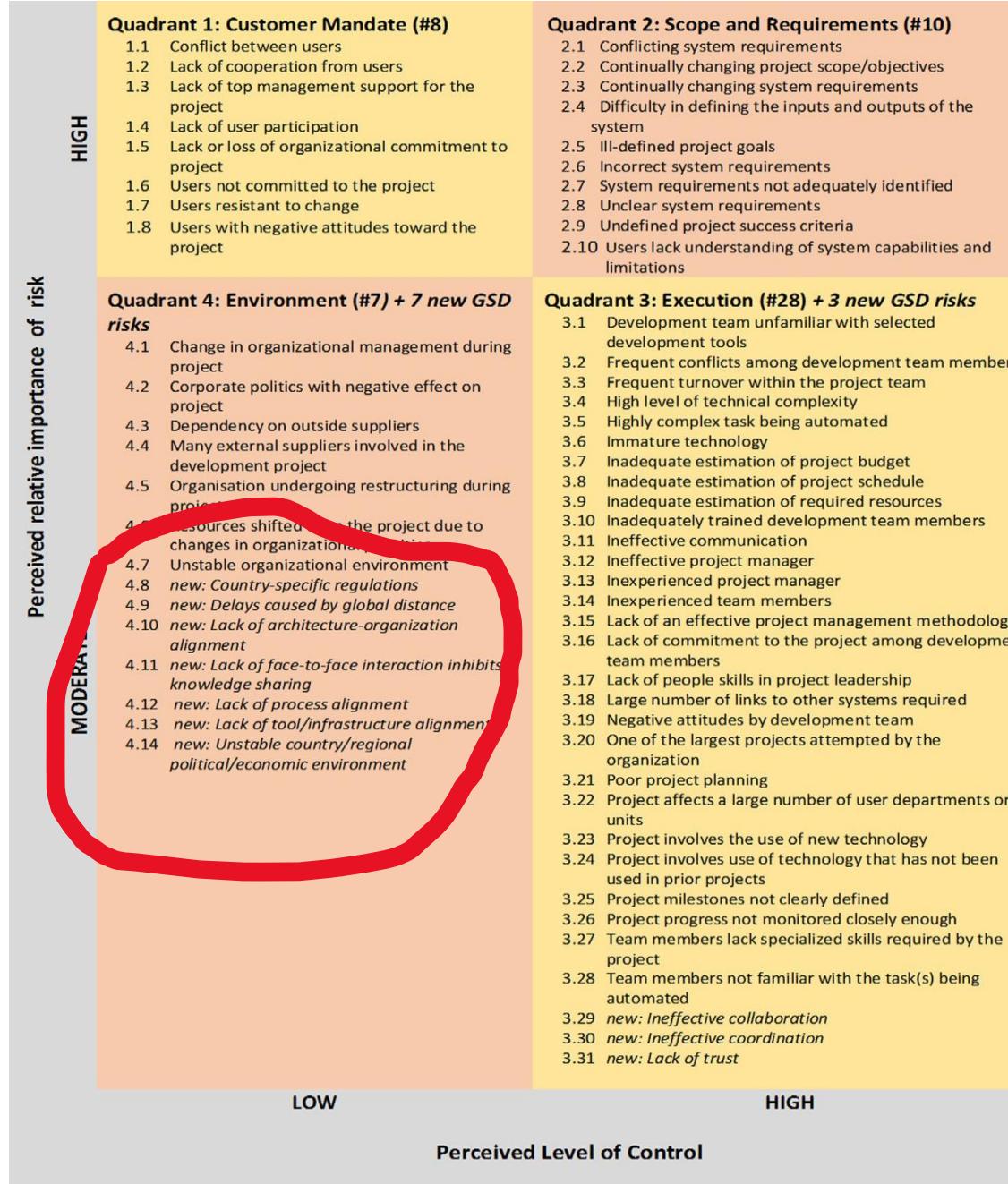


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, 56(1), 54-78.

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

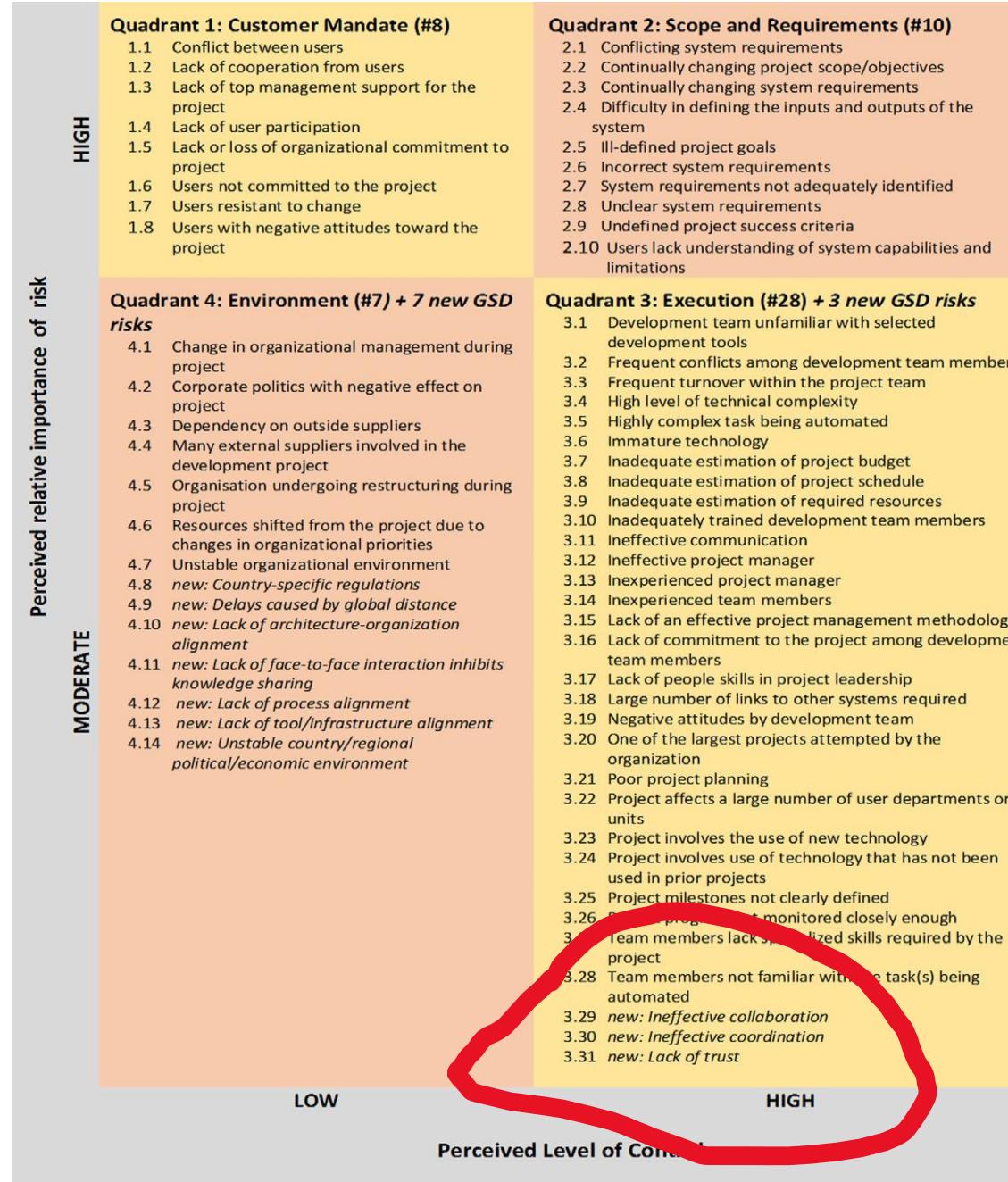


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, 56(1), 54-78.



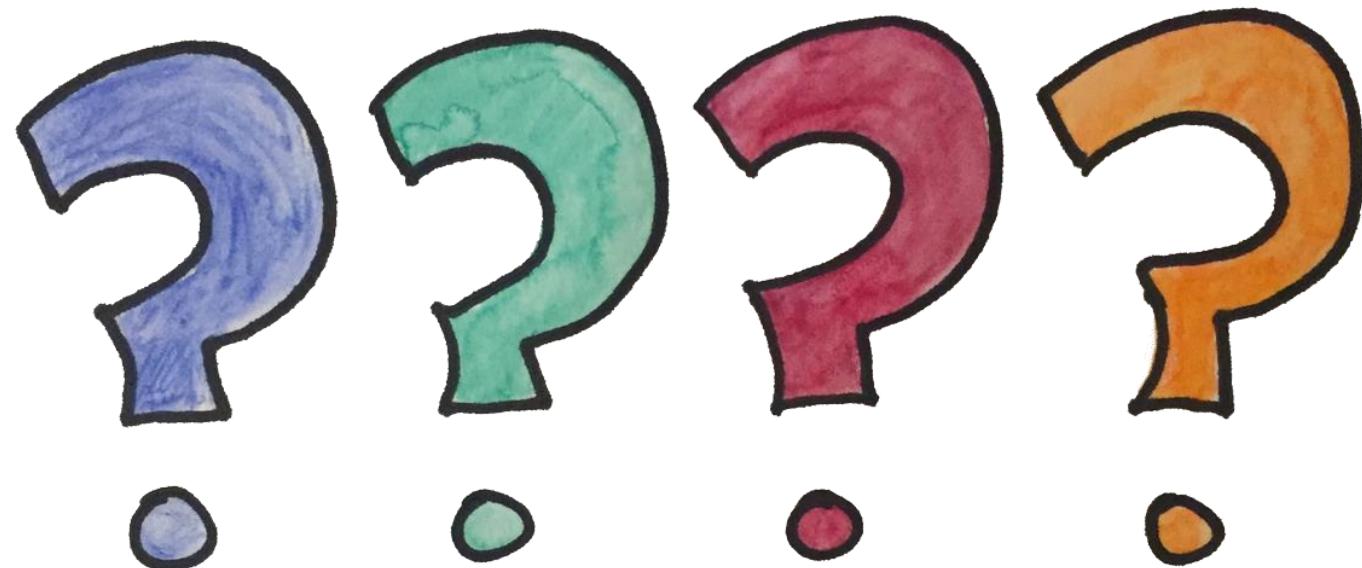
Utility and Academic Impact?

- **Google Scholar Citations [19/05/2022 – 2021 paper]**

- 1) Evaluating the role of scrum methodology for **risk management** in information technology enterprises
- 2) Adoption of Large-Scale Scrum Practices through the Use of Management 3.0
- 3) Agile transformation, from classical-to agile project management in a multidisciplinary production environment, a case study
- 4) Tools Engineers Need to **Minimize Risk around CI/CD Pipelines** in the Cloud
- 5) Servitization **Program Management Process Based On Scaled Agility**: A Facilitating Framework
- 6) Analyzing **SAFe Practices with Respect to Quality Requirements**: Findings from a Qualitative Study
- 7) Customer Controlled Managed Services Processes for Productivity Gains without Breaking Contractual Obligations: An Exploratory Study
- 8) Evaluating AGILE adoption in software delivery organizations
- 9) Analyzing **SAFe Practices with Respect to Quality Requirements**: Findings from a Qualitative Study
- 10) Toward Unveiling How **SAFe Framework Supports Agile** in Global Software Development
- 11) An effective agile development process by a hybrid **intelligent effort estimation protocol**
- 12) Agile and generic work values of British vs Indian IT workers: a **culture-clash** case
- 13) Systematic Literature Review: **Causes of Rework** in GSD
- 14) Från kaffeautomaten till digitala mötesrum: Mjukvaruutvecklare upplevelser av att arbeta med agila utvecklingsmetoder under rådande pandemic *From the coffee machine to digital meeting rooms: Software developers' experiences of working with agile development methods during the current pandemic*
- 15) **Modelo de evaluación de metodologías de desarrollo de software web Evaluation model of web software development methodologies**



- Nga mihi
- And
- Questions?



AUT

ENSE 701 – Virtual Team Guidance (based on Comp501 presentation)



Course lecturer –
Assoc. Prof. Tony Clear



Outline

1. COVID19 and Working as a Virtual Team
2. Some Guidance from the literature
 1. Jarvenpaa & Leidner (1998)
 2. Gordon (2017)
3. Hybrid Workplaces
4. Stages in the life of a virtual team
5. Stages of your Team Project Assignment
6. Key actions at each stage

COVID19 and Working as a Virtual Team

1. Lockdown meant working online as a team
2. Need to balance home life and study
3. Need to work around any technology issues
4. Need to communicate actively
5. Let others know if you are not going to be able to:
 1. Be at a meeting
 2. Deliver your agreed contribution
 3. Perform your role
 4. Meet milestones
6. Agree a backup plan!
7. Be sensitive to one another and your needs in a difficult time
8. But committing to the work and one another will help
 1. Bring satisfaction through achievement
 2. Bring a sense of mutual support
9. Let engaging in the learning give you structure in an uncertain time

Virtual Teams – Some Guidance from the Literature

1. Virtual Teams have been an active field of study for 35 years
2. My own research spans 25 of those
3. But still challenging!
4. See the issues discussed in my recent *Inroads* column

Clear, T. (2021). Loosening Ties: Permanently Virtual Teams and the Melting Iceberg of Relationship. *ACM Inroads*, 12(3), 6-8. <https://doi.org/10.1145/3479419>

Beyond Virtual Teams –Hybrid Workplaces

1. The word “hybrid” has become one popular umbrella label attributed to various work-related terms.
2. These days, we often read about hybrid workplaces or hybrid offices, hybrid work or working, as well as hybrid teams.
3. Google Workspace experts define *hybrid work* as “*a spectrum of flexible work arrangements in which an employee’s work location and/or hours are not strictly standardized.*”
4. In other words, anything that lies in the middle of “in the office, nine till five” and “anywhere around the world at any time.”

Smite, D., Christensen, E. L., Tell, P., & Russo, D. (2023). The Future Workplace: Characterizing the Spectrum of Hybrid Work Arrangements for Software Teams. *IEEE Software*, 40(2), 34-41. <https://doi.org/10.1109/MS.2022.3230289>

Hybrid Workplaces and Team Typologies

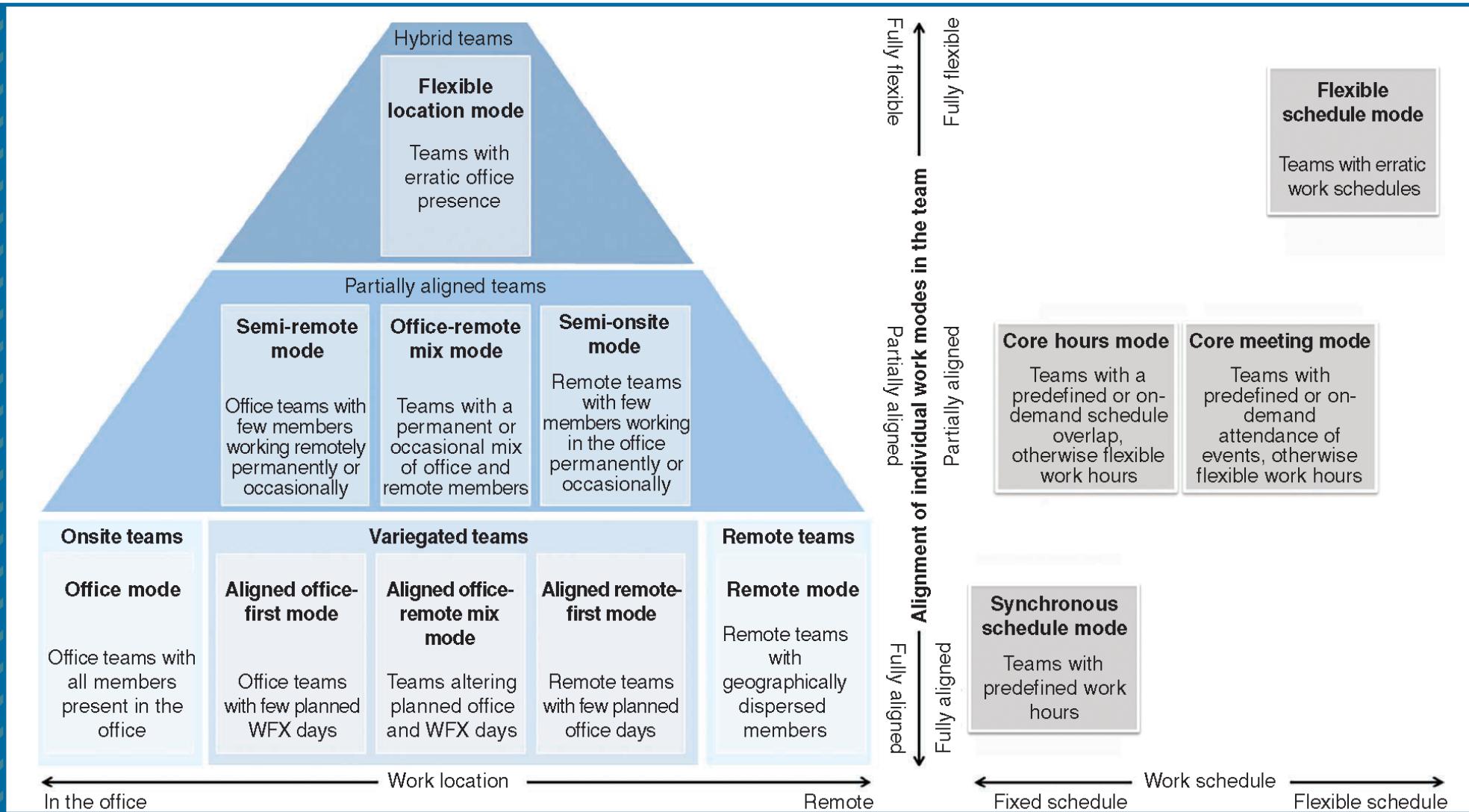
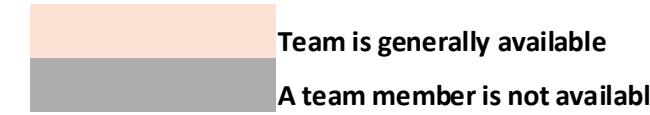


FIGURE 1. Team typology and the spectrum of work arrangements.

Navigating BCIS Team availability and location?

Manage My Lunch Team General Availability

Time	MON	TUE	WED	THU	FRI	SAT	SUN
8:00							
8:30							
9:00							
9:30							
10:00							
10:30							
11:00							
11:30							
12:00							
12:30							
13:00							
13:30							
14:00							
14:30							
15:00							
15:30							
16:00							
16:30							
17:00							



Virtual Teams – Jarvenpaa & Leidner (1999)

"in global virtual teams, trust might take on a form of swift trust with some variations.

Trust might be imported, but is more likely created via a communication behavior established in the first few keystrokes.

Communication that rallies around the project and tasks appears to be necessary to maintain trust.

Social communication that complements rather than substitutes for task communication may strengthen trust.

Finally, responding behaviors are as critical as initiating behaviors and members have to explicitly verbalize their commitment, excitement and optimism" (Jarvenpaa & Leidner, 1999).

Jarvenpaa, S., & Leidner, D. (1999). Communication and Trust in Global Virtual Teams. *Organization Science*, 10(6), 791-815.

Jarvenpaa, S., & Leidner, D. (1998). Communication and Trust in Global Virtual Teams. *Journal of Computer Mediated Communication*, 3(4).

Virtual Teams – Jarvenpaa & Leidner (1998)

The trust facilitating behaviours and actions are depicted in the table below.

Communication Behaviors that facilitated trust early in a group's life	Communication Behaviors that helped maintain trust later in a group's life
Social communication	Predictable communication
Communication of enthusiasm	Substantial and timely responses
Member actions that facilitated trust early in a group's life	Member actions that helped maintain trust later in a group's life
Coping with technical uncertainty	Successful transition from social to Procedural to task focus
Individual initiative	Positive leadership
	Phlegmatic response to crises

Table 6: Trust Facilitating Communication Behaviours And Member Actions (from Jarvenpaa & Leidner, 1998)

Virtual Teams – Gordon (2017)

In summary, the study shows that the most appropriate interactive leadership styles are significant and different from those of face-to-face teams. Furthermore, these styles are affected by, and affect, each team stage. Exercising the explaining and directive styles, along with the questioning style, in all stages of a team, but particularly in stages 1 and 2, may be the most effective way to elicit actions. As a result of this action research it was agreed to make the following six findings more explicit to the GlobCom team, including:

Gordon, A. (2017). *Leadership Interaction in Global Virtual Teams: Roles Models and Challenges* (PhD Thesis). Auckland University of Technology, Auckland. Retrieved from <http://hdl.handle.net/10292/10769>

Virtual Teams – Gordon (2017 cont'd)

1. Team stages have different levels of activity and demands and identifying them can help structure the workload.
2. Delegated roles are more likely to elicit actions and justify the creation of additional roles.
3. The directive and explanatory interaction styles are the most effective in eliciting actions with additional styles added depending on the situation.
4. Team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum.
5. The apologetic interaction style, despite it being socioemotional with its intention to be caring, is likely to diminish telepresence.
6. The team mentor role generates feedback and may need to be more active earlier.

Gordon, A. (2017). *Leadership Interaction in Global Virtual Teams: Roles Models and Challenges* (PhD Thesis). Auckland University of Technology, Auckland. Retrieved from <http://hdl.handle.net/10292/10769>

Stages in the Life of a Virtual Team

In a broad sense we can think of virtual teams having three stages:

- Beginning
- Middle
- End

Stages of your team project assignment

- Beginning
 - Team Member Allocation
 - Team Project Draft Proposal
 - **Team Proposal Presentation (assessed)**
- Middle
 - Portfolio Review
- End
 - **Team Project Presentation (assessed)**
 - **Team Portfolio Submission (assessed)**
 - **Team Portfolio [individual components] Submission (assessed)**

Key actions at each stage

- Beginning
 - (stage 1) Team Member Allocation
 - (stage 2) Team Project Draft Proposal
 - **(stage 3) Team Proposal Presentation (assessed)**
- Middle
 - (stage 4) Portfolio Progress Review
- End
 - **(Stage 5) Team Project Presentation (assessed)**
 - **(Stage 6) Team Portfolio Submission (assessed)**
 - **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Key actions at each stage – Virtual

- Beginning
 - (stage 1) Team Member Allocation – completed prior to lockdown but team engagement unclear
 - (stage 2) Team Project Draft Proposal – completed?? but team engagement unclear
 - **(stage 3) Team Proposal Presentation (assessed) – Due first week after break**

Additional Actions to Support Virtual Team Functioning

- Team icebreaker – using Team wiki (see next slide)
- Team Leader selection [complementing roles identified from stage 3]
- *Refer to Jarvenpaa & Leidner – actions early in team's life (social communication, communication of enthusiasm, coping with uncertainty, individual initiative)*
- *Refer to Gordon – defined roles elicit actions; interaction by giving directions and explanations helps elicit action*

Key actions at each stage – Icebreaker

- Team icebreaker – using Team wiki under groups on Blackboard
 - To check your team (under Teams tutorials – labs channel)
https://teams.microsoft.com/l/file/7B3E5B1E-D3EB-4A9D-9FCC-16CFB2C6B13A?tenantId=5e022ca1-5c04-4f87-8db7-d588726274e3&fileType=xlsx&objectUrl=https%3A%2F%2Fautuni.sharepoint.com%2Fsites%2FCOMP501_2021_02ComputingTechnologyinSocietySem22021%2FShared%20Documents%2FLectures%2FAssignment%203%20Team%20Assigned.xlsx&baseUrl=https%3A%2F%2Fautuni.sharepoint.com%2Fsites%2FCOMP501_2021_02ComputingTechnologyinSocietySem22021&serviceName=teams&threadId=19:6bb6686b76fc4085a61e645979b7c7c2@thread.tacv2&groupId=3fb30b0c-e8ab-43cf-998a-2af3a591f321
- Team Leader selection
- The Icebreaking phase involves virtual team (VT) members getting to know each other and appointing a group leader.
- **Task 1:** Participants in each VT become acquainted with one another and each student creates a “page” in their group’s wiki to introduce themselves to the team.
- Students in each VT interact in their VT wiki and each group member creates their own page there. It is suggested that in their page each student provide some information about things like their birthplace, favourite travel destination, interests and hobbies, their future career plans and their photo, and also some web links to make it more informative.
- **Task 2:** Based on their introductions, the students in each VT need to **select a leader** for their virtual team. Use the VT Discussion boards or Teams or other agreed way of communicating among your members – make sure you agree how you will communicate!
- Also agree how you will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – Mid Project

- Middle
 - (stage 4) Portfolio Progress Review

Additional Actions to Support Virtual Team Functioning

- Team Leader initiative required
 - Plan time for joint review of separate components so they can be integrated
 - Consider establishing an editor role to ensure consistent style and approach
- *Refer to Jarvenpaa & Leidner – actions later in team's life (predictable communication; substantial and timely responses; successful transition from social, to procedural to task focus; positive leadership; phlegmatic response to crises)*
- *Refer to Gordon – team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum; delegated roles are more likely to elicit actions and justify the creation of additional roles*
 - Make sure to reinforce teams approach to how they will communicate!
 - Make sure team roles and tasks are planned and reinforced
 - Also reinforce approach and how team will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – End Project

- End
- **(Stage 5) Team Project Presentation (assessed)**
- **(Stage 6) Team Portfolio Submission (assessed)**
- **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Additional Actions to Support Virtual Team Functioning

- Team Leader initiative required – also contact TA or Lecturer if any concerns
- Make sure team roles, tasks, deadlines are reinforced
- Confirm presentation roles (stage 5)
- Include time for joint review of separate components so they can be integrated
- Review the need and contribution of the editor role to ensure consistent style and approach
- *Refer to Jarvenpaa & Leidner – actions later in team's life (predictable communication; substantial and timely responses; successful transition from social, to procedural to task focus; positive leadership; phlegmatic response to crises)*
- *Refer to Gordon – team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum; ; delegated roles are more likely to elicit actions and justify the creation of additional roles*
- Make sure to reinforce teams approach to how they will communicate!
- Also reinforce approach and how team will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – End Project

- End - Final
- **(Stage 5) Team Project Presentation (assessed)**
- **(Stage 6) Team Portfolio Submission (assessed)**
- **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Additional Actions to Support Virtual Team Functioning

- Confirm who is submitting the team portfolio
- Remember to submit your own components
- Remember to thank your team members for their contributions, their effort and collegial approach to supporting one another in doing good work ☺

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)



< 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



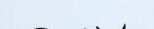
All Documents



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??

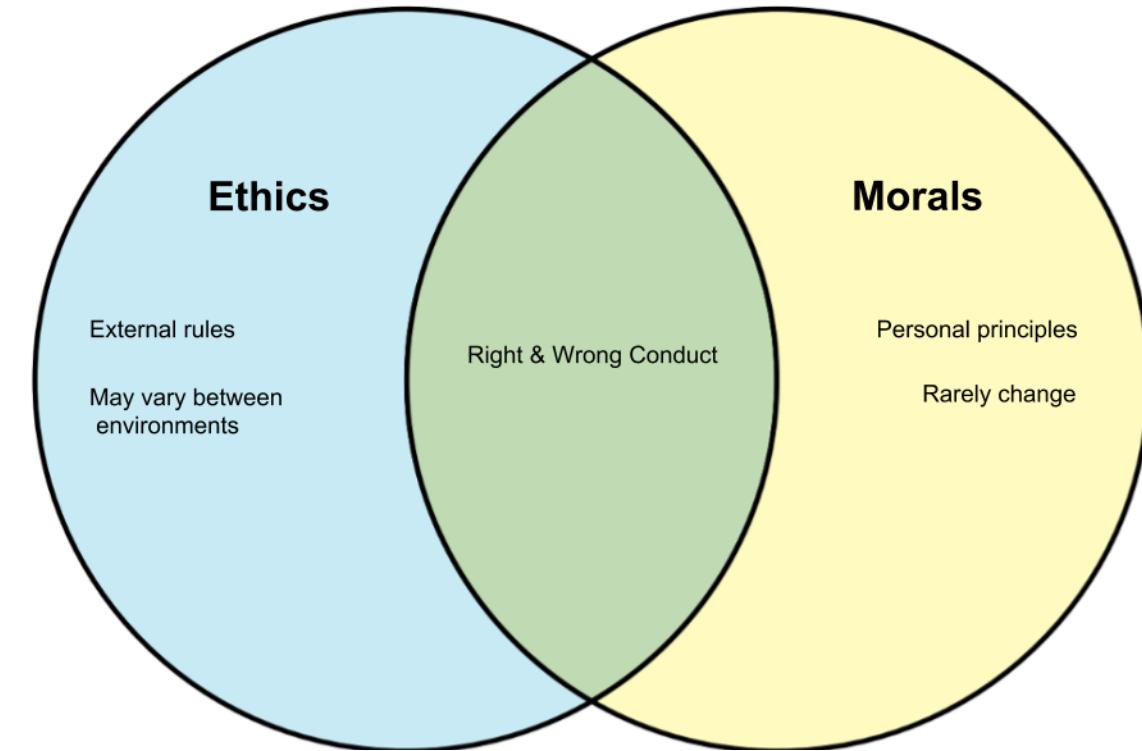
Computing Professionals - Responsibilities

- 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

Technology

101

		Topics of Ethical Analysis							
		Responsibility		Ethical Issues					
Levels of Social Analysis	Individual	Professional	Quality of Life	Use of Power	Risks & Reliability	Property Rights	Privacy	Equity & Access	Honesty & Deception
	Individuals								
	Communities & Groups								
	Organizations								
	Cultures								
	Institutional Sectors								
	Nations								
	Global								

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.

Martin et al.



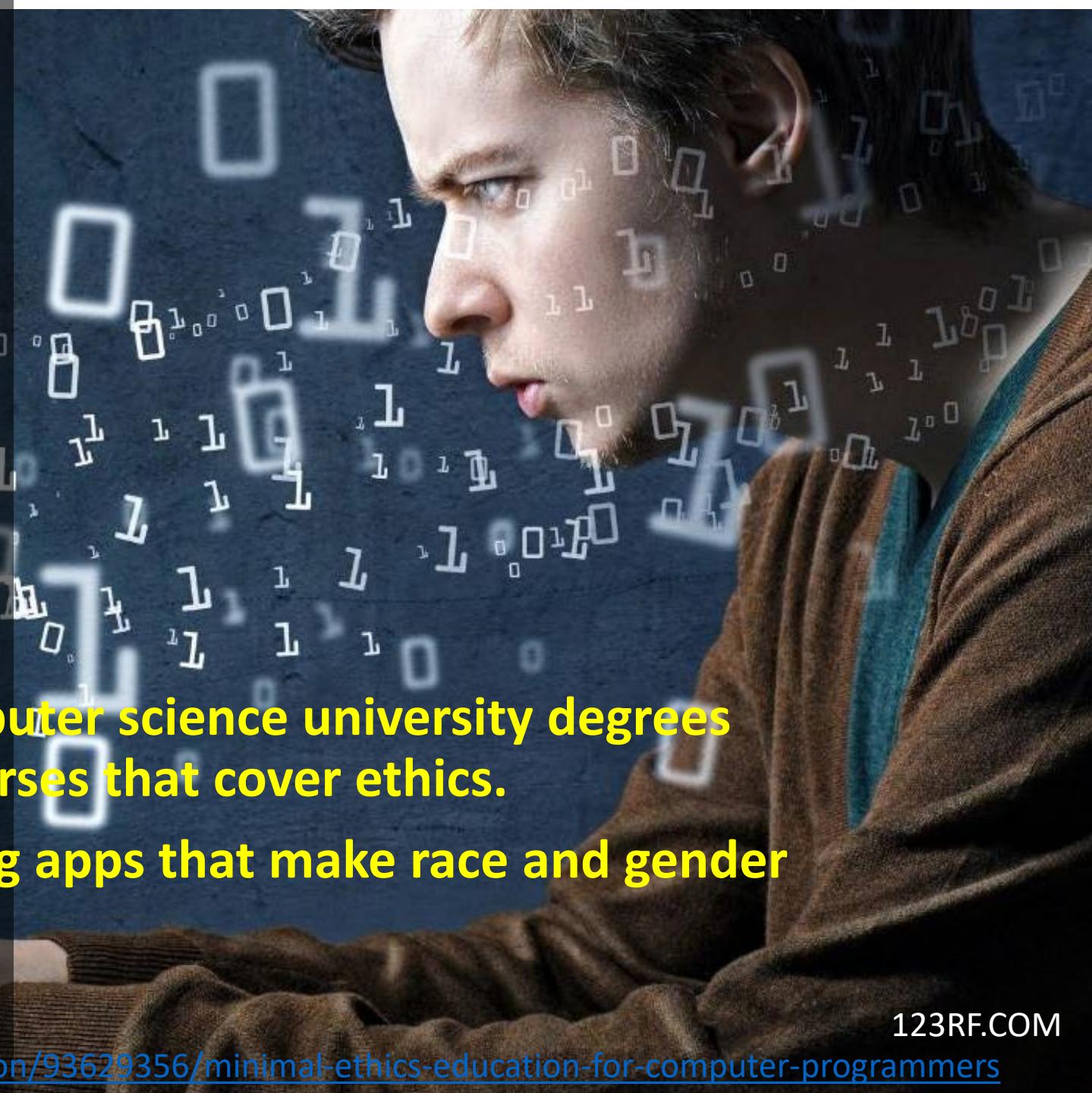
What is ethics?

- *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?

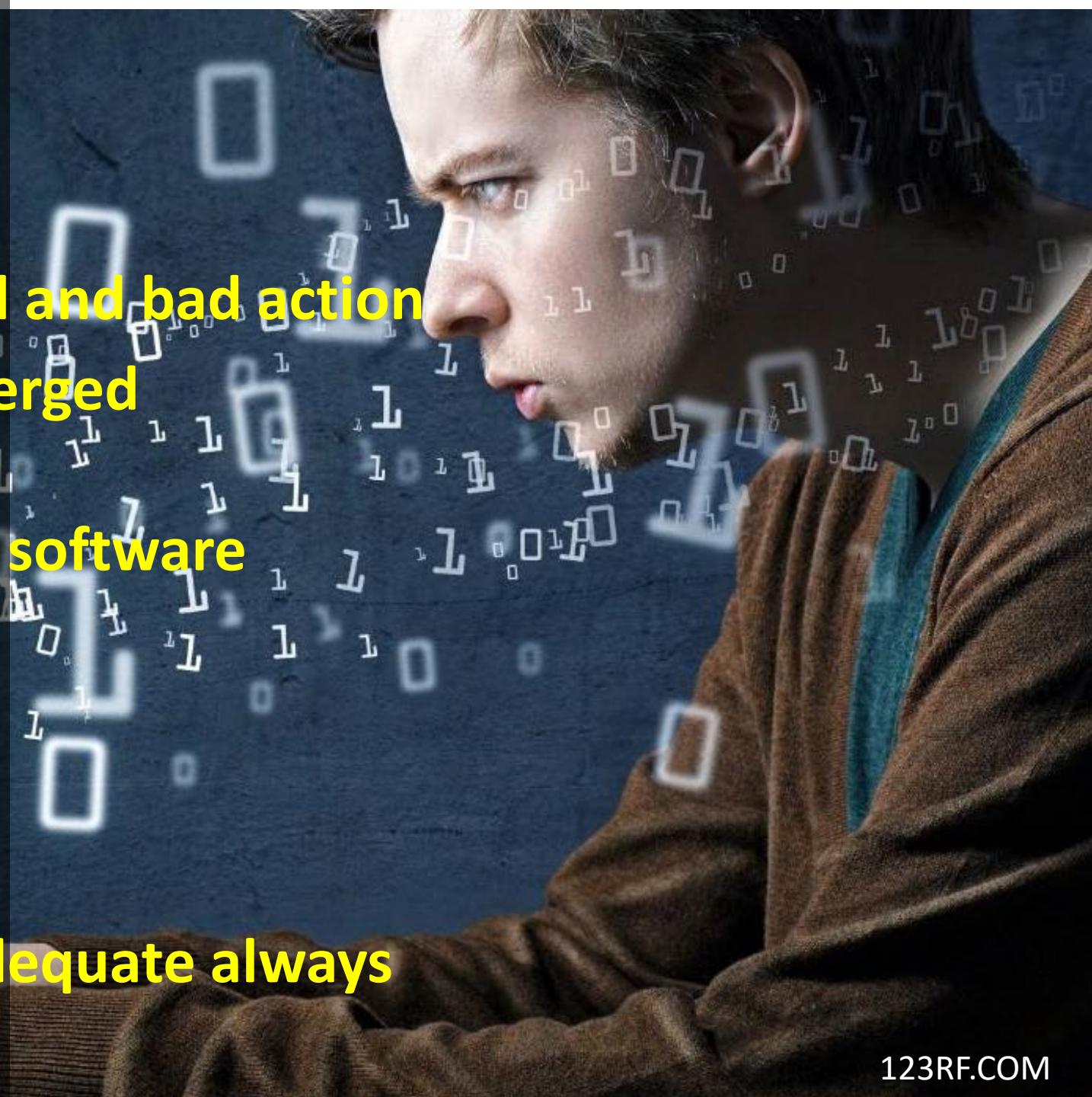
- **Most of New Zealand's computer science university degrees do not include papers or courses that cover ethics.**
- **Some developers are creating apps that make race and gender divides worse*.**





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



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)

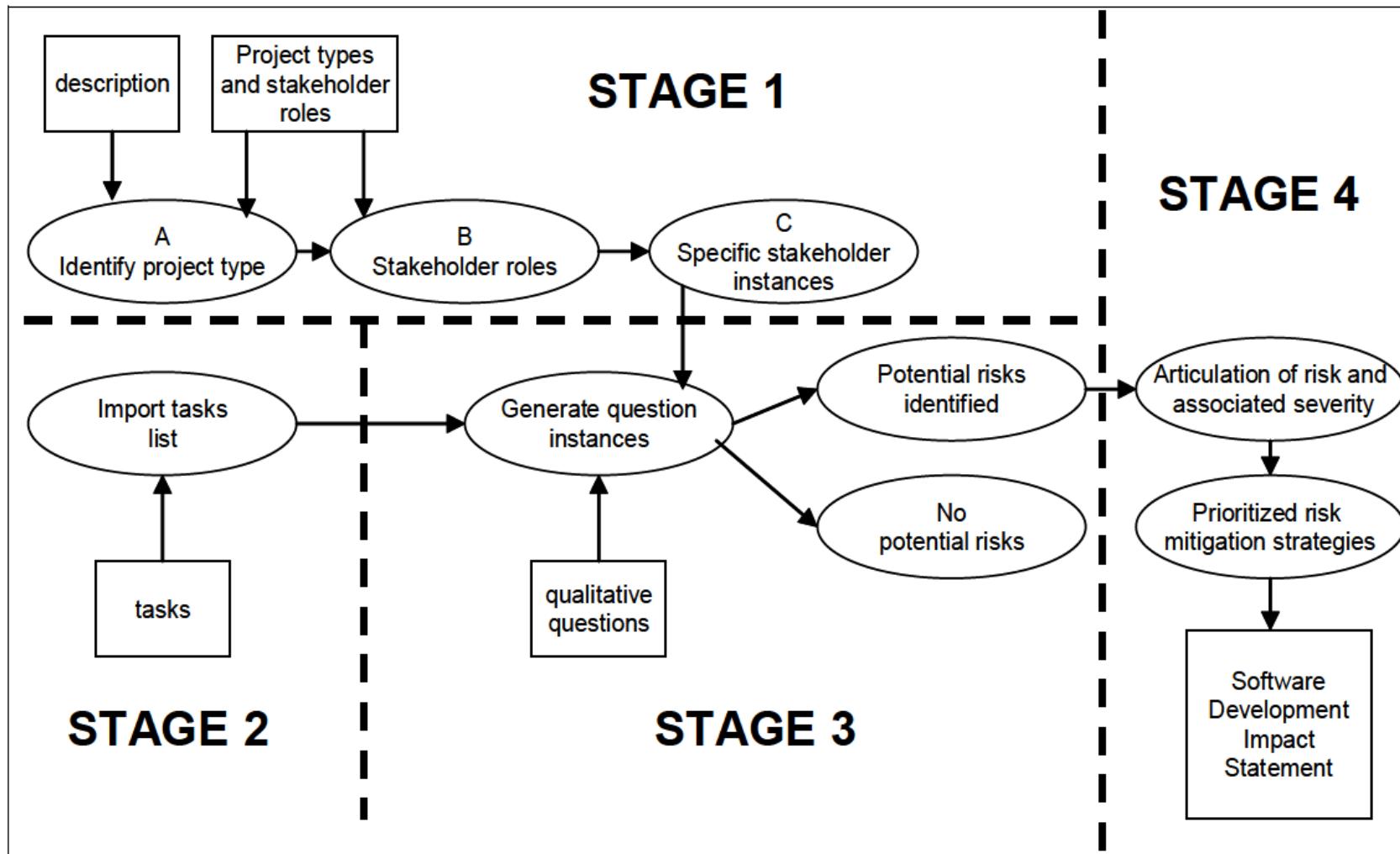
- The TRAION project, then, was entering into sensitive areas. For Maori, “**Identity and worth were found in family and tribal connectedness** [and] ...identity was linked to both ancestry and place” (King, 2003, p. 77).
- As a consequence, Maori people have **known sensitivities about research related 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

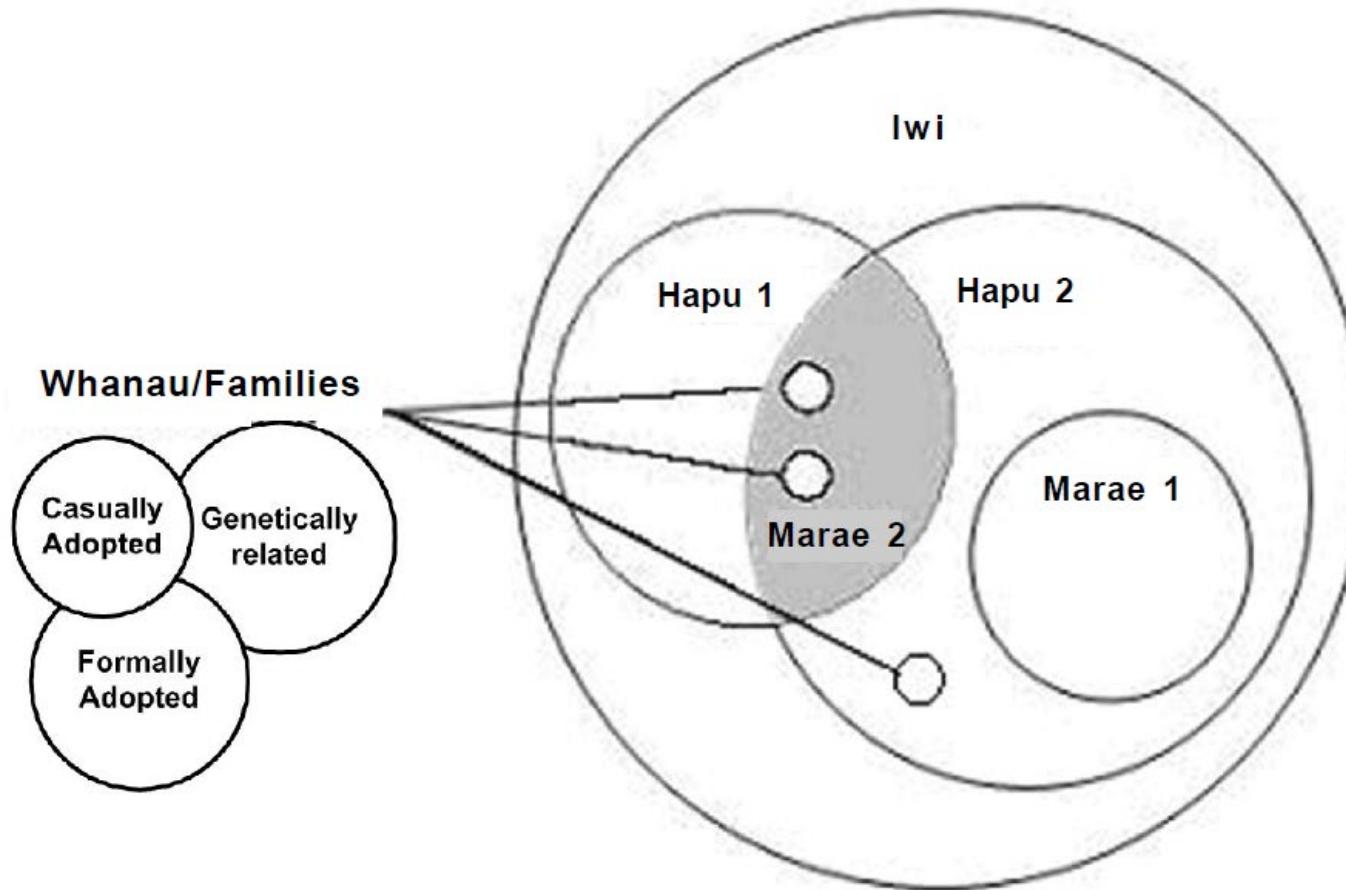
- In the **first cluster of *client developer communication***, the key Maori ethical questions related to **who could legitimately represent the *iwi* as the project client**. ...historical situations in which the authority of the chiefs had been subverted and the authority structures of Maori society 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?

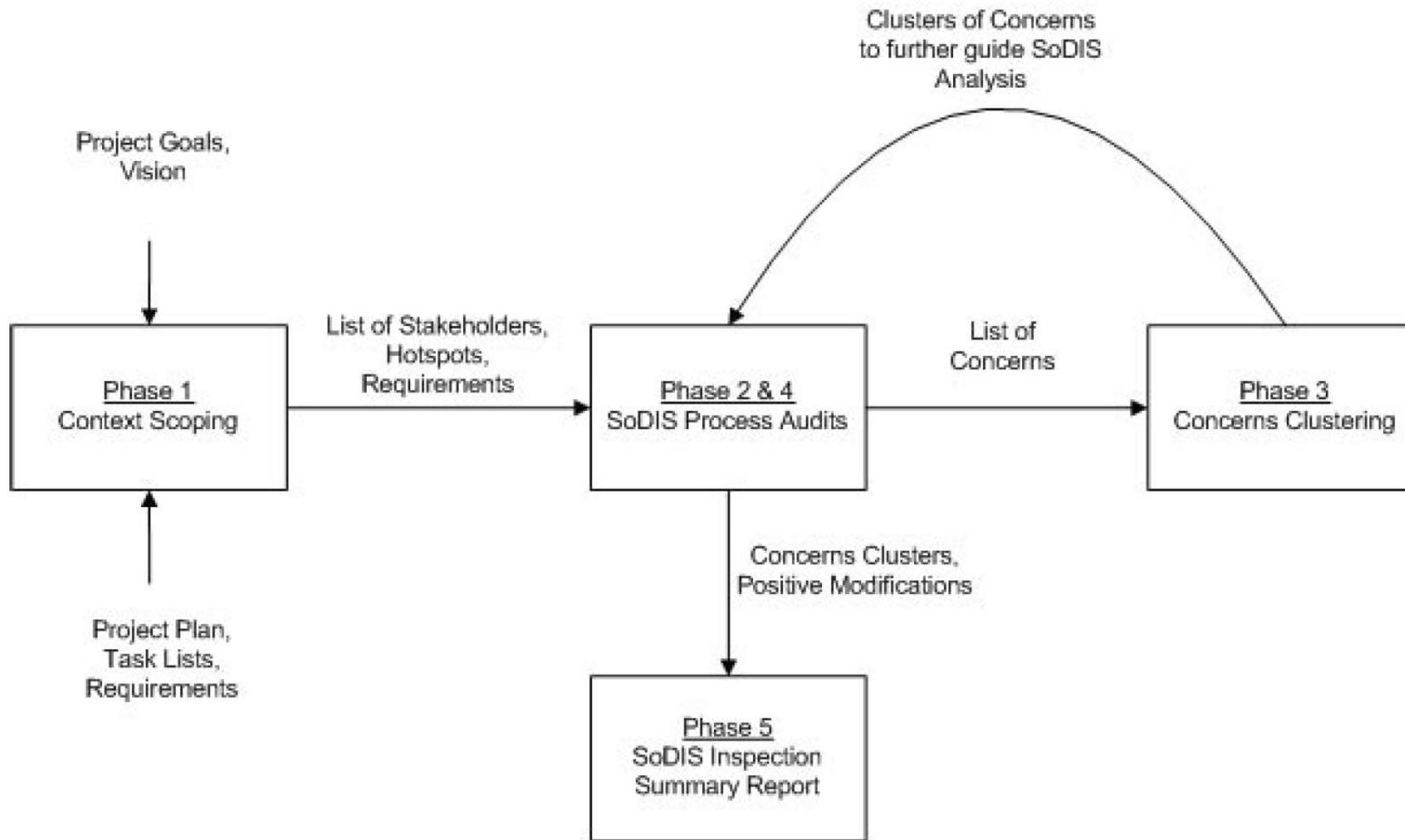


Figure 1. The SoDIS Inspection Process

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

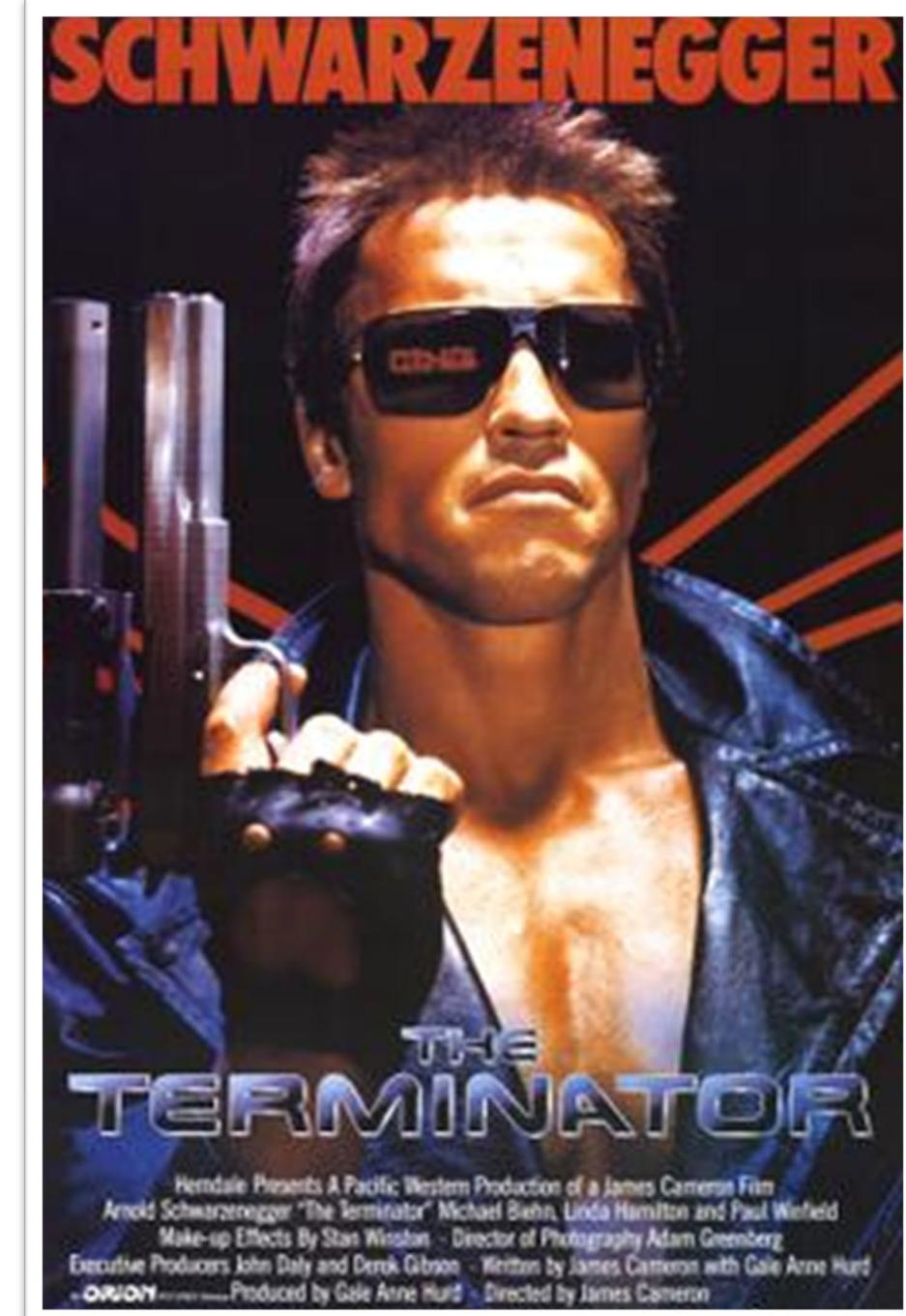
- **5.3 SoDIS Inspection Outcome - Recommendations**
- Where the AUT risk assessment team were able to derive solutions for concerns, or positive modification suggestions, these were presented in the report. For example:
 - Ensuring a managed data conversion process with careful plans for checking data 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?

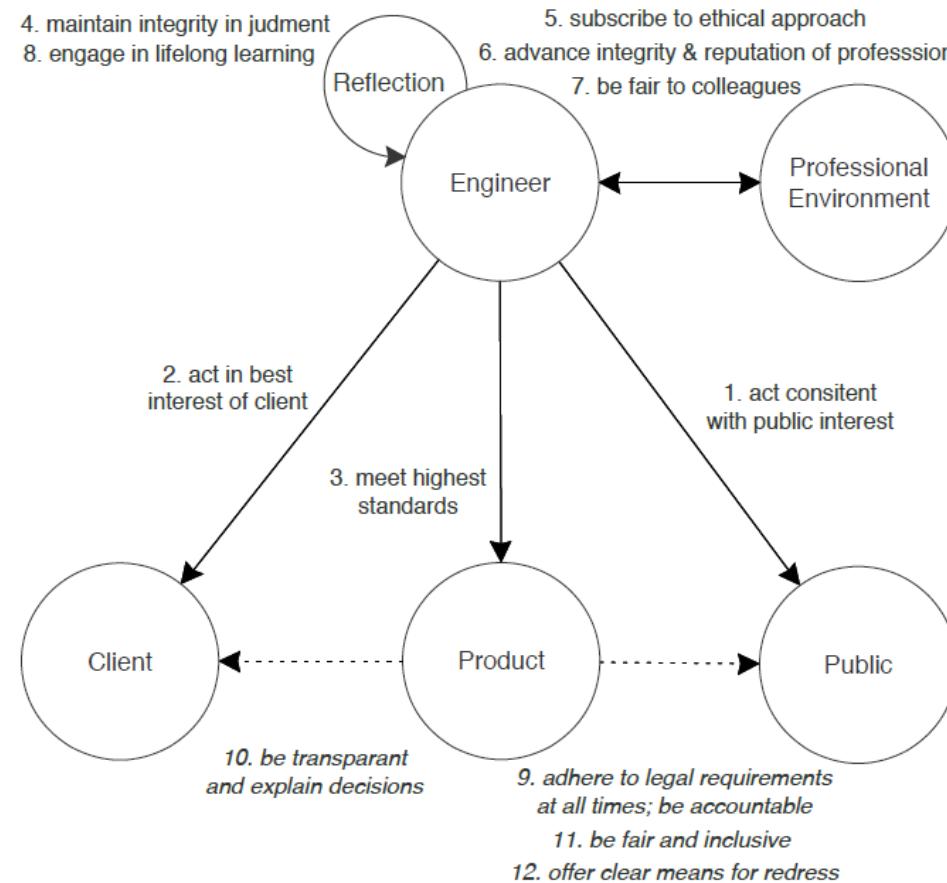


Figure 1: Schematic overview of the Code of Ethics of IEEE/ACM; circles represent actors, arrows represent ethical principles, and dotted arrows suggest new ethical principles for autonomous systems, such as self-adaptive systems.

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

turnitin

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

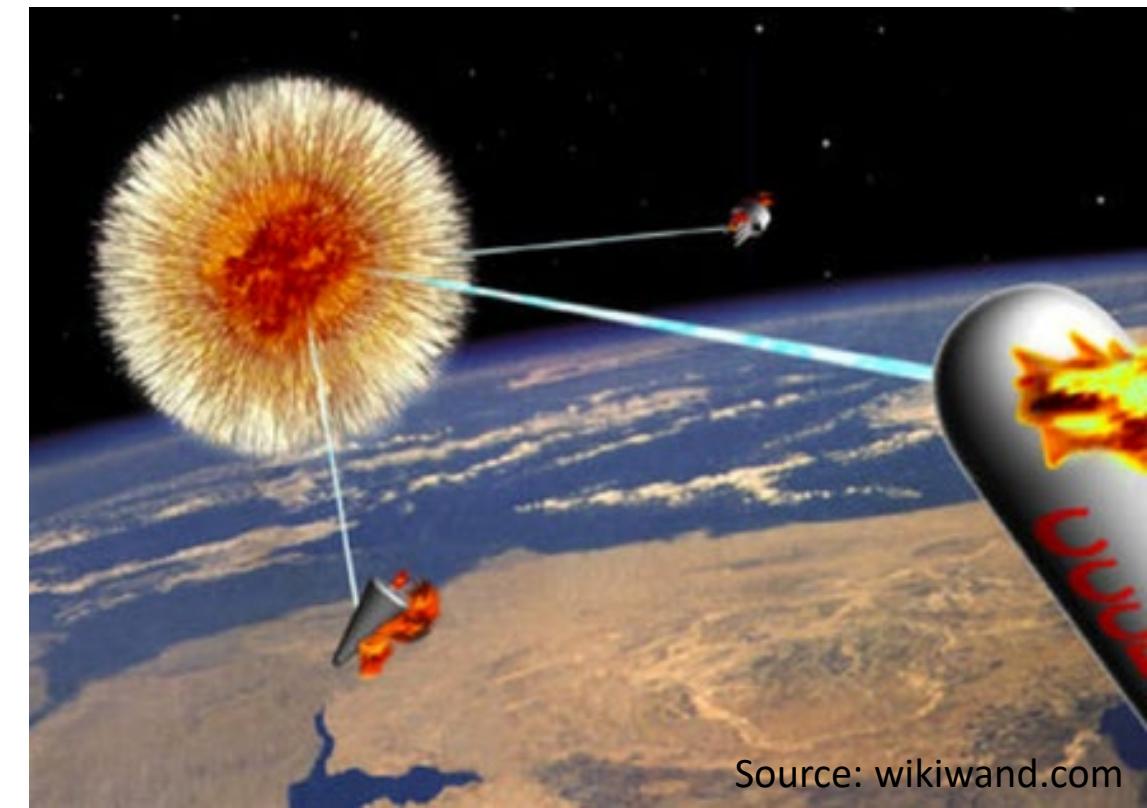
One critic, David Parnas, decided to go public with his concerns about SDI.



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 vacuum 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

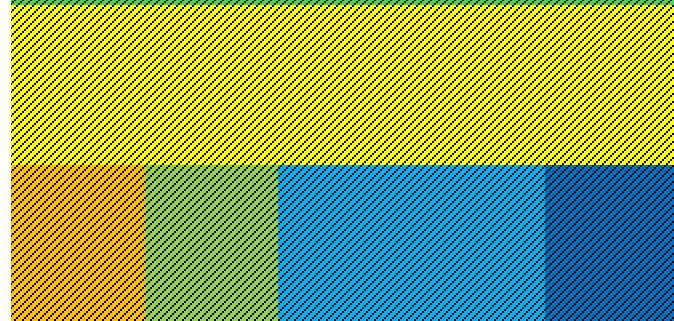
No

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

Review
Questionnaire



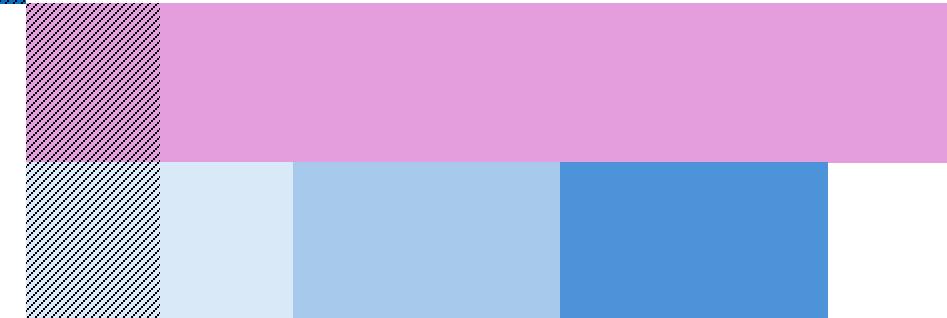
Assgt 1A -
Techstack



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:

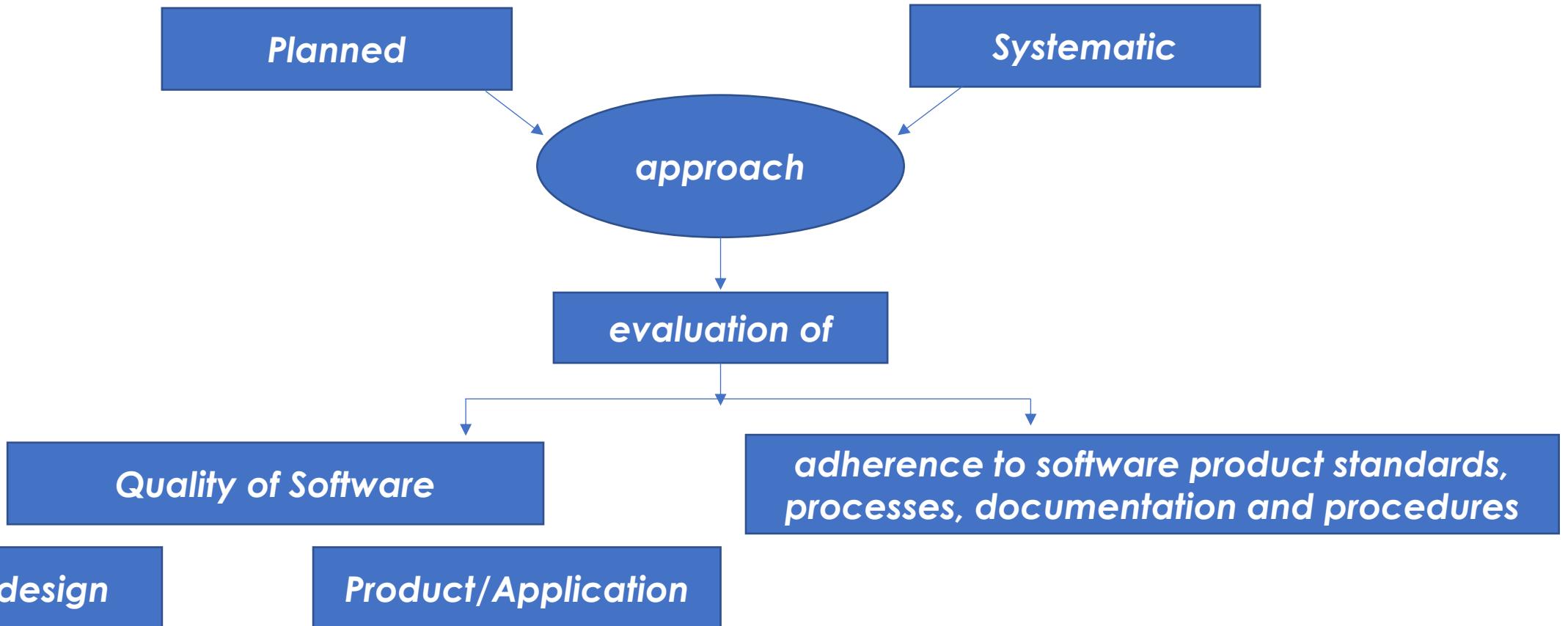
Revision History

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

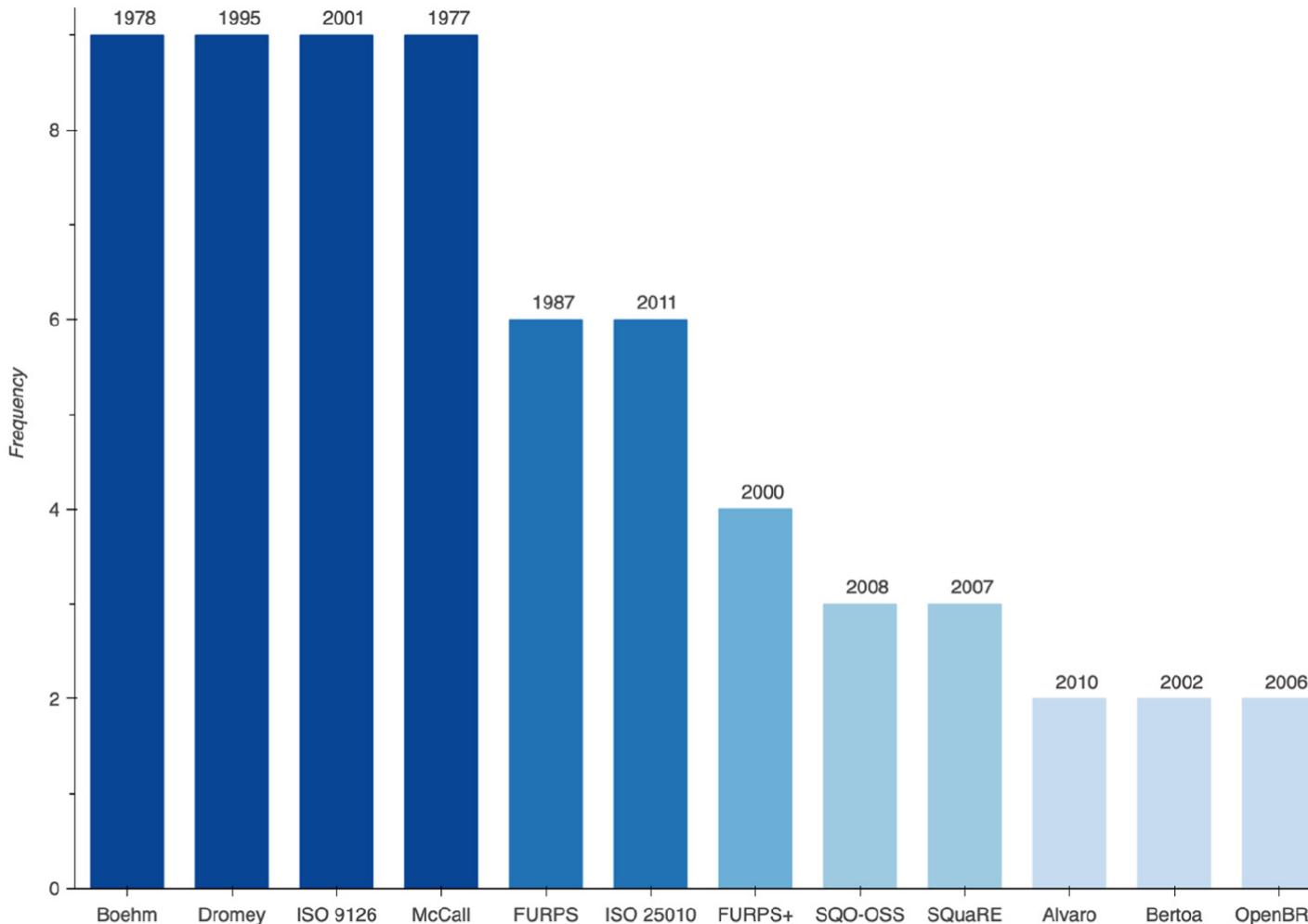


Fig. 1. Frequencies of software quality models with the year each model was proposed.

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

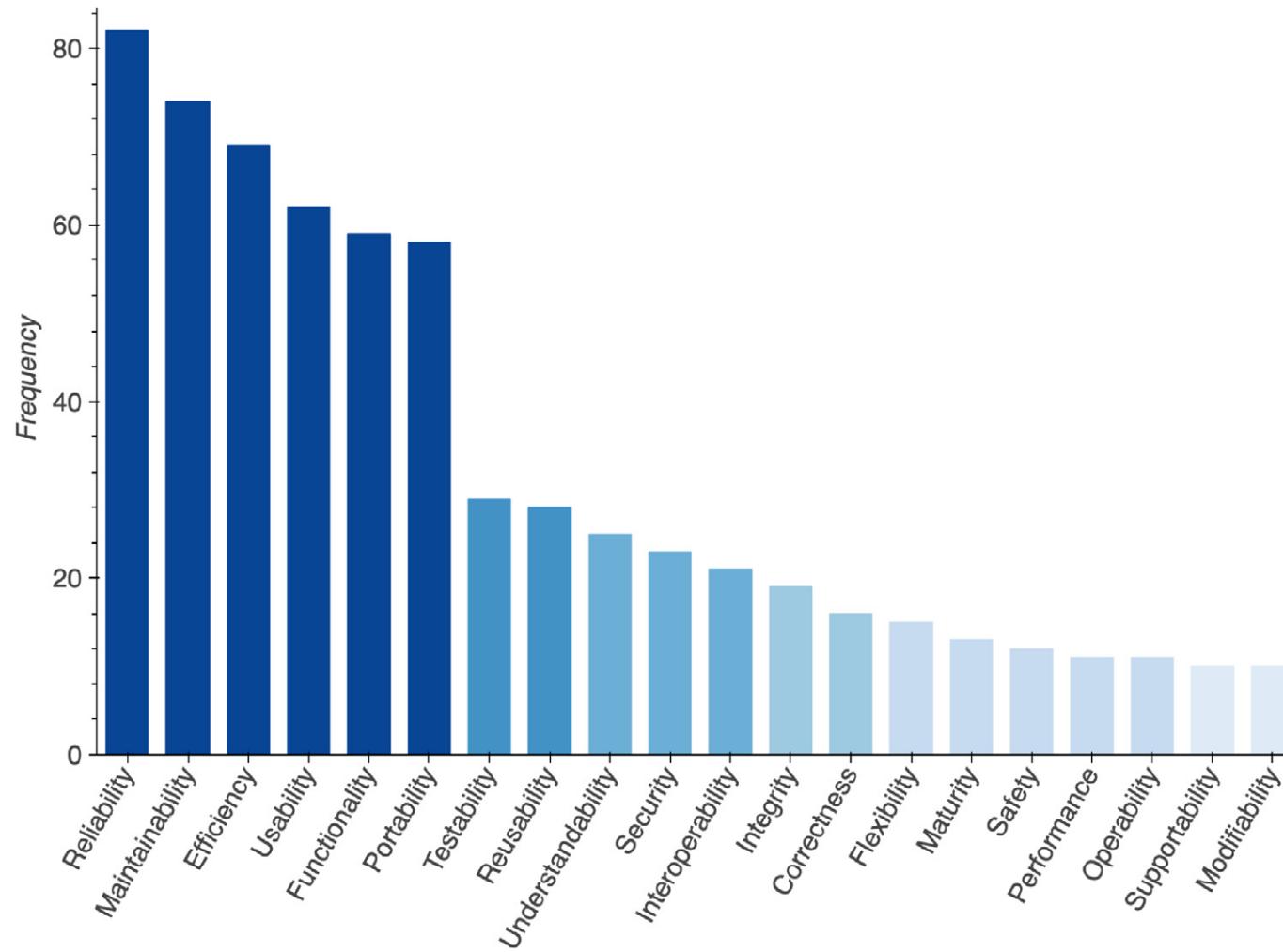


Fig. 2. Frequencies of top twenty software quality characteristics.

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

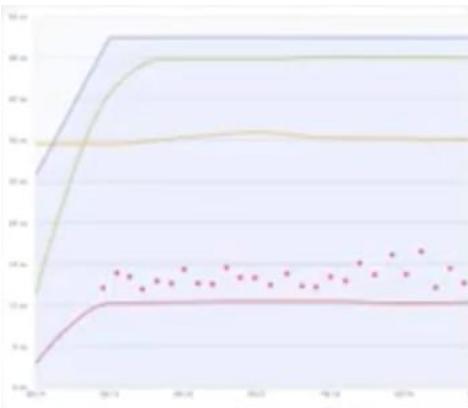
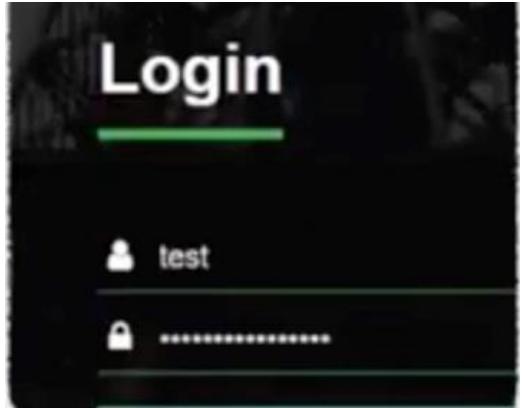
A strong test suite is like a warm blanket – feel ok to release, experiment

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
```

```

    .withUserEmail(registeredUser.getEmail())
    .withRegisteredUser(registeredUser)
    .withUserPasswordHashed(userRegistrationRequest.getPassword())
    .withUserStatus(UserStatus.ACTIVE);
}

@Given("a user registration")
public void aUserRegistration() {
    UserBuilder userBuilder = new UserBuilder();
    userBuilder.withUsername("user");
    userBuilder.withEmail("user@example.com");
    userBuilder.withPassword("password");
    userBuilder.withUserStatus(UserStatus.ACTIVE);
    User user = userBuilder.build();
    when("the user registers")
        .given("a user registration")
        .when("the user enters their details")
        .and("the user enters their password")
        .and("the user agrees to the terms and conditions")
        .and("the user clicks the register button")
        .then("the user is registered")
        .and("the user receives a confirmation email")
        .and("the user's account is active");
}

```

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

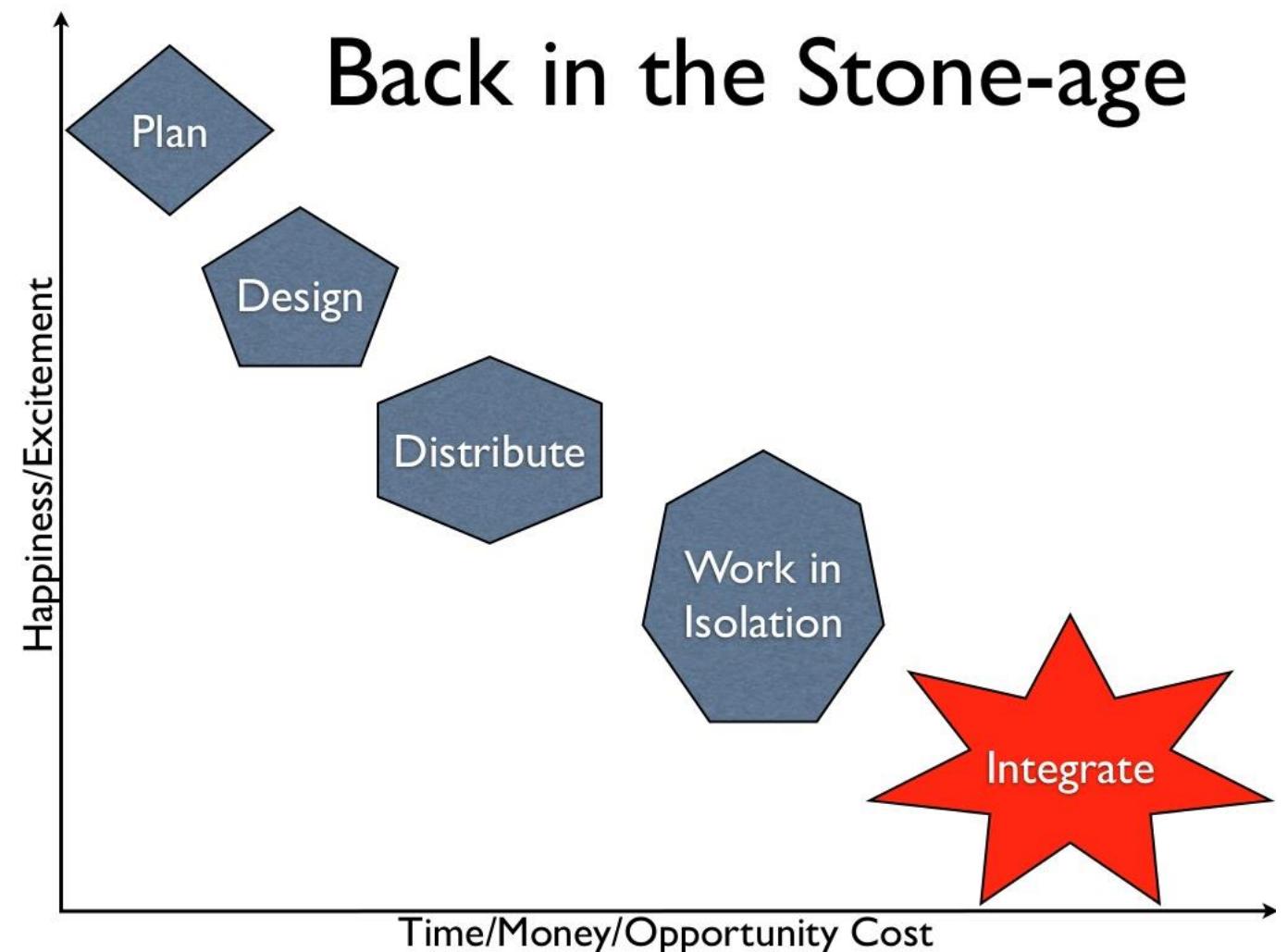


COORDINATION

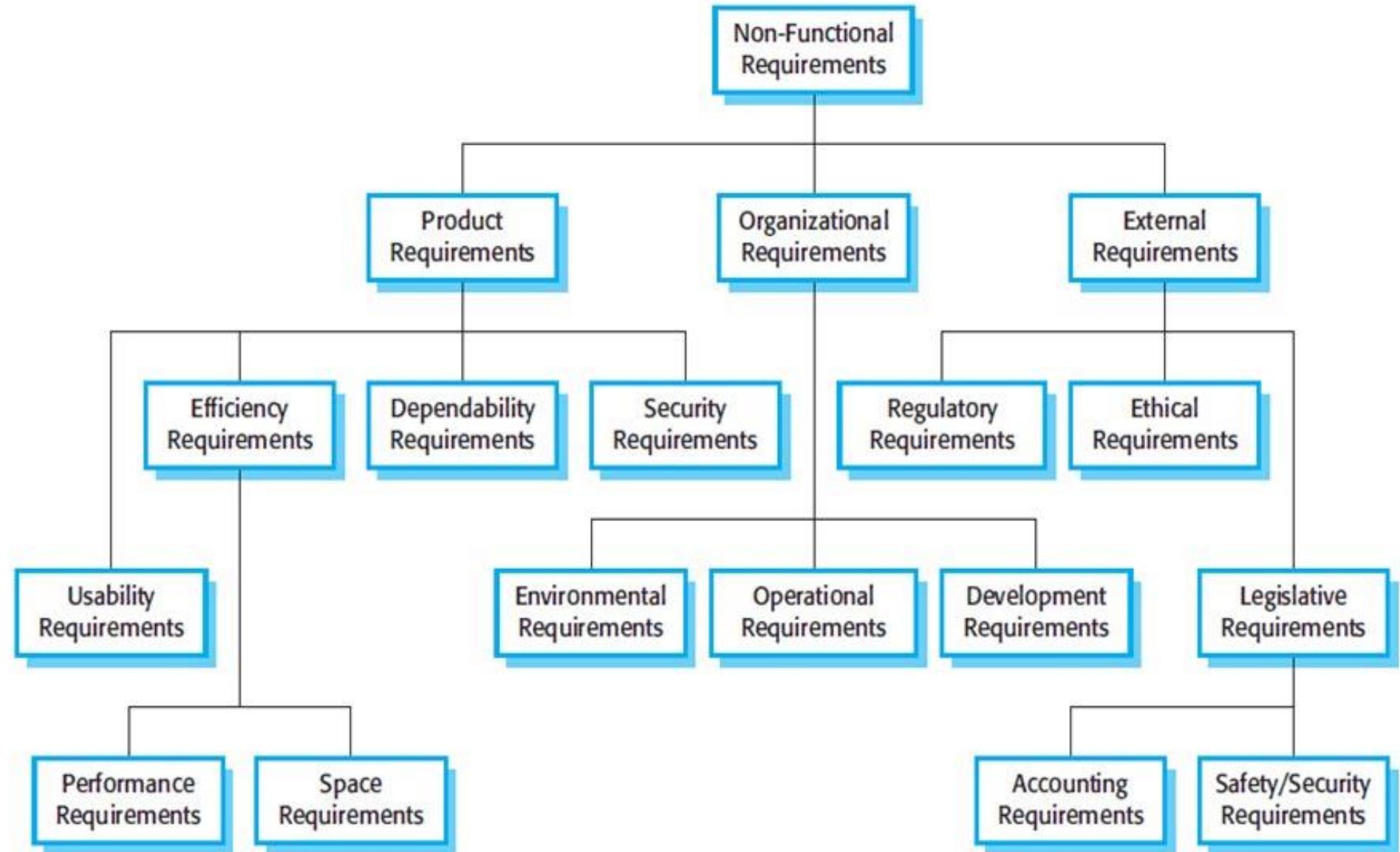
It works on my machine

I didn't realise you were
doing it that way

I didn't understand what
you were doing



Non-Functional Requirements or Quality Requirements

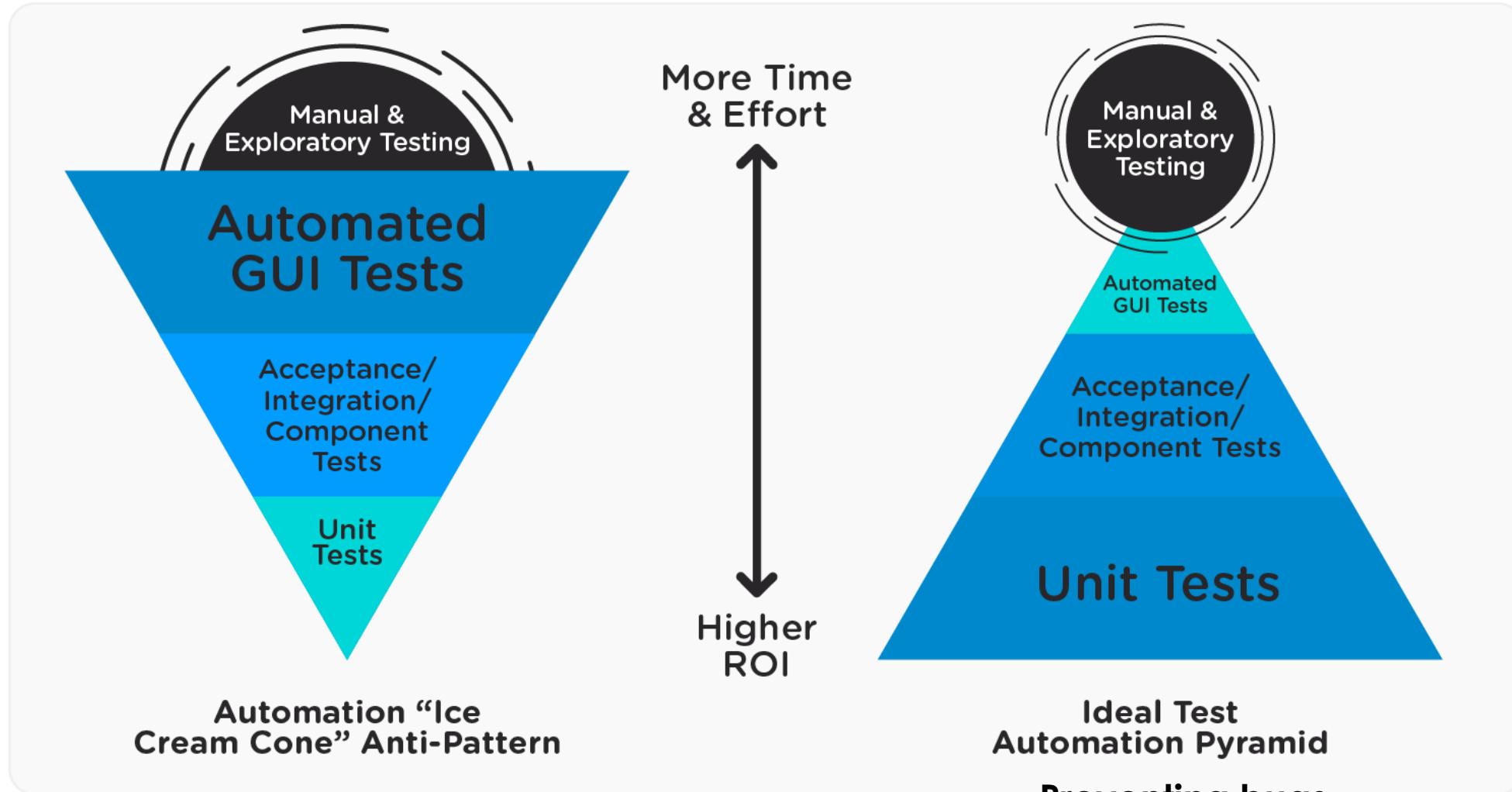


Non-Functional Requirements or Quality Requirements

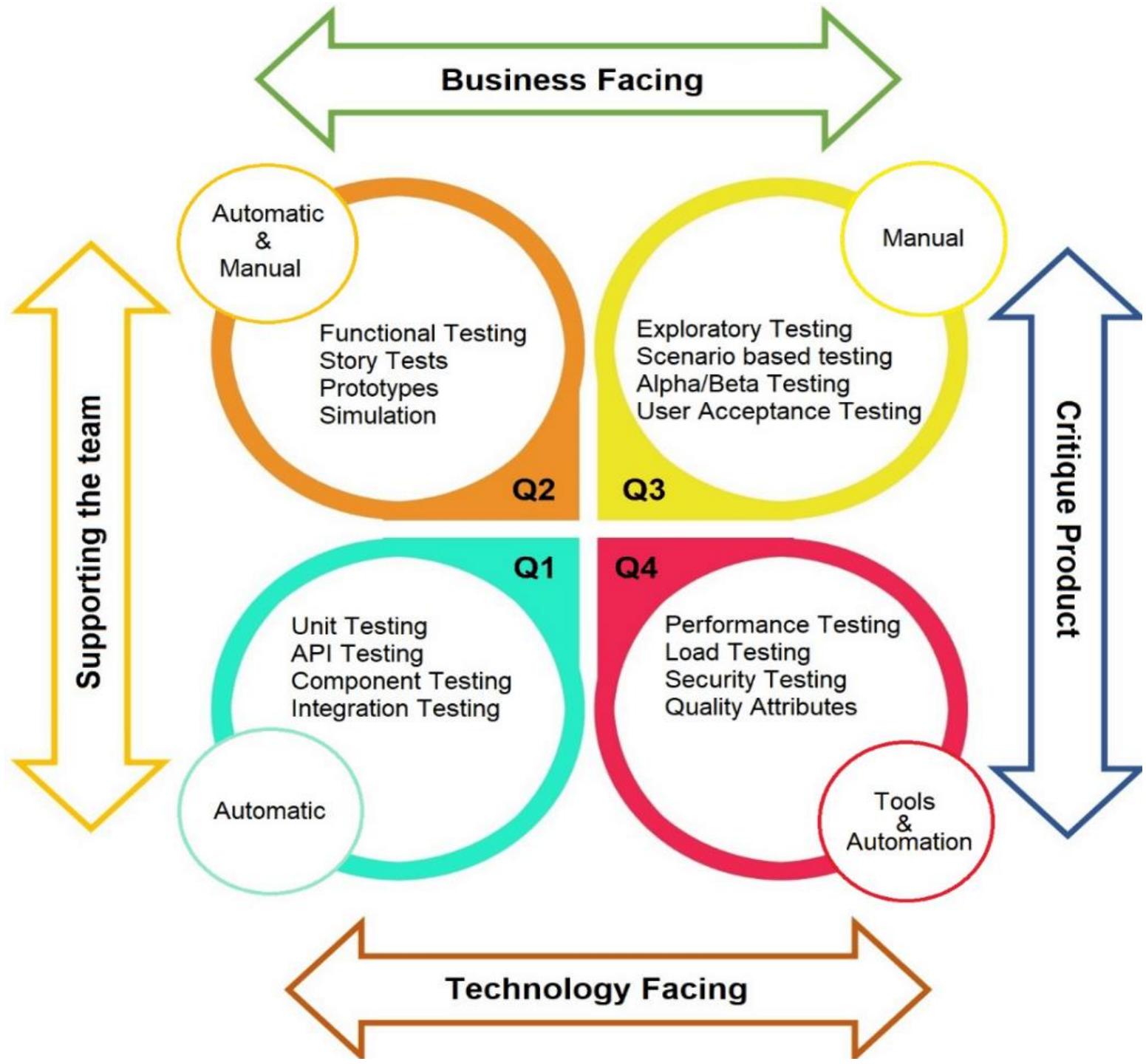
Functional Requirements	Non Functional Requirements
<ul style="list-style-type: none">• Product features	<ul style="list-style-type: none">• Product property
<ul style="list-style-type: none">• Describe the actions with which the user work is concerned	<ul style="list-style-type: none">• Describe the 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))



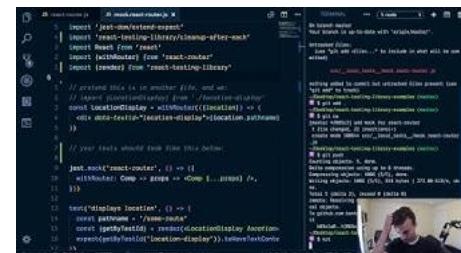
<https://www.youtube.com/watch?v=7r4xVDI2vho&t=30s>

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>

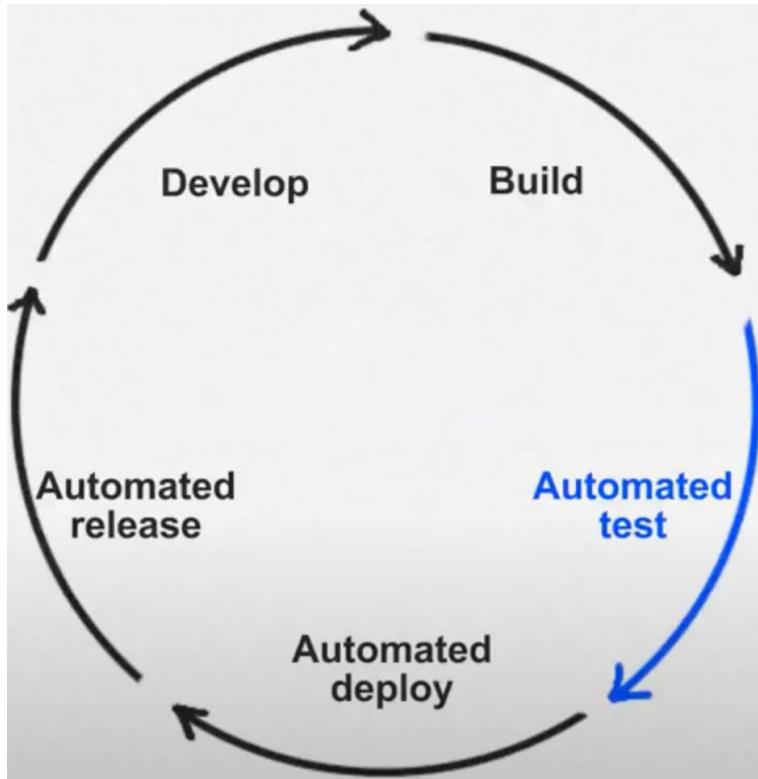
Complete Guide to Component testing with Jest for beginners. Crash course on Jest and mocking

<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)



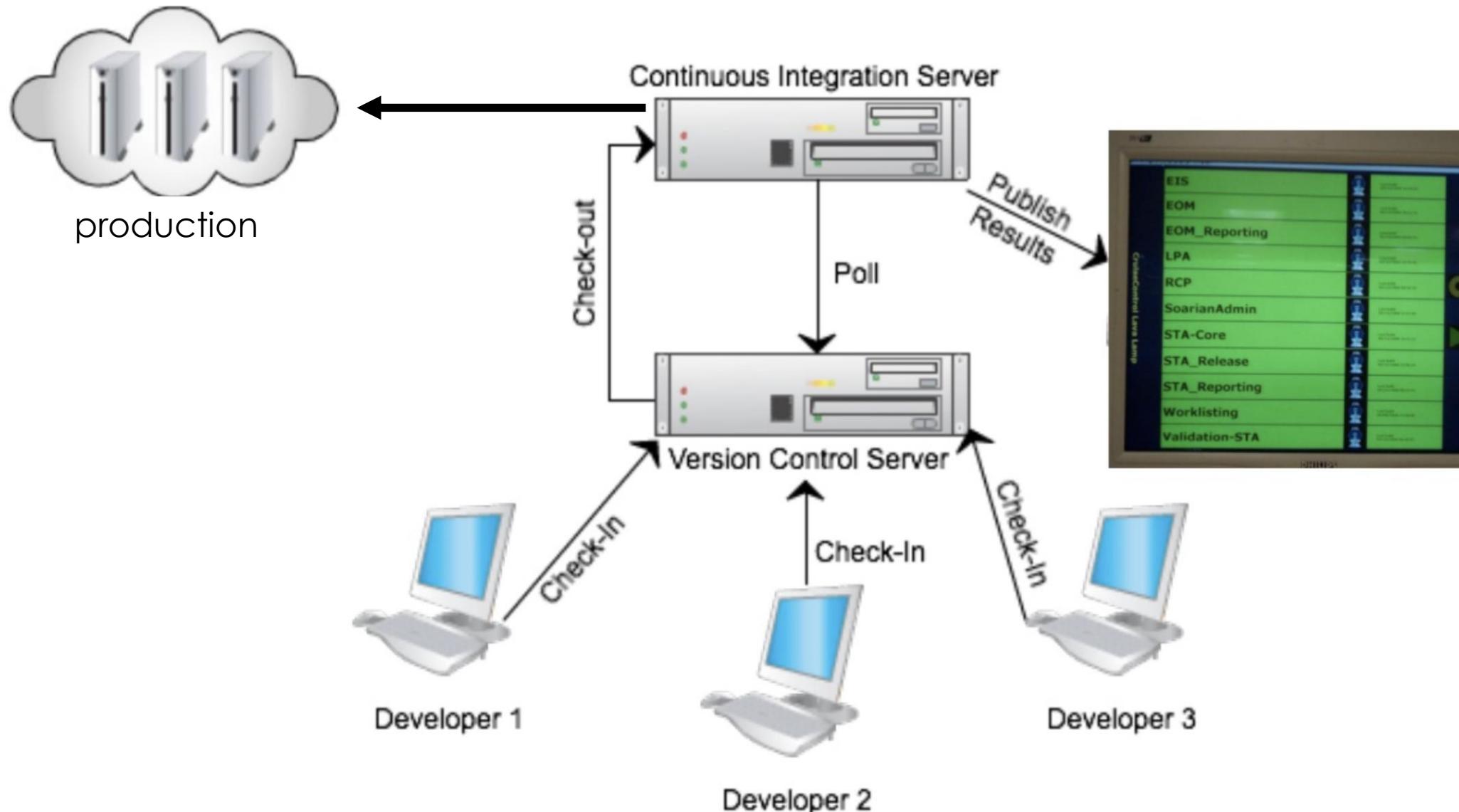
Continuous Delivery and DevOps ways of working rely on test automation to make rapid deployment possible

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?

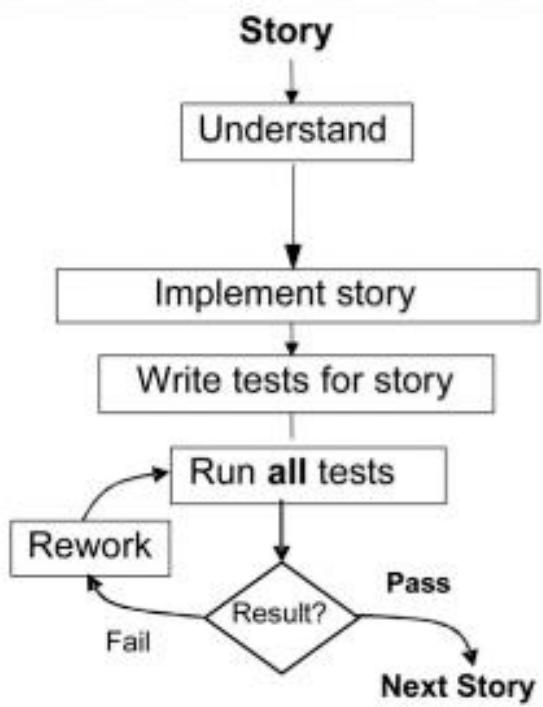
When Test Driven Development Goes Wrong

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.

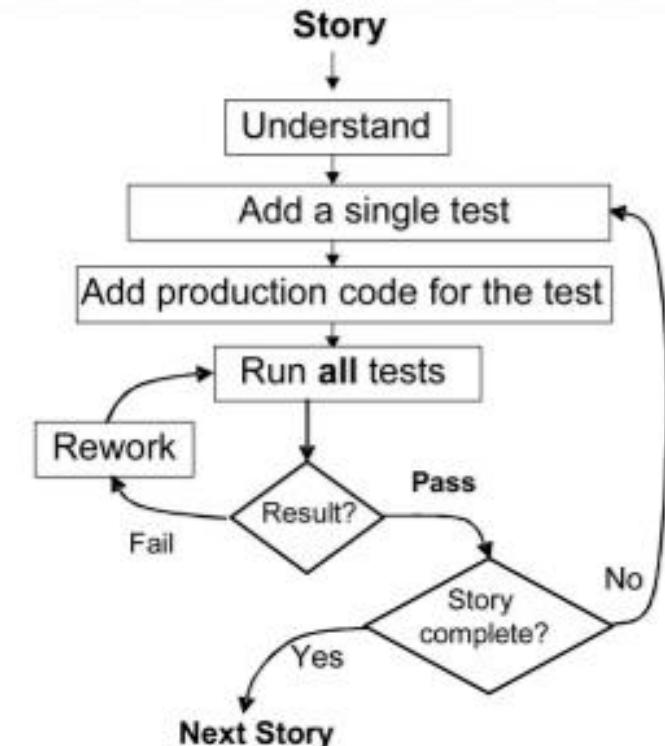
The interesting thing is that when it goes wrong, it may be at its most valuable. TDD is a cornerstone of Continuous Delivery, BDD and DevOps. Using it to give us valuable, efficient feedback on the quality of our designs is at the heart of its value but is often missed by people who are new to it. Dave 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

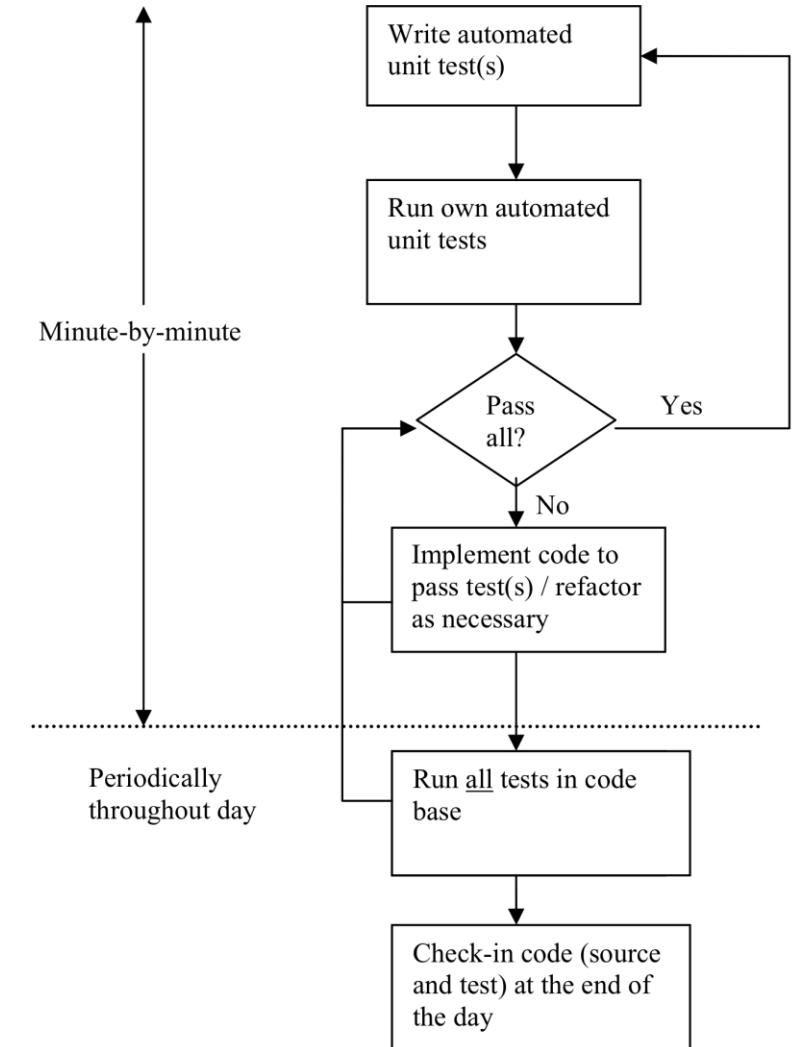
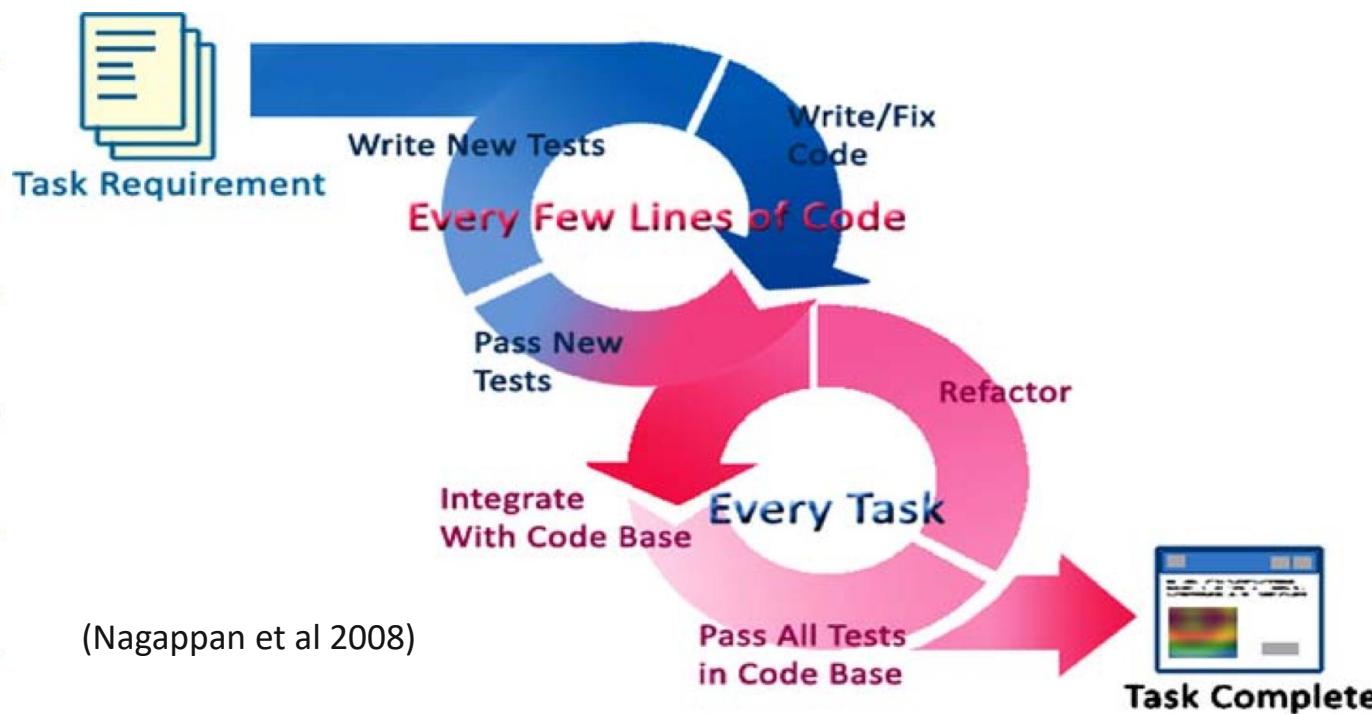


Figure 1: Test-Driven Development

(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

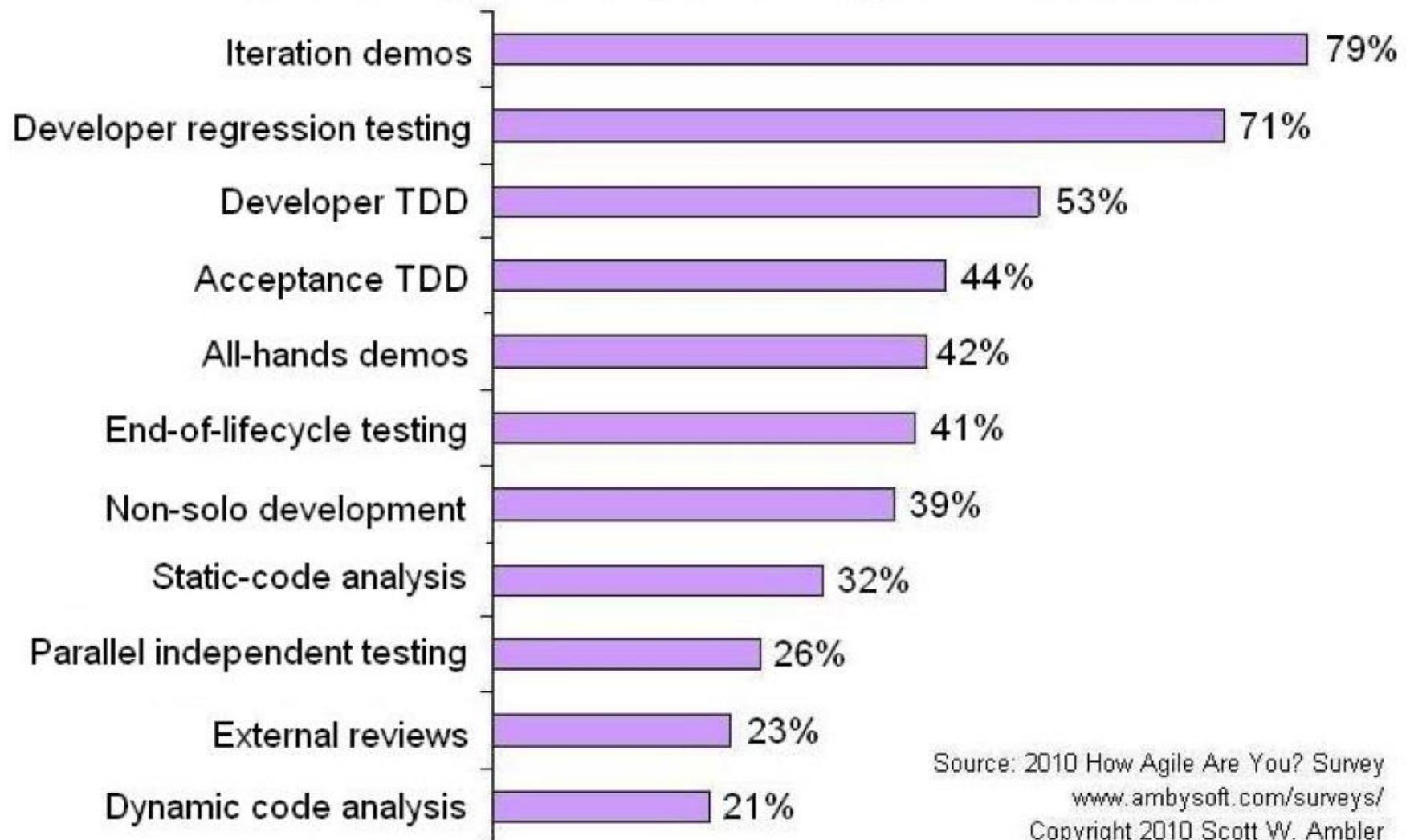
Sometimes we need a test database or **staging database** so we can do testing with data that isn't in the production database, and set up certain initial database conditions

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 Siniasto¹ and Pekka Abrahamsson²

¹ F-Secure Oyj,
Elektroniikkatie 3, FIN-90570 Oulu, Finland
Maria.Siniasto@f-secure.com
² VTT Technical Research Centre of Finland,
P.O. Box 1100, FIN-90571 Oulu, 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 [2] 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)@cs.shef.ac.uk

Abstract— Recently there has been a focus on test first and test driven development; several empirical studies have tried to assess the advantage that these methods give over testing after development. The results have been mixed. In this paper we investigate nine teams who tested during coding to examine the effect it had on the external quality of their code. Of the top three performing teams two used a documented testing strategy and the other an ad-hoc approach to testing. We conclude that their success appears to be related to testing culture where the teams proactively test rather than carry out only what is required in a mechanical fashion.

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 after 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?

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

Test first (TF) is an established development technique which is essentially the same as *test driven development* (TDD). In both cases the aim is to write tests before writing the functional code. This should in theory aid development as the tests form the basis of the specification, design and functional tests, whereas testing after the code is written is regarded as only a testing technique [1; 2]. Whilst TF and TDD are well defined, the traditional or *test last* technique is interpreted differently by the studies in the literature investigating this phenomenon. The studies' definitions of *test last* can be divided into roughly four categories, which we will refer to as TL 1 to 4:

TL-1: Unspecified traditional method. Most experiments provided at least a few hints as to the method that the non-TF teams should follow, however some did not [3-5]. The following statement was typical of the broad definitions that these papers used: "the control group which followed the traditional process" ([5], p132). As no further discussion was made about the process followed we can't make generalizations about what affected their performance.

TL-2: Specified traditional method. This method is typified by manual or ad-hoc testing and was used in three studies [6-8]. The method was defined thus: "no automated tests were written and the project was developed in traditional mode with a large up-front design and manual testing after the software was implemented" ([7], p.1) and "[the specified traditional method] involved a large up-front design and manual testing after the software was implemented" ([8], p.1).

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

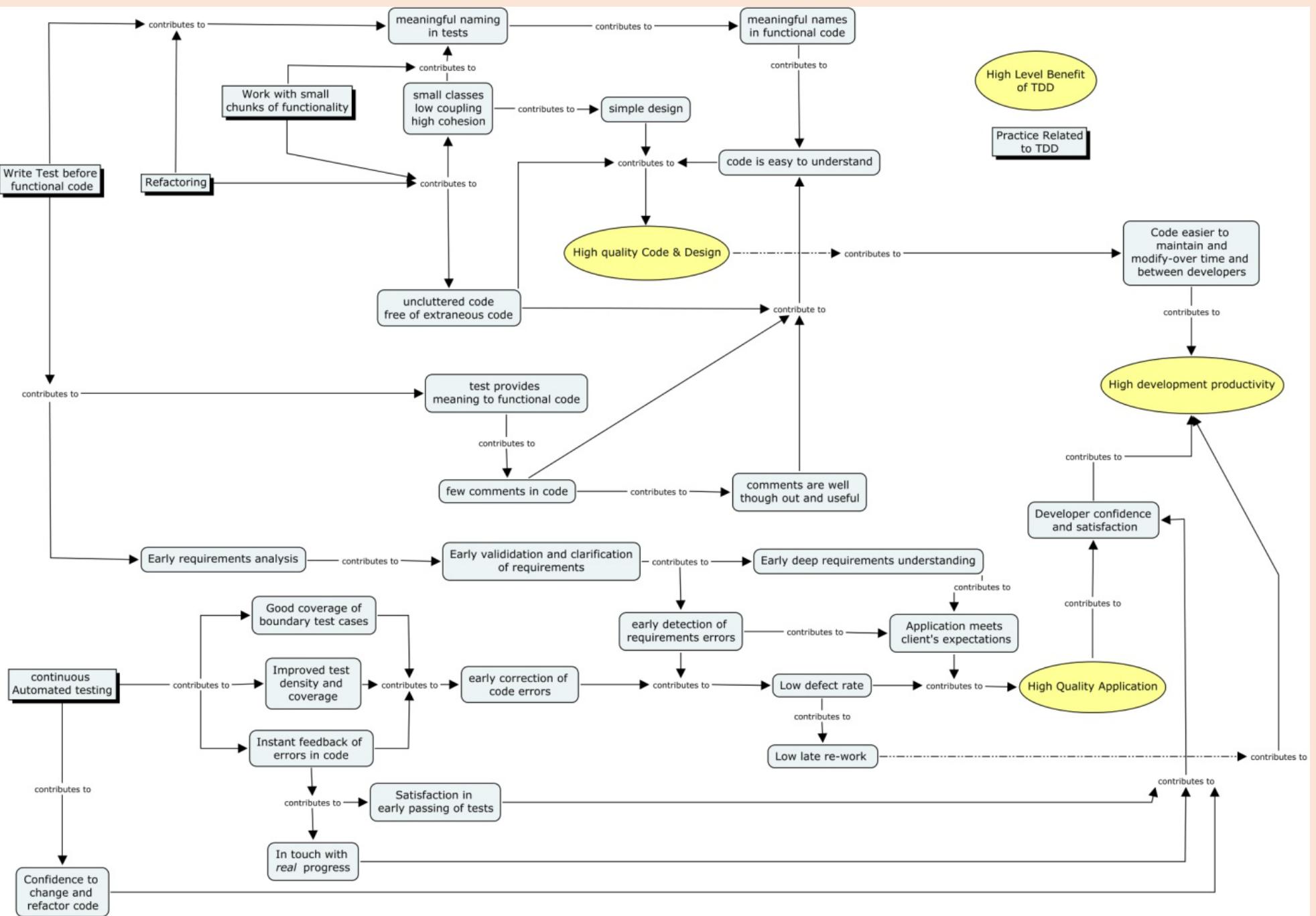


Figure 2. Causal network of TDD benefits and contributing factors.

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!

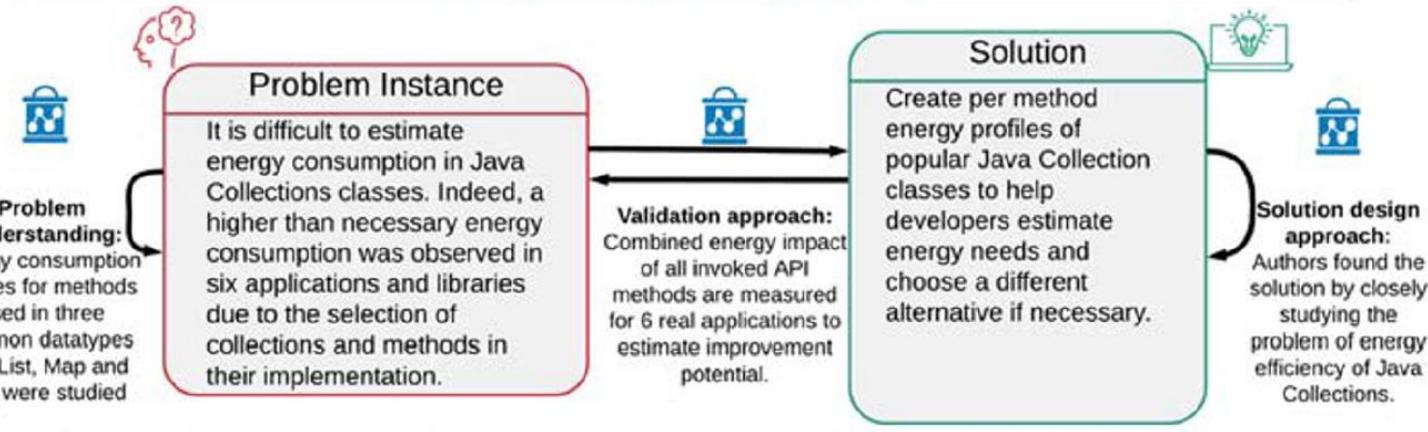


dfucci Davide Fucci

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

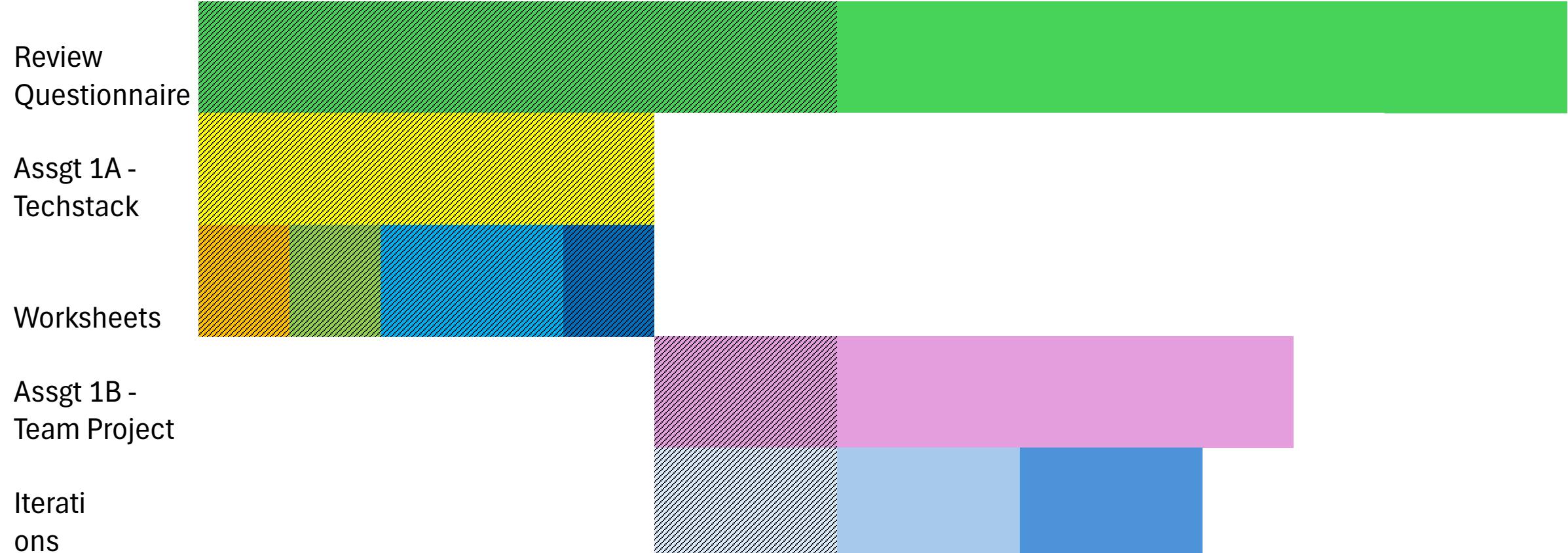
Week 8



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>

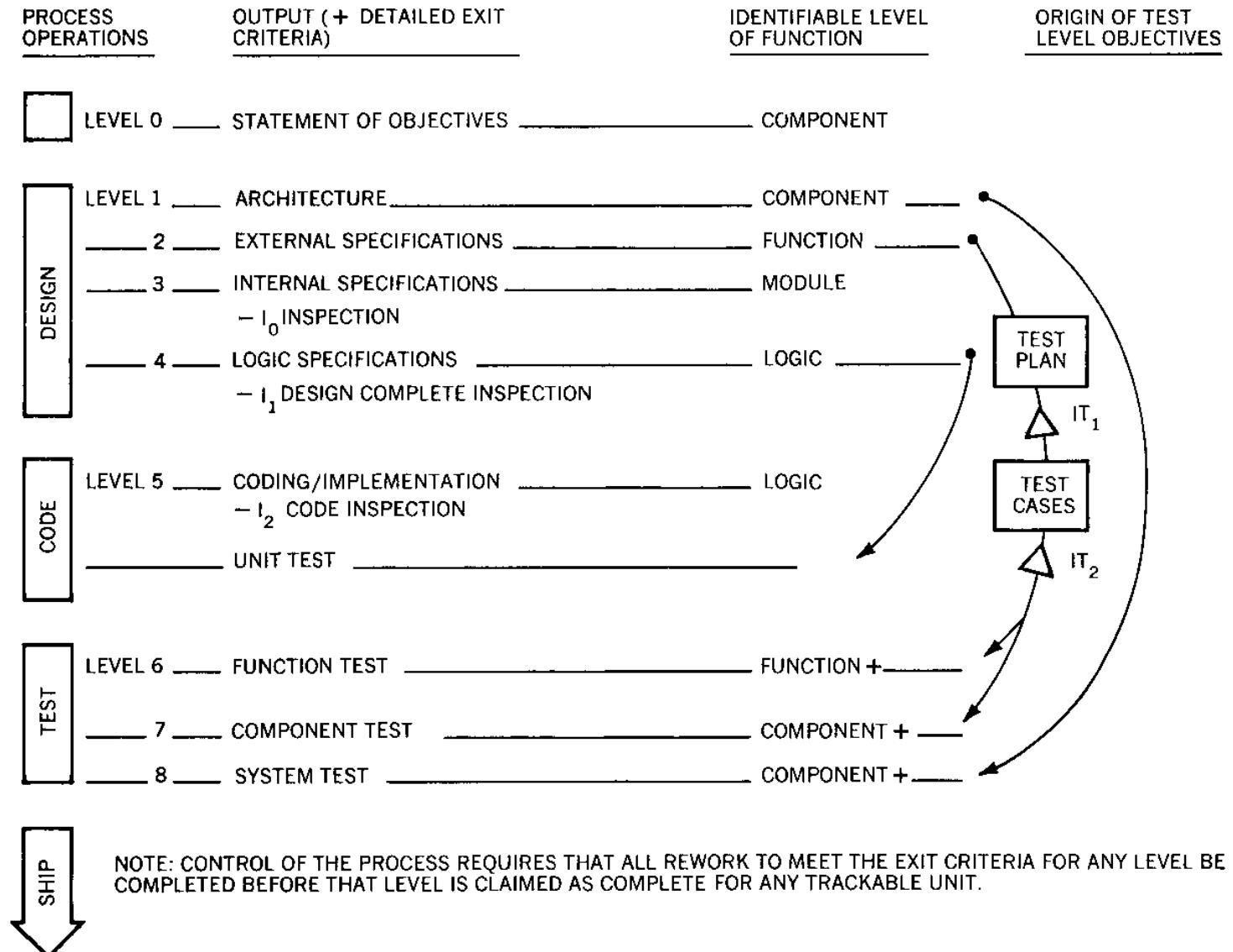
https://link.springer.com/chapter/10.1007/978-3-642-59412-0_35

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>

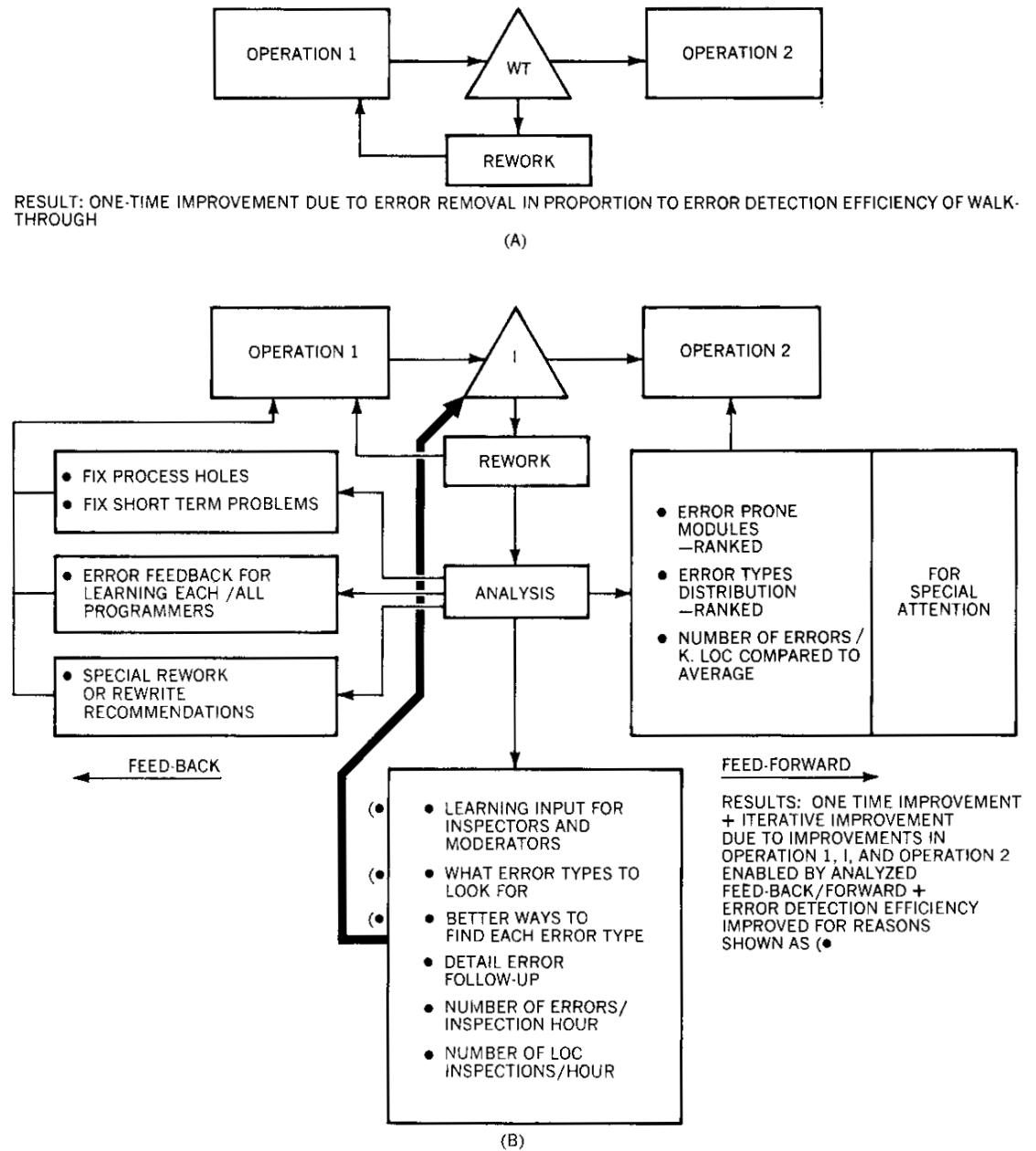
Fagan Inspections

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.



Mobbing just extends the benefits of pair programming with more people working on a coding problem?



The team took “Mob Programming” to the next level.



Agile leadership in mob programm

Mob Programming



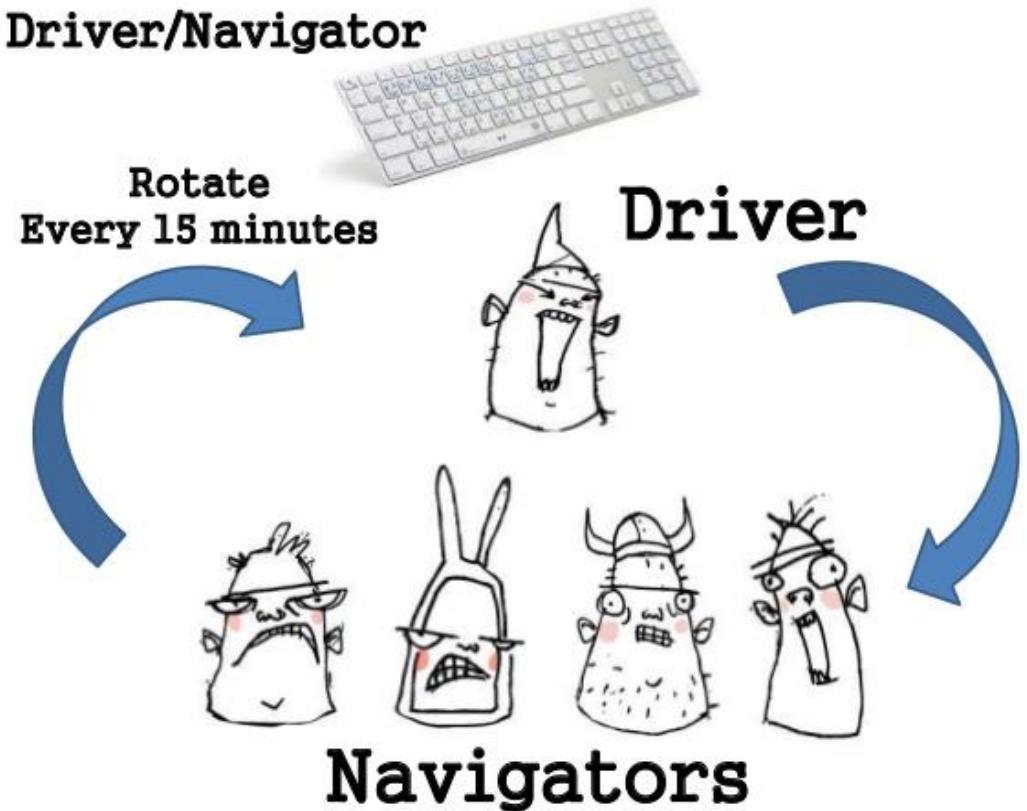
Woody Zuill began popularizing it again from **2013**

Mob Programming

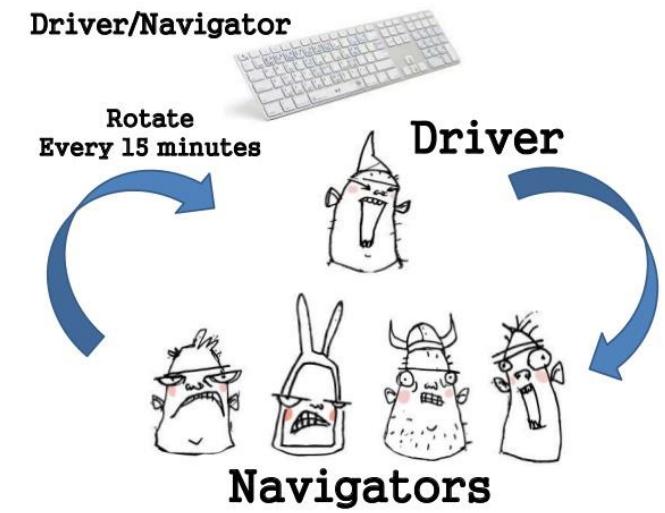
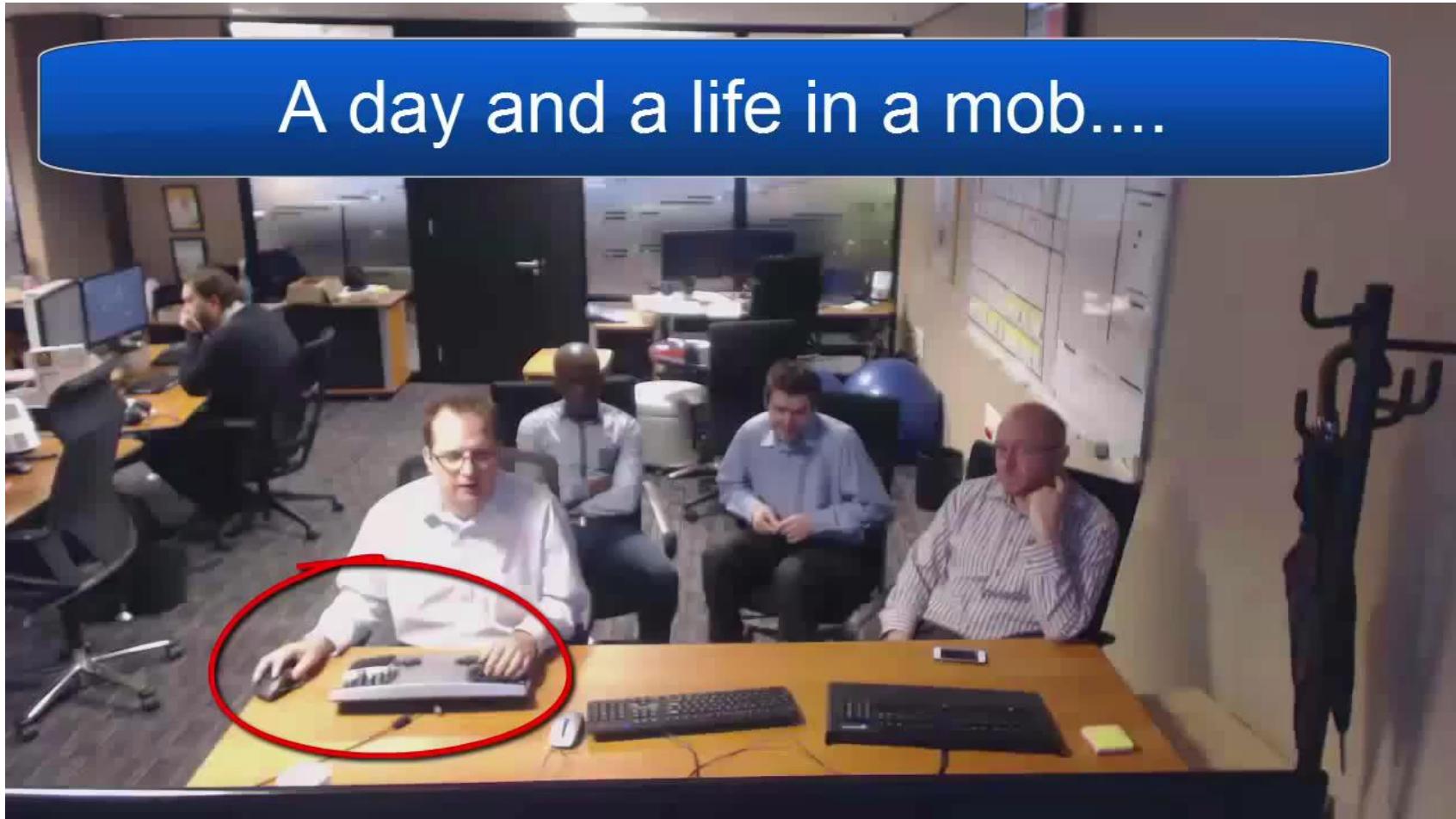
All the brilliant minds working on
the same thing...
at the same time...
in the same space...
on the same computer...

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

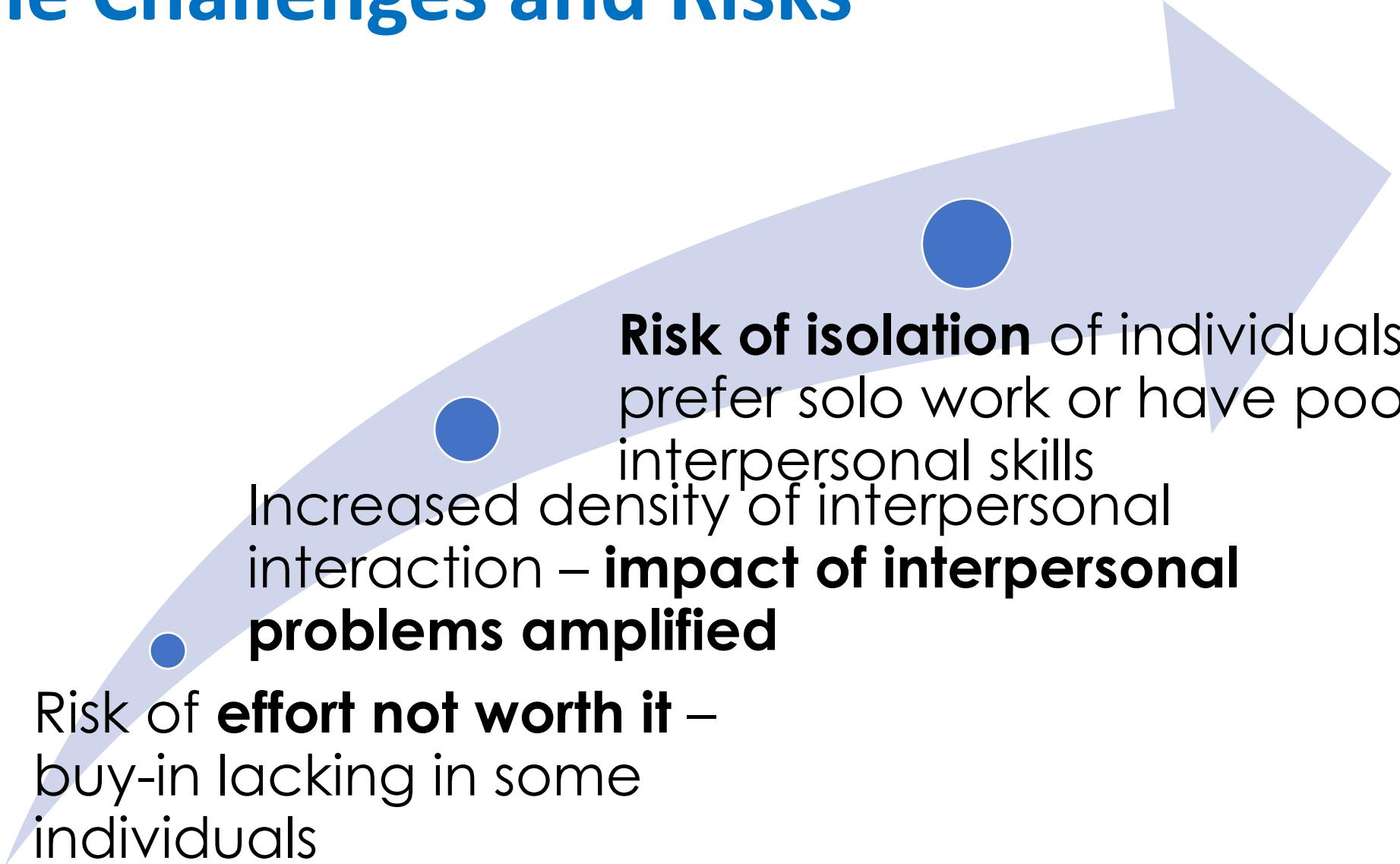


Fewer interruptions



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>

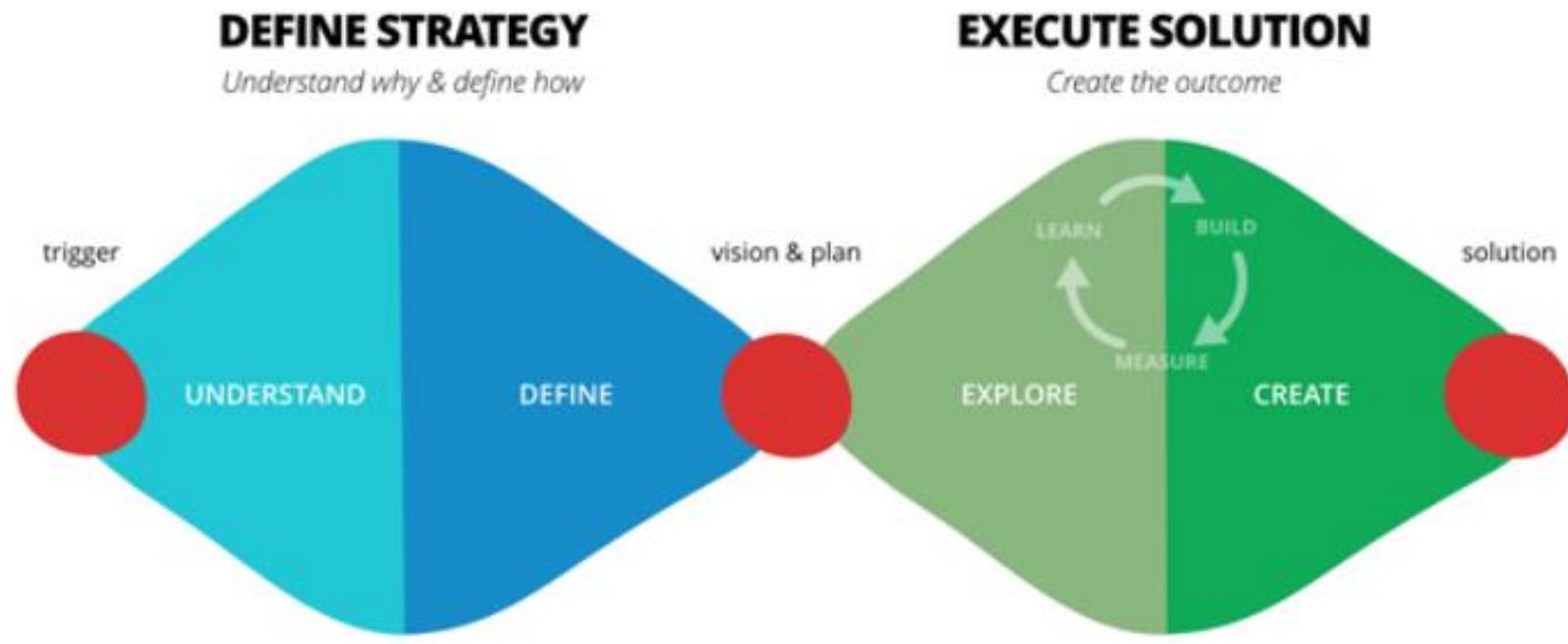
<http://scrummastertoolbox.libsyn.com/how-to-find-what-agile-retrospective-format-works-for-your-team-justin-chapman>

https://www.ponolabs.com/labs/wp-content/uploads/2019/03/Team_Canvas_v5.pdf

Retrospectives

Double diamond decision making

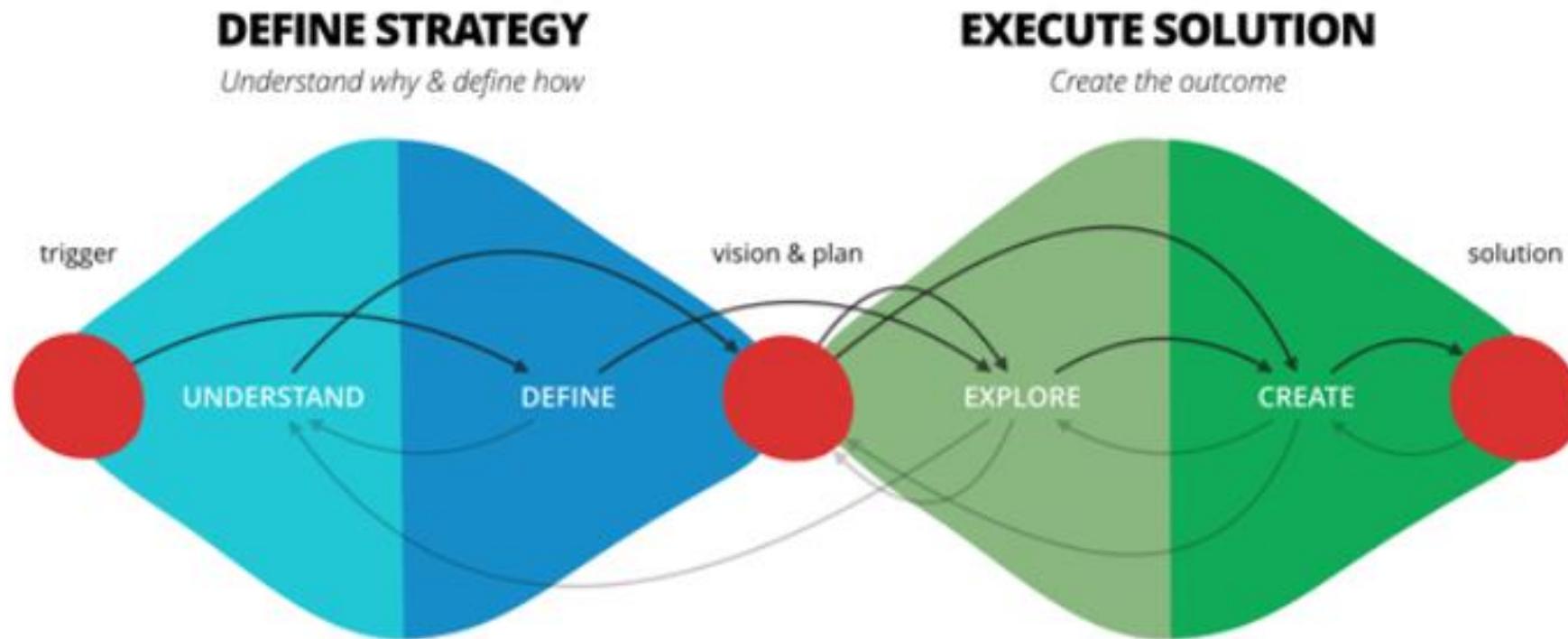
<https://www.thoughtworks.com/insights/blog/double-diamond#:~:text=The%20'Double%20Diamond'%20process%20maps,of%20thinking%20that%20designers%20use.&text=It%20describes%20significant%20up%2Dfront,to%20produce%20a%20final%20solution>



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

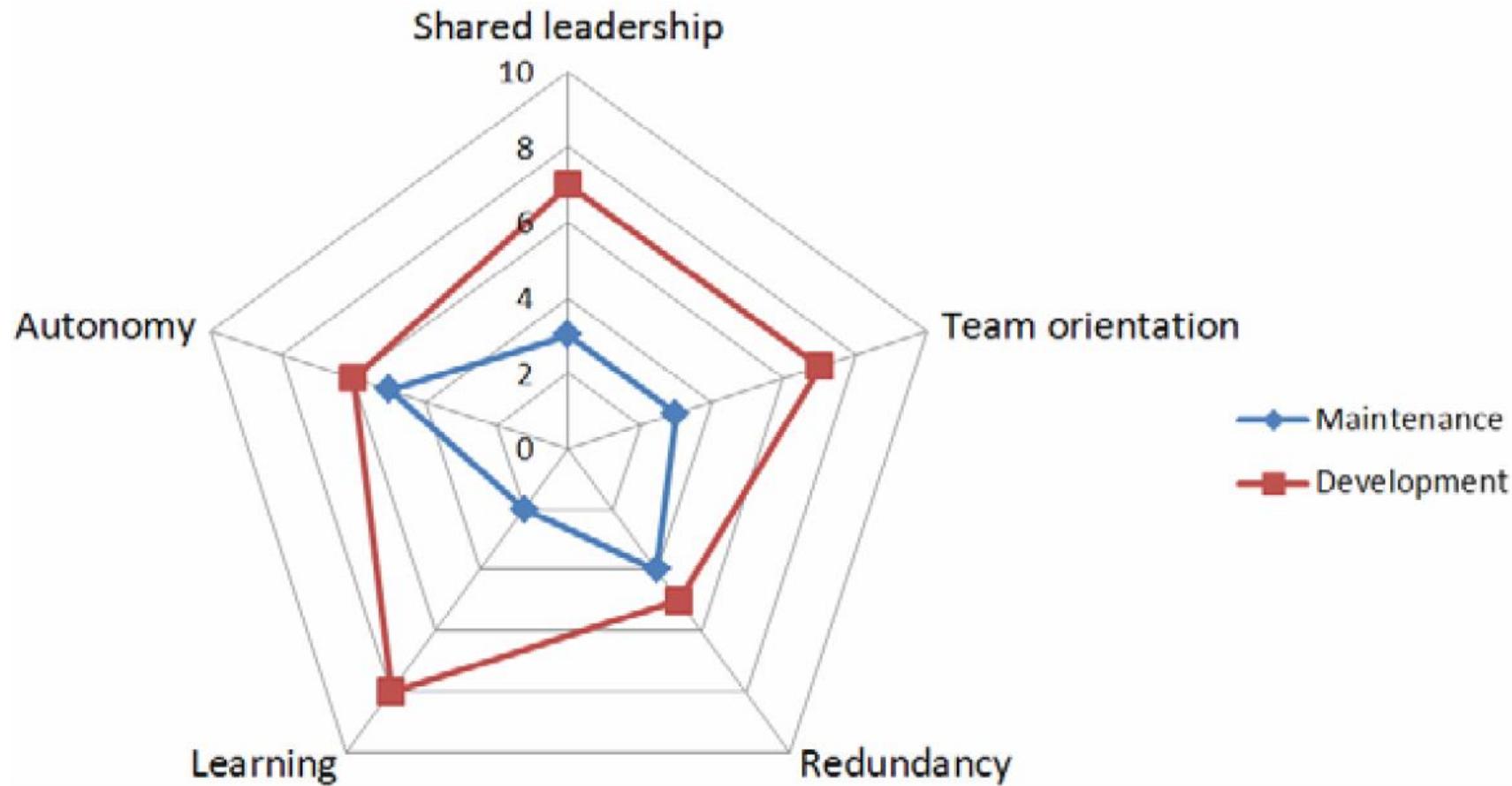


Fig. 1. A plot of teamwork characteristics of the two teams

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

CISE ENSE701

I has a question...



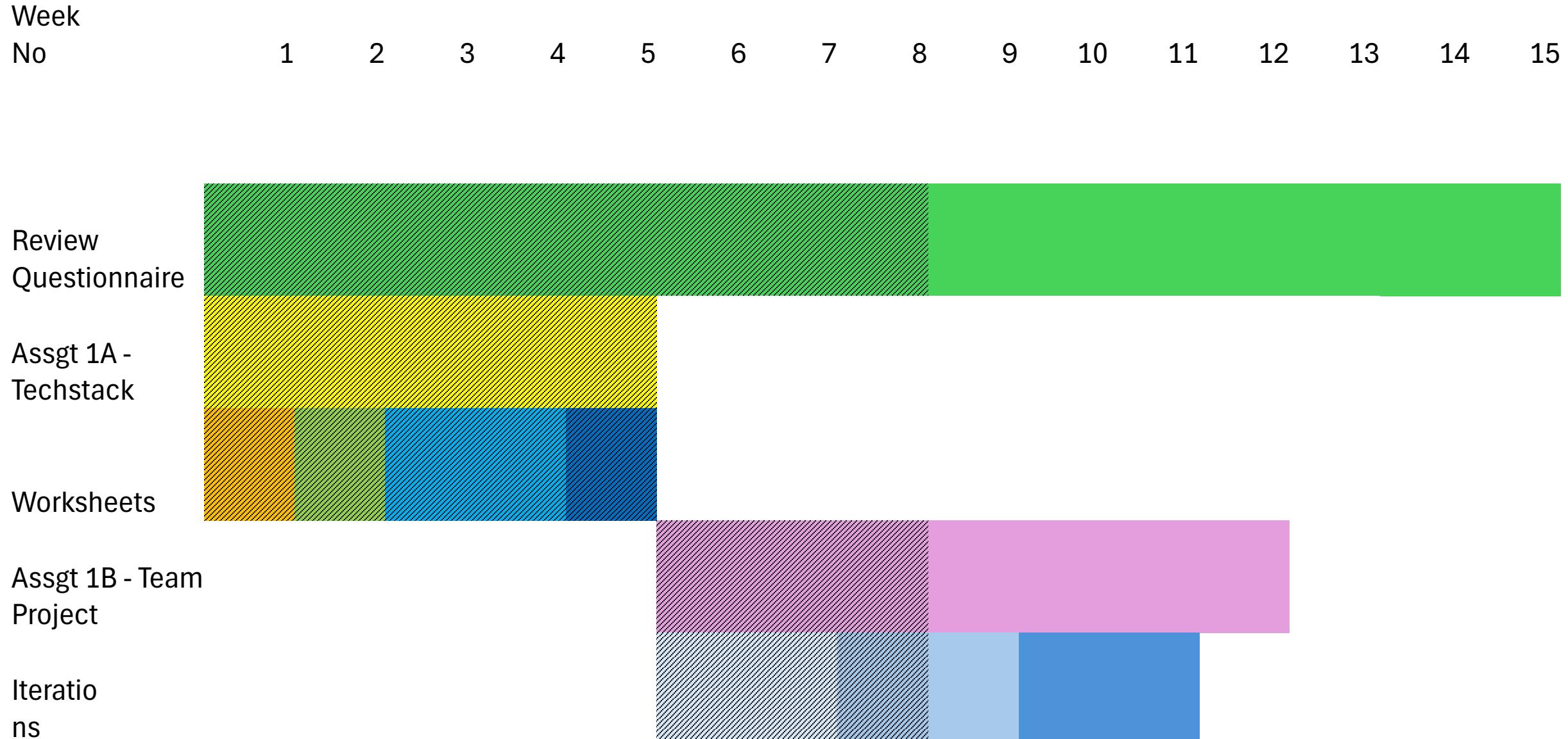
45

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

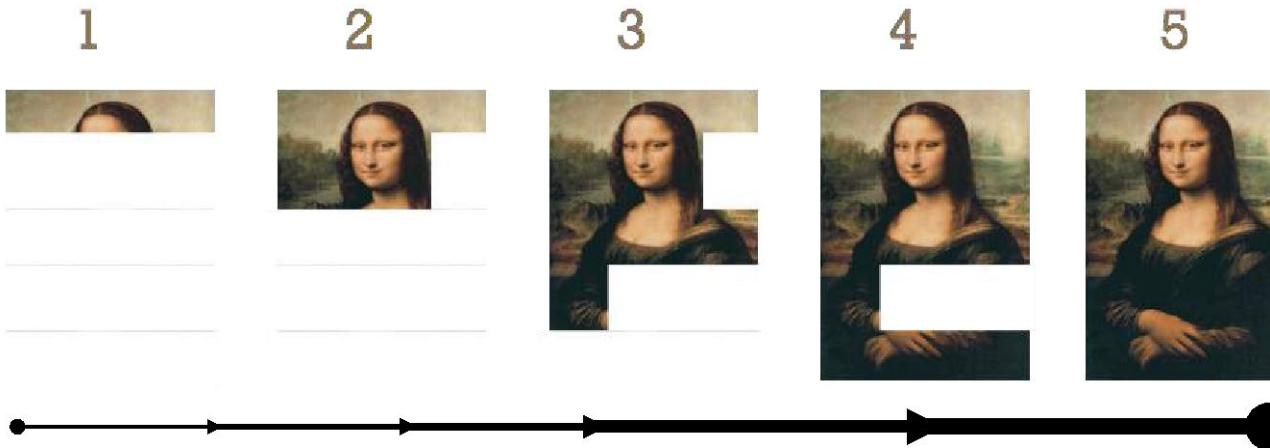
Is useful to users – solves a problem

Behaves as expected

Specifications and Scenarios Based on Acceptance criteria

Coding is a craft

Crafting code is different to just writing it!

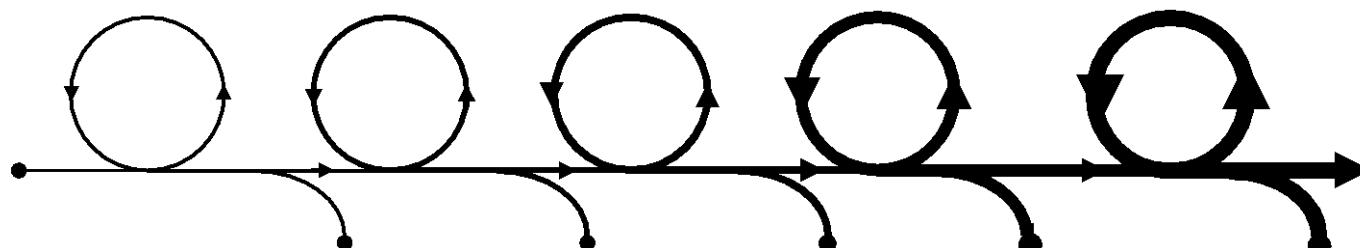


We do NOT create a full design then build from the ground up until we have the finished product.



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

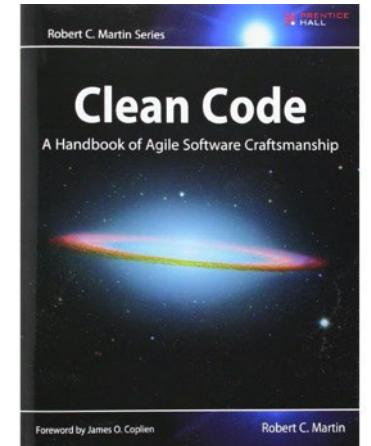
Functions should take as few arguments as possible, preferably none

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("DatabaseErrors.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            db.LogError("An error occurred: ", ex);
15            File.WriteAllText("\LocalErrors.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*.

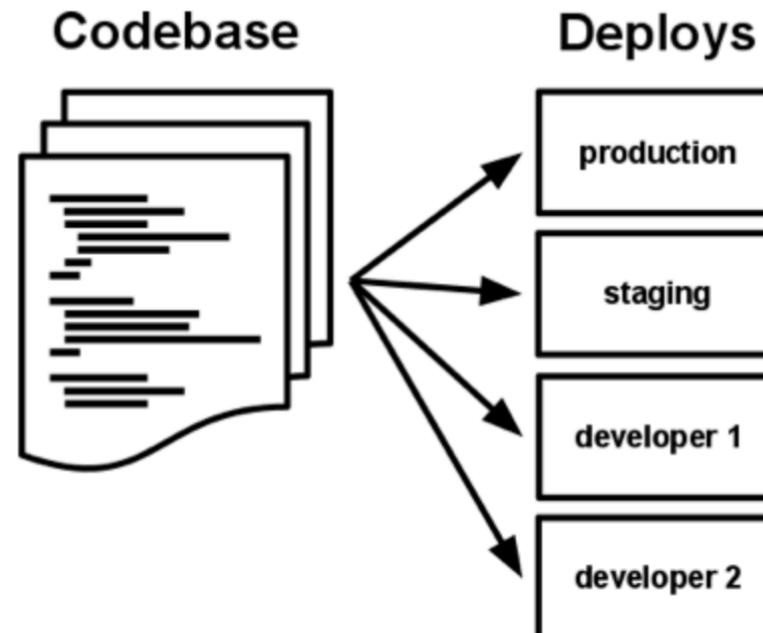
A *codebase* is any single repo (in a centralized revision control system like Subversion), or any set of repos who share a root commit (in a decentralized revision control system like Git).

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?

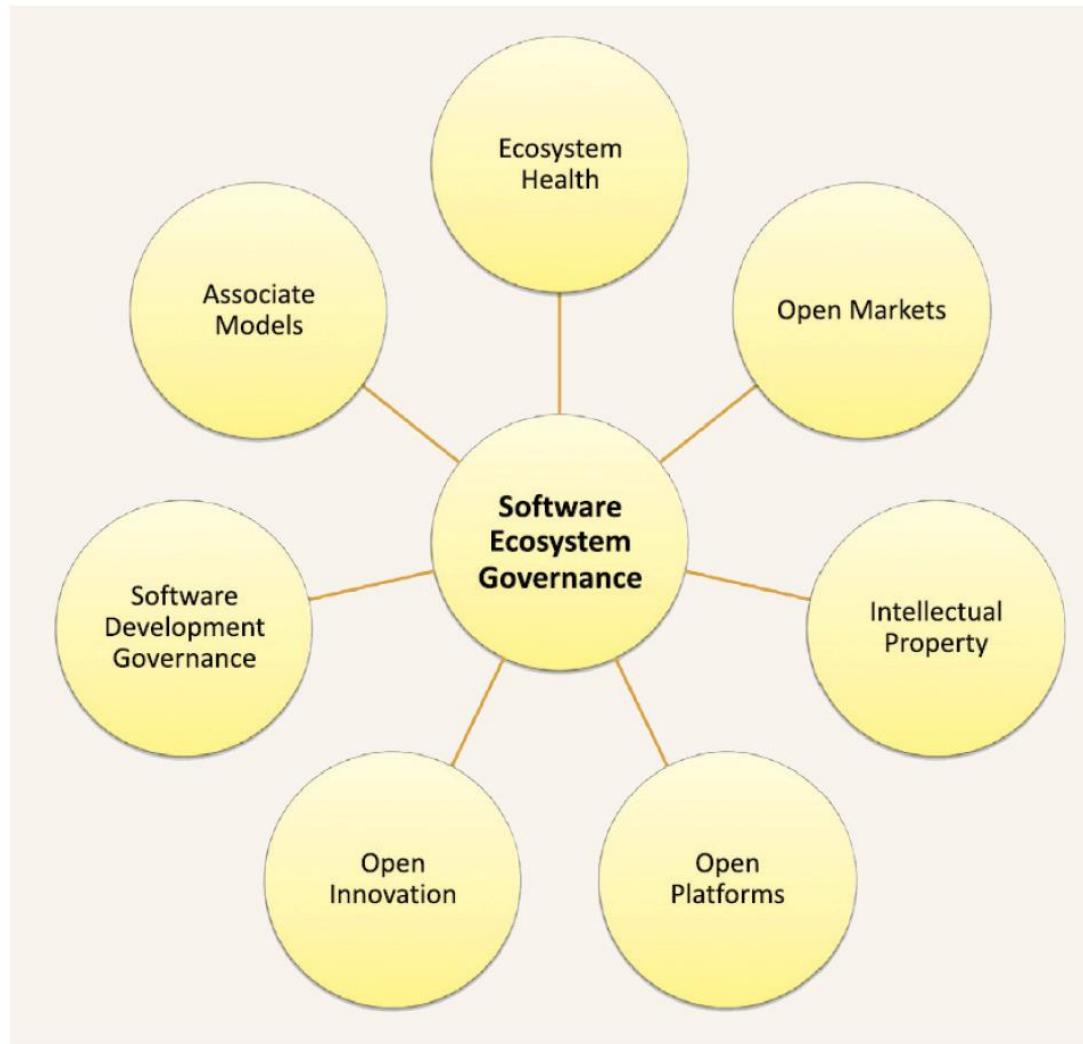


Figure 1: Seven Focus Areas of the SEG-M² [5, fig. 2]

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

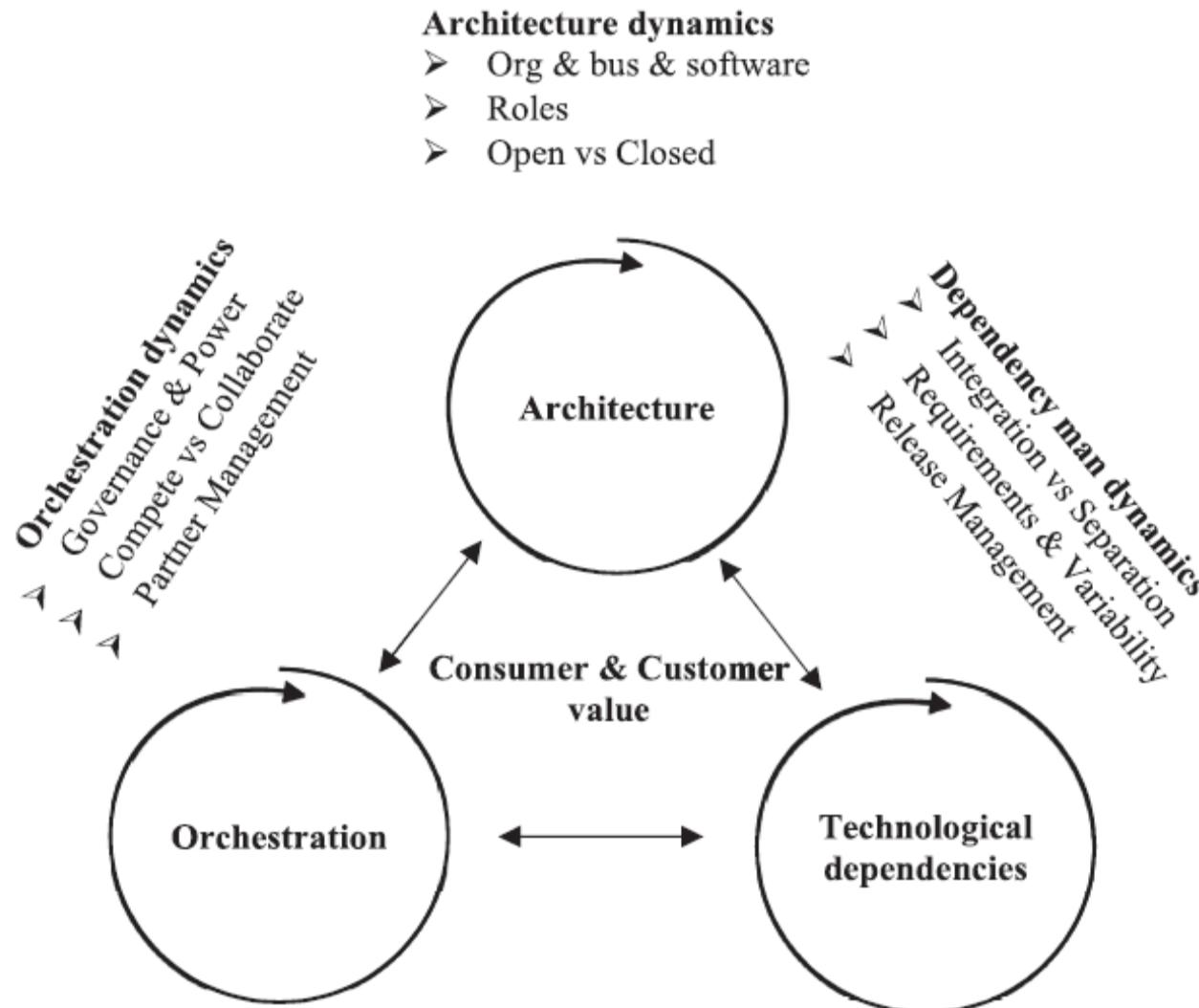


Fig. 1. SECO configuration and contingencies.

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

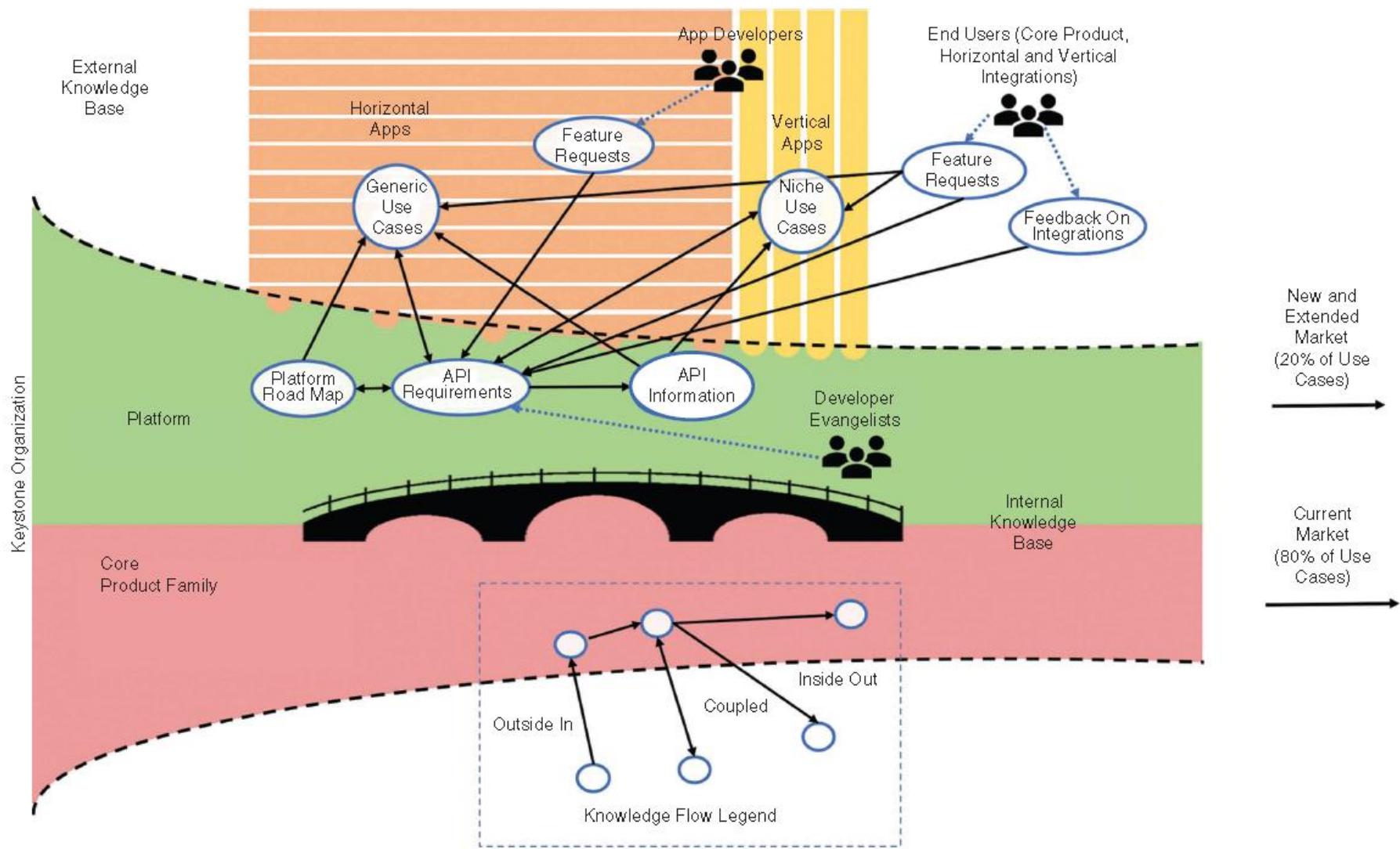


FIGURE 1. Requirements knowledge flows in the SECO, as adapted from the OI model by Chesbrough and Bogers.² The knowledge flow legend indicates the type of knowledge flow represented by the directional arrows.

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>



#171478945



Questions and Comments....



Tony Clear 2024 S2

I has a question...

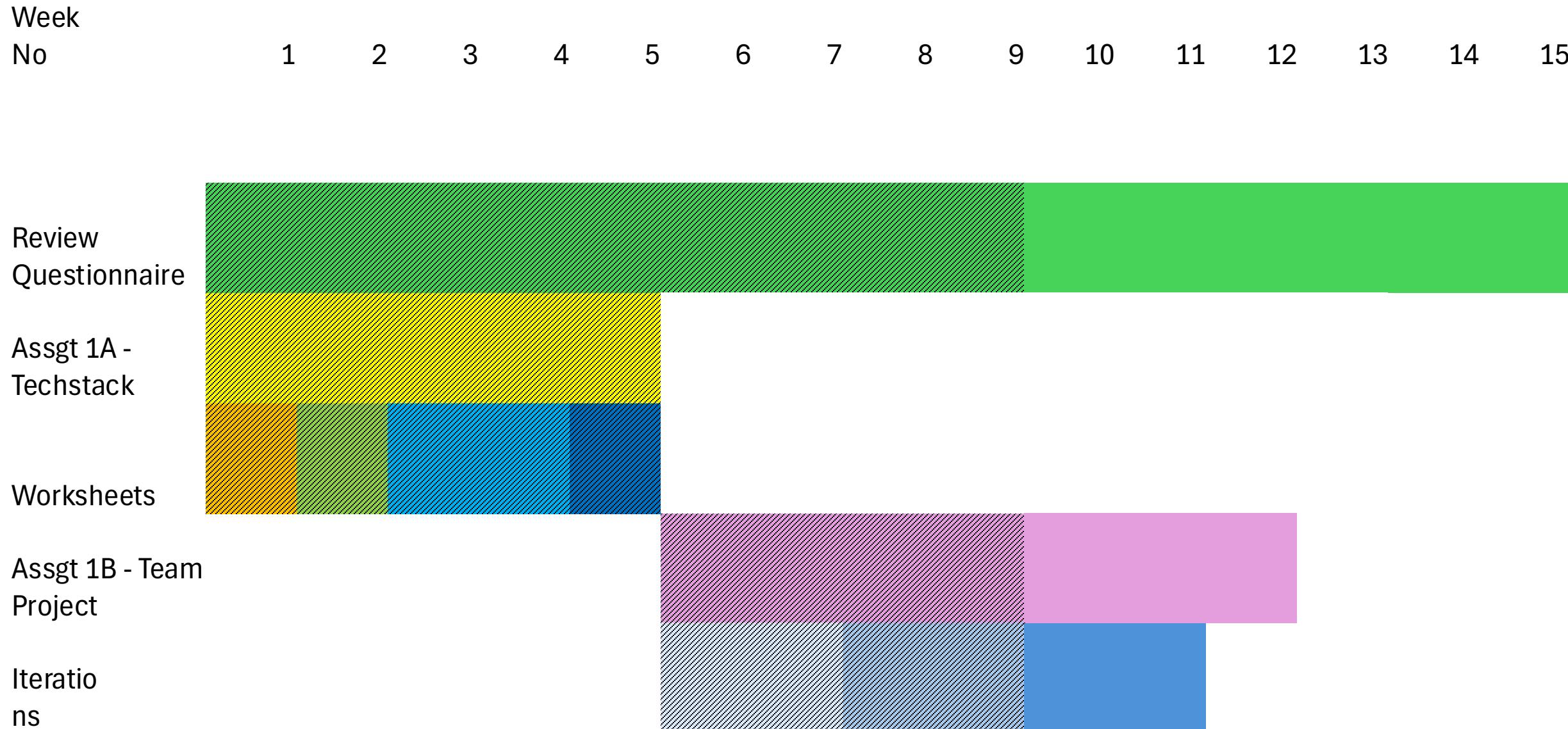


Software Risks and Scale

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?

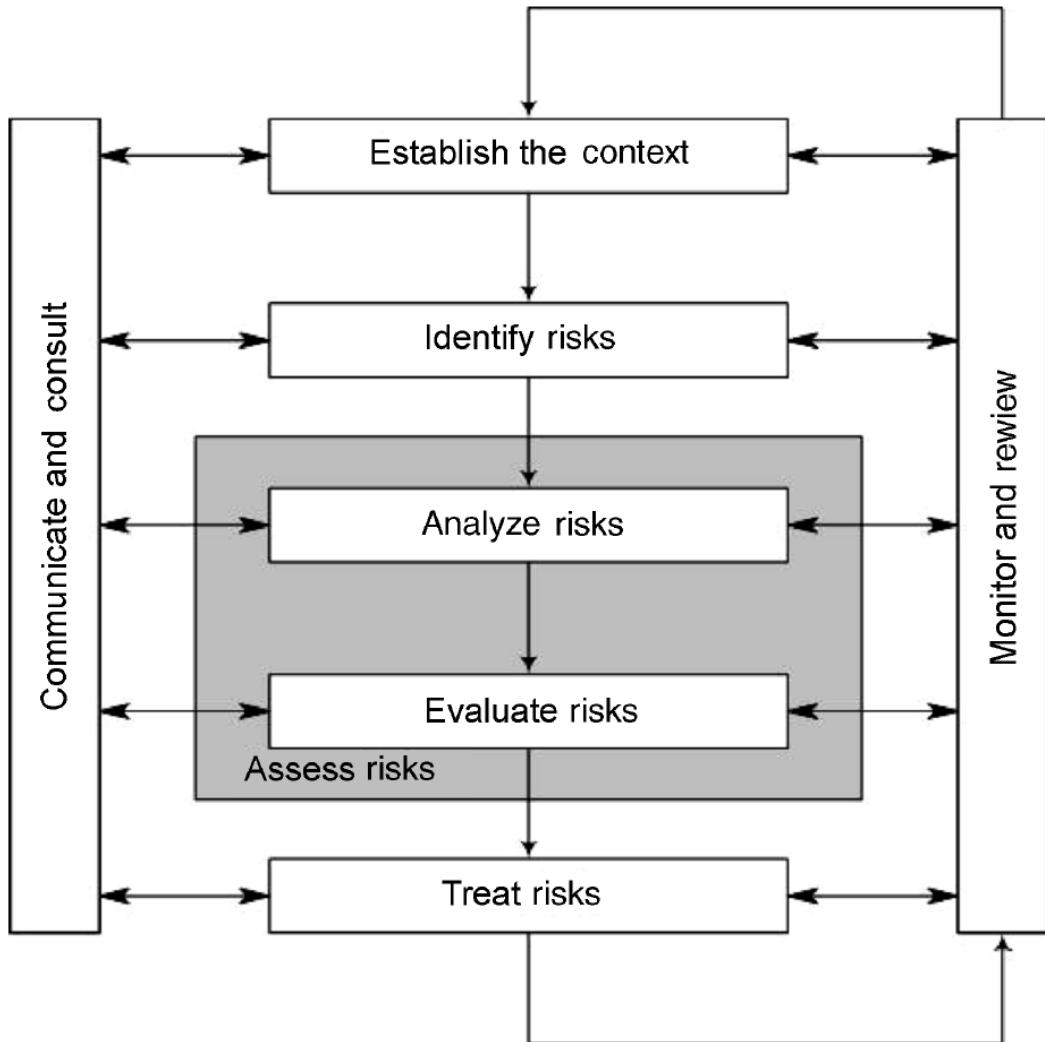


FIGURE 18.1 Risk management (AS/NZS, 1999 p.16).

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?

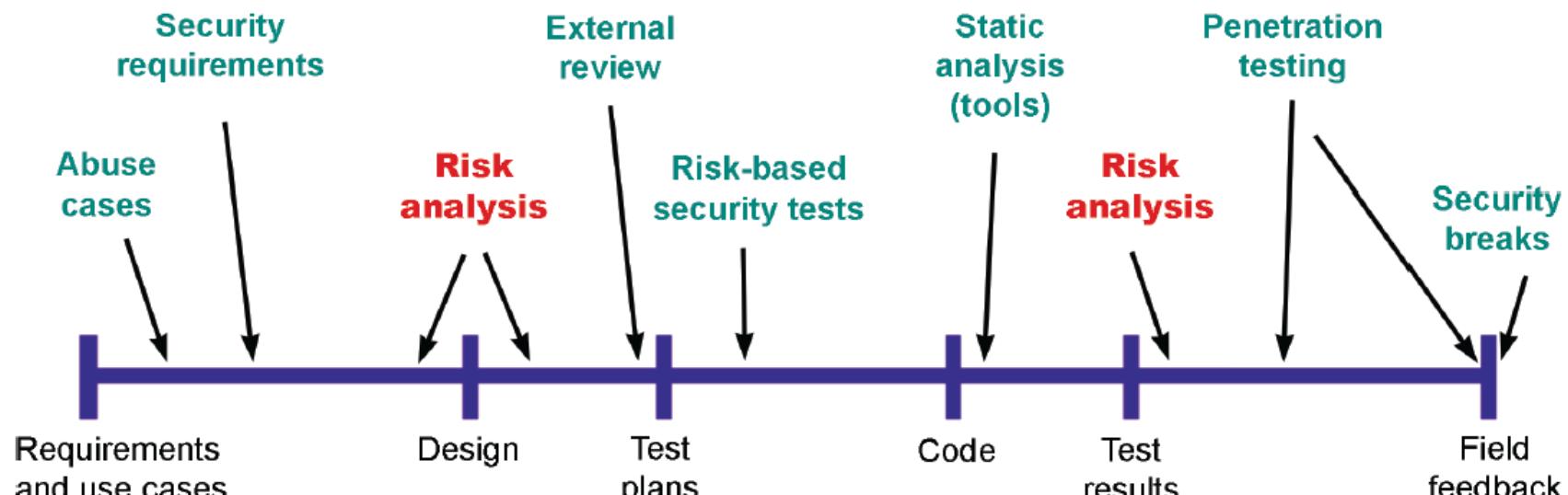


FIGURE 18.2 Hilson's security life cycle.

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.

McGinn, M. (2011). Software and Risk – Stakeholders and SoDIS? *Journal of Computer Information Systems*, 51(1), 10–19.

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?

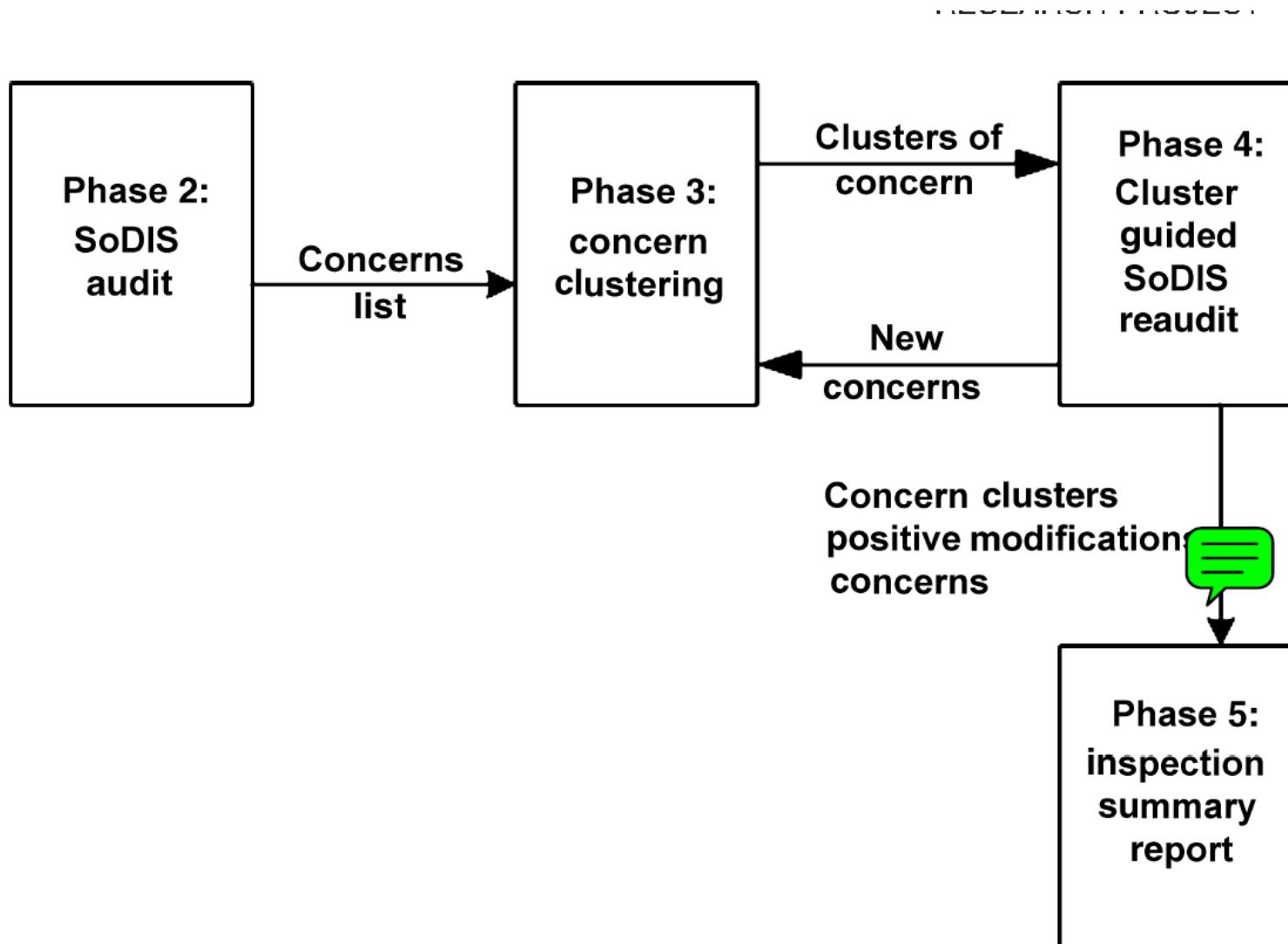


FIGURE 18.8 SoDIS inspection phases 2–5.

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

;64
g a
t
IS
on



#171478945



Questions and Comments....



Tony Clear 2024 S2

I has a question...



Agile, History and Ways of Working

Week 11

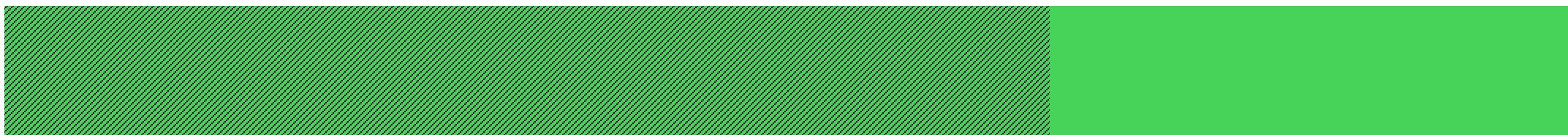


Taking Stock

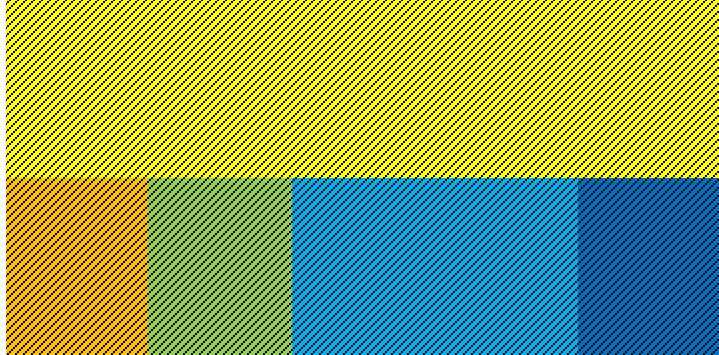
Week
No

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

Review
Questionnaire

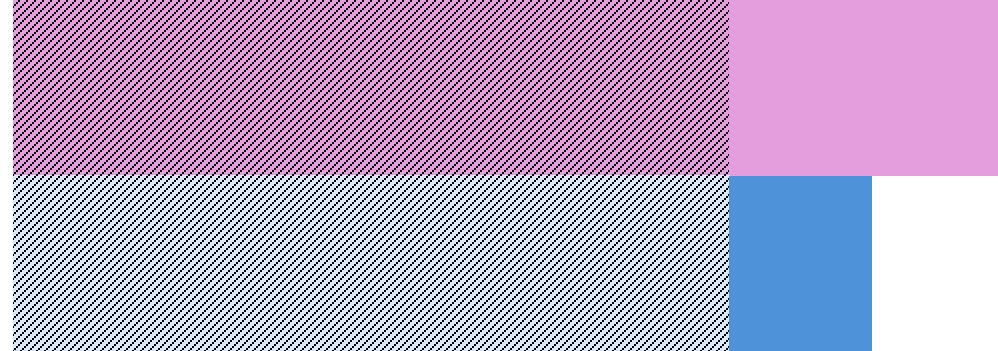


Assgt 1A -
Techstack



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**



#171478945



Questions and Comments....



Tony Clear 2024 S2

CISE ENSE701

I has a question...



COMP 501 – Part Two



Course lecturer –
Assoc Prof Tony Clear

AUT



COMP 501 – Part B Requirements and Modelling

Course lecturer – Assoc Prof Tony Clear

Computing Technology in Society

Paper Overview





Systems Analysis and Design

System Development

- **Systems Analysis and Design** [*may have other names*]
 - Step-by-step process for developing high-quality information systems
 - **Information systems:** Combination of technology, people, and data to perform certain business functions
- **What Does a ‘Systems Analyst’ Do?** [*may have many other names*]
 - Plans, develops, and maintains information systems
 - Manages IT projects, including tasks, resources, schedules, and costs
 - Conducts meetings, delivers presentations, and writes memos, reports, and documentation

Systems Analysis and Design

- Step-by-step process for developing high-quality Enterprise Systems (ES)
- Who is responsible?
 - Organizations may have an IT department
 - IT department may have a software engineering unit
 - Software engineering units consist of development team (s) (systems analysts/designers/developers), a quality assurance team (testers) and a documentation team (technical writers).
 - Systems Analysts focus on the analysis/design work
 - They also plan, help develop/test, and maintain information systems

Techniques to Understand the Business to Design the System

Business Process modeling (BPM)

- BPM is carried out by business analysts and managers
- BPM is the activity of representing processes (operations) & information needs of a business
- Business process modeling notation (BPMN)

Business Profile

- Mission statement
- **Organizational structure**
- Business processes
- Strengths Markets
- IT infrastructure
- Customers

Business Models

- Business model - graphically displays one or more **business process**
- **Business process** - is a specific set of transactions, events and results that can be described and documented

Systems Development Skills: Systems Analyst

- Investigates, analyzes, designs, develops, installs, evaluates, and maintains a company's information systems
- Constantly interacts with users and managers within and outside the organization
- **Knowledge, Skills, and Education**
 - Technical knowledge
 - Communication and business skills
 - **Critical thinking skills**
 - Education - A college degree in information systems, science, or business
 - Some IT experience is required
 - Certification
 - Helps IT professionals learn new skills and gain recognition for their efforts

Systems Development Skills: Systems Analyst

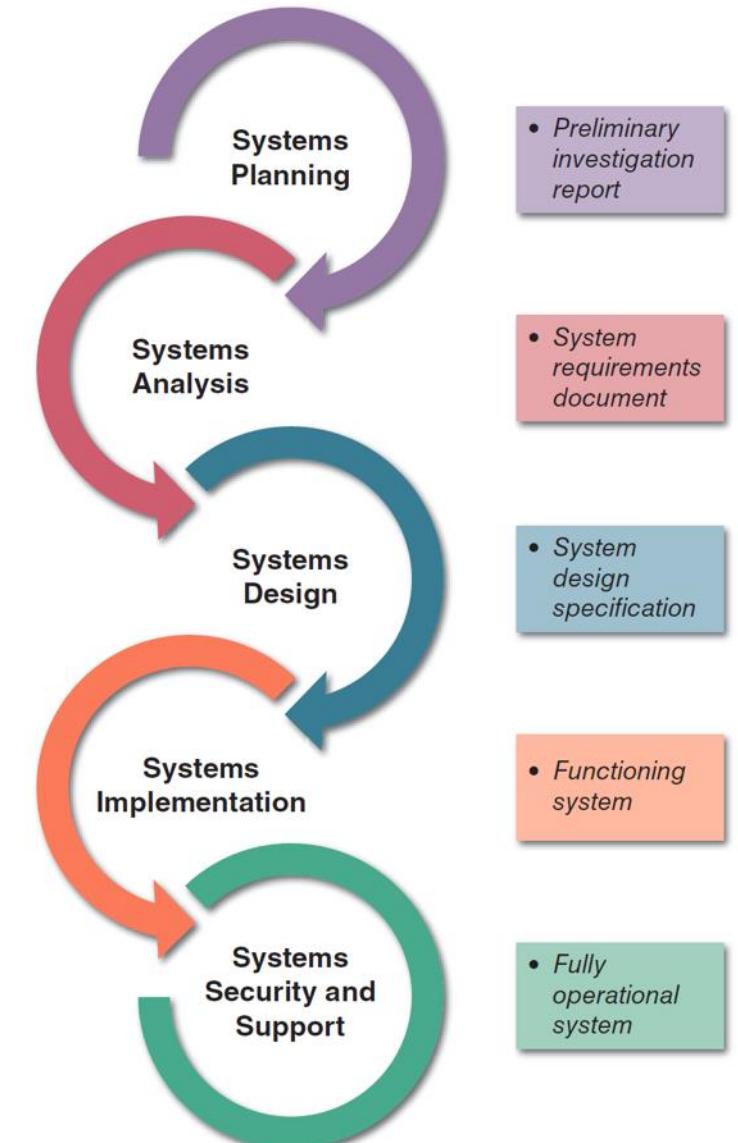
Technically qualified	Business oriented	Highly motivated	Communication skills (teamwork)	Strong analytical and critical thinking skills
<ul style="list-style-type: none">• Degree in IT, CS, SE or IS• Certification course	<ul style="list-style-type: none">• Understand about business processes or operations, customers, organizational structures, values, culture• Know more about the IT needs of the business than managers would know themselves• Provide business solutions (ES) based on business needs rather than the technology trend.	<ul style="list-style-type: none">• You have firm belief and courage• Have a positive attitude; do job with pride and passion• Show respect; recognize other people's good qualities or achievements.• Have empathy & sincerity	<ul style="list-style-type: none">• Understand problems• exchanging information all the time• build relationships & connections• communicate negative or difficult messages without creating conflict or destroying trust	<ul style="list-style-type: none">• break a problem down in parts• provide solution to each part• Suggest new ideas

Systems Analyst: The Role

- **Role**
 - Acts a translators to managers and programmers
 - A company's best line of defense in an IT disaster
 - Most valuable skill - The ability to listen
 - Seeks feedback from users to ensure that systems do not deviate from accomplishing set objectives
- **Tasks systems analyst does to help develop Enterprise Systems**
 - Translate business requirements into IT projects
 - Plan projects; develop schedules & estimate's cost
 - Document business profile, (review/document business processes), create models (DFDs)
 - Help to create various designs
 - Select hardware & select software packages
 - Conduct meetings, deliver presentations, write memos, produce reports and other documentation in regards to project
 - Test
 - Train users

Systems Development Method for Structured Analysis

- The SDLC model usually includes five steps
 - Systems planning
 - Systems analysis
 - Systems design
 - Systems implementation
 - Systems support and security

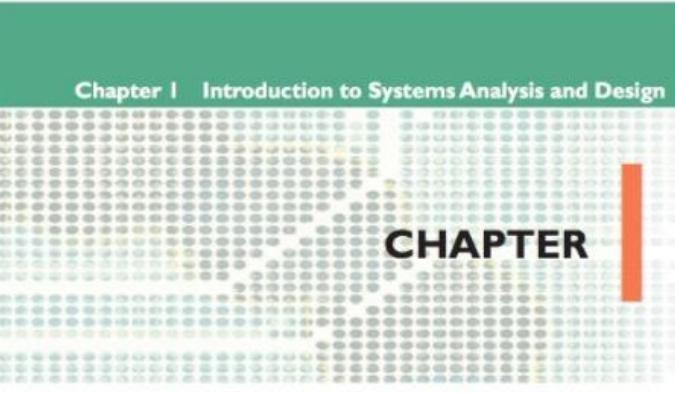


Related Chapter in eBook

Table Of Contents

CP1117: INFS500 Enterprise Systems (Custom)
Tiller

- ii Title page
- iii Copyright page
- iv Table of Contents
- 1 Extract 1 Introduction to Systems Analysis and Design
- 40 Extract 2 Analysing the Business Case
- 70 Extract 3 Data and Process Modeling
- 106 Extract 4 Development Strategies
- 138 Extract 5 Managing Systems Implementation
- 186 Extract 6 Managing Systems Support and Security
- 239 Extract 7 Enterprise Systems
- 277 Extract 8 Information and Decision Support Systems
- 331 Extract 9 Knowledge Management and Specialised...



Chapter 1 is the first of three chapters in the systems planning phase. This chapter describes the role of information technology in today's dynamic business environment. This chapter describes the development of information systems, systems analysis and design concepts, and various systems development methods. This chapter also describes the role of the information technology department and its people.

LEARNING OBJECTIVES

When you finish this chapter, you should be able to:

- Describe the impact of information technology
- Define systems analysis and design and the role of a systems analyst
- Define an information system and describe its components
- Explain how to use business profiles and models
- Explain Internet business strategies and relationships, including B2C and B2B
- Identify various types of information systems and explain who uses them
- Distinguish among structured analysis, object-oriented analysis, and agile methods

Introduction to Systems Analysis and Design

The chapter includes four "Case in Point" discussion questions to help contextualize the concepts described in the text. The "Question of Ethics" invites examination of the ACM's code of ethics and those of a developing systems analyst.

CHAPTER CONTENTS

- 1.1 Introduction
- 1.2 What Is Information Technology?
Case in Point 1.1: Cloud Nine Financial Advisors
- 1.3 Information System Components
- 1.4 Business Today
- 1.5 Modeling Business Operations
- 1.6 Business Information Systems
- 1.7 What Information Do Users Need?
- 1.8 Systems Development Tools
- 1.9 Systems Development Methods
- 1.10 The Information Technology Department
Case in Point 1.2: Global Hotels and Momma's Motels
- Case in Point 1.3: What Should Lisa

Modelling Enterprise Systems

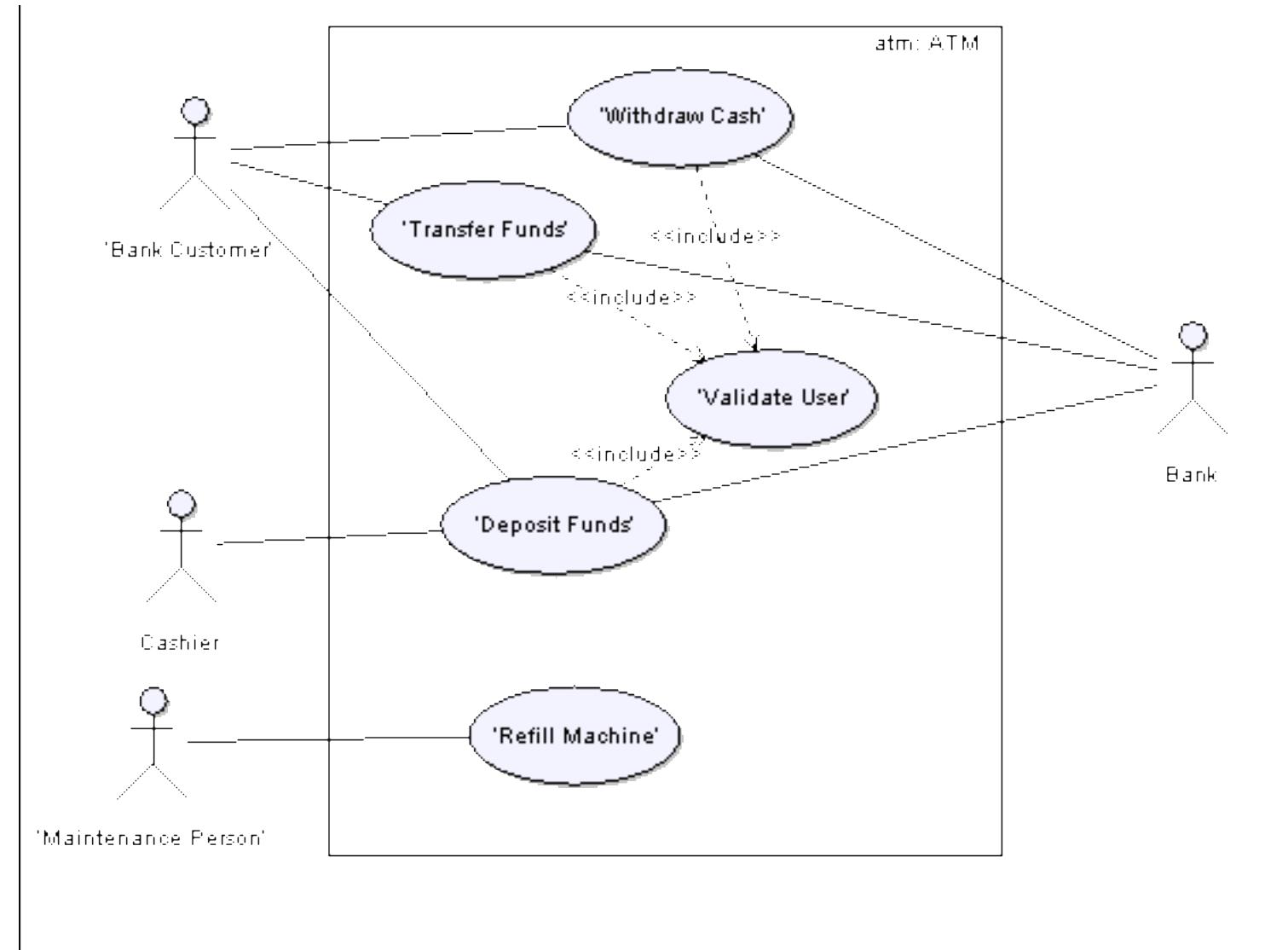
Modelling

- Analysts and Developers will use various models to learn, get feedback, and communicate what needs to be implemented and how it needs to be implemented
 - Requirements model
 - Describes the features that a system must provide,
 - **Data Flow Diagrams (DFD)**
 - Object model (O/O analysis/design)
 - Describes objects, which combine data and processes
 - Data model
 - Describes data structures and design
 - Network model
 - Describes the design and protocols of telecommunications links
 - Process model
 - Describes the logic of a process that programmers use to write code.

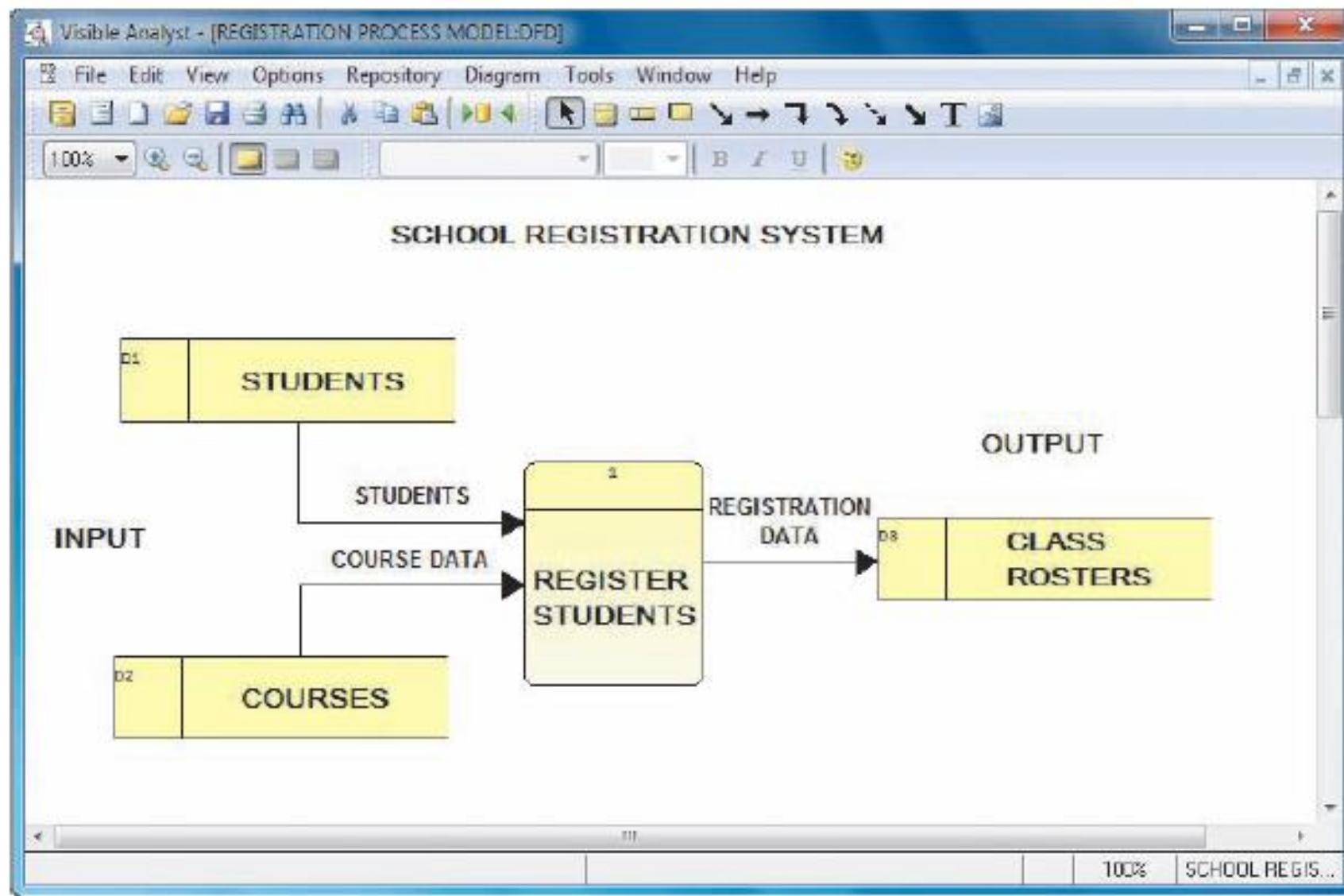


We will learn DFDs in labs

Use case diagram / model



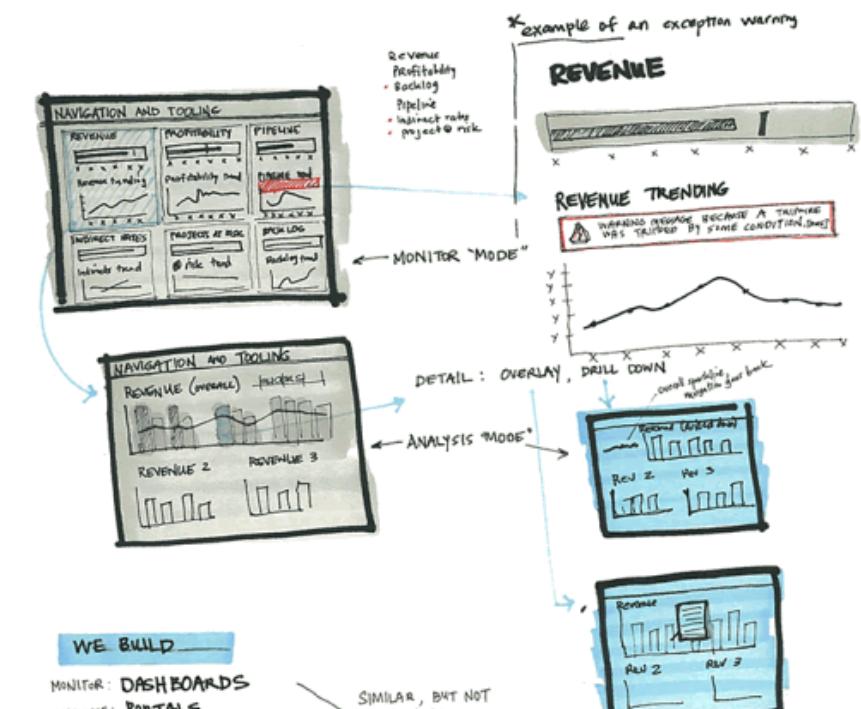
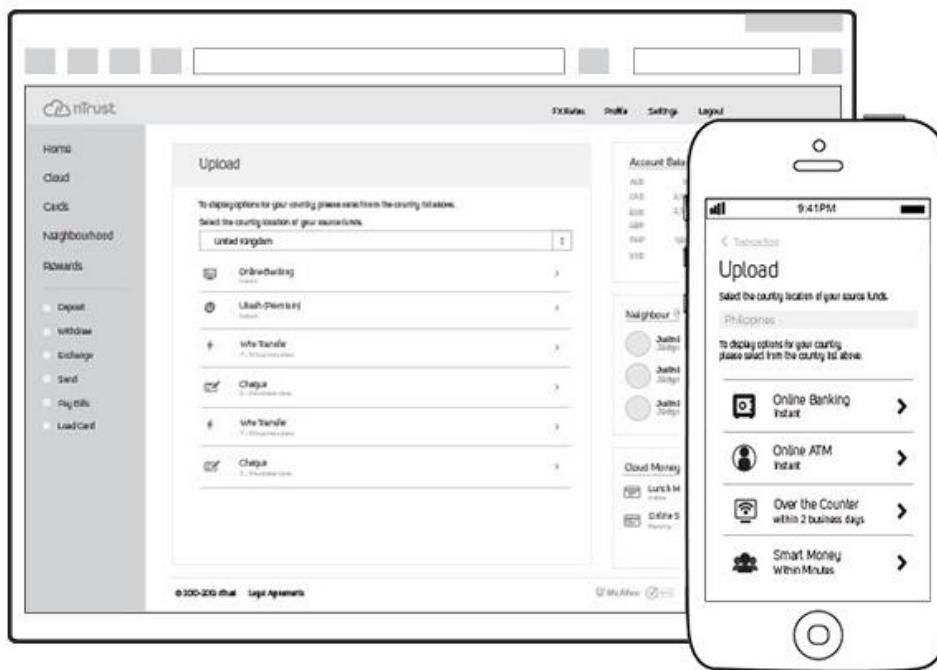
Data flow model example



Systems Development Technique

- Prototyping

- Allows to identify, test, and get a feedback on features
- It is a working version of a system but not fully functional system
- Test input, output and user interface before making a final implementation decision



Systems Development Tools

- A confusing plethora of tools to help automate parts of the development lifecycle

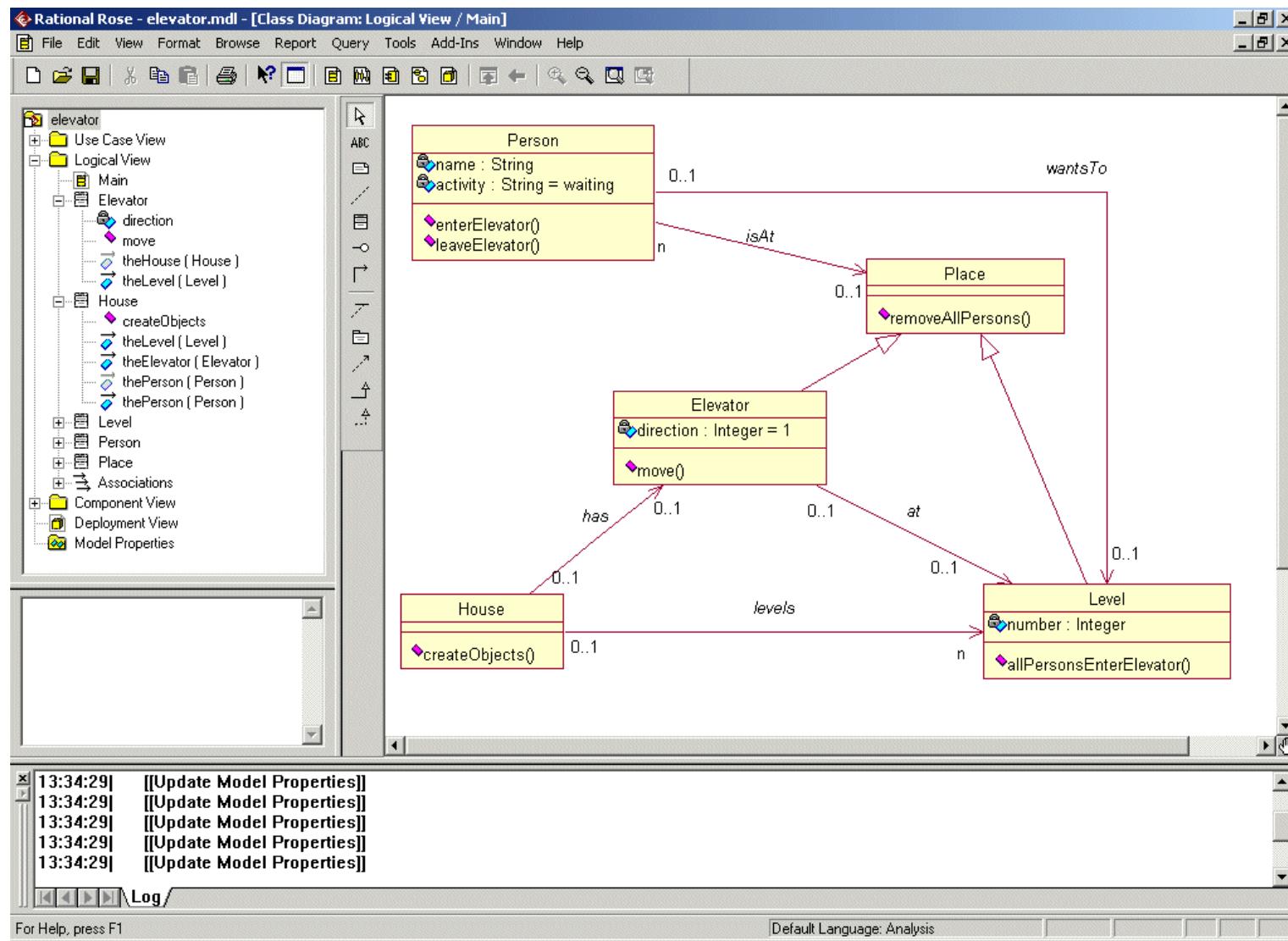
https://digital.ai/sites/default/files/pictures/2020-06/Digital.ai_Periodic-Table-of-DevOps.pdf

PERIODIC TABLE OF DEVOPS TOOLS (V2)																	
1	Fm	Gh	Github	Os	Fr	Fm	ScM	Database Mgmt	Build	Ch	Pu	An	Sl	Dk	Aws	2	Fm
3	Os	Gt	Git	Dm	Dbmaestro		CI	Repo Mgmt	Testing	Chef	Puppet	Ansible	Salt	Docker	AmazonWeb Services		
4	Pd						Deployment	Config / Provisioning	Containerization	Otto	BladeLogic	Vagrant	Terraform	Rk	Az		
11	Fm	Bb	Bitbucket	Lb	Liquibase		Cloud / IaaS / Pass	Release Mgmt	Collaboration				rkt				
12	Os						Bl / Monitoring	Logging	Security								
19	Os	Gl	GitLab	Rg	Redgate	Mv	Gradle	At									
20	En																
21	Os																
22	Os																
23	Os																
24	Os																
25	Fr																
26	Os																
27	Fr																
28	Os																
29	Pd																
30	Os																
31	Pd																
32	Os																
33	Os																
34	Os																
35	Os																
36	En																
37	Os																
38	En																
39	Os																
40	Os																
41	Os																
42	Fr																
43	Os																
44	Fr																
45	Os																
46	Fm																
47	Pd																
48	Fm																
49	Fr																
50	Fr																
51	Os																
52	Os																
53	Fr																
54	Os																
55	Os																
56	En																
57	Fr																
58	Os																
59	Os																
60	Fr																
61	Fr																
62	Fr																
63	Os																
64	Fm																
65	Fm																
66	Os																
67	En																
68	Fm																
69	En																
70	En																
71	Os																
72	Fm																
73	En																
74	En																
75	Os																
76	Os																
77	Fr																
78	Os																
79	En																
80	Os																
81	Os																
82	Os																
83	Fm																
84	Pd																
85	En																
86	En																
87	Fm																
88	En																
89	Os																
90	En																
91	En																
92	En																
93	En																
94	En																
95	En																
96	En																
97	En																
98	Pd																
99	Fm																
100	Fr																
101	Pd																
102	Fm																
103	Fm																
104	Pd																
105	En																
106	Os																
107	Fm																
108	Os																
109	Os																
110	En																
111	Os																
112	Os																
113	En																
114	Fm																
115	Fm																
116	Os																
117	Os																
118	Os																
119	Os																
120	En																

Xlr	Ur	Bm	Hp	Au	Pl	Sr	Tfs	Tr	Jr	Rf	Sl	Fd	Pv	Sn			
XL Release	UrbanCode Release Process	BMC Release Process	HP Cedar	Automic	Plutora Release	Serena Release	Team Foundation	Trello	Jira	HipChat	Slack	Flowdock	Pivotal Tracker	ServiceNow			
Kibana	New Relic	Nagios	Zabbix	Datalog	Elasticsearch	StackState	Splunk	Logentries	Sumo Logic	Logstash	Graylog	Snort	Tripwire	Fortify			

Follow @xebialabs

CASE tool – Rational Rose





Agile Models Distilled: Potential Artifacts for Agile Modeling

[Home](#) [Start Here](#) [Core Practices](#) [Disciplines](#) [Artifacts](#) [Resources](#) [Contact Me](#)

Choose Your WoW!

To be effective, the principle **Multiple Models** tells us that agile modelers should know a wide variety of modeling techniques so that they have the skills and knowledge to **apply the right artifact(s)** for the situation at hand. Unfortunately this is easier said than done. This page links to summary descriptions of a wide variety of modeling artifacts. Each page describes the artifact, provides an example or two, and provides links to suggested resources.

Some, but not all, of the potential models that you may want to create on a software development project include:

AGILE Modelling – Potential Models

- Acceptance Test
- Business Rule (Template)
- Change Case (Template)
- Class Responsibility Collaborator (CRC) model
- Constraint
- Contract model (Template)
- Data Flow Diagram (DFD)
- Domain Model
- Essential/Abstract Use Case (Template)
- Essential/Abstract User Interface Prototype
- Feature
- Free-Form Diagrams
- Flow Chart
- Glossary
- Logical Data Model (LDM)
- Mind Map
- Network Diagram
- Object Role Model (ORM) Diagram
- Personas
- Physical Data Model (PDM)
- Robustness Diagram
- Security Threat Model
- System Use Case (Template)
- Technical Requirement
- UML Activity Diagram
- UML Class Diagram



We will learn DFDs in labs

<http://agilemodeling.com/artifacts/>

Tutorial Session 6: Modelling Dataflows

1. Modelling dataflows and why?

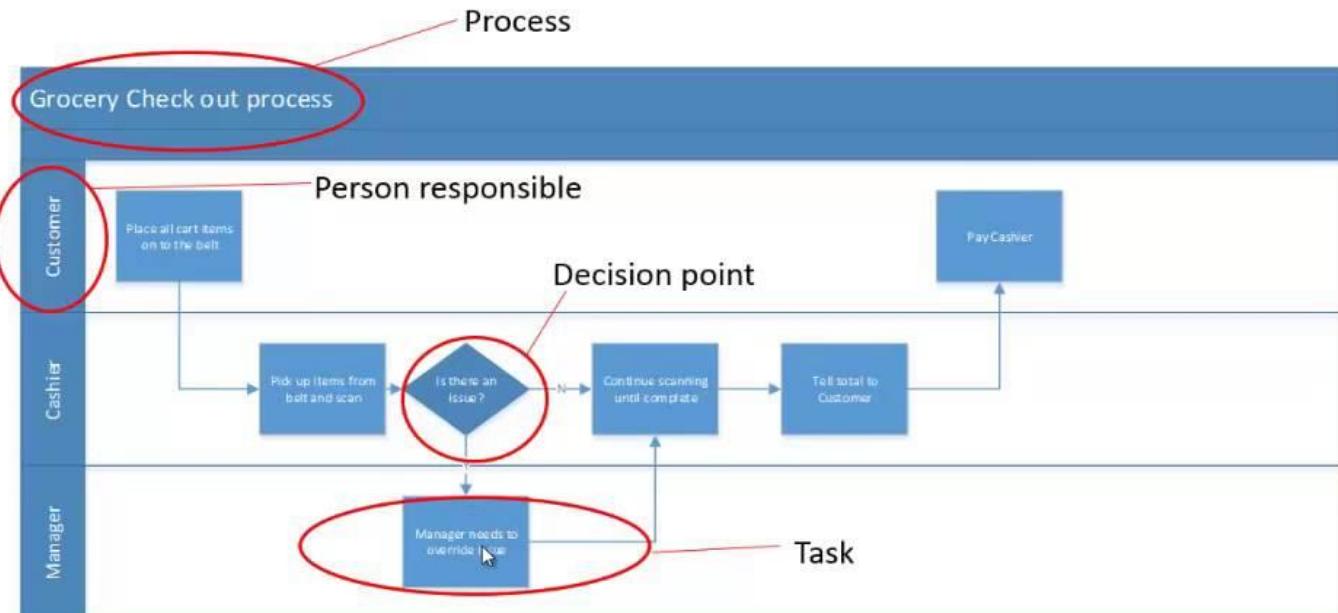
2. Blog Case – Digital Government: Remodelling a Process
The Service: Applying for NZ Superannuation
<https://www.digital.govt.nz/blog/user-focussed-content-equals-good-business/>

3. DFD's

Modelling Dataflows – Swimlane Process Maps

<https://www.youtube.com/watch?v=wQxnzLu7TqU>

Swim Lane Process Map



Modelling Dataflows – Swimlane Process Maps

1. Modelling Current and Future States

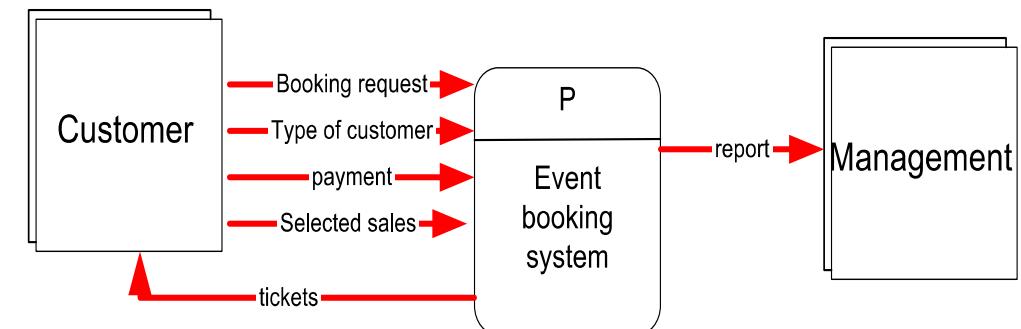
-
- Current State Modelling
- <https://www.youtube.com/watch?v=lmjvmcuhaKY>
-
- Future State Modelling and Process Improvement Design
- <https://www.youtube.com/watch?v=cAkPmSmK6ZM>

The background of the slide features a large, abstract graphic composed of numerous small, thin-lined triangles. These triangles are primarily colored in shades of orange and red, creating a warm, fire-like appearance. They are arranged in a way that suggests a sense of motion or flow, with many triangles pointing towards the right side of the frame.

Data Flow Diagrams (DFDs)

Objectives of this section

- Describe data and process modeling concepts and tools, including data flow diagrams, and process descriptions
- Describe the symbols used in data flow diagrams and explain the rules for their use
- Exposure to drawing **context diagram**



Process Modelling using DFDs

- DFDs show how the system transforms input data into useful information
- Technique for organising and documenting a system's **processes, inputs, outputs and data stores (storage)**
- Also includes the **external entities**.
- Process modelling is structured analysis tool which deals with a business process from the systems owners' and systems users' point of view
- What the system does or must do
- We use DFD to capture system's components (features) and external entities

Process modeling: Data Flow Diagrams (DFD)

- A data flow diagram (DFD) shows how **data moves** through an information system but does **not show program logic or processing steps**
- **Levels of modelling**
 - Context Diagram
 - High level (little detail, completeness of structure)
 - Low level (more detail, decomposed to smaller parts)

System Outline

- The first step towards identifying the components for the process driven model is to complete a System Outline.
- A System Outline identifies the system **inputs, outputs, processes, files (data storage) and external entities**. The processes are the events that occur in the system.
- The details from which the system outline can be derived, will be obtained by interviewing the users throughout the analysis and design phases of the **Systems Development Life Cycle**.

System Outline

Sample system outline. A large collection of these will be created to meet the needs of a complex system.

System Outline

Title	System	Document	Name	Sheet
Input				Processes
Files (Datastores)				Outputs
External Entities				
Author			Date	

DFD Notations

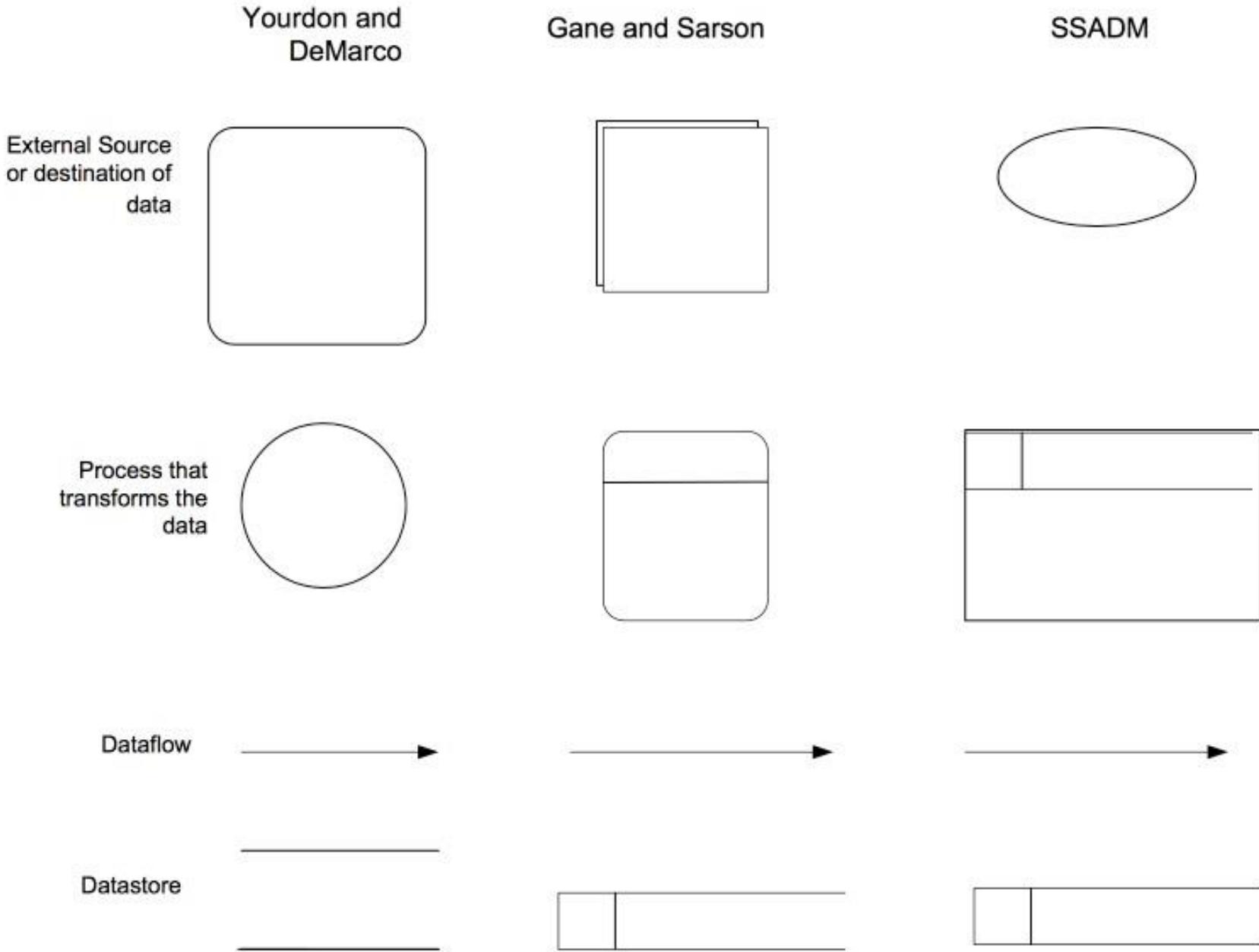
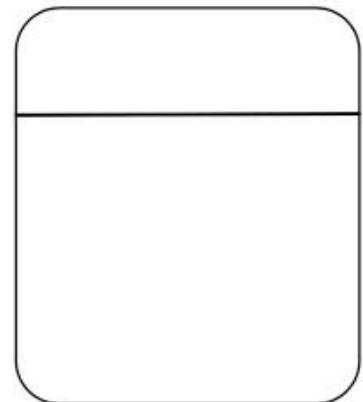


Figure 2.5: Various DFD notations

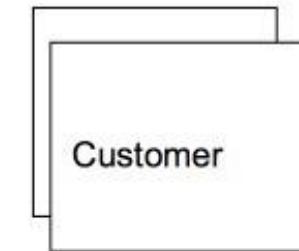
Data Flow Diagrams(DFDs) – Process Symbol

- Drawn as **rectangle with round corners**. Each process must be numbered
- Name of the process starts with a verb followed by a singular object e.g. calculate gross pay, produce invoice, validate customer
- Receives input data and produces output that has a different content, form, or both
- Contain the business logic, also called business rules
- Referred to as a black box



Data Flow Diagrams(DFDs) - External Entities

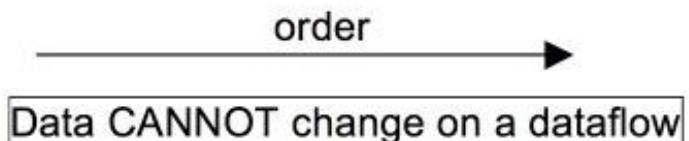
- Provide input or receive output from the system
- **Source** (provides data [input] to the system) or a **Sink** (gets the data [output] from the system)
- Drawn as a **square with another square behind it**
- E.g. Customer, Supplier, IRD, Employee etc.



(a) **External Entity** is a source or sink for data. This is drawn as a square with another square behind it. Examples: Customer, IRD, Employee, Supplier

Data Flow Diagrams(DFDs) – Data Flow

- Drawn as an **arrow**. Arrow head indicates the direction of the flow
- Dataflow name is singular, written on top of the arrow
- E.g. Invoice, payslip, timesheet, order



Data CANNOT change on a dataflow

(b) **Dataflow**. Drawn as an arrow. The arrowhead indicates the direction of the flow. The dataflow name is usually singular. Examples: invoice, payslip, timesheet, order. We are only concerned with the data content of the flow — NOT with the number of copies of the data.

Data Flow Diagrams(DFDs)

- **Data store (storage)**

- Drawn as an **open-ended rectangle**
- Data is stored for use by a process at a later time
- Data storage also provides input into another process
- **Name of data store is usually plural**
- Each data store is usually numbered for reference purpose e.g. D1
- **Customer master file named as customers, employee master file named as employees**

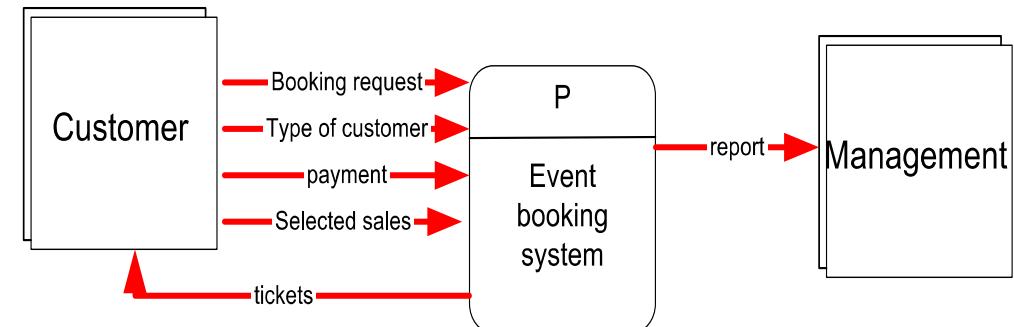
- **Duplicates**

- Sometimes it is necessary to repeat the a symbol on a DFD in order to avoid crossing lines



Drawing context diagram

- Draw the context diagram so that it fits on one page
- Has a **single process** representing the entire system. The name of the process is name of the system
- Identify **all the external entities** to go with the process.
- Shows the system as a single process with external entities
- External entities show all the input and output they provide and receive from the system
- Context diagram helps the analyst to gain an overall view of the system he/she is investigating



DFD Modelling

1. DFD's

Enrolling in the University

<http://agilemodeling.com/artifacts/dataFlowDiagram.htm>

2. Blog Case – Digital Government

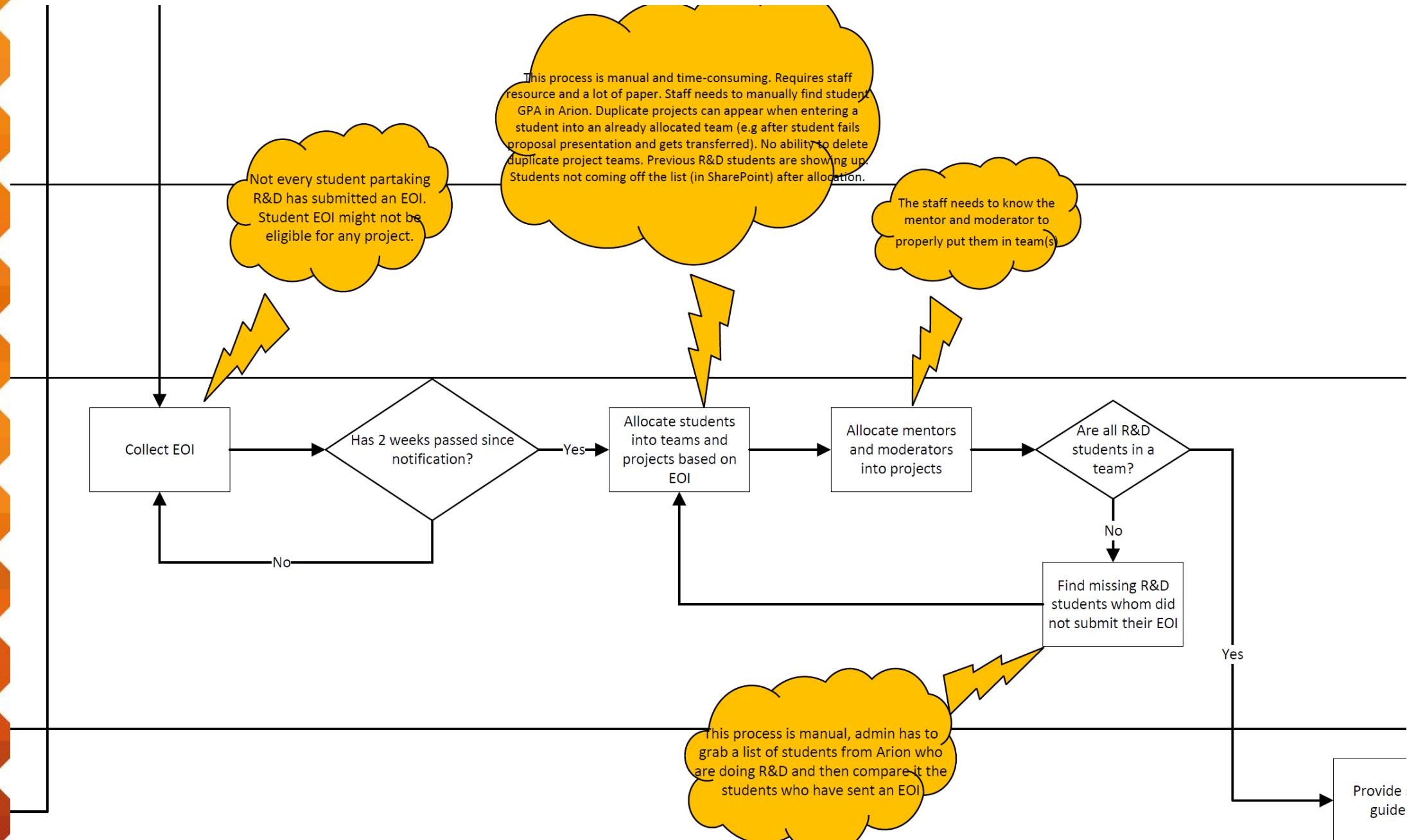
The Service: Applying for NZ Superannuation

<https://www.digital.govt.nz/blog/user-focussed-content-equals-good-business/>

DFD Modelling – BCIS R&D Project - S2/ 2021

1. Consultation Report
2. Pain points – to see next slide
3. Discussion and issues
4. Options and Implications

DFD Modelling – BCIS R&D Project - Pain Points



Tutorial Session 6: Modelling Dataflows (1)

THE STATIONERY STORES SYSTEM

The Stationery Stores System is a case study used by Park Place Training (see References) to teach the general principles of structured analysis and design. For the purpose of illustrating the approach to Mk II function point counting, we will only use a small part of the overall case study. This part concerns the processes from the time when the Stores sees the need to re-order some stationery, through the selection of possible suppliers based on past performance, the preparation of requests for quote, and their issue to suppliers, and the receipt and registration of quotes from suppliers.

In the case study, we are at a point where Data Flow Diagrams ("DFD's") have been produced for the required processes (referred to as a 'First Cut Functional Design'), data entity analysis has been carried out, and the

Tutorial Session 6: Modelling Dataflows (2)

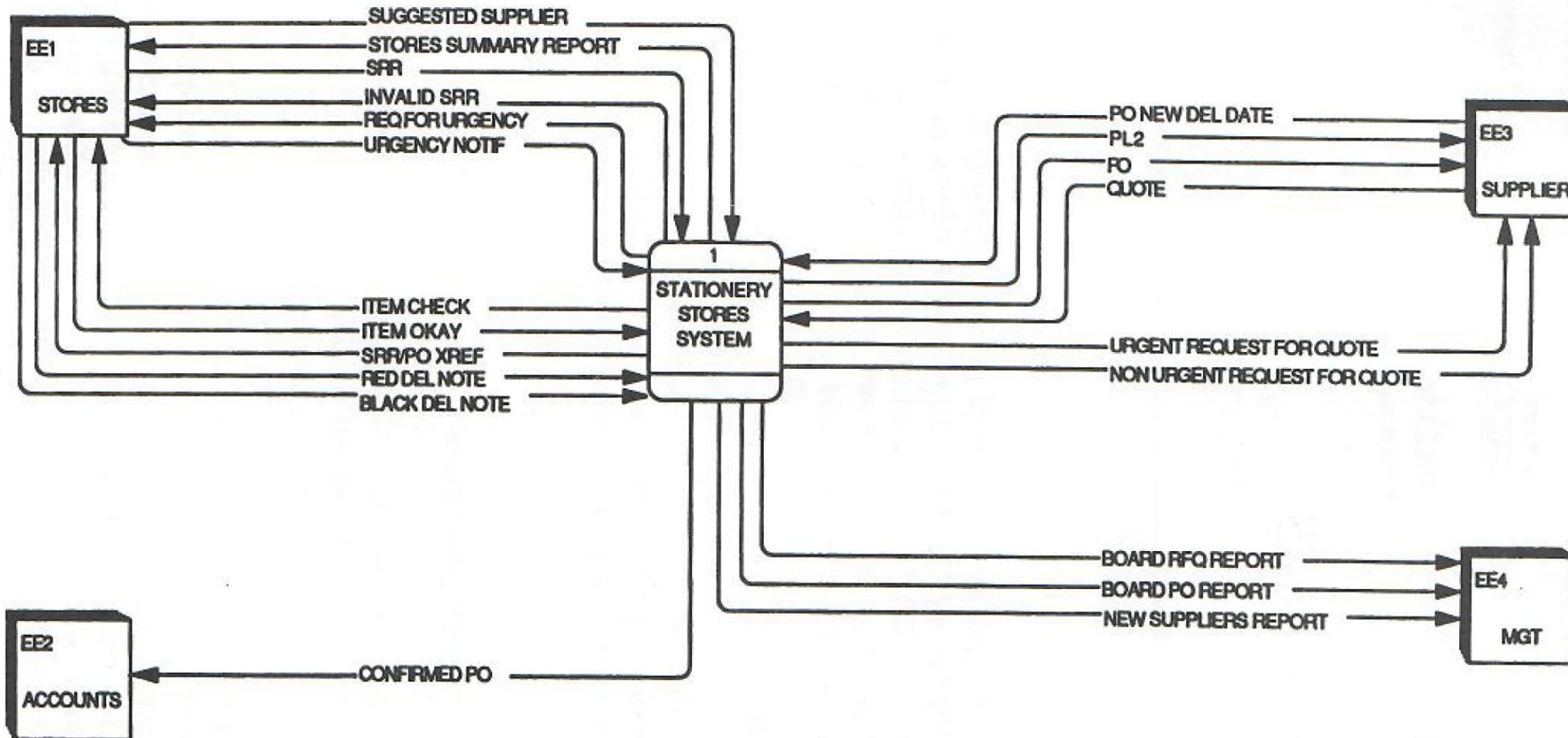


Figure 26 Stationery Stores First Cut Functional Design—Context

Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

Tutorial Session 6: Modelling Dataflows (3)

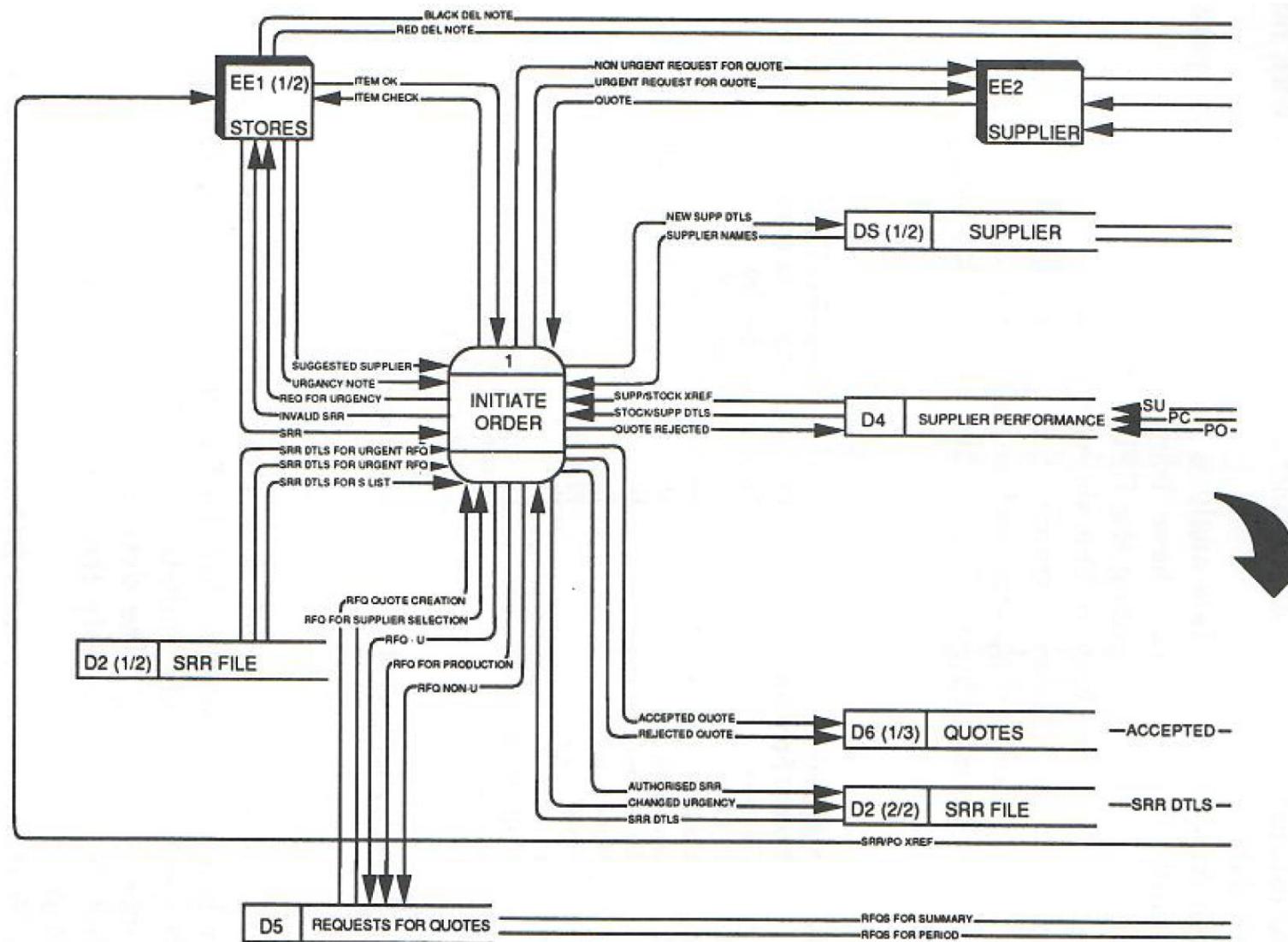


Figure 28(a) First Cut Functional Design—Level 0

Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

A large, abstract graphic on the left side of the slide features a repeating pattern of orange triangles. These triangles are arranged in a way that creates a sense of depth and movement, resembling a stylized flame or a rising sun. The colors range from bright orange at the top to darker shades of orange and red towards the bottom.

Extend your knowledge

1. Watch a video on – what is an Information System
<https://youtu.be/Qujsd4vkqFI>
2. Explore job information about Business / System Analyst
<http://tinyurl.com/ztzulel>

References

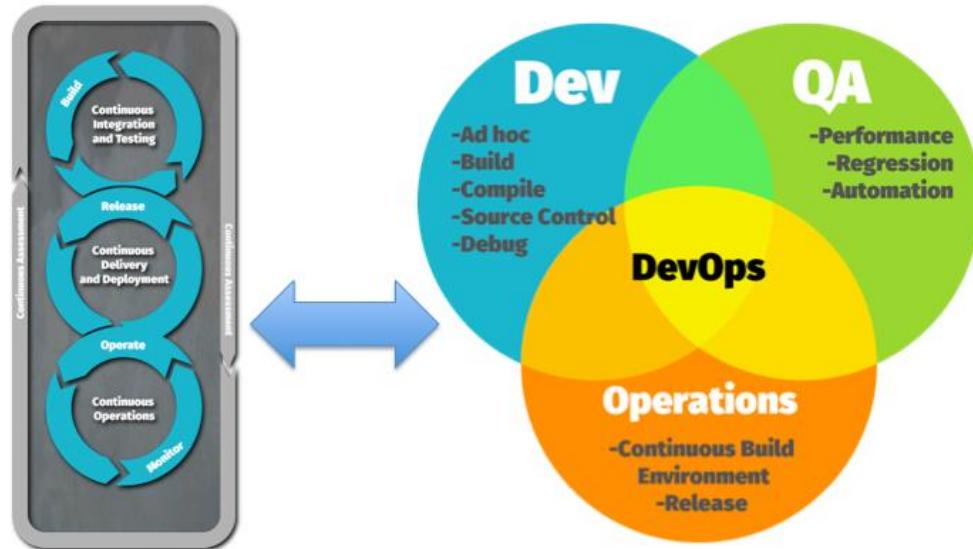
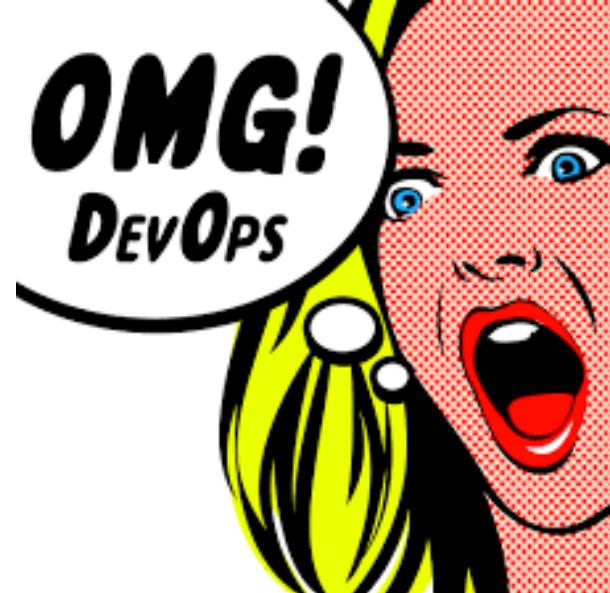
- Baecker, R. M. (2019). *Computers and society: Modern perspectives*: Oxford University Press, USA.
- Kaczmarczyk, L. C. (2016). *Computers and society: computing for good*: CRC Press.
- Stair, R., & Reynolds, G. (2020). *Principles of information systems*: Cengage Learning.
- Litchfield, A. (2017). *INFS500 Enterprise Systems - Bachelor of Computer and Information Science: Process Modelling Workbook*. Auckland: Auckland University of Technology.
- Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

Week 12

Course Review

Empirical Software Engineering

The DevOps Way of Working

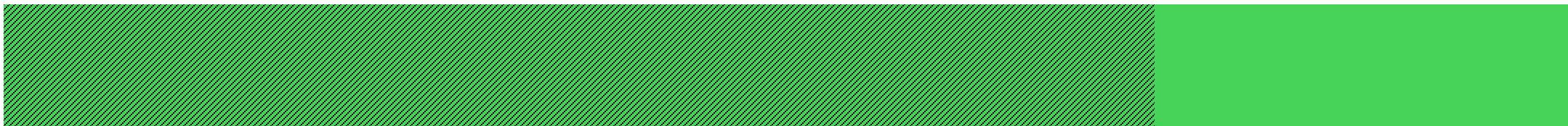


Taking Stock

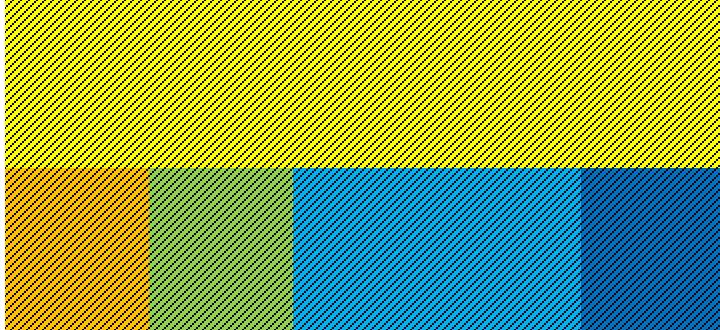
Week
No

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

Review
Questionnaire

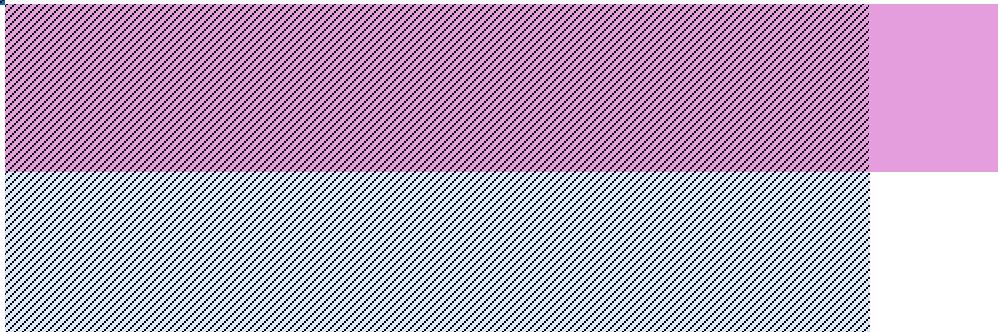


Assgt 1A -
Techstack



Worksheets

Assgt 1B - Team
Project



Iteratio
ns

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

CI/CD
PAIR and MOB
TDD

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?

What potential requirements are embedded in user reviews? Machine learning and text processing.

cyber-physical systems

How can team "health" be monitored and diagnosed?

In 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.

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^(✉), 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

Waqar 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

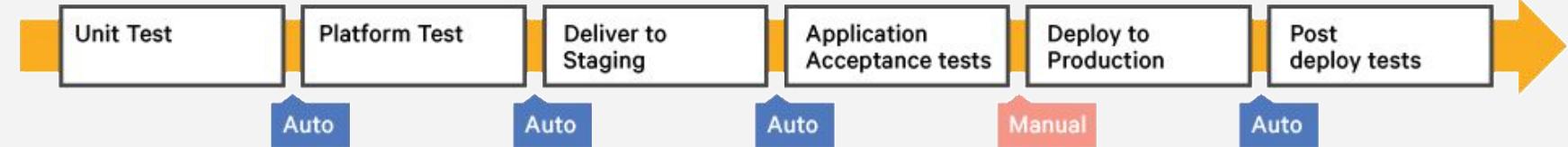
Daily standups

Sprint planning

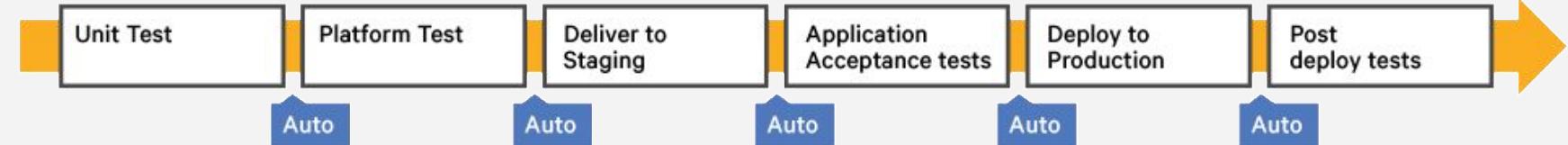
Sprint review

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

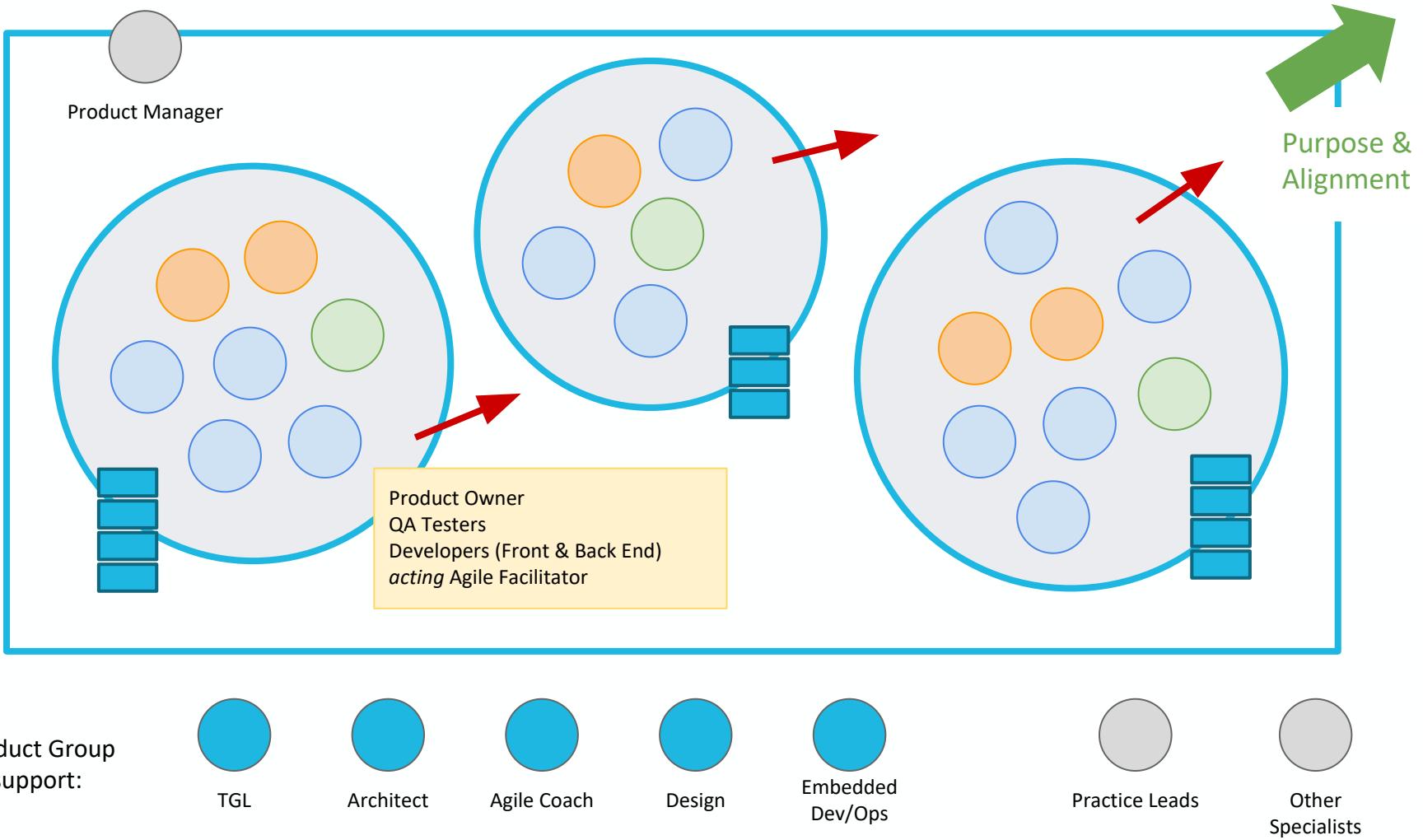
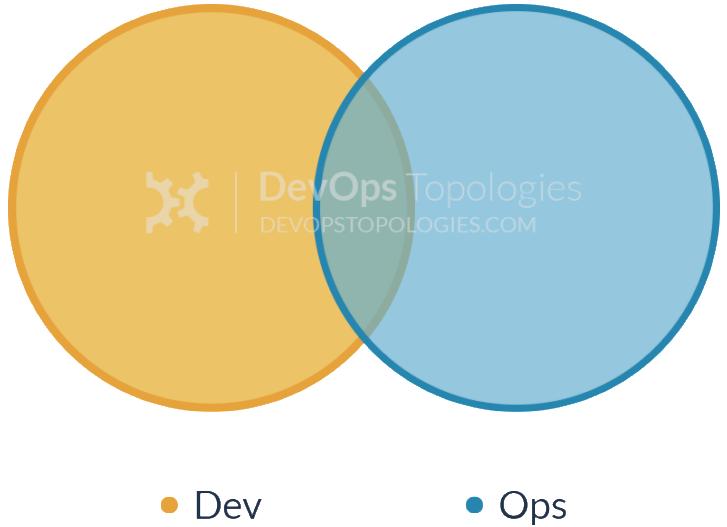


Diagram Used with permission of Hady Osman

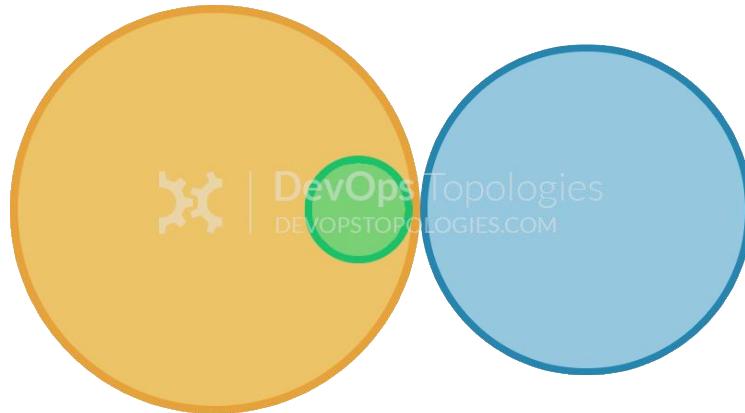
Team structures – patterns and anti-patterns



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

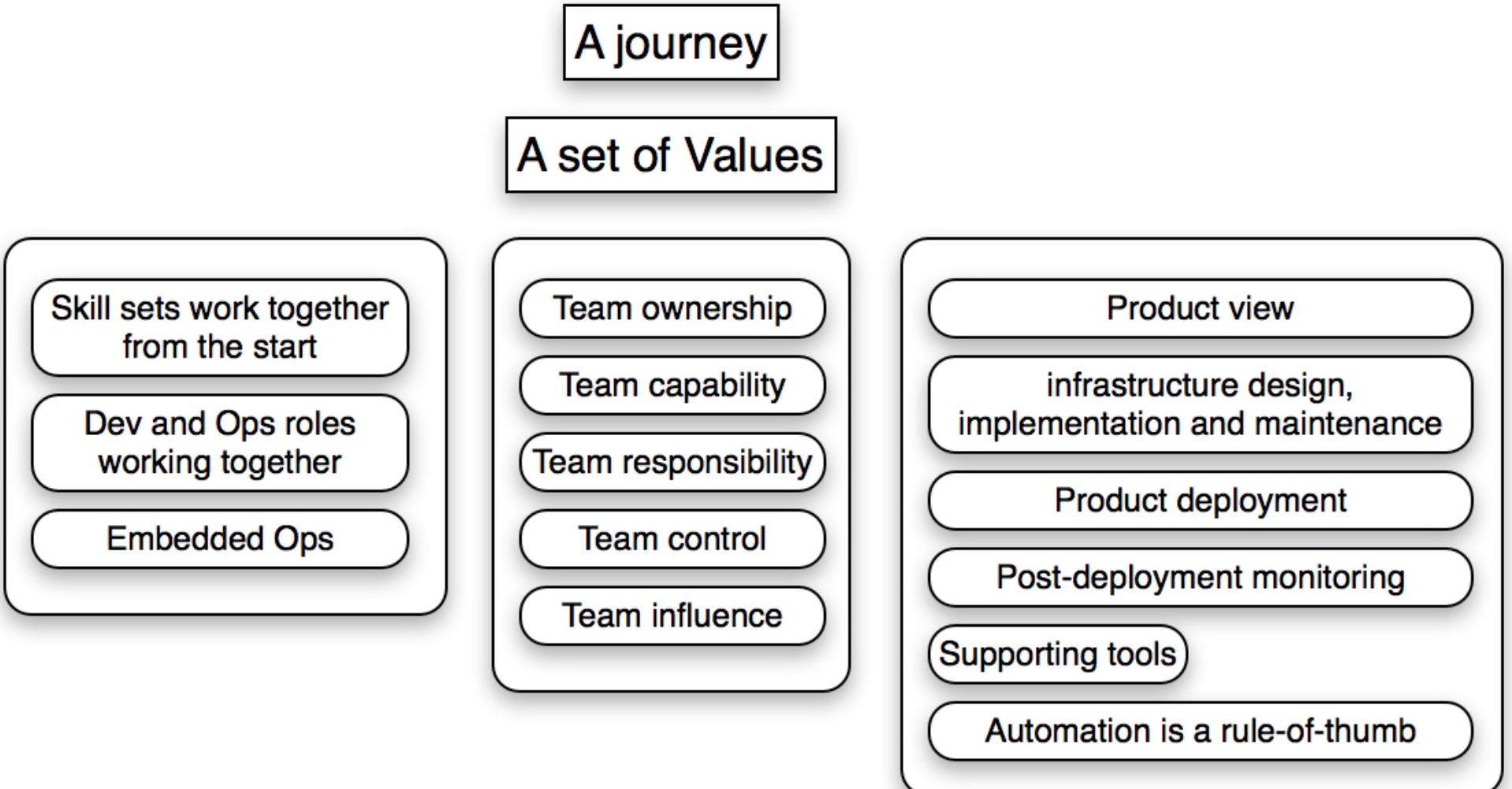
Reduce risk

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

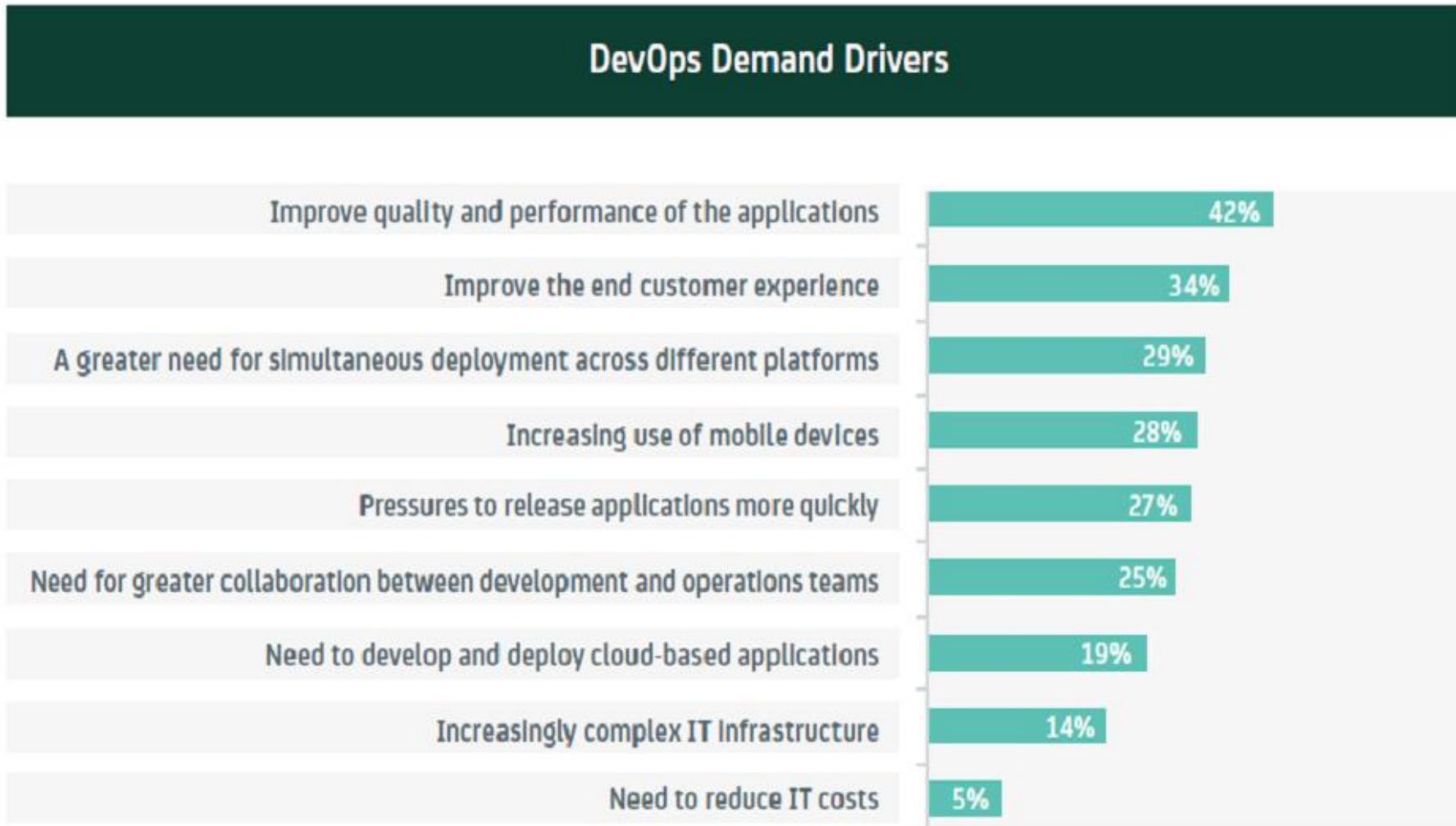


Figure 2 what is driving need for DevOps?

[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**.

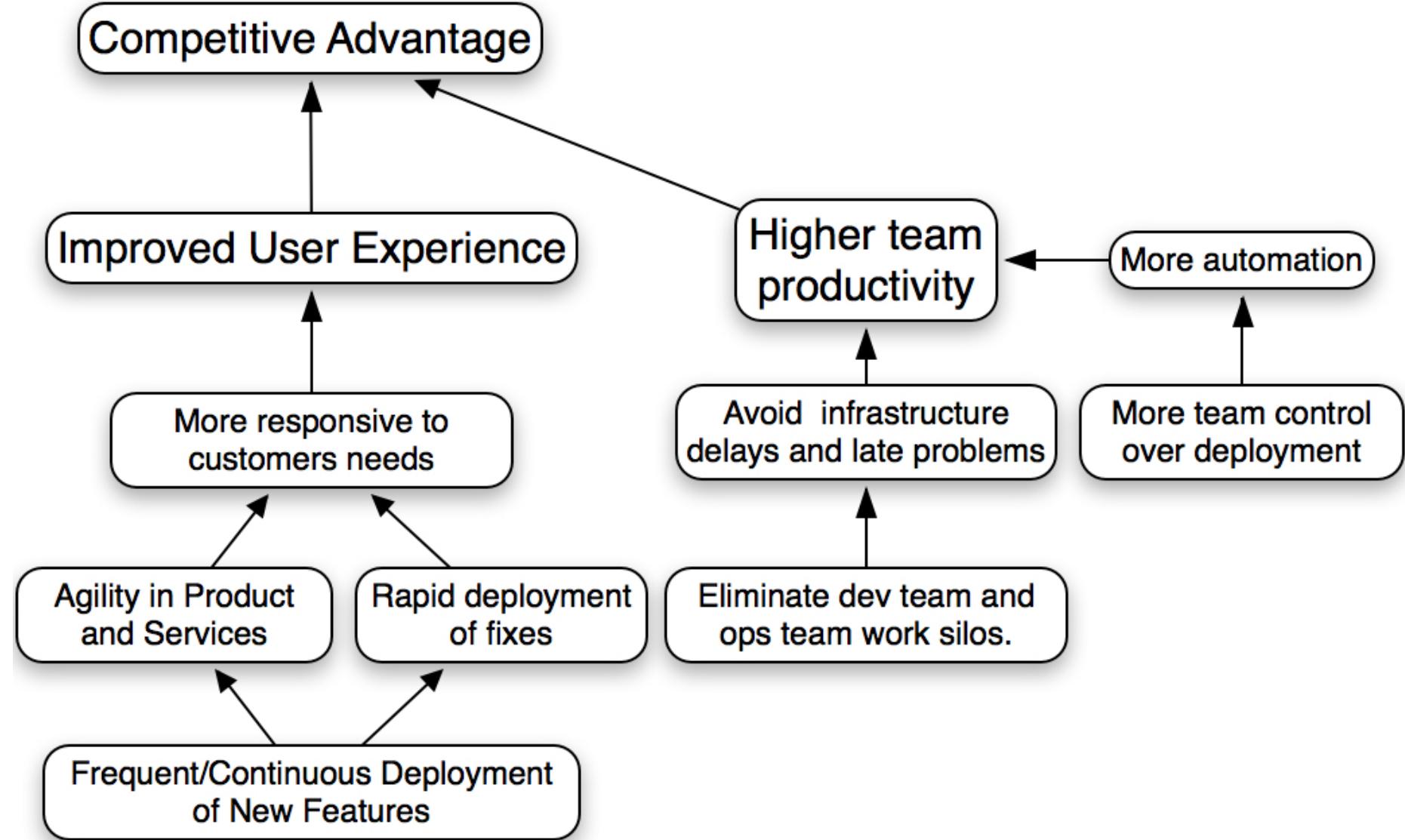
Operation teams were accountable for performance and uptime, development teams were in a better position to improve it

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)*

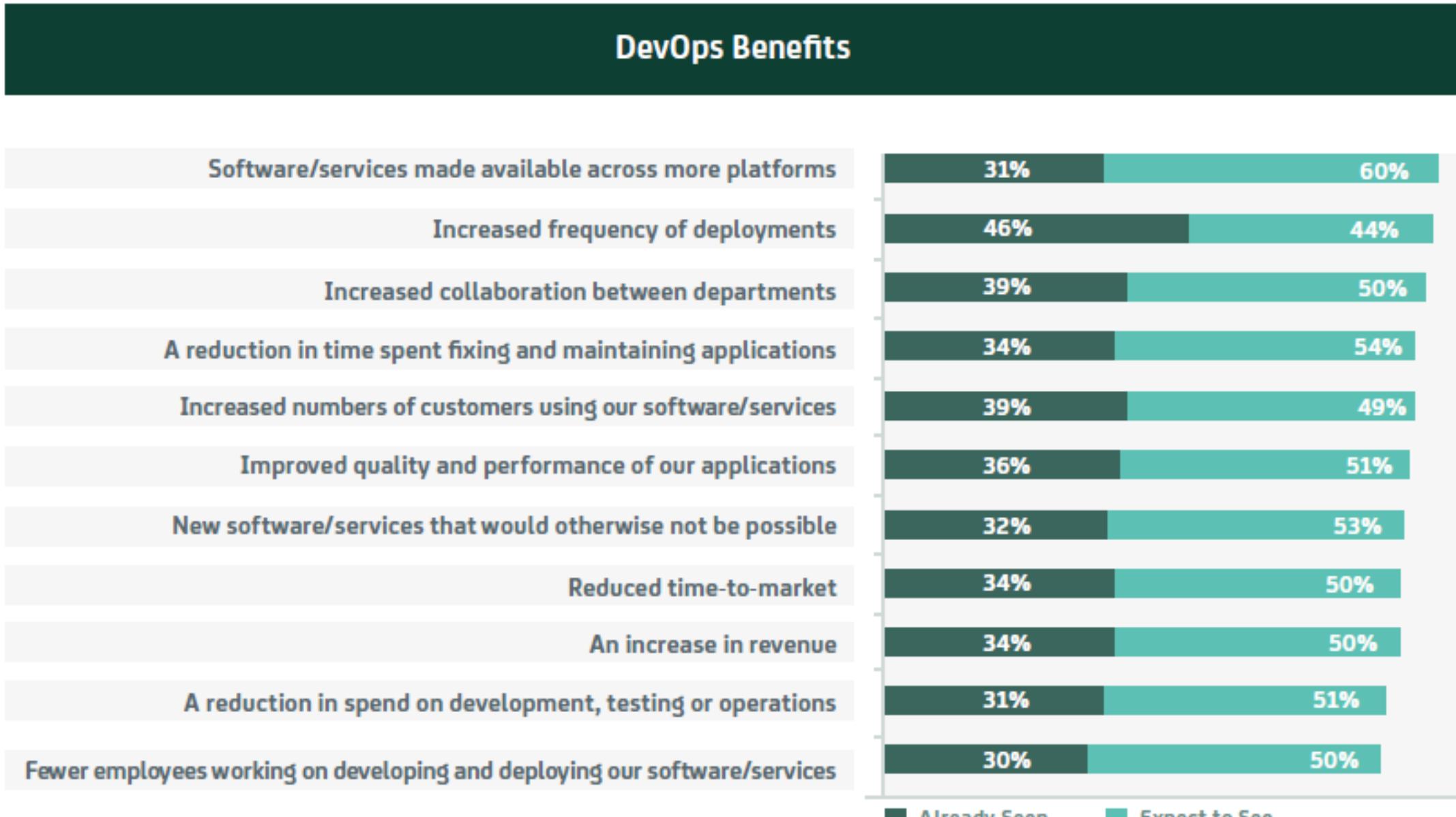
You can build so much better integrity. You build your own solutions that work for your own environment. *(Embedded Ops)*

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]

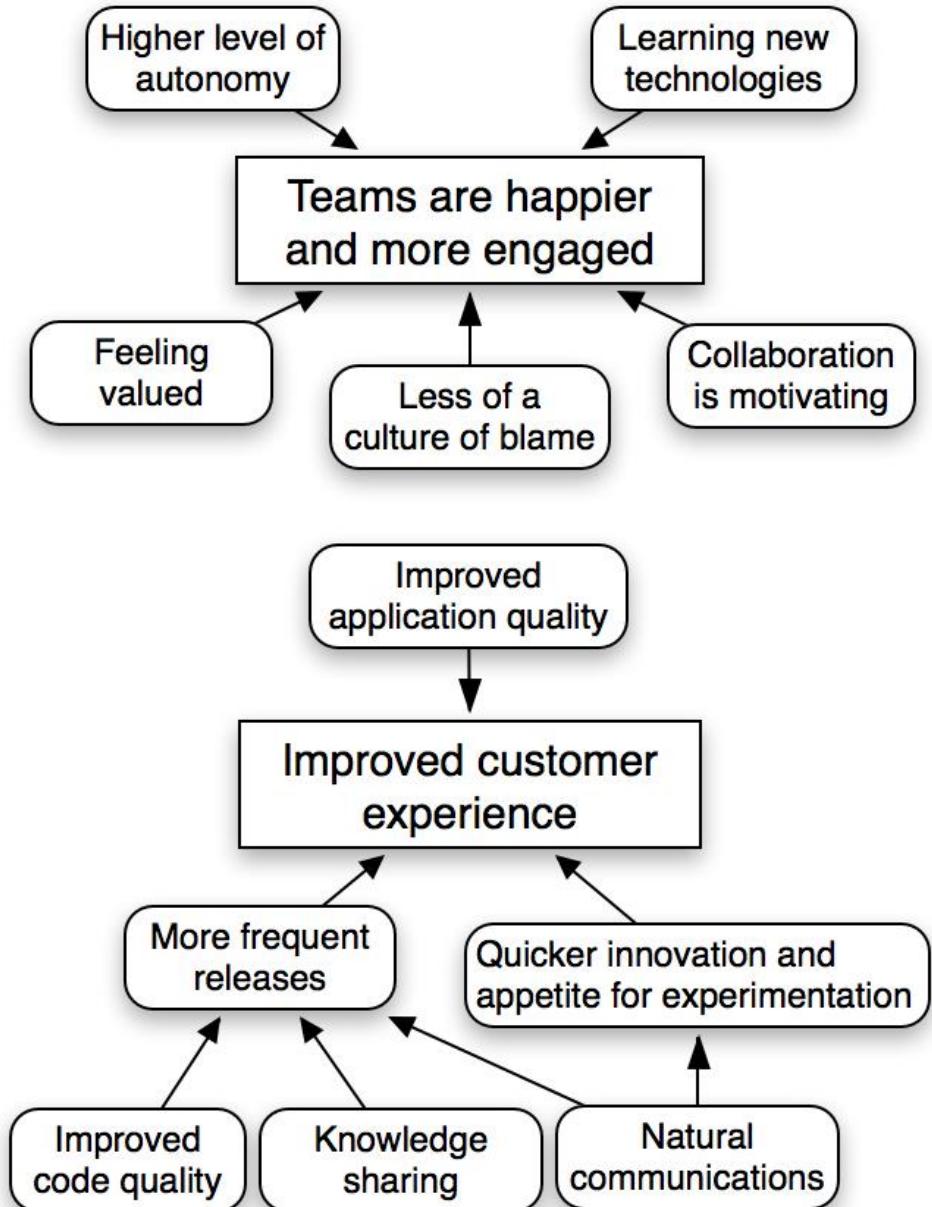


Benefits of DevOps Adoption

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?

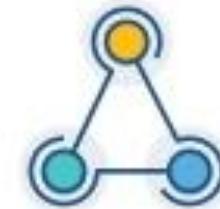
What capabilities, culture, practices, technologies and tools are significant enablers of successful DevOps adoption?



Agile and Lean



Automation



Service Frameworks



Enablers and success factors

Trusting the production team to deploy was difficult (QA Release manager)

Lots of communications – you can never have enough. (Developer)

The switch from a monolithic application to microservices was a big thing. (Team lead Infrastructure)

The team can set up their own dashboard for monitoring whatever they think is important post-deployment of a feature. (Ops in Team)

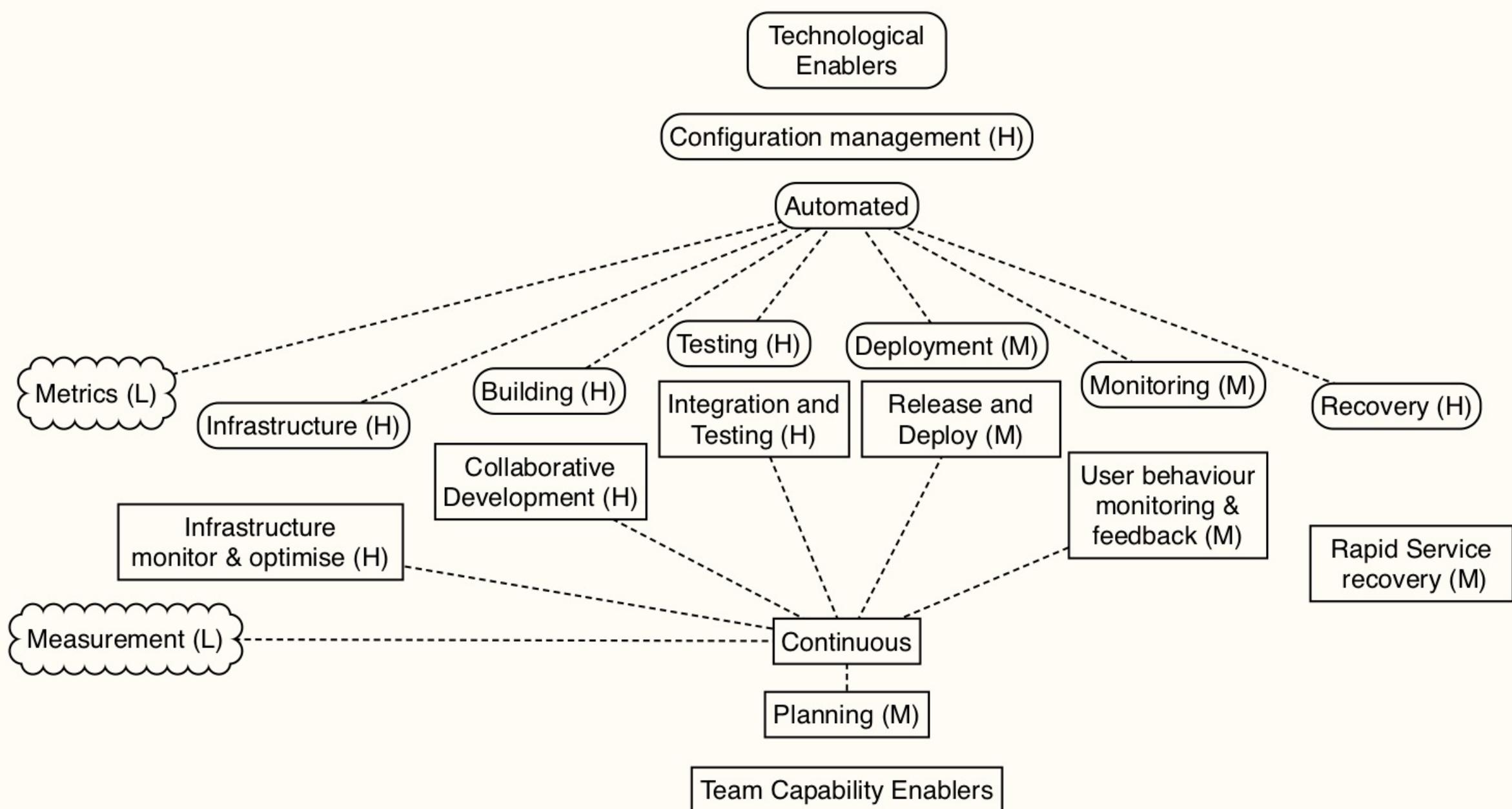
Cross training the development team on ops and having ops take part in the production team meetings. It's hard just getting the right people, though. (Training manager)

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)

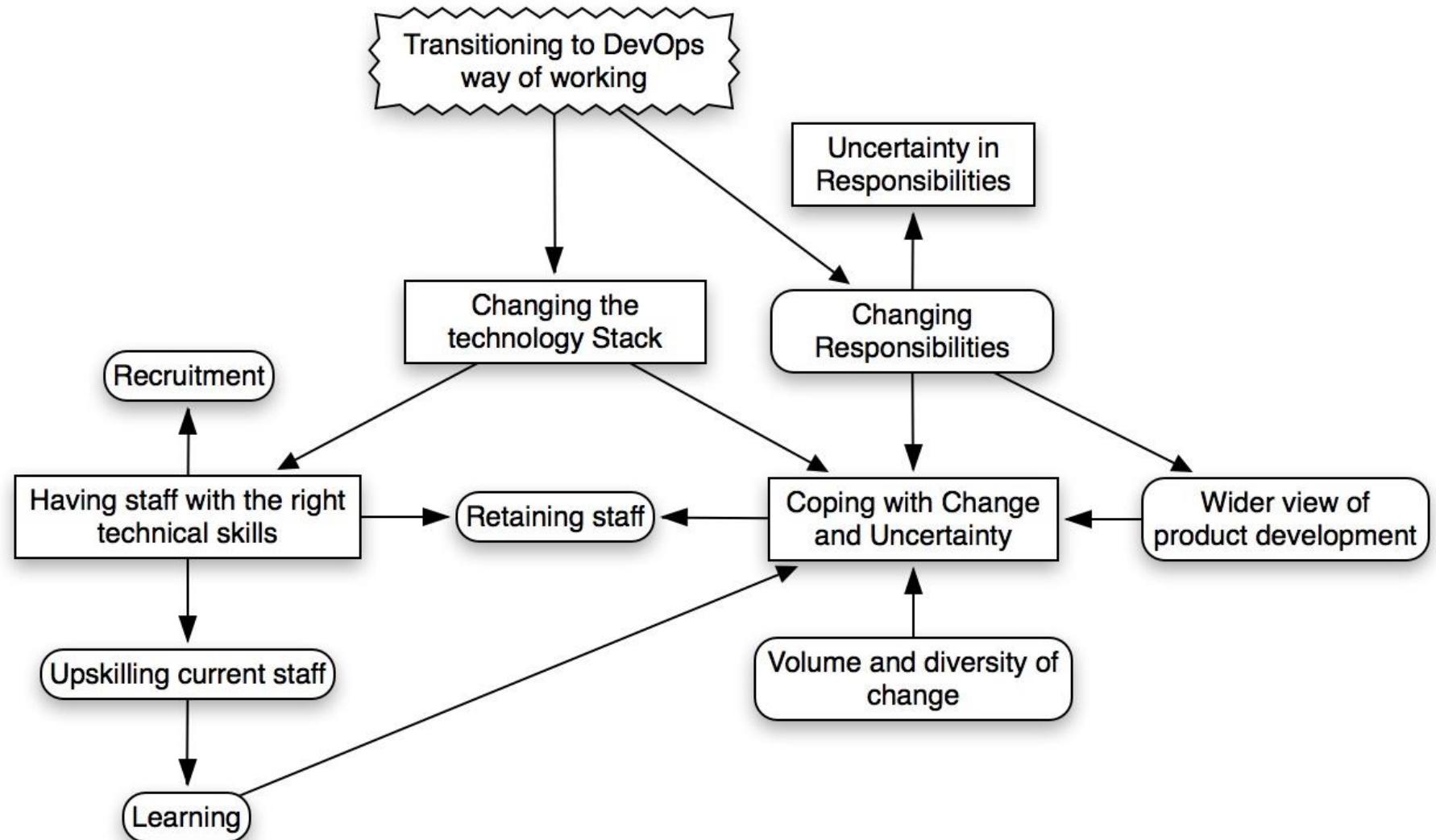
Changing the technology stack to the cloud and micro-services was a big enabler, but it was incredible complex. (Team lead Infrastructure)

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)

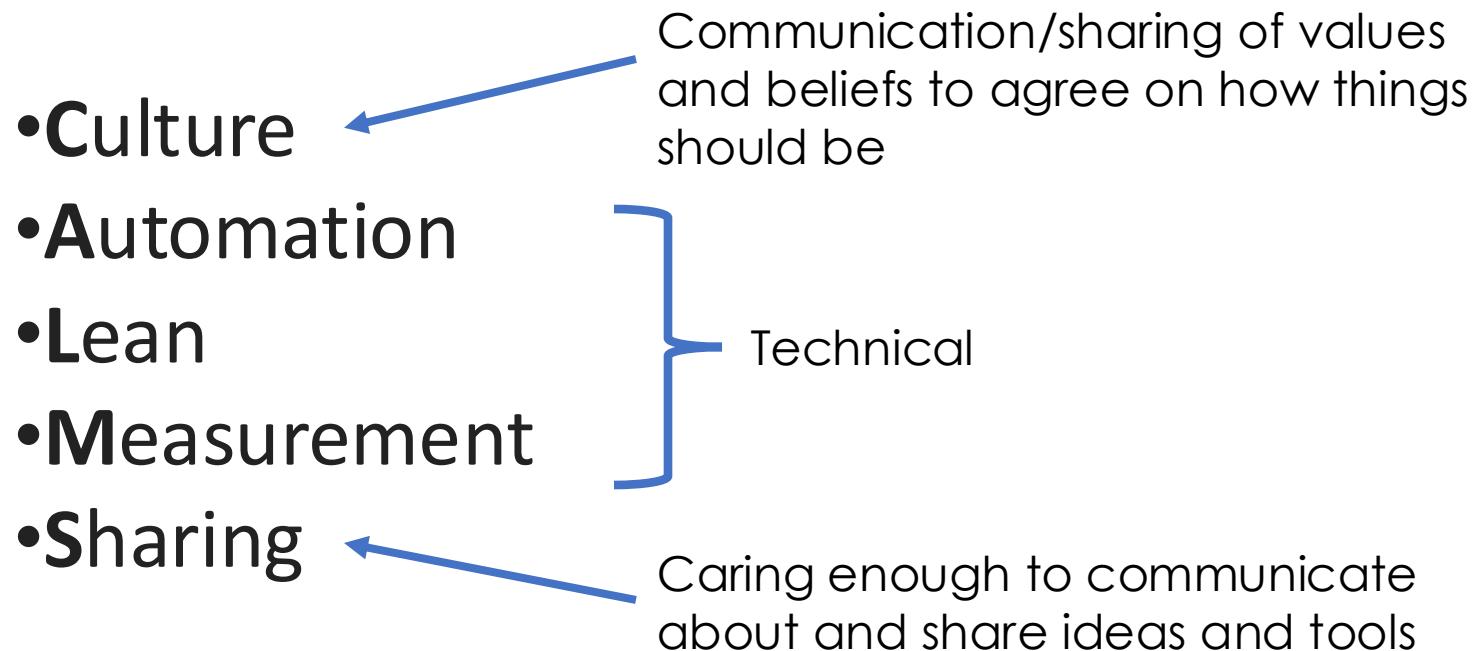
There are so many new tools and ideas, it's hard just keeping up. (Tester/QA)

Challenges in working the DevOps way



DevOps principles: Keep your CALM(S) first (then select some tools to support this...)

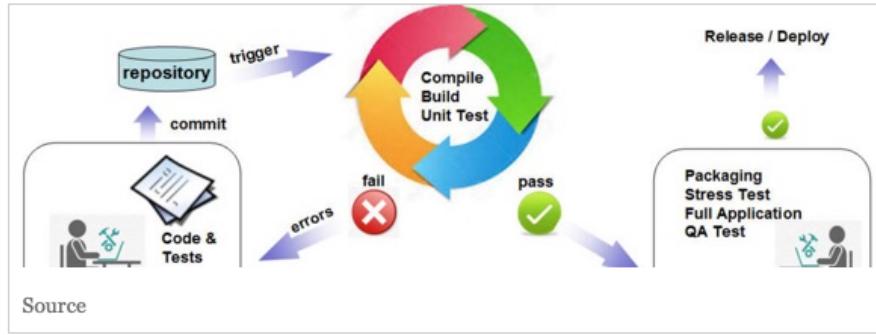
CALMS is a very accurate description of how to practice DevOps:



<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.



#171478945



Thank you!

Questions and Comments....



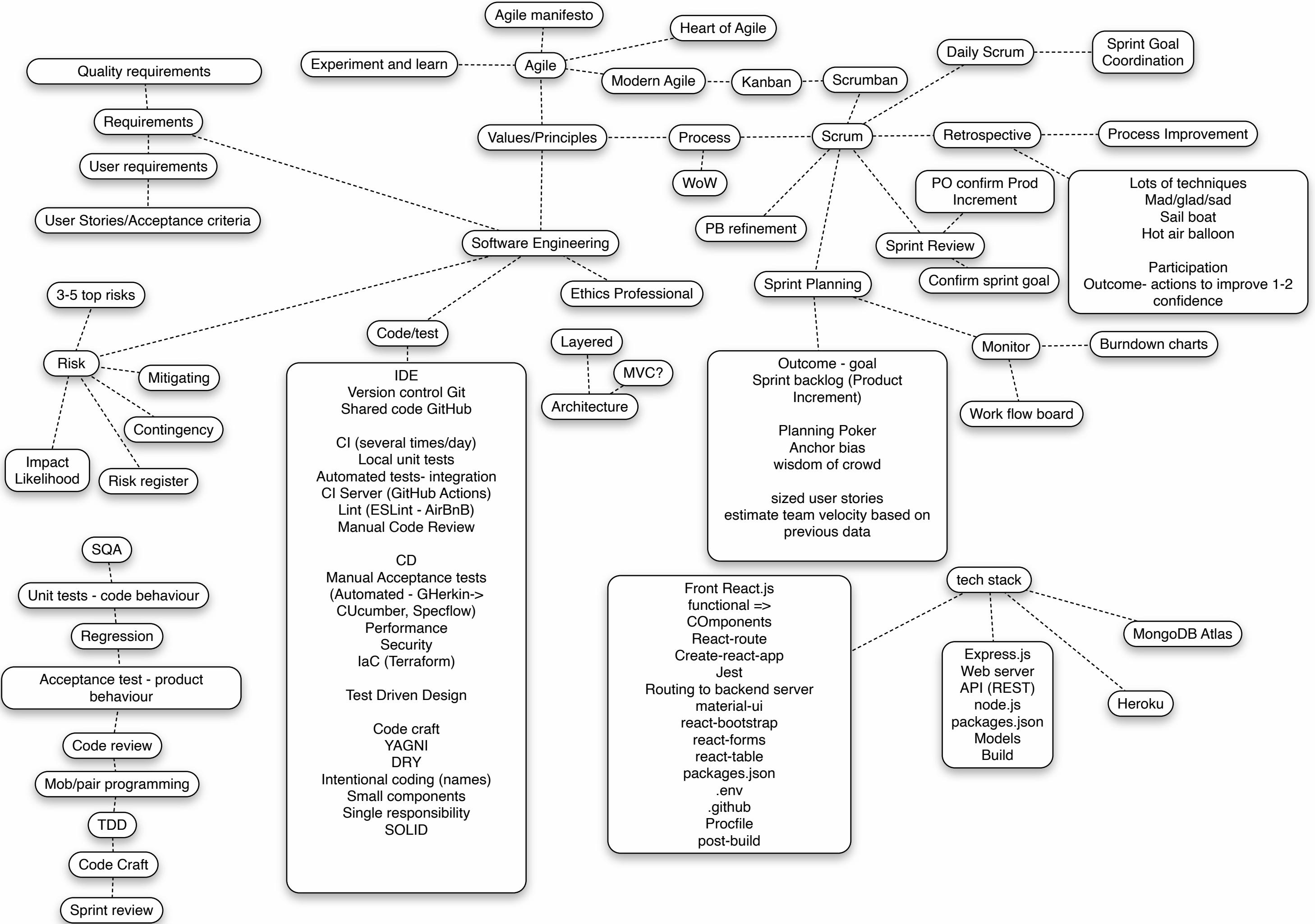
Tony Clear 2024

CISE ENSE701

I has a question...



50



Agile is eating the World



Why and What does it Even Mean ?

Jan 2, 2018, 07:35am EST

Forbes magazine

Why Agile Is Eating The World



Steve Denning Senior Contributor
Leadership Strategy

I write about 21st century leadership, Agile, innovation & narrative.

The Wall Street Journal

Why Software Is Eating The World

By Marc Andreessen
August 20, 2011

The meaning has got confused...



70 different Agile practices

Mob programming

DevSecOps

Fast Agile

ShapeUp!

<https://craigsmith.id.au/2015/12/03/yow-2015-40-agile-methods-in-40-minutes/>

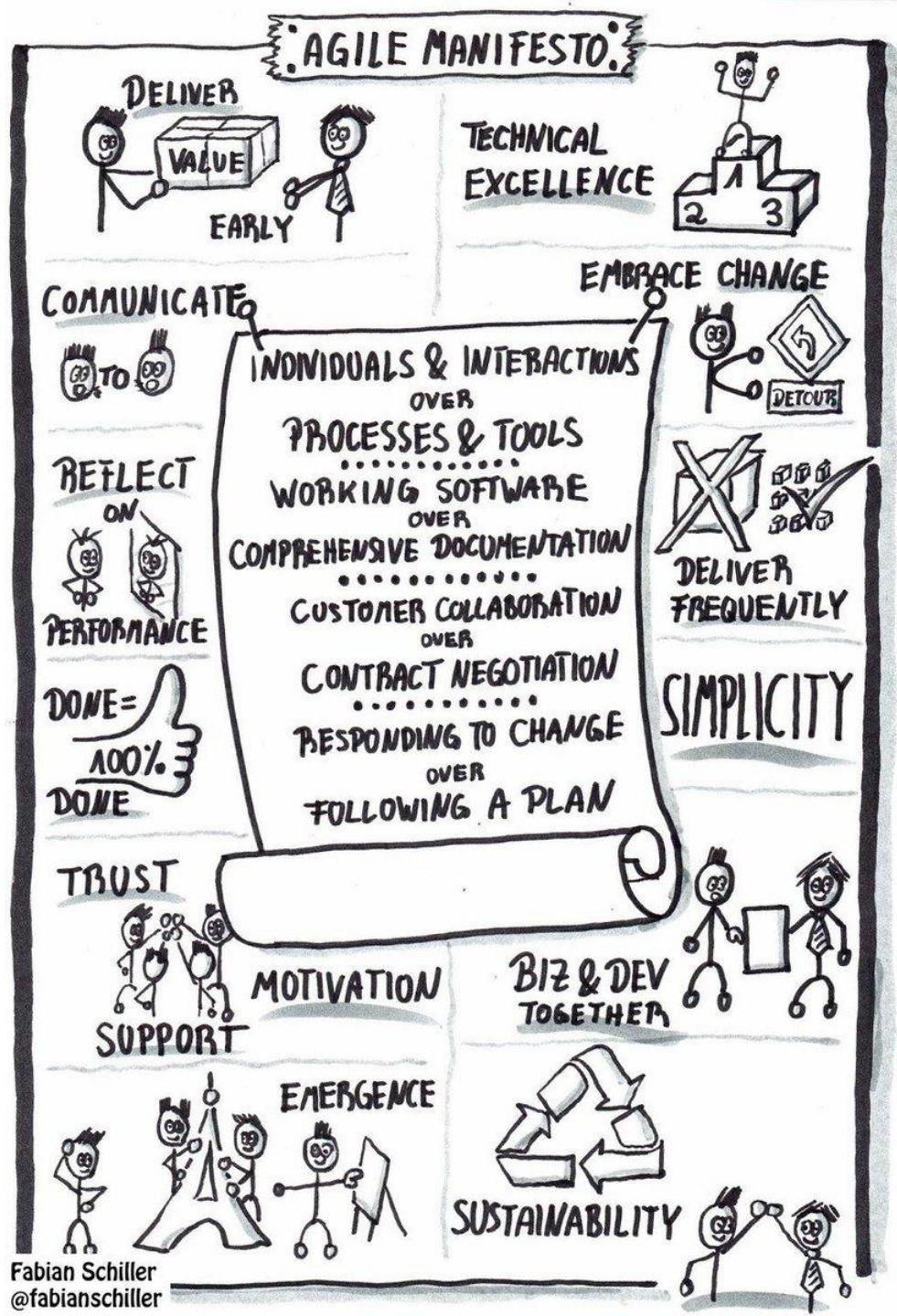
Agile Manifesto

This describes a set of values, beliefs and principles

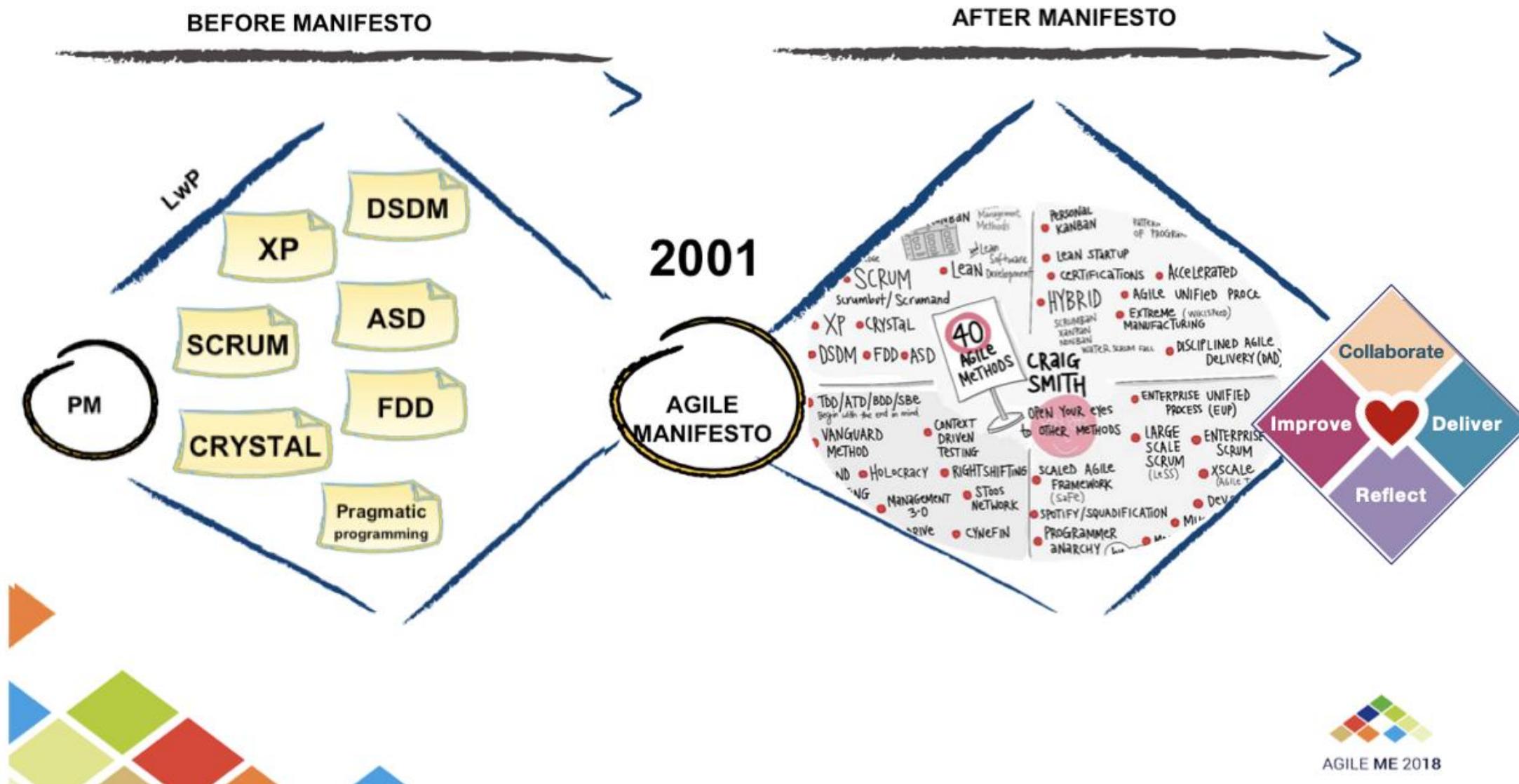
A Mindset....

That guides how to behave, interact and design processes to do work

Focus on Software Development Work



Pierre Hervouet's recap of history:



Let's simplify it....the heart of Agile

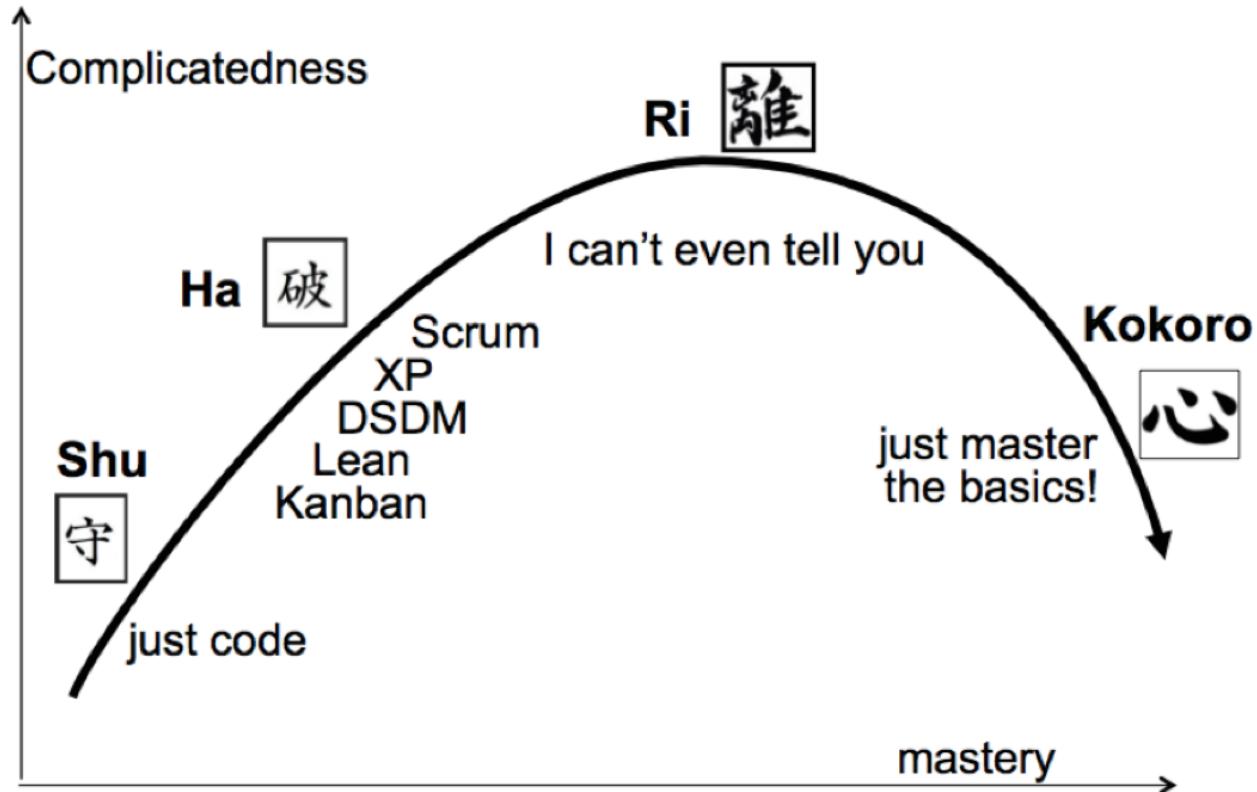


Figure 1. The Shu-Ha-Ri-Kokoro progression.

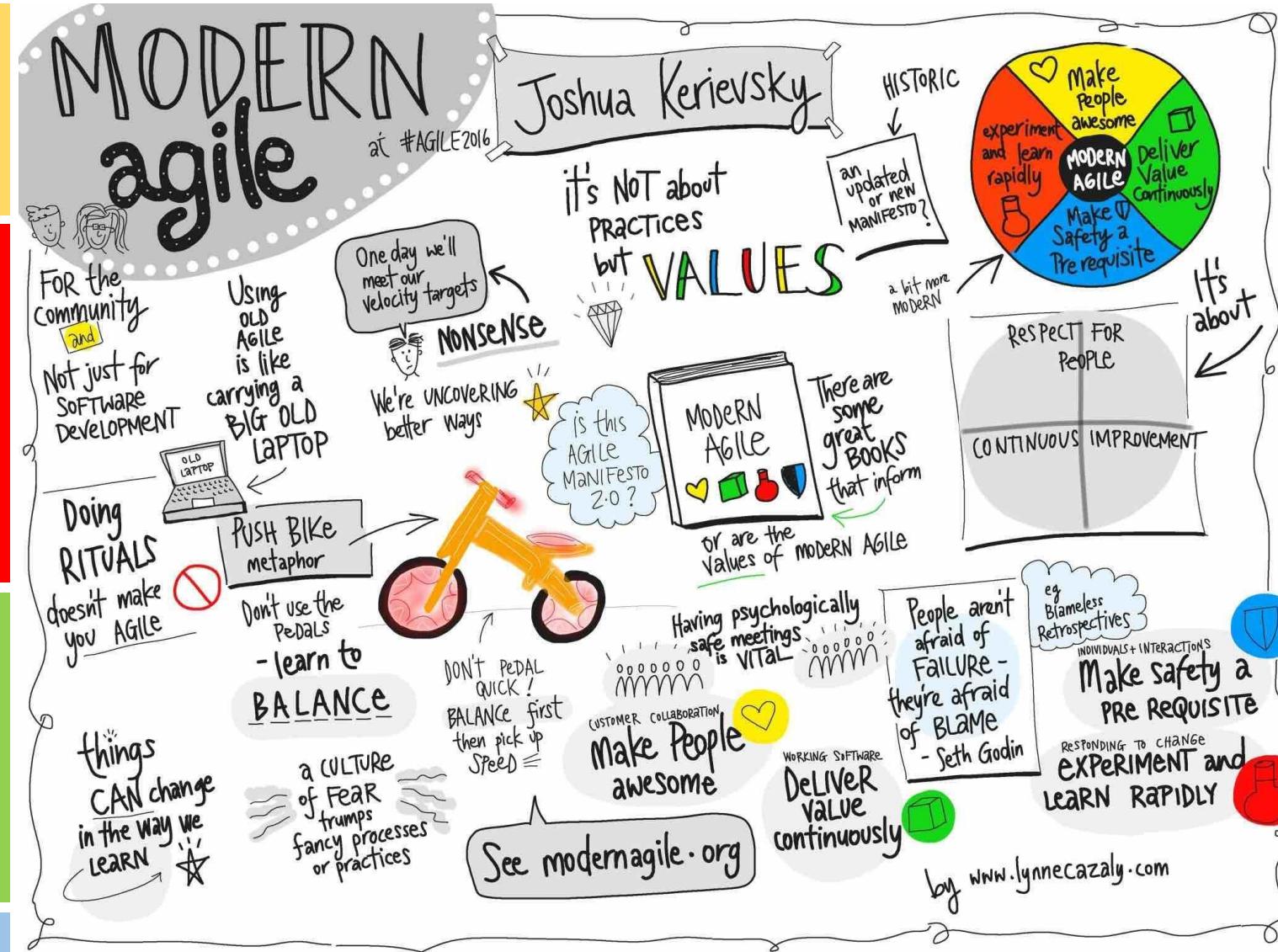
<https://alistair.cockburn.us/wp-content/uploads/2018/02/The-Heart-of-Agile-Technical-Report.pdf>

We learn their context and pain points, what holds them back and what they aspire to achieve. How can we make them awesome?

We learn rapidly by experimenting frequently. We make our experiments "safe to fail" so we are not afraid to conduct more experiments. When we get stuck or aren't learning enough, we take it as a sign that we need to learn more by running more experiments.

In modern agile we ask ourselves, "How could valuable work be delivered faster?" Delivering value continuously requires us to divide larger amounts of value into smaller pieces that may be delivered safely now rather than later.

We protect people's time, information, reputation, money, health and relationships. And we endeavor to make our collaborations, resilient and safe.



The goal of Agile mindset, values, principles, practices

“Making money” is not the goal

“Being agile” is not the goal.

“Working software” is not the goal.

Agile & Scrum & working software are means to achieving the goal.

Everyone must focus on the goal

What are your values related to why and how you work and interact and collaborate? What is the GOAL?

To create product or service that delights your customers

What will make it delight them?

It's of value to your clients

What is the value?

The product or service

...makes the world your client lives in and the way they work and interact easier/faster/competitive

Delight a customer =

“Providing a continuous stream of additional value to customers and delivering it sooner”

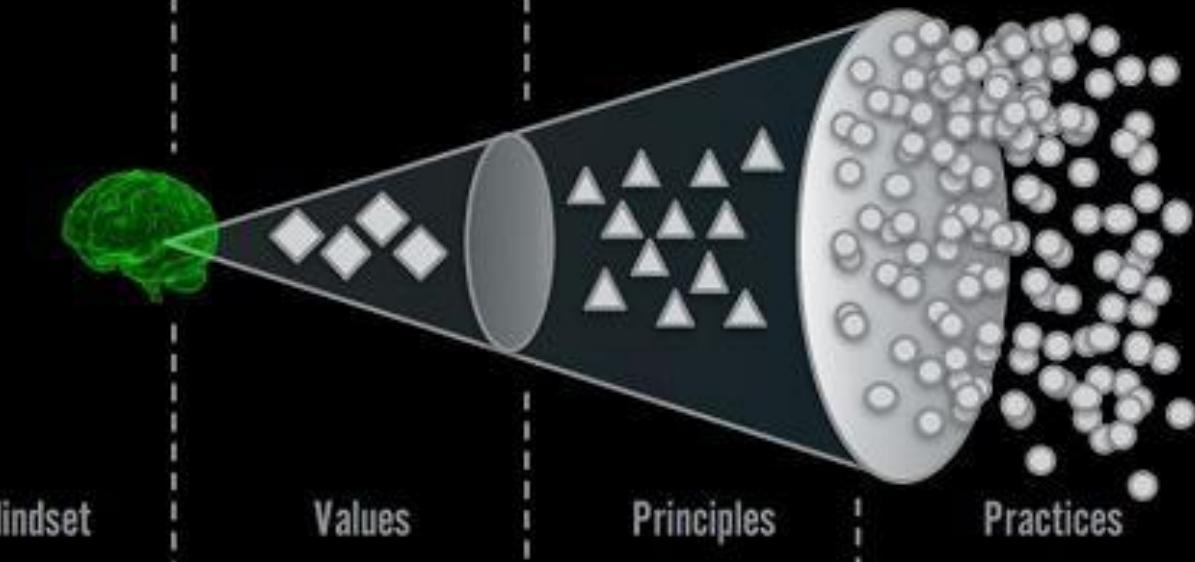
(Denning)

What is the value you create for your customers and do you care?

From mind set to process

WHAT IS AGILE?

AGILE IS A MINDSET DESCRIBED BY 4 VALUES DEFINED BY 12 PRINCIPLES MANIFESTED THROUGH AN UNLIMITED NUMBER OF PRACTICES

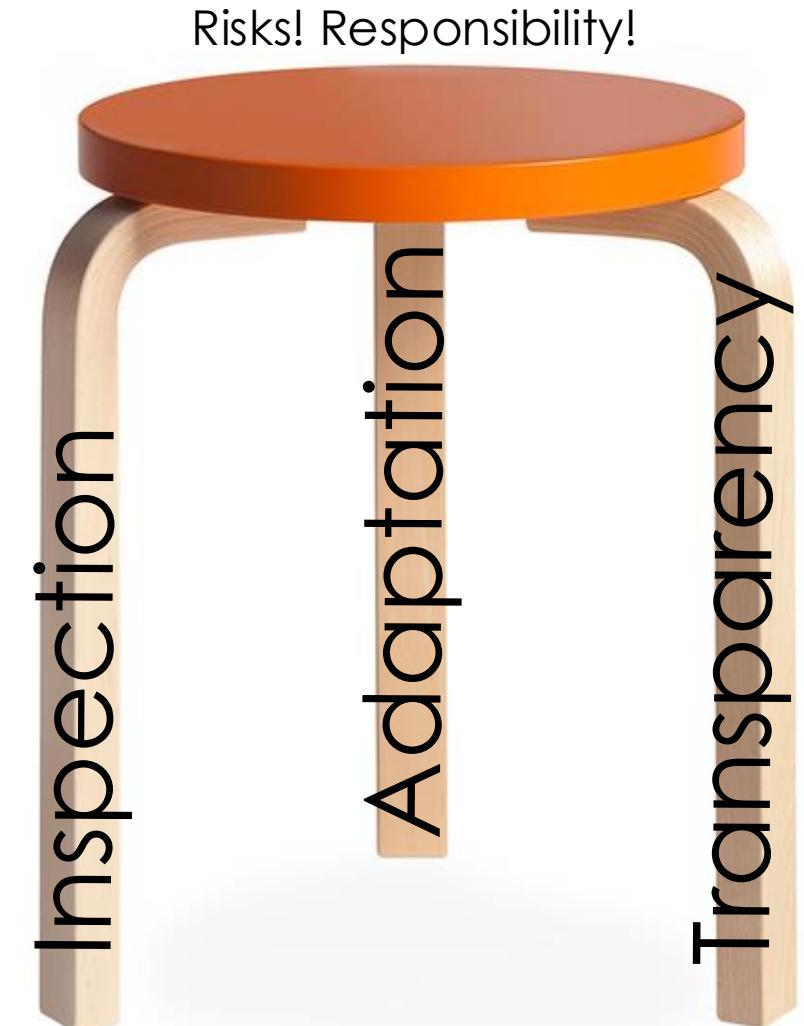


Scrum Values

There is a focus on understanding and making progress towards goals

Product Goal -> Sprint Goals that are about customer value/needs

Agile values lead to an empirical process control control system – Scrum is such a system

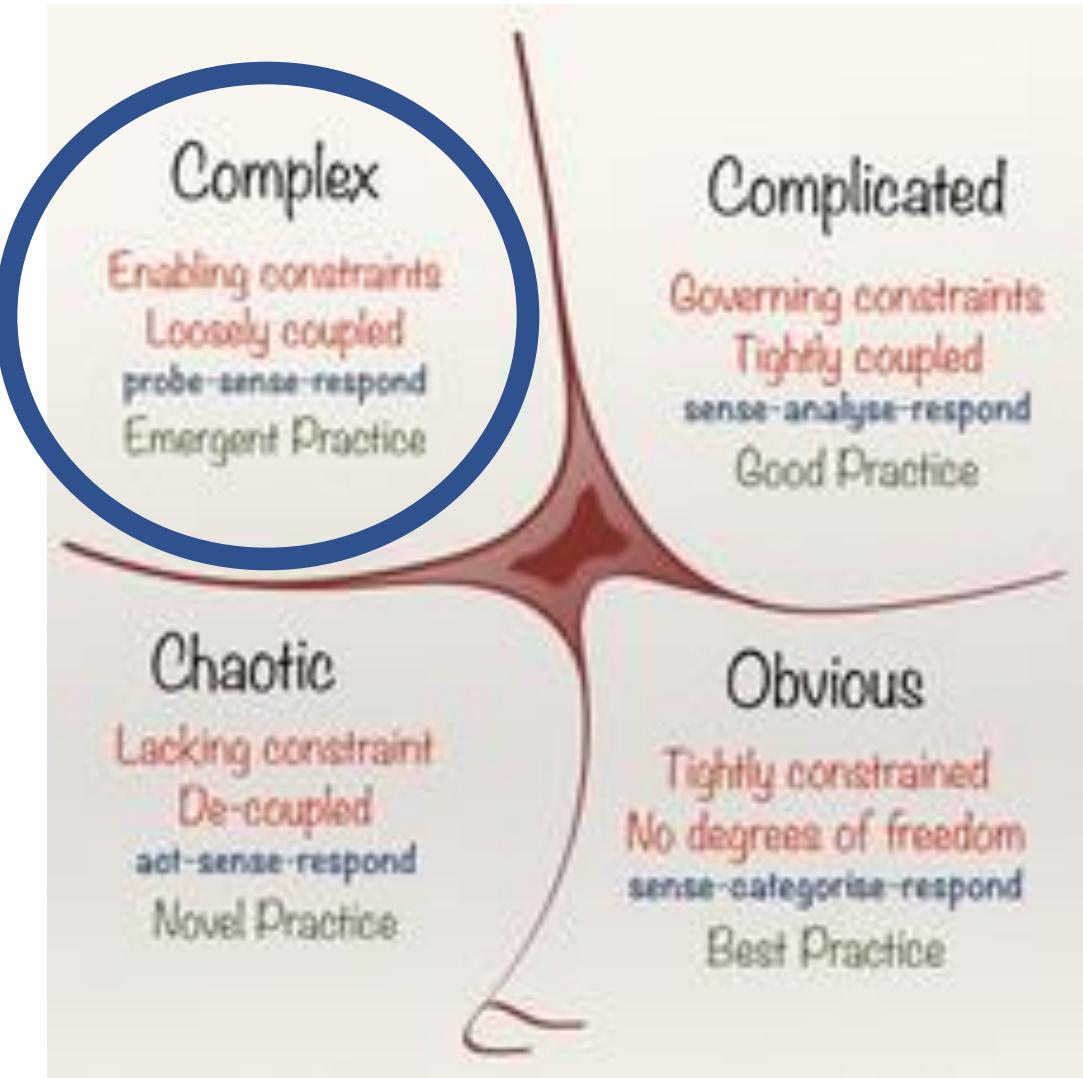


Why the need to be Agile?

Person
Team
Organisation

Work together in a way that aligns with Agile values and principles

Complex work does not have best practice



1 The accelerating pace of change ...

<https://medium.com/ml-everything/ai-optimists-vs-pessimists-and-why-the-singularity-isnt-near-5d3a614dbd45>



2045 Surpasses brainpower equivalent to that of all human brains combined

2 ... and exponential growth in computing power ...

Computer technology, shown here climbing dramatically by powers of 10, is now progressing more each hour than it did in its entire first 90 years



Colossus

The electronic computer, with 1,500 vacuum tubes, helped the British crack German codes during WW II



UNIVAC I

The first commercially marketed computer, used to tabulate the U.S. Census, occupied 943 cu. ft.



Apple II

At a price of \$1,298, the compact machine was one of the first massively popular personal computers



Power Mac G4

The first personal computer to deliver more than 1 billion floating-point operations per second

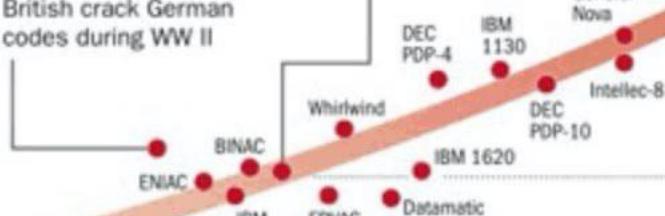
COMPUTER RANKINGS

By calculations per second per \$1,000



Analytical engine

Never fully built, Charles Babbage's invention was designed to solve computational and logical problems



Surpasses brainpower of mouse in 2015

Surpasses brainpower of human in 2023

Break free from Method prison



#noprojects – Optimize continuous delivery, flow, improvements and benefits rather than resources and time



Put effort into delivering benefits rather than meeting triple constraints



When will the software (work) deliver value next rather than when will the project be finished



Stream of value with uncertain stop date rather than a project handover
(projects end - software continues to evolve until retired)



Late requirements accepted rather than considered costly and undesirable



Keep learning from experience and experiments (double loop learning)

What does it mean to be Agile?

Person
Team
Organisation

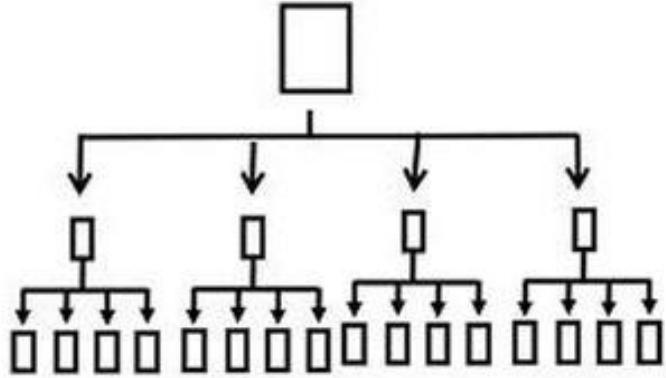
The Law of the Customer

The Law of the Team

The law of the Network

The Age of Agile
Denning

The law of the customer



The bureaucratic organization

- Internally focused
- Fixed mindset
- Defend existing advantages
- Make money for shareholders



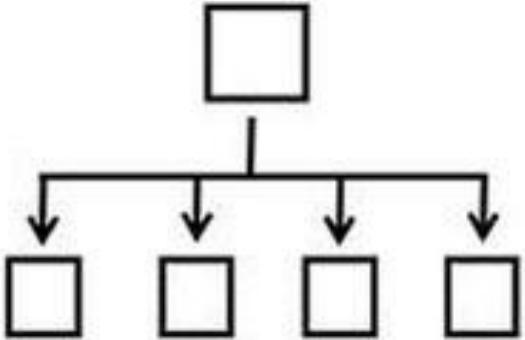
The Agile organization

- Externally focused
- Growth mindset
- Create new advantages
- Deliver value to customers

Agile practitioners are obsessed with delivering value to customers.

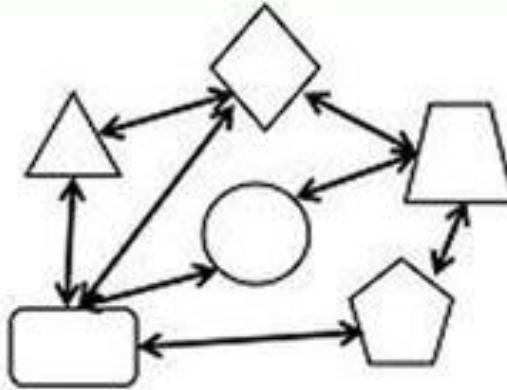
Everyone in the organization has a clear line of sight to the ultimate customer and can see how their work is adding value to that customer—or not. If their work isn't adding value to any customer or user, then an immediate question arises as to why the work is being done at all. The firm adjusts everything—goals, values, principles, processes, systems, practices, data structures, incentives—to generate continuous new value for customers and ruthlessly eliminate anything that doesn't contribute.

The law of teams



The bureaucratic team

- Top down
- Individual responsibilities
- Little interaction

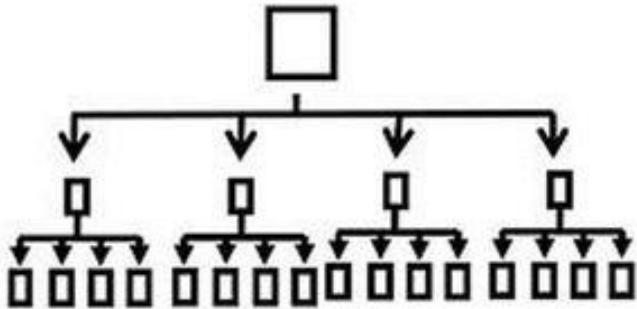


The Agile team

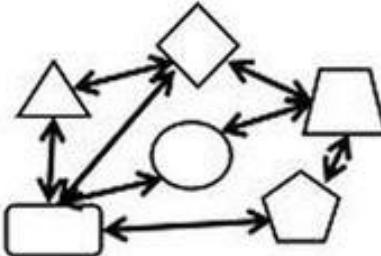
- Autonomous
- Cross-functional
- Much interaction

Agile practitioners share a mindset that work should in principle be done in small autonomous cross-functional teams working in short cycles on relatively small tasks and getting continuous feedback from the ultimate customer or end user.

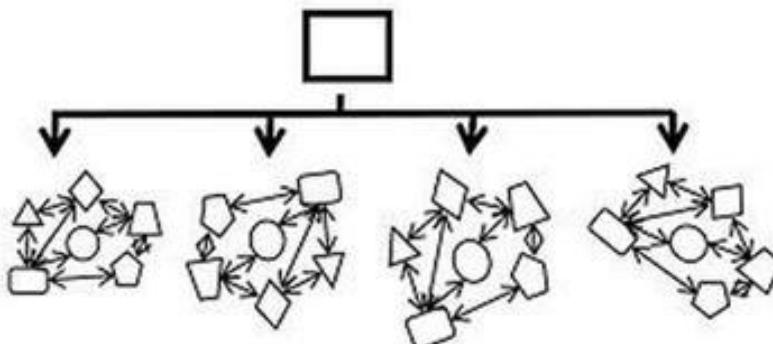
The law of networks



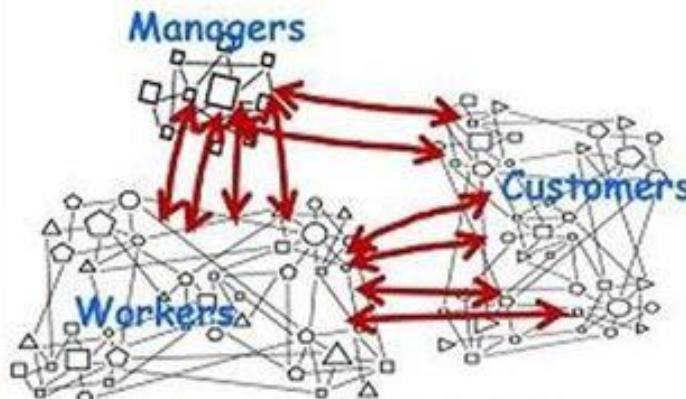
1. Bureaucracy



2. Agile team



3. Agile teams in a bureaucracy



4. Agile network

When Agile teams are housed within a bureaucracy, collaboration between teams can be just as much a problem as it is between silos in a pure bureaucracy.

Influence in Software Ecosystem and Software Communities

Jingchang Chen

What is a software ecosystem (SECO)?

An evolution from software product line. (Bosch, 2009)

What is a software ecosystem?

“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 under-pinned by a common technological platform or market and operate through the exchange of information, resources and artifacts” (Jansen et al., 2009)

“A software ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions” (Bosch, 2009)

A set of connected services based on a common technology foundation in a shared market.

Real-world examples

Microsoft Windows

Apple iOS, MacOS

Linux

Google Chrome

...

VS Code

React

USB Standard

Why SECO is popular?

Benefits (Barbosa & Alves, 2011):

- Reduce the (individual) development cost
- Welcome new players
- Accelerate the evolution
- Create new markets...

Especially good for small to medium enterprises,
which is the majority in the industry (Fischer et al.,
2000)

Challenge:

- Requirement engineering process - Influence matters! (Schaarschmidt et al., 2015; Linåker et al., 2019; Damian et al., 2021)

What brings influence?

There is a tie between contribution and influence,
especially in open-source software communities.

- Lines of code
- Number of pull requests

Overall, changes to the source code

Methodology, Data Source, Tools



Methodology

De

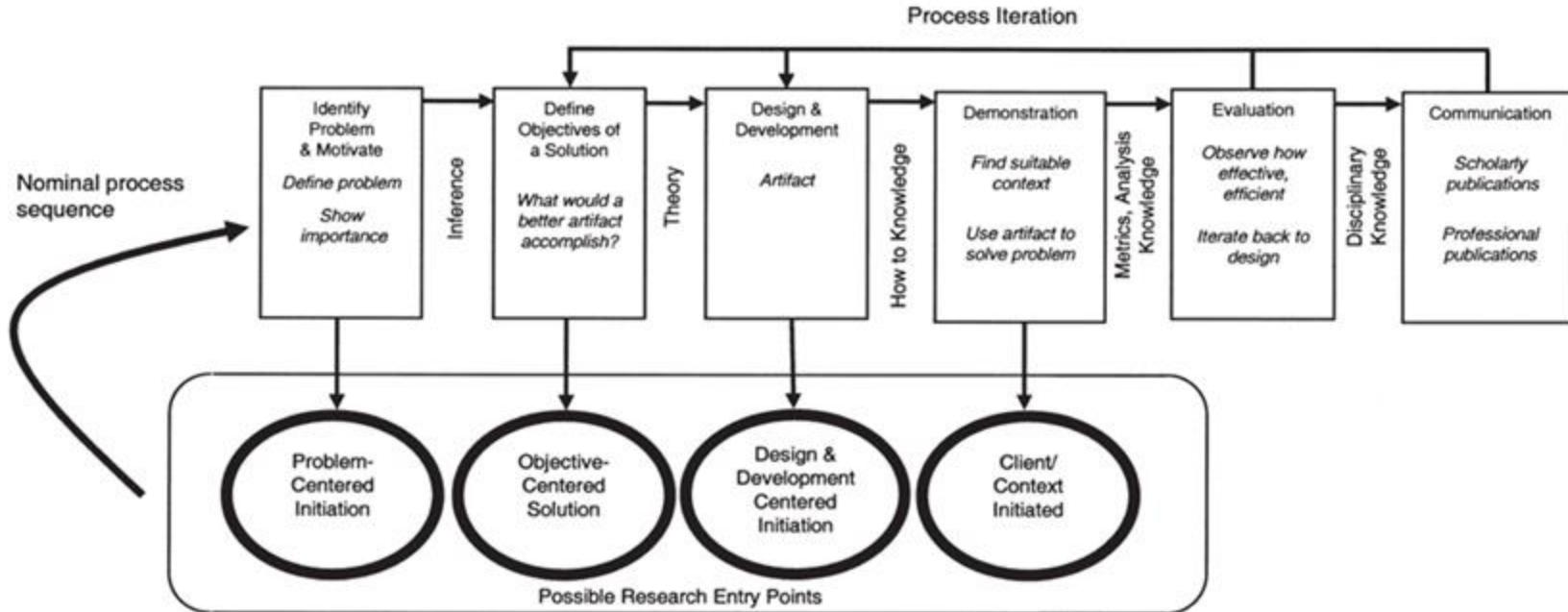


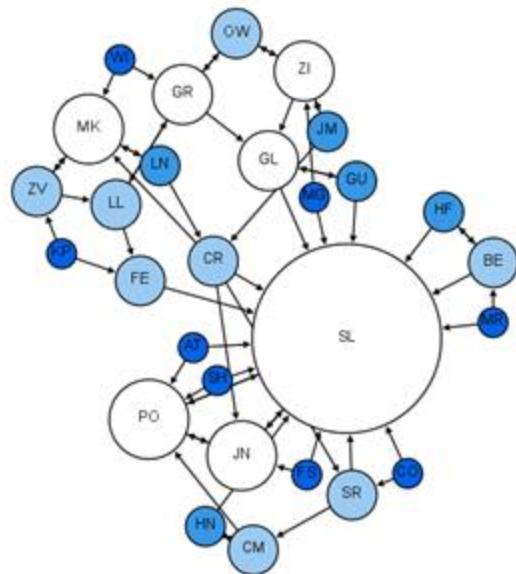
Figure 1. DSRM Process Model

Data Source

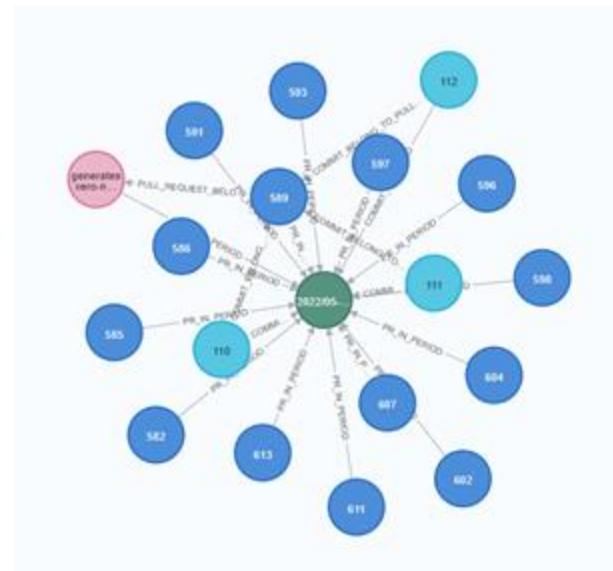
- Private
- Open-source
 - GitHub
 - GitTorrent (Gousios & Spinellis, 2012)
 - GHArchive

Tools

- Social Network Analysis
 - A tool to study relationships
 - Neo4j
 - A graphic relational database



^{**} A social network. Cited from Wikipedia.

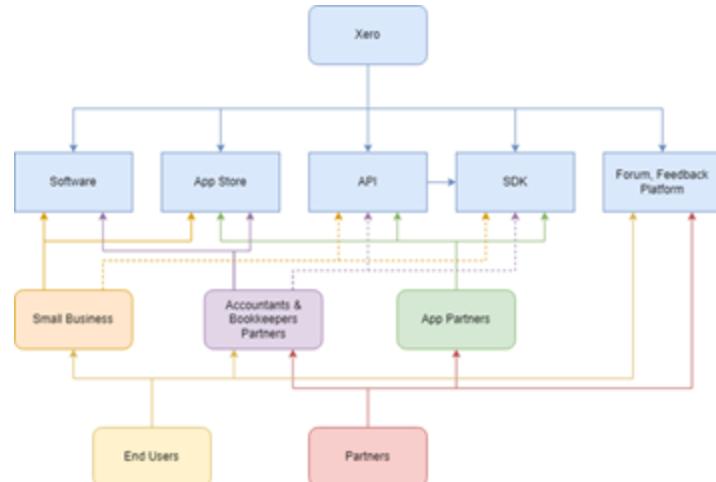


My study



Xero

1. Breaking down its structure



2. Identifying available public data sources

- Official website.
- Developer forum (deprecated).
- User forum.
- Xero API customer feedback platform.
- Application store.
- Official documents.
- GitHub repositories.

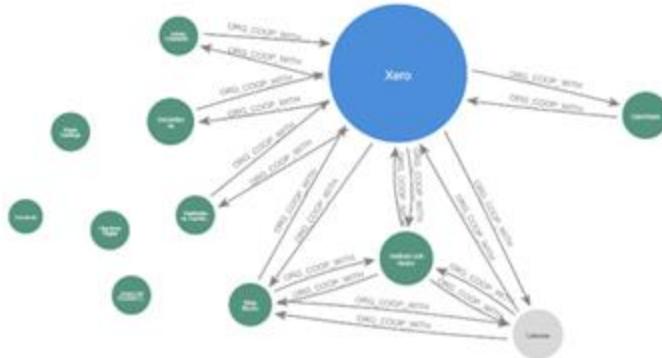
Xero

3. Mining GitHub

- ### ○ GitHub RESTful API

4. Building networks

- Node: Individual/Organisation
 - Edge: Both commit in one patch (pull request)
 - Aggregated by time period (versions/tags)



Xero

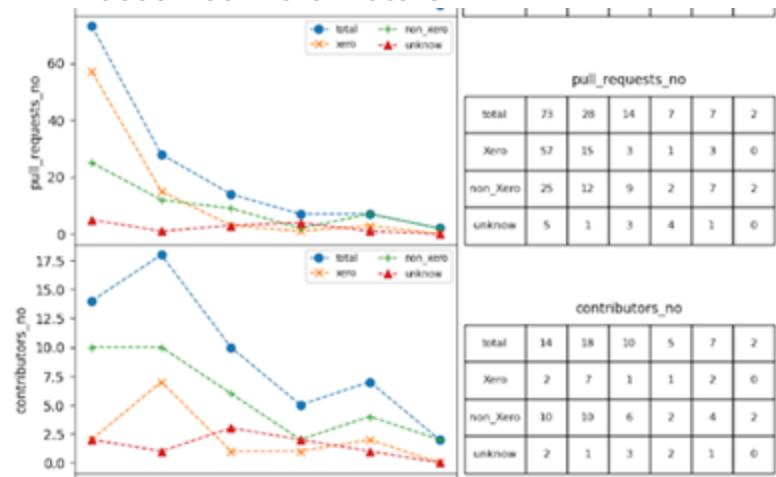
Results:

1. A large number of outsourcers are hired

Period	Orchestrator	Niche Player	End User	External Developer	Outsource	Unclear
2017/05-2017/10	1	1	2	0	6	1
2017/11-2018/04	1	1	0	2	4	3
2018/05-2018/10	1	0	1	0	1	3
2018/11-2019/04	1	0	0	0	2	0
2019/05-2019/10	1	0	1	1	1	1
2019/11-2020/04	0	0	1	0	1	0
Total	1	2	4	3	12	8

1. Most contributors are short-term contributors, focusing on the issue they encountered

3. In an SDK project, the amount of contribution and contributors decreases when the product becomes more mature



React

1. Mining React GitHub repo
 - o Use GArchive and GitTorrent to recover the missing pieces
2. Building social network
 - o Node: Individual
 - o Edge: Comment in the same pull request
 - o Aggregated by time period (versions/tags)
 - o Strong tie VS weak tie



3. Finding cliques
 - o Bron-Kerbosch algorithm
4. Labeling comments
 - o Manually label (code) a part of them
 - o Train machine learning model to label the rest

Raw Quote Example	Theme	Relevant Codes
<i>"You guys told me to do the sketchy hack instead of always reflecting the correct value. I don't like the solution and sebastian advocated for, but I needed to choose my battles in the interest of moving this diff forward."</i>	Conflict	Disagreement, Disappointment, Doubt, Highlighting Issues, Correction, Contrasting Needs or Perspective
<i>"I've opened #2273 to track the status of 'setInherHTML' vs 'createNodesFromMarkup'"</i>	Project Workflow	Process, Making a Plan, Labelling, Taking Responsibility, Assignment, Ending Discussions, Redirection
<i>"@jofb you made this change. What did you mean?"</i>	Interdependence	Relying on Social Connections, Accountability, Accepting Work, Reference, Demonstration
<i>"I know you are just following what's already here, but I think we should consolidate these constants."</i>	Improvement	Giving Advice, Giving Clarification, Agreement, Refinement, Making Suggestions, Gratitude
<i>"Would you mind explaining what this is good for?"</i> <i>"-raised hands: woohoo!"</i>	Inquiry	Seeking Advice, Seeking Clarification, Asking for Help
	Miscellaneous	-

React

Results:

- Conflicts exist in discussions in higher diverse teams

(Diversity Blau Index Score)	Ethnic	Nationality	Gender
Conflict Present-(23/39)	0.096	0.515	0.152
Conflict Absent-(16/39)	0.041	0.486	0.145

- These conflicts are usually beneficial (productive conflicts) (Schulz-Hardt et al., 2002), leading to more merged pull requests

Collaborative Groups	Conflict	No Conflict
Average Merged Pull Requests / Group	172.00	53.00
Average Merged Pull Requests / Group / Version	20.31	10.75

- The teams with conflicts also have more interdependence and inquiry comments in discussions, which shows a higher level of psychological safety

	Interdependence	Inquiry	p value
Groups (Conflict Present)	16	971	0.000
Groups (Conflict Absent)	0	194	0.000

- These teams also last longer, compared with groups with lower diversity and low/no conflicts

Thanks for watching!

References

- J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, 2009, vol. 9, pp. 111-119.
- S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *2009 31st International Conference on Software Engineering - Companion Volume*, 16-24 May 2009 2009, pp. 187-190, doi: 10.1109/ICSE-COMPANION.2009.5070978.
- O. Barbosa and C. Alves, "A systematic mapping study on software ecosystems," in *Proceedings of the Third International Workshop on Software Ecosystems*, 2011, pp. 15-26.
- E. Fischer, E. Fisher, and R. Reuber, Industrial clusters and SME promotion in developing countries (no. 3). *Commonwealth Secretariat*, 2000.
- M. Schaarschmidt, G. Walsh, and H. F. O. von Kortzfleisch, "How do firms influence open source software communities? A framework and empirical analysis of different governance modes," *Information and Organization*, vol. 25, no. 2, pp. 99-114, 2015/04/01/ 2015, doi: <https://doi.org/10.1016/j.infoandorg.2015.03.001>.
- J. Linåker, B. Regnell, and D. Damian, "A Community Strategy Framework—How to obtain influence on requirements in meritocratic open source software communities?," *Information and Software Technology*, vol. 112, pp. 102-114, 2019.
- D. Damian, J. Linåker, D. Johnson, T. Clear, and K. Blincoe, "Challenges and Strategies for Managing Requirements Selection in Software Ecosystems," *IEEE Software*, vol. 38, no. 6, pp. 76-87, 2021.
- K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45-77, 2007.
- A. Marin and B. Wellman, "Social network analysis: An introduction," *The SAGE handbook of social network analysis*, pp. 11-25, 2011.
- S. Schulz-Hardt, M. Jochims, and D. Frey, "Productive conflict in group decision making: Genuine and contrived dissent as strategies to counteract biased information seeking," *Organizational Behavior and Human Decision Processes*, vol. 88, no. 2, pp. 563–586, 2002.

Representing Global DevSecOps to Usefully Support Software Engineering Practice

Gavin Zhao

PhD Candidate

School of Engineering, Computer and Mathematical Sciences

Faculty of Design & Creative Technologies

Auckland University of Technology

City Campus, Auckland, New Zealand

Email: gavin.zhao@autuni.ac.nz



CONTENTS

① Key Concepts

- DevOps
- Security
- DevSecOps
- GSE

② Research Design

- Research goals
- Research questions
- Research methods:
MLR + Delphi study

③ MLR on DevSecOps

- MLR search
- Thematic analysis
- MLR results
- DevSecOps models
- Publications

④ Delphi study

- Preparing
- Conducting
- Analyzing
- Delphi vs MLR Results
- Local vs Global



1. Key Concepts – DevOps

DevOps can be defined as a software process, methodology, movement, even a culture, aimed at bridging the gap between Development (Dev) and Operations (Ops).



DevOps

A large, dark blue arrow pointing to the right, with the word "DevOps" written in white capital letters on its left side.

Why DevOps?

- gap between Dev & Ops getting closer along with the adoption of clouds, micro-services, and containers;
- better collaboration and teamwork between Dev & Ops;
- faster development and easier deployment by CI/CD;
- higher effectiveness by automation;
- higher product quality and greater business value.



1. Key Concepts – Security in SE

Security

- Security is an important non-functional requirement of software development but is often devalued in DevOps programs.
- It includes security of the software development environment (security threats in the factory) and security of the software being developed (software security testing)
- Growing importance of security includes privacy to users in larger scale systems, SaaS, globally distributed systems, and its conflicts with rapid delivery cycles.
- Use of containers, cloud and server-less computing brings increasing security complications.



1. Key Concepts – DevSecOps

DevSecOps/SecDevOps is created as a security-orientation expansion to DevOps for integrating security into DevOps by improving the collaboration between development, operation and security teams.

DevSecOps

Benefits:

- shifts security to the early stages (shift-left);
- carries out continuous security in the entire Software Development Lifecycle (SDLC);
- employs automation to reduce the need for manual security support.



1. Key Concepts – Global Software Engineering

Global Software Engineering (GSE) is a business strategy aimed to find specialized and diverse resources by accessing “a global pool” of skilled human resources, to improve competitiveness by accessing a global market.



Benefits:

- reduce costs due to possible salary savings
- shorten duration due to time-zone effectiveness and round-the-clock productivity



1. Key Concepts – Global Software Engineering

GSE depends on distributed teams with stakeholders from different locations, different time zones, and even different organizational and national cultures.



GSE faces challenges from geographical, temporal, linguistic and cultural distances so that it is particularly associated with the 3C Collaboration model (Communication, Coordination and Cooperation).

GSE and DevOps/DevSecOps:

They all belong to Collaborative Software Engineering (CoSE), which is about creating the organizational structures, reward structures, and work breakdown structures that afford effective work towards the goal.



2. Research Design – Research Goals

Research Objectives / Potential Contributions:

- to observe, document and analyze the current state of DevSecOps;
- to compare the differences between local and global DevSecOps;
- to develop a conceptual framework for global DevSecOps guided by experts in the domains of DevOps, security, and GSE.



2. Research Design – Research Questions

RQ 1

How do the experts prioritize the identified challenges, practices, tools and metrics of DevSecOps?

Sub-question 1.1. What additional DevSecOps challenges, practices, tools and metrics could be collected from the experts?

Sub-question 1.2. Will the experts have 'dissent' opinions on the prioritization due to their different roles, e.g. academic, industrial, technical and managerial.

RQ 2

What are the experts' opinions on DevSecOps in GSE contexts?

Sub-question 2.1. How is it different between local and global settings?

Sub-question 2.2. What additional challenges, practices, tools and metrics when it comes to a global setting?



3. MLR on DevSecOps – What is MLR



Multi-vocal Literature Review (MLR) : is a special form of systematic literature review which does not only use formally published literature (WL) but also includes unpublished work (GL).

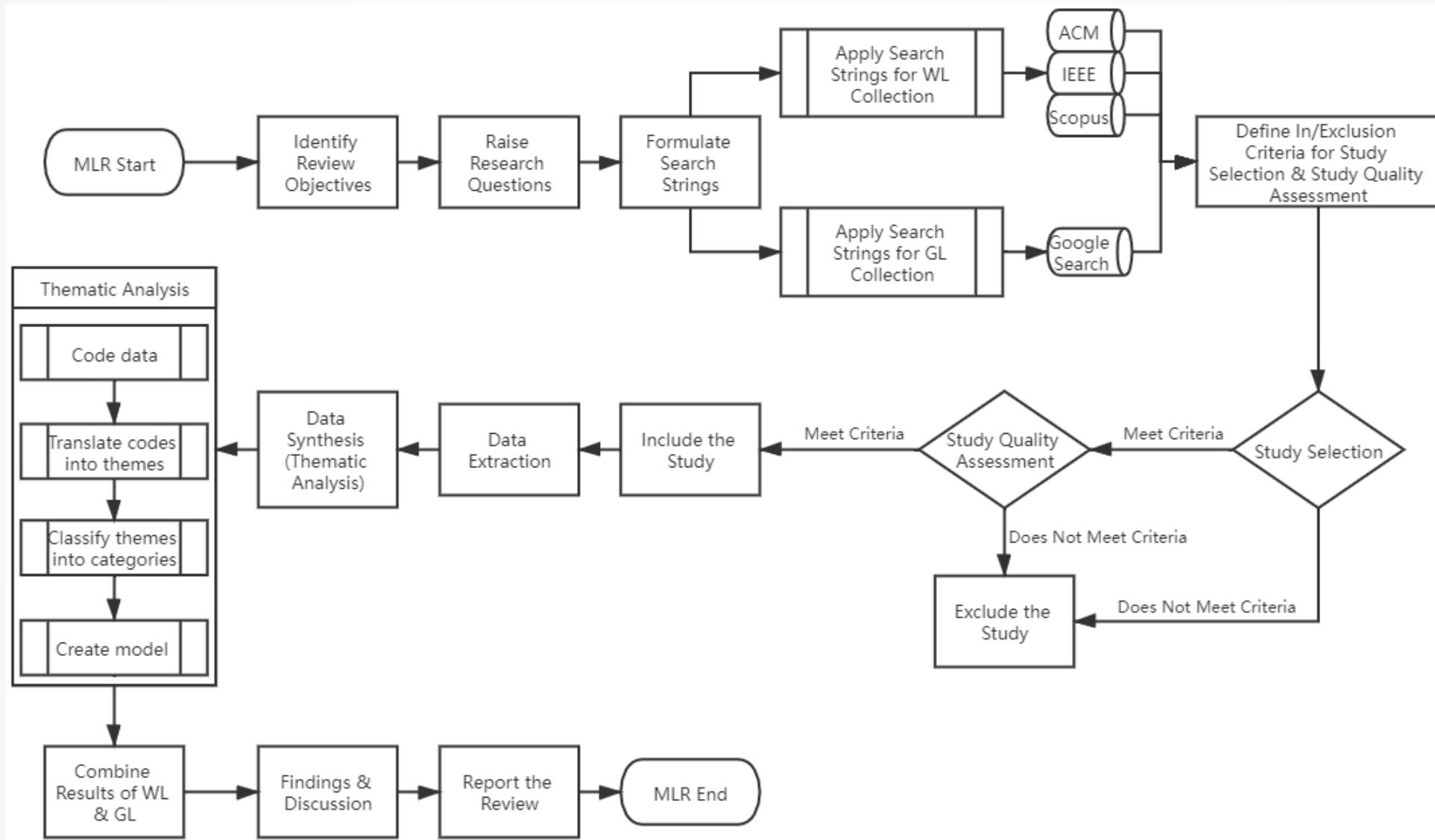
- White Literature (WL): formally published literature, e.g. journal articles and conference papers.
- Grey Literature (GL): unpublished work, e.g. technical reports, websites, blog posts, etc.

Reason for using MLR:

DevSecOps is an increasingly topical concept in both academic and industrial settings, the voices from academia and industry are equally essential.



3. MLR on DevSecOps – MLR Process





3. MLR on DevSecOps – MLR Search

*Search String 1 = "DevOps" AND ("security" OR "secure" OR "safe")
OR "SecDevOps" OR "DevSecOps"*

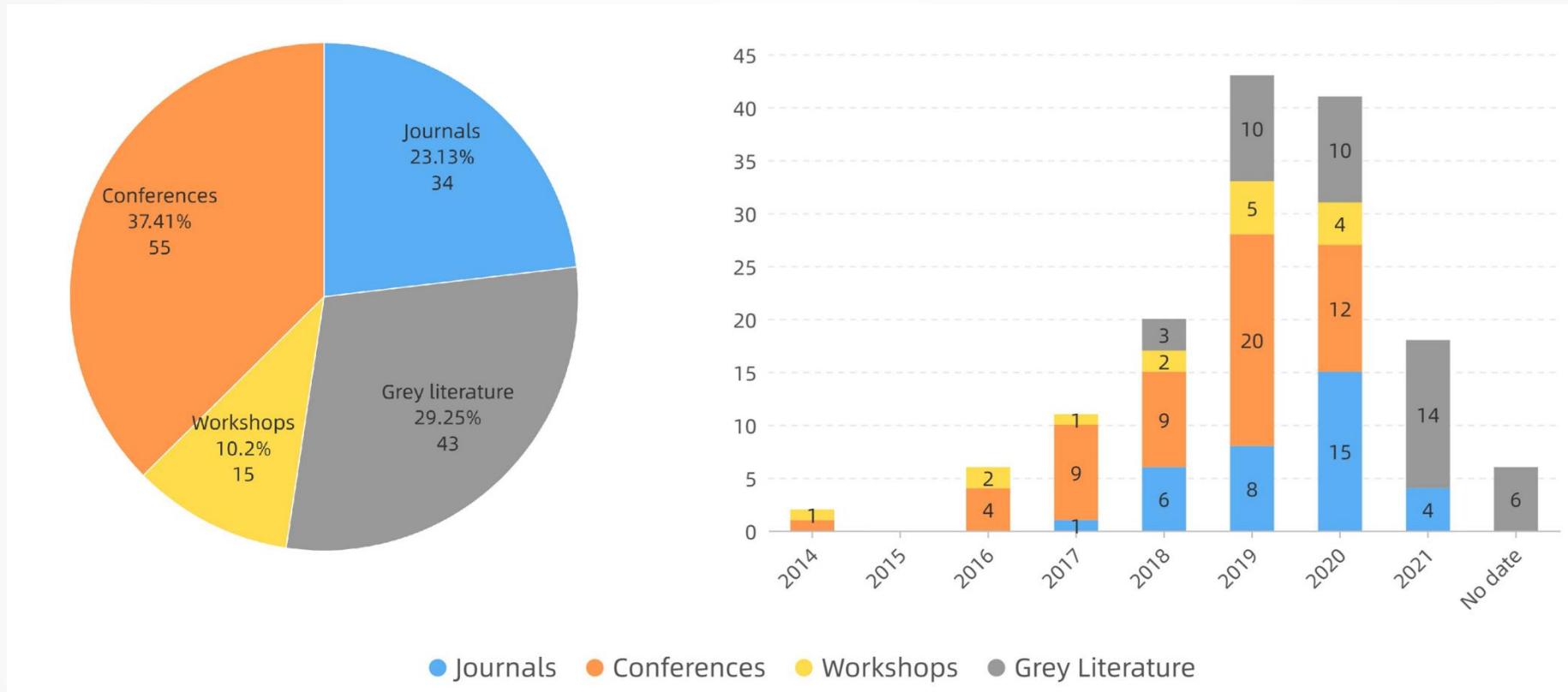
*Search String 2 = "DevOps" AND ("security" OR "secure" OR "safe") AND ("global
software engineering" OR "global software development" OR "GSE" OR "GSD" OR
"globally distributed software*") OR "SecDevOps" OR "DevSecOps"*

Summary of MLR search execution		
Search Steps	Search 1 Results	Search 2 Results
	WL / GL	WL / GL
Applying Search String	327 / 180 studies	90 / 100 studies
Study Pre-selection	238 / 56 studies	66 / 3 studies
Study Selection and QA	112 (acm-28, ieee-46, scopus-38) / 42 studies	2 (acm-2) / 0 study

Lack of global dimension of DevSecOps in the existing literature



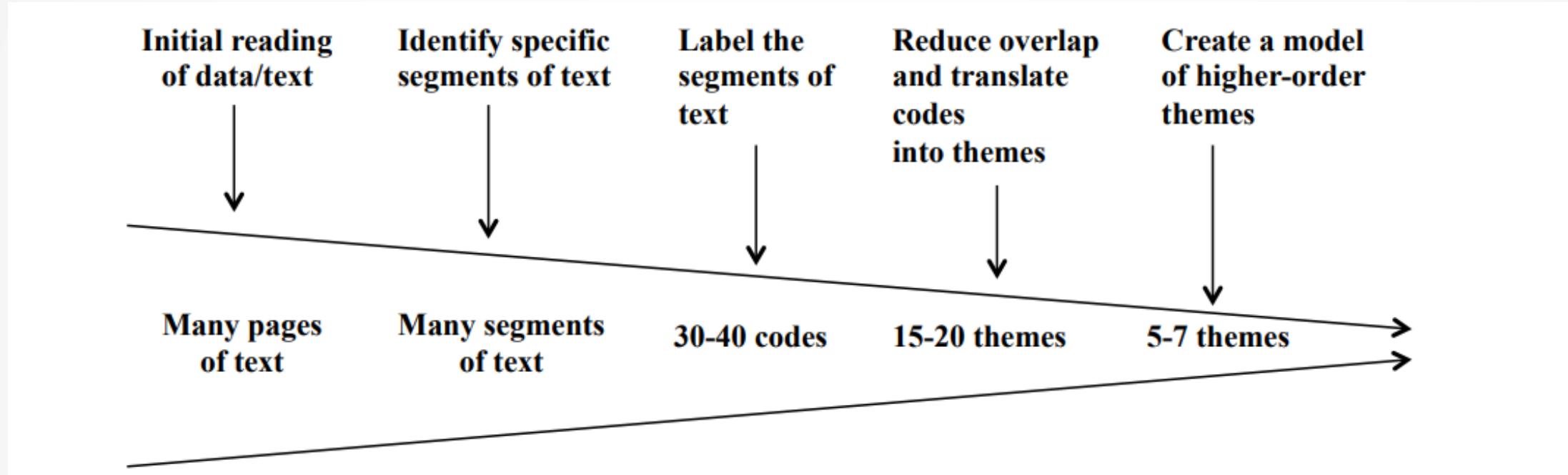
3. MLR on DevSecOps – MLR Search



Number of included papers based on source types and published years



3. MLR on DevSecOps – Thematic Analysis



Thematic Analysis (TA) is a method for identifying, analyzing and reporting themes with qualitative data.



3. MLR on DevSecOps – Thematic Analysis

Thematic analysis and synthesis results

Source	Extract data	Code data	Translate codes into themes	Classify themes into categories
WL	33 DevSecOps definitions	49 codes	14 themes	4 categories: OPC, PC, Technology, Business
	106 DevSecOps challenges	74 codes	36 themes	4 categories: OPC, PC, Technology, Business
	267 DevSecOps practices	86 codes	41 themes	3 categories: OPC, PC, Technology
	16 DevSecOps metrics	41 codes	15 themes	4 categories: OPC, PC, Technology, Business
	33 DevSecOps tools	33 codes	11 themes	Single category: Technology
GL	15 DevSecOps definitions	35 codes	20 themes	4 categories: OPC, PC, Technology, Business
	54 DevSecOps challenges	49 codes	16 themes	3 categories: OPC, PC, Technology
	143 DevSecOps practices	106 codes	54 themes	4 categories: OPC, PC, Technology, Business
	6 DevSecOps metrics	16 codes	6 themes	2 categories: PC, Business
	45 DevSecOps tools	45 codes	14 themes	Single category: Technology

Themes were classified into four categories:

- Organization, People & Culture (OPC)
- Process Capabilities (PC)
- Technology
- Business



3. MLR on DevSecOps – MLR Results

To the knowledge, we find that most of existing literature only contains two of the three terms:

- DevOps + Security (DevSecOps), no GSE;
- DevOps + GSE (global DevOps), no Security.

Absence of global

Reasons:

- no differences between local and global DevSecOps;
- security is a centralized and control-oriented function in organizations, so global aspects are not prominent;
- construction of search string;
- research gap in this direction.



3. MLR on DevSecOps – MLR Results

Key Findings

- Five major aspects of DevSecOps have been identified in the included (white and grey) literature: **Definitions, Challenges, Practices, Tools/Technologies, Metrics/Measurement.**
- Related **themes** of each aspect have been collected and analyzed by performing a **Thematic Analysis (TA)** process.
- A **Challenge-Practice-Tool-Metric (CPTM) model** for DevSecOps has been built by integrating the themes of the latter four aspects.
- The **absence of global DevSecOps** has been identified.

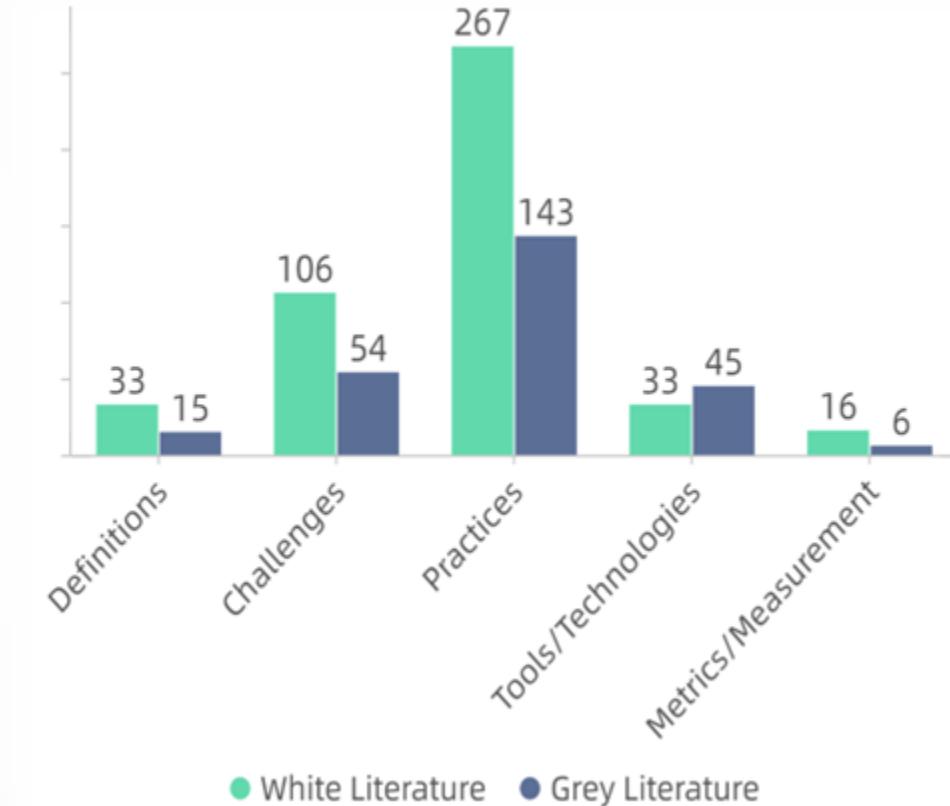


3. MLR on DevSecOps – MLR Results

Five aspects of DevSecOps have been identified in the literature:

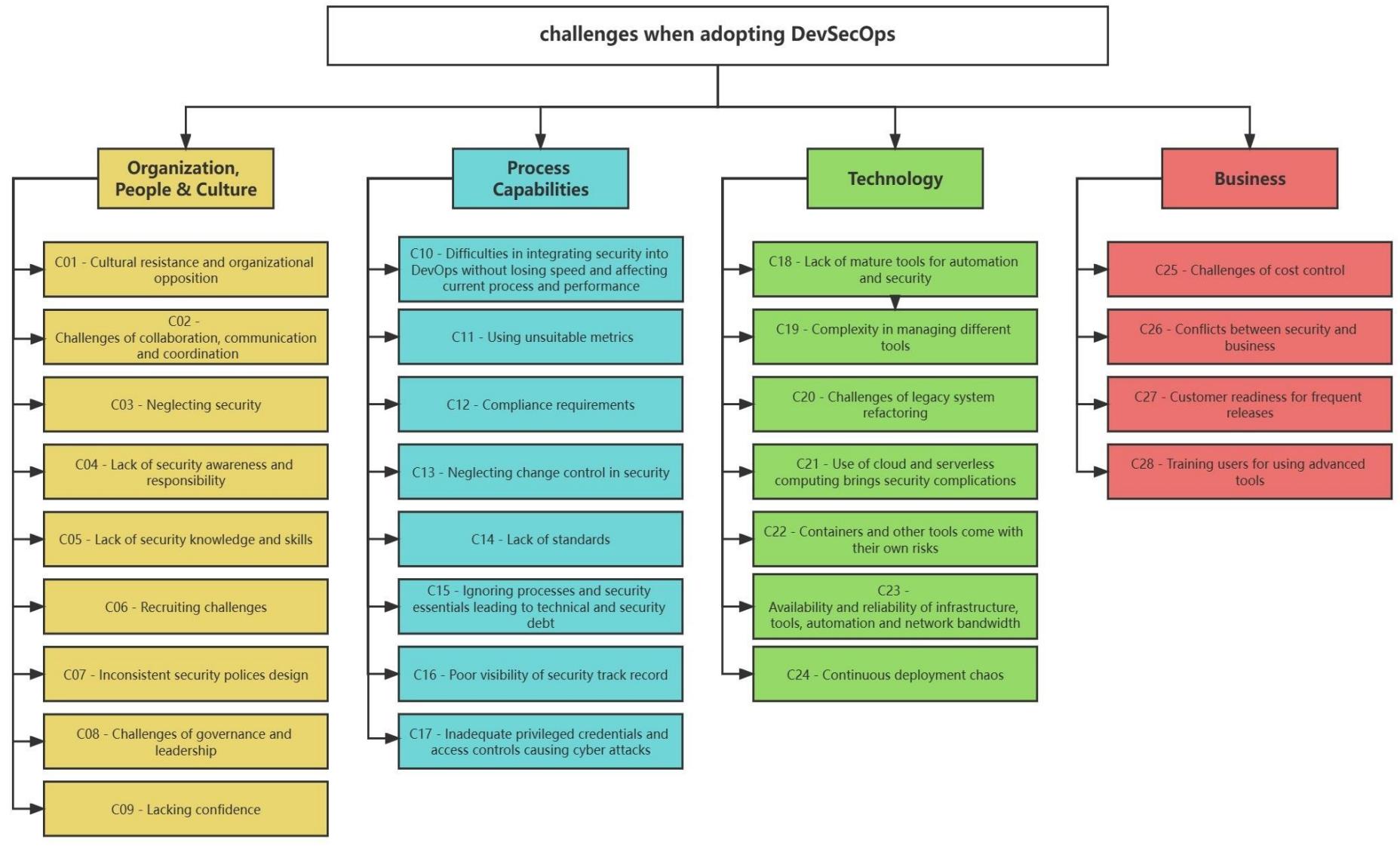
- **Definitions:** definitions for the term “DevSecOps” and equivalent terms;
- **Challenges:** problems, concerns and uphill tasks that are faced during implementing DevSecOps;
- **Practices:** DevOps and security activities suited for DevSecOps;
- **Tools/Technologies:** specific tools and technological approaches that can be used for DevSecOps;
- **Metrics/Measurement:** means for measuring security level and DevSecOps maturity.

In comparison of WL and GL results (academic and industrial), WL contributes more to phenomenological researches, defining concepts and identifying DevSecOps challenges and practices; GL contributes more to the business perspective, focusing on practical DevSecOps tools and metrics to provide solutions.





3. MLR on DevSecOps – MLR Results



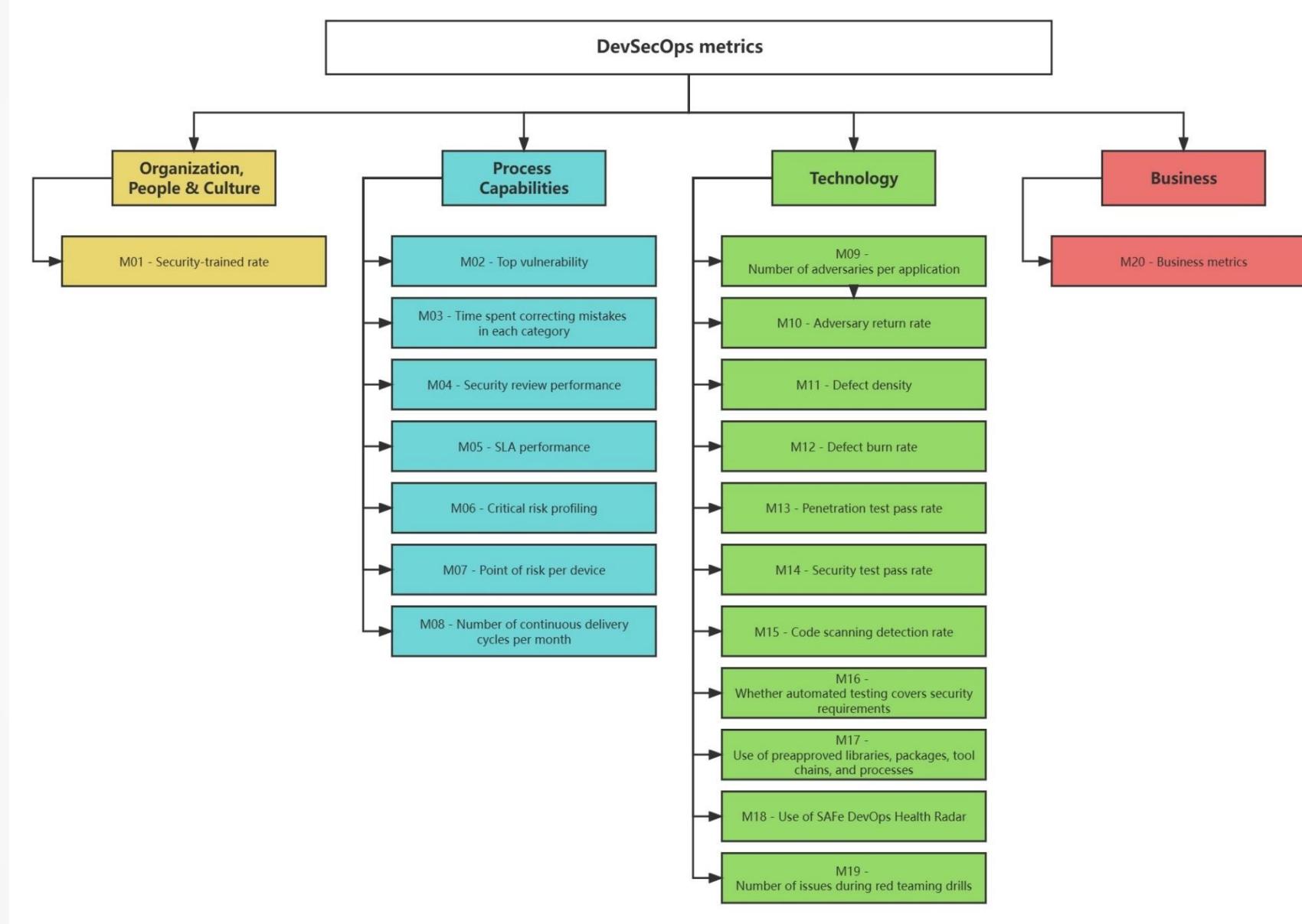


3. MLR on DevSecOps – MLR Results



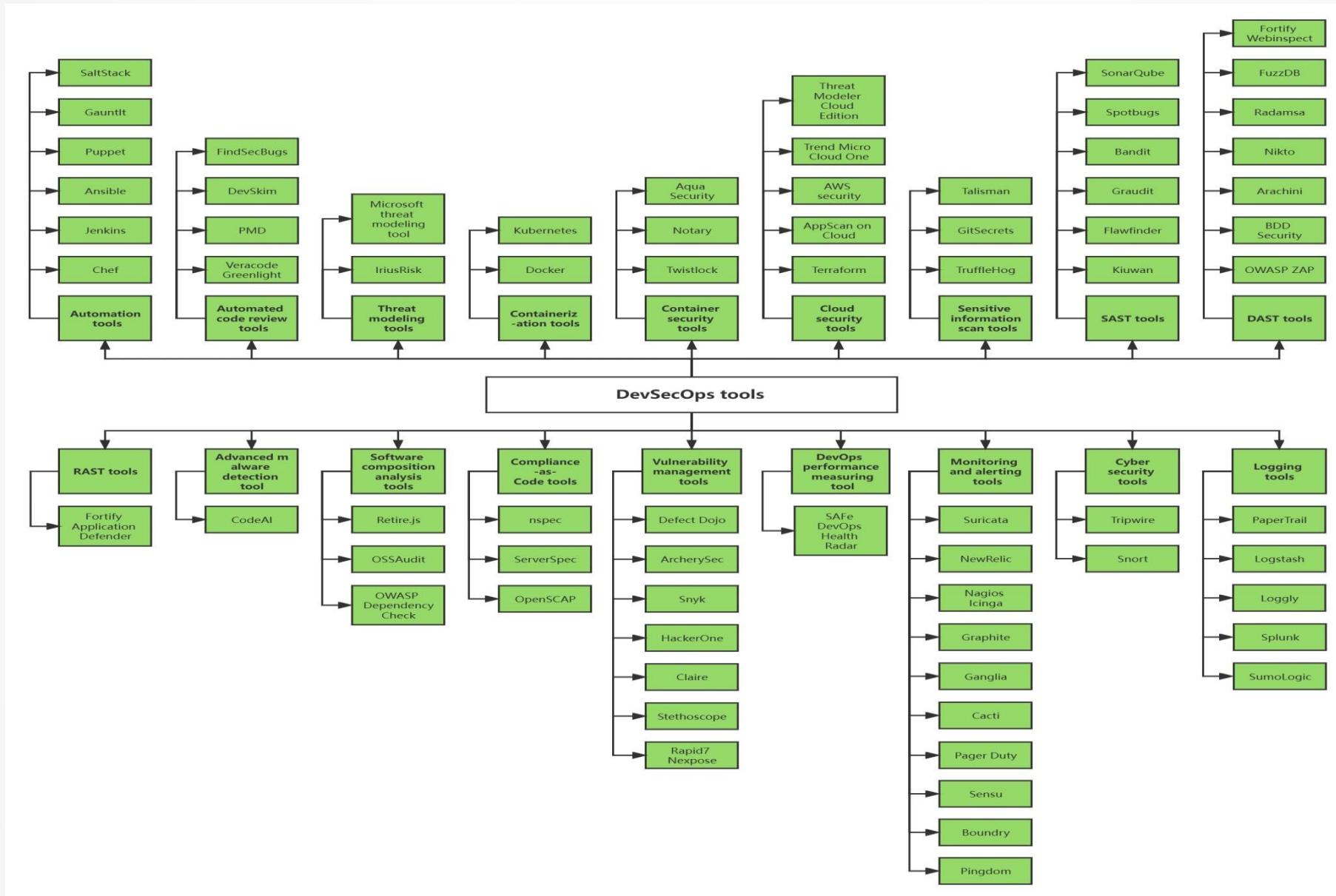


3. MLR on DevSecOps – MLR Results



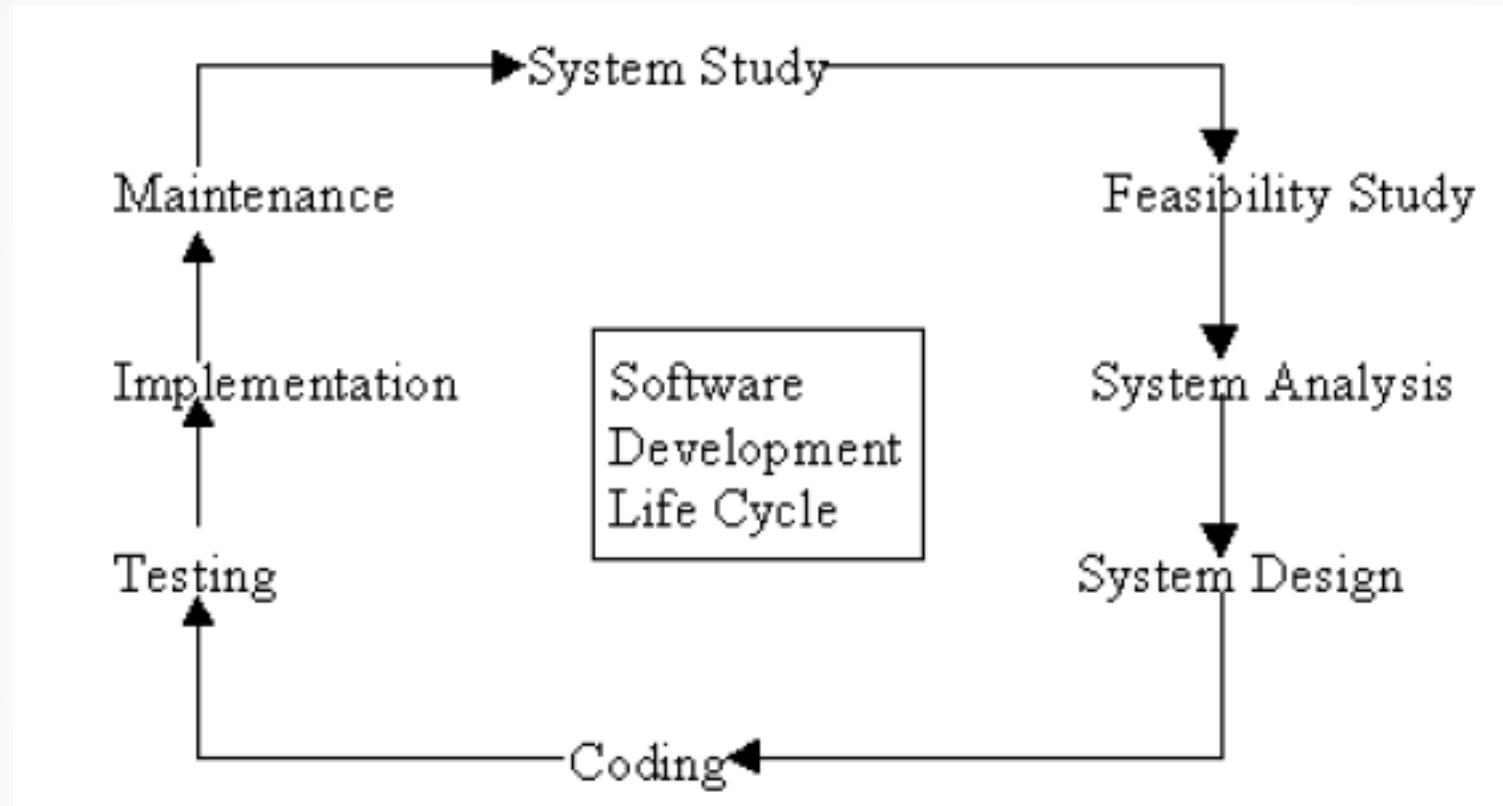


3. MLR on DevSecOps – MLR Results





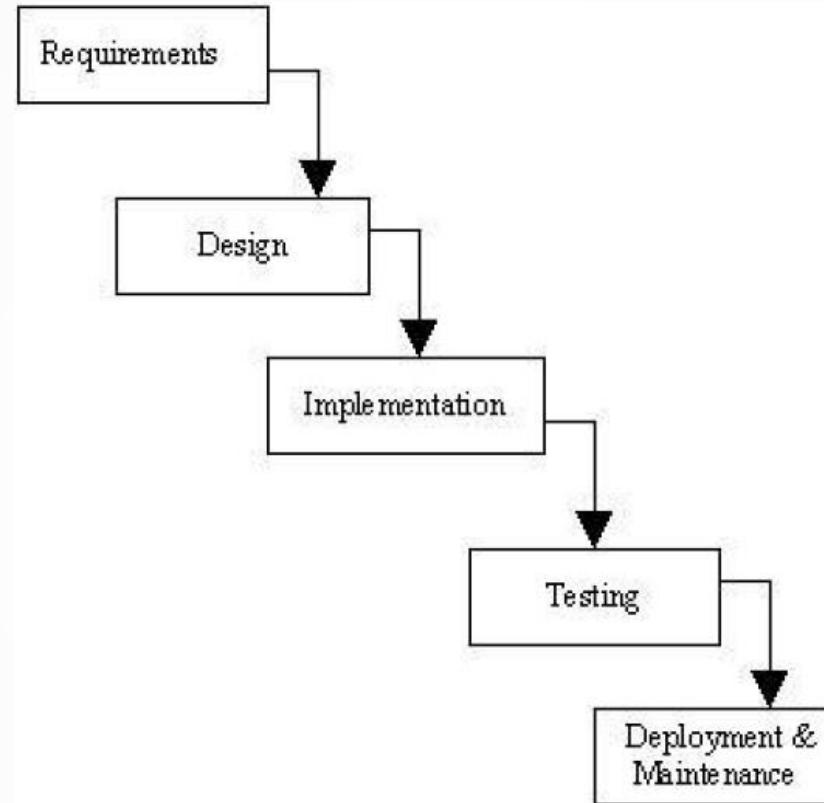
3. MLR on DevSecOps – Common SDLC Model



Common Steps in a Software Development Lifecycle (SDLC) Model



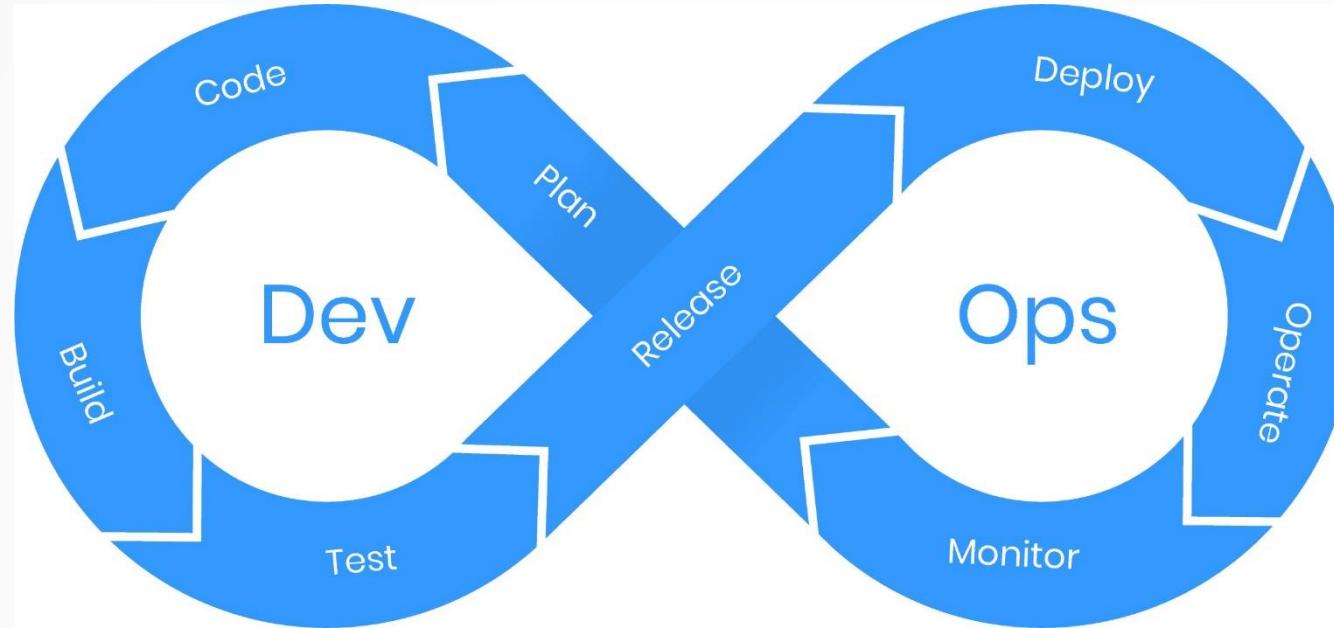
3. MLR on DevSecOps – Waterfall Model



Waterfall Model



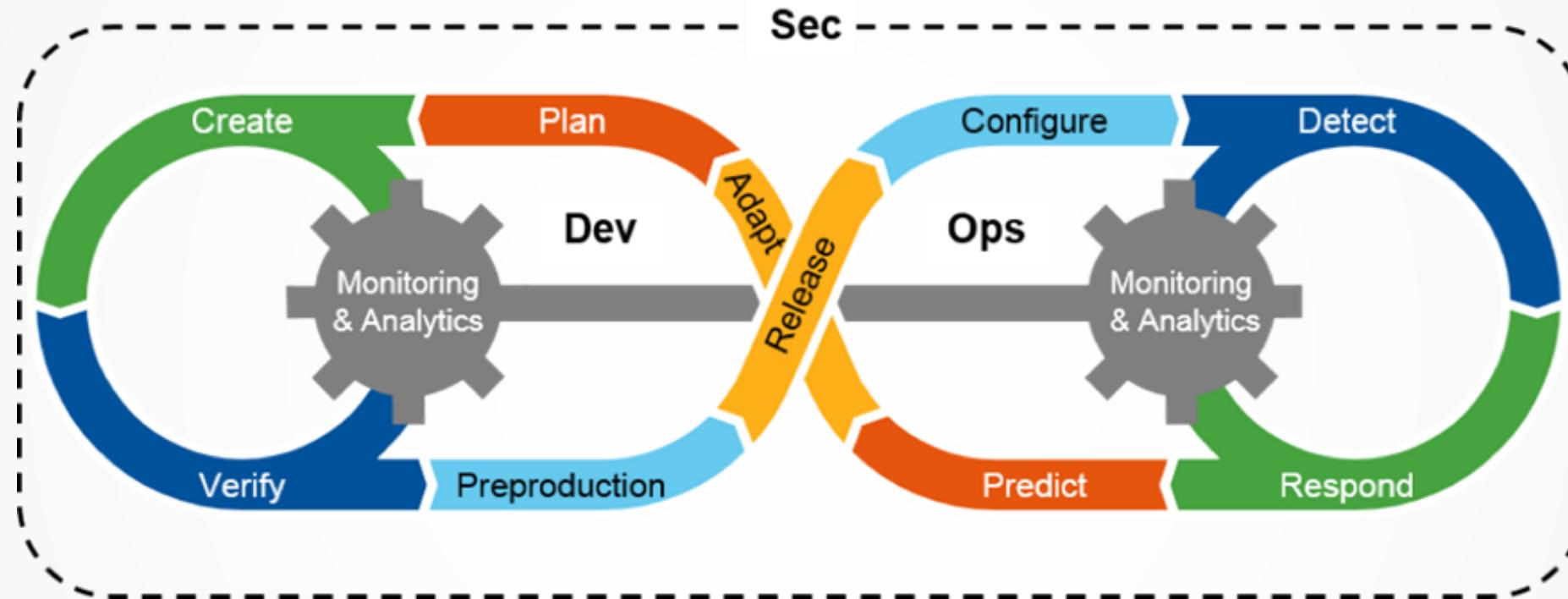
3. MLR on DevSecOps – DevOps Model



DevOps Model by Jireh Tech



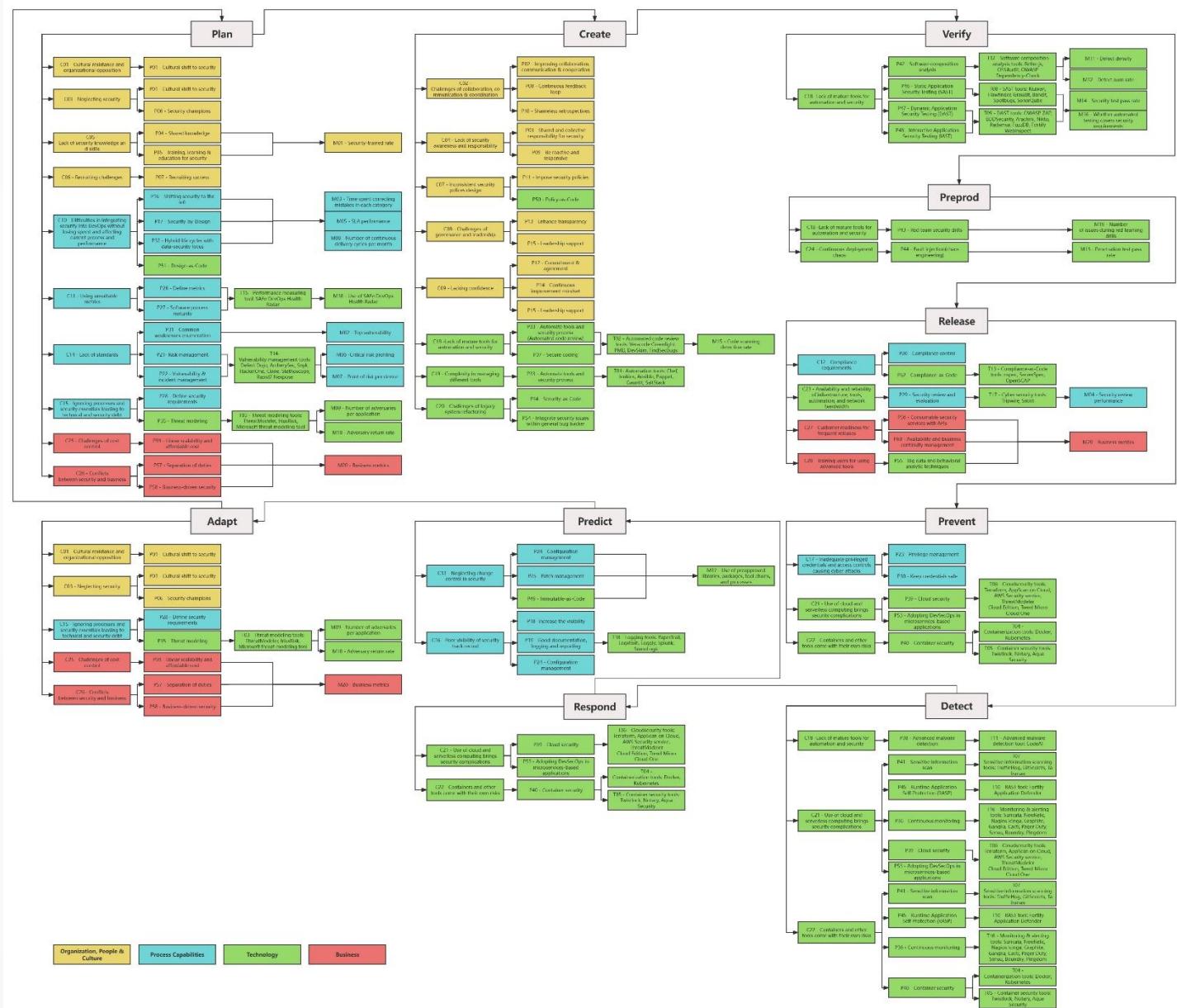
3. MLR on DevSecOps – DevSecOps Model



DevSecOps Model by Gartner



3. MLR on DevSecOps – CPTM Model by us





3. MLR on DevSecOps – Our Publication

The Journal of Systems and Software 214 (2024) 112063

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss







Identifying the primary dimensions of DevSecOps: A multi-vocal literature review[☆]

Xiaofan Zhao*, Tony Clear, Ramesh Lal

Auckland University of Technology, 55 Wellesley Street East, Auckland Central, Auckland, New Zealand

ARTICLE INFO

Dataset link: <https://doi.org/10.5281/zenodo.7959584>

Keywords:

Multivocal literature review
DevSecOps
DevOps
Security
Global software engineering

ABSTRACT

Context: Security as a key non-functional requirement of software development is often ignored and devalued in DevOps programs, with security seen as an inhibitor to high velocity required in DevOps implementation. Hence, the DevSecOps approach as a security-orientated expansion to DevOps, has aimed to integrate security into DevOps implementation by promoting collaboration among development, operation and security teams. DevSecOps is a topical concept and rapidly emerging area of practice in both academic and industrial settings.

Objective: We reviewed both the white and grey literature to identify recent researches and practical trends of DevSecOps, aiming to: (a) review, document and analyze the current state of DevSecOps in the existing literature; (b) investigate the application of DevSecOps in Global Software Engineering (GSE) contexts.

Method: A Multi-vocal Literature Review on DevSecOps and its global application was conducted, by executing a dual-track strategy including white (104 studies) and grey (43 studies) literature from 2012 to 2021. A Thematic Analysis was performed to identify, synthesize and analyze the themes within data for reporting the MLR results.

Results: Through the Multi-vocal Literature Review and Thematic Analysis, this paper identifies five major aspects of DevSecOps (Definitions, Challenges, Practices, Tools/Technologies, and Metrics/Measurement); collects related themes of each aspect; and generates a **Challenge-Practice-Tool-Metric (CPTM)** model by integrating the themes of the latter four aspects within a lifecycle model. Moreover, an unexplored area relating to the global application of DevSecOps has been identified.

Conclusion: Based on MLR results, a CPTM (Challenge-Practice-Tool-Metric) model is built to reveal the current status of DevSecOps. The model provides a breakdown and a broad landscape of DevSecOps, from which researchers and practitioners may select an area of focus to improve their knowledge or practice. With DevSecOps spanning the many stages of the lifecycle, we believe the model will enable emphases and absences such as global aspects to be investigated.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.



4. Delphi Study on DevSecOps – What is Delphi

Delphi (aka expert survey method) is a data collection method to solicit opinions from experts in certain domains by conducting multiple rounds of interview/survey, aiming to generate insights and get consensus on controversial subjects with limited information. The key differences between ordinary surveys and Delphi method are the iteration and the use of the feedback.

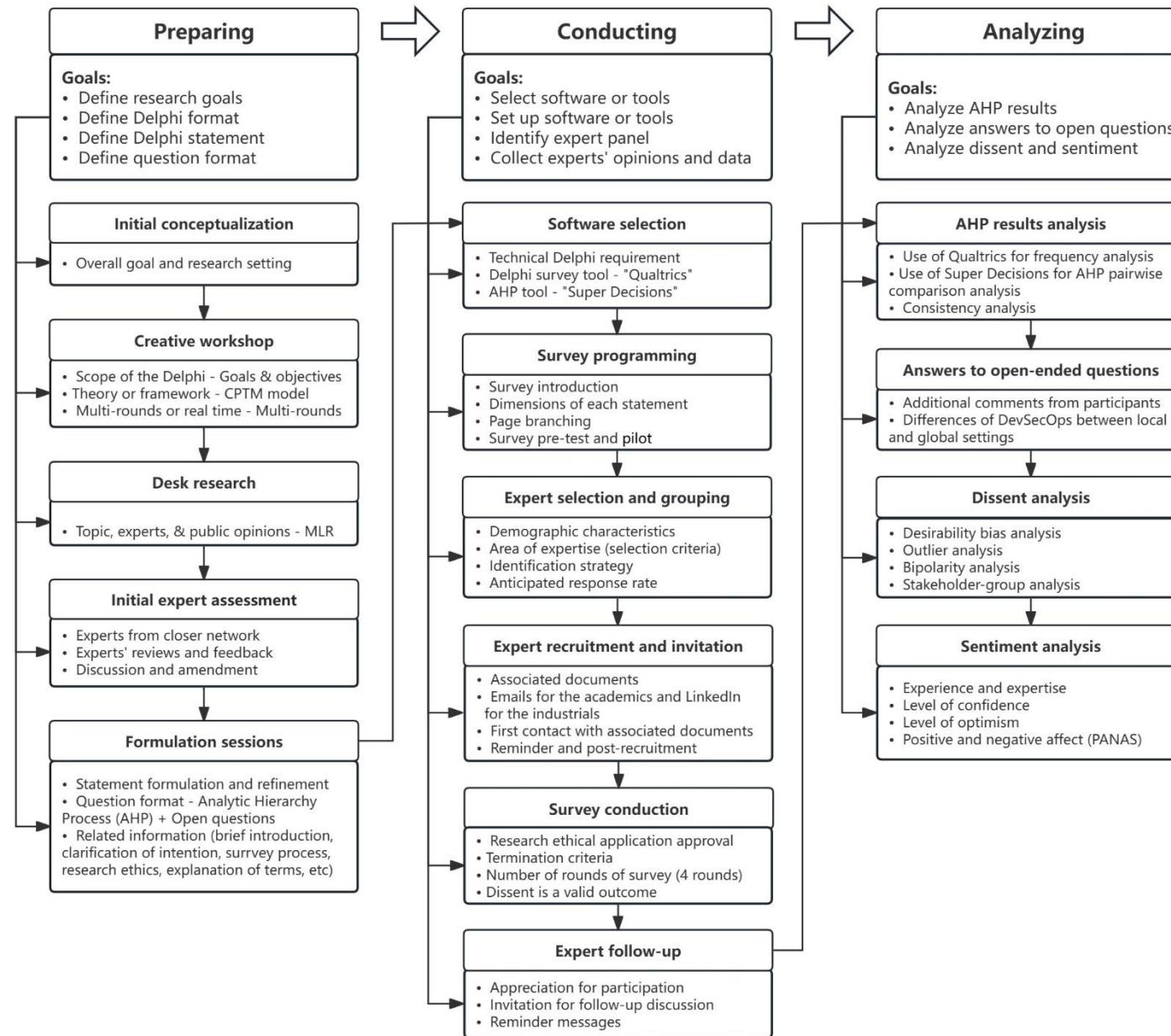
Delphi Study

Benefits (reasons for selecting Delphi):

- Delphi is best suited for developing new concepts and setting the unexplored directions (absence of global DevSecOps in MLR);
- Online surveys with international experts in anonymous format, without geographical restriction;
- Delphi is often used for testing and improving a set of outcomes from literature review, that's exactly what happened to this research.

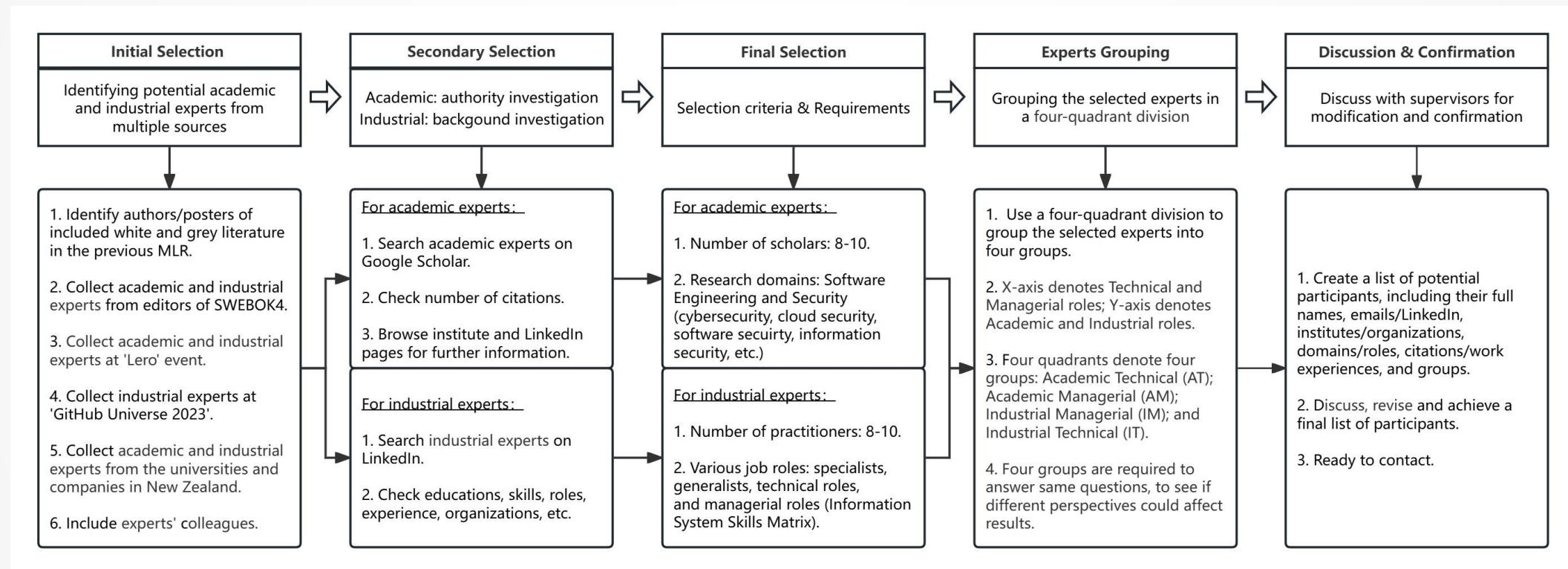


4. Delphi Study – Process





4. Delphi Study – Expert Selection





4. Delphi Study – Expert Selection Criteria

Criteria	Requirements
Number of experts	Minimum of 20 experts.
Professional experience	Minimum 5 years.
Field of work	<ul style="list-style-type: none">• Academia: 10-15 experts.• Industry: 10-15 experts (various roles).• Non-profit organizations (NPO): No formal number required.• Overlapped fields: No formal number required.
Domain of expertise	Academia: Software Engineering (SE) and Information Security (IS) Industry: various job roles in the Information System Skills Matrix (specialist, generalists, technical roles, and managerial roles)



4. Delphi Study – Industrial Expert Roles

		Managerial	
Specialist	Enterprise Product Marketing Manager	IT Service Management Consultant	
	Content Marketing Manager	Business Relationship Manger	
	Technical Program Manager	Head of Growth and Community	
	Sales Manager	Researcher and Evangelist	
	Principal Cloud Economist	Content Writer	
	Strategic SEO/PPC	Technical Writer	
	Cybersecurity Content Writer	Content Editor	
	Cybersecurity Journalist	Author	
			Generalist
		Distinguished Engineer	
		Master Inventor	
		CTO	
		Full Stack Engineer	
		Developer & IT	
		Software Engineer	
		Security Analyst	
		Network Engineer	
			Technical



4. Delphi Study – Survey Conduct

Round of survey	Estimated duration	Goal and activities
Round One	15 minutes	Rate the importance of identified DevSecOps challenges
Round Two	20 minutes	Assess the revised challenges (minimize the number of comparisons to reduce time and avoid inconsistency); Rate the importance/adoption of identified DevSecOps practices
Round Three	30 minutes	Assess the revised practices (minimize the number of comparisons to reduce time and avoid inconsistency); Rate the importance/adoption of identified DevSecOps tools and metrics.
Round Four	20 minutes	Assess the revised tools and metrics; Assess the refined CPTM model (open questions).



4. Delphi Study – Analyzing (Challenges)

DevSecOps Challenge	Category of Challenges	Category Normalized Value	Challenge Normalized Value within Category	Ranking within Category	Overall Priority	Overall Ranking
C01-Cultural resistance and organizational opposition	OPC	0.551570177	0.055256975	8	0.0304781	12
C02-Challenges of collaboration, communication and coordination	OPC	0.551570177	0.059057908	7	0.032574581	11
C03-Neglecting security	OPC	0.551570177	0.115978761	5	0.063970426	6
C04-Lack of security awareness and responsibility	OPC	0.551570177	0.211032393	1	0.116399174	1
C05-Lack of security knowledge and skills, need for training	OPC	0.551570177	0.131483634	4	0.072522451	4
C06-Recruiting challenges	OPC	0.551570177	0.147661027	2	0.081445419	2
C07-Inconsistent security policies design	OPC	0.551570177	0.114243675	6	0.063013404	7
C08-Challenges of governance and leadership	OPC	0.551570177	0.138017136	3	0.076126136	3
C09-Lacking confidence	OPC	0.551570177	0.027268491	9	0.015040487	21
C10-Difficulties in integrating security into DevOps without losing speed and affecting current process and performance	PC	0.234334769	0.071004516	6	0.016638827	19
C11-Using unsuitable metrics	PC	0.234334769	0.035989662	8	0.008433629	26
C12-Compliance requirements	PC	0.234334769	0.156604968	3	0.036697989	10
C13-Neglecting change control in security	PC	0.234334769	0.116865132	4	0.027385564	13
C14-Lack of standards	PC	0.234334769	0.083647625	5	0.019601547	17
C15-Ignoring processes and security essentials leading to technical and security debt	PC	0.234334769	0.190502711	2	0.044641409	9
C16-Poor visibility of security track record	PC	0.234334769	0.067877958	7	0.015906166	20
C17-Inadequate privileged credentials and access controls causing cyber attacks	PC	0.234334769	0.277507429	1	0.065029639	5
C18-Lack of mature tools for automation and security	Technology	0.091299172	0.138395784	4	0.01263542	24
C19-Complexity in managing different tools	Technology	0.091299172	0.15628105	3	0.01426833	22
C20-Challenges of legacy system refactoring	Technology	0.091299172	0.267836211	1	0.024453224	15
C21-Use of cloud and serverless computing brings security complications	Technology	0.091299172	0.076141135	6	0.006951623	27
C22-Containers and other tools come with their own risks	Technology	0.091299172	0.057907956	7	0.005286948	28
C23-Availability and reliability of infrastructure, tools, automation, and network bandwidth	Technology	0.091299172	0.183943479	2	0.016793887	18
C24-Continuous deployment chaos	Technology	0.091299172	0.119494386	5	0.010909738	25
C25-Challenges of cost control	Business	0.122795882	0.107909667	4	0.013250883	23
C26-Conflicts between security and business	Business	0.122795882	0.480364189	1	0.058986744	8
C27-Customer readiness for frequent releases	Business	0.122795882	0.215819334	2	0.026501725	14
C28-Training users for using advanced tools	Business	0.122795882	0.19590681	3	0.024056549	16



4. Delphi Study – Analyzing (Challenges)

DevSecOps Challenge	Category of Challenges	Category Total Frequency in MLR	Frequency in MLR	Ranking within Category in MLR	Overall Ranking in MLR
C01-Cultural resistance and organizational opposition	OPC	42	7	3	6
C02-Challenges of collaboration, communication and coordination	OPC	42	20	1	2
C03-Neglecting security	OPC	42	3	4	9
C04-Lack of security awareness and responsibility	OPC	42	3	5	10
C05-Lack of security knowledge and skills, need for training	OPC	42	9	2	5
C06-Recruiting challenges	OPC	42	3	6	11
C07-Inconsistent security policies design	OPC	42	2	7	14
C08-Challenges of governance and leadership	OPC	42	1	8	19
C09-Lacking confidence	OPC	42	1	9	20
C10-Difficulties in integrating security into DevOps without losing speed and affecting current process and performance	PC	26	11	1	4
C11-Using unsuitable metrics	PC	26	3	3	12
C12-Compliance requirements	PC	26	5	2	7
C13-Neglecting change control in security	PC	26	1	6	21
C14-Lack of standards	PC	26	2	4	15
C15-Ignoring processes and security essentials leading to technical and security debt	PC	26	1	7	22
C16-Poor visibility of security track record	PC	26	1	8	23
C17-Inadequate privileged credentials and access controls causing cyber attacks	PC	26	2	5	16
C18-Lack of mature tools for automation and security	Technology	50	19	2	3
C19-Complexity in managing different tools	Technology	50	1	6	25
C20-Challenges of legacy system refactoring	Technology	50	4	3	8
C21-Use of cloud and serverless computing brings security complications	Technology	50	21	1	1
C22-Containers and other tools come with their own risks	Technology	50	3	4	13
C23-Availability and reliability of infrastructure, tools, automation, and network bandwidth	Technology	50	1	7	26
C24-Continuous deployment chaos	Technology	50	1	5	24
C25-Challenges of cost control	Business	6	2	1	17
C26-Conflicts between security and business	Business	6	2	2	18
C27-Customer readiness for frequent releases	Business	6	1	3	27
C28-Training users for using advanced tools	Business	6	1	4	28



4. Delphi Study – Analyzing (Delphi vs MLR)

DevSecOps Challenge	Category of Challenges	Category Ranking in MLR	Category Ranking in Delphi	Ranking within Category in MLR	Ranking within Category in Delphi	Overall Ranking in MLR	Overall Ranking in Delphi
C01-Cultural resistance and organizational opposition	OPC	2	1	3	8	6	12
C02-Challenges of collaboration, communication and coordination	OPC	2	1	1	7	2	11
C03-Neglecting security	OPC	2	1	4	5	9	6
C04-Lack of security awareness and responsibility	OPC	2	1	5	1	10	1
C05-Lack of security knowledge and skills, need for training	OPC	2	1	2	4	5	4
C06-Recruiting challenges	OPC	2	1	6	2	11	2
C07-Inconsistent security policies design	OPC	2	1	7	6	14	7
C08-Challenges of governance and leadership	OPC	2	1	8	3	19	3
C09-Lacking confidence	OPC	2	1	9	9	20	21
C10-Difficulties in integrating security into DevOps without losing speed and affecting current process and performance	PC	3	2	1	6	4	19
C11-Using unsuitable metrics	PC	3	2	3	8	12	26
C12-Compliance requirements	PC	3	2	2	3	7	10
C13-Neglecting change control in security	PC	3	2	6	4	21	13
C14-Lack of standards	PC	3	2	4	5	15	17
C15-Ignoring processes and security essentials leading to technical and security debt	PC	3	2	7	2	22	9
C16-Poor visibility of security track record	PC	3	2	8	7	23	20
C17-Inadequate privileged credentials and access controls causing cyber attacks	PC	3	2	5	1	16	5
C18-Lack of mature tools for automation and security	Technology	1	4	2	4	3	24
C19-Complexity in managing different tools	Technology	1	4	6	3	25	22
C20-Challenges of legacy system refactoring	Technology	1	4	3	1	8	15
C21-Use of cloud and serverless computing brings security complications	Technology	1	4	1	6	1	27
C22-Containers and other tools come with their own risks	Technology	1	4	4	7	13	28
C23-Availability and reliability of infrastructure, tools, automation, and network bandwidth	Technology	1	4	7	2	26	18
C24-Continuous deployment chaos	Technology	1	4	5	5	24	25
C25-Challenges of cost control	Business	4	3	1	4	17	23
C26-Conflicts between security and business	Business	4	3	2	1	18	8
C27-Customer readiness for frequent releases	Business	4	3	3	2	27	14
C28-Training users for using advanced tools	Business	4	3	4	3	28	16



4. Delphi Study – Analyzing (Local vs Global)

How does DevSecOps differ in local and global settings?

Opinion	Proportion
Extremely different	7.7%
Slightly different	53.8
Not different	30.8%
Uncertain	7.7%

Differences between local and global DevSecOps

- ❖ It amplifies challenges in team collaboration and communication.
- ❖ More remote work, less direct communication, more ease of misunderstanding, increased communication needs in writing.
- ❖ Synchronization/Coordination between remote teams.
- ❖ Risks/Threats spanning multiple regions.
- ❖ Independent development can propagate risks in a non-uniform way.
- ❖ Issues of data residency, for example, different nation's regulations about personal data can complicate things more than if everything is done in a single country/region.
- ❖ Everyone views DevSecOps differently. Such phenomenal magnified in global settings compared to what we see in local settings.
- ❖ Disagreements on best practices.
- ❖ Compliance with external regulations worldwide.
- ❖ Business focus and level of importance is different.



References

- [1] W. Hussain, T. Clear, S. MacDonell, Emerging trends for global devops: A new zealand perspective, in: 2017 IEEE 12th International Conference on Global Software Engineering, IEEE, 2017, pp. 21–30. doi:10.1109/icgse.2017.16.
- [2] H. Myrbakken, R. Colomo-Palacios, Devsecops: A multivocal literature review, in: Software Process Improvement and Capability Determination, Vol. 770, Springer, Cham, 2017, pp. 17–29. doi:10.1007/978-3-319-67383-7_2.
- [3] K. Carter, Francois raynaud on devsecops, IEEE Software 34 (5) (2017) 93–96. doi:10.1109/ms.2017.3571578.
- [4] Z. Ahmed, S. C. Francis, Integrating security with devsecops: Techniques and challenges, in: 2019 International Conference on Digitization, IEEE, 2019, pp. 178–182. doi:10.1109/icd47981.2019.9105789.
- [5] V. Garousi, M. Felderer, M. M̄antyl̄a, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology 106 (2019) 101–121. doi: 10.1016/j.infsof.2018.09.006.
- [6] P. Chestna, Appsec in a devops world (2016). URL <https://owasp.org/www-pdf-archive/2017-04-20-AppSecDevops.pdf>
- [7] J. Whitehead, I. Mistrik, J. Grundy, A. van der Hoek, Collaborative Software Engineering: Concepts and Techniques, Springer, 2010, pp. 1–30, (inbook). doi:10.1007/978-3-642-10294-3.<https://doi.org/10.1016/j.infsof.2012.05.002>
- [8] A. Vizcaíno, F. Garcíá, M. Piattini, S. Beecham, A validated ontology for global software development, Computer Standards and Interfaces 46 (2016) 66–78. doi:10.1016/j.csi.2016.02.004.
- [9] E. O. Conchuir, P. J. °Agerfalk, H. H. Olsson, B. Fitzgerald, Global software development: Where are the benefits?, CACM 52 (8) (2009) 127–131. doi:10.1145/1536616.1536648.
- [10] Lip-Wah Ho, Tek-Tjing Lie, Paul TM Leong, and Tony Clear. 2018. Developing offshore wind farm siting criteria by using an international Delphi Method. *Energy Policy* 113: 53–67. <http://doi.org/10.1016/j.enpol.2017.10.049>
- [11] D. S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: 2011 International Symposium on Empirical Software Engineering and Measurement, IEEE, 2011, pp. 275–284. doi:10.1109/ESEM.2011.36.



References

- [12] Orli Weiser, Yoram M. Kalman, Carmel Kent, and Gilad Ravid. 2022. 65 competencies. *Communications of the ACM* 65, 3: 58–66. <http://doi.org/10.1145/3467018>
- [13] Daniel Beiderbeck, Nicolas Frevel, Heiko A. von der Gracht, Sascha L. Schmidt, and Vera M. Schweitzer. 2021. Preparing, conducting, and analyzing Delphi Surveys: Cross-disciplinary practices, New Directions, and advancements. *MethodsX* 8: 101401. <http://doi.org/10.1016/j.mex.2021.101401>
- [14] A. Mishra, D. Dubey, Ga comparative study of different software development life cycle models in different scenarios, IJARCSMS 1 (5) (2013) 64–69.
- [15] J. Tech, What is devops. URL <http://www.jirehtechconsulting.com/what-is-devops/>
- [16] N. MacDonald, I. Head, Devsecops: How to seamlessly integrate security into devops (2016). URL <https://www.gartner.com/en/documents/3463417>
- [17] V. Braun, V. Clarke, Using thematic analysis in psychology, *Qualitative Research in Psychology* 3 (2) (2006) 77–101. doi: 10.1191/1478088706qp063oa.
- [18] V. Braun, V. Clarke, Thematic analysis: a reflexive approach (2020). URL <https://www.psych.auckland.ac.nz/en/about/thematic-analysis.html>
- [19] V. Braun, V. Clarke, Guidelines for reviewers and editors evaluating thematic analysis manuscripts (2019). URL <https://cdn.auckland.ac.nz/assets/psych/about/our-research/documents/TA%20website%20update%2010.8.17%20review%20checklist.pdf>
- [20] X. Zhao, T. Clear and R. Lal, Identifying the primary dimensions of DevSecOps: A multi -vocal literature review. *The Journal of Systems & Software* (2024), doi: <https://doi.org/10.1016/j.jss.2024.112063>.

The end, thanks.

Presenter: Gavin Zhao

School of Engineering, Computer and Mathematical Sciences
Faculty of Design and Creative Technologies
Auckland University of Technology



Generative AI and its Implications for Competencies: Work in Progress

Associate Professor Tony Clear
Department of Computer Science
and Software Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz

Clear, T., Cajander, Å., Clear, A., McDermott, R., Bergqvist, A., Daniels, M., Divitini, M., Forshaw, M., Humble, N., Kasinidou, M., Kleanthous, S., Kultur, C., Parvini, G., Polash, M., & Zhu, T. (2024). **A Plan for a Joint Study into the Impacts of AI on Professional Competencies of IT Professionals and Implications for Computing Students.** In *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 2). ACM. <https://doi.org/https://doi.org/10.1145/3649405.3659527>

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.

- An international ACM Innovation and Technology in Computer Science Education (ITiCSE) Conference working group
 - currently underway virtually and convening in July 2024:
- ***WG 2: A Multi-Institutional-Multi-National Study into the Impacts of AI on Work Practices of IT Professionals and Implications for Computing Students***
 - <https://iticse.acm.org/2024/working-groups/#wg2>
- *This coordinated, multinational working group is dedicated to examining the ramifications of AI integration within the IT sector.*
- *Employing qualitative research methods and conducting thematic analysis on interview data gathered from IT professionals [i.e. industry practitioners such as software developers] representing diverse contexts,*
- *the working group endeavours to uncover profound insights into how AI impacts work engagement, socio-technical dynamics, and the cultivation of professional competencies.*

OVERVIEW

- The concept of an ITiCSE working group
- Past working groups – history and context
- Progress of this working group
- Leaders & Rationale
- Members
- WG Steps taken so far
- Data Analysis Strategy
- Competencies
- Early insights?
- Implications for the teaching of software engineering ? - *tentative*

A Guide to ITiCSE Working Groups

Alison Clear & Tony Clear

*[from presentation to ITiCSE 2016 Working
Groups, Arequipa, Peru]*

What is an ITiCSE Working Group?

A concept unique to the ITiCSE conference

Investigating an interesting, unresolved topic in Computer Science Education

A diverse group of people, different countries, different cultures, different institutions

How is a Working Group Proposed

Concept proposed in January and submitted to ITiCSE

Successful proposals published on the ITiCSE website

Interested people apply to join

Groups of 5 to around 10 participants work electronically prior to commencement of the conference

Continue to work together for the duration of the conference

Work together face to face for the two days before the conference

Working Groups Reports Publication

Produce a mature draft report on the last day of the conference

Rigorous cycle of blind reviewing

Final report published in the ACM digital library

Outcome

- Substantial research paper
- greater than 20 pages
- Breadth and depth
- Published as a set of proceedings in the ACM Digital Library
- Unique experience

Historical Perspectives on the Computing Curriculum - Report of the ITiCSE'97 working group on Historical Perspectives in Computing Education

A Framework for Enhancing the Social Good in Computing Education: A Values Approach

Naturally Occurring Data as Research Instrument: Analyzing Examination Responses to Study the Novice Programmer

Research Perspectives on the Objects-Early Debate

Integrating cultural issues into the computer
and information technology curriculum

Computing educators oral history
project: seeking the trends

What's in a Name?: International
Interpretations of Computing
Education Terminology

Computing and sustainability: evaluating
resources for educators

Comments from Working Group Chairs

- “A chance to really leverage resources that you can’t do individually, gives you the ability to do a much wider search of the topic when you have 5-10 people together. A lot greater resources to tackle a topic”
- John Barr, July 2016

- “Builds a sense of trust”
- Tony Clear, July 2016

Comments from Working Group Chairs

- “Dynamic to be able to brainstorm to create and generate new ideas in person, just doesn’t work on your own”
- “Don’t have to be deep in a field, a great way to be initiated in the field and get up to speed quickly”
- “Get to meet people who have similar interests from all across the world”
- John Barr, July 2016

Comments from other WG participants

- Wonderful opportunity to meet new colleagues from around the world to meet on a problem of interest in CS Education
- Cary Laxer, July 2016
- One of the better community building ideas, f2f time with colleagues who have research interests in common
- Mats Daniels, Keynote speaker ITiCSE 2016

Planning the Work - Pre

- Think about preparatory activity
- Design and allocate tasks to members
- Design and prepare protocol, instruments questionnaires
- Arrange human subjects ethics protocols
- Collect examples
- Review selected literature
- Analyse data sets

Planning the Work - During

- Take stock of preparatory work completed
- Agree an agenda and plan
- Share contact details and conf. commitments
- Consider a draft structure for the report
- Work through ideas that require whole group input
- Divide the work and operate in parallel to produce the sections

Selected References

- [1] J. Noll, S. Beecham, and I. Richardson, "Global Software Development and Collaboration: Barriers and Solutions" *ACM Inroads*, vol. 1, no. 3, pp. 66-78, Sept 2010. [Online]. Available: <http://doi.acm.org.ezproxy.aut.ac.nz/10.1145/1709424.1709428>.
- [2] Clear, T., Beecham, S., Barr, J., Daniels, M., Mcdermott, R., Oudshoorn, M., Savickaite, A., and Noll, J., 2015. Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review. In *Proceedings of the Working Group Reports of the 2015 on Innovation & Technology in Computer Science Education Conference*, N. Ragonis and P. Kinnunen Eds. ACM, New York, 1-39. DOI= <http://dx.doi.org/http://dx.doi.org/10.1145/2858796.2858797>.
- [3] Beecham, S., Clear, T., Barr, J. and Noll, J. Protocol for Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review, Limerick, Ireland 2015.
- [4] T. Clear, S. Beecham, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Developments in Global Software Engineering Education," in *46th ASEE/IEEE Frontiers in Education Conference.*, D. Trytten, H. Matusovich, and M. Castro Eds. Erie, PA: IEEE, 2016.
- [5] S. Beecham, T. Clear, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Preparing Tomorrow's Software Engineers for Work in a Global Environment," *IEEE Software*, vol. 34, no. 1, pp. 9-12, Jan/Feb 2017, doi: 10.1109/MS.2017.16.
- [6] S. Beecham, T. Clear, D. Damian, J. Barr, J. Noll, and W. Scacchi, "How Best to Teach Global Software Engineering? Educators Are Divided," *IEEE Software*, vol. 34, no. 1, pp. 16-19, Jan/Feb 2017, doi: 10.1109/MS.2017.12.
- [7] T. Clear and S. Beecham, "Global Software Engineering Education Practice Continuum Special Issue of the ACM Transactions on Computing Education," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 2, p. 7, 2019.
- [8] Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., Lunt, B., Maiorana, F., Pears, A. and Pitt, F. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ACM, New York, 2020.

Planning the Work – Pre 1

- Think about preparatory activity
 - Leaders initiated project and data collection [Sweden, UK, NZ]
 - MIMN design
- Design and allocate tasks to members
 - members selected – 15
 - countries and time zones
 - NZ [but on the move], Australia, Scotland, UK,
 - Norway, Sweden, US, Cyprus, Canada

Planning the Work – Pre 2

- Subgroups established – 4
 - Literature Review
 - Data Analysis
 - Practitioner Impacts
 - Competencies
 - Spreads the work and the time zone diffs.
- Design and prepare protocol, instruments questionnaires
- Arrange human subjects ethics protocols
 - leaders completed, separately by institution
 - Data sharing agreement developed

Planning the Work – Pre 3

- Collect examples
 - in country interviews under way
 - zoom and automatic transcription and translation service
 - (GDPR certified) - Sweden
 - MS -Teams and auto transcription – NZ
 - Zoom and manual transcription by exception
 - Interviews on the move, time zones and B&B wifi!
 - Getting buy-in
 - Information Sheets, consent forms
 - Semi-structured interviews common schedule

Planning the Work – Pre 4

- Review selected literature
 - subgroup allocated
 - Some subgroup specific literature searches
- Analyse data sets
 - subgroup allocated
 - secure central repository established [UU - ALLVIS]
 - common data sharing agreement established
 - draft data analysis protocol developed

Data Analysis Protocol - Options

- ## Reviewing Options

- literature on protocol templates has been consulted (Brereton et al., 2008; King et al., 2018; Stol and Fitzgerald, 2014).
- The excerpt below from Brereton et al., (2008), although positioned at the **overall case study level**, makes some highly relevant points:
 - 1) no existing data analysis template was found to be available,
 - 2) the two phases of “*creating instruments and protocols*” and “*analysing data*”
 - see over - were considered most applicable for this protocol.
- Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). *Using a protocol template for case study planning*. Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering,
- King, N., Brooks, J., & Tabari, S. (2018). *Template analysis in business and management research. Qualitative methodologies in Organization studies: volume II: methods and possibilities*, 179-206.
- Stol, K.-J., & Fitzgerald, B. (2014). *Research protocol for a case study of crowdsourcing software development* (Lero Technical Report -TR_2014_03). <https://cora.ucc.ie/handle/10468/7039>

Developing a Data Analysis Protocol

- Full Case Study Protocol
- *"Eisenhardt (1989) concerned with using case studies to develop theories suggests the following activities:*
 - **Getting started** by defining the research question and *a priori* questions but not defining hypotheses or theory.
 - **Selecting cases** by considering a particular population and using theoretical concerns to focus on specific cases.
 - **Crafting instruments and protocols** using multiple data collection methods, using qualitative and quantitative data, and preferably multiple researchers.
 - **Entering the field** i.e. incorporating field notes with data analyses and using flexible and opportunistic data collection methods.
 - **Analysing the data** both within case and across cases.
 - **Shaping hypotheses** by iterative tabulation of evidence looking for identified constructs, replication logic across cases, and looking for evidence to explain why relationships exist.
 - **Enfolding the literature** i.e. comparing with existing similar and conflicting literature.
 - **Reaching closure** using the concept of "theoretical saturation" which says researchers stop looking for more cases/data when they believe more data will only give a marginal improvement to the existing results". (Brereton et al., 2008)

Forms of Template Analysis

- King et al., (2018) presenting “*Template Analysis*” :
- “*Thematic analysis is widely acknowledged as an accessible and useful approach to the analysis of rich and meaningful qualitative data—indeed, Clarke and Braun (2013) describe thematic analysis as the ‘basic’ method of qualitative data analysis*”.
- *Template Analysis a particular style of thematic analysis*
- distinguish between *Generic Template Analysis* as a *method*, and as applied within a broader research *methodology* such as grounded theory or Interpretative Phenomenological Analysis [IPA],
- observe that issues about differing philosophical, theoretical or methodological positions can be better accommodated, as noted below:
- *Generic styles of thematic analysis can provide researchers more flexibility and adaptability to the particular requirements of their own work—rather than applying a methodology as a whole package.*

Steps in Template Analysis

- In template analysis the general steps in the process are defined by King et al., (2018) as below:
- *procedural steps characteristically followed in Template Analysis are:*
 - *Familiarization with the data*
 - *Preliminary coding*
 - *Clustering*
 - *Developing the initial template*
 - *Modifying the template*
 - *Defining the ‘final’ template*
 - *Using the template to interpret the data*
 - *Writing-up*
 -

WG Data Analysis Subgroup

- data analysis sub-group will conduct the three steps prior to developing the initial template “*Familiarization with the data, preliminary coding, clustering*”, using a subset of the transcript data, to derive an initial template using a defined spreadsheet for coding each transcript.
- The template will be refined as the process progresses as noted by (*King et al., 2018*):
 - *Revisions might include: re-defining themes to increase or narrow their scope (shown through moving them up or down hierarchical levels), moving themes between clusters, adding new themes—or even entire new clusters—and deleting themes that have become redundant as the template has developed.*
- Version one of the template allows for coding of each question.
- For some questions a predefined set of deductive codes as an initial set of codes for that question,
- for other questions a more inductive strategy may be more suitable.

WG Analysis - Work Allocation

- Decisions about coding parties need to be made.
- It is presumed at this stage that work will be distributed amongst the WG members to share the load, once the template is sufficiently stable.
- Members of the WG will generally code whole transcripts, possibly adopting a pair coding strategy, first individually coded then by comparison between pairs.
- But it may be preferable to assign questions to sub-groups, e.g. question 4.3 by the competencies sub-group?
-
- The transcripts will be stored in the Uppsala ALLVIS repository, and made available to members whose institutions have signed the data sharing agreement.

Planning the Work – During the WG Mtg

Partly In Progress

- Take stock of preparatory work completed
- Agree an agenda and plan
- Share contact details and conf. commitments
- Consider a draft structure for the report
- Work through ideas that require whole group input
- Divide the work and operate in parallel to produce the sections

Planning the Work – During -2 TBD

- Allocate a paper editor
- Review components and drafts as they come in
- Prepare for presentation day one of conference
 - Update audience on topic & progress
 - Seek feedback in selected areas
- Build consensus towards a final draft
- Produce consolidated mature draft and handover to WG Chairs on the last day of the conference before leaving
- Advise a list of three sound reviewers

Planning the Work – Post **TBD**

- Ensure responsibility allocated for final draft
- Share timetables [holidays and semester commitments]
- Work on final draft [version tracking mechanism in place] and share amongst whole or core team
- Submit and wait for reviews
- Make changes as advised by reviewers
- Submit camera ready copy and hope!

CC2020 PROJECT

CURRICULUM

DIRECTIONS



Software for
a better world

Clear, A., Parrish, A., &
CC2020 Task Force.
(2020). *Computing
Curricula 2020 - CC2020*

*Paradigms for Future
Computing Curricula*
Retrieved from New
York:

<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

A Computing Curricula Series Report
2020 December 31

Computing Curricula 2020

CC2020

Paradigms for Global Computing Education

encompassing undergraduate programs in
Computer Engineering
Computer Science
Cybersecurity
Information Systems
Information Technology
Software Engineering
with data science

**A Framework for
Structuring
Competency
Related
Responses?**



Association for
Computing Machinery



IEEE
computer
society

CC2020 PROJECT MODEL

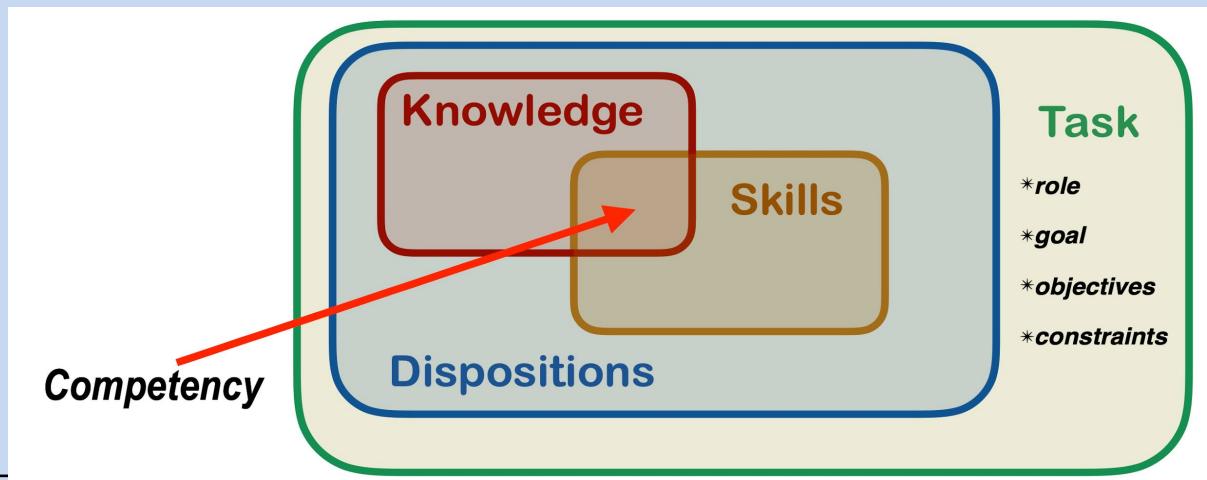
ASPECTS OF KNOWLEDGE BASED LEARNING [Established Curriculum Paradigm]

- KA's [knowledge areas]
- KU's [knowledge units]
- LO's [Learning outcomes]

COMPONENTS OF COMPETENCY BASED LEARNING [CC2020 MODEL]

– **Competency = [Knowledge+ Skills+ Dispositions] in Task**

- A competency structure (see Fig. 1) shows knowledge, skills, and dispositions that are observable in the accomplishment of a task, a task that prescribes purpose within a work context [24].



CC2020 – COMPETENCY BASED LEARNING

- Knowledge is the “*know-what*” dimension of competency that is factual.
- Skills express the “*know-how*” and usually develop over time and with practice.
- Dispositions frame the “*know-why*” dimension of competency, which prescribes a requisite character or quality in task performance.
- Task is the construct that frames the skilled application of knowledge and makes dispositions concrete.

SUBJECT/CONTENT KNOWLEDGE

3.2.1 *Knowledge Vocabulary for Computer Science Competencies.*

The CS2013 document divides computer science knowledge into 18 KAs subdivided into 225 KUs. For example, the Software Engineering KA is divided into ten KUs:

- (1) SE/Software Processes
- (2) SE/Software Project Management
- (3) SE/Tools and Environments
- (4) SE/Requirements Engineering
- (5) SE/Software Design
- (6) SE/Software Construction
- (7) SE/Software Verification and Validation
- (8) SE/Software Evolution
- (9) SE/Software Reliability
- (10) SE/Formal Methods

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

PROFESSIONAL & FOUNDATIONAL KNOWLEDGE

Table 4: Professional & Foundational Knowledge Areas extending CS2013

Tag	Knowledge Area
PK-1	Oral Communication & Presentation
PK-2	Written Communication
PK-3	Problem Solving and Trouble Shooting
PK-4	Project/Task Organization & Planning
PK-5	Collaboration and Teamwork
PK-6	Research and Self-Learning
PK-7	Multi-Task Prioritization & Management
PK-8	Relationship Management
PK-9	Analytical and Critical Thinking
PK-10	Time Management
PK-11	Quality Assurance / Control
PK-12	Mathematics and Statistics
PK-13	Ethical Intercultural Perspectives

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

SKILLS

Table 4.3. Levels of Cognitive Skills Based on Bloom's Taxonomy

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Exhibit memory of previously learned materials by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, and giving descriptions.	Solve problems in new situations by applying acquired knowledge, facts, techniques, and rules in a different way.	Examine and break information into parts by identifying motives or causes; make inferences and find evidence to support solutions.	Present and defend opinions by making judgments about information, validity of ideas, or quality of material.	Compile information together in a different way by combining elements in a new pattern or by proposing alternative solutions.

Clear, A., Parrish, A., & CC2020 Task Force. (2020). *Computing Curricula 2020 - CC2020 – Paradigms for Future Computing Curricula* Retrieved from New York:
<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

CC2020 DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., ... Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

SPECIFYING COMPETENCIES?

KA-Ref#	Title	Competency Statement	Dispositions	Knowledge-Skill Pairs				Traced Competencies	Notes
				KU / Broader Knowledge Description	Topic/LO Description	Knowledge Element	Paired Skill Level		
Our reference number goes here. E.g., "KA-##"	Short Title	Competency statement goes here (Natural Language) that embeds the task/context and suggests the K-S-D breadth and depth	List top applicable dispositions: <ID+descriptor>. Note that dispositions are not aligned with KS-pairs.	CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level	List other competency reference #'s that this competency expects or depends upon. Often null.	Any notes or comments
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Often null
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Note deviations from Bloom's level in CS2013
				CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	New IDs for extensions to CS2013	Bloom's Level		
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		

Figure 5: Competency specification in a tabular format

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

EXPRESSING COMPETENCIES?

“...most appropriate to express competencies at the level of entire KAs, **from the point of view of a graduate or professional.**

When we “designed” a competency statement, we focused on a particular knowledge area and **looked for collections of topics and/or learning outcomes (LOs) that could be observed in tasks and at particular skill levels.**

...there was significant variance in how this was approached, and how the resulting competency was formulated. **Dispositions were often implied at first, but refinement of the statements made these both more clear and more relevant.”**

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020*

ACM Conference on Innovation and Technology in Computer Science Education. New York: ACM.

A SAMPLE COMPETENCY STATEMENT

Table 7: IS-4: IS/Basic Machine Learning

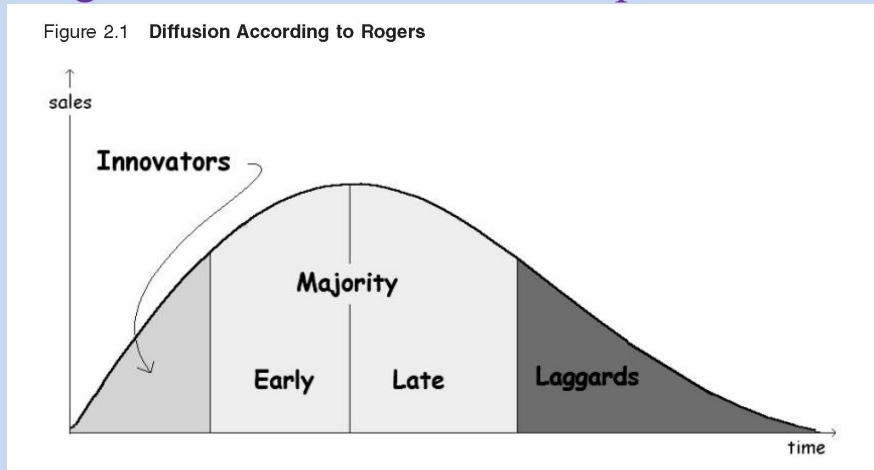
Statement	Dispositions
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.	D-6 (Responsible) D-10 (Meticulous)

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

Table 7: IS-4: IS/Basic Machine Learning

Statement		Dispositions		
Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.			D-6 (Responsible)	D-10 (Meticulous)
IS/Basic Machine Learning	List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised.	IS-BML-1	B-II	
	Identify examples of classification tasks, including the available input features and output to be predicted	IS-BML-2	B-II	
	Explain the difference between inductive and deductive learning	IS-BML-3	B-II	
	Describe over-fitting in the context of a problem	IS-BML-4	B-II	
	Apply the simple statistical learning algorithm such as Naive Bayesian Classifier to a classification task and measure the classifier's accuracy	IS-BML-5	B-III	
Prof. & Foundational Knowledge	Analytical and Critical Thinking	PK-9	B-III	
	Mathematics and Statistics	PK-12	B-III	
IS/Basic Machine Learning	Definition and examples of broad variety of machine learning tasks, including classification	IS-BML-a	B-II	
	Inductive Learning	IS-BML-b	B-II	
	Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees	IS-BML-c	B-III	
	The over-fitting problem	IS-BML-d	B-II	
	Measuring classifier accuracy	IS-BML-e	B-III	

- Seems to be early days yet?
- Variable acceptance and adoption
- Outright ban to enthusiastic incorporation in ways of working



Libai, B., Mahajan, V., & Muller, E. (2017). Can you see the chasm?: Innovation diffusion according to rogers, bass, and moore. In *Review of marketing research* (pp. 38-57). Routledge.

- Active experimentation – personal and institutional pilot studies - FOMO
- Some vendor guided adoptions e.g. Microsoft and Copilot [Copilot not an Auto-pilot]
- Large Corporate risk assessment before a standard approach adopted

IMPLICATIONS FOR TEACHING SE 1?

- A shift to a dialogic model of interacting with systems
- Prompt Engineering and refinement of strategies
- API's to connect with backend chatbots easily implemented [e.g. Vercel
<https://vercel.com/changelog/next-js-ai-chatbot-2-0>]
- Good reviews:
 - 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>
- What to do about various forms of cheating?

IMPLICATIONS FOR TEACHING SE 2?

- What to do about various forms of cheating?
- How to shift to acceptable and valuable behaviour
- Just today's 4GL and end-user development fad?
- How to develop judgement and handling reviews and errors
- Hallucinations an inherent design of LLM technology?
- How to co-exist with Gen-AI services?
- How to redesign courses and assessments?

IMPLICATIONS FOR TEACHING SE 3?

- Commercial services and downsides
- Cost and accessibility issues?
- What to do about various forms of cheating?
- Observed violations of service agreements – who reports miscreants?
- Privacy and IP rights
- Whose work and evolving citation standards?
- AI systems and embedded biases
- Ethical awareness

ACADEMIC INTEGRITY - THEN?

You **can** use AI when writing or preparing a presentation if you use it to help you improve small aspects of your work, such as:

- grammar and punctuation, and
- formal terminology.

You **cannot** use AI to generate ideas when writing, preparing a presentation or creating an artwork/artefact (unless otherwise specified in your assessment instructions). The ideas have to come from you and your course materials.

<https://canvas.aut.ac.nz/courses/7624/pages/referencing>
9/01/2023 - and evolving

 After drafting a paragraph for a group assignment, Jude uses the Grammarly AI writing assistant to check their writing. Grammarly identifies spelling mistakes and where the writing is too wordy. Jude reads the suggested changes and explanations about why some of their writing can be improved. They then make some improvements to their work.

Jude has acted appropriately because they have:

- only used AI to identify small errors in their own work,
- used the AI's explanations to learn more about academic writing, and
- made their own improvements.

 Karl quickly writes an essay on the day it needs to be submitted. He provides ChatGPT with the assessment task instructions, a list of required readings, and his essay, and he then prompts ChatGPT to write a better version. ChatGPT completely reorganises the structure and content of Karl's work. Karl then submits ChatGPT's version because it looks better than his own.

Karl has not acted appropriately because he has:

- submitted work that he did not do himself (he submitted ChatGPT's work), and this is a breach of AUT's academic integrity guidelines. Even if Karl had acknowledged his use of ChatGPT in the essay, this would still be inappropriate because he did not write the essay himself.

Figure 4: Appropriate and inappropriate uses of AI when writing and presenting

Reference

AAIN Generative AI Working Group. (2023). *AAIN Generative Artificial Intelligence Guidelines*, Australian Academic Integrity Network. <https://doi.org/10.26187/sbwr-kq49> ↗
<https://doi.org/10.26187/sbwr-kq49>

ACADEMIC INTEGRITY - NOW?



Submit only your own work*

Acknowledge all sources of information you use by:

- using the appropriate referencing style, and
- paraphrasing or quoting any words/ideas that are not your own.



Do not submit work done by others, such as:

- other students*
- friends or relatives
- assignment writing services
- artificial intelligence software**, like ChatGPT.

Do not submit work that you have previously submitted for assessment.

<https://canvas.aut.ac.nz/courses/7624/pages/academic-integrity> 10/05/2024

- and evolving

Figure 1: How to maintain academic integrity

* Unless the work is for a designated (group) task

** If your assessment requires the approved use of artificial intelligence, you must acknowledge wherever you do so in your work with an appropriate in-text reference (see guidelines for APA [↗](https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312), Chicago [↗](https://aut.ac.nz.libguides.com/turabian/personalcomms#s-lg-box-22370438), and Harvard [↗](https://aut.ac.nz.libguides.com/c.php?g=919289&p=6648988#s-lg-box-22370440) referencing styles).

ACADEMIC INTEGRITY – NOW?

Artificial intelligence software

Referencing the information generated by an algorithm or artificial intelligence software tool, such as ChatGPT.

Credit the author of the algorithm/AI tool with a reference list entry and an in-text citation.

Reference list format

Who	When	What	Where
Author of AI tool.	(Year released).	<i>Title of tool</i> (Version) [Description].	URL to access tool

Reference list example

OpenAI. (2023). *ChatGPT* (Mar 14 version) [Large language model]. <https://chat.openai.com/chat>

In-text citation examples

OpenAI (2023) generated the following response when...

...that indicates a limitation of the software (OpenAI, 2023).

In your writing

- Describe how you used the tool.
- Provide the prompt you used.
- Provide any portion of the relevant text that was generated in response.
- Document the exact text created because tools like ChatGPT generate unique responses in each chat session, even if given the same prompt.

More information

<https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312>

<https://apastyle.apa.org/blog/how-to-cite-chatgpt>

ChatGPT – Terms of Use



What you can do. Subject to your compliance with these Terms, you may access and use our Services.

In using our Services, you must comply with all applicable laws as well as our Sharing & Publication Policy, Usage Policies, and any other documentation, guidelines, or policies we make available to you.

What you cannot do. You may not use our Services for any illegal, harmful, or abusive activity. For example, you may not:

- Use our Services in a way that infringes, misappropriates or violates anyone's rights.
- ...
- Represent that Output was human-generated when it was not.
-
- Use Output to develop models that compete with OpenAI.

OpenAI. (2024,
January 31 2024).
Terms of Use.
Retrieved
10/05/2024 from
<https://openai.com/policies/terms-of-use>

ChatGPT – Terms of Use

Your content. You may provide input to the Services (“Input”), and receive output from the Services based on the Input (“Output”). Input and Output are collectively “Content.”

“Content.” You are responsible for Content, including ensuring that it does not violate any applicable law or these Terms. You represent and warrant that you have all rights, licenses, and permissions needed to provide Input to our Services.

Our use of content. We may use Content to provide, maintain, develop, and improve our Services, comply with applicable law, enforce our terms and policies, and keep our Services safe.

Opt out. If you do not want us to use your Content to train our models, you can opt out by following the instructions in this Help Center article. Please note that in some cases this may limit the ability of our Services to better address your specific use case.

OpenAI. (2024, January 31 2024). *Terms of Use*. Retrieved 10/05/2024 from <https://openai.com/policies/terms-of-use>

BROADER IMPLICATIONS

- Giving data into the datacube of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
<https://commons.wikimedia.org/w/index.php?curid=58277452>



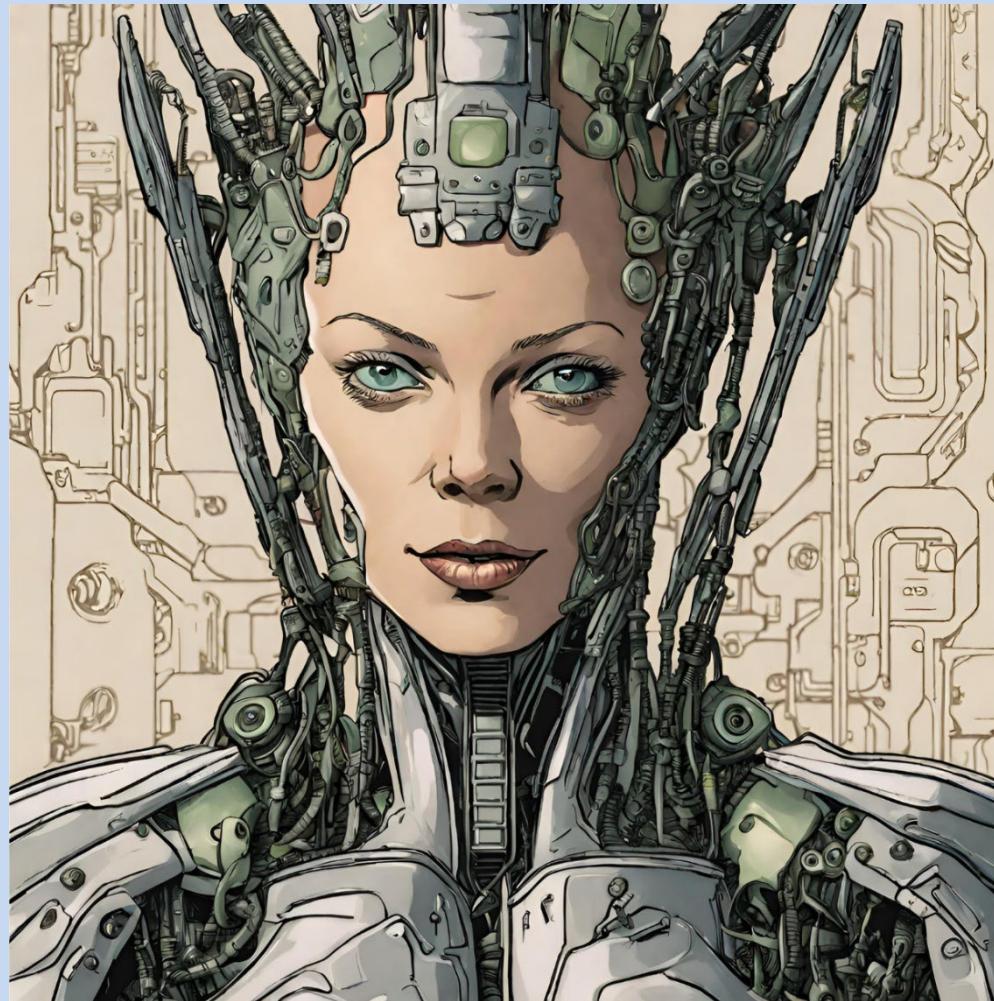
IMPLICATIONS?

- Giving data into the maw of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
https://upload.wikimedia.org/wikipedia/commons/5/52/Trekkie_-_Borg_-_Star_Trek_Convention_%2889504912575%29.jpg



IMPLICATIONS FOR TEACHING SE?

- Giving data into the clutches of the BORG and Getting it Back?



[https://www.mediawiki.org/wiki/File:
Borg_Queen_by_Canva_AI.png](https://www.mediawiki.org/wiki/File:Borg_Queen_by_Canva_AI.png)

Polski: Artystyczny wizerunek królowej Borg z uniwersum Star trek wygenerowany przez oprogramowanie Canva Magic Multimedia

English: Borg Queen from Star Trek universe artistic vision generated by Canva Magic Multimedia

4 April 2024

Own work

[Canva Magic Multimedia](#)

CONCLUSION

- Covered ITiCSE working groups and this one
 - *Implications of [Gen] AI for IT professionals and competencies*
- Covered WG creation and progress to date
- Outlined Data Analysis Strategy and Protocol
- Defined Competencies from a CC2020 perspective
- Discussed early insights about the state of the art?
- Explored tentative implications for the teaching of software engineering ?
- Still early work
- To be fleshed out in the final report [after WG meets 4 – 7 July and revisions by end of year - all going well ☺]

Generative AI and its Implications for Competencies: Work in Progress

Questions?



Associate Professor Tony Clear
Department of Computer Science and Software
Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz



Planning the Work – During -2

- Allocate a paper editor
- Review components and drafts as they come in
- Prepare for presentation day one of conference
 - Update audience on topic & progress
 - Seek feedback in selected areas
- Build consensus towards a final draft
- Produce consolidated mature draft and handover to WG Chairs on the last day of the conference before leaving
- Advise a list of three sound reviewers

Planning the Work - Post

- Ensure responsibility allocated for final draft
- Share timetables [holidays and semester commitments]
- Work on final draft [version tracking mechanism in place] and share amongst whole or core team
- Submit and wait for reviews
- Make changes as advised by reviewers
- Submit camera ready copy and hope!

DEFINING & DESCRIBING SKILLS

Table 5: Sample CC2020 Skill Level Vocabulary [3]

	I Remembering	II Understanding	III Applying	IV Analyzing	V Evaluating	VI Creating
Definitions	Exhibit memory of previously learned materials...	Demonstrate understanding of facts and ideas...	Solve problems in new situations by applying...	Examine and break information into parts...	Present and defend opinions by making judgments about information...	Compile information together in a new way...
Verbs	Choose, define, find, how, label, list, match, name, omit, recall, relate, select, show, spell, tell, what, when, where, which, who, why	Classify, compare, contrast, demonstrate, explain, extend, illustrate, infer, interpret, outline, relate, rephrase, show, summarize, translate	Apply, build, choose, construct, develop, experiment with, identify, interview, make use of, model, organize, plan, select, solve, utilize	Analyze, assume, categorize, classify, compare, contrast, discover, dissect, distinguish, divide, examine, function, infer, inspect, list, relate, simplify, survey, take part in, test for	Agree, appraise, assess, award, choose, conclude, criteria, criticize, decide, deduct, defend, determine, disprove, estimate, evaluate, explain, interpret, judge, justify, measure, prioritize, prove, rate, recommend, select	Adapt, build, change, combine, compose, construct, create, design, develop, elaborate, estimate, improve, invent, make up, maximize, minimize, modify, optimize, originate, plan, predict, propose solution, solve, test theory

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

ON TEACHING DISPOSITIONS?

- in discussing whether dispositions could be taught [Clear, 2021], I noted that “... *a disposition* “concerns not what abilities people have, **but how people are disposed to use those abilities**” [Schussler, 2006].

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?”

[Clear, 2021]

Clear, T. (2021). THINKING ISSUES: Is Agility a Disposition and Can it be Taught? . *ACM Inroads*, 12(1), 13-14.
doi:10.1145/3447870

CC2020 DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science

Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science*

Education. New York: ACM.



Generative AI: a Critique

Associate Professor Tony Clear
Department of Computer Science
and Software Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz

Clear, T. (2024). THINKING ISSUES: Large Language Models, the 'Doctrine of Discovery' and 'Terra Nullius' Declared Again? *ACM Inroads*, 15(2), 6–9. <https://doi.org/10.1145/3638564>

- Large Language Models (LLMs) and
- recent hype around generative AI and ChatGPT,
- profound questions around data, rights, and ownership claims,
- how any such claims might be viewed critically by computing educators and their students

Clear, T. (2024). THINKING ISSUES: Large Language Models, the 'Doctrine of Discovery' and 'Terra Nullius' Declared Again? *ACM Inroads*, 15(2), 6–9. <https://doi.org/10.1145/3638564>

- Google™ at the vanguard of a “a third wave of colonization
- for countries such as New Zealand,
- **First by empire and the gun,**
- **Then by the dollar and economic might,**
- **Now by the shaping of discourse through distorted delivery of information.”**
- So, do large language models and the possibilities provided by generative AI merely represent another extension of this “third wave” or are they radically different?

Clear, T. (2006, Dec). Google™ - "Do No Evil" - Yeah Right! *SIGCSE Bulletin*, 38(4), 8-10.
<https://doi.org/https://doi.org/10.1145/1189136.1189142>

SURVEILLANCE CAPITALISM

- Shoshana Zuboff [11,12]
- strategies adopted by ‘big tech’ companies
- pre-emptively appropriating rights to newly conceived forms of data,
- expanding to the new world of ‘big data’
- and the economic sea change known as
- “surveillance capitalism.”

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- ““Big data’ constituted by capturing small data
- from individuals’ computer-mediated actions and utterances in their
- pursuit of effective life.” [11]
- Big tech companies accumulating “not only surveillance assets and capital, but
- also rights … accomplished through a form of unilateral declaration that most
- closely resembles the social relations of a pre-modern absolutist authority.” [11]

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- Zuboff refers to a calculated secrecy, “concealing a new political equation in which
- Google’s concentrations of computational power
- **brush aside users’ decision rights** as easily as King Kong might shoo away
- an ant, all accomplished offstage where no one can see?” [12]

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

- “euphemisms operate as those on the earliest maps of the North American continent,
- Whole regions labelled with terms such
- as “heathens,” “infidels,” “idolaters,”
- “primitives,” “vassals,” and “rebels.”
- On those euphemisms, native peoples—their places and claims—were
- deleted from the invaders’ moral and legal equations,
- legitimating the acts of taking and breaking that paved the way for church and monarchy”

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- New Zealand may frame itself as a “small, advanced economy,”
- let’s be quite clear—
- we, along with the citizens of many other countries, are the newly colonized!

- So how does this process of what we
- might term ‘neo-digi-colonization’ work?
- how might AI and the controllers of LLMs
- go about usurping our ‘places and claims’?
- One well known strategy is that of
- “Move fast and Break things” [10],
- where speed and greed rather than sense predominate

- “In high-stakes AI research,
- data work is often seen as low-level grunt
- work ... and incentive structures generally
- encourage a ‘move fast and break things’
- mentality over careful scientific work.

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.05532*.

THE NEED FOR OPENNESS

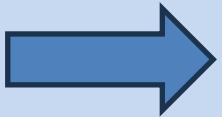
- Liesenfeld and colleagues
- argue for greater “openness in the fast-moving field of instruction-tuned large language models.
- We have found projects at varying stages of implementation,
- documentation, and useability.
- Most of them offer access to source code and some aspects of pre-training data,
- Sometimes in legally ambiguous ways.”

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.05532*.

- “There are many shades of openness...yet all of the projects surveyed here
- are significantly more open than ChatGPT.
- ChatGPT was announced in a company blog post
- rolled out to the public with an interface
- designed to capture as much free human labour as possible,
- but without any technical documentation.” [6]
- **‘Free labour’ has of course been a long-standing marker of colonization!**

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.0553*

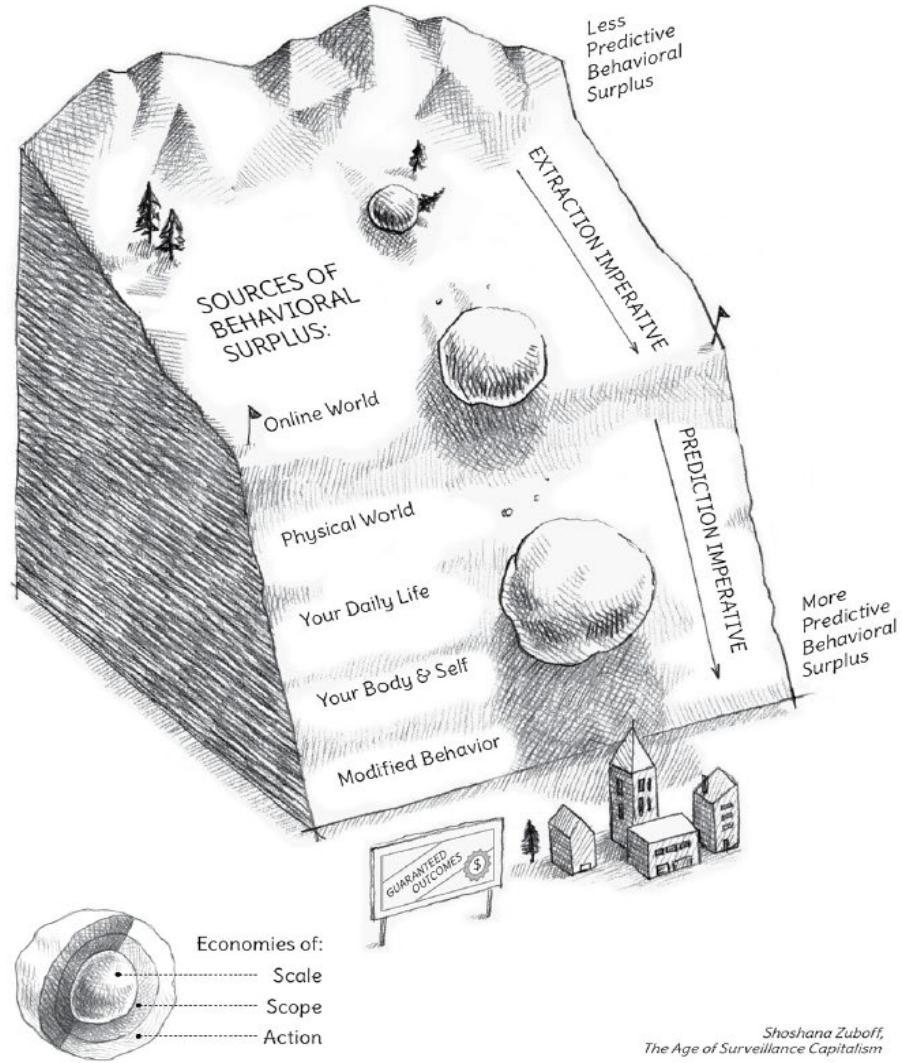
Data supports..



Prediction



Enables Control



Zuboff, S.
(2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

Figure 3: The Dynamic of Behavioral Surplus Accumulation

LAND AND PROPERTY CLAIMS

- So if our data is to become our newly appropriated land
- how do we typically lay claims to property and land?
- Mechanisms for property claims based on Roman law (Simpson, 1993) and identify:
 - Occupation (only when ‘belonging to no-one’ or ‘terra nullius’ for land)
 - Conquest
 - Cession (by treaty)

Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

DOCTRINE OF DISCOVERY

- European colonization had sanction from the Catholic Church, through
 - the ‘doctrine of discovery,’
 - provided legal justification for the occupation of ‘newly discovered’ territories by the colonizing authorities who laid claim to them.
 - “The doctrine was laid out in a series of papal “bulls,” or decrees; the first one was issued in 1452. They authorized colonial powers such as Spain and Portugal to seize lands and subjugate people in Africa and the “New World,” as long as people on the lands were not Christians.

DOCTRINE REPUDIATED

- Nearly 500 years after papal decrees were used to rationalize Europe's colonial conquests,
- the Vatican repudiated those decrees on Thursday,
- the "Doctrine of Discovery" that was used to justify snuffing out Indigenous people's culture and livelihoods
- is not part of the Catholic faith. [3]

Chappell, B. (2023, March). The Vatican repudiates 'Doctrine of Discovery,' which was used to justify colonialism.

NPR. <https://www.npr.org/2023/03/30/1167056438/vatican-doctrine-of-discovery-colonialism-indigenous>

- while the legitimacy of the doctrine has now belatedly been repudiated and
- “the mindset of cultural or racial superiority which allowed for that objectification or subjection of people has been renounced,” [3]
- much of the legal basis for land ownership in settler societies still relies upon it.
- As one example, Bess has reported that:
- The United States Supreme Court in 2005, relying on a series of Indian law cases going back to 1823, specifically cited the Doctrine in its decision denying the right of the Oneida Indian Nation of New York to regain its territory. Justice Ruth Bader Ginsburg wrote in the 2005 decision. “Under the Doctrine of Discovery ...fee title to the land occupied by Indians when the colonists arrived became vested in the sovereign – first the discovering European nation and later the original States and the United States.”

TERRA NULLIUS IN AUSTRALIA?

- Simpson's critique of the use of the doctrine and the tweaking of the definition of the term "terra nullius" in Australia,
- "precedent is a deity greater than universally accepted history in some cases... the judiciary ignored international law and history and called its decisions 'precedent';
- in Mabo, it rewrote international law and the common law, and
- called the decision 'justice'.
- What must the original inhabitants of this land make of such mysticism?"
- **So summarising the basis for claims below**

Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

"**Occupation** derives from the natural mode of acquisition in Roman law known as *occupatio*. *Occupatio* could only confer title over objects which were *res nullius* - i.e. belonging to no-one".

The doctrine, of course, became known as *terra nullius* when it was applied exclusively to land rather than objects generally. If land was *terra nullius* it could be acquired through occupation. The corollary to this was that title **could only be acquired through occupation if the land was *terra nullius***. *Terra nullius* was land that was either deserted or uninhabited ...or inhabited by uncivilized or disorganized groups (this was the general international law view).

In cases where the land was occupied by peoples having a system of social organization, land could only be acquired or colonized through either **conquest** or **cession (treaty)**. This was typically the practice in Asia, Latin America and North America... In Asia, *terra nullius* was thought to have little relevance to the well-organized tribal societies in existence at the time of European colonization, and most territorial acquisitions occurred by **cession or treaty**... The Spanish, on the other hand, acquired sovereignty over Latin America by **conquest**..while in North America a whole variety of methods were used ranging from treaties to conquest, but generally not mere occupation."

In New Zealand the Maori people were thought to fall into category of **cession**, and therefore treaties were concluded between the indigenous inhabitants and the European settlers... Notoriously, of course, Australia was regarded as falling into the category of ***terra nullius***.

Figure 1: Methods of acquisition for Land and Property [Ex. 9]

AI AND PROPERTY RIGHTS?

- What about AI and property rights.
- How will these legal games play out in the face of today's new “terra nullius” being
- the data libraries, datasets, and scrapings from territories of the internet,
- Newly ‘discovered’ by the creators of LLMs such as ChatGPT
- and their design and use of generative AI systems based on these
- implicit territorial claims?



Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

- In the face of this new attempted form of colonization,
- major risks exist for users and owners of systems,
- and major battles lie ahead over intellectual property rights.
- In the New Zealand context, the users of chatbots such as ChatGPT should exercise caution over ownership issues with copyright experts questioning “who ‘owns’ parts of the essay, song lyrics, poems, speeches, blogs or other features that the chatbot spouts out? [arguing that] the chatbot technology arrived so quickly that users have not had time to think through the implications.”

- New Zealand copyright expert Moon has made the point that
 - “somewhere along the way, those words, sentence sequences,
 - images and sounds are likely to have been input by a human.
-
- And under New Zealand law, that content is automatically protected
 - by copyright for the life of the author or creator, and 50 years beyond that person’s
 - death.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- “in the US, only works produced by a human can be registered.
- But in New Zealand, all computer-generated work, including ‘new’ content created by a chatbot like ChatGPT, is protected under copyright for 50 years.
- But apart from ChatGPT outputs, there is a copyright risk for AI users who “educate”
- or “train” their systems to generate material, Moon says. That includes AI systems
- like GitHub Copilot, used to help write computer programs, which are protected
- under literary works.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- Further “Moon predicts there could well be a debate over whether New Zealand
 - should drop its copyright protection for computer-generated works altogether.
-
- He thinks it’s a debate worth having, given that few other countries offer that
 - protection.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- In Australia concerns raised about the Books3 dataset,
- chief executive of Australia's Copyright Agency,
- described the Books3 development as 'a free kick to big tech' at the expense of Australia's creative and cultural life.
- 'We're going to need greater transparency – how these tools have been developed, trained, how they operate – before people can truly understand what their legal rights might be,' she said...
- Australian copyright law protects creators of original content from data scraping.

Burke, K. (2023, 28/09/2023). 'Biggest act of copyright theft in history': thousands of Australian books allegedly used to train AI model. *The Guardian*. <https://www.theguardian.com/australia-news/2023/sep/28/australian-books-training-ai-books3-stolen-pirated>

AI AND LAWSUITS - THE US?

- Litigation in the US against ChatGPT creator OpenAI over use of allegedly pirated book datasets, Books1 and Books2 (which do not appear to be affiliated with Books3) has already commenced.
- *“The New York Times has sued OpenAI and Microsoft for the unpermitted use of Times articles to train GPT large language models. The case could have a significant impact on the relationship between generative AI and copyright law, particularly with respect to fair use, and could ultimately determine whether and how AI models are built.”* [Pope, 2024]

Pope, A. (2024, 26 July). NYT v. OpenAI: The Times's About-Face. *Blog Essay*.
<https://harvardlawreview.org/blog/2024/04/nyt-v-openai-the-timess-about-face/>

- risks and legal exposure for
- educational institutions and students using
- generative AI through LLMs and systems
- such as ChatGPT are unclear.

- **Problem**

- Proliferating data collection, advanced algorithms, and powerful computers have made it easy to piece together information about individuals' private lives from public information as controls over information privacy become increasingly ineffective

- **Policy Implications**

- Proliferating data collection, use, and publication present rapidly accumulating risks of private information disclosure that require regulation to mitigate.

- Traditional approaches to anonymization, deidentification, and disclosure control fail to protect information at its current scale and are entirely unable to deal with new ways of utilizing information, such as generative AI.

- Inherently imperfect legal and technical solutions must balance individuals' and stakeholders' needs for data privacy and accuracy.

- Altman, M., Cohen, A., & Nissim, K. (2024). *ACM TechBrief: Data Privacy Protection*. Association for Computing Machinery.

COLONIZATION - OLD AND NEW STRATEGIES

...well-rehearsed strategies for colonization shine through.

...mindsets of cultural and racial superiority that justified the legal subterfuge of “terra nullius,”

when ‘big tech’ seeks to arrive and take the digitized traces of our lives and use them without permission or recompense to generate copies and derivative analogues for their own commercial purposes.

A REGULATORY RACE?

In the face of this inexorable private ‘land grab’

An ongoing ‘regulatory race’ is one necessary response as a strategy for asserting human rights through the public sphere.

Capitalist societies have long shown the ability to regulate natural monopolies for the common good,

for instance the European Union’s definition of ‘big tech’ companies as ‘utility providers’ of “very large commercial online platforms.”

THE EU AND REGULATION

Ironically the European Union, the original source of colonisation, has shown insight through the ability to act with respect to AI

through its recent “Regulation on Artificial Intelligence (the EU AI Act)” [5].

The Act classifies AI into four levels of risk based on the intended use of a system:

- 1) unacceptable;
- 2) high;
- 3) limited, and
- 4) minimal risk,

where the Act is most concerned with ‘high-risk AI.’

THE EU AND REGULATION

I can envisage the dominant large language models in due course being similarly defined as ‘utility platforms’ with accompanying and evolving regulations,

as a likely development to rebalance the private and public spheres.

With our students we need to become aware that

resistance and vigilance on the part of citizens in these data wars

will be a necessary part of this ongoing struggle against neo-digi-colonization.

IMPLICATIONS FOR TEACHING SE 1?

- A shift to a dialogic model of interacting with systems
- Prompt Engineering and refinement of strategies
- API's to connect with backend chatbots easily implemented [e.g. Vercel
<https://vercel.com/changelog/next-js-ai-chatbot-2-0>]
- Good reviews of the 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>

IMPLICATIONS FOR TEACHING SE 2?

- What to do about various forms of cheating?
- How to shift to acceptable and valuable behaviour
- Just today's 4GL and end-user development fad?
- How to develop judgement and handling of reviews and errors
- Hallucinations an inherent design of LLM technology?
- How to co-exist with Gen-AI services?
- How to redesign courses and assessments?

JUDGEMENT AND ERRORS?

- How to develop judgement and handling of reviews and errors

...overall feedback very detailed, long, and not always well ordered.

48 % of the generated feedback is incomplete and/or not fully correct, containing incorrect classifications, redundancies, inconsistencies, or problematic explanations.

...can make it more difficult for students to understand the feedback, increasing the cognitive load [41].

Some comments in feedback mention, generics, concurrency, or improvements on the provided interface, ...likely to overwhelm novices who do not yet know these concepts.

To conclude, using GPT-4 Turbo for automatically generating feedback does not seem to be advisable. The same applies to students using it without guidance or prior instruction.

- Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-Generation for Programming Exercises With GPT-4. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1 (pp. 31-37).

IMPLICATIONS FOR TEACHING SE 3?

- Commercial services and downsides
- Cost and accessibility issues?
- What to do about various forms of cheating?
- Observed violations of service agreements – who reports miscreants?
- Privacy and IP rights
- Whose work and evolving citation standards?
- AI systems and embedded biases
- Ethical awareness

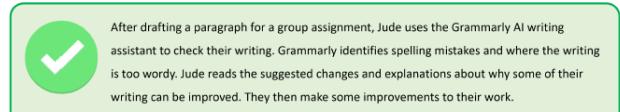
ACADEMIC INTEGRITY - THEN?

You **can** use AI when writing or preparing a presentation if you use it to help you improve small aspects of your work, such as:

- grammar and punctuation, and
- formal terminology.

You **cannot** use AI to generate ideas when writing, preparing a presentation or creating an artwork/artefact (unless otherwise specified in your assessment instructions). The ideas have to come from you and your course materials.

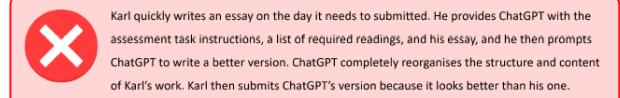
<https://canvas.aut.ac.nz/courses/7624/pages/referencing>
9/01/2023 - and evolving



After drafting a paragraph for a group assignment, Jude uses the Grammarly AI writing assistant to check their writing. Grammarly identifies spelling mistakes and where the writing is too wordy. Jude reads the suggested changes and explanations about why some of their writing can be improved. They then make some improvements to their work.

Jude has acted appropriately because they have:

- only used AI to identify small errors in their own work,
- used the AI's explanations to learn more about academic writing, and
- made their own improvements.



Karl quickly writes an essay on the day it needs to be submitted. He provides ChatGPT with the assessment task instructions, a list of required readings, and his essay, and he then prompts ChatGPT to write a better version. ChatGPT completely reorganises the structure and content of Karl's work. Karl then submits ChatGPT's version because it looks better than his own.

Karl has not acted appropriately because he has:

- submitted work that he did not do himself (he submitted ChatGPT's work), and this is a breach of AUT's academic integrity guidelines. Even if Karl had acknowledged his use of ChatGPT in the essay, this would still be inappropriate because he did not write the essay himself.

Figure 4: Appropriate and inappropriate uses of AI when writing and presenting

Reference

AAIN Generative AI Working Group. (2023). *AAIN Generative Artificial Intelligence Guidelines*, Australian Academic Integrity Network. <https://doi.org/10.26187/sbwr-kq49> ↗
<https://doi.org/10.26187/sbwr-kq49>

ACADEMIC INTEGRITY - NOW?



Submit only your own work*

Acknowledge all sources of information you use by:

- using the appropriate referencing style, and
- paraphrasing or quoting any words/ideas that are not your own.



Do not submit work done by others, such as:

- other students*
- friends or relatives
- assignment writing services
- artificial intelligence software**, like ChatGPT.

Do not submit work that you have previously submitted for assessment.

<https://canvas.aut.ac.nz/courses/7624/pages/academic-integrity> 10/05/2024

- and evolving

Figure 1: How to maintain academic integrity

* Unless the work is for a designated (group) task

** If your assessment requires the approved use of artificial intelligence, you must acknowledge wherever you do so in your work with an appropriate in-text reference (see guidelines for APA [↗](https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312), Chicago [↗](https://aut.ac.nz.libguides.com/turabian/personalcomms#s-lg-box-22370438), and Harvard [↗](https://aut.ac.nz.libguides.com/c.php?g=919289&p=6648988#s-lg-box-22370440) referencing styles).

ACADEMIC INTEGRITY – NOW?

Artificial intelligence software

Referencing the information generated by an algorithm or artificial intelligence software tool, such as ChatGPT.

Credit the author of the algorithm/AI tool with a reference list entry and an in-text citation.

Reference list format

Who	When	What	Where
Author of AI tool.	(Year released).	<i>Title of tool</i> (Version) [Description].	URL to access tool

Reference list example

OpenAI. (2023). *ChatGPT* (Mar 14 version) [Large language model]. <https://chat.openai.com/chat>

In-text citation examples

OpenAI (2023) generated the following response when...

...that indicates a limitation of the software (OpenAI, 2023).

In your writing

- Describe how you used the tool.
- Provide the prompt you used.
- Provide any portion of the relevant text that was generated in response.
- Document the exact text created because tools like ChatGPT generate unique responses in each chat session, even if given the same prompt.

More information

<https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312>

<https://apastyle.apa.org/blog/how-to-cite-chatgpt>

ChatGPT – Terms of Use

What you can do. Subject to your compliance with these Terms, you may access and use our Services.

In using our Services, you must comply with all applicable laws as well as our Sharing & Publication Policy, Usage Policies, and any other documentation, guidelines, or policies we make available to you.

What you cannot do. You may not use our Services for any illegal, harmful, or abusive activity. For example, you may not:

- Use our Services in a way that infringes, misappropriates or violates anyone's rights.
- ...
- Represent that Output was human-generated when it was not.
-
- Use Output to develop models that compete with OpenAI.

OpenAI. (2024,
January 31 2024).

Terms of Use.

Retrieved
10/05/2024 from
<https://openai.com/policies/terms-of-use>

ChatGPT – Terms of Use

Your content. You may provide input to the Services (“Input”), and receive output from the Services based on the Input (“Output”). Input and Output are collectively “Content.”

“Content.” You are responsible for Content, including ensuring that it does not violate any applicable law or these Terms. You represent and warrant that you have all rights, licenses, and permissions needed to provide Input to our Services.

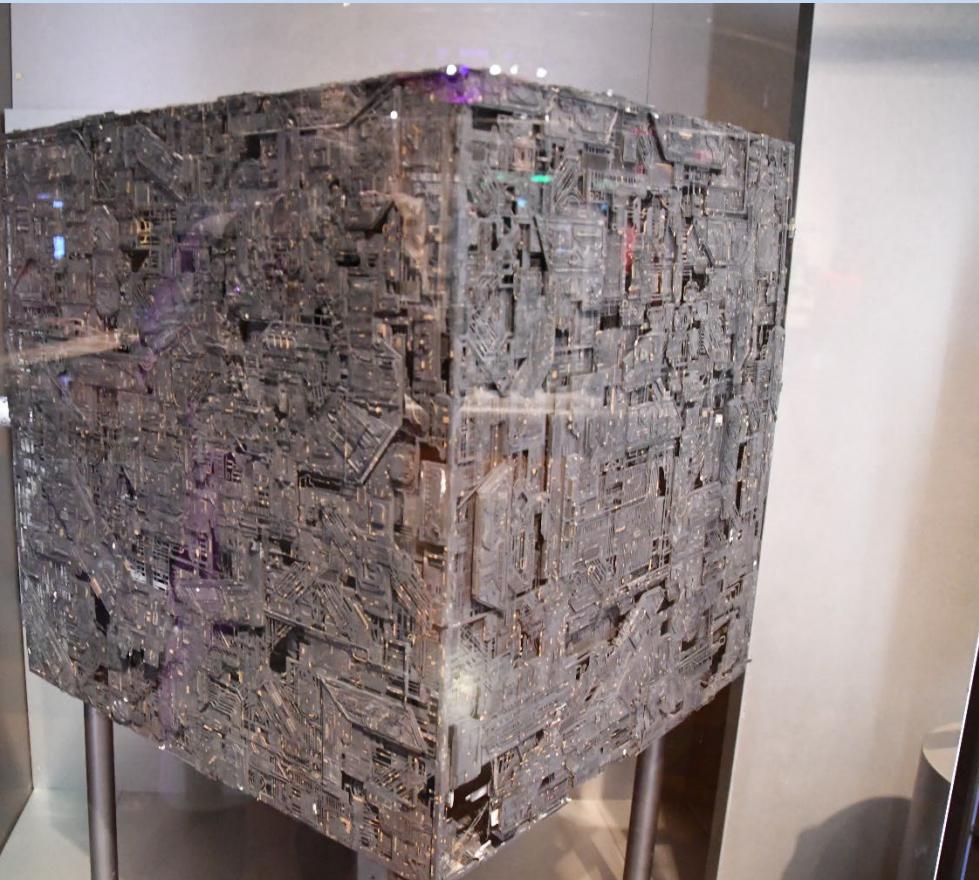
Our use of content. We may use Content to provide, maintain, develop, and improve our Services, comply with applicable law, enforce our terms and policies, and keep our Services safe.

Opt out. If you do not want us to use your Content to train our models, you can opt out by following the instructions in this Help Center article. Please note that in some cases this may limit the ability of our Services to better address your specific use case.

OpenAI. (2024, January 31 2024). *Terms of Use*. Retrieved 10/05/2024 from <https://openai.com/policies/terms-of-use>

BROADER IMPLICATIONS

- Giving data into the datacube of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
<https://commons.wikimedia.org/w/index.php?curid=58277452>



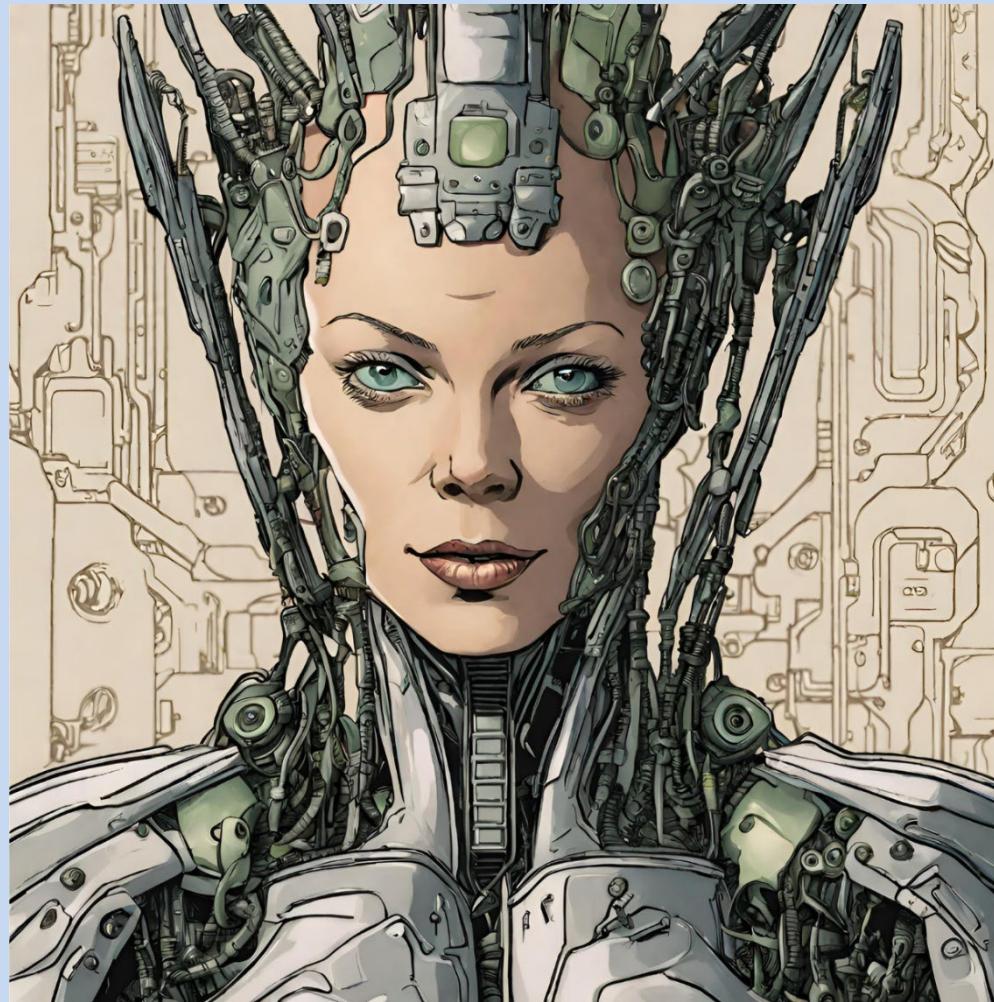
IMPLICATIONS?

- Giving data into the maw of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
https://upload.wikimedia.org/wikipedia/commons/5/52/Trekkie_-_Borg_-_Star_Trek_Convention_%2889504912575%29.jpg



IMPLICATIONS FOR TEACHING SE?

- Giving data into the clutches of the BORG and Getting it Back?



[https://www.mediawiki.org/wiki/File:
Borg_Queen_by_Canva_AI.png](https://www.mediawiki.org/wiki/File:Borg_Queen_by_Canva_AI.png)

Polski: Artystyczny wizerunek królowej Borg z uniwersum Star trek wygenerowany przez oprogramowanie Canva Magic Multimedia

English: Borg Queen from Star Trek universe artistic vision generated by Canva Magic Multimedia

4 April 2024

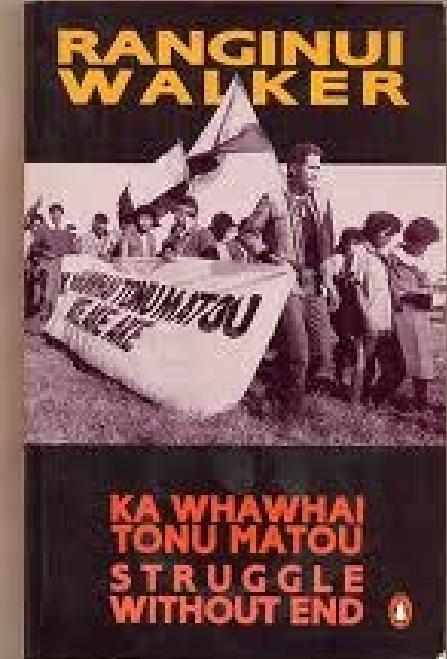
Own work

[Canva Magic Multimedia](#)

CONCLUSION

- Discussed Inroads column on LLM's
- Covered waves of colonization and technology
- Covered surveillance capitalism
- Our lives and data!
- Neo-digi-colonization
- The need for openness
- The doctrine of discovery
- Making property and land claims
- Links with AI and intellectual property rights
- Data Privacy
- Regulatory races
- Implications for Competencies and SE Education

Generative AI a Critique: Questions?



Associate Professor Tony Clear
Department of Computer Science and Software
Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz





IT Professionals

NEW ZEALAND



IT Professionals NZ

Code of Ethics

The mandatory code outlining ethical and professional requirements for IT Professionals in New Zealand.

May 2017

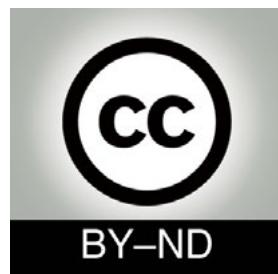
About ITP

IT Professionals New Zealand (ITP) is the professional body of the IT sector in New Zealand.

ITP has a proud history spanning over 50 years and has been a part of the computing and IT sector in New Zealand since formation in 1960.

ITP Vision

ITP is the authoritative voice of the IT profession that leads professional development and good practice in IT.



This document is made available under the following license:

Creative Commons Attribution-No Derivative Works 3.0 New Zealand

You are free to **share** (copy, distribute and transmit the work) under the following conditions:

- ▶ **Attribution:** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- ▶ **No Derivative Works:** You may not alter, transform, or build upon this work.

Waiver: Note that any of these conditioned may be waived if you get permission from IITP.

Full details at <http://creativecommons.org/licenses/by-nd/3.0/nz/>

The ITP Code of Ethics

This ITP Code of Ethics document is split into three sections:

Section 1 - Code of Ethics _____ **Page 4**

This is the ITP Code of Ethics in its entirety. The Code is made up of 8 Tenets, each related to the others but also covering an important area related to ethics and conduct.

Section 2 - Guidelines and Interpretation _____ **Page 6**

This material does not form a part of the Code, but rather is used to assist in the interpretation and implementation of the Code of Ethics.

Section 3 - Breaches and Disciplinary Process _____ **Page 14**

This section is taken from Schedule Four of the ITP Bylaws, and outlines the procedures by which alleged breaches of the Code will be dealt with, including the provision for a hearing of the *Discipline and Professional Conduct Board*.

This ITP Code of Ethics was formally adopted by resolution of a Special General Meeting held for that purpose in Wellington, New Zealand, on Friday 28th May 2010. From 2010 to 2015 this was known as the “Code of Professional Conduct”, however was renamed back to Code of Ethics via a Special General Meeting on 30 June 2015.

The Code replaced the previous ITP Code of Ethics.

Section 1:

The ITP New Zealand Code of Ethics

ITP Code of Ethics

IT Professionals will practice with:

- 1. Good faith** - Members shall treat people with dignity, good faith and equality; without discrimination; and have consideration for the values and cultural sensitivities of all groups within the community affected by their work;
- 2. Integrity** - Members shall act in the execution of their profession with integrity, dignity and honour to merit the trust of the community and the profession, and apply honesty, skill, judgement and initiative to contribute positively to the well-being of society;
- 3. Community-focus** - Members' responsibility for the welfare and rights of the community shall come before their responsibility to their profession, sectional or private interests or to other members;
- 4. Skills** - Members shall apply their skills and knowledge in the interests of their clients or employers for whom they will act without compromising any other of these Tenets;
- 5. Continuous Development** - Members shall develop their knowledge, skills and expertise continuously through their careers, contribute to the collective wisdom of the profession, and actively encourage their associates to do likewise;
- 6. Informed Consent** - Members shall take reasonable steps to inform themselves, their clients or employers of the economic, social, environmental or legal consequences which may arise from their actions;
- 7. Managed Conflicts of Interest** - Members shall inform their clients or employers of any interest which may be, or may be perceived as being, in conflict with the interests of their clients or employers, or which may affect the quality of service or impartial judgement;
- 8. Competence** - Members shall follow recognised professional practice, and provide services and advice carefully and diligently only within their areas of competence.

These Tenets comprise the essence of the Code of Ethics. Breaches shall be dealt with as provided in the ITP Bylaws. They should be read and applied in conjunction with Supplements to the Code of Ethics.

The Supplements will be maintained and updated by the Institute so as to be timely and relevant. Members of the Institute have a duty to be cognisant of the Supplements' content especially those which are applicable to the areas of professional expertise and activity in which they are involved.

Section 2:

*Supplement to the Code:
Guidance and Interpretation*

Supplement to the ITP Code of Ethics:

Guidance and Interpretation

This supplement does not form a part of the Code, however should be read in conjunction with the code to provide context and interpretation.

This Supplement includes the following sections:

1: Context of the Tenets of the Code	8
2: General Matters	8
3: The Community	9
4: Qualifications and Competence	10
5: Clients and Employers	11
6: Ethical Dilemmas	12
7: Maintenance of the Code	13

1: Context of the Tenets of the Code

- 1.1** The Code of Ethics is issued under the provisions of clause 9 of the Constitution of the Institute of IT Professionals NZ Incorporated (trading as IT Professionals New Zealand) and is binding on all members in all grades of the Institute, including the Institute itself.
- 1.2** This Code of Ethics is to be read and interpreted in conjunction with the Institute's Constitution, Bylaws and this Guidance and Interpretation.
- 1.3** The respect which society accords the technology professions is earned and maintained by its members demonstrating a strong and consistent commitment to ethical values. These commitments are additional to the obligations, which every member of society is required to observe, such as obeying the law of the legislative authority within which they are working as well as those of their own country as applicable, and reflect the additional responsibility expected of all professionals.
- 1.4** Therefore the Institute must maintain an appropriate Code of Ethics, to make it available for the information of the public and to enforce it impartially. This Code must be responsive to the changing expectations of both the public and the profession and the global standards to which the Institute of IT Professionals subscribes.
- 1.5** This code is based upon the principles of:
- Interests of the community
 - Respect for the individual
 - Interests of the client
 - Professional integrity

and supported by the values of:

- Competence
- Truth
- Social justice and
- Ethical behaviour

- 1.6** Eight ethical Tenets form the basis of the Code to guide members in achieving the high ideals of professional life. To assist in the interpretation of the Code, guidelines are set out below to support these Tenets.

2: General Matters

- 2.1** The purpose of this supplement is to assist members in applying the Tenets of the Code of Ethics which have been written in broad terms. Whilst not specifically part of the Code of Ethics, the supplement should be read in conjunction with the Code and members will be expected to have considered the content of the Code, this and any other supplements in any matter of professional conduct.
- 2.2** The supplements are not prescriptive but form the basis, in conjunction with the Constitution and Bylaws, of the Institute's concept of Professional Conduct and Practice.

3: The Community

3.1 “Community”, in the context of the Code, refers to all groups in society including members’ own workplaces. The first three Tenets of the Code refer to the Community and may be considered to include:

- Acting and working in a way such that the health, safety and well-being of employees and colleagues are not endangered;
- Ensuring that work undertaken meets community expectations by adopting the norms of recognised professional practice; or communicating any attendant risks or limitations, and their effect, in any work undertaken which does not accord with convention;
- Being vigilant in ‘duty of care’ toward members of the community;
- Communicating the results of work undertaken in a clear and unambiguous way;
- Raising real or perceived conflicts of interest, or issues which may not be in the community interest, at an early stage of involvement;
- Commitment to the principles of sustainable development of the planet’s resources and seeking to minimise adverse environmental impacts of their work or applications of technology for both present and future generations;
- Not being involved in any activity which is known to be fraudulent, dishonest or not in the interests of the community (as described); and
- Not accepting reward or compensation from any more than one party, without the clear understanding and acceptance of all parties.

3.2 In summary, these Tenets of the Code require members to be mindful of more than their technical and professional responsibilities and their immediate employer or client.

3.3 Many of the requirements demand no more than sound management practices such as Occupational Health and Safety Plans and care of colleagues and staff. The Code goes further in its obligation for members to be aware of the consequences of all of their actions in the practice of their professions. The essence of “professionalism” is in remaining aware of these obligations and in making sound and informed decisions when faced with any conflict of responsibilities which may, and likely will, arise.

4: Qualifications and Competence

- 4.1** Qualifications denote the foundation of knowledge that a member has achieved through formal education, experience, post graduate learning or a combination from all of these sources.
- 4.2** Competence is demonstrated by application of knowledge and skills to provide service, advice or opinion to clients or employers.
- 4.3** The Tenets of the Code which relate to these themes may be considered to include an expectation of members to:
- not misrepresent the qualifications and competences of themselves or those in their employ or under their supervision;
 - not undertake any assignment which is outside their competence; and if requested to do so, to bring to their client's or employer's attention to the need to access further expertise;
 - seek expert assistance on encountering any professional issue or problem outside the range of experience or competence;
 - not expect their employees to undertake work for which the employees have little or no demonstrated competence, other than in a supervised capacity; and
 - keep themselves competent and informed by continuous professional development.
- 4.4** By carefully limiting the professional work undertaken within the limits of their qualifications and competence, members protect the interests of the community, clients, employers and themselves.
- 4.5** A mistake or error of judgement that a member might make within the limits of competence and qualification, even though it may be judged as negligence, will not be considered as unethical behaviour.

5: Clients and Employers

- 5.1** Members have a duty to provide loyal and competent service to their clients and employers. “Loyalty” implies looking out for their interests, giving fearless advice, providing strict confidentiality.
- 5.2** A client is the recipient of professional service, advice or opinion. Members are encouraged to be always mindful of the question “who is the client?” In some circumstances it may also be the employer. An employee of a private company has dual duties to employer and client. An academic employed by a University has students as clients.
- 5.3** When a member is serving both an employer and a client, there is potential for competing and conflicting interests.
- 5.4** The Tenets of the Code which relate to these themes include an expectation of members to:
- not disclose or use any confidential information gained in the course of their employment without permission, unless disclosure is a legal requirement or withholding the information would be to the detriment of the community;
 - be truthful, factual and free from exaggeration in advertising or promoting their services to potential clients, either by advertisement or direct approach;
 - keep clients and employers informed of any known or potential conflict of interest;
 - not accept payment for particular service or information from more than one source without disclosure to all parties, unless it is apparent that the service or information is intended for multiple use;
 - not undertake assignments under conditions which may compromise their ability to satisfy client or employer needs in a professional manner;
 - advise clients and employers of the level of risk, or any possible adverse consequences, of any instruction given which is outside the norms of conventional practice;
 - not attempt to supplant others who are already engaged by a client; nor to damage in any way the reputation of competitors by way of comparison or denigration;
 - when acting as an expert witness to a court or tribunal, be mindful of their primary obligation as an expert witness to the court or the hearing, and not to the engaging party.

6: Ethical Dilemmas

- 6.1** An ethical dilemma occurs when one Tenet of the Code cannot be honoured without apparent breach of another. It is recommended that any member finding themselves in such a situation should consult with the affected parties and attempt an ethical resolution or, failing that, refer the matter to the Discipline and Professional Conduct Board for advice.
- 6.2** In any professional career, ethical challenges or dilemmas will arise. Teachers may be caught between the demands of their employers and the needs of their students; employees may be caught between the performance targets of their employers and the expectations of their clients; consultants may be caught between the expectations of their clients and the interests of the community; public service professionals may be caught between the directives of superiors and the well being of staff.
- 6.3** Resolution of ethical dilemmas is ultimately a matter of personal responsibility, taking into account the principles which lie behind the Tenets of the Code. Members of the Institute are encouraged to share their ethical dilemmas with a trusted colleague, or refer such matters to the Institute's National Board or CEO if they wish. Either of those may further refer the matter to the PCB for consideration
- 6.4** Members seeking guidance in higher ethical issues are recommended to consider documents such as the United Nations Declaration of Human Rights, the Treaty of Waitangi and the New Zealand Bill of Rights. It is recognised that members may also feel subject to religious obligations which may impose ethical and moral dilemmas. In such cases, members are advised to consult with their religious guides and may refer such matters to the Institute's National Board or CEO if they wish. Either of those may further refer the matter to the Discipline and Professional Conduct Board for consideration.
- 6.5** Another aspect of ethical dilemma is the occurrence of a conflict of interest. A conflict of interest may be real, potential or perceived. It arises when relationships exist among such entities as clients, employers, employees, vendors, colleagues or any combination that can, may or actually influence another entity to a possibly detrimental extent. These extents include diminished commitment, pecuniary or other gain, undue influence
- 6.6** The issue can also be difficult because a perceived conflict of interest may be as consequential as a real occurrence. This happens when those who perceive a conflict act in ways that are influenced by the perception. As such, perceived conflicts of interest may have to be managed or otherwise dealt with as if real. The Code of Practice details some strategies relevant to conflict of interest resolution or management.

7: Maintenance of the Code

7.1 The role of IT Professionals NZ is:

- to provide rigorous and fair processes for dealing with complaints and charges made against members; and
- to provide counsel and support to members confronting ethical dilemmas or any other difficulties in relation to their own or others' ethical behaviour.

7.2 ITP has a Discipline and Professional Conduct Board which "is responsible for the implementation and oversight of professional conduct standards as they apply to all membership classes and to maintain alignment of such standards with kindred bodies and international guidelines."¹

7.3 Where appropriate, the Discipline and Professional Conduct Board will, as circumstances warrant, propose additions, changes or deletions within both this supplement and Supplement 2, the Code of Practice. Members are encouraged to submit material for Board consideration where they feel that guidance and interpretation and/or the Code of Practice might be improved.

7.4 The *Code of Practice* endeavours to provide guidance and examples within specific areas of Information Technology practice. As such, it is likely to change far more frequently than the other documents mentioned. Members are encouraged to refer to the Code regularly (either on the Institute's web site or in downloadable form or as a printed copy by request from the Institute).

7.5 The two Codes, particularly, form the basis upon which your conduct as a member of the Institute might be measured by the public, employers, clients and your colleagues. It is also the basis against which any disciplinary proceedings of the Institute will seek comparison.

7.6 Schedule 4 of the ITP Bylaws, dealing with breaches of the Code of Ethics, provides additional information.

¹ Professional Conduct Board – Mandate, Matthews, P., et al., 2009. *Information Technology Certified Professional, IITP ITCP Certification Model*. New Zealand Computer Society Inc.

Section 3:

ITP Bylaws Schedule 4: **ITP Disciplinary Process**

ITP Bylaws Schedule Four:

Breach of the Code of Ethics

1 Preamble

"The processes undertaken in the consideration of complaints or the resolution of disputes will be in strict accord with Rules determined by the Institute of IT Professionals NZ's National Board. The rules of natural justice will be paramount in all processes. The rules of natural justice include the right for a respondent to know the details of a complaint and the supporting evidence; the right to provide evidence in defence; and the right for an unbiased determination made by those who hear all the evidence."

- Institute of IT Professionals NZ, Code of Ethics

2 Procedure

The procedure for dealing with an alleged breach of the Code of Ethics by a member shall be as follows:

Lodging the Complaint

- 2.1 A complaint, being an allegation of a breach of the Code of Ethics, may be made by any individual or entity concerning any person who is a Institute member or was a member at the time relevant to the complaint.
- 2.2 Placing such a complaint shall signify acceptance that the matter shall be determined in the manner prescribed in this Schedule of the Bylaws, including the clauses dealing with publication of the findings of this consideration of the complaint. The complainant also specifically waives any right to take civil action against the Institute should he or she disagree with the process or findings of the Institute.
- 2.3 Nothing in this Schedule shall preclude the complainant or others taking civil or criminal proceedings against the member in question, and this process should not be seen as an alternative to doing so if appropriate.
- 2.4 The complaint must be made on the prescribed form setting out particulars of the alleged breach and attaching any documentation or other relevant details. The complaint shall be addressed to the President, or if it is in relation to the conduct of the President, to the Deputy President. This person shall be the "Receiver" of the complaint.

Receiving the Complaint

- 2.5 The President (or Deputy President in the case of a complaint against the President) may delegate to the Chief Executive or any member of the National Board to act on his or her behalf as Receiver to discharge the remainder of his or her responsibilities in relation to the matter if he or she sees fit. For example, if there is a Conflict of Interest, or if he or she is unable to fulfill the Receiver's requirements within the time required.

- 2.6 The Receiver must acknowledge receipt of the complaint in writing, and notify the member of the complaint in writing (attaching a copy of the complaint and all documentation provided with the complaint), including that the complaint has been referred to the Discipline and Professional Conduct Board who will hold a hearing into the complaint.

Referral to Discipline and Professional Conduct Board

- 2.7 The Receiver must refer the complaint to the Chair of the Discipline and Professional Conduct Board within 7 days of receipt.
- 2.8 If the Chair of the Discipline and Professional Conduct Board is not available to complete the requirements in this Schedule in a timely manner the IITP President may authorise another member of the Board or any member of the IITP National Board to Chair proceedings and discharge the duties of the Chair as outlined in this Schedule.

Frivolous or Vexatious Complaints

- 2.9 At any stage of proceedings if the Chair of the Discipline and Professional Conduct Board believes that the complaint is frivolous or vexatious they may immediately call for a vote of the Discipline and Professional Conduct Board as to whether it is indeed frivolous or vexatious. This vote may be conducted in any way prescribed by the Chair, including electronic via email.
- 2.10 If the Discipline and Professional Conduct Board votes by a three quarters majority that the complaint is frivolous or vexatious it shall be immediately dismissed.

Hearing of the Discipline and Professional Conduct Board

- 2.11 The Chair of the Discipline and Professional Conduct Board shall forward a copy of the complaint and all supporting information to the members of the Discipline and Professional Conduct Board and arrange a hearing of the Board, which shall be between 3 and 6 weeks from the date the complaint is referred. The Chair of this Board may choose a hearing time outside this timeframe if it is necessary to ensure a fair hearing, however the hearing must be conducted in a timely fashion.
- 2.12 The quorum for the hearing shall be two thirds of the membership of the Discipline and Professional Conduct Board. The hearing may be held by Teleconference or in person.
- 2.13 Hearings are confidential and the evidence provided during a hearing is not published, other than that to support a published verdict (see below).
- 2.14 The complainant will have the option of addressing the hearing of the Discipline and Professional Conduct Board in person (where the hearing is held in person) or by Teleconference for the purpose of providing further verbal evidence. The complainant may equally opt not to do so. If they are to address the hearing they may only be present for the portion of the hearing set aside for that purpose.

- 2.15 The Discipline and Professional Conduct Board may call on any witnesses or other persons to provide verbal evidence to the hearing, but has no authority to compel. Any witness may opt to provide evidence by way of a signed statement. Any witness providing verbal evidence may only be present for the portion of the hearing set aside for that purpose.
- 2.16 In keeping with the principles of natural justice, the member alleged to have committed the breach will have the option of being present during all verbal evidence but may not address the complainant, witnesses, or the Board at that time.
- 2.17 The member alleged to have committed the breach will have the option of addressing the hearing of the Discipline and Professional Conduct Board in person (where the hearing is held in person) or by Teleconference for the purpose of providing verbal evidence. The member may also choose to provide evidence in writing in advance by signed statement. The complainant may equally opt not to do either and this shall not be construed as evidence of acceptance of the allegation of breach.
- 2.18 The member alleged to have committed the breach may choose to be represented.

Outcome of Hearing

- 2.19 The Discipline and Professional Conduct Board shall deliberate in private until such time that:
 - 2.19.1 The Board rules by a three quarters majority that a significant breach has occurred. In this case a breach shall be found proved and the finding, along with a detailed justification for the finding, shall be referred to the National Board;
 - 2.19.2 The Board rules by a three quarters majority that a technical breach occurred, but finds that the breach is trivial or trifling and the matter is dismissed;
 - 2.19.3 The Board rules by a three quarters majority that no breach has occurred and the matter is dismissed;
 - 2.19.4 The Board cannot agree by a three quarters majority that a breach has occurred, and the Chair determines that further deliberation would be fruitless. The matter is therefore found to be unproved and is dismissed.
- 2.20 The complainant and the member shall be informed of the outcome without delay.

Penalty Imposed

- 2.21 In the case of a breach, the Discipline and Professional Conduct Board shall agree a penalty consistent with both the gravity of the breach and previous penalties for similar breaches from the options provided in the Institute's Constitution. A report outlining the findings and penalty shall be forwarded to the National Board without delay.
- 2.22 In the case of a breach, the Discipline and Professional Conduct Board shall solely decide the penalty, if any, from the remedies provided in the Constitution. The penalty is not subject to appeal.

- 2.23 The Discipline and Professional Conduct Board shall determine whether the existence, details, finding, complainant identity, and/or penalty shall be released publicly, and no other member or participant shall release in part or full any determination or detail related to the hearing or complaint other than that which this Board chooses to release.
- 2.24 If the hearing uncovers significant criminal or civil wrongdoing the National Board may opt to refer the matter to the NZ Police or other law enforcement agency by way of a formal complaint, or commence other legal proceedings as it sees fit.

Appealing the Determination of the Hearing

- 2.25 An Appeal against any finding may be made by either the Complainant or Member concerned for any of the following reasons:
 - 2.25.1 If further evidence that was not previously available becomes available that is materially different to any considered during the hearing and which, on balance, could change the outcome;
 - 2.25.2 If the procedure outlined in this schedule was not adhered to, and the breach is more than trivial and may have materially changed the outcome of the hearing;
 - 2.25.3 A member of the Discipline and Professional Conduct Board had a significant undeclared Conflict of Interest which may have impacted upon his or her impartiality.
- 2.26 Simply not agreeing with the determination or penalty shall not be grounds for appeal.
- 2.27 Any appeal should be made to the original Receiver within 21 days of the determination of the Discipline and Professional Conduct Board. The Receiver will initially determine whether, on the balance of probability, the appeal meets the criteria outlined in 2.25 and if so, shall formally forward the appeal to the National Board.
- 2.28 The National Board shall consider the matters raised in the appeal and determine whether the appeal shall stand. If the appeal stands the Board may modify the determination or penalty as it sees fit.

ACM Ethics

The Official Site of the Association for Computing Machinery's Committee on Professional Ethics



ACM Code of Ethics and Professional Conduct

Adopted by ACM Council 6/22/18.

Preamble

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.

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. Additionally, the Code serves as a basis for remediation when violations occur. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle.

Section 1 outlines fundamental ethical principles that form the basis for the remainder of the Code. Section 2 addresses additional, more specific considerations of professional responsibility. Section 3 guides individuals who have a leadership role, whether in the workplace or in a volunteer professional capacity. Commitment to ethical conduct is required of every ACM member, and principles involving compliance with the Code are given in Section 4.

The Code as a whole is concerned with how fundamental ethical principles apply to a computing professional's conduct. The Code is not an algorithm for solving ethical problems; rather it serves as a basis for ethical decision-making. When thinking through a particular issue, a computing professional may find that multiple principles should be taken into account, and that different principles will have different relevance to the issue. Questions related to these kinds of issues can best be answered by thoughtful consideration of the fundamental ethical principles, understanding that the public good is the paramount consideration. The entire computing profession benefits when the ethical decision-making process is accountable to and transparent to all stakeholders. Open discussions about ethical issues promote this accountability and transparency.

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.

This principle, which concerns the quality of life of all people, affirms an obligation of computing professionals, both individually and collectively, to use their skills for the benefit of society, its members, and the environment surrounding them. This obligation includes promoting fundamental human rights and protecting each individual's right to autonomy. An essential aim of computing professionals is to minimize negative consequences of computing, including threats to health, safety, personal security, and privacy. When the interests of multiple groups conflict, the needs of those less advantaged should be given increased attention and priority.

Computing professionals should consider whether the results of their efforts will respect diversity, will be used in socially responsible ways, will meet social needs, and will be broadly accessible. They are encouraged to actively contribute to society by engaging in pro bono or volunteer work that benefits the public good.

In addition to a safe social environment, human well-being requires a safe natural environment. Therefore, computing professionals should promote environmental sustainability both locally and globally.

1.2 Avoid harm.

In this document, "harm" means negative consequences, especially when those consequences are significant and unjust. Examples of harm include unjustified physical or mental injury, unjustified destruction or disclosure of information, and unjustified damage to property, reputation, and the environment. This list is not exhaustive.

Well-intended actions, including those that accomplish assigned duties, may lead to harm. When that harm is unintended, those responsible are obliged to undo or mitigate the harm as much as possible. Avoiding harm begins with careful consideration of potential impacts on all those affected by decisions. When harm is an intentional part of the system, those responsible are obligated to ensure that the harm is ethically justified. In either case, ensure that all harm is minimized.

To minimize the possibility of indirectly or unintentionally harming others, computing professionals should follow generally accepted best practices unless there is a compelling ethical reason to do otherwise. Additionally, the consequences of data aggregation and emergent properties of systems should be carefully analyzed. Those involved with pervasive or infrastructure systems should also consider Principle 3.7.

A computing professional has an additional obligation to report any signs of system risks that might result in harm. If leaders do not act to curtail or mitigate such risks, it may be necessary to "blow the whistle" to reduce potential harm. However, capricious or misguided reporting of risks can itself be harmful. Before reporting risks, a computing professional should carefully assess relevant aspects of the situation.

1.3 Be honest and trustworthy.

Honesty is an essential component of trustworthiness. A computing professional should be transparent and

provide full disclosure of all pertinent system capabilities, limitations, and potential problems to the appropriate parties. Making deliberately false or misleading claims, fabricating or falsifying data, offering or accepting bribes, and other dishonest conduct are violations of the Code.

Computing professionals should be honest about their qualifications, and about any limitations in their competence to complete a task. Computing professionals should be forthright about any circumstances that might lead to either real or perceived conflicts of interest or otherwise tend to undermine the independence of their judgment. Furthermore, commitments should be honored.

Computing professionals should not misrepresent an organization's policies or procedures, and should not speak on behalf of an organization unless authorized to do so.

1.4 Be fair and take action not to discriminate.

The values of equality, tolerance, respect for others, and justice govern this principle. Fairness requires that even careful decision processes provide some avenue for redress of grievances.

Computing professionals should foster fair participation of all people, including those of underrepresented groups. Prejudicial discrimination on the basis of age, color, disability, ethnicity, family status, gender identity, labor union membership, military status, nationality, race, religion or belief, sex, sexual orientation, or any other inappropriate factor is an explicit violation of the Code. Harassment, including sexual harassment, bullying, and other abuses of power and authority, is a form of discrimination that, amongst other harms, limits fair access to the virtual and physical spaces where such harassment takes place.

The use of information and technology may cause new, or enhance existing, inequities. Technologies and practices should be as inclusive and accessible as possible and computing professionals should take action to avoid creating systems or technologies that disenfranchise or oppress people. Failure to design for inclusiveness and accessibility may constitute unfair discrimination.

1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.

Developing new ideas, inventions, creative works, and computing artifacts creates value for society, and those who expend this effort should expect to gain value from their work. Computing professionals should therefore credit the creators of ideas, inventions, work, and artifacts, and respect copyrights, patents, trade secrets, license agreements, and other methods of protecting authors' works.

Both custom and the law recognize that some exceptions to a creator's control of a work are necessary for the public good. Computing professionals should not unduly oppose reasonable uses of their intellectual works. Efforts to help others by contributing time and energy to projects that help society illustrate a positive aspect of this principle. Such efforts include free and open source software and work put into the public domain. Computing professionals should not claim private ownership of work that they or others have shared as public resources.

1.6 Respect privacy.

The responsibility of respecting privacy applies to computing professionals in a particularly profound way. Technology enables the collection, monitoring, and exchange of personal information quickly, inexpensively, and often without the knowledge of the people affected. Therefore, a computing professional should become conversant in the various definitions and forms of privacy and should understand the rights and responsibilities associated with the collection and use of personal information.

Computing professionals should only use personal information for legitimate ends and without violating the rights of individuals and groups. This requires taking precautions to prevent re-identification of anonymized data or unauthorized data collection, ensuring the accuracy of data, understanding the provenance of the data, and protecting it from unauthorized access and accidental disclosure. Computing professionals should establish transparent policies and procedures that allow individuals to understand what data is being collected and how it is being used, to give informed consent for automatic data collection, and to review, obtain, correct inaccuracies in, and delete their personal data.

Only the minimum amount of personal information necessary should be collected in a system. The retention and disposal periods for that information should be clearly defined, enforced, and communicated to data subjects. Personal information gathered for a specific purpose should not be used for other purposes without the person's consent. Merged data collections can compromise privacy features present in the original collections. Therefore, computing professionals should take special care for privacy when merging data collections.

1.7 Honor confidentiality.

Computing professionals are often entrusted with confidential information such as trade secrets, client data, nonpublic business strategies, financial information, research data, pre-publication scholarly articles, and patent applications. Computing professionals should protect confidentiality except in cases where it is evidence of the violation of law, of organizational regulations, or of the Code. In these cases, the nature or contents of that information should not be disclosed except to appropriate authorities. A computing professional should consider thoughtfully whether such disclosures are consistent with the Code.

2. PROFESSIONAL RESPONSIBILITIES.

A computing professional should...

2.1 Strive to achieve high quality in both the processes and products of professional work.

Computing professionals should insist on and support high quality work from themselves and from colleagues. The dignity of employers, employees, colleagues, clients, users, and anyone else affected either directly or indirectly by the work should be respected throughout the process. Computing professionals should respect the right of those involved to transparent communication about the project. Professionals should be cognizant of any serious negative consequences affecting any stakeholder that may result from poor quality work and should resist inducements to neglect this responsibility.

2.2 Maintain high standards of professional competence, conduct, and ethical practice.

High quality computing depends on individuals and teams who take personal and group responsibility for

acquiring and maintaining professional competence. Professional competence starts with technical knowledge and with awareness of the social context in which their work may be deployed. Professional competence also requires skill in communication, in reflective analysis, and in recognizing and navigating ethical challenges. Upgrading skills should be an ongoing process and might include independent study, attending conferences or seminars, and other informal or formal education. Professional organizations and employers should encourage and facilitate these activities.

2.3 Know and respect existing rules pertaining to professional work.

“Rules” here include local, regional, national, and international laws and regulations, as well as any policies and procedures of the organizations to which the professional belongs. Computing professionals must abide by these rules unless there is a compelling ethical justification to do otherwise. Rules that are judged unethical should be challenged. A rule may be unethical when it has an inadequate moral basis or causes recognizable harm. A computing professional should consider challenging the rule through existing channels before violating the rule. A computing professional who decides to violate a rule because it is unethical, or for any other reason, must consider potential consequences and accept responsibility for that action.

2.4 Accept and provide appropriate professional review.

High quality professional work in computing depends on professional review at all stages. Whenever appropriate, computing professionals should seek and utilize peer and stakeholder review. Computing professionals should also provide constructive, critical reviews of others’ work.

2.5 Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.

Computing professionals are in a position of trust, and therefore have a special responsibility to provide objective, credible evaluations and testimony to employers, employees, clients, users, and the public. Computing professionals should strive to be perceptive, thorough, and objective when evaluating, recommending, and presenting system descriptions and alternatives. Extraordinary care should be taken to identify and mitigate potential risks in machine learning systems. A system for which future risks cannot be reliably predicted requires frequent reassessment of risk as the system evolves in use, or it should not be deployed. Any issues that might result in major risk must be reported to appropriate parties.

2.6 Perform work only in areas of competence.

A computing professional is responsible for evaluating potential work assignments. This includes evaluating the work’s feasibility and advisability, and making a judgment about whether the work assignment is within the professional’s areas of competence. If at any time before or during the work assignment the professional identifies a lack of a necessary expertise, they must disclose this to the employer or client. The client or employer may decide to pursue the assignment with the professional after additional time to acquire the necessary competencies, to pursue the assignment with someone else who has the required expertise, or to forgo the assignment. A computing professional’s ethical judgment should be the final guide in deciding whether to work on the assignment.

2.7 Foster public awareness and understanding of computing, related technologies, and their consequences.

As appropriate to the context and one's abilities, computing professionals should share technical knowledge with the public, foster awareness of computing, and encourage understanding of computing. These communications with the public should be clear, respectful, and welcoming. Important issues include the impacts of computer systems, their limitations, their vulnerabilities, and the opportunities that they present. Additionally, a computing professional should respectfully address inaccurate or misleading information related to computing.

2.8 Access computing and communication resources only when authorized or when compelled by the public good.

Individuals and organizations have the right to restrict access to their systems and data so long as the restrictions are consistent with other principles in the Code. Consequently, computing professionals should not access another's computer system, software, or data without a reasonable belief that such an action would be authorized or a compelling belief that it is consistent with the public good. A system being publicly accessible is not sufficient grounds on its own to imply authorization. Under exceptional circumstances a computing professional may use unauthorized access to disrupt or inhibit the functioning of malicious systems; extraordinary precautions must be taken in these instances to avoid harm to others.

2.9 Design and implement systems that are robustly and usably secure.

Breaches of computer security cause harm. Robust security should be a primary consideration when designing and implementing systems. Computing professionals should perform due diligence to ensure the system functions as intended, and take appropriate action to secure resources against accidental and intentional misuse, modification, and denial of service. As threats can arise and change after a system is deployed, computing professionals should integrate mitigation techniques and policies, such as monitoring, patching, and vulnerability reporting. Computing professionals should also take steps to ensure parties affected by data breaches are notified in a timely and clear manner, providing appropriate guidance and remediation.

To ensure the system achieves its intended purpose, security features should be designed to be as intuitive and easy to use as possible. Computing professionals should discourage security precautions that are too confusing, are situationally inappropriate, or otherwise inhibit legitimate use.

In cases where misuse or harm are predictable or unavoidable, the best option may be to not implement the system.

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.

People—including users, customers, colleagues, and others affected directly or indirectly—should always be the central concern in computing. The public good should always be an explicit consideration when evaluating tasks associated with research, requirements analysis, design, implementation, testing, validation, deployment, maintenance, retirement, and disposal. Computing professionals should keep this focus no matter which methodologies or techniques they use in their practice.

3.2 Articulate, encourage acceptance of, and evaluate fulfillment of social responsibilities by members of the organization or group.

Technical organizations and groups affect broader society, and their leaders should accept the associated responsibilities. Organizations—through procedures and attitudes oriented toward quality, transparency, and the welfare of society—reduce harm to the public and raise awareness of the influence of technology in our lives. Therefore, leaders should encourage full participation of computing professionals in meeting relevant social responsibilities and discourage tendencies to do otherwise.

3.3 Manage personnel and resources to enhance the quality of working life.

Leaders should ensure that they enhance, not degrade, the quality of working life. Leaders should consider the personal and professional development, accessibility requirements, physical safety, psychological well-being, and human dignity of all workers. Appropriate human-computer ergonomic standards should be used in the workplace.

3.4 Articulate, apply, and support policies and processes that reflect the principles of the Code.

Leaders should pursue clearly defined organizational policies that are consistent with the Code and effectively communicate them to relevant stakeholders. In addition, leaders should encourage and reward compliance with those policies, and take appropriate action when policies are violated. Designing or implementing processes that deliberately or negligently violate, or tend to enable the violation of, the Code's principles is ethically unacceptable.

3.5 Create opportunities for members of the organization or group to grow as professionals.

Educational opportunities are essential for all organization and group members. Leaders should ensure that opportunities are available to computing professionals to help them improve their knowledge and skills in professionalism, in the practice of ethics, and in their technical specialties. These opportunities should include experiences that familiarize computing professionals with the consequences and limitations of particular types of systems. Computing professionals should be fully aware of the dangers of oversimplified approaches, the improbability of anticipating every possible operating condition, the inevitability of software errors, the interactions of systems and their contexts, and other issues related to the complexity of their profession—and thus be confident in taking on responsibilities for the work that they do.

3.6 Use care when modifying or retiring systems.

Interface changes, the removal of features, and even software updates have an impact on the productivity of users

and the quality of their work. Leaders should take care when changing or discontinuing support for system features on which people still depend. Leaders should thoroughly investigate viable alternatives to removing support for a legacy system. If these alternatives are unacceptably risky or impractical, the developer should assist stakeholders' graceful migration from the system to an alternative. Users should be notified of the risks of continued use of the unsupported system long before support ends. Computing professionals should assist system users in monitoring the operational viability of their computing systems, and help them understand that timely replacement of inappropriate or outdated features or entire systems may be needed.

3.7 Recognize and take special care of systems that become integrated into the infrastructure of society.

Even the simplest computer systems have the potential to impact all aspects of society when integrated with everyday activities such as commerce, travel, government, healthcare, and education. When organizations and groups develop systems that become an important part of the infrastructure of society, their leaders have an added responsibility to be good stewards of these systems. Part of that stewardship requires establishing policies for fair system access, including for those who may have been excluded. That stewardship also requires that computing professionals monitor the level of integration of their systems into the infrastructure of society. As the level of adoption changes, the ethical responsibilities of the organization or group are likely to change as well. Continual monitoring of how society is using a system will allow the organization or group to remain consistent with their ethical obligations outlined in the Code. When appropriate standards of care do not exist, computing professionals have a duty to ensure they are developed.

4. COMPLIANCE WITH THE CODE.

A computing professional should...

4.1 Uphold, promote, and respect the principles of the Code.

The future of computing depends on both technical and ethical excellence. Computing professionals should adhere to the principles of the Code and contribute to improving them. Computing professionals who recognize breaches of the Code should take actions to resolve the ethical issues they recognize, including, when reasonable, expressing their concern to the person or persons thought to be violating the Code.

4.2 Treat violations of the Code as inconsistent with membership in the ACM.

Each ACM member should encourage and support adherence by all computing professionals regardless of ACM membership. ACM members who recognize a breach of the Code should consider reporting the violation to the ACM, which may result in remedial action as specified in the ACM's [Code of Ethics and Professional Conduct Enforcement Policy](#).

The Code and guidelines were developed by the ACM Code 2018 Task Force: Executive Committee Don Gotterbarn (Chair), Bo Brinkman, Catherine Flick, Michael S Kirkpatrick, Keith Miller, Kate Varansky, and Marty J Wolf. Members: Eve Anderson, Ron Anderson, Amy Bruckman, Karla Carter, Michael Davis, Penny Duquenoy, Jeremy Epstein, Kai Kimppa, Lorraine Kisselburgh, Shrawan Kumar, Andrew McGetrick, Natasha

Milic-Frayling, Denise Oram, Simon Rogerson, David Shama, Janice Sipior, Eugene Spafford, and Les Waguespack. The Task Force was organized by the ACM Committee on Professional Ethics. Significant contributions to the Code were also made by the broader international ACM membership. This Code and its guidelines were adopted by the ACM Council on June 22nd, 2018.

This Code may be published without permission as long as it is not changed in any way and it carries the copyright notice. Copyright (c) 2018 by the Association for Computing Machinery.

Be sure to check out our guide on [Using the Code](#) for decision making for practicing engineers.

The official repository of the ACM Code of Ethics and Professional Conduct is <https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct>. This Code constitutes [Bylaw 15](#) of the Bylaws of the Association for Computing Machinery.

ACM Ethics

Proudly powered by WordPress.

ACM Ethics

The Official Site of the Association for
Computing Machinery's Committee on
Professional Ethics

The joint ACM/IEEE-CS Software Engineering Code was published as: Don Gotterbarn, Keith Miller, and Simon Rogerson. 1997. Software engineering code of ethics. *Commun. ACM* 40, 11 (November 1997), 110-118.

DOI: [10.1145/265684.265699](https://doi.org/10.1145/265684.265699)

Note that this code is for anyone that is a member of the software engineering profession, regardless of ACM membership status. You may also wish to consult [The Code for all ACM members](#) (regardless of profession).

Thanks to [SEERI](#) for these translations of the SE Code: 

Want to contribute a translation? Get it touch via the [Contact Us](#) page.

The Software Engineering Code of Ethics and Professional Practice

Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices and jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering.

Software Engineering Code of Ethics and Professional Practice (Short Version)

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details 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 Eight 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 that 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.

Software Engineering Code of Ethics and Professional Practice (Full Version)

PREAMBLE

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code be used in isolation to justify errors of omission or commission. The list of Principles and Clauses is not exhaustive. The Clauses should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations. The Code is not a simple ethical algorithm that generates ethical decisions. In some situations standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the

circumstances.

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.

The dynamic and demanding context of software engineering requires a code that is adaptable and relevant to new situations as they occur. However, even in this generality, the Code provides support for software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions that are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. As this Code expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

PRINCIPLES

Principle 1: PUBLIC

Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

- 1.01. Accept full responsibility for their own work.
- 1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.
- 1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.
- 1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.
- 1.05. Cooperate in efforts to address matters of grave public concern caused by software, its installation, maintenance, support or documentation.
- 1.06. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.
- 1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that

can diminish access to the benefits of software.

1.08. Be encouraged to volunteer professional skills to good causes and contribute to public education concerning the discipline.

Principle 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. In particular, software engineers shall, as appropriate:

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.

2.02. Not knowingly use software that is obtained or retained either illegally or unethically.

2.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.

2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.

2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

2.06. Identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.

2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client.

2.08. Accept no outside work detrimental to the work they perform for their primary employer.

2.09. Promote no interest adverse to their employer or client, unless a higher ethical concern is being compromised; in that case, inform the employer or another appropriate authority of the ethical concern.

Principle 3: PRODUCT

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.

3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.

- 3.04. Ensure that they are qualified for any project on which they work or propose to work by an appropriate combination of education and training, and experience.
- 3.05. Ensure an appropriate method is used for any project on which they work or propose to work.
- 3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.
- 3.07. Strive to fully understand the specifications for software on which they work.
- 3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements and have the appropriate approvals.
- 3.09. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work and provide an uncertainty assessment of these estimates.
- 3.10. Ensure adequate testing, debugging, and review of software and related documents on which they work.
- 3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.
- 3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.
- 3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.
- 3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.
- 3.15. Treat all forms of software maintenance with the same professionalism as new development.

Principle 4: JUDGMENT

Software engineers shall maintain integrity and independence in their professional judgment. In particular, software engineers shall, as appropriate:

- 4.01. Temper all technical judgments by the need to support and maintain human values.
- 4.02. Only endorse documents either prepared under their supervision or within their areas of competence and with which they are in agreement.
- 4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.
- 4.04. Not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.

4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.

4.06. Refuse to participate, as members or advisors, in a private, governmental or professional body concerned with software related issues, in which they, their employers or their clients have undisclosed potential conflicts of interest.

Principle 5: MANAGEMENT

Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance . In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.

5.02. Ensure that software engineers are informed of standards before being held to them.

5.03. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files and information that is confidential to the employer or confidential to others.

5.04. Assign work only after taking into account appropriate contributions of education and experience tempered with a desire to further that education and experience.

5.05. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates.

5.06. Attract potential software engineers only by full and accurate description of the conditions of employment.

5.07. Offer fair and just remuneration.

5.08. Not unjustly prevent someone from taking a position for which that person is suitably qualified.

5.09. Ensure that there is a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed.

5.10. Provide for due process in hearing charges of violation of an employer's policy or of this Code.

5.11. Not ask a software engineer to do anything inconsistent with this Code.

5.12. Not punish anyone for expressing ethical concerns about a project.

Principle 6: PROFESSION

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

6.01. Help develop an organizational environment favorable to acting ethically.

- 6.02. Promote public knowledge of software engineering.
 - 6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.
 - 6.04. Support, as members of a profession, other software engineers striving to follow this Code.
 - 6.05. Not promote their own interest at the expense of the profession, client or employer.
 - 6.06. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.
 - 6.07. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful.
 - 6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.
 - 6.09. Ensure that clients, employers, and supervisors know of the software engineer's commitment to this Code of ethics, and the subsequent ramifications of such commitment.
 - 6.10. Avoid associations with businesses and organizations which are in conflict with this code.
 - 6.11. Recognize that violations of this Code are inconsistent with being a professional software engineer.
 - 6.12. Express concerns to the people involved when significant violations of this Code are detected unless this is impossible, counter-productive, or dangerous.
 - 6.13. Report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counter-productive or dangerous.
- Principle 7: COLLEAGUES**
- Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:
- 7.01. Encourage colleagues to adhere to this Code.
 - 7.02. Assist colleagues in professional development.
 - 7.03. Credit fully the work of others and refrain from taking undue credit.
 - 7.04. Review the work of others in an objective, candid, and properly-documented way.
 - 7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.
 - 7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures

for protecting passwords, files and other confidential information, and security measures in general.

7.07. Not unfairly intervene in the career of any colleague; however, concern for the employer, the client or public interest may compel software engineers, in good faith, to question the competence of a colleague.

7.08. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.

Principle 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. In particular, software engineers shall continually endeavor to:

8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance and testing of software and related documents, together with the management of the development process.

8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.

8.03. Improve their ability to produce accurate, informative, and well-written documentation.

8.04. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.

8.05. Improve their knowledge of relevant standards and the law governing the software and related documents on which they work.

8.06. Improve their knowledge of this Code, its interpretation, and its application to their work.

8.07. Not give unfair treatment to anyone because of any irrelevant prejudices.

8.08. Not influence others to undertake any action that involves a breach of this Code.

8.09. Recognize that personal violations of this Code are inconsistent with being a professional software engineer.

This Code was developed by the ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices (SEPP):

Executive Committee: Donald Gotterbarn (Chair), Keith Miller and Simon Rogerson;

Members: Steve Barber, Peter Barnes, Ilene Burnstein, Michael Davis, Amr El-Kadi, N. Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Mark Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manuel J. Norman, Douglas Phillips, Peter Ron Prinzivalli, Patrick Sullivan, John Weckert, Vivian Weil, S. Weisband and Laurie Honour Werth.

This Code may be published without permission as long as it is not changed in any way and it carries the copyright notice. Copyright (c) 1999 by the Association for Computing Machinery, Inc. and the Institute for Electrical and Electronics Engineers, Inc.

ACM Ethics

Proudly powered by WordPress.



June 27, 2023

PRINCIPLES FOR THE DEVELOPMENT, DEPLOYMENT, AND USE OF GENERATIVE AI TECHNOLOGIES*

Introduction

Generative Artificial Intelligence (AI) is a broad term used to describe computing techniques and tools that can be used to create new content, including: text, speech and audio, images and video, computer code, and other digital artifacts.¹ While such systems offer tremendous opportunities for benefits to society, they also pose very significant risks.² The increasing power of generative AI systems, the speed of their evolution, broad application, and potential to cause significant or even catastrophic harm means that great care must be taken in researching, designing, developing, deploying, and using them. Existing mechanisms and modes for avoiding such harm likely will not suffice.

* Lead authors of this document for USTPC were Ravi Jain, Jeanna Matthews, and Alejandro Saucedo. Important contributions were made by Harish Arunachalam, Brian Dean, Advait Deshpande, Simson Garfinkel, Andrew Gross, Jim Hendler, Lorraine Kisselburgh, Srivatsa Kundurthy, Marc Rotenberg, Stuart Shapiro, and Ben Shneiderman. Assistance also was provided by: Ricardo Baeza-Yates, Michel Beaudouin-Lafon, Vint Cerf, Charalampos Chelmis, Paul DeMarinis, Nicholas Diakopoulos, Janet Haven, Ravi Iyer, Carlos E. Jimenez-Gomez, Mark Pastin, Neeti Pokhriyal, Jason Schmitt, and Darryl Scriven.

¹ The first set of generative AI advances rest on very large AI models that are trained on an extremely large corpus of data. Examples that are text-oriented include BLOOM, Chinchilla, GPT-4, LaMDA, and OPT, as well as conversation oriented models like Bard, ChatGPT, and others. By definition, this is a rapidly evolving area. This list of examples, therefore, is by no means intended to be exhaustive. Similarly, the principles advanced in this document also are certain to evolve in response to changing circumstances, technological capabilities, and societal norms.

² Generative AI models and tools offer significant new opportunities for enhancing numerous online experiences and services, automating tasks normally done by humans, and assisting and enhancing human creativity. From another perspective, such models and tools also have raised significant concerns about multiple aspects of information and its use, including accuracy, disinformation, deception, data collection, ownership, attribution, accountability, transparency, bias, user control, confidentiality, privacy, and security. Generative AI also raises important questions outside the scope of this document, including many about the replacement of human labor and jobs by AI-based machines and automation.

This statement puts forward principles and recommendations for best practices in these and related areas based on a technical understanding of generative AI systems.³ The first four principles, which are specific to generative AI, address issues regarding limits of use, ownership, personal data control, and correctability. The following four principles were derived and adapted from the joint *ACM Statement on Principles for Responsible Algorithmic Systems*⁴ released in October 2022. These pertain to transparency, auditability and contestability, limiting environmental impacts, and security and privacy.⁵

This statement also reaffirms and includes five principles from the joint statement as originally formulated and has been informed by the January 2023 [ACM TechBrief: Safer Algorithmic Systems](#).

The following instrumental principles, consistent with the ACM Code of Ethics,⁶ are intended to foster fair, accurate, and beneficial decision-making concerning generative and all other AI technologies:

Generative AI-Specific Principles

- 1. Limits and guidance on deployment and use:** In consultation with all stakeholders, current law and regulation should be reviewed and applied as written or revised to limit the deployment and use of generative AI technologies when required to minimize harm. No high-risk AI system should be allowed to operate without clear and adequate safeguards, including a “human in the loop” and clear consensus among relevant stakeholders that the system’s benefits will substantially outweigh its potential negative impacts.

³ Technical considerations do not, however, exist in a vacuum. In many cases, they thus have led us to also recommend that legal, regulatory, and policy issues raised by generative AI be discussed transparently among multiple stakeholders. The goal of such efforts must be appropriately robust frameworks for oversight of these technologies grounded firmly in technical fundamentals and practice. The safe and responsible use of generative AI will be possible only with the transparent and consistent collaboration over time of all impacted stakeholders.

⁴ [Statement on Principles for Responsible Algorithmic Systems](#), ACM Technology Policy Council and its Europe and U.S. Technology Policy Committees (October 26, 2022) <https://www.acm.org/binaries/content/assets/public-policy/final-joint-ai-statement-update.pdf> (Joint Statement).

⁵ Multiple additional principles articulated in the joint statement also remain germane and are restated in the last section of this document. They concern legitimacy and competency, minimizing harms, interpretability and explainability, maintainability, and accountability and responsibility.

⁶ The ACM Code of Ethics and Professional Conduct was designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle. See <https://www.acm.org/code-of-ethics>.

Providers⁷ should undertake extensive impact assessments prior to the deployment of such technologies to thoughtfully ensure that the benefits to society of any such deployment outweigh its risks. One approach is to define a hierarchy of risk levels, with unacceptable risk at the highest level and minimal risk at the lowest level.⁸ Such categorizations must include the risk that users who attribute human characteristics or behavior to generative AI systems inappropriately, may be more likely to rely upon such systems' outputs and experience harm.

Providers of generative AI systems released to the general public should provide recommendations for the correct and responsible use of those systems, and also provide sufficient information about such systems to permit expert evaluation of their risks and impacts.⁹ Finally, providers should enable mechanisms to allow generative AI systems to be deactivated unilaterally by external means in emergency situations.

2. Ownership: Inherent aspects of how generative AI systems are structured and function are not yet adequately accounted for in intellectual property (IP) law and regulation.¹⁰ Such

⁷ "Providers" is used in this document to mean all entities that deliver generative AI technologies, components, systems, or applications to users or other entities. This may include developers; model, dataset, subsystem, platform, system, or application providers; and parties such as sellers, resellers, integrators, or marketers.

⁸ Various bodies such as the National Institute of Standards and Technology (NIST), the Institute of Electrical and Electronics Engineers (IEEE), and the European Union (EU) have made recommendations that are relevant in this regard. (NIST has formulated a risk management framework while the IEEE and EU articulate a risk hierarchy.) See respectively: National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, January 2023 [<https://doi.org/10.6028/NIST.AI.100-1>]; *IEEE Standard for System, Software, and Hardware Verification and Validation*, 1012-2016 [<https://ieeexplore.ieee.org/document/8055462>]; and *Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts*, 2021/0106 (COD), April 21, 2023 [https://eurlex.europa.eu/resource.html?uri=cellar:e0649735-a372-11eb-9585-01aa75ed71a1.0001.02/DOC_1&format=PDF].

Risk assessments of generative AI systems should be done by teams of cross-disciplinary experts and public, private, and non-governmental bodies, and with broad public input. We also note that generative AI systems are complex, not yet fully understood, and may demonstrate emergent behaviors and emergent risks that are not predictable simply by extrapolating from their existing capabilities. This is an area that thus needs substantial further research. Another such area is that of bias which, while a risk in AI systems in general, has become a particularly significant concern with the large language models used in generative AI.

⁹ Generative AI providers should also provide meta-information about models to enable experts and trained members of the community to understand them and evaluate their impacts. Such information might productively include datasheets, model cards, model whitepapers, factsheets, and detailed impact assessments. Well-designed dashboards also could give users a clearer understanding of the impact of their decisions of how best to use generative AI systems, and greater control over their output.

¹⁰ It is not currently possible, for example, for users or creators of generative AI systems to definitively say which portions of a training dataset adhere to which copyrights or licenses, which portions of that dataset may have directly or indirectly contributed to a particular generated artifact, and consequently what the copyright and licensing implications of that artifact may be. This not only creates an issue for creators whose works have been used to generate artifacts, but also for users of those artifacts who may be exposed to the risk of substantial penalties for copyright violations.

regimes thus should be reviewed and, where necessary, revised to strengthen protections for human creators without placing undue restrictions on lawful permissive access to copyrighted material (e.g., pursuant to fair use or fair dealing provisions in the US and Europe)¹¹ or diminishing the overall creative commons.¹²

3. **Personal data control:** Generative AI systems should allow a person to opt out of their data being used to train the system or facilitate its generation of information. In many cases, the default choice should be for a person to explicitly opt into their data being used. At minimum, such systems should provide mechanisms to allow any person to opt out of their personal data, including their biometric data, being used for such purposes.¹³ If a person opts out of providing data once a model has been trained, there should be a mechanism in place to update the model to remove that individual's data.
4. **Correctability:** Providers of generative AI systems should create and maintain public repositories where errors made by the system can be noted and, optionally, corrections made. If an error is discovered and noted, providers should develop transparent mechanisms that allow stakeholders to track providers' progress toward eliminating errors, including the retraining of models and other mitigations as needed.

¹¹ In the United States, a person's original and creative works are automatically copyrighted when first "fixed in a medium of tangible expression." Generally, absent prior approval by the copyright holder, works cannot be used unless deemed a "fair use" under a four-factor statutory test, or they are subject to a limited number of express other statutory exceptions. Other countries may or may not provide similar protection for works created within their own jurisdictions.

¹² Areas of creative work that have traditionally fallen outside of IP controls, such as artistic style, become contentious when a generative AI tool is able to reduce demand for the efforts of human creators through automated mimicry, especially without citation of the works of human creators in a training set. This is especially critical since, unlike a human, the tool can do so quickly and at large scale. At the same time, while traditional notions of fair and acceptable use of copyrighted works allow for certain digital processes to be carried out on them (e.g., to display the works on a screen), it is not clear that this "authorization" will include their use as training data for AI to generate further artifacts in all jurisdictions. Other unforeseen scenarios or outcomes about the uses of generative AI for creative works that either test the boundaries of existing laws and regulations or lack any legal precedent may emerge in the future. We note with concern, for example, attempts at for-profit monetization of human-generated work available through a creative commons and/or publicly available dataset with explicit or implicit human IP attached that contravenes the original intent of or arrangements under which the IP was made available. Such use cases, and doubtless many others, must be addressed by new statutes or judicially resolved on a case-by-case basis.

¹³ Biometric data has been afforded particular protection in some jurisdictions. In the United States, for example, regulation of its use is a matter of state law, both common and statutory. See, e.g., the *Illinois Biometric Information Privacy Act*, 740 ILCS 14 (2008), which places limits on the use of personal images and likenesses [<https://www.ilga.gov/legislation/ilcs/ilcs3.asp?ActID=3004&ChapterID=57>]. The European Union's General Data Protection Regulation provides broad similar protection. *Regulation (EU) 2016/679 of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC*, April 2016 [<https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>].

Adapted Prior Principles

5. **Transparency:** Any application or system that utilizes generative AI should conspicuously disclose that it does so to the appropriate stakeholders. In particular, where generative AI is being used to simulate human agents, at all times individuals must be promptly and clearly informed that they are interacting with a system as opposed to a human.¹⁴ Further, generative AI systems should warn users that information the system generates may contain errors, and that their authoritative tone or other attributes may be misleading. In addition, to prevent unintended or malicious misrepresentation (e.g., “deepfakes”), generative AI systems should provide a mechanism that permits information they generate to be unambiguously identified by third parties as having been AI produced. Such techniques may include cryptographic or steganographic markers.
6. **Auditability and contestability:** Providers of Generative AI systems should ensure that system models, algorithms, data, and outputs can be recorded where possible (with due consideration to privacy), so that they may be audited and/or contested in appropriate cases. It is also important that providers of Generative AI systems have appropriate auditing strategies in place so citizens, consumer groups, and industry bodies can review and comment on them over time to facilitate their correction and potential retraining.
7. **Limiting environmental impacts:** Given the large environmental impacts of Generative AI models,¹⁵ we recommend that consensus on methodologies be developed to measure, attribute, and actively reduce such impacts. In particular, the total environmental costs to society, including those that are externalized by providers of the technology, must be determinable and attributed to the relevant entities in the ecosystem. Finally, sustainability issues also should be considered and accounted for during a system's entire life cycle.¹⁶

¹⁴ The necessity for transparency becomes even more critical when generative AI intentionally simulates human agents as some users may anthropomorphize such systems inappropriately. A related issue is that some generative AI systems can present their outputs in authoritative language and a manner that conveys their confidence to users. However, the systems are ultimately limited by their training datasets, and the quantity and quality of training sets as well the techniques used can lead to subtle errors. This may cause users to miss errors that have been generated by the AI (sometimes called “hallucinations”) or be lulled into not checking for them adequately, if at all.

¹⁵ The cumulative estimated carbon emissions of recently released Generative AI models have been estimated to greatly exceed those of more traditional AI models. As the use of Generative AI grows such emissions could increase significantly. See C.J. Wu et al., Sustainable AI: Environmental Implications, Challenges and Opportunities, Conference on Machine Learning and Systems (MLSys), 2022.

[https://www.researchgate.net/publication/355843251_Sustainable_AI_Environmental_Implications_Challenges_and_Opportunities]

¹⁶ Such analysis must extend beyond simply focusing on operational efficiency during system training or inference to include, e.g., the tradeoff between AI performance and environmental impact, or techniques to reduce or reuse model training runs or artifacts.

8. **Heightened security and privacy:** Generative AI systems are susceptible to a broad range of new security¹⁷ and privacy¹⁸ risks, including new attack vectors and malicious data leaks, among others. Their use, therefore, requires heightened risk-mitigation controls to ensure that relevant security and privacy best practices are verifiably and consistently employed throughout the model life cycle, and that these can be effectively audited, both internally and as appropriate by third parties.

Reaffirmed Principles

Five additional principles articulated in our October 2022 *joint statement* also continue to apply as originally written to generative and other AI systems. They are reaffirmed and included here for completeness and ease of reference:

9. **Legitimacy and competency:** Designers of algorithmic systems should have the management competence and explicit authorization to build and deploy such systems. They also need to have expertise in the application domain, a scientific basis for the systems' intended use, and be widely regarded as socially legitimate by stakeholders impacted by the system.¹⁹ Legal and ethical assessments must be conducted to confirm that any risks introduced by the systems will be proportional to the problems being addressed, and that any benefit-harm trade-offs are understood by all relevant stakeholders.
10. **Minimizing harm:** Managers, designers, developers, users, and other stakeholders of algorithmic systems should be aware of the possible errors and biases involved in their design, implementation, and use, and the potential harm that a system can cause to individuals and society. Organizations should routinely perform impact assessments on systems they employ to determine whether the system could generate harm, especially discriminatory harm, and to apply appropriate mitigations. When possible, they should learn from measures of actual performance, not solely patterns of past decisions that may themselves have been discriminatory.

¹⁷ For example, the use of generative AI models to generate computer code presents substantial security risks. Such models are typically trained on code repositories. If any credentials are stored with the code, malicious actors could exploit the model to output valid keys. Indeed, they could go even further and introduce malware in response to queries, whether by poisoning training data or corrupting system outputs. Analogous security risks also exist for many other types of generative AI models.

¹⁸ The inherently necessary use of large training dataset and model sizes for generative AI systems can lead to privacy issues becoming more likely or severe than for smaller models or datasets. Models may directly or indirectly infer personally identifiable information (such as employment, home address, and family data) of particular individuals, which are then susceptible to data leaks. Similarly, there are risks of reverse engineering training data from trained models. (Although models amalgamate training data, it has been proven that training examples may nonetheless be recovered in this process.) See for example, Carlini et al., *Quantifying Memorization Across Neural Language Models*, Conference on Learning Representation, 2023. [<https://iclr.cc/virtual/2023/oral/12637>]

¹⁹ Projects with no clear scientific basis (e.g., inferring personality traits from facial images) should not be deployed.

- 11. Interpretability and explainability:** Managers of algorithmic systems are encouraged to produce information regarding both the procedures that the employed algorithms follow (interpretability) and the specific decisions that they make (explainability). Explainability may be just as important as accuracy, especially in public policy contexts or any environment in which there are concerns about how algorithms could be skewed to benefit one group over another without acknowledgement. It is important to distinguish between explanations and after-the-fact rationalizations that do not reflect the evidence, or the decision-making process used to reach the conclusion being explained.
- 12. Maintainability:** Evidence of all algorithmic systems' soundness should be collected throughout their life cycles, including documentation of system requirements, the design or implementation of changes, test cases and results, and a log of errors found and fixed.²⁰ Proper maintenance may require retraining systems with new training data and/or replacing the models employed.
- 13. Accountability and responsibility:** Public and private bodies should be held accountable for decisions made by algorithms they use, even if it is not feasible to explain in detail how those algorithms produced their results. Such bodies should be responsible for entire systems as deployed in their specific contexts, not just for the individual parts that make up a given system. When problems in automated systems are detected, organizations responsible for deploying those systems should document the specific actions that they will take to remediate the problem and under what circumstances the use of such technologies should be suspended or terminated.

²⁰ Otherwise, the system may become less appropriate as inputs drift from those originally anticipated, or if the underlying real-world conditions change (e.g., facial recognition systems are used on a wider or different demographic than was present in the training data).