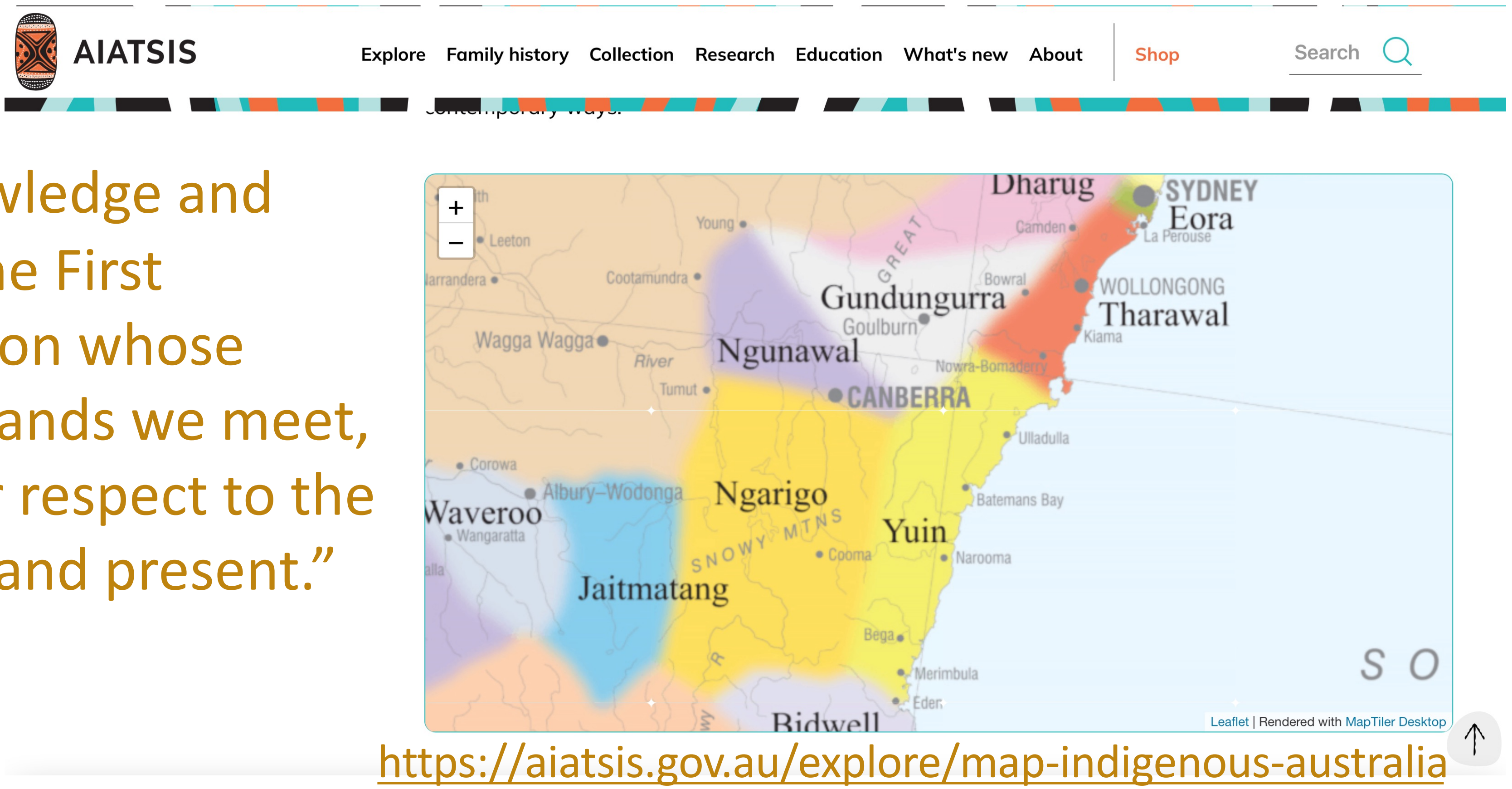
Week:
1 of 12

COMP 2120 / COMP 6120

INTRODUCTION TO AGILE

A/Prof Alex Potanin

ANU Acknowledgment of Country

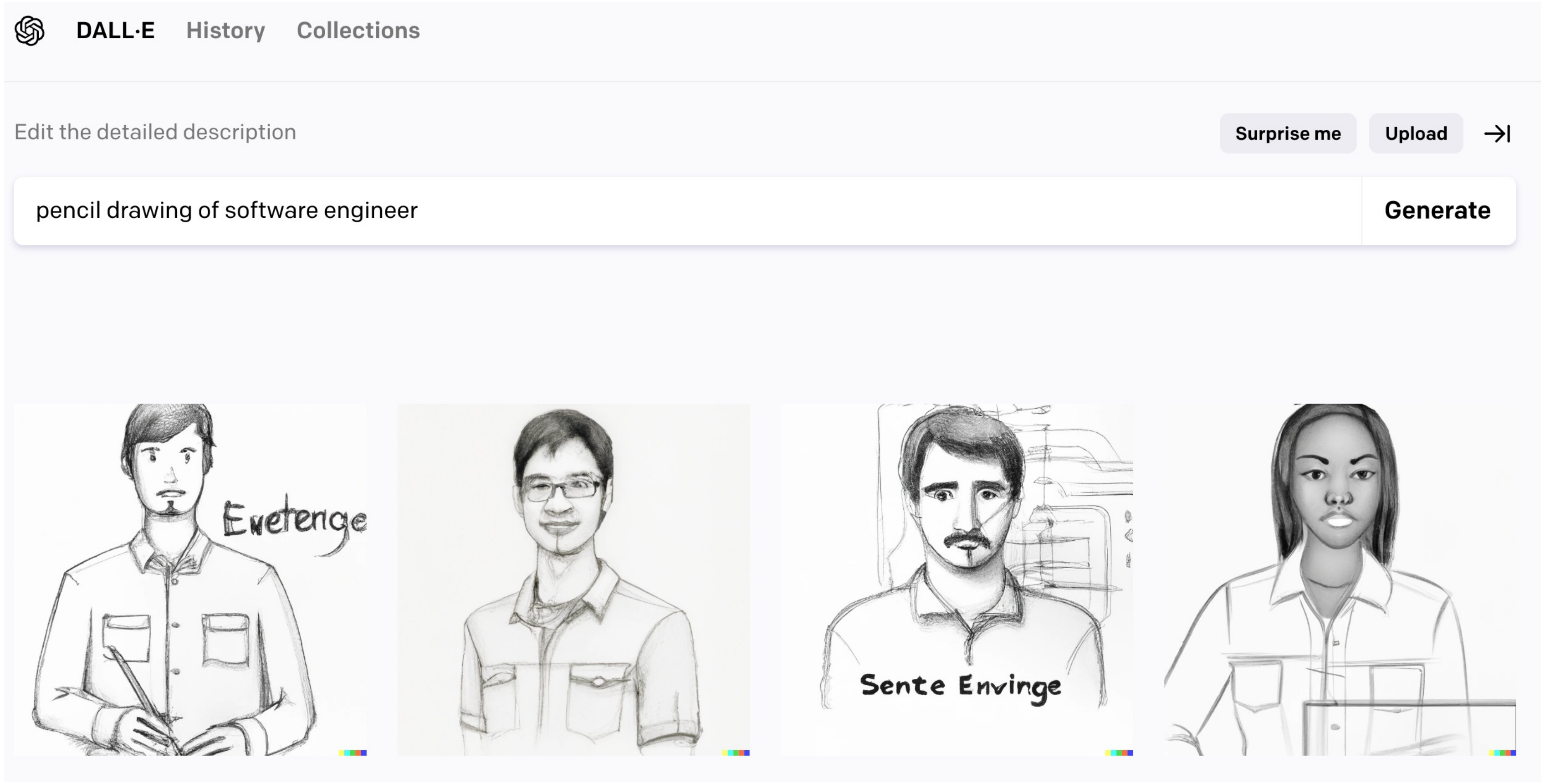


"We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present."

https://aiatsis.gov.au/explore/map-indigenous-australia

Today

- Course Overview
- Disasters
- Software Products
- Product Vision and Product Management
- *Estimating Effort*
- *Agile*
- *Scrum*

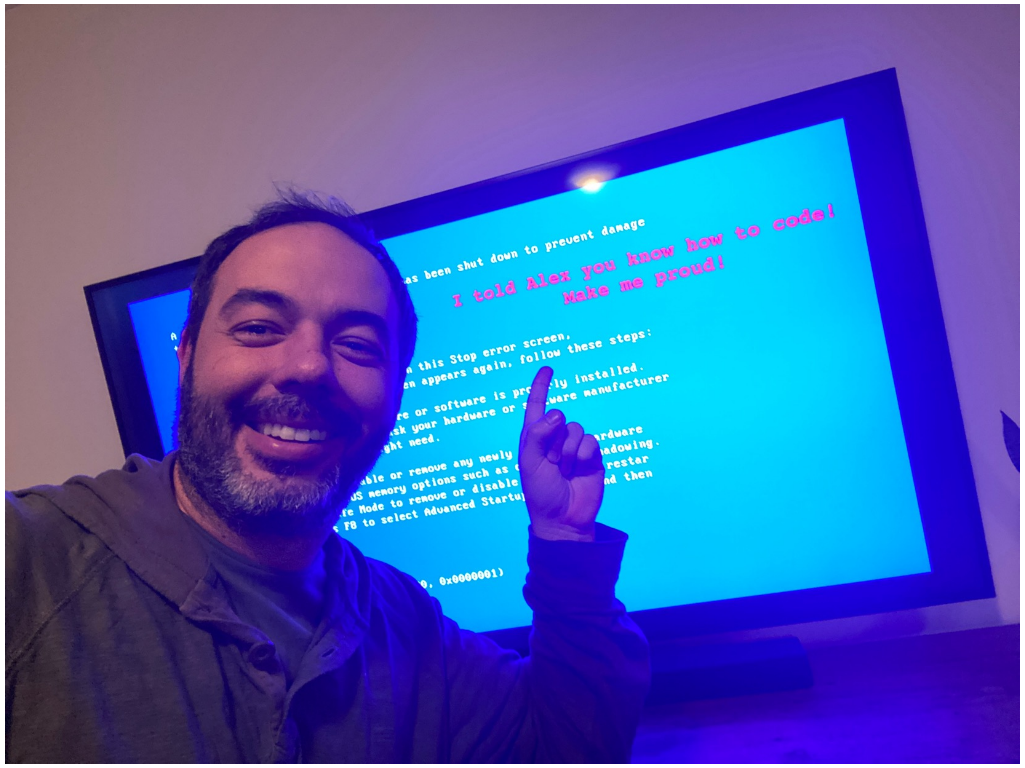


pencil drawing of software engineer

Course Overview

COMP 2120 / COMP 6120 Overview

- Title: “Software Engineering”
- 4th course in the sequence of:
  - COMP 1100 “Programming as Problem Solving”
  - COMP 1110 “Structured Programming”
  - COMP 2100 “Software Design Methodologies”
  - COMP 2120 “Software Engineering”



COMP 2100 / 6442 vs COMP 2120 / 6120

COMP2100/6442: *Software Engineering: Software Construction*

1. Apply fundamental programming concepts for medium scale programs
2. Understand basic types and the benefits of static typing, with understanding of generics, subtyping, and overloading, and their roles in structuring programs
3. Map programming language abstractions through to execution environment; use non-source (text) internal representations of programs (e.g., abstract syntax trees); sketch low-level run-time representations of core language constructs (objects and closures)
4. …

COMP2120/6120: *Advanced Software Engineering: Building Large Systems*

1. Apply modern requirements gathering and software design techniques in the context of a realistic software engineering process
2. Evaluate other people's code contributions using modern formal code inspection approaches
3. Apply correctly techniques for ensuring and assessing the quality of software
4. Competently analyse a modern large-scale software project with continuous integration
5. Work co-operatively in a team to solve a software engineering problem
6. Evaluate and justify the prioritisation of requirements

About Us

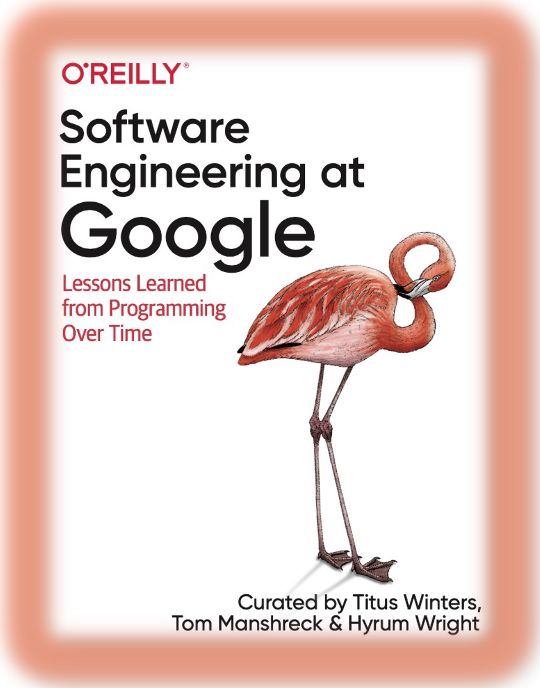
- Alex Potanin: from Russia via South Korea and Czech Republic to New Zealand and now Australia. Family from USA, Germany, China, Indonesia, and New Zealand. ☺
- Interested in *Secure* Programming Languages and Software Architecture (*Module* systems and *Object Capabilities*): https://potanin.github.io/ and http://wyvernlang.github.io
- Fabian Muehlboeck: from Austria via USA back to Austria (but arrived in Australia via a spelling mistake on the ticket). ☺
- Interested in *Usable* Programming Languages and Gradual Typing, as well as showing how to break all the modern languages like Java etc: https://fabian.muehlboeck.name/

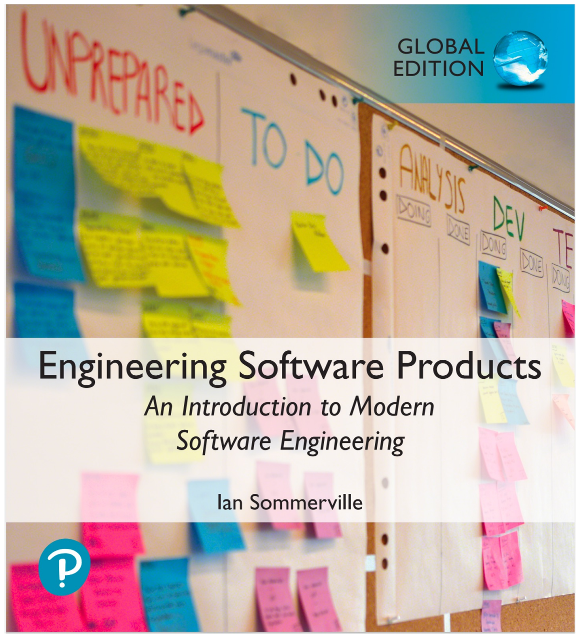
Office Hours: *catch me if you can* right after our lecture here

COMP 2120 / COMP 6120 Overview

- https://comp.anu.edu.au/courses/comp2120/
- *Use ED Discussion for the Forums!*
- **Tutorials prepare you for the assignments and future real-life work toolchains!**
- ***Sign up for a tutorial by the end of week 1! Groups formed based on the tutorials.***
- (Weeks 1 & 2) Assignment 1 on Agile Basics (5%)
- (Weeks 3, 4, 5) Assignment 2 (Group) on Inspection (15%)
- (Weeks 6, 7, 8) Assignment 3 (Group) on Adding Features (15%)
- (Due end of W11 and done in W12) Assignment 4 (Group) is Presentation on PR (5%)
- (Weeks 9, 10, 11, 12) Assignment 5 (Group) on PR for a real Open Source Project (15%)
- Final In-Person 3 Hour Closed Book Written/Typed-In (In The Lab) Exam (45%)
- **NB!** We plan to randomly rearrange the groups from people in the same tutorial in 2/3/4

COMP 2120 / 6120 Books


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright


GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

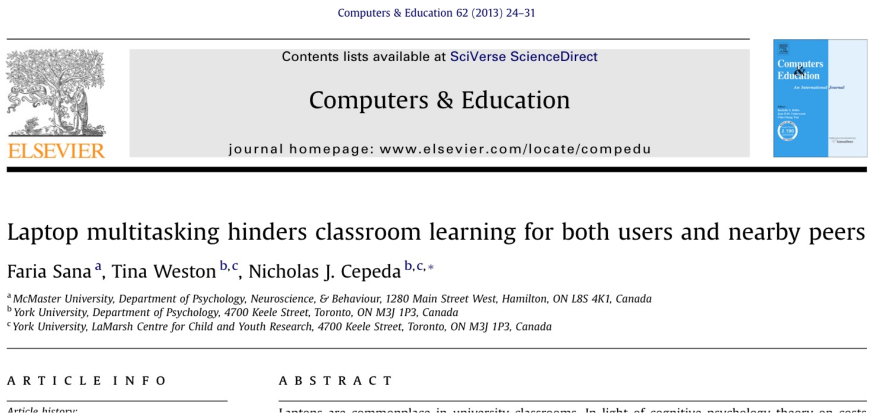
Name

Relevant Personal Link or Previous software development experience?

Manage Expectations or Software development ambitions?

HELLO
my name is
Inigo Montoya.
You killed my father.
Prepare to die.

"…participants who multitasked on a laptop during a lecture scored lower on a test compared to those who did not multitask, and participants who were in direct view of a multitasking peer scored lower on a test compared to those who were not. The results demonstrate that *multitasking on a laptop poses a significant distraction to both users and* ***fellow students*** *and can be detrimental to comprehension of lecture content."*



Computers & Education 62 (2013) 24–31

Contents lists available at SciVerse ScienceDirect

Computers & Education

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

ELSEVIER

Laptop multitasking hinders classroom learning for both users and nearby peers

Faria Sana [a], Tina Weston [b,c], Nicholas J. Cepeda [b,c,*]

[a] McMaster University, Department of Psychology, Neuroscience, & Behaviour, 1280 Main Street West, Hamilton, ON L8S 4K1, Canada
[b] York University, Department of Psychology, 4700 Keele Street, Toronto, ON M3J 1P3, Canada
[c] York University, LaMarsh Centre for Child and Youth Research, 4700 Keele Street, Toronto, ON M3J 1P3, Canada

ARTICLE INFO

Article history:

ABSTRACT

Laptops are commonplace in university classrooms. In light of cognitive psychology theory on costs

Teamwork: Expectations

- Meet regularly in your assigned tutorial slot – *in person*
- Divide work and integrate
- Establish a process
- **Set and document clear responsibilities and expectations**
  - Possible Roles: Coordinator, Scribe, Checker, Monitor
- Every team member should understand the entire solution
- **NB!** We will change the teams in assignments 2,3,4 by randomly rearranging people within the same tutorial to ensure you have a common meeting time every week (the SECOND hour of your 2 hour lab session where the tutor won't be present)

Teamwork: Dealing with problems

- *Most teams will encounter some problem*
- Openly report even minor team issues in individual part of assignments
- In-class discussions and case studies
- Additional material throughout semester
- The tutor will help you every week and I will try to attend at least one of your meetings

Teamwork: Planning and Communication Professionalism

- Trello, Microsoft Project, …
- GitHub or GitLab Wiki, Google Docs, …
- Email, Slack, Facebook Groups, …

- Being a professional means, you should work well with others
- The best professionals are those who make those around them better
- If you feel someone is not treating you or someone else in a professional manner, you have two options:
  - If you feel you have the standing to do so, speak up!
  - Reach out to the course staff, and we will meet with you privately to discuss it, as well as preserve your anonymity

Accessibility

(formerly Access and Inclusion)

supports students at the Australian National University whose participation in academic studies is impacted by...

- Disability (physical or learning)
- Mental Health Condition/s
- Chronic medical condition/s
- Short-term illness/injury

Accessibility also supports:

- Carers
- Elite Athletes

If your circumstances are identified here and you require support to achieve your academic goals, please visit the Accessibility Website to find out about registering.

💻: website link here https://www.anu.edu.au/students/contacts/accessibility

Special Exam Arrangements (SEAs)

Check the Deadlines to renew EAPs and request SEAs!!!
Students who do not inform Accessibility within the specified timeframes should be aware that their SEAs may not be implemented for the assessment or examination period.

New registrations after deadlines are considered on a case-by-case basis.

Contact Us

Accessibility
University Experience Division
Di Riddell Student Centre (Level 3)
Acton, ACT 2601

Contact number: +61 2 6125 5036
Email link here: access.inclusion@anu.edu.au
Website link here: www.anu.edu.au/students/contacts/access-inclusion

It is the student's responsibility to ensure that you have a valid Education Access Plan (EAP) in place with A&I at least two (2) weeks prior to examination periods if you want Special Exam Arrangements (SEAs), and that you contact your Course Convenor to arrange your SEAs.

CECC

CLASS REPRESENTATIVES

Class Student Representation is an important component of the teaching and learning quality assurance and quality improvement processes within the ANU College of Engineering and Computer Science (CECC).

Each semester, we put out a call for Course Representatives for all ANU College of Engineering, Computing and Cybernetics (CECC) courses. Students can nominate themselves for one or more of the courses they are enrolled in.

Australian National University

Roles and responsibilities:

The role of Student Representatives is to provide ongoing constructive feedback on behalf of the student cohort to Course Conveners and to Associate Directors (Education) for continuous improvements to the course.

- Act as the official liaison between your peers and convener.
- Be available and proactive in gathering feedback from your classmates.
- Attend regular meetings, and provide reports on course feedback to your course convener
- Close the feedback loop by reporting back to the class the outcomes of your meetings.

Note: Class representatives will need to be comfortable with their contact details being made available via Wattle to all students in the class.

For more information regarding roles and responsibilities, contact:

ANUSA CECC representatives: sa.cecs@anu.edu.au

Why become a class representative?

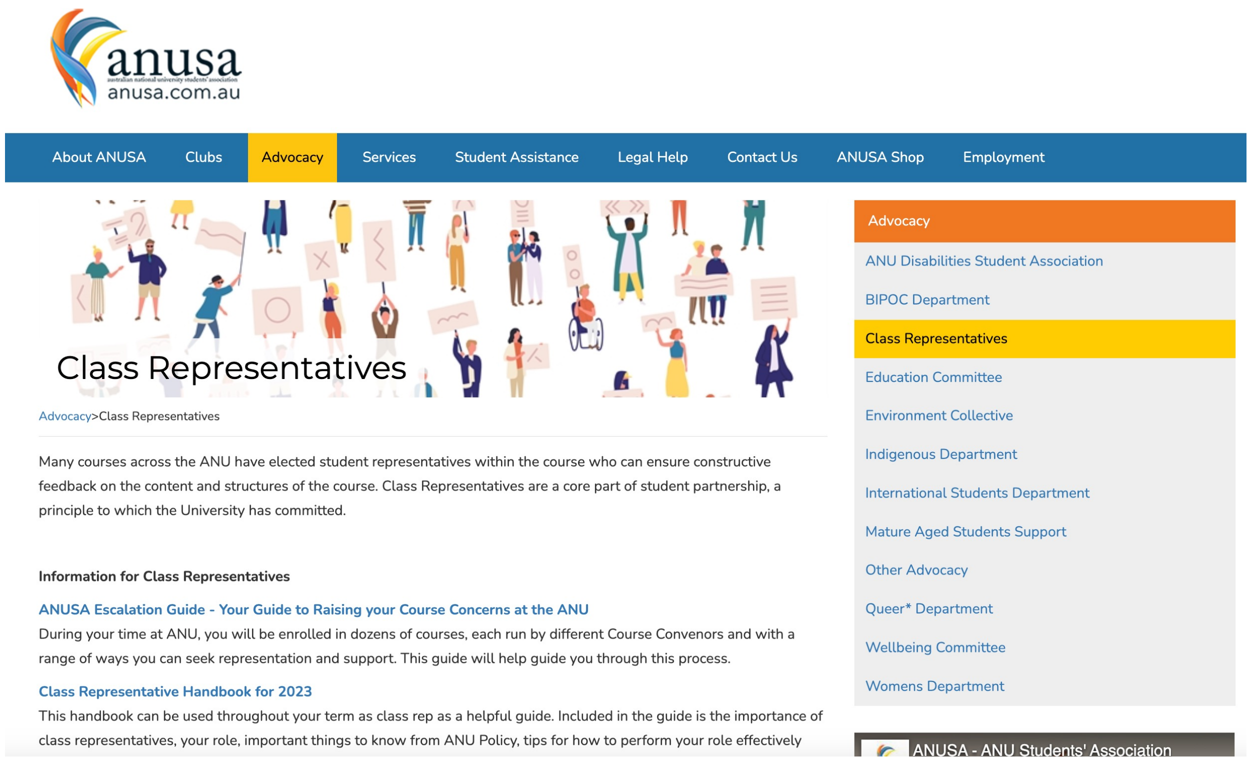
- **Ensure students have a voice** to their course convener, lecturer, tutors, and College.
- **Develop skills sought by employers**, including interpersonal, dispute resolution, leadership and communication skills.
- **Become empowered**. Play an active role in determining the direction of your education.
- **Become more aware of issues influencing your University** and current issues in higher education.
- **Course design and delivery.** Help shape the delivery of your current courses, as well as future improvements for following years.

Want to be a class representative?
Nominate today!

Please nominate yourself to your course convener by end of Week 1!

Class Representative(s)

- https://anusa.com.au/advocacy/classreps/
- Please email alex.potanin@anu.edu.au if you are keen with a one paragraph description of why you would be good at it by *23:59 Sunday, end of Week 1.*



Class Representatives

Advocacy>Class Representatives

Many courses across the ANU have elected student representatives within the course who can ensure constructive feedback on the content and structures of the course. Class Representatives are a core part of student partnership, a principle to which the University has committed.

Information for Class Representatives

ANUSA Escalation Guide - Your Guide to Raising your Course Concerns at the ANU
During your time at ANU, you will be enrolled in dozens of courses, each run by different Course Convenors and with a range of ways you can seek representation and support. This guide will help guide you through this process.

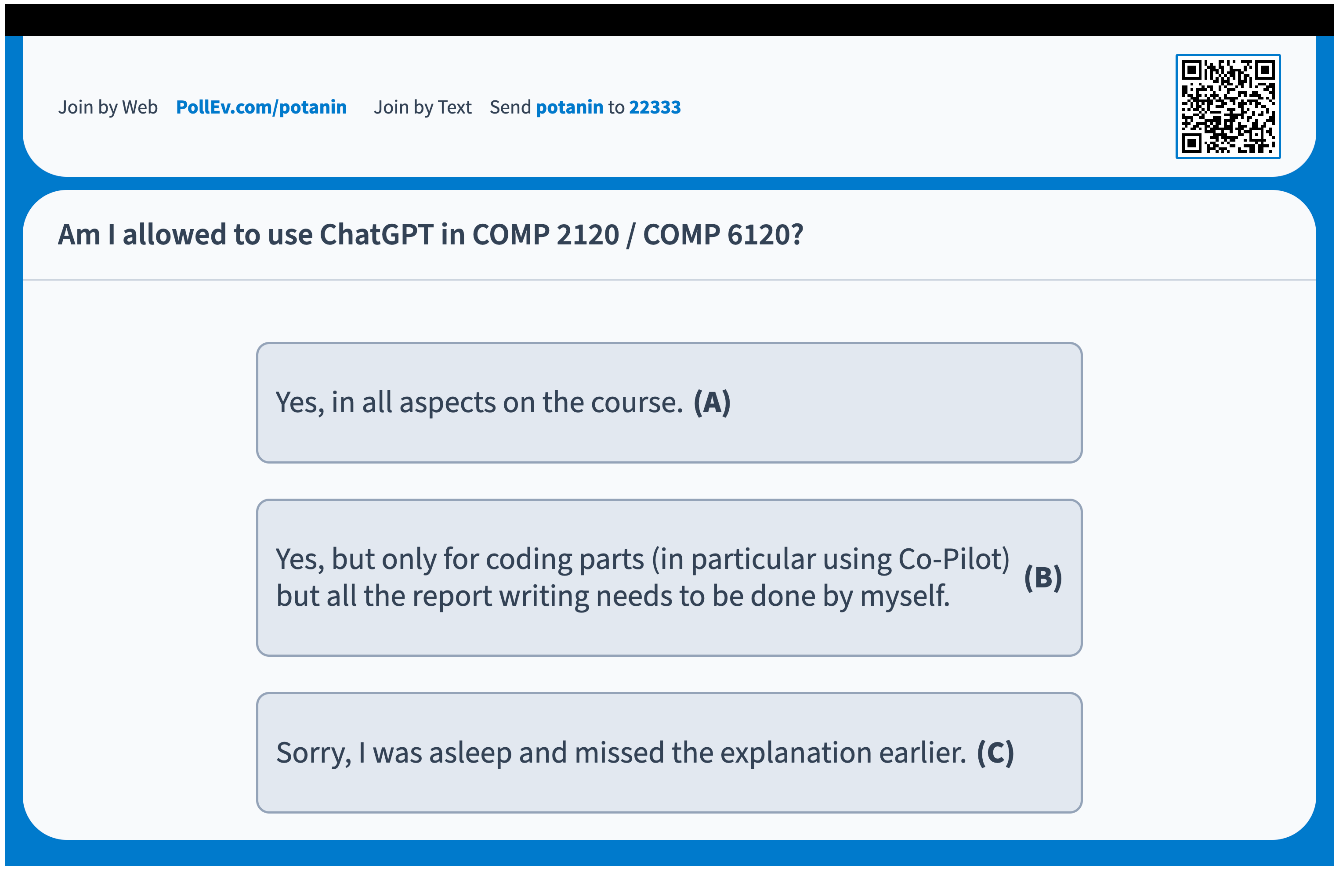
Class Representative Handbook for 2023
This handbook can be used throughout your term as class rep as a helpful guide. Included in the guide is the importance of class representatives, your role, important things to know from ANU Policy, tips for how to perform your role effectively

ChatGPT

Can I use ChatGPT or CoPilot etc?

- While we encourage you to get familiar with these tools and you may use CoPilot for the coding parts of our assignments, the point of this course is to teach and assess good report-writing techniques that would allow you to use ChatGPT successfully in the future. In the final paper-based exam you won’t be able to use ChatGPT so we do not want you to use it for the internal assessment items either to be able to receive appropriate feedback or to be able to pass the exam in the end. Good communication skills are essential in the software engineering profession so this course is designed to prepare you for it, including future use of ChatGPT where appropriate. This is similar to how there is no point using CoPilot when doing COMP1100, COMP1110, and COMP2100 as you learn the coding skills to be able to make smart use of CoPilot in courses such as the COMP2120 where we don’t mind you using generated code in an appropriate manner. You may be able to utilise ChatGPT as appropriate in the future once you learn how to write the appropriate text and communicate. This is similar to how primary school kids need to learn basic arithmetic before they can make good use of a calculator in a higher level of school.

Poll Everywhere Time!



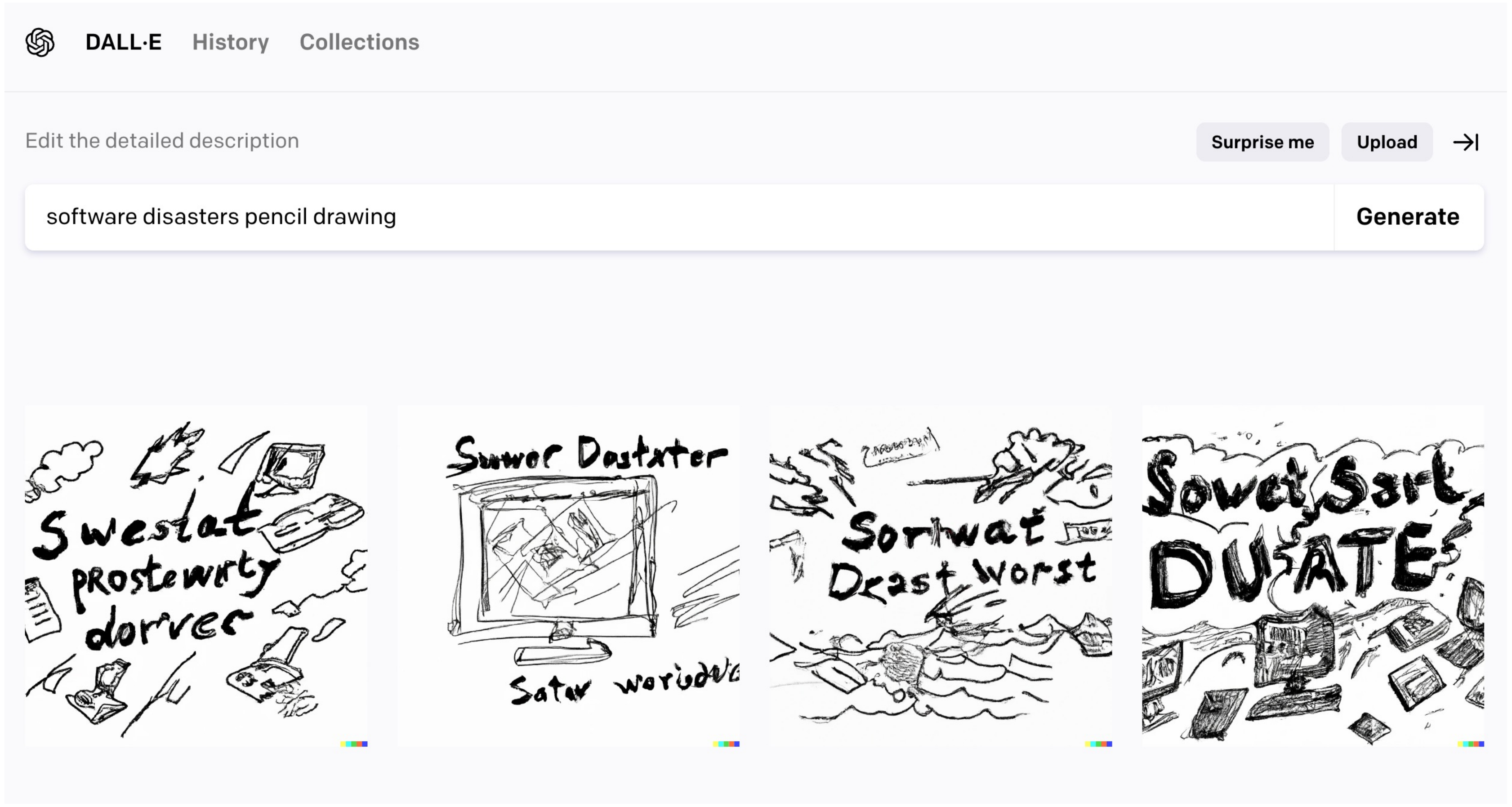
Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

Am I allowed to use ChatGPT in COMP 2120 / COMP 6120?

Yes, in all aspects on the course. **(A)**

Yes, but only for coding parts (in particular using Co-Pilot) but all the report writing needs to be done by myself. **(B)**

Sorry, I was asleep and missed the explanation earlier. **(C)**



software disasters pencil drawing

Disasters

Any slide with 17-313 Logo is based on this cool CMU course: https://cmu-313.github.io/

Software is Everywhere!

Lewis Hamilton rues Mercedes error that cost Australian Grand Prix win

- **Vettel wins after taking advantage of virtual-safety-car situation**
- **Mercedes data told Hamilton not to open greater lead**

Lewis Hamilton (left) finished second to Sebastian Vettel after a probably software error from Mercedes. Photograph: William West/AFP/Getty Images

Lewis Hamilton said he would prefer to trust his racer's instinct rather than instruction from his Mercedes team after they admitted it was likely a software error cost him victory in the opening grand prix of the season in Australia.

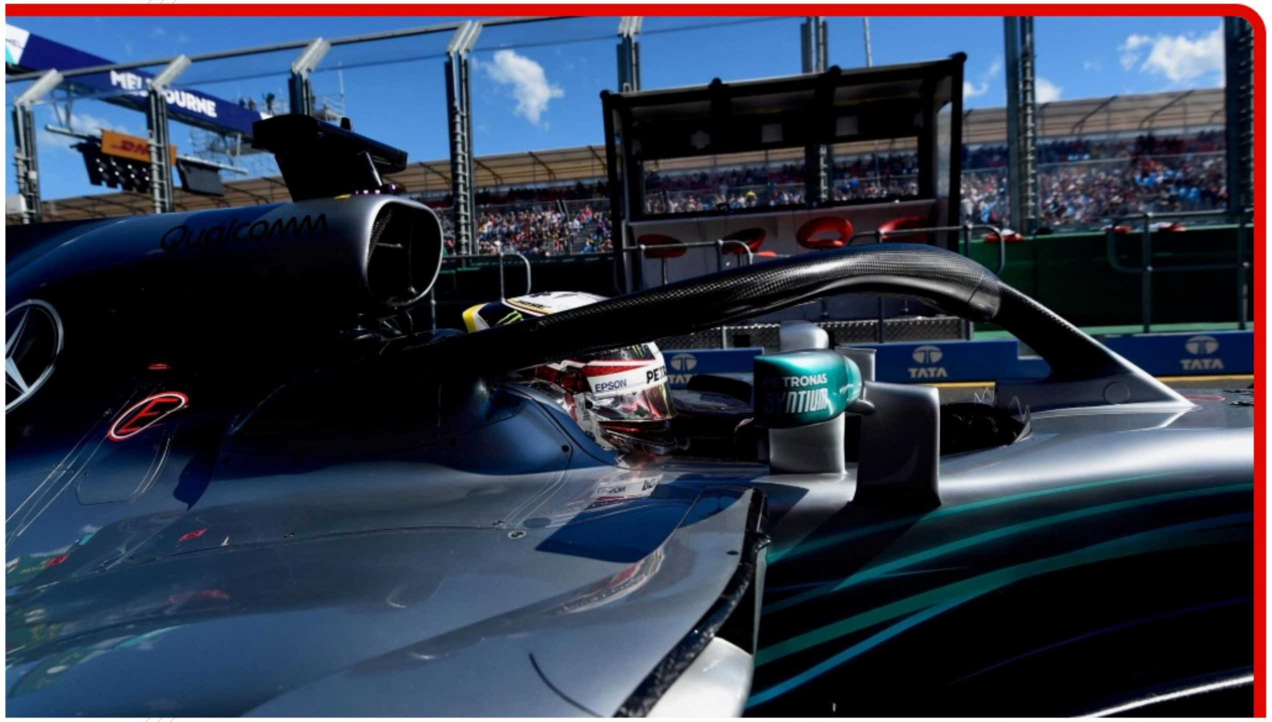
Hamilton was passed by Ferrari's Sebastian Vettel when the German took a pit stop after the virtual safety car had been deployed. Mercedes had not advised Hamilton to open up a greater lead since their data said the gap he held was sufficient. But that information proved incorrect.

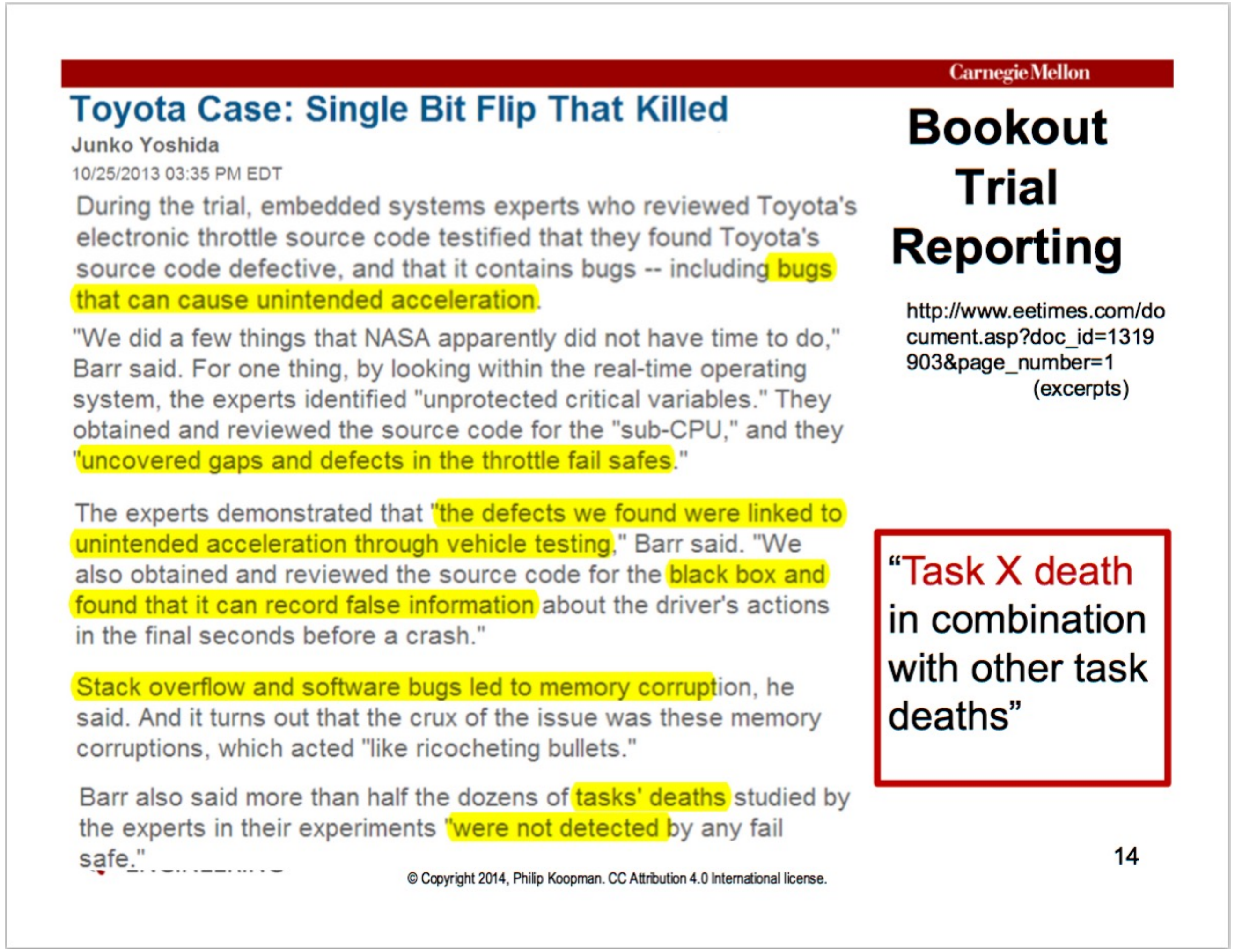
"Definitely," was Hamilton's reply when asked if he would prefer to use his intuition over computer analysis. "Today it is such a team effort but when you are relying on so much data, so much technology to come out with the

Software glitch cost Hamilton victory - Mercedes

25 March 2018

MERCEDES | AUSTRALIA | HAMILTON





Carnegie Mellon

Toyota Case: Single Bit Flip That Killed

Junko Yoshida
10/25/2013 03:35 PM EDT

During the trial, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found Toyota's source code defective, and that it contains bugs -- including bugs that can cause unintended acceleration.

"We did a few things that NASA apparently did not have time to do," Barr said. For one thing, by looking within the real-time operating system, the experts identified "unprotected critical variables." They obtained and reviewed the source code for the "sub-CPU," and they "uncovered gaps and defects in the throttle fail safes."

The experts demonstrated that "the defects we found were linked to unintended acceleration through vehicle testing," Barr said. "We also obtained and reviewed the source code for the black box and found that it can record false information about the driver's actions in the final seconds before a crash."

Stack overflow and software bugs led to memory corruption, he said. And it turns out that the crux of the issue was these memory corruptions, which acted "like ricocheting bullets."

Barr also said more than half the dozens of tasks' deaths studied by the experts in their experiments "were not detected by any fail safe."

Bookout Trial Reporting

http://www.eetimes.com/document.asp?doc_id=1319903&page_number=1
(excerpts)

"Task X death in combination with other task deaths"

14

© Copyright 2014, Philip Koopman. CC Attribution 4.0 International license.

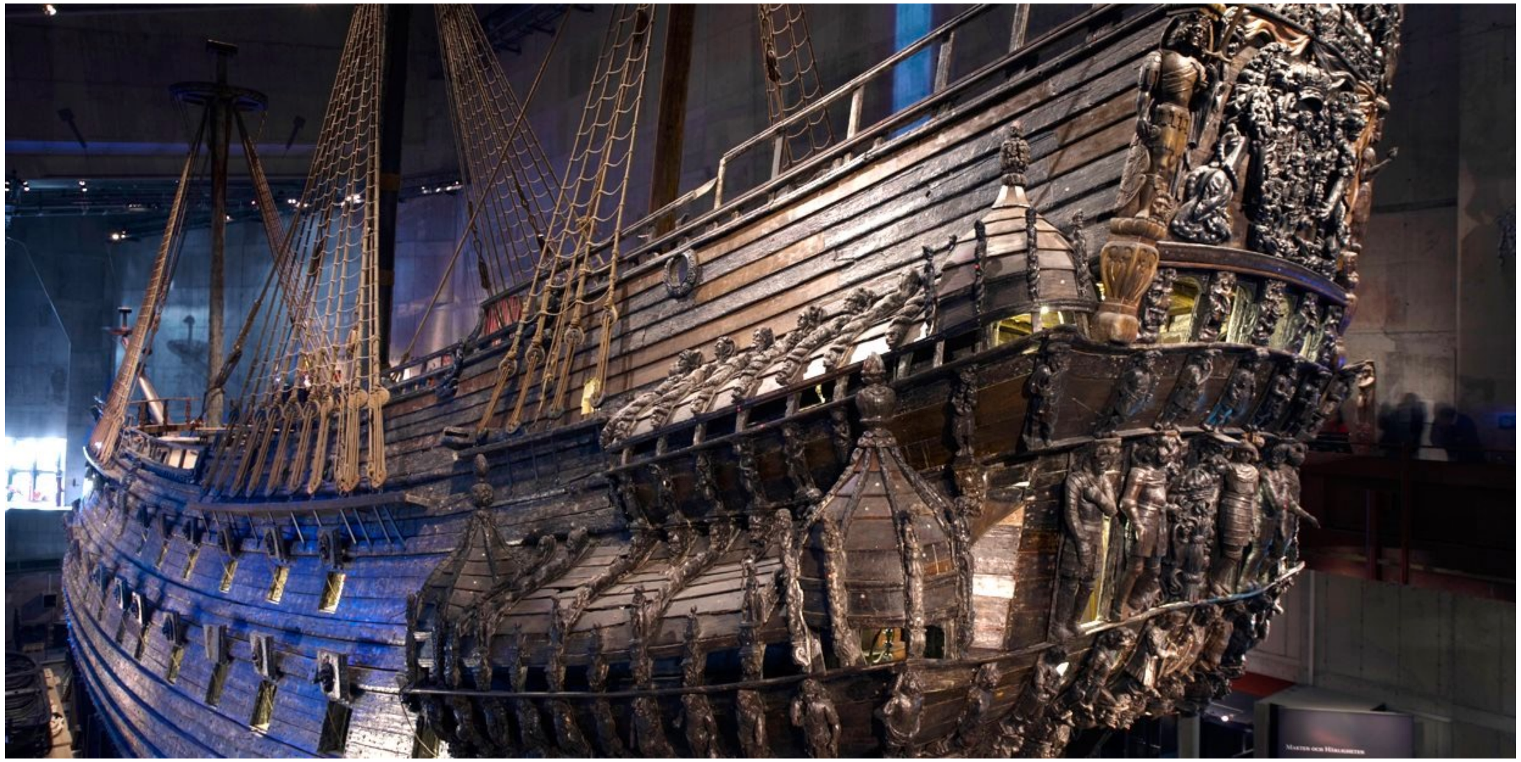
THE VERGE
BOEING
REDLINE
The many human errors that brought down the Boeing 737 Max

17-313
Foundations of
SOFTWARE
ENGINEERING

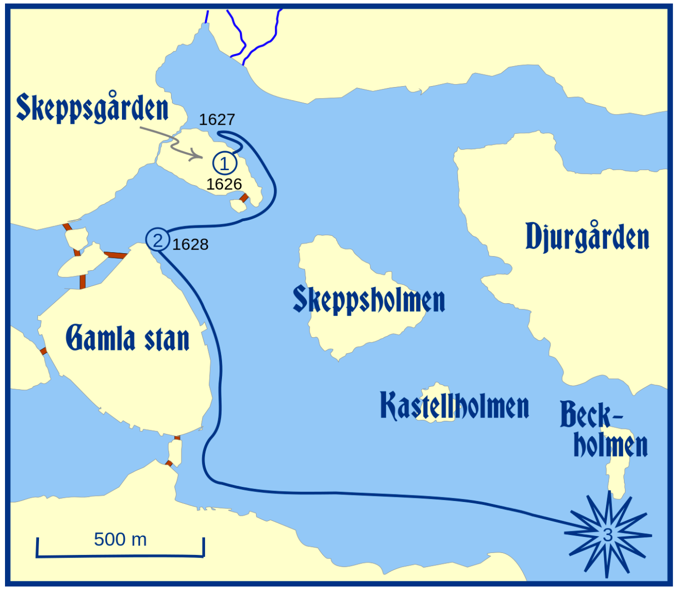
STAND 2
S 37° 00
E 174° 4
DELIVER
LIFE VEST

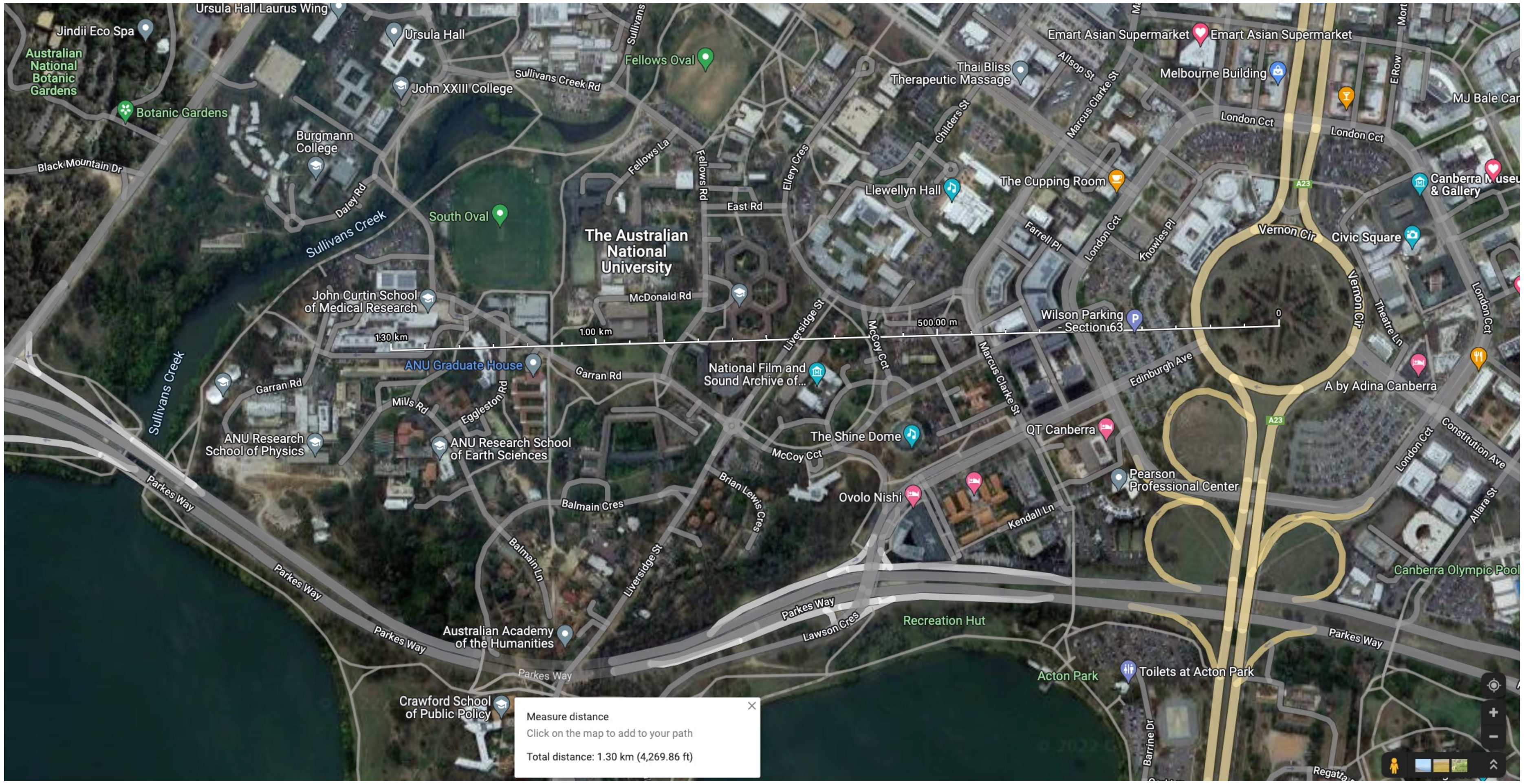
ESCAPE ROPE

Vasa



Vasa


Skeppsgården
1627
1
1626
2
1628
Djurgården
Skeppsholmen
Gamla stan
Kastellholmen
Beck-
holmen
3
500 m



Measure distance

Click on the map to add to your path

Total distance: 1.30 km (4,269.86 ft)

What happened is now called "Vasa syndrome"

- Changing shipbuilding orders
- No specifications for modified keel
- Shifting armaments requirements
- Shipwright's death
- No way to calculate stability, stiffness, or sailing characteristics
- Failed pre-launch stability tests

Requirements

Teams

Metrics

QA

Software *Engineering?*

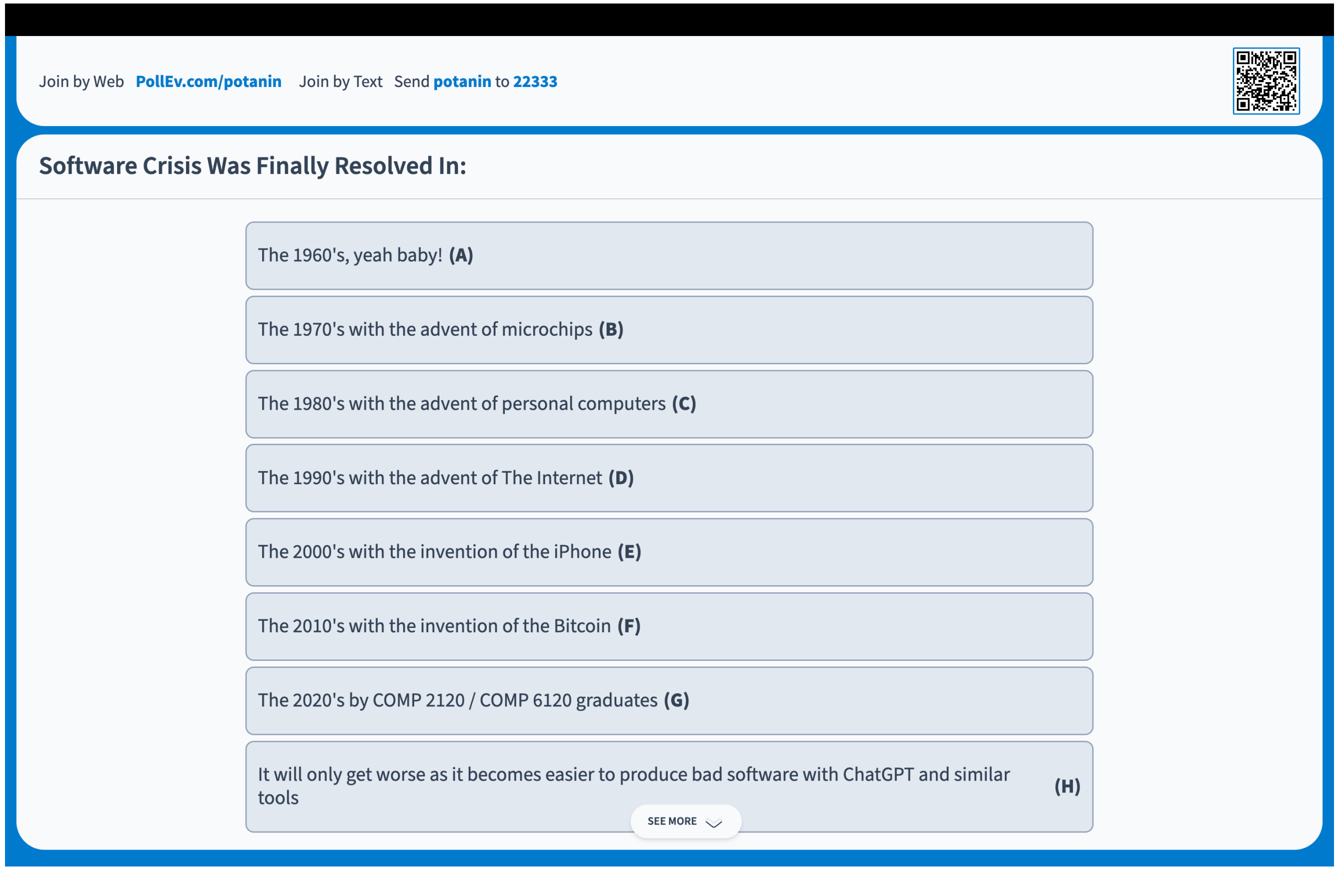
What is **engineering**? And how is it different from **hacking/programming**?

1968 NATO Conference on Software Engineering

- Provocative Title
- Call for Action
- "Software crisis"

Margaret Hamilton

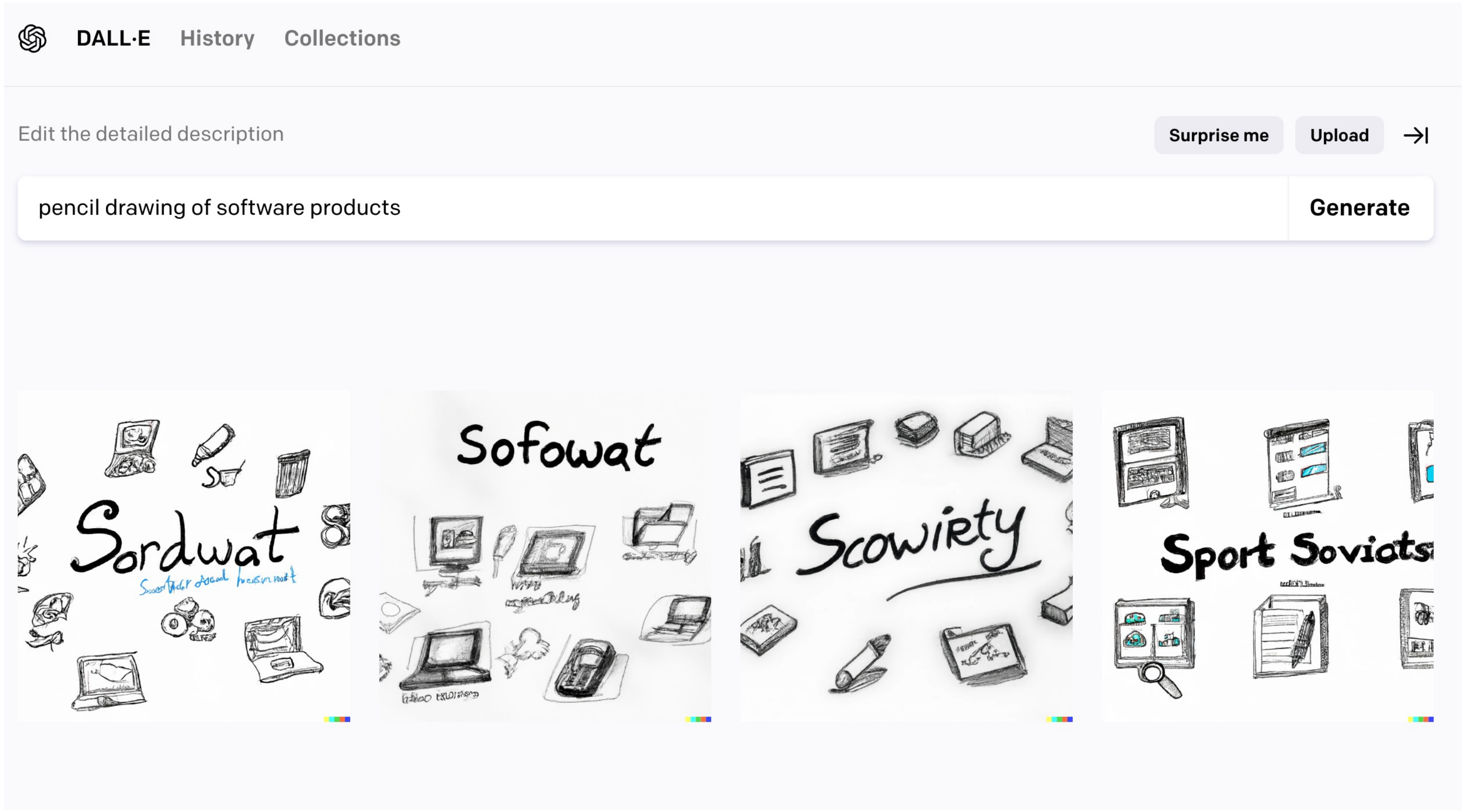
Poll Everywhere Time!



Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

Software Crisis Was Finally Resolved In:

- The 1960's, yeah baby! **(A)**
- The 1970's with the advent of microchips **(B)**
- The 1980's with the advent of personal computers **(C)**
- The 1990's with the advent of The Internet **(D)**
- The 2000's with the invention of the iPhone **(E)**
- The 2010's with the invention of the Bitcoin **(F)**
- The 2020's by COMP 2120 / COMP 6120 graduates **(G)**
- It will only get worse as it becomes easier to produce bad software with ChatGPT and similar tools **(H)**



pencil drawing of software products

Software Products

Software Products

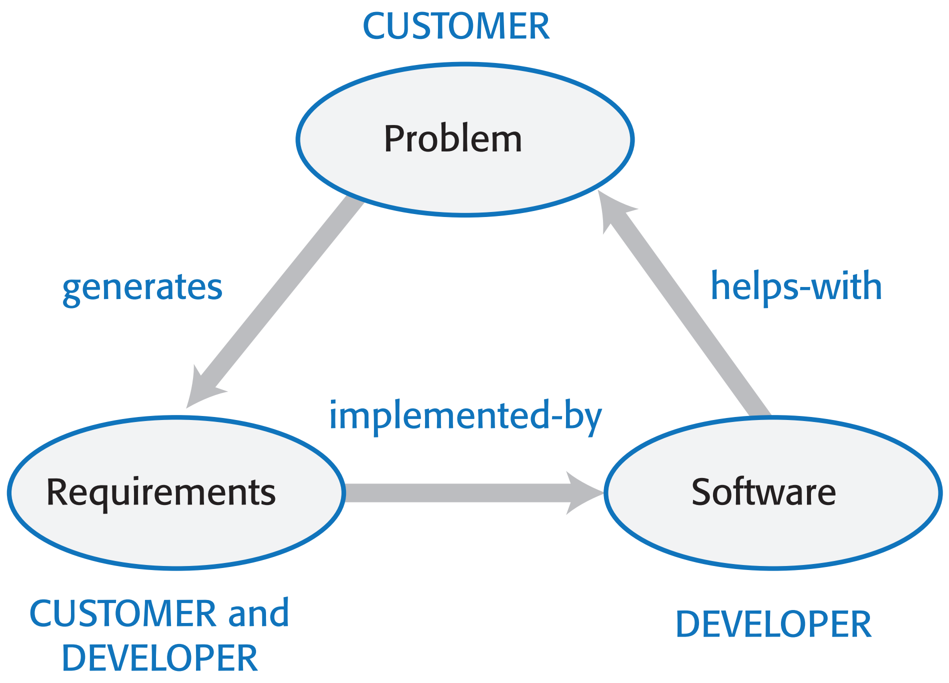
Any slide with ESP Book in the top right corner is Copyright by Ian Sommerville 2018

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Software products are generic software systems that provide functionality that is useful to a range of customers.
- Many different types of products are available from large-scale business systems (e.g. MS Excel) through personal products (e.g. Evernote) to simple mobile phone apps and games (e.g. Sudoku).
- Software product engineering methods and techniques have evolved from software engineering techniques that support the development of one-off, custom software systems.
- Custom software systems are still important for large businesses, government and public bodies. They are developed in dedicated software projects.

Project-Based Software Engineering

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville



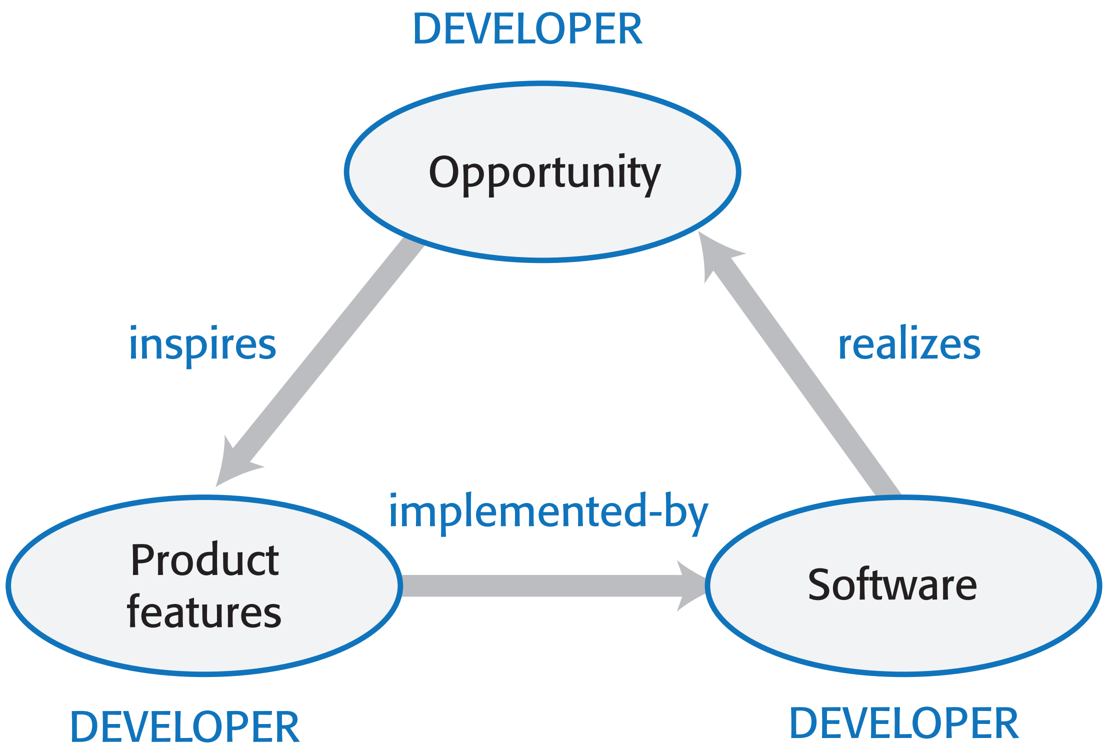
Project-Based Software Engineering

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- The starting point for the software development is a set of 'software requirements' that are owned by an external client and which set out what they want a software system to do to support their business processes.
- The software is developed by a software company (the contractor) who design and implement a system that delivers functionality to meet the requirements.
- The customer may change the requirements at any time in response to business changes (they usually do). The contractor must change the software to reflect these requirements changes.
- Custom software usually has a long-lifetime (10 years or more) and it must be supported over that lifetime.

Product Software Engineering

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville



Product Software Engineering

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- The starting point for product development is a business opportunity that is identified by individuals or a company. They develop a software product to take advantage of this opportunity and sell this to customers.
- The company who identified the opportunity design and implement a set of software features that realize the opportunity and that will be useful to customers.
- The software development company are responsible for deciding on the development timescale, what features to include and when the product should change.
- Rapid delivery of software products is essential to capture the market for that type of product.

Software Product Lines & Platforms

Software product line
A set of software products that share a common core. Each member of the product line includes customer-specific adaptations and additions. Software product lines may be used to implement a custom system for a customer with specific needs that can't be met by a generic product.

Platform
A software (or software+hardware) product that includes functionality so that new applications can be built on it. An example of a platform that you probably use is Facebook. It provides an extensive set of product functionality but also provides support for creating 'Facebook apps'. These add new features that may be used by a business or a Facebook interest group.

Software Execution Models

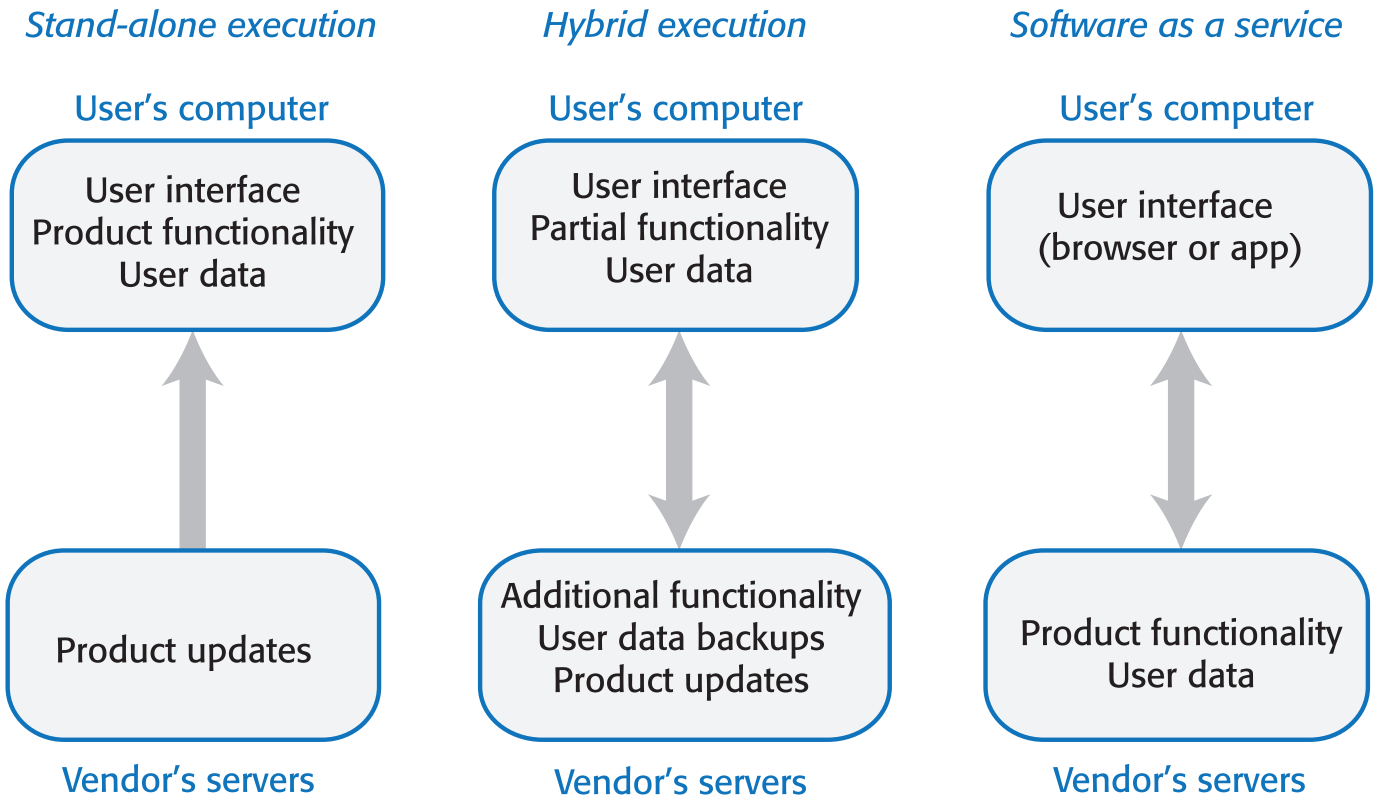
- **Stand-alone** The software executes entirely on the customer's computers.

- **Hybrid** Part of the software's functionality is implemented on the customer's computer but some features are implemented on the product developer's servers.

- **Software service** All of the product's features are implemented on the developer's servers and the customer accesses these through a browser or a mobile app.

Software Execution Models

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION



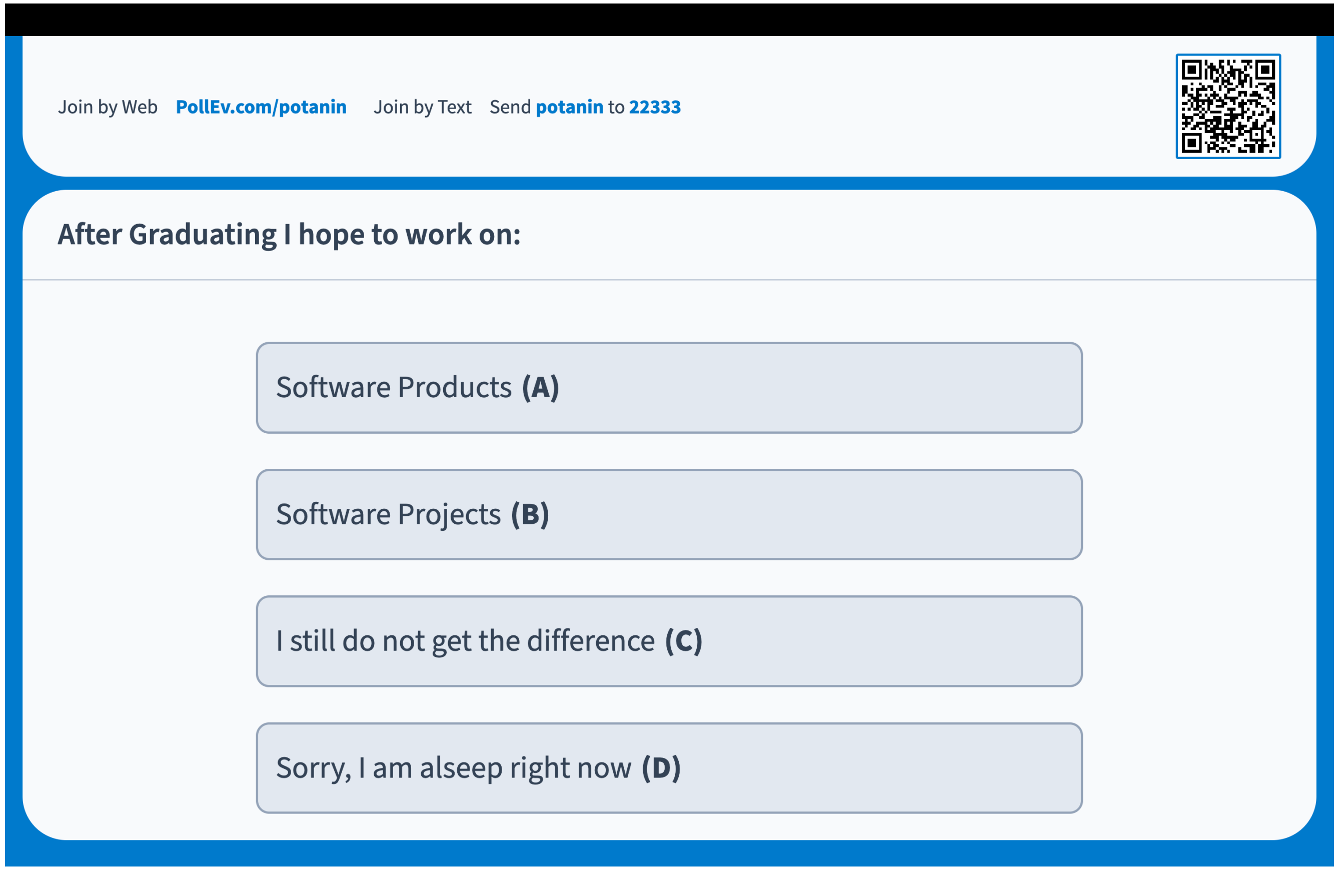
Comparable Software Development

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

The key feature of product development is that there is no external customer that generates requirements and pays for the software. This is also true for other types of software development:

- **Student projects** Individuals or student groups develop software as part of their course. Given an assignment, they decide what features to include in the software.
- **Research software** Researchers develop software to help them answer questions that are relevant to their research.
- **Internal tool development** Software developers may develop tools to support their work - in essence, these are internal products that are not intended for customer release.

Poll Everywhere Time!



Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

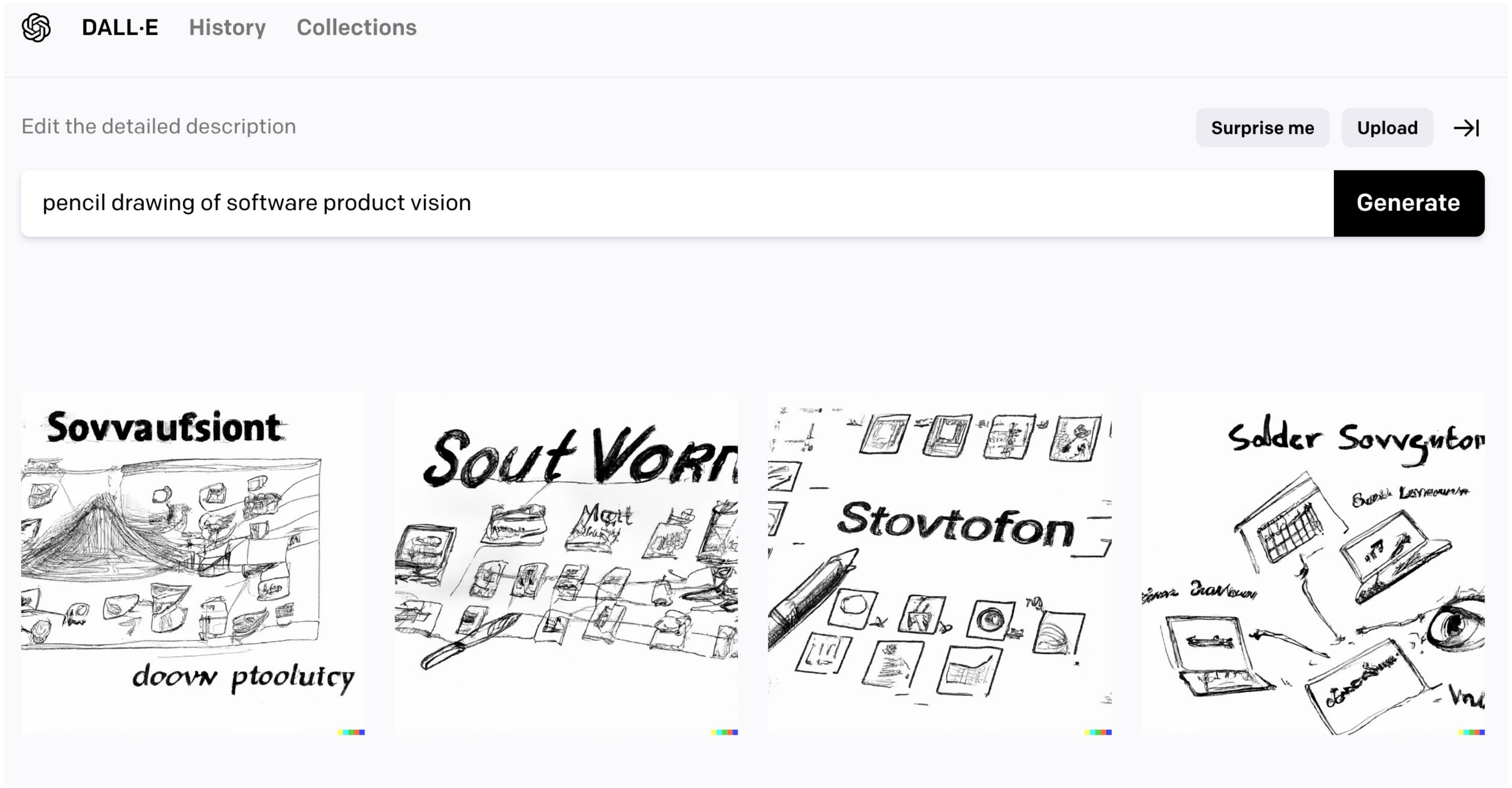
After Graduating I hope to work on:

Software Products **(A)**

Software Projects **(B)**

I still do not get the difference **(C)**

Sorry, I am alseep right now **(D)**



pencil drawing of software product version

Product Vision and Product Management

The Product Vision

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- The starting point for software product development is a 'product vision'.
- Product visions are simple statements that define the essence of the product to be developed.
- The product vision should answer three fundamental questions:
  - What is the product to be developed?
  - Who are the target customers and users?
  - Why should customers buy this product?

Moore’s Vision Template

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- FOR (target customer)
- WHO (statement of the need or opportunity)
- The (PRODUCT NAME) is a (product category)
- THAT (key benefit, compelling reason to buy)
- UNLIKE (primary competitive alternative)
- OUR PRODUCT (statement of primary differentiation)

Vision Template Example

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION

“FOR a mid-sized company's marketing and sales departments WHO need basic CRM functionality, THE CRM-Innovator is a Web-based service THAT provides sales tracking, lead generation, and sales representative support features that improve customer relationships at critical touch points. UNLIKE other services or package software products, OUR product provides very capable services at a moderate cost.”

Developing Product Vision

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Domain experience
The product developers may work in a particular area (say marketing and sales) and understand the software support that they need. They may be frustrated by the deficiencies in the software they use and see opportunities for an improved system.

Product experience
Users of existing software (such as word processing software) may see simpler and better ways of providing comparable functionality and propose a new system that implements this. New products can take advantage of recent technological developments such as speech interfaces.

Developing Product Vision

Customer experience
The software developers may have extensive discussions with prospective customers of the product to understand the problems that they face, constraints, such as interoperability, that limit their flexibility to buy new software, and the critical attributes of the software that they need.

Prototyping and playing around
Developers may have an idea for software but need to develop a better understanding of that idea and what might be involved in developing it into a product. They may develop a prototype system as an experiment and 'play around' with ideas and variations using that prototype system as a platform.

Vision Statement for the iLearn (ESP Book Running Example)

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

FOR teachers and educators WHO need a way to help students use web-based learning resources and applications, THE iLearn system is an open learning environment THAT allows the set of resources used by classes and students to be easily configured for these students and classes by teachers themselves. UNLIKE Virtual Learning Environments, such as Moodle, the focus of iLearn is the learning process rather than the administration and management of materials, assessments and coursework. OUR product enables teachers to create subject and age-specific environments for their students using any web-based resources, such as videos, simulations and written materials that are appropriate.

Schools and universities are the target customers for the iLearn system as it will significantly improve the learning experience of students at relatively low cost. It will collect and process learner analytics that will reduce the costs of progress tracking and reporting.

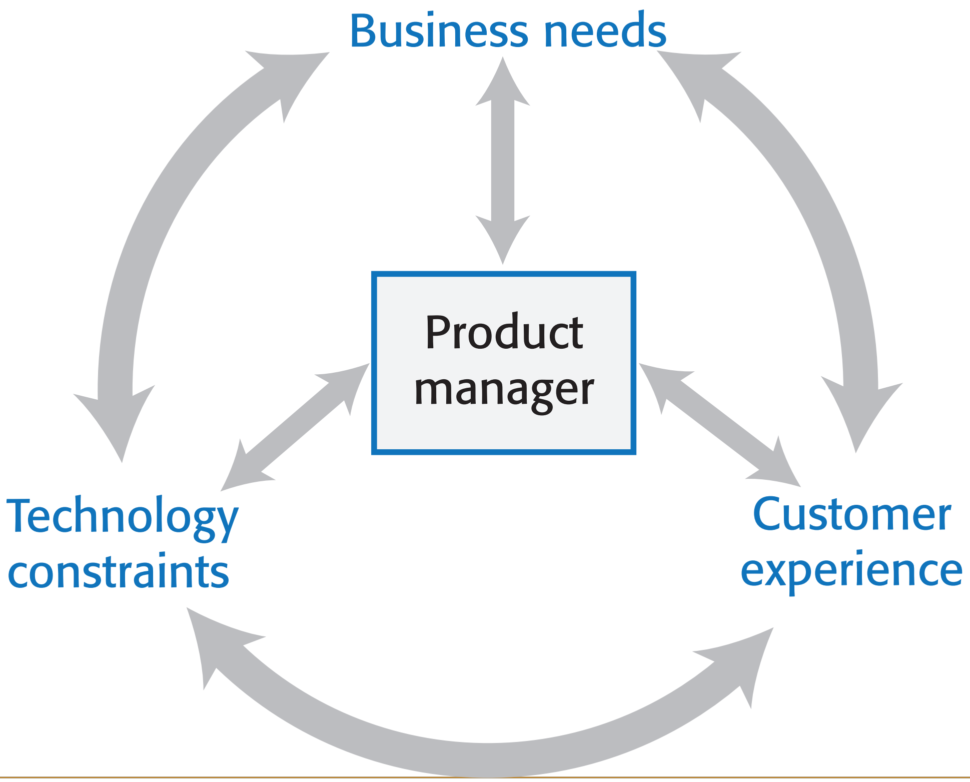
Software Product Management

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Software product management is a business activity that focuses on the software products developed and sold by the business.
- Product managers (PMs) take overall responsibility for the product and are involved in planning, development and product marketing.
- Product managers are the interface between the organization, its customers and the software development team. They are involved at all stages of a product’s lifetime from initial conception through to withdrawal of the product from the market.
- Product managers must look outward to customers and potential customers rather than focus on the software being developed.

Product Management Concerns

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION



Product Management Concerns

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

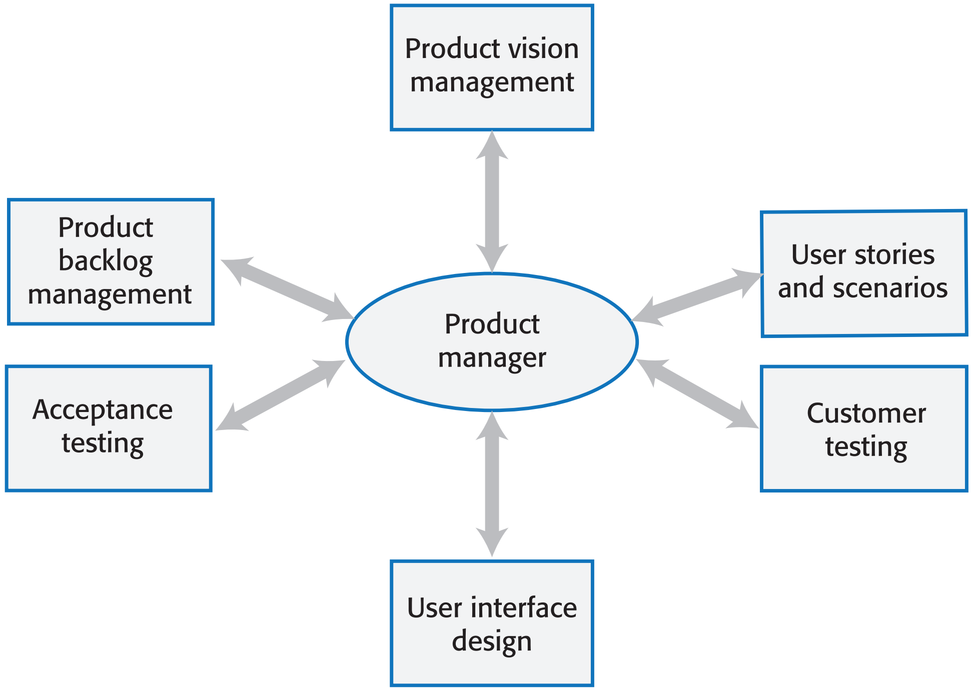
- **Business needs** PMs have to ensure that the software being developed meets the business goals of the software development company.

- **Technology constraints** PMs must make developers aware of technology issues that are important to customers.

- **Customer experience** PMs should be in regular contact with customers and potential customers to understand what they are looking for in a product, the types of users and their backgrounds and the ways that the product may be used.

Technical Interactions of Product Managers

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville


Product vision management
Product backlog management
User stories and scenarios
Product manager
Acceptance testing
Customer testing
User interface design

Technical Interactions of Product Managers

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Product vision management
  - The product manager may be responsible for helping with the development of the product vision. The should always be responsible for managing the vision, which involves assessing and evaluating proposed changes against the product vision. They should ensure that there is no ‘vision drift’
- Product roadmap development
  - A product roadmap is a plan for the development, release and marketing of the software. The PM should lead roadmap development and should be the ultimate authority in deciding if changes to the roadmap should be made.

Technical Interactions of Product Managers

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION

- User story and scenario development
  - User stories and scenarios are used to refine a product vision and identify product features. Based on his or her knowledge of customers, the PM should lead the development of stories and scenarios.
- Product backlog creation and management
  - The product backlog is a prioritized 'to-do' list of what has to be developed. PMs should be involved in creating and refining the backlog and deciding on the priority of product features to be developed.
- Acceptance testing
  - Acceptance testing is the process of verifying that a software release meets the goals set out in the product roadmap and that the product is efficient and reliable. The PM should be involved in developing tests of the product features that reflect how customers use the product.

Technical Interactions of Product Managers

UNPREPARED
TO DO
ANALYSIS
DEV
GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Customer testing
  - Customer testing involves taking a release of a product to customers and getting feedback on the product's features, usability and business. PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.
- User interface design
  - Product managers should understand user limitations and act as surrogate users in their interactions with the development team. They should evaluate user interface features as they are developed to check that these features are not unnecessarily complex or force users to work in an unnatural way.

Product Prototyping

- Product prototyping is the process of developing an early version of a product to test your ideas and to convince yourself and company funders that your product has real market potential.
  - You may be able to write an inspiring product vision, but your potential users can only really relate to your product when they see a working version of your software. They can point out what they like and don't like about it and make suggestions for new features.
  - A prototype may be also used to help identify fundamental software components or services and to test technology.
- Building a prototype should be the first thing that you do when developing a software product. Your aim should be to have a working version of your software that can be used to demonstrate its key features.
- Plan to throw-away the prototype after development and to re-implement the software, taking account of issues such as security and reliability.

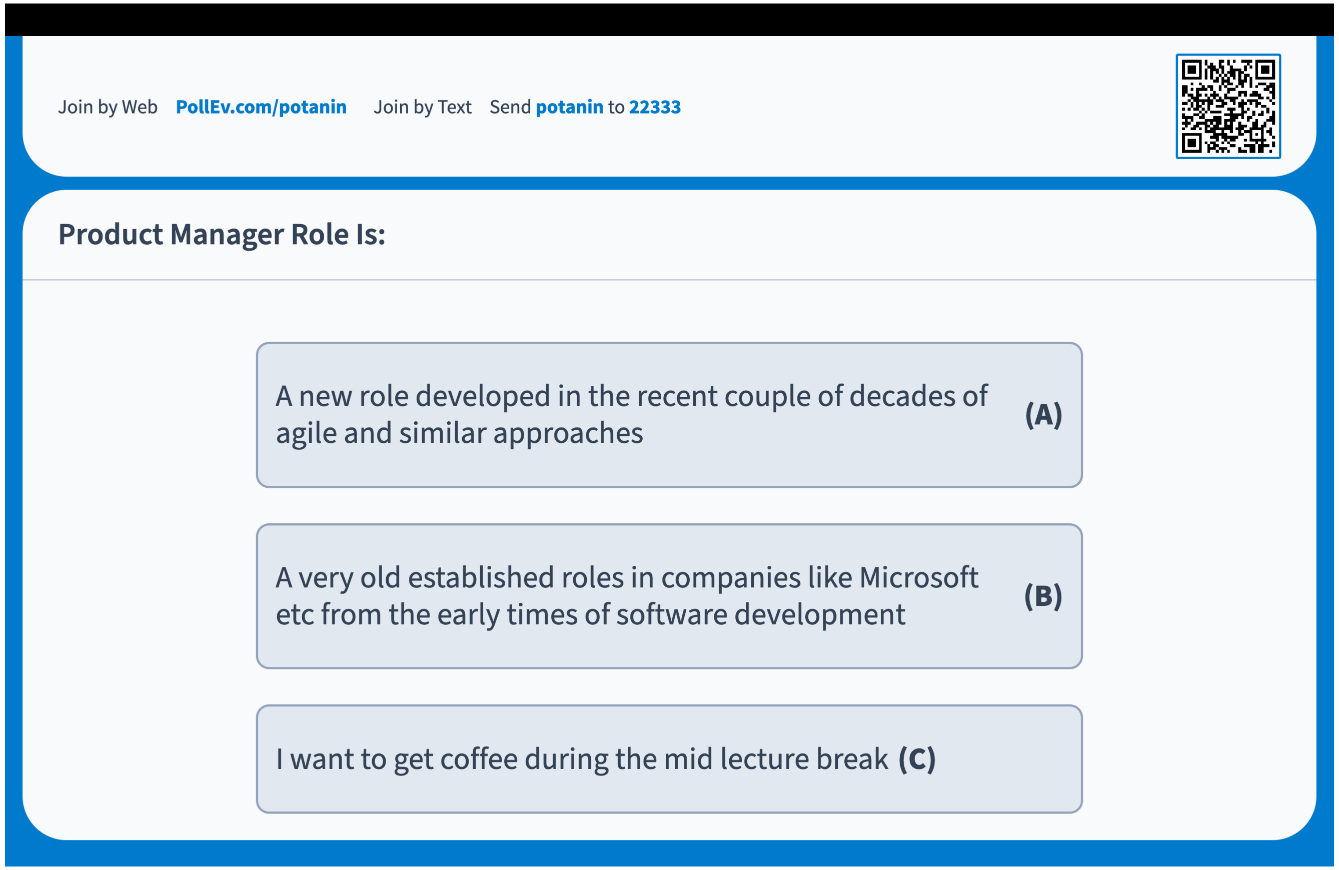
Two-Stage Prototyping

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- **Feasibility demonstration** You create an executable system that demonstrates the new ideas in your product. The aims at this stage are to see if your ideas actually work and to show funders and/or company management the original product features that are better than those in competing products.

- **Customer demonstration** You take an existing prototype created to demonstrate feasibility and extend this with your ideas for specific customer features and how these can be realized. Before you develop this type of prototype, you need to do some user studies and have a clearer idea of your potential users and scenarios of use.

Poll Everywhere Time!



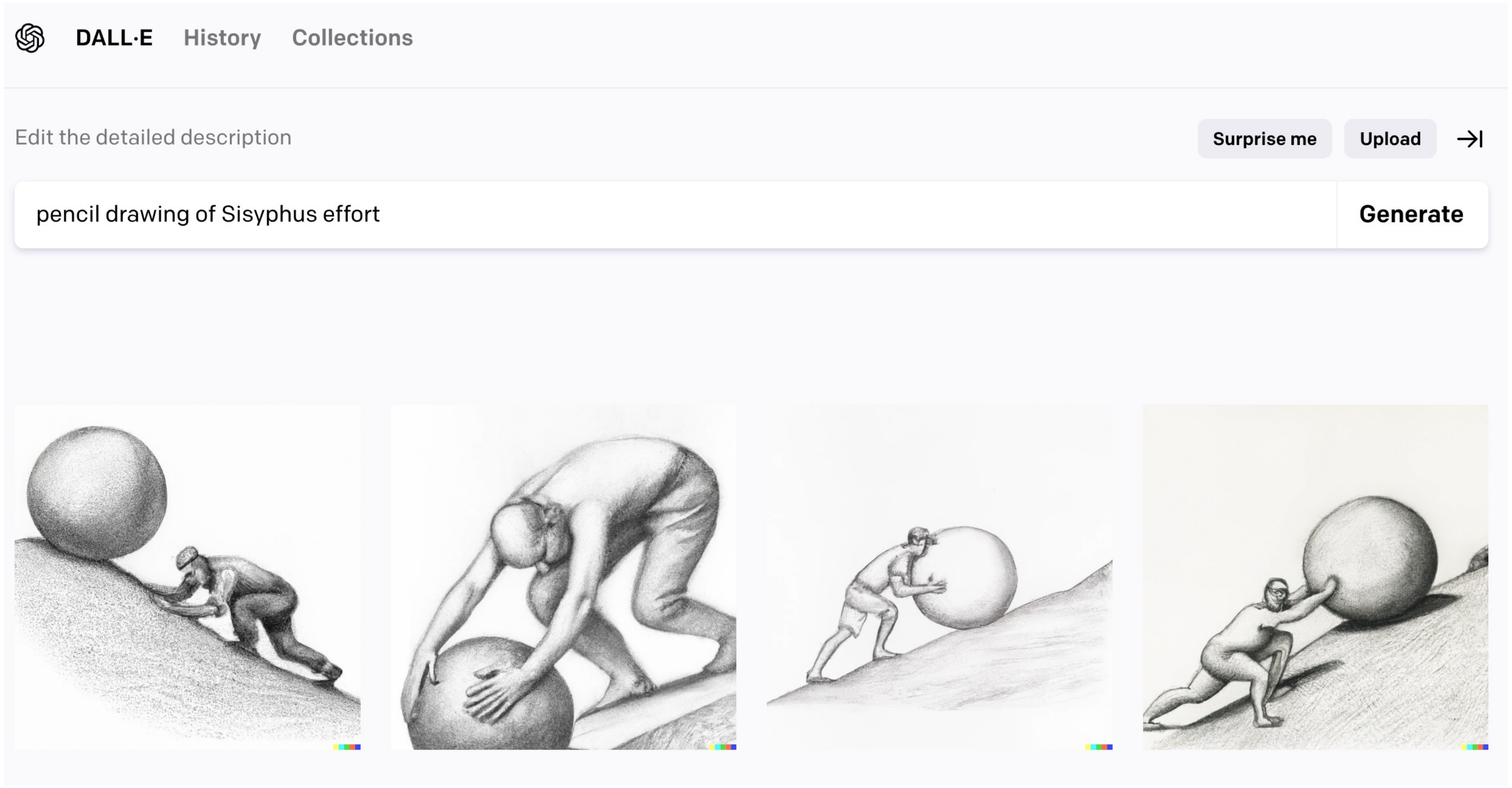
Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

Product Manager Role Is:

A new role developed in the recent couple of decades of agile and similar approaches **(A)**

A very old established roles in companies like Microsoft etc from the early times of software development **(B)**

I want to get coffee during the mid lecture break **(C)**



pencil drawing of Sisyphus effort

Estimating Effort

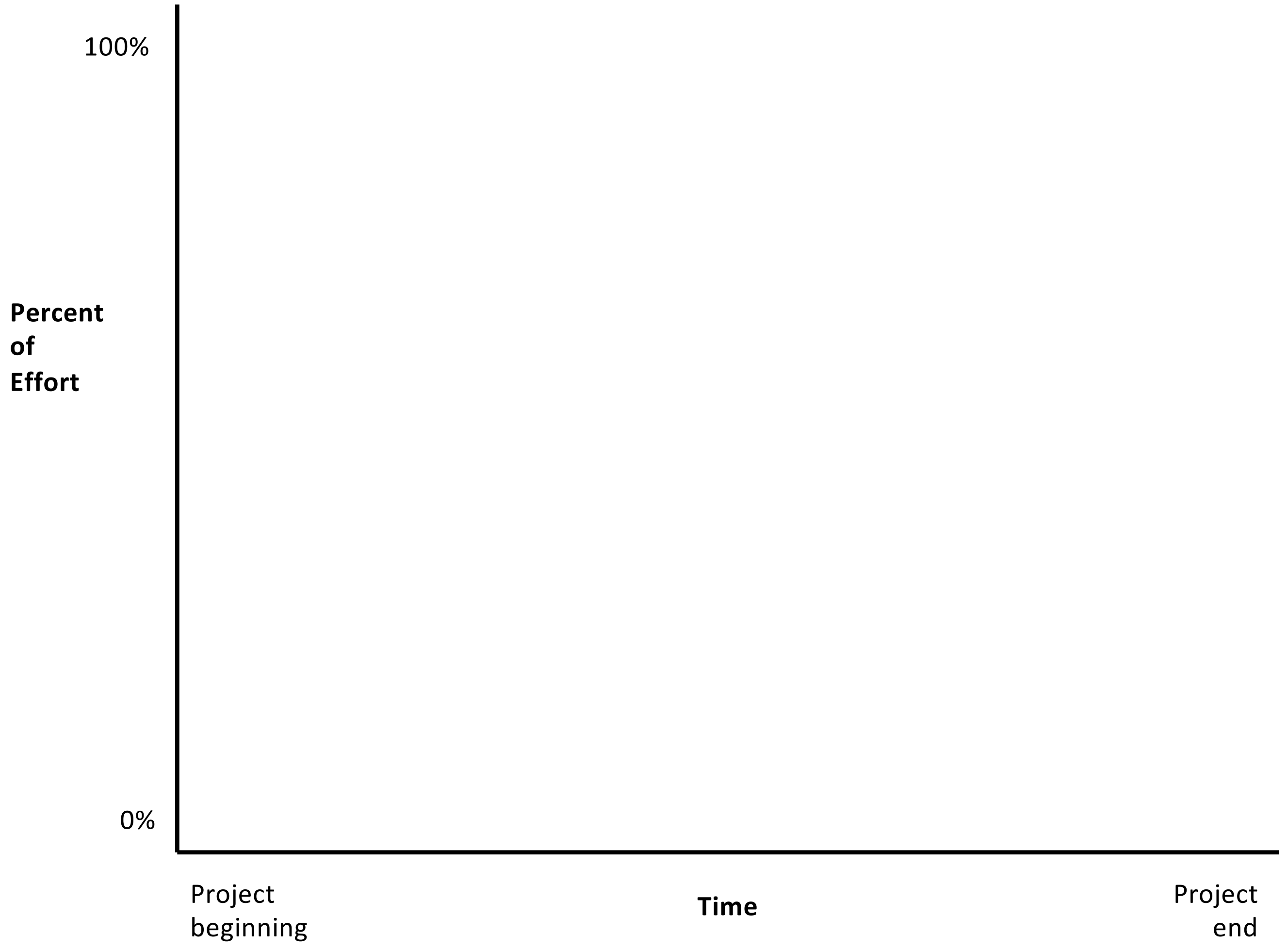
Software Process

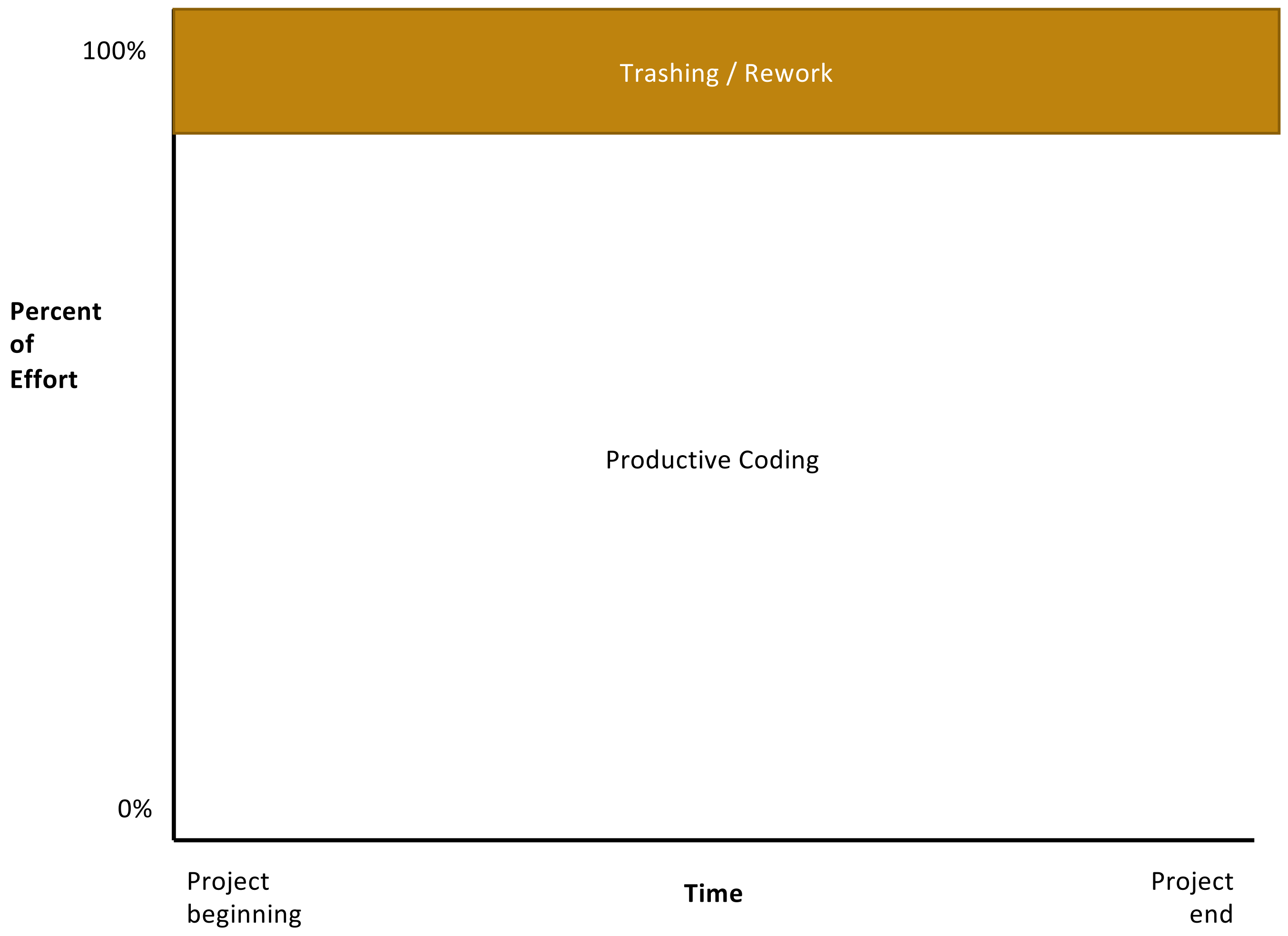
"The set of activities and associated results that produce a software product"

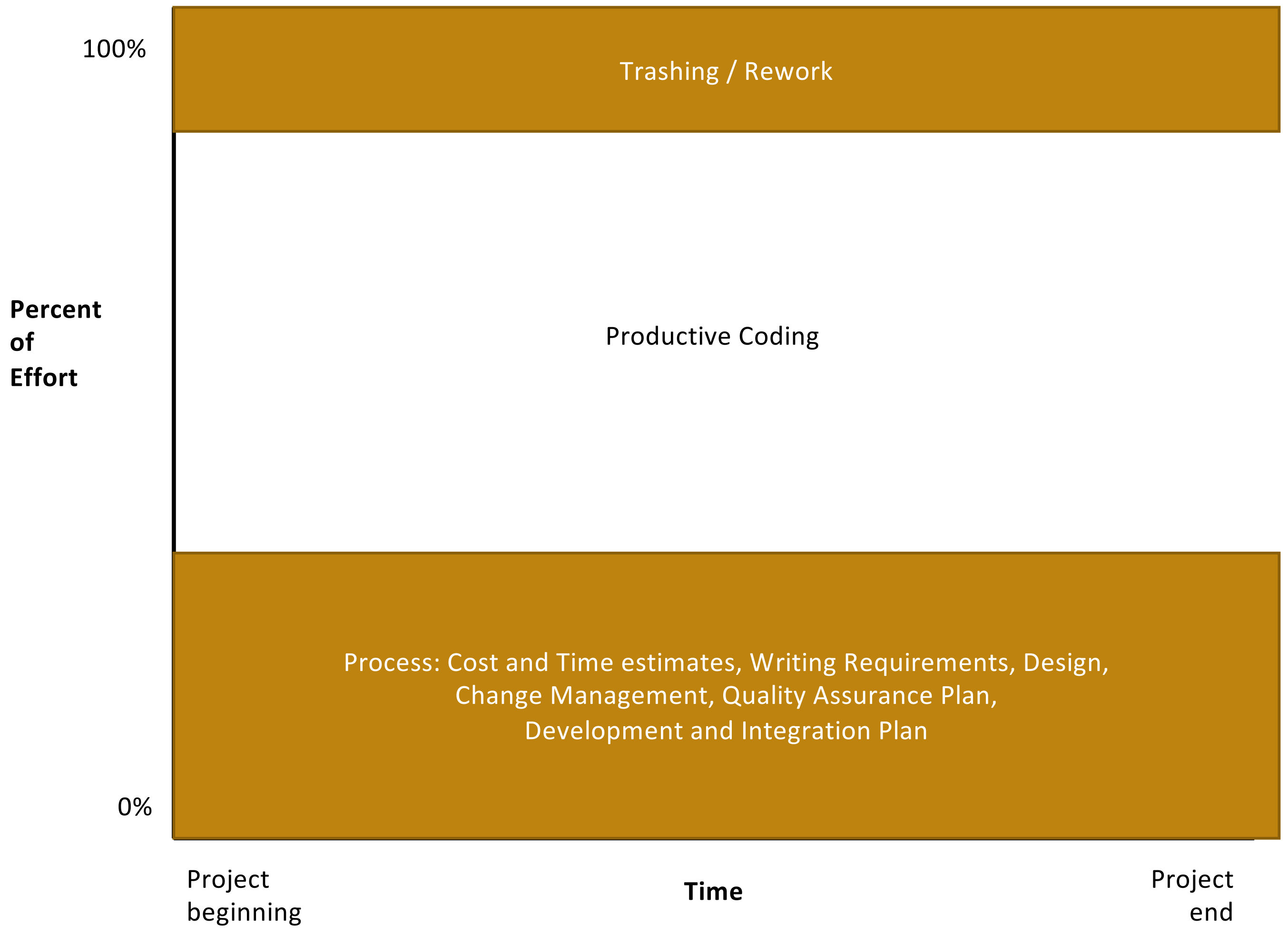
Sommerville, SE, ed. 8

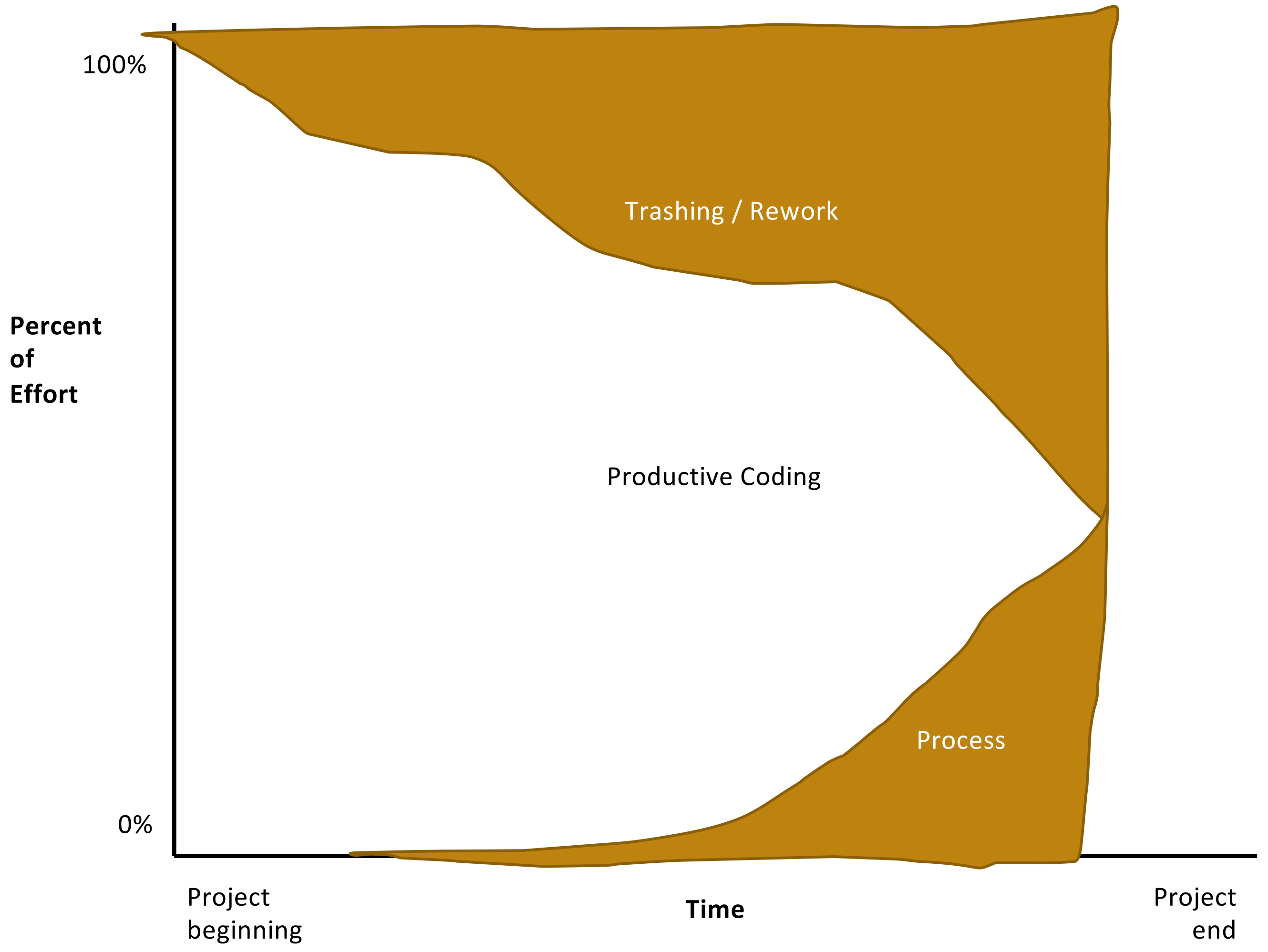
How to develop software?

1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1


100%
Percent of Effort
0%
Project beginning
Time
Project end


100%
Trashing / Rework
Percent of Effort
Productive Coding
0%
Project beginning
Time
Project end


100%
Trashing / Rework
Percent of Effort
Productive Coding
Process: Cost and Time estimates, Writing Requirements, Design, Change Management, Quality Assurance Plan, Development and Integration Plan
0%
Project beginning
Time
Project end

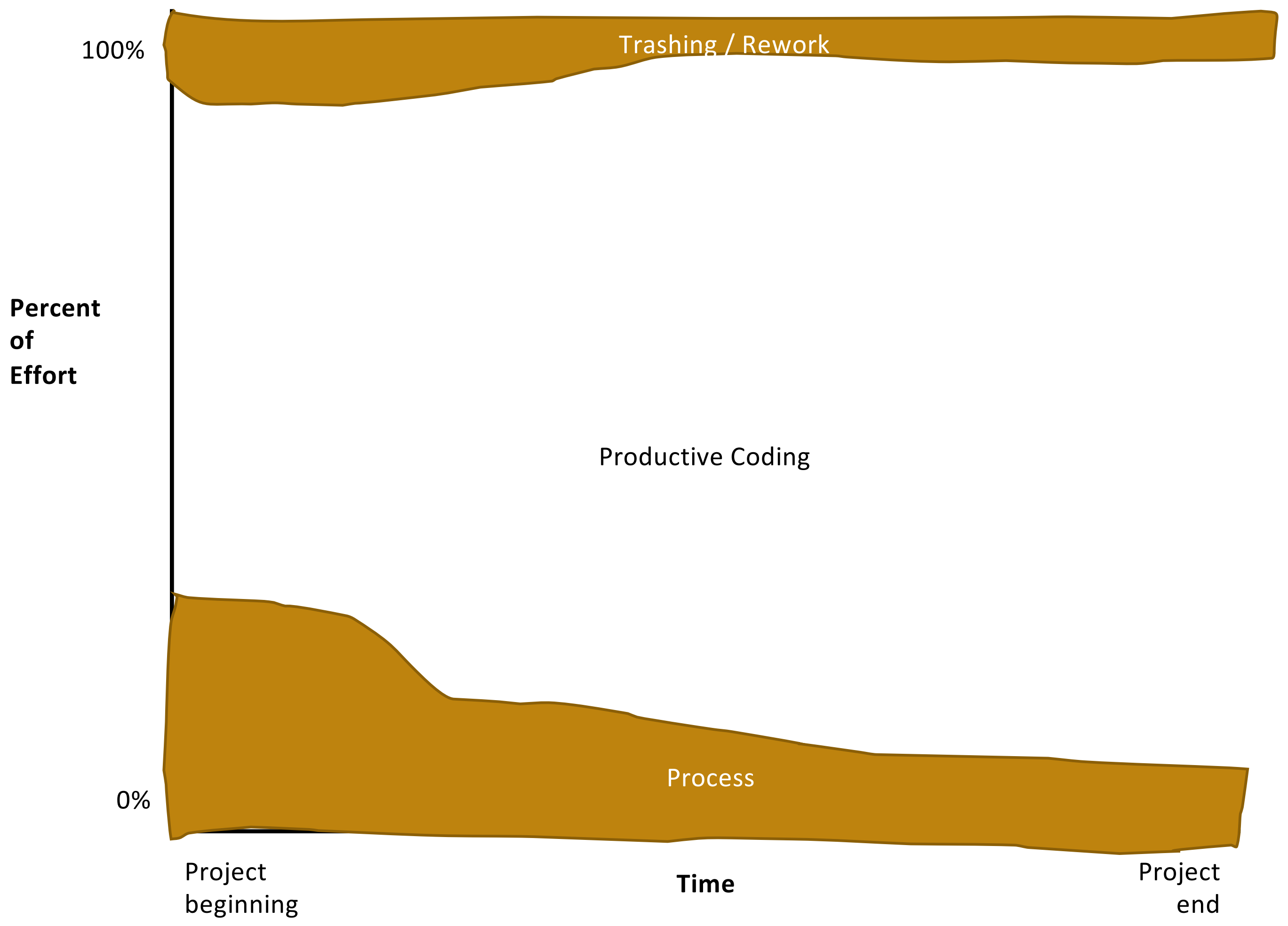

100%
0%
Percent of Effort
Trashing / Rework
Productive Coding
Process
Project beginning
Time
Project end

Example of Process Decisions

- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller tasks and schedule and monitor them
- Planning and conducting quality assurance
- Have daily status meetings
- Use Docker containers to push code between developers and operation

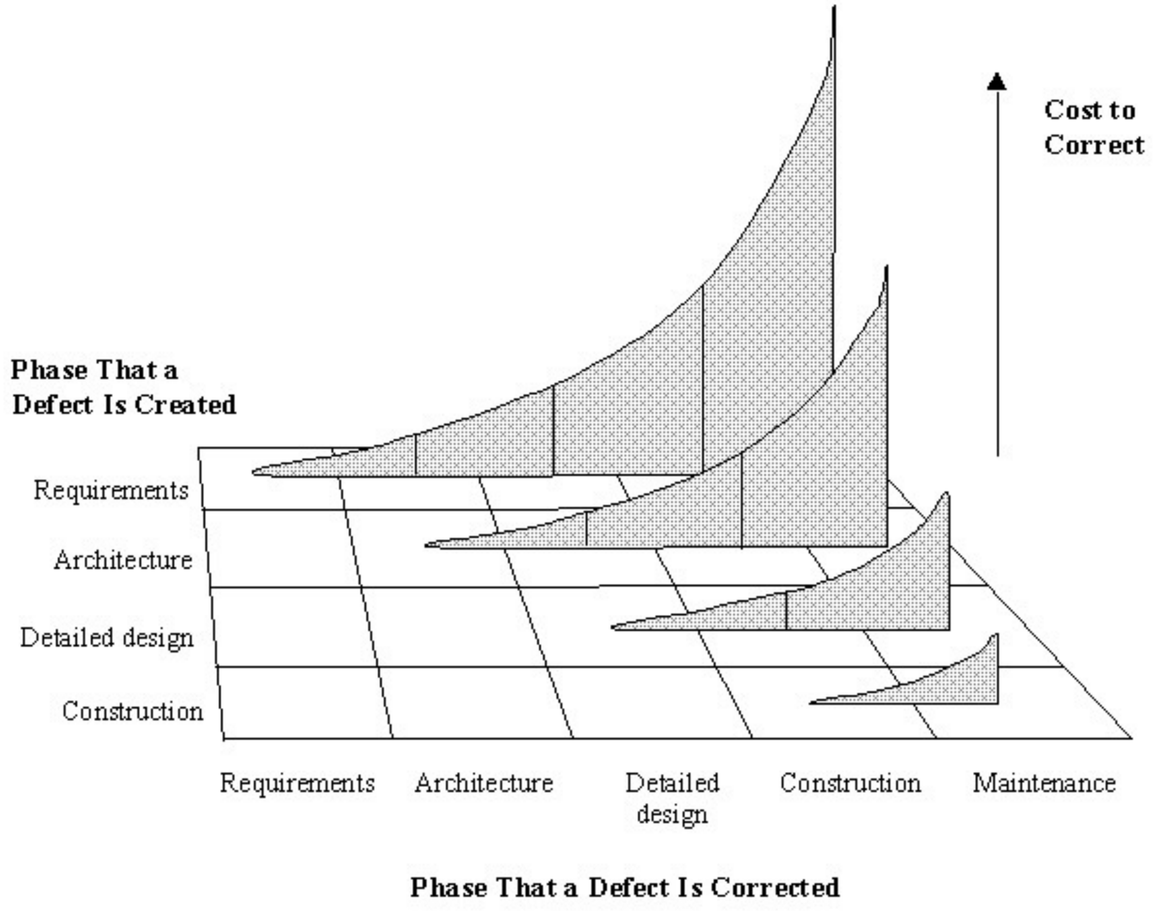
Example process issues

- Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
- Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- Defect Tracking: Bug reports collected informally, forgotten
- System Integration: Integration of independently developed components at the very end of the project. Interfaces out of sync.
- Source Code Control: Accidentally overwritten changes, lost work.
- Scheduling: When project is behind, developers are asked weekly for new estimates.


100%
Trashing / Rework
Percent of Effort
Productive Coding
Process
0%
Project beginning
Time
Project end

Hypothesis

- Process increases flexibility and efficiency
- Upfront investment for later greater returns


Cost to Correct
Phase That a Defect Is Created
Requirements
Architecture
Detailed design
Construction
Requirements
Architecture
Detailed design
Construction
Maintenance
Phase That a Defect Is Corrected

17-313
Foundations of
SOFTWARE
ENGINEERING

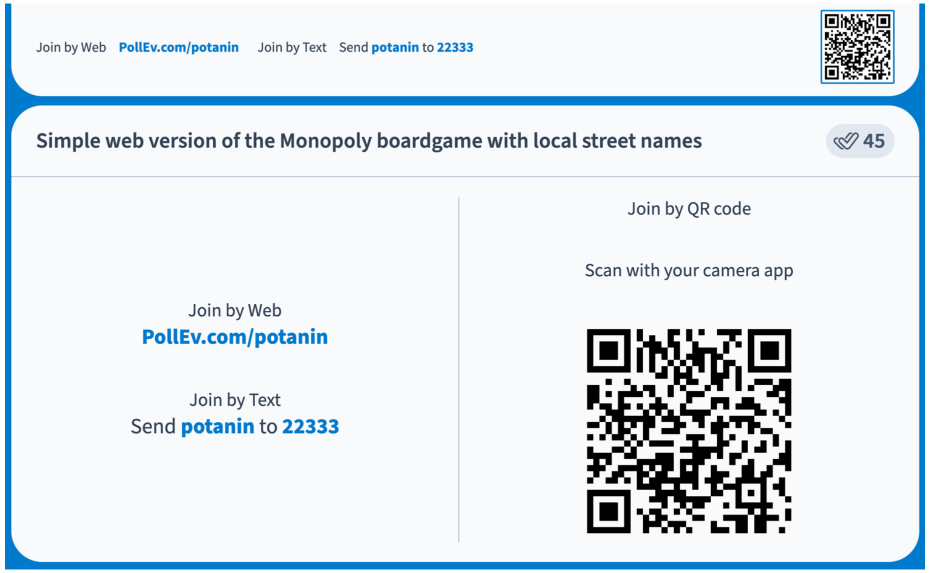
Task: Estimate Time

- A: Simple web version of the Monopoly boardgame with local street names
  - Team: just you
- B: Bank smartphone app
  - Team: you with team of 4 developers, one experienced with iPhone apps, one with background in security

- Estimate in 8h days (20 work days in a month, 220 per year)

Poll Everywhere (To Show)

Poll Everywhere Time!


Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

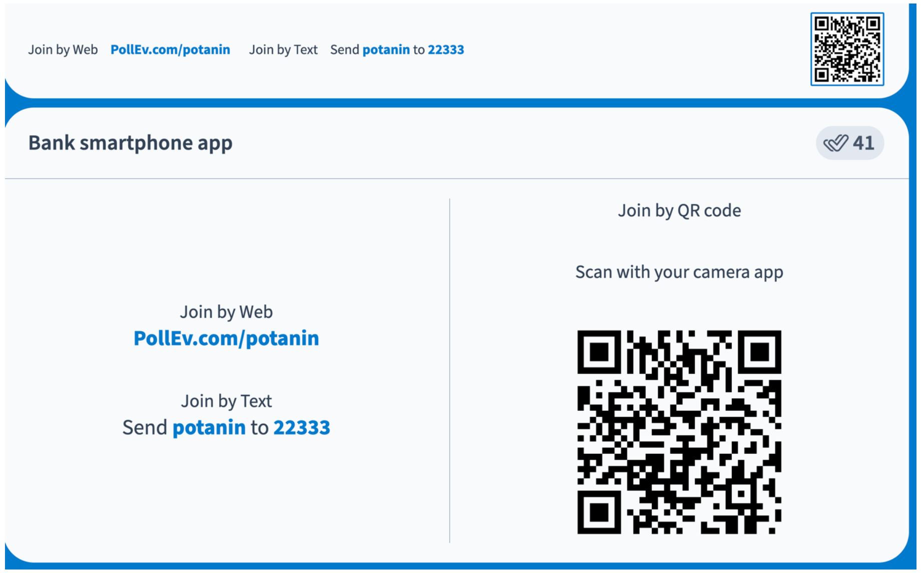
Simple web version of the Monopoly boardgame with local street names 45

Join by Web
PollEv.com/potanin

Join by Text
Send **potanin** to **22333**

Join by QR code

Scan with your camera app


Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

Bank smartphone app 41

Join by Web
PollEv.com/potanin

Join by Text
Send **potanin** to **22333**

Join by QR code

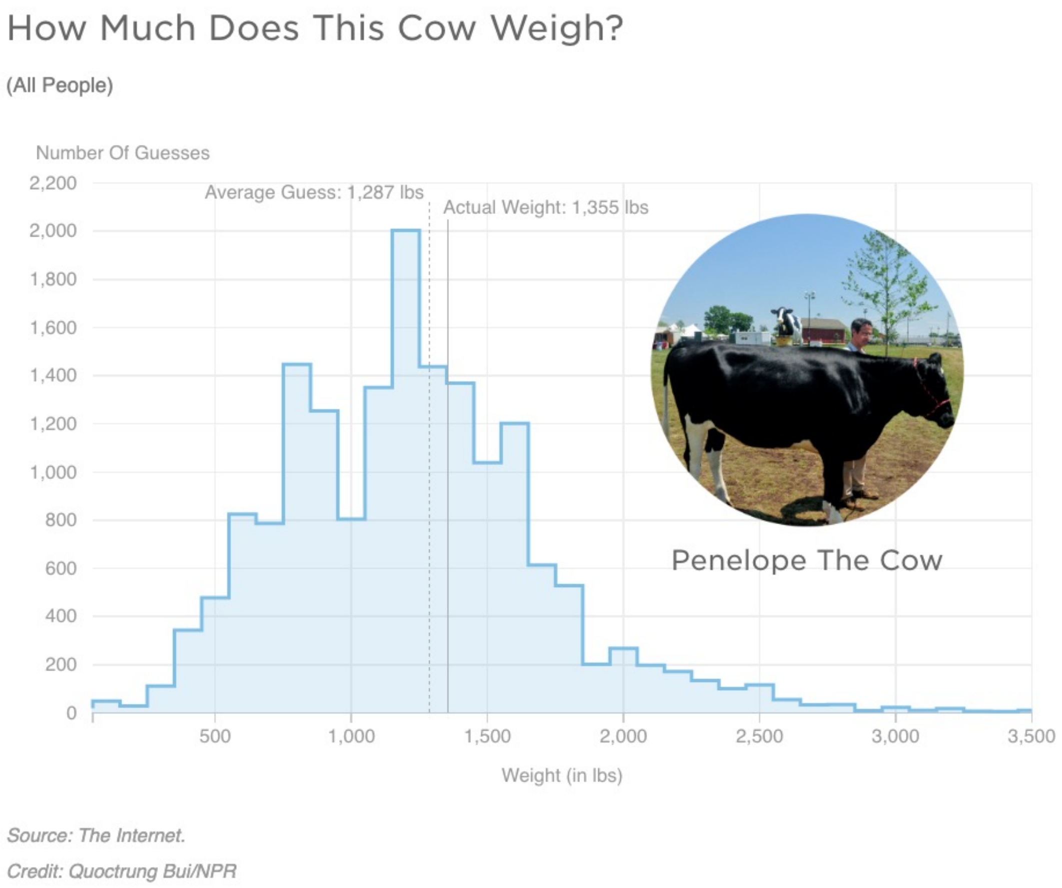
Scan with your camera app

Revise Time Estimate

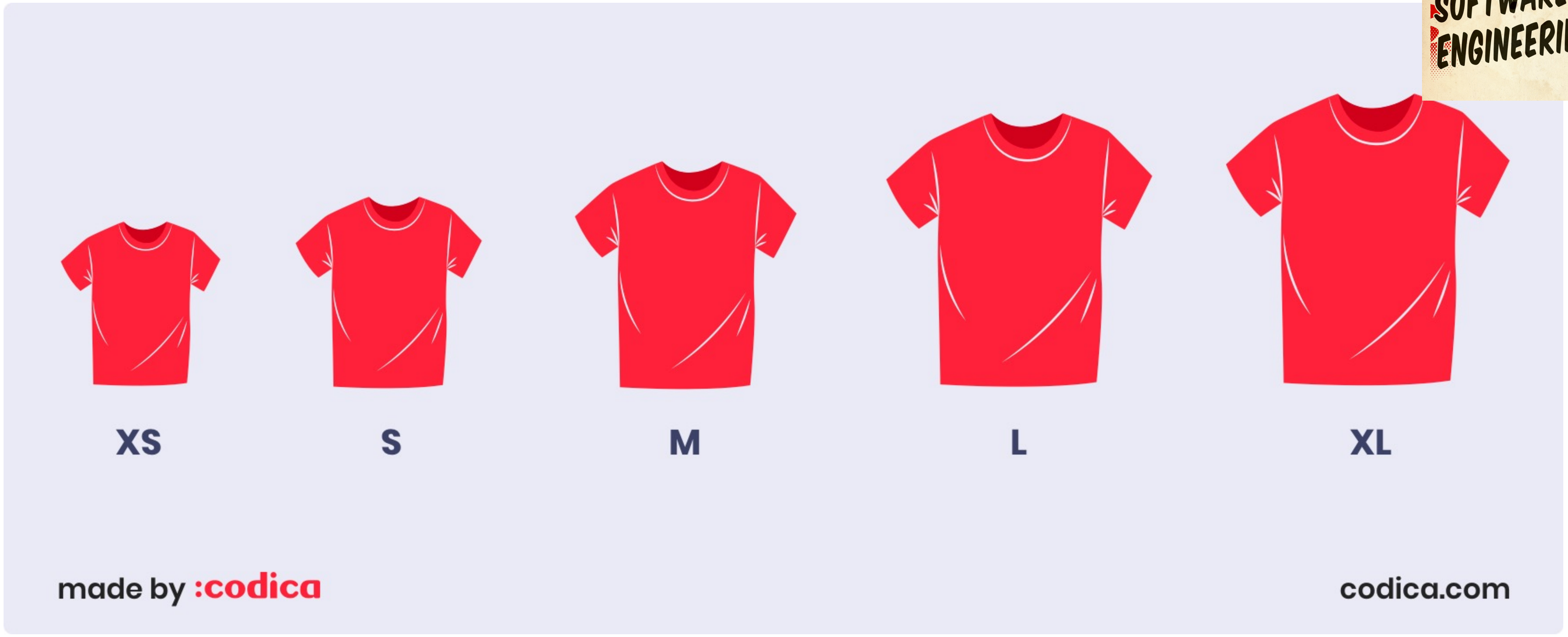
- Do you have comparable experience to base an estimate off of?
- How much design do you need for each task?
- Break down the task into ~5 smaller tasks and estimate them.
- Revise your overall estimate if necessary

π

17-313
Foundations of
SOFTWARE
ENGINEERING


How Much Does This Cow Weigh?
(All People)
Number Of Guesses
2,200
2,000
1,800
1,600
1,400
1,200
1,000
800
600
400
200
0
Average Guess: 1,287 lbs
Actual Weight: 1,355 lbs
Penelope The Cow
500
1,000
1,500
2,000
2,500
3,000
3,500
Weight (in lbs)
Source: The Internet.
Credit: Quoctrung Bui/NPR

17-313
Foundations of
SOFTWARE
ENGINEERING


XS
S
M
L
XL
made by :codica
codica.com

Measuring Progress?

- "I'm almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week."

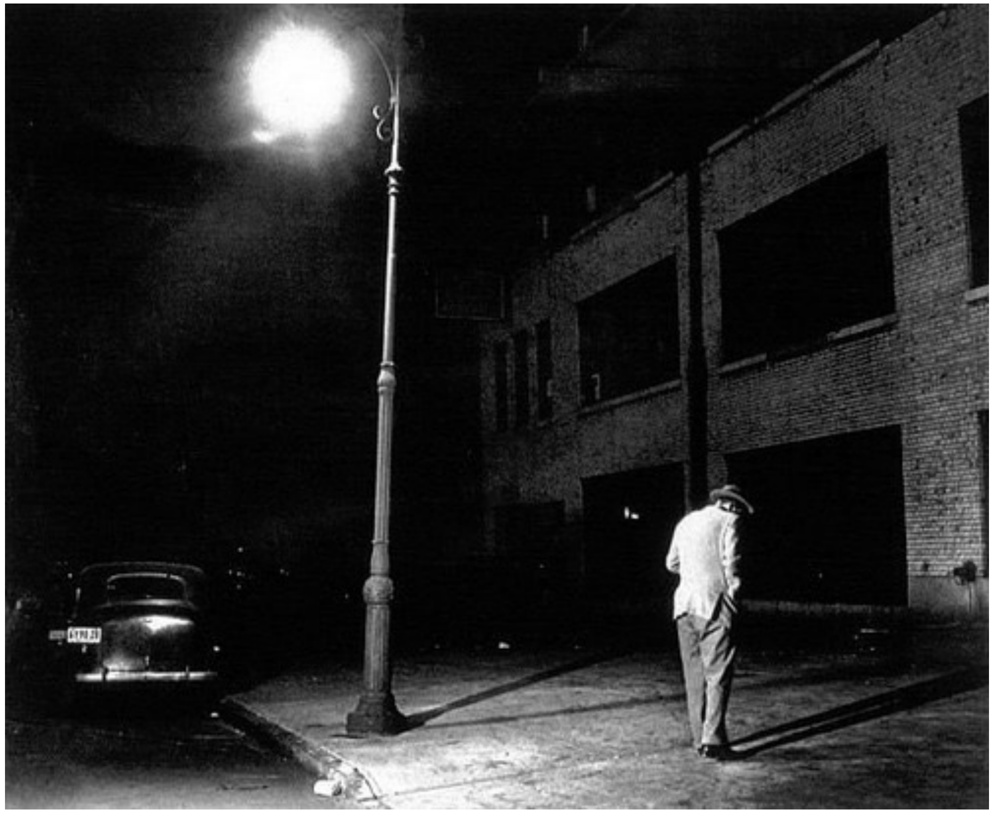
I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.

THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

https://xkcd.com/612/

Measuring Progress?

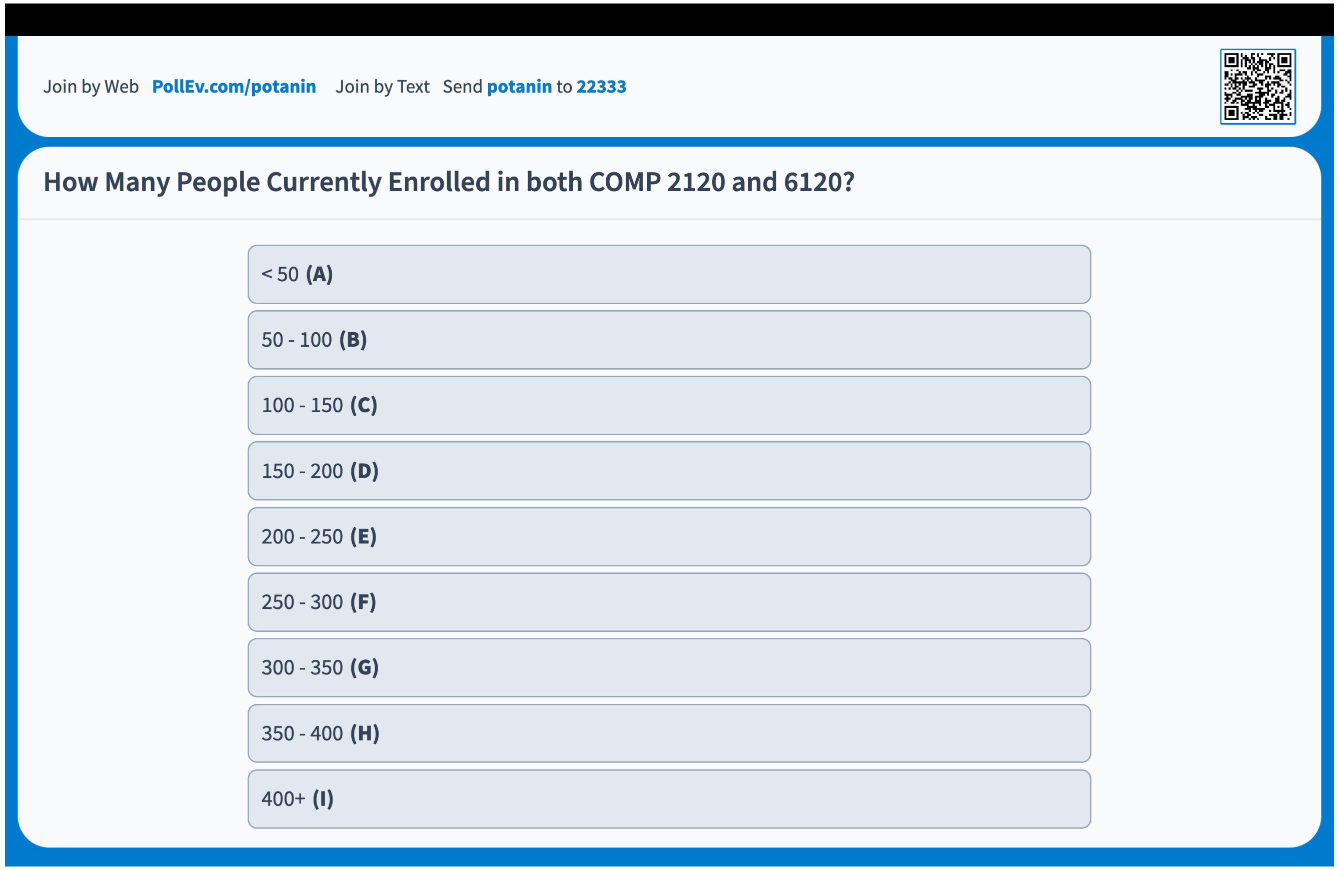
- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?



Milestones and deliverables

- Making progress observable, especially for software
- Milestone: clear end point of a (sub)tasks
  - For project manager
  - Reports, prototypes, completed subprojects
  - "80% done" not a suitable mile stone
- Deliverable: Result for customer
  - Similar to mile stone, but for customers
  - Reports, prototypes, completed subsystems

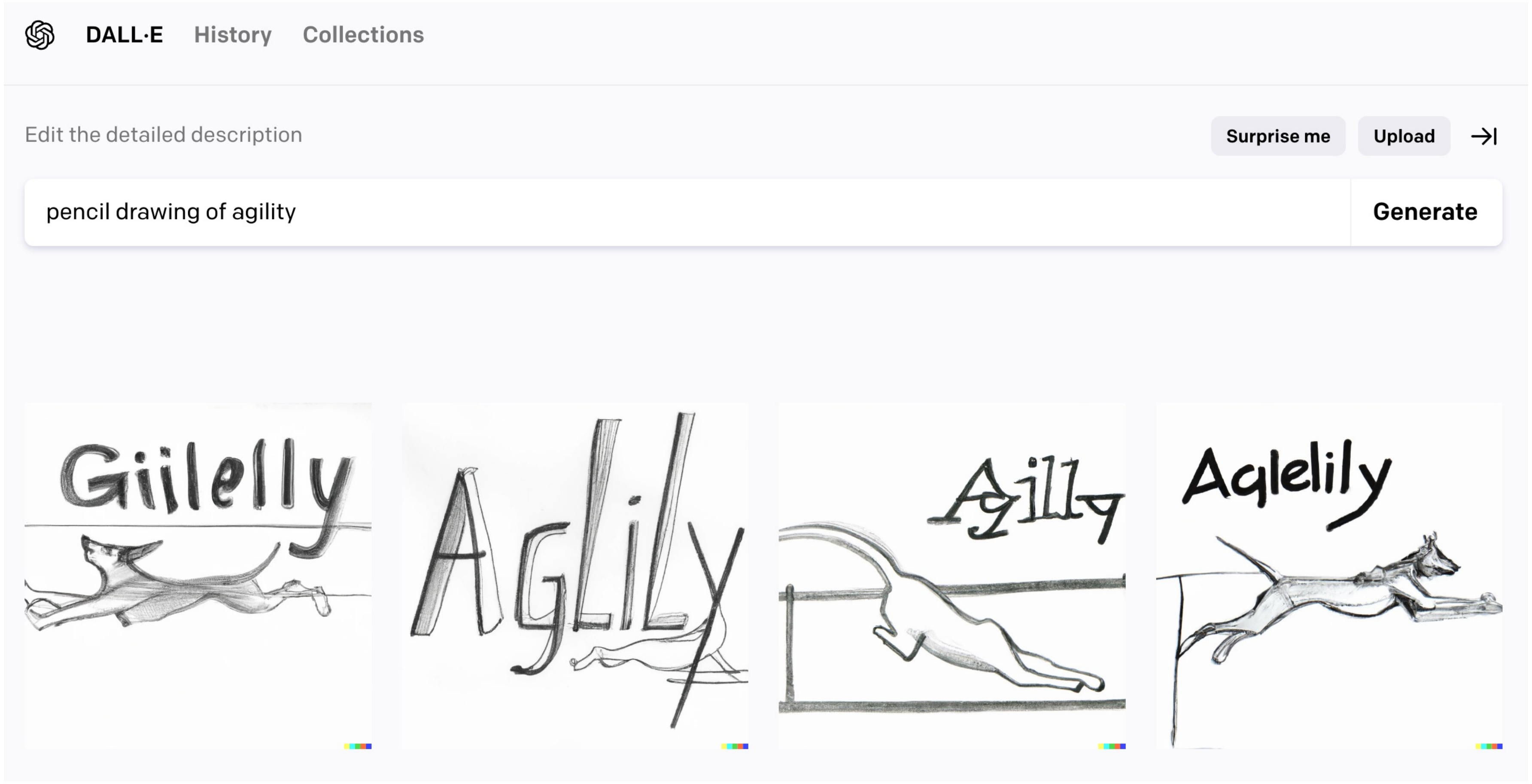
Poll Everywhere Time!



Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

How Many People Currently Enrolled in both COMP 2120 and 6120?

- < 50 **(A)**
- 50 - 100 **(B)**
- 100 - 150 **(C)**
- 150 - 200 **(D)**
- 200 - 250 **(E)**
- 250 - 300 **(F)**
- 300 - 350 **(G)**
- 350 - 400 **(H)**
- 400+ **(I)**


DALL·E
History
Collections
Edit the detailed description
Surprise me
Upload
pencil drawing of agility
Generate
Giilelly
AgLiLy
Ajilly
Aqlelily

Agile

Agile Software Engineering

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Software products must be brought to market quickly so rapid software development and delivery is essential.
- Virtually all software products are now developed using an agile approach.
- Agile software engineering focuses on delivering functionality quickly, responding to changing product specifications and minimizing development overheads.
- A large number of 'agile methods' have been developed.
  - There is no 'best' agile method or technique.
  - It depends on who is using the technique, the development team and the type of product being developed

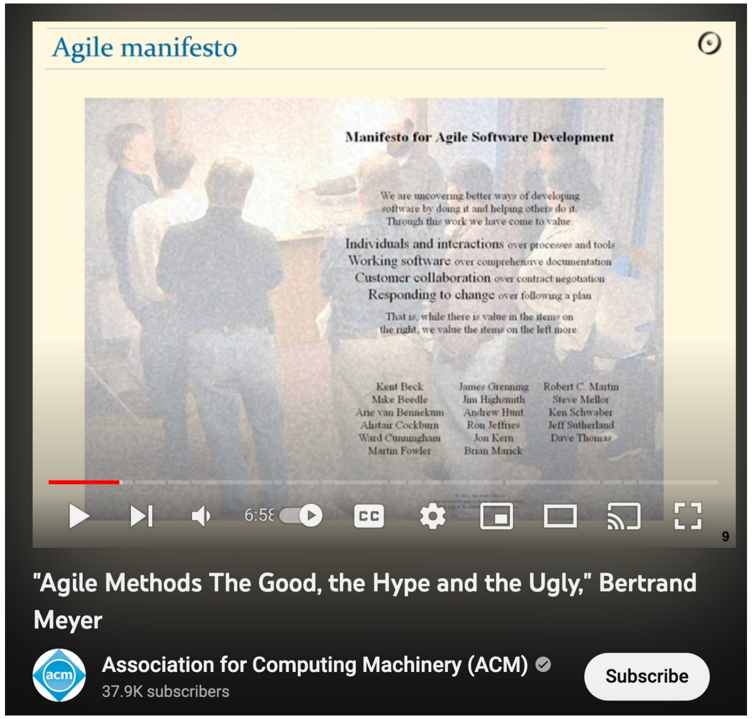
Agile Methods

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- Plan-driven development evolved to support the engineering of large, long-lifetime systems (such as aircraft control systems) where teams may be geographically dispersed and work on the software for several years.
    - This approach is based on controlled and rigorous software development processes that include detailed project planning, requirements specification and analysis and system modelling.
    - However, plan-driven development involves significant overheads and documentation and it does not support the rapid development and delivery of software.
- Agile methods were developed in the 1990s to address this problem.
    - These methods focus on the software rather than its documentation, develop software in a series of increments and aim to reduce process bureaucracy as much as possible.

The Agile Manifesto

- We are uncovering better ways of developing software by doing it and helping others to 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.
- While there is value on the items on the right, we value the items on the left more.


Agile manifesto

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

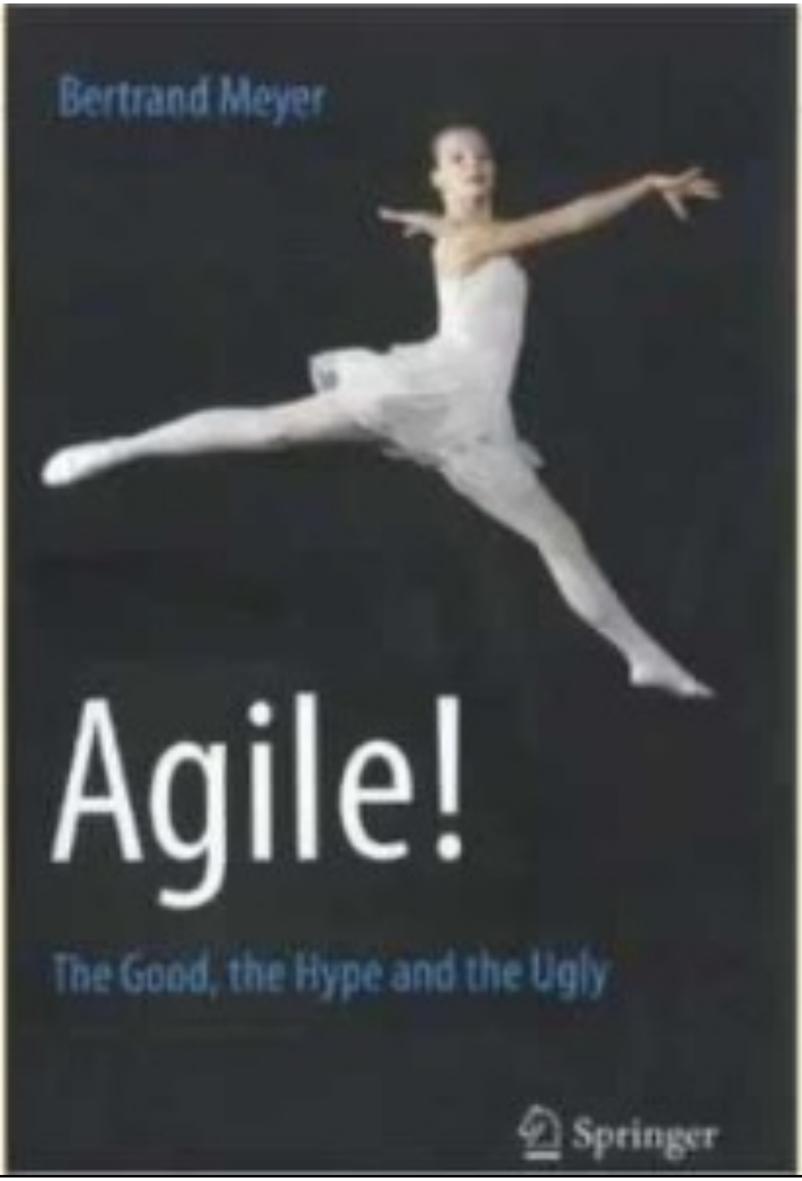
That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

"Agile Methods The Good, the Hype and the Ugly," Bertrand Meyer

Association for Computing Machinery (ACM)
37.9K subscribers

https://www.youtube.com/watch?v=ffkIQrq-m34


Bertrand Meyer
Agile!
The Good, the Hype and the Ugly
Springer

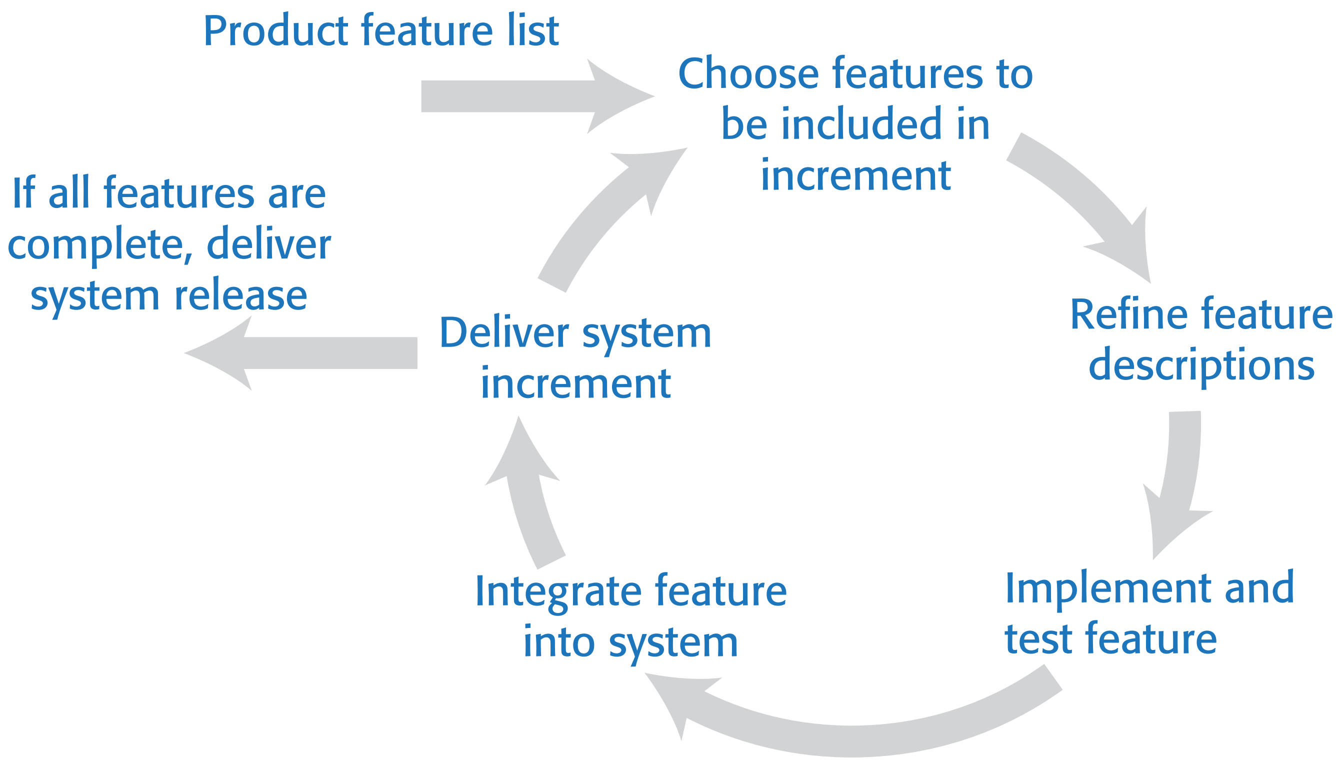
Incremental Development

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- All agile methods are based around incremental development and delivery.
- Product development focuses on the software features, where a feature does something for the software user.
- With incremental development, you start by prioritizing the features so that the most important features are implemented first.
  - You only define the details of the feature being implemented in an increment.
  - That feature is then implemented and delivered.
- Users or surrogate users can try it out and provide feedback to the development team. You then go on to define and implement the next feature of the system.

Incremental Development

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION


Product feature list
Choose features to be included in increment
Refine feature descriptions
Implement and test feature
Integrate feature into system
Deliver system increment
If all features are complete, deliver system release

Incremental Development Activities

Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Choose features to be included in an increment
Using the list of features in the planned product, select those features that can be implemented in the next product increment.

Refine feature descriptions
Add detail to the feature descriptions so that the team have a common understanding of each feature and there is sufficient detail to begin implementation.

Incremental Development Activities

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Implement and test
Implement the feature and develop automated tests for that feature that show that its behaviour is consistent with its description.

Integrate feature and test
Integrate the developed feature with the existing system and test it to check that it works in conjunction with other features.

Deliver system increment
Deliver the system increment to the customer or product manager for checking and comments. If enough features have been implemented, release a version of the system for customer use.

Agile Development Principles

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Involve the customer
Involve customers closely with the software development team. Their role is to provide and prioritize new system requirements and to evaluate each increment of the system.

Embrace change
Expect the features of the product and the details of these features to change as the development team and the product manager learn more about it. Adapt the software to cope with changes as they are made.

Develop and deliver incrementally
Always develop software products in increments. Test and evaluate each increment as it is developed and feed back required changes to the development team.

Agile Development Principles

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

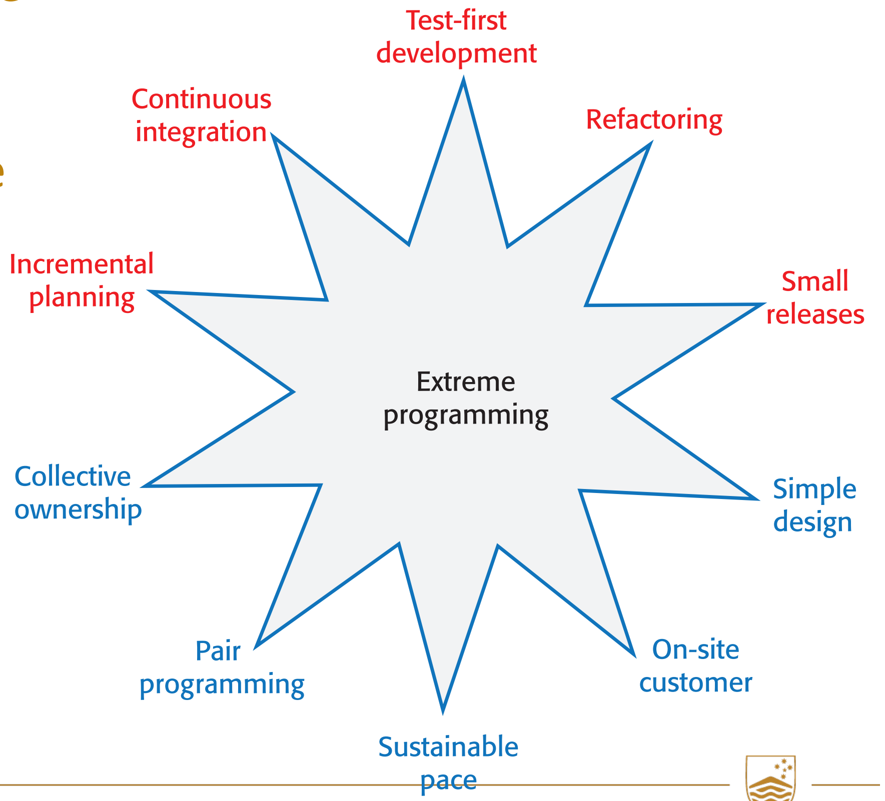
Maintain simplicity
Focus on simplicity in both the software being developed and in the development process. Wherever possible, do what you can to eliminate complexity from the system.

Focus on people, not things
Trust the development team and do not expect everyone to always do the development process in the same way. Team members should be left to develop their own ways of working without being limited by prescriptive software processes.

Extreme Programming (XP)

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- The most influential work that has changed software development culture was the development of Extreme Programming (XP).
- The name was coined by Kent Beck in 1998 because the approach was developed by pushing recognized good practice, such as iterative development, to 'extreme' levels.
- Extreme programming focused on 12 new development techniques that were geared to rapid, incremental software development, change and delivery.
- Some of these techniques are now widely used; others have been less popular.



Widely Adopted XP Practices

Incremental planning/user stories
There is no 'grand plan' for the system. Instead, what needs to be implemented (the requirements) in each increment are established in discussions with a customer representative. The requirements are written as user stories. The stories to be included in a release are determined by the time available and their relative priority.

Small releases
The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the previous release.

Widely Adopted XP Practices

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Test-driven development
Instead of writing code then tests for that code, developers write the tests first. This helps clarify what the code should actually do and that there is always a 'tested' version of the code available. An automated unit test framework is used to run the tests after every change. New code should not 'break' code that has already been implemented.

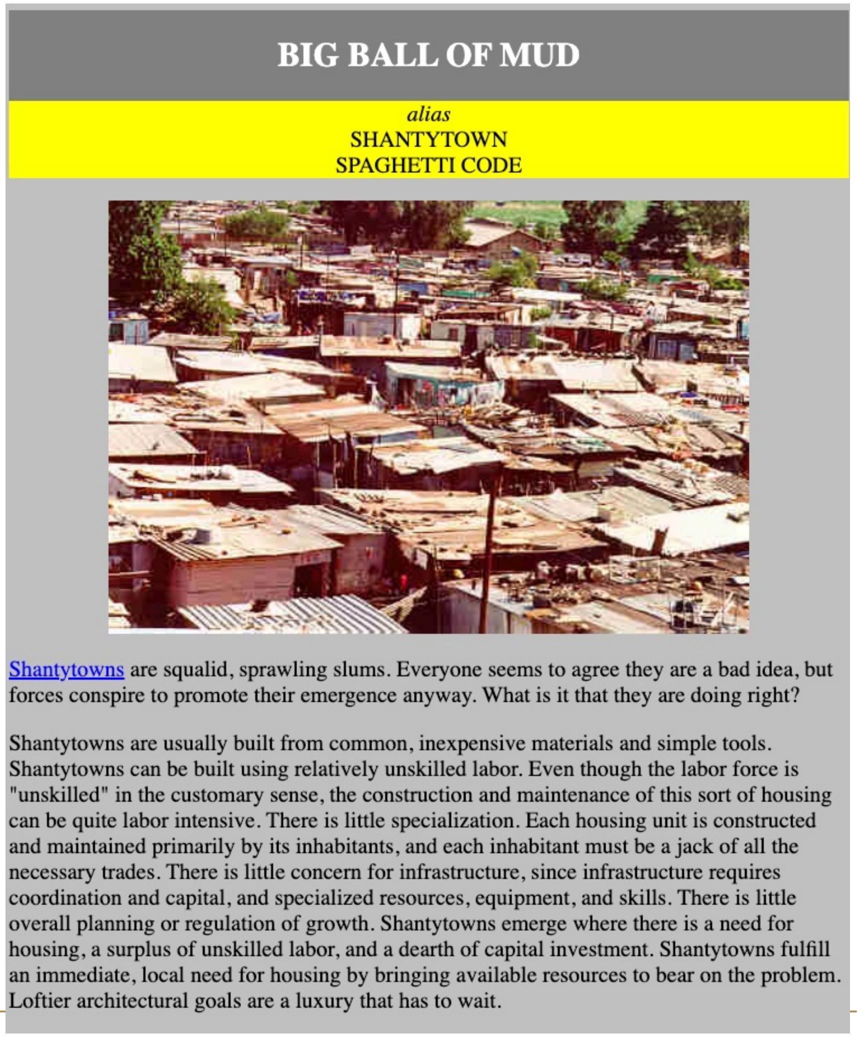
Continuous integration
As soon as the work on a task is complete, it is integrated into the whole system and a new version of the system is created. All unit tests from all developers are run automatically and must be successful before the new version of the system is accepted.

Widely Adopted XP Practices

Refactoring

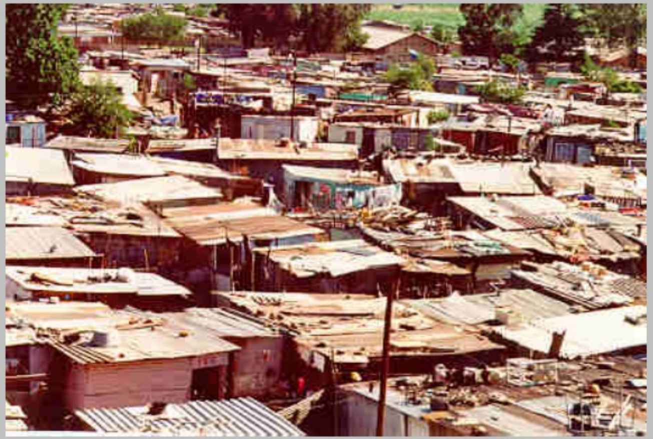
Refactoring means improving the structure, readability, efficiency and security of a program. All developers are expected to refactor the code as soon as potential code improvements are found. This keeps the code simple and maintainable.

http://www.laputan.org/mud/



BIG BALL OF MUD

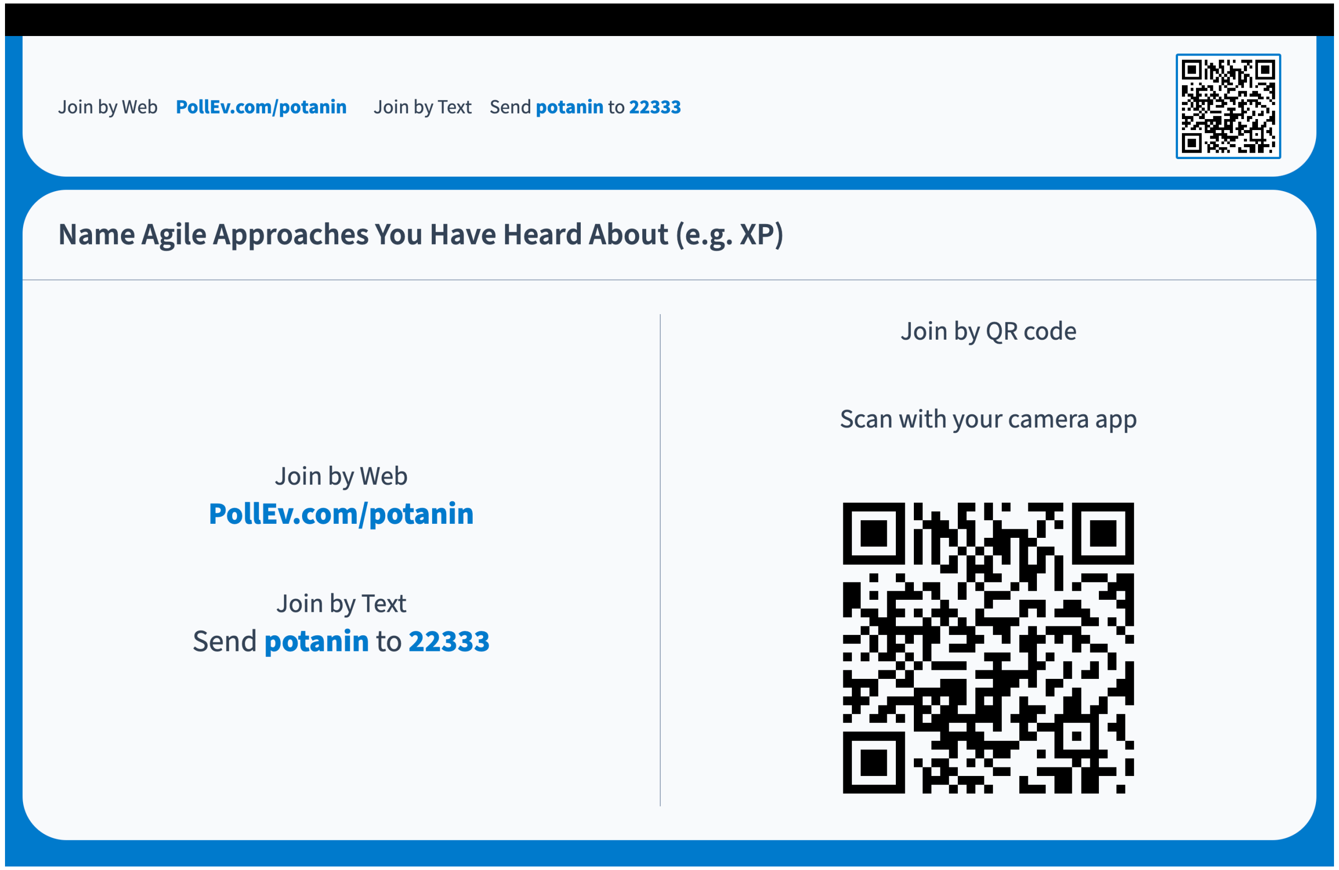
alias
SHANTYTOWN
SPAGHETTI CODE



Shantytowns are squalid, sprawling slums. Everyone seems to agree they are a bad idea, but forces conspire to promote their emergence anyway. What is it that they are doing right?

Shantytowns are usually built from common, inexpensive materials and simple tools. Shantytowns can be built using relatively unskilled labor. Even though the labor force is "unskilled" in the customary sense, the construction and maintenance of this sort of housing can be quite labor intensive. There is little specialization. Each housing unit is constructed and maintained primarily by its inhabitants, and each inhabitant must be a jack of all the necessary trades. There is little concern for infrastructure, since infrastructure requires coordination and capital, and specialized resources, equipment, and skills. There is little overall planning or regulation of growth. Shantytowns emerge where there is a need for housing, a surplus of unskilled labor, and a dearth of capital investment. Shantytowns fulfill an immediate, local need for housing by bringing available resources to bear on the problem. Loftier architectural goals are a luxury that has to wait.

Poll Everywhere Time!



Join by Web PollEv.com/potanin Join by Text Send **potanin** to **22333**

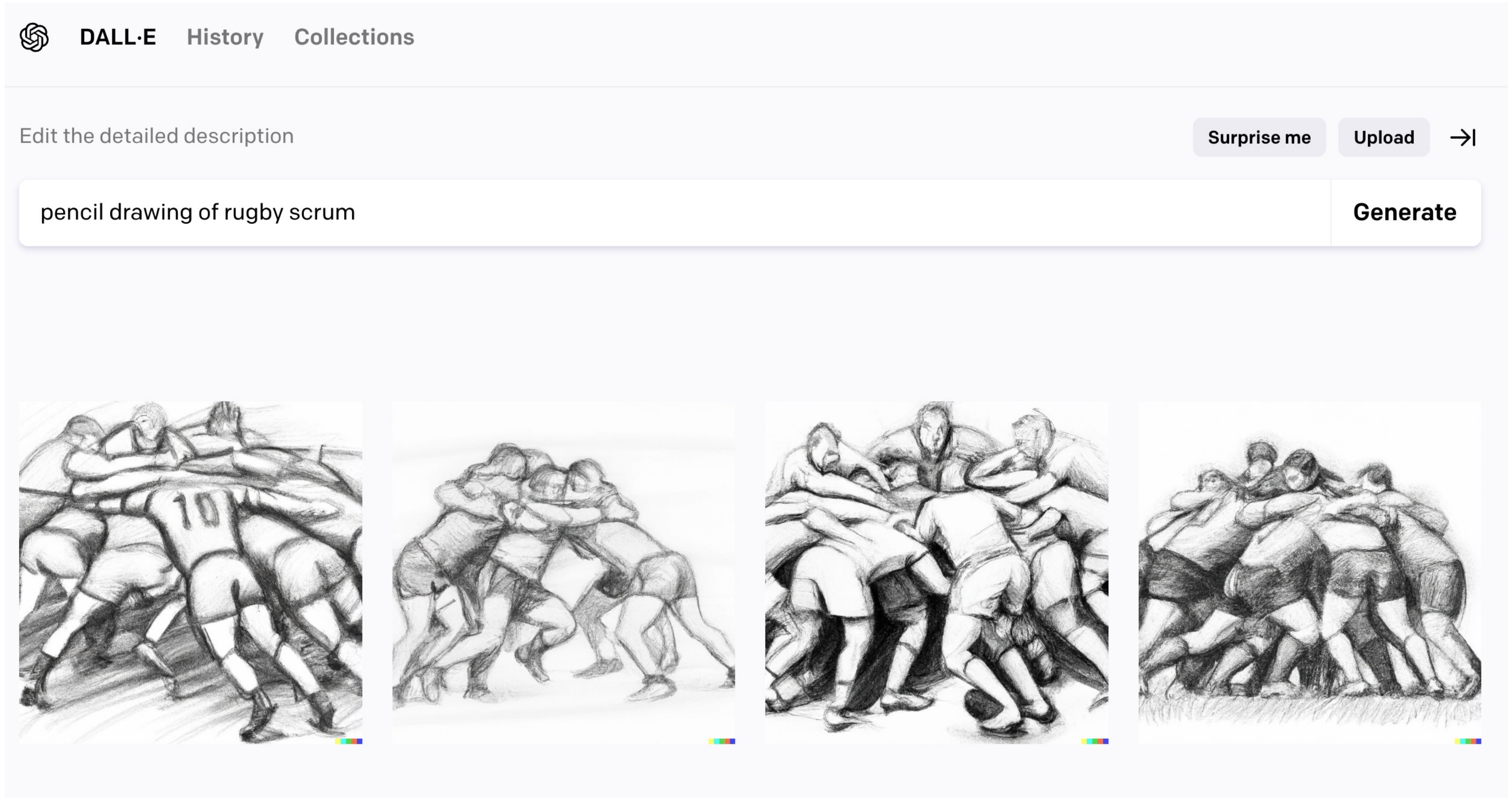
Name Agile Approaches You Have Heard About (e.g. XP)

Join by Web
PollEv.com/potanin

Join by Text
Send **potanin** to **22333**

Join by QR code

Scan with your camera app

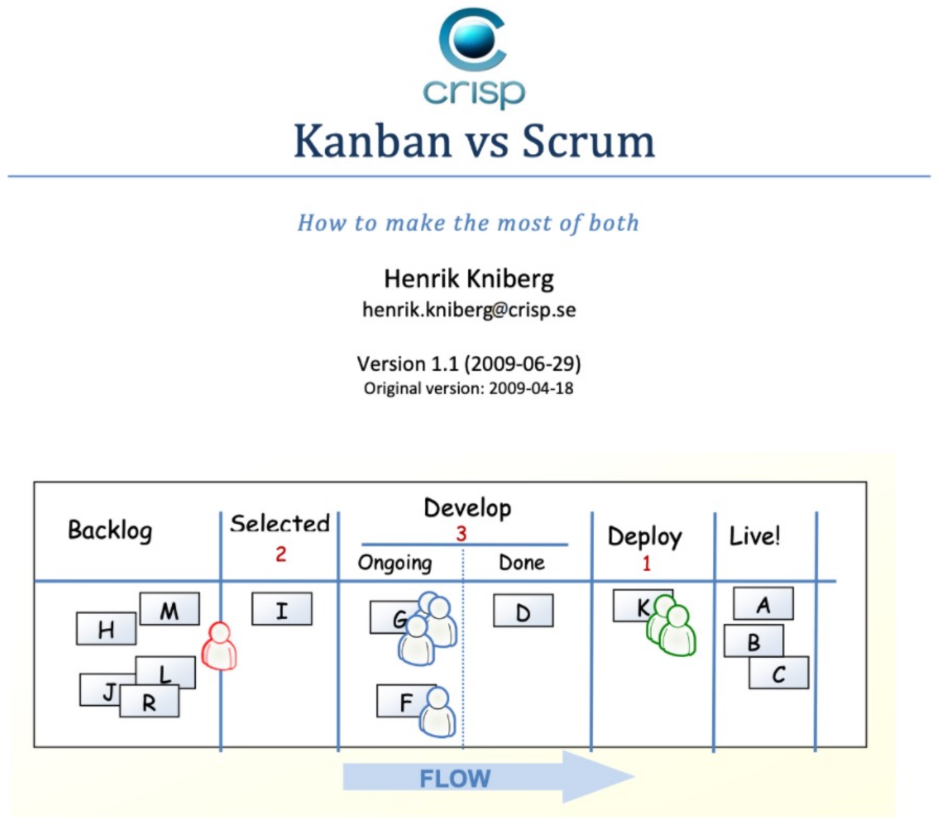


Edit the detailed description

pencil drawing of rugby scrum

Scrum

What Approaches are Out There?


crisp

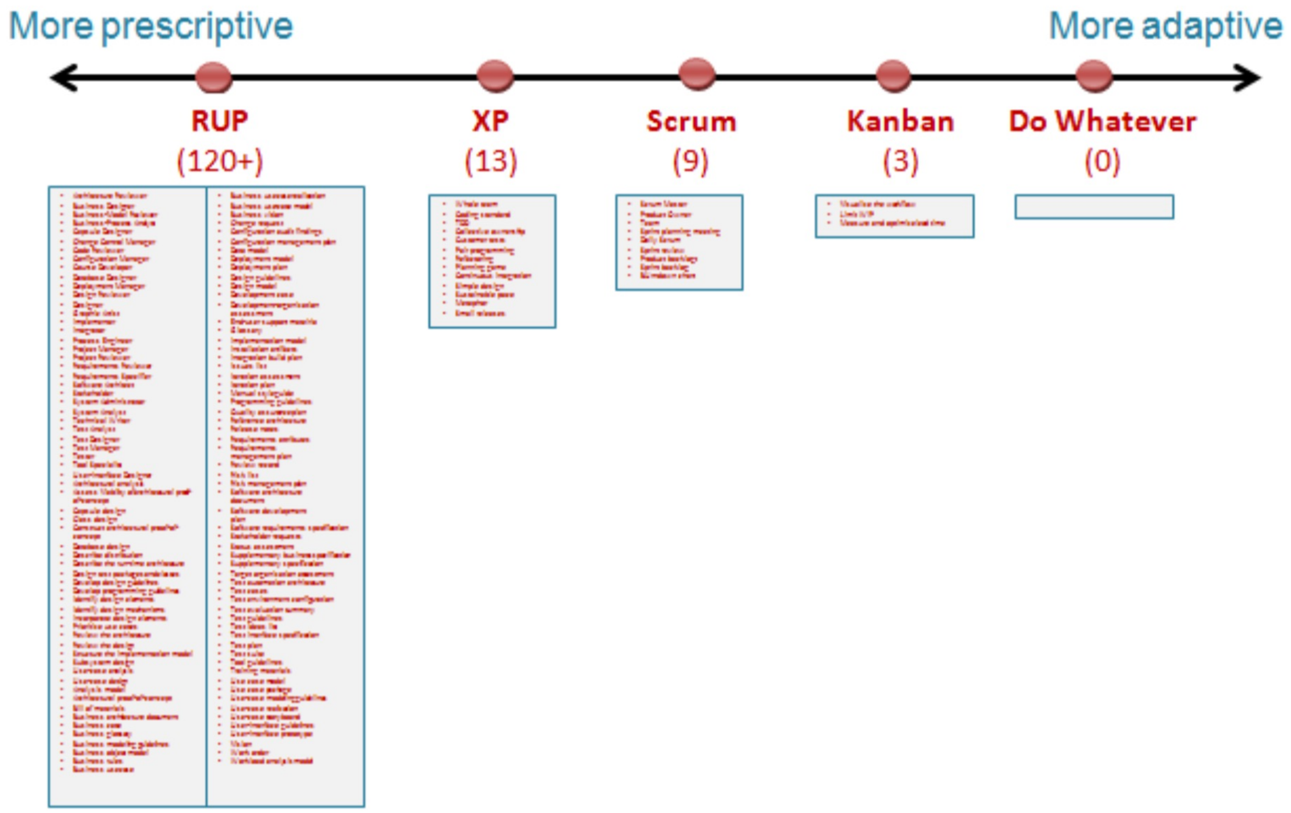
Kanban vs Scrum

How to make the most of both

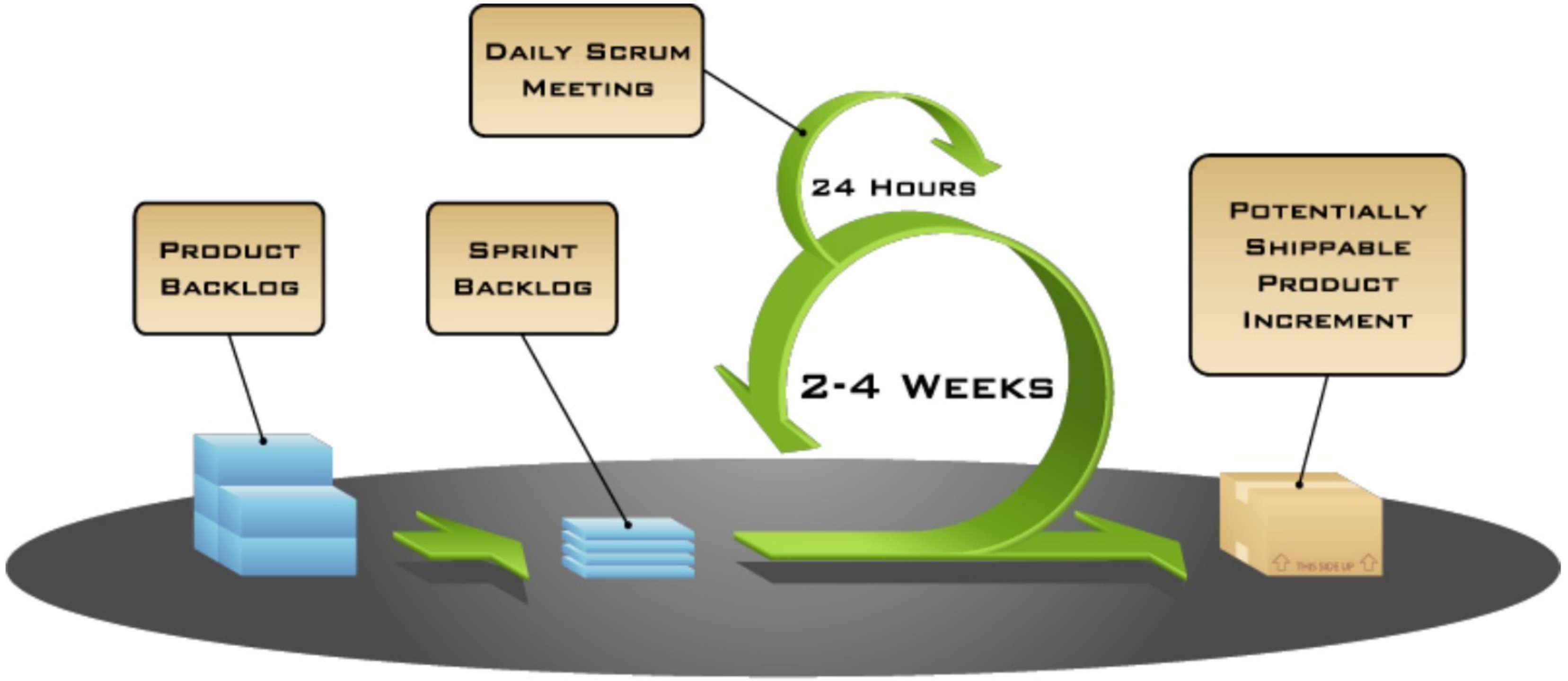
Henrik Kniberg
henrik.kniberg@crisp.se

Version 1.1 (2009-06-29)
Original version: 2009-04-18

What Approaches are Out There?



Brief intro to Scrum



Scrum Terminology

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Product
The software product that is being developed by the Scrum team.

Product owner
A team member who is responsible for identifying product features and attributes. They review work done and help to test the product.

Product backlog
A to-do list of items such as bugs, features and product improvements that the Scrum team have not yet completed.

Development team
A small self-organising team of five to eight people who are responsible for developing the product.

Scrum Terminology

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Sprint
A short period, typically two to four weeks, when a product increment is developed.

Scrum
A daily team meeting where progress is reviewed and work to be done that day as discussed and agreed.

ScrumMaster
A team coach who guides the team in the effective use of Scrum.

Potentially shippable product increment
The output of a sprint which should be of high enough quality to be deployed for customer use.

Velocity
An estimate of how much work a team can do in a single sprint.

Key Roles in Scrum

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

The Product Owner is responsible for ensuring that the development team are always focused on the product they are building rather than diverted into technically interesting but less relevant work.

- In product development, the product manager should normally take on the Product Owner role.

The ScrumMaster is a Scrum expert whose job is to guide the team in the effective use of the Scrum method. The developers of Scrum emphasize that the ScrumMaster is not a conventional project manager but is a coach for the team. They have authority within the team on how Scrum is used.

- In many companies that use Scrum, the ScrumMaster also has some project management responsibilities.

Scrum and Sprints

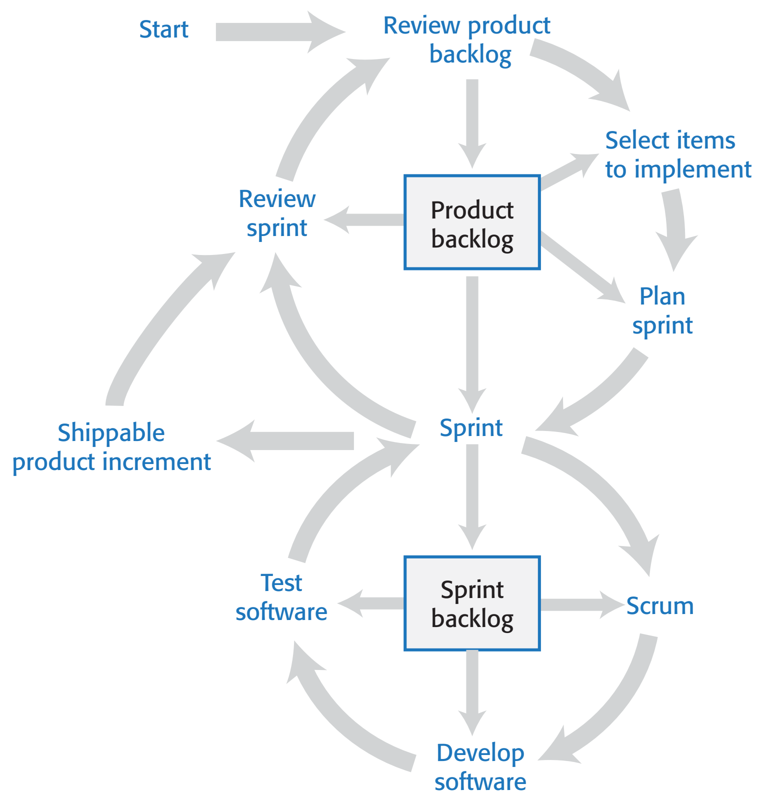
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville
GLOBAL EDITION

- In Scrum, software is developed in sprints, which are fixed-length periods (2 - 4 weeks) in which software features are developed and delivered.
- During a sprint, the team has daily meetings (Scrums) to review progress and to update the list of work items that are incomplete.
- Sprints should produce a ‘shippable product increment’. This means that the developed software should be complete and ready to deploy.

Elements of Scrum

- Products:
  - Product Backlog
  - Sprint Backlog

- Process:
  - Sprint Planning Meeting
  - Daily Scrum Meeting
  - Sprint Retrospective
  - Sprint Review Meeting



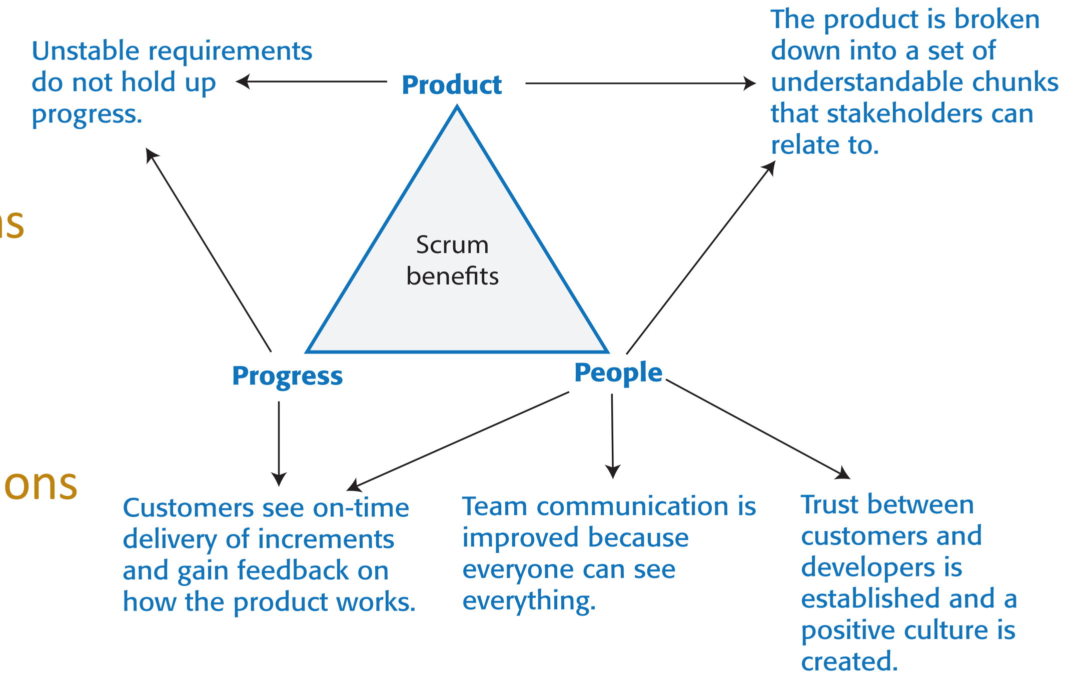
Key Scrum Practices

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Product backlog
This is a to-do list of items to be implemented that is reviewed and updated before each sprint.

Timeboxed sprints
Fixed-time (2-4 week) periods in which items from the product backlog are implemented,

Self-organizing teams
Self-organizing teams make their own decisions and work by discussing issues and making decisions by consensus.


Scrum benefits
Product
Progress
People
Unstable requirements do not hold up progress.
The product is broken down into a set of understandable chunks that stakeholders can relate to.
Customers see on-time delivery of increments and gain feedback on how the product works.
Team communication is improved because everyone can see everything.
Trust between customers and developers is established and a positive culture is created.

Product Backlogs

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

- The product backlog is a list of what needs to be done to complete the development of the product.
- The items on this list are called product backlog items (PBIs).
- The product backlog may include a variety of different items such as product features to be implemented, user requests, essential development activities and desirable engineering improvements.
- The product backlog should always be prioritized so that the items that be implemented first are at the top of the list.

Examples of Product Backlog Items

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

1. As a teacher, I want to be able to configure the group of tools that are available to individual classes. (feature)
2. As a parent, I want to be able to view my childrens’ work and the assessments made by their teachers. (feature)
3. As a teacher of young children, I want a pictorial interface for children with limited reading ability. (user request)
4. Establish criteria for the assessment of open source software that might be used as a basis for parts of this system. (development activity)
5. Refactor user interface code to improve understandability and performance. (engineering improvement)
6. Implement encryption for all personal user data. (engineering improvement)

Product Backlog Item States

GLOBAL EDITION
Engineering Software Products
An Introduction to Modern Software Engineering
Ian Sommerville

Ready for consideration

These are high-level ideas and feature descriptions that will be considered for inclusion in the product. They are tentative so may radically change or may not be included in the final product.

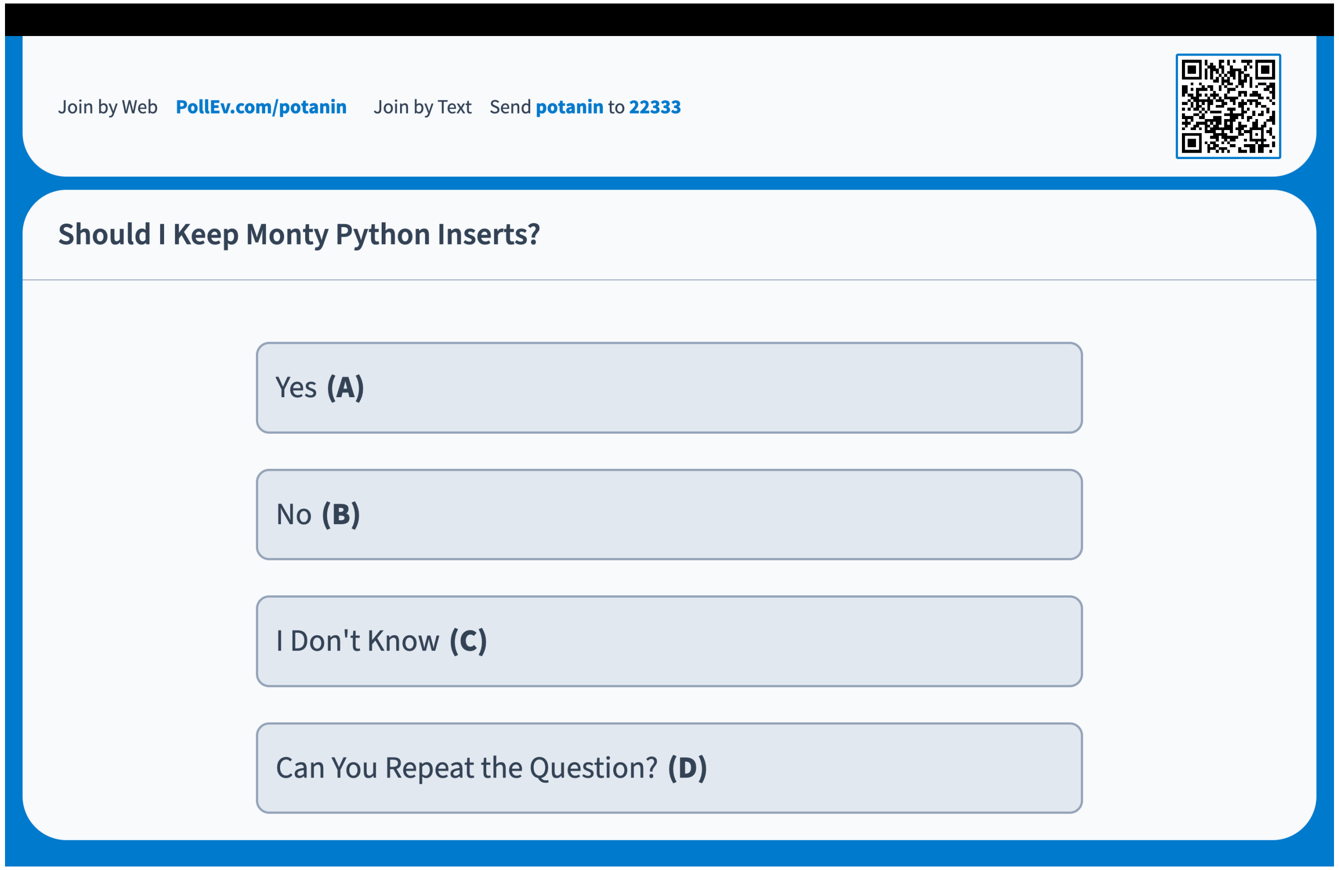
Ready for refinement

The team has agreed that this is an important item that should be implemented as part of the current development. There is a reasonably clear definition of what is required. However, work is needed to understand and refine the item.

Ready for implementation

The PBI has enough detail for the team to estimate the effort involved and to implement the item. Dependencies on other items have been identified.

Poll Everywhere Time!



Join by Web **PollEv.com/potanin** Join by Text Send **potanin** to **22333**

Should I Keep Monty Python Inserts?

- Yes **(A)**
- No **(B)**
- I Don't Know **(C)**
- Can You Repeat the Question? **(D)**