Week:
1 of 12

COMP 2120 / COMP 6120

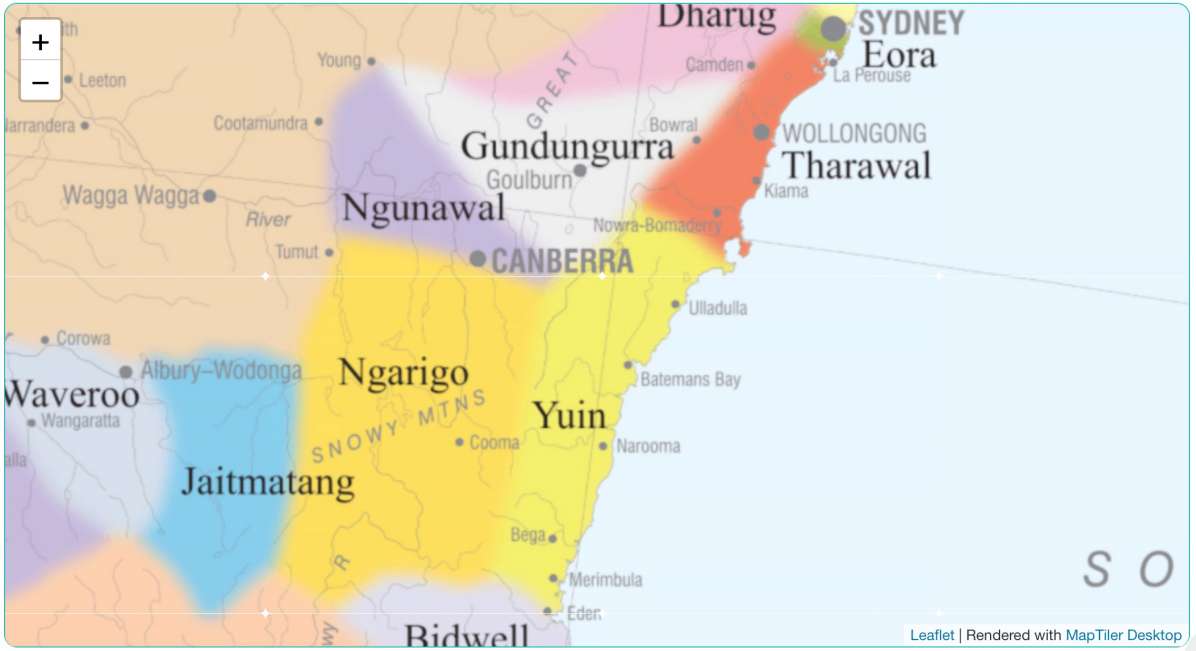
AGILE

A/Prof Alex Potanin and Dr Melina Vidoni

ANU Acknowledgment of Country

AIATSIS Explore Family history Collection Research Education What's new About Shop Search

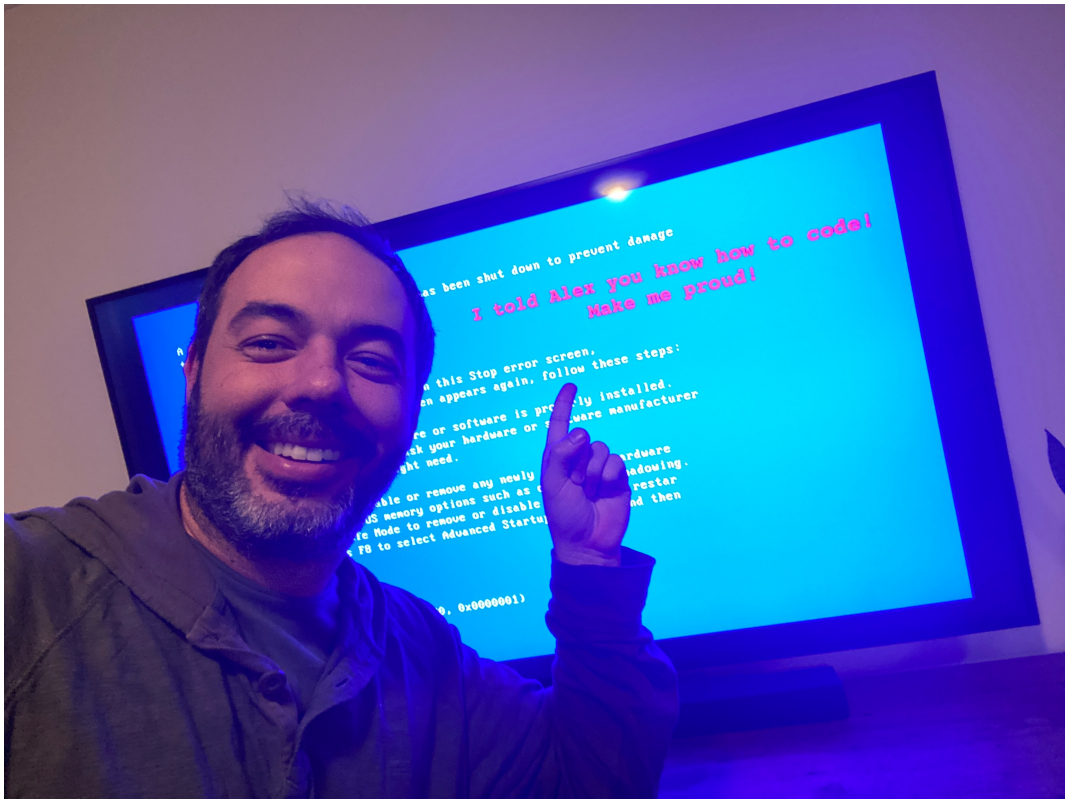
"We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present."



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

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”



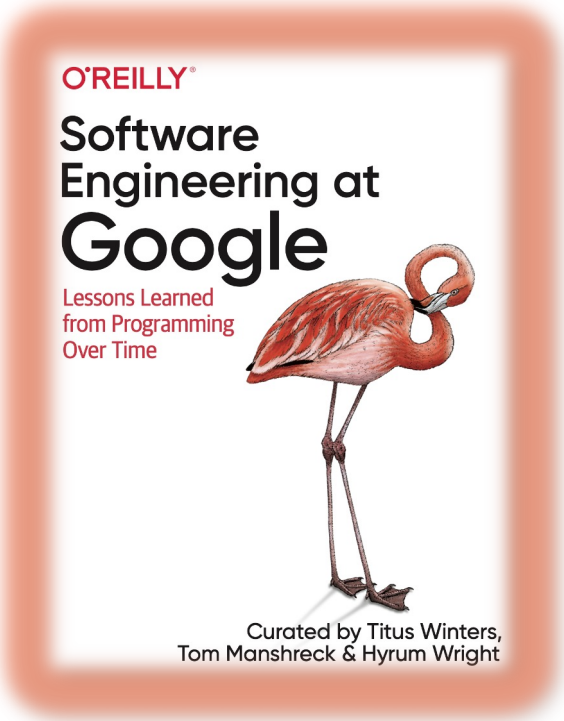
About Me

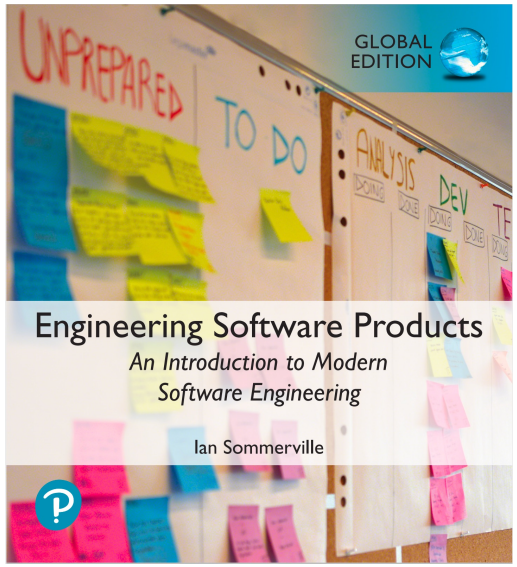
- Alex Potanin: from Russia via South Korea and Czech Republic to New Zealand and now Australia. Family from USA, Germany, China, Indonesia, and New Zealand. ☺
- Interested in *Secure* Programming Languages and Software Architecture (*Module* systems and *Object Capabilities*).
- **U**sable **M**odules **L**ab **(UML)** – let me know if you want to join our research group
- http://wyvernlang.github.io
- Office Hours: Wednesday 3pm to 5pm in N328 on the 3rd Floor of 108 North Road

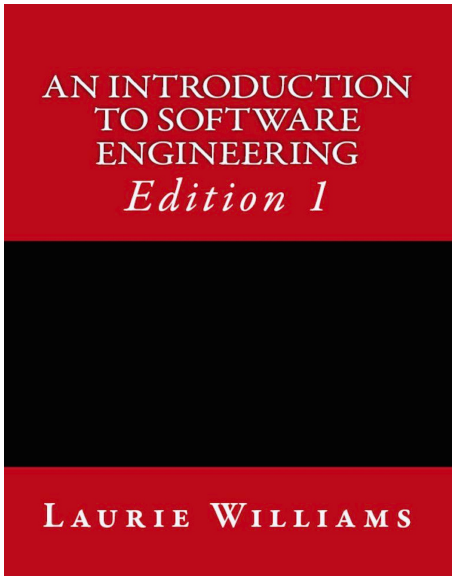
COMP 2120 / COMP 6120 Overview

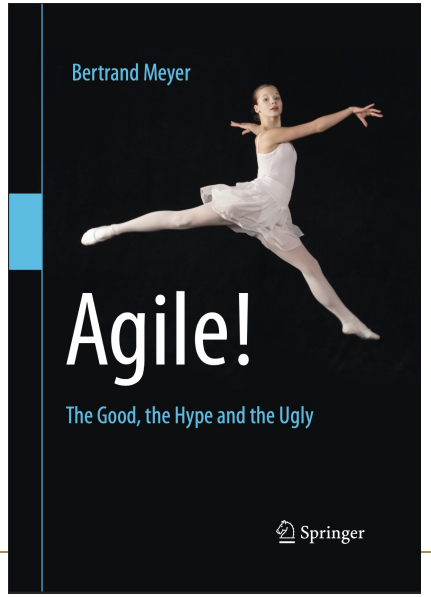
- https://comp.anu.edu.au/courses/comp2120/
- *Use Piazza for the Forums!*
- Individual Assignment on Agile Basics (5%)
- Group Assignment 1 on Inspection (20%)
- Group Assignment 2 on Adding Features (20%)
- Group Assignment 3 on PR for a real Open Source Project (20%)
- Group Presentation (5%)
- Final Take Home Exam (30%)

COMP 2120 / 6120 Books









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.

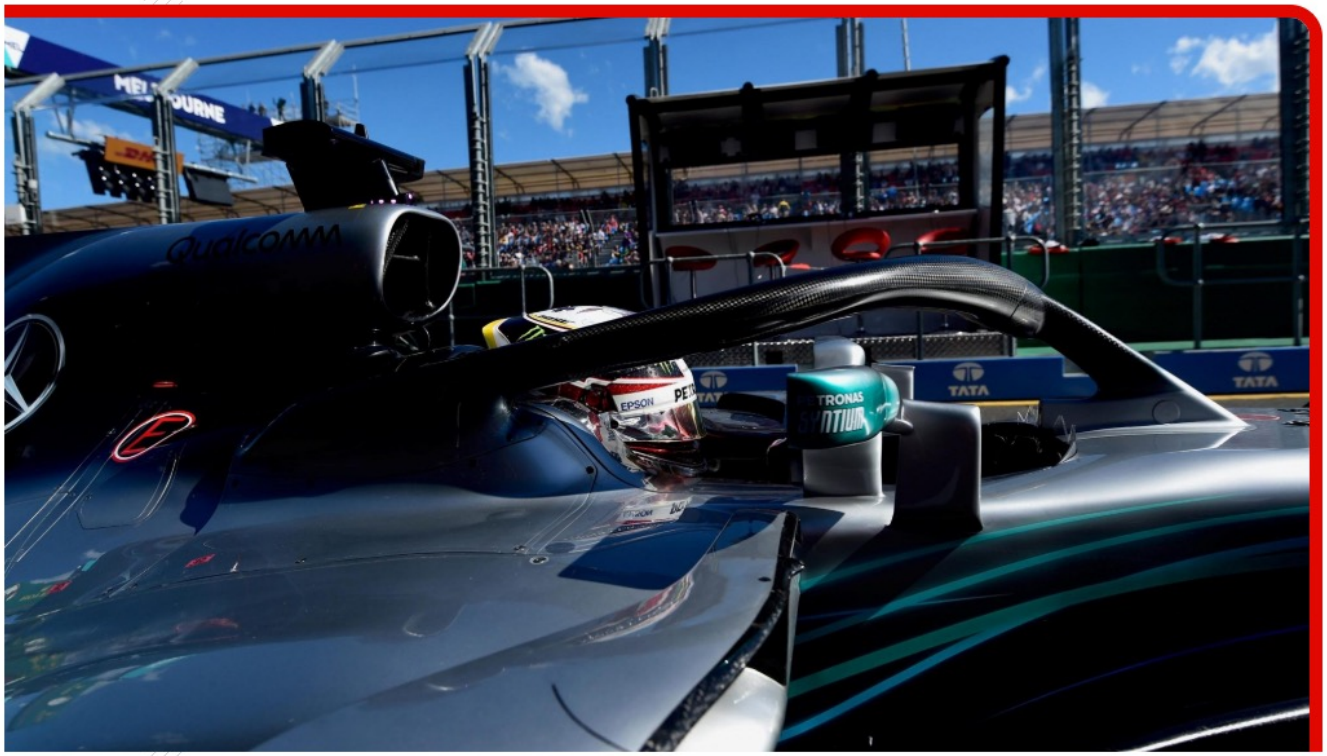
Hamilton was passed by Ferrari's Sebastian Vettel when the German took a pit stop after the virtual safety car had been deployed. Mercedes had not advised Hamilton to open up a greater lead since their data said the gap he held was sufficient. But that information proved incorrect.

"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."

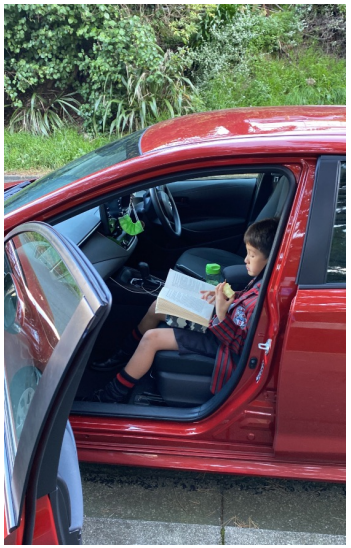
Barr also said more than half the dozens of tasks' deaths studied by the experts in their experiments "were not detected by any fail safe."

Bookout Trial Reporting

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

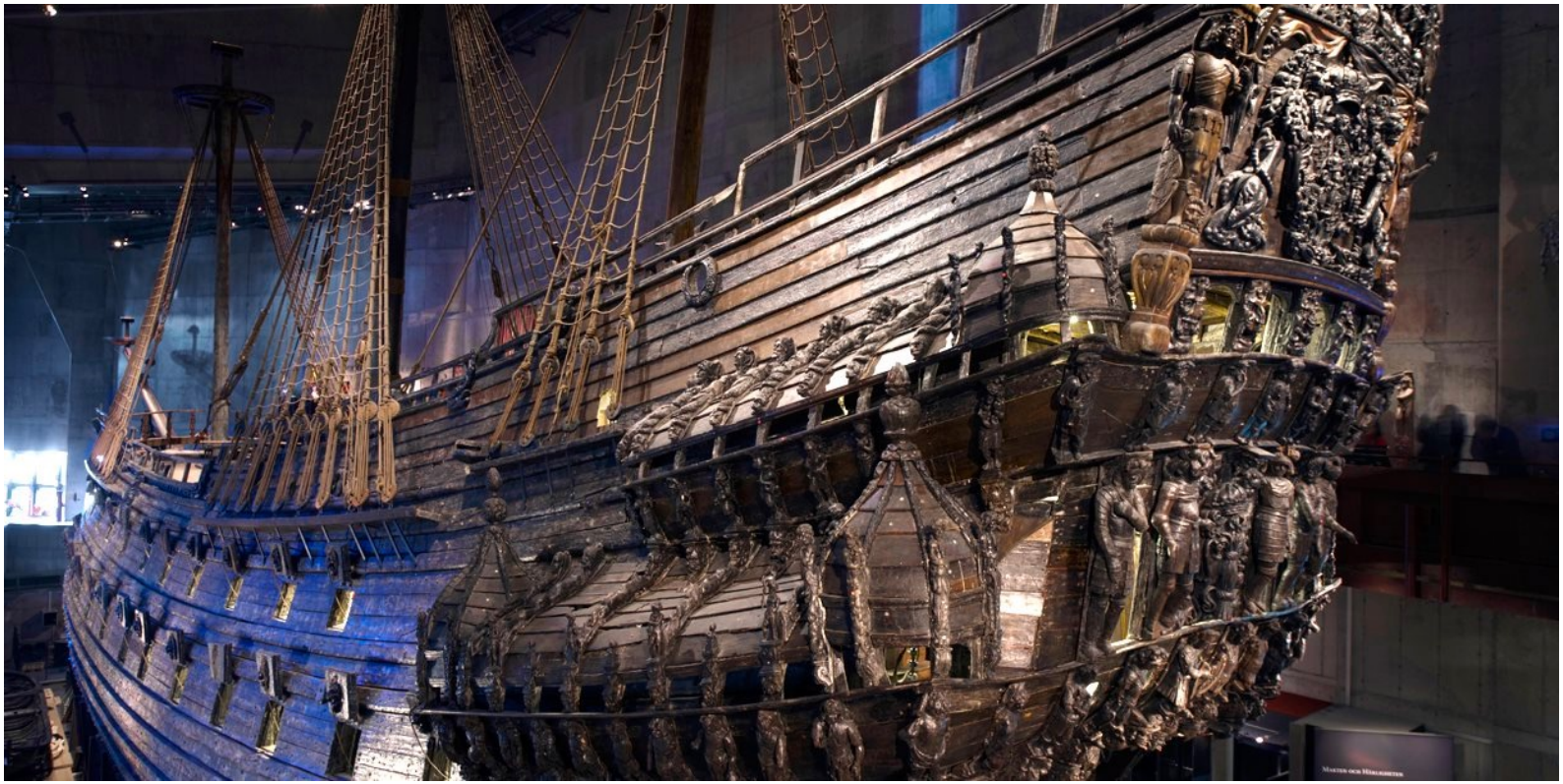
"Task X death in combination with other task deaths"

14



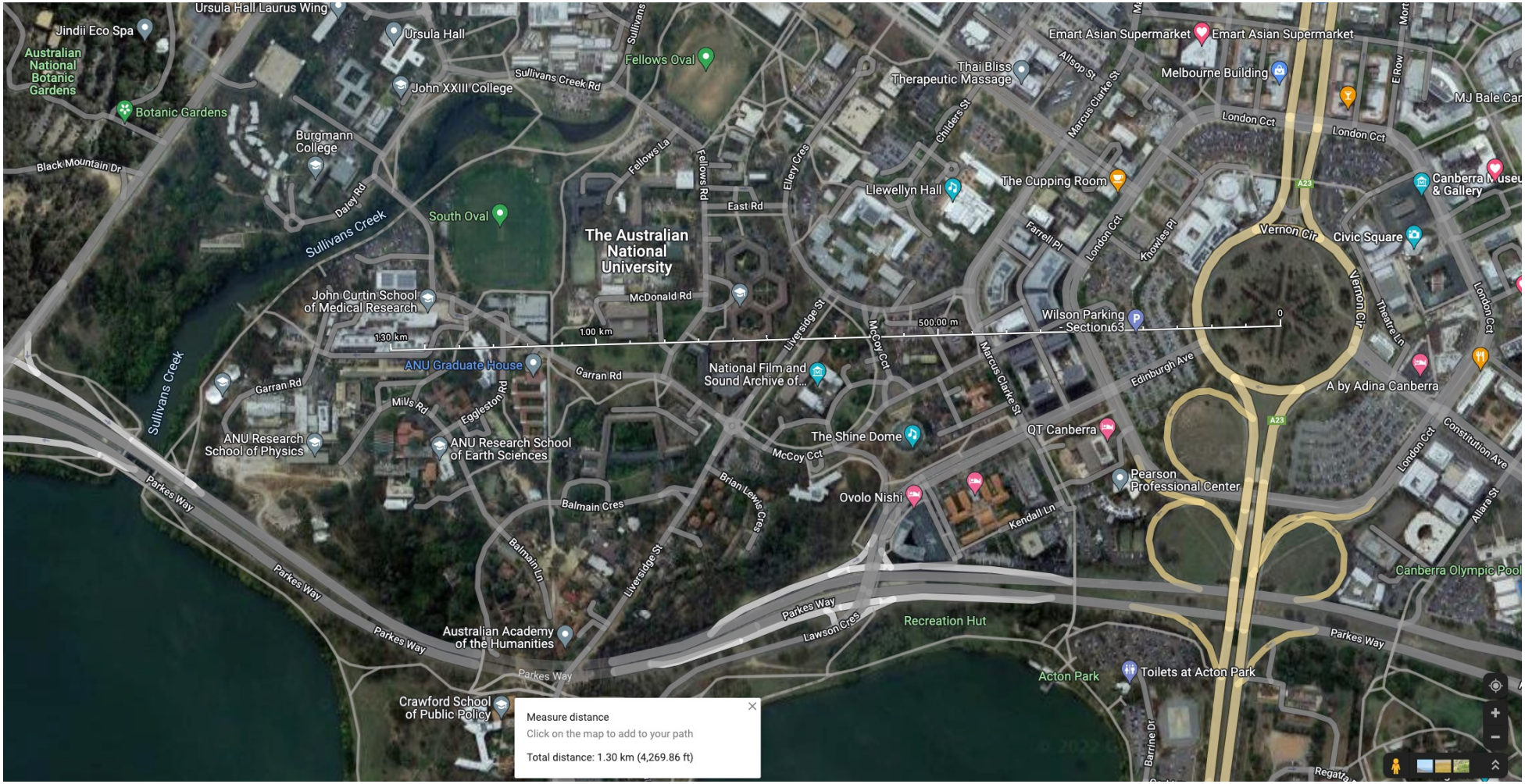

LIFE VEST

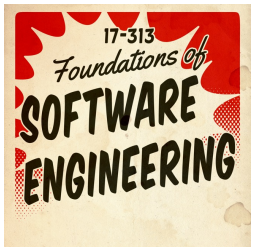
Vasa



Vasa


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





What happened is now called “Vasa syndrome”

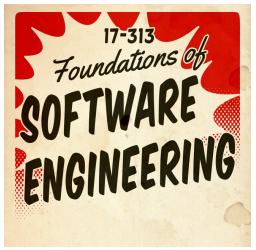
- Changing shipbuilding orders
- No specifications for modified keel
- Shifting armaments requirements
- Shipwright’s death
- No way to calculate stability, stiffness, or sailing characteristics
- Failed pre-launch stability tests

Requirements

Teams

Metrics

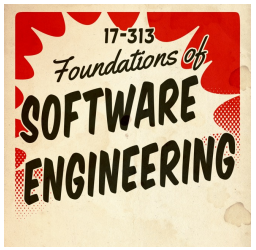
QA



Software *Engineering*?

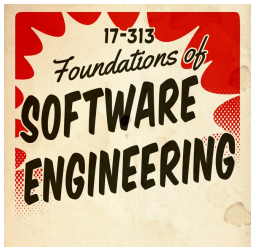
What is **engineering**? And how is it different from **hacking/programming**?

1968 NATO Conference on Software Engineering



- Provocative Title
- Call for Action
- “Software crisis”

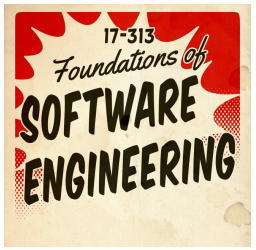
Margaret Hamilton



Name

Previous software development experience?

Software development ambitions?



"…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

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* (unless you are away from Canberra)
- Divide work and integrate
- Establish a process
- **Set and document clear responsibilities and expectations**
  - Possible Roles: Coordinator, Scribe, Checker, Monitor
  - Rotate roles every assignment
- Every team member should understand the entire solution

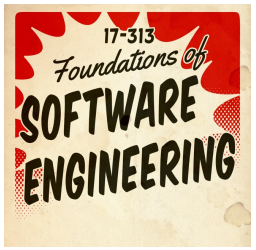
Teamwork: Dealing with problems

- *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 In-Team Communication

- Asana, Trello, Microsoft Project, …
- Github Wiki, Google docs, …
- Email, Slack, Facebook groups, …

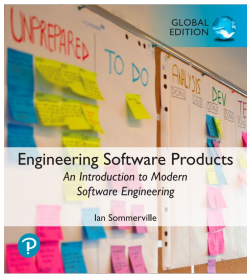
Professionalism



- 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

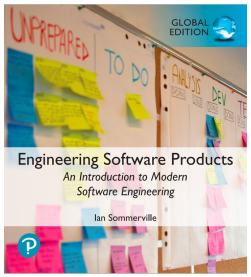
Software Products

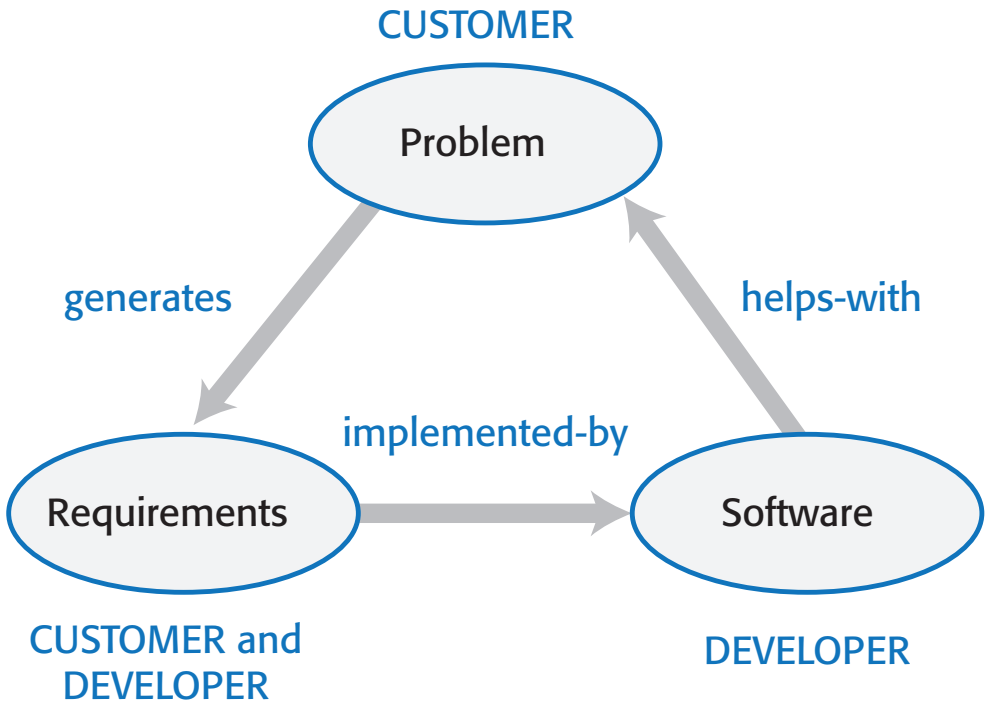
Any slide with ESP Book in the top right corner is Copyright by Ian Sommerville 2018



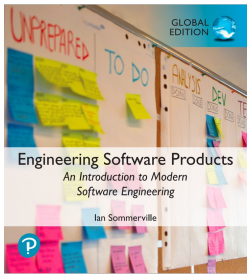
- Software products are generic software systems that provide functionality that is useful to a range of customers.
- Many different types of products are available from large-scale business systems (e.g. MS Excel) through personal products (e.g. Evernote) to simple mobile phone apps and games (e.g. 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



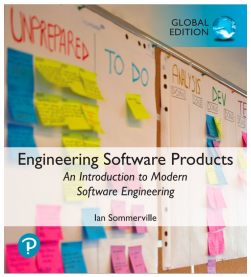


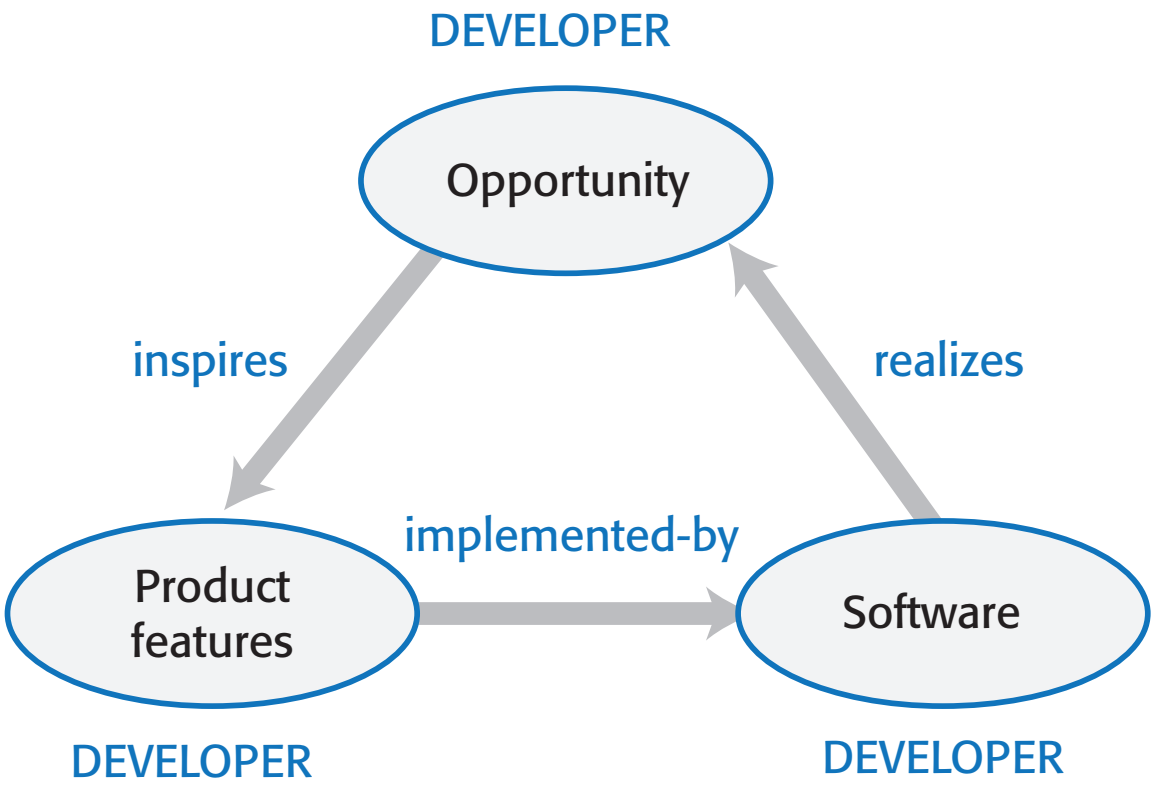
Project-Based Software Engineering



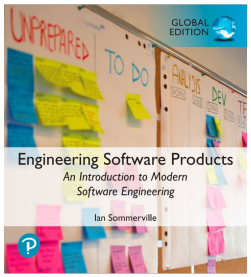
- The starting point for the software development is a set of 'software requirements' that are owned by an external client and which set out what they want a software system to do to support their business processes.
- The software is developed by a software company (the contractor) who design and implement 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



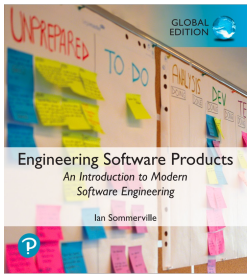


Product Software Engineering



- 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



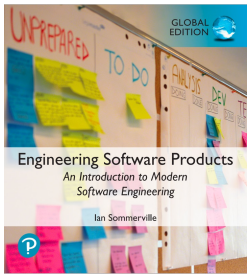
Software product line

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

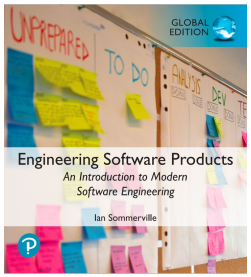
A software (or software+hardware) product that includes functionality so that new applications can be built on it. An example of a platform that you probably use is Facebook. It provides an extensive set of product functionality but also provides support for creating 'Facebook apps'. These add new features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

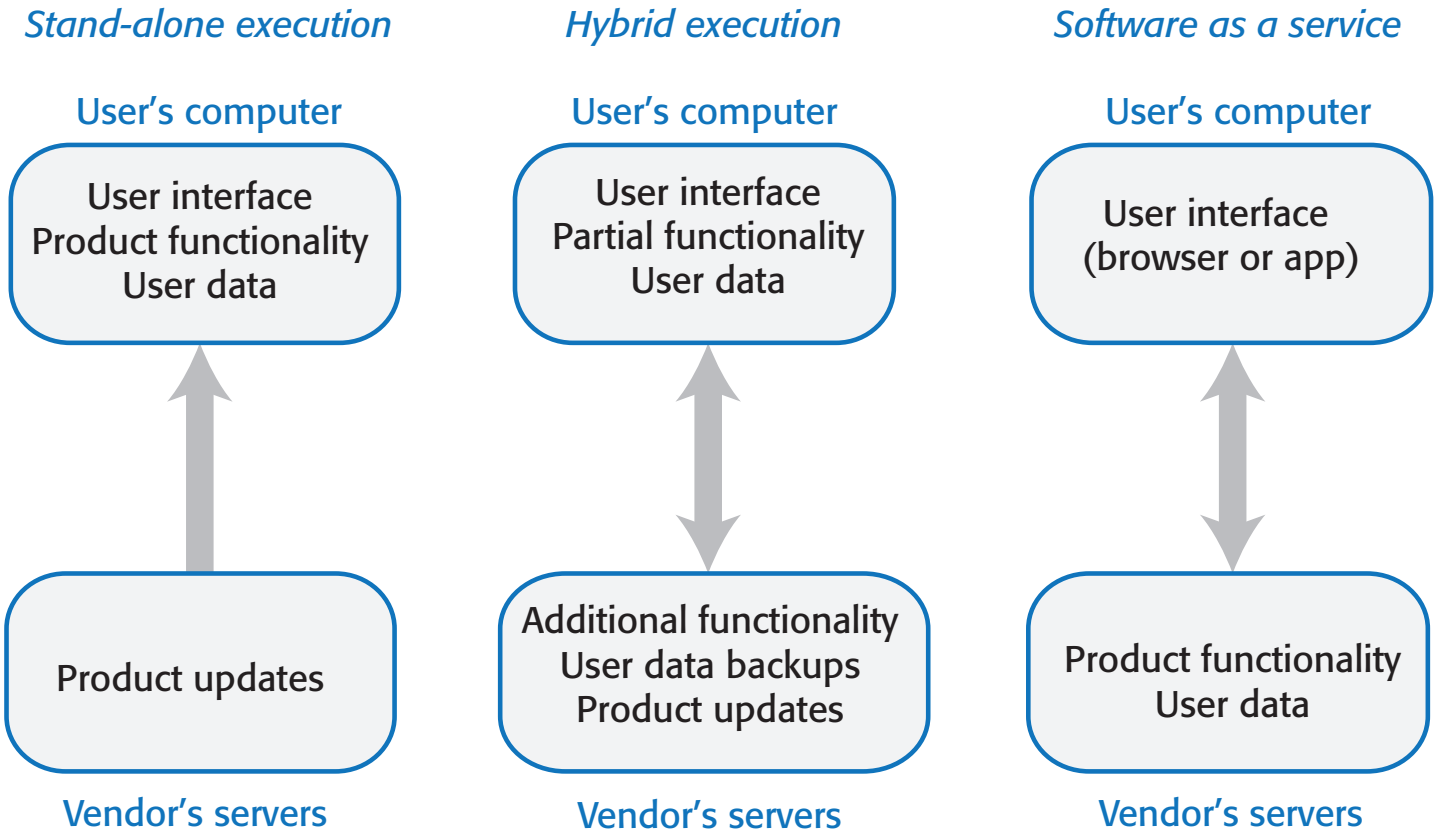
Software Execution Models



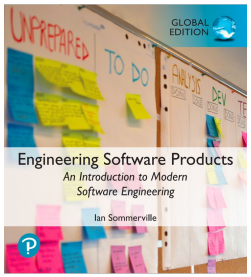
- **Stand-alone** The software executes entirely on the customer's computers.
- **Hybrid** Part of the software's functionality is implemented on the customer's computer but some features are implemented on the product developer's servers.
- **Software service** All of the product's features are implemented on the developer's servers and the customer accesses these through a browser or a mobile app.

Software Execution Models





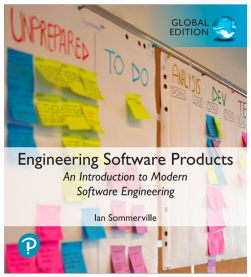
Comparable Software Development



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:

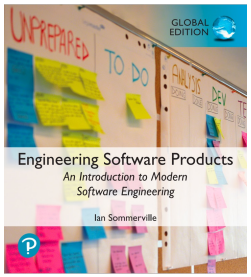
- **Student projects** Individuals or student groups develop software as part of their course. Given an assignment, they decide what features to include in the software.
- **Research software** Researchers develop software to help them answer questions that are relevant to their research.
- **Internal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

The Product Vision



- 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



- 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)

Reminds me of how to Write Research Paper Abstract ("selling"):

By Kent Beck (the founder of Extreme Programming/Agile) on "how to get a paper accepted at SPLASH/OOPSLA":

Abstract. The abstract is your four sentence summary of the conclusions of your paper. I try to have four sentences in my abstract.

The **first** states the problem.

The **second** states why the problem is a problem.

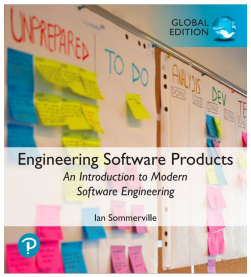
The **third** is my startling sentence.

The **fourth** states the implication of my startling sentence.

An abstract for this paper done in this style would be:

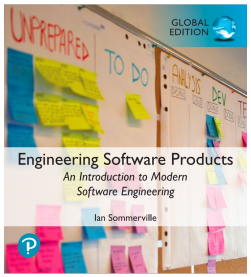
The rejection rate for OOPSLA papers in near 90%. Most papers are rejected not because of a lack of good ideas, but because they are poorly structured. Following four simple steps in writing a paper will dramatically increase your chances of acceptance. If everyone followed these steps, the amount of communication in the object community would increase, improving the rate of progress.

Vision Template Example



"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



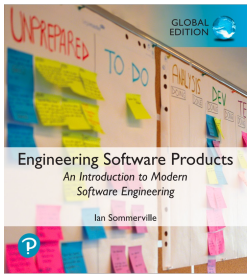
Domain experience

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



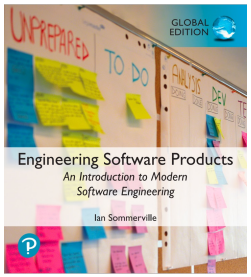
Customer experience

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.

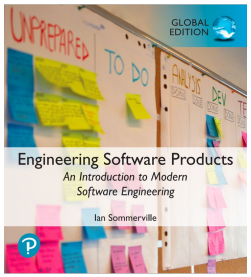
Vision Statement for the iLearn (ESP Book Running Example)



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.

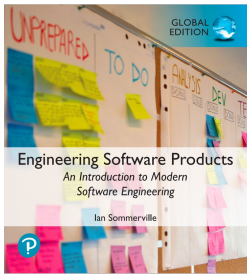
Schools and universities are the target customers for the iLearn system as it will significantly improve the learning experience of students at relatively low cost. It will collect and process learner analytics that will reduce the costs of progress tracking and reporting.

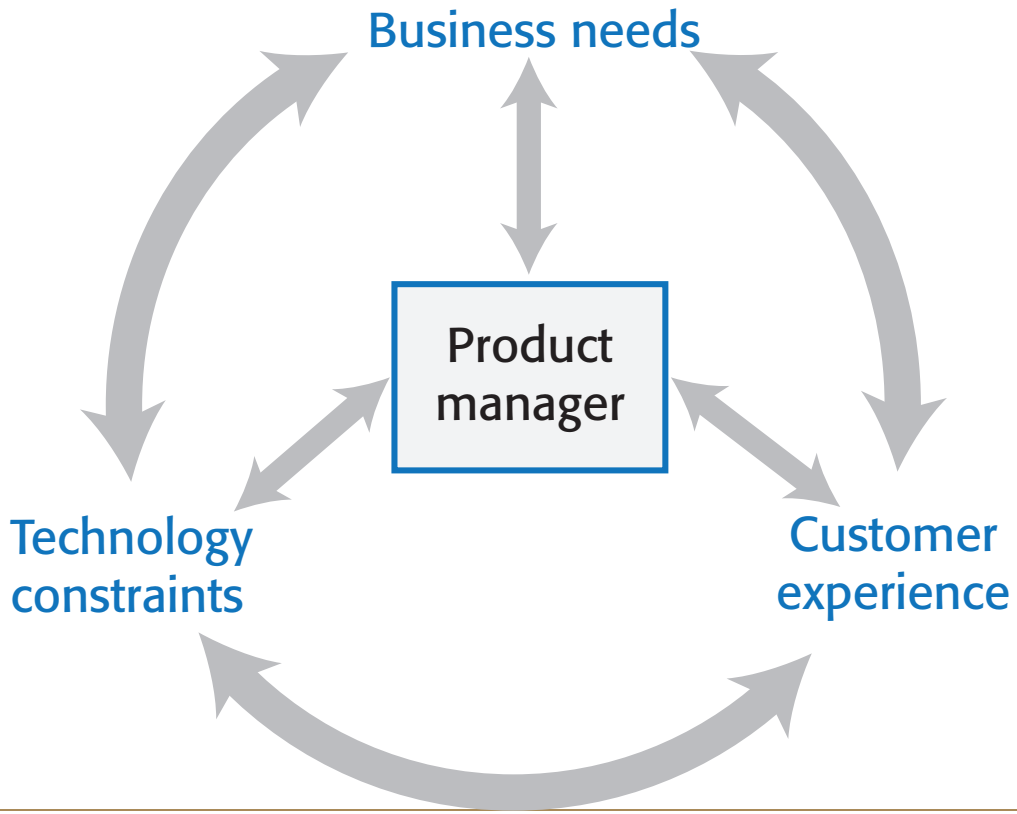
Software Product Management



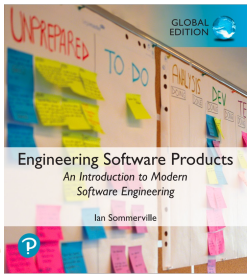
- Software product management is a business activity that focuses on the software products developed and sold by the business.
- Product managers (PMs) take overall responsibility for the product and are involved in planning, development and product marketing.
- Product managers are the interface between 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





Product Management Concerns

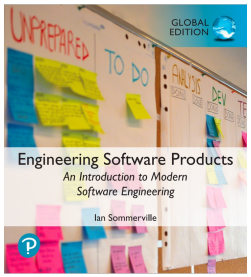


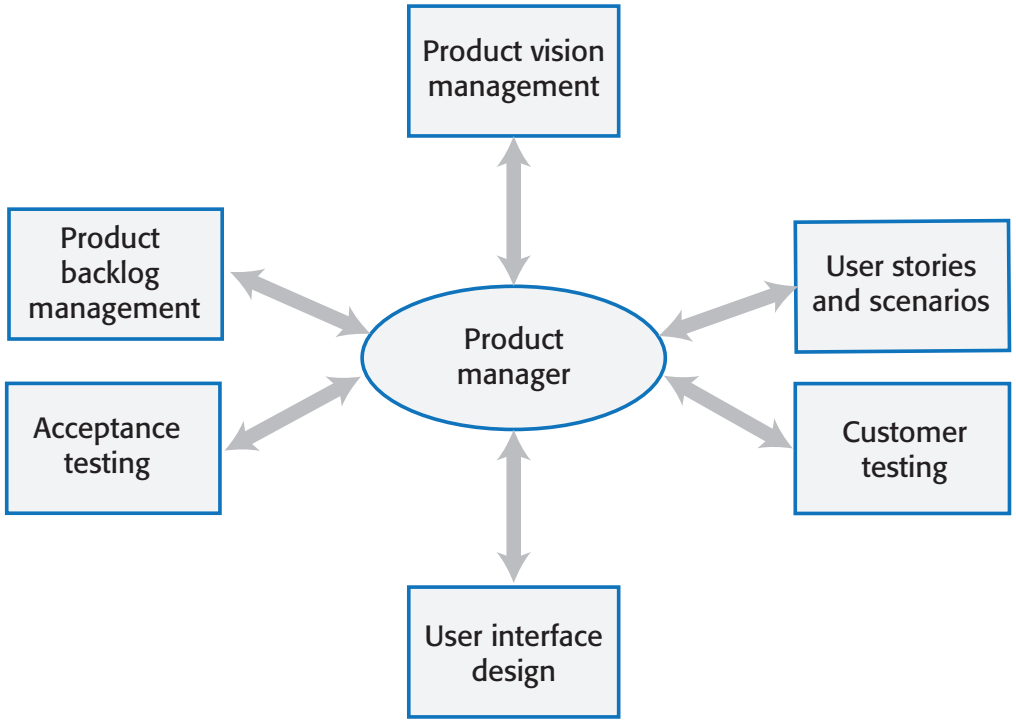
- **Business needs** PMs have to ensure that the software being developed meets the business goals of the software development company.

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

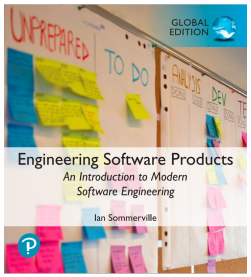
- **Customer experience** PMs should be in regular contact with customers and potential customers to understand what they are looking for in a product, the types of users and their backgrounds and the ways that the product may be used.

Technical Interactions of Product Managers



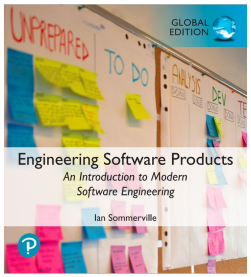


Technical Interactions of Product Managers



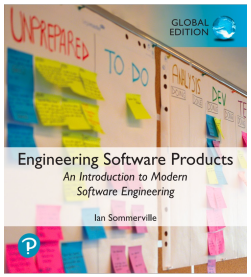
- Product vision management
  - The product manager may be responsible for helping with the development of the product vision. The should always be responsible for managing the vision, which involves assessing and evaluating proposed changes against the product vision. They should ensure that there is 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



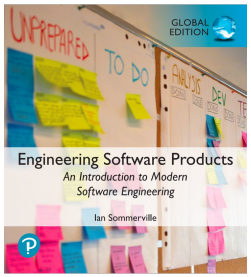
- User story and scenario development
  - User stories and scenarios are used to refine a product vision and identify product features. Based on his or her knowledge of customers, the PM should lead the development of stories and scenarios.
- Product backlog creation and management
  - The product backlog 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



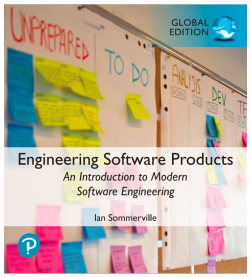
- Customer testing
  - Customer testing involves taking a release of a product to customers and getting feedback on the product's features, usability and business. PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.
- User 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



- **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.

Access & Inclusion

Are a team of DisAbility and Equity Advisors who support ANU students whose participation in academic studies is impacted by:

- Disability ~ physical or learning
- mental health condition/s
- ongoing chronic medical condition/s,
- short term illness/ injury.

As well as:

- Carers

If your circumstances are listed above and you require support to achieve your academic goals, please visit the Access and Inclusion website to find out about registering.

For further information:

💻: http://www.anu.edu.au/students/health-wellbeing/diversity-inclusion

✉: access.inclusion@anu.edu.au

A&I Special (Alternative) Exam Arrangements

It is the student's responsibility to ensure that they have a valid Education Access Plan (EAP) in place with A&I at least 2 weeks prior to examination periods.

Deadlines to renew EAPs and request SEAs for 2021:

Semester 1:

- Mid-semester exams: 12 March 2021
- End of semester exams: 14 May 2021

Semester 2:

- Mid-semester exams: 13 August 2021
- End of semester exams: 15 October 2021

- Students should be aware that failure to inform A&I within this timeframe will result in SEAs not being implemented for the upcoming examination period.
- New registrations after the deadline will be considered on a case-by-case basis

FOR FURTHER INFORMATION:

💻: HTTP://WWW.ANU.EDU.AU/STUDENTS/HEALTH-WELLBEING/DIVERSITY-INCLUSION

✉: ACCESS.INCLUSION@ANU.EDU.AU

CECS 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 (CECS).

The role of Class 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.

Roles and responsibilities:

- Act as the official liaison between your peers and convener.
- Be creative, 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.

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.

- 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 CECS representatives: sa.cecs@anu.edu.au
- PARSA CECS representatives: parsa.cecs@anu.edu.au

Want to be a class representative? Nominate today!

Please nominate yourself to your course convener by **5 August 2022**.

You are free to nominate yourself whether you are currently on-campus or studying remotely.

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 Friday 5th of August 2022*
- If more than 4 people apply, I may (or may not depending on numbers) do a Piazza Poll and that will close at 23:59 Monday 8th of August in time for Tuesday lecture…

Australian National University

Staying COVID safe on campus

An overview of measures we've put in place and things you need to do to stay COVID safe on campus.

Stay home if you're sick – get tested and follow health advice

Keep your vaccinations up to date

Wear a mask inside, as required

Keep your distance from others where possible

Hand hygiene – use hand sanitiser or wash hands frequently

Get tested if you've been exposed or are symptomatic

Tell us if you test positive – notify us using the form on the ANU COVID-19 website and let your supervisor or course convenor know too

Improved ventilation - we're modifying the systems in our buildings which includes changing CO2 setpoints, introducing more fresh air and having it circulated more frequently, and upgrading our air filtration systems. More detailed information is available on the ANU COVID-19 website

Key dates

- **Monday 25 July** – Semester 2 begins
- **Monday 1 August** – last day to add Semester 2 courses on ISIS
- **Wednesday 31 August** – Semester 2 census date
- **Friday 28 October** – Semester 2 ends
- **Thursday 3 - Saturday 19 November** – Semester 2 examination period

Events

We're running a full program of COVID safe events, check out www.anu.edu.au/events for more

Stay up to date

Visit www.anu.edu.au/covid-19-advice and read On Campus (in your inbox every Tuesday)

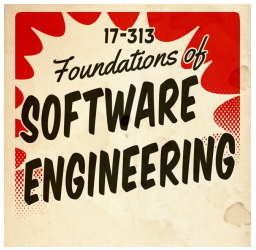
BE KIND TO EACH OTHER — LOOK AFTER YOURSELVES — REACH OUT FOR SUPPORT

Mini Break in Monday Lecture

Learning Goals

- Recognize the importance of process
- Understand the difficulty of measuring progress
- Identify why software development has project characteristics
- Use milestones for planning and progress measurement
- Understand backlogs and user stories
- Know your team!

Software Process

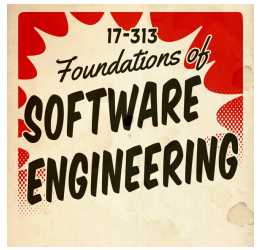


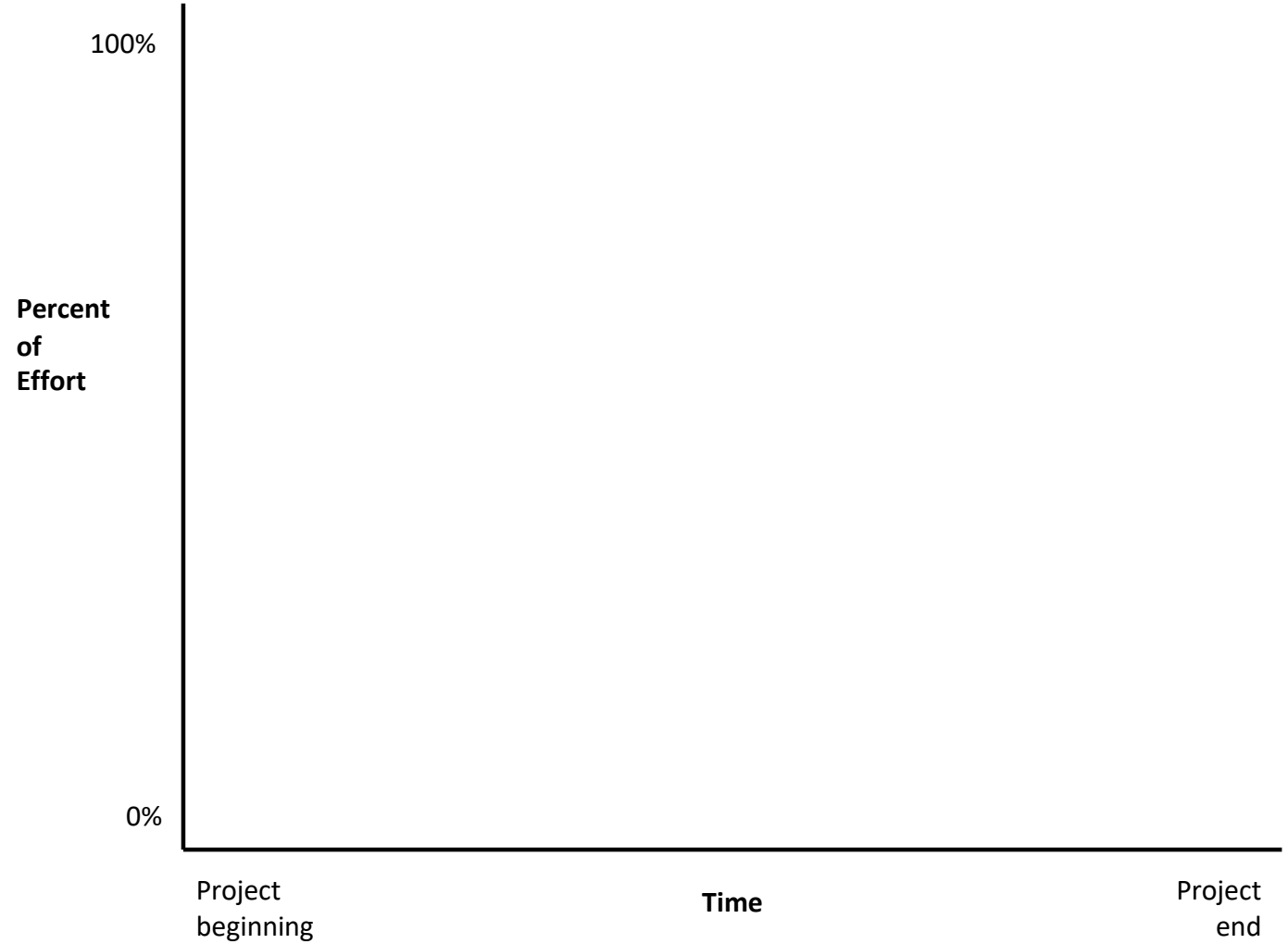
"The set of activities and associated results that produce a software product"

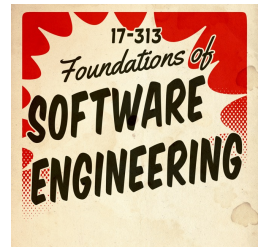
Sommerville, SE, ed. 8

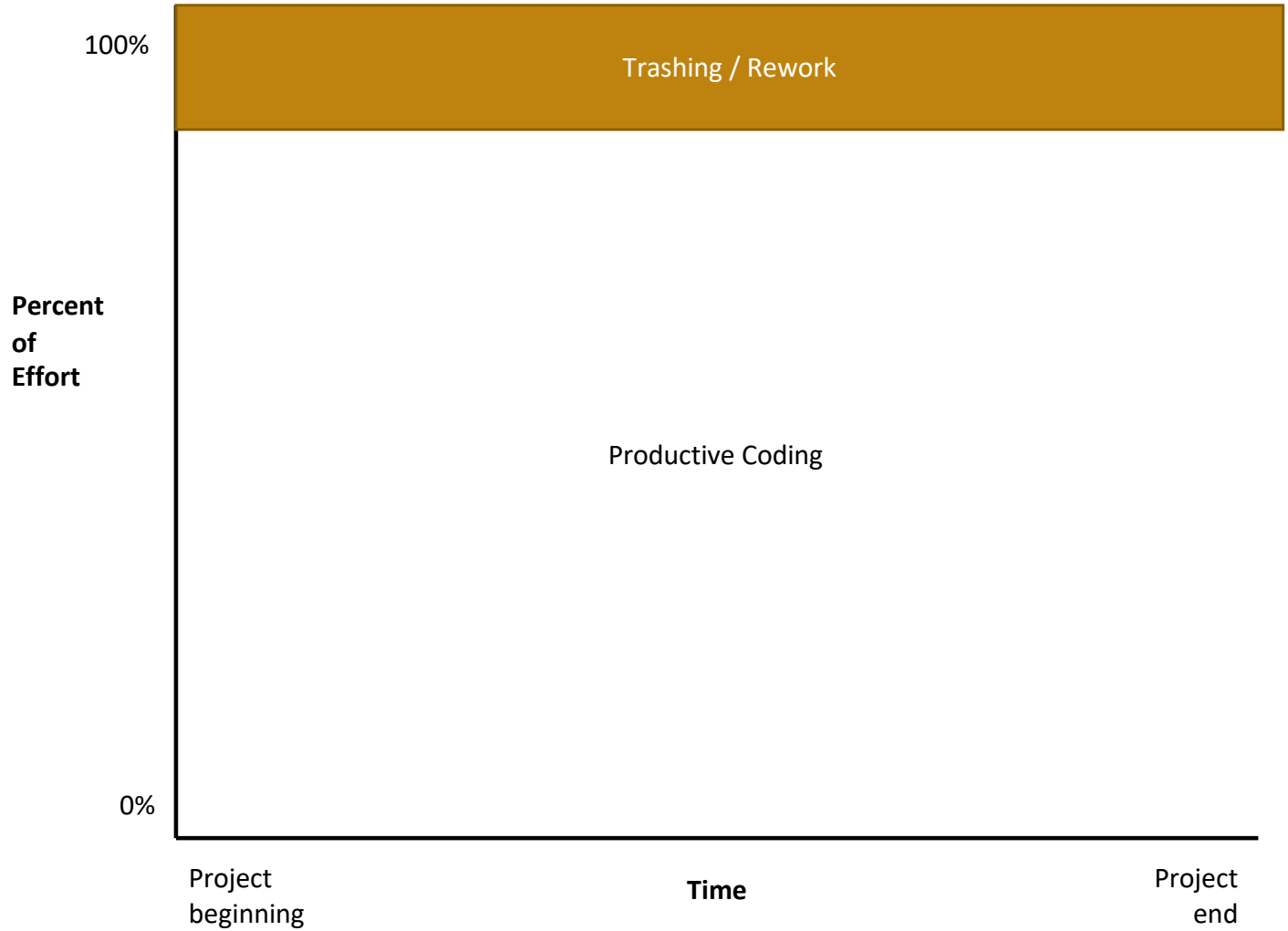
How to develop software?

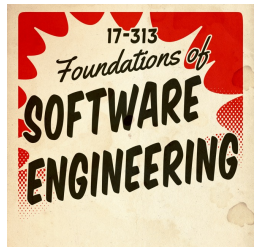
1. Discuss the software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1


17-313
Foundations of
SOFTWARE
ENGINEERING

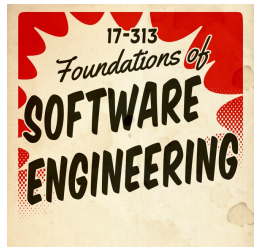

100%
Percent
of
Effort
0%
Project
beginning
Time
Project
end

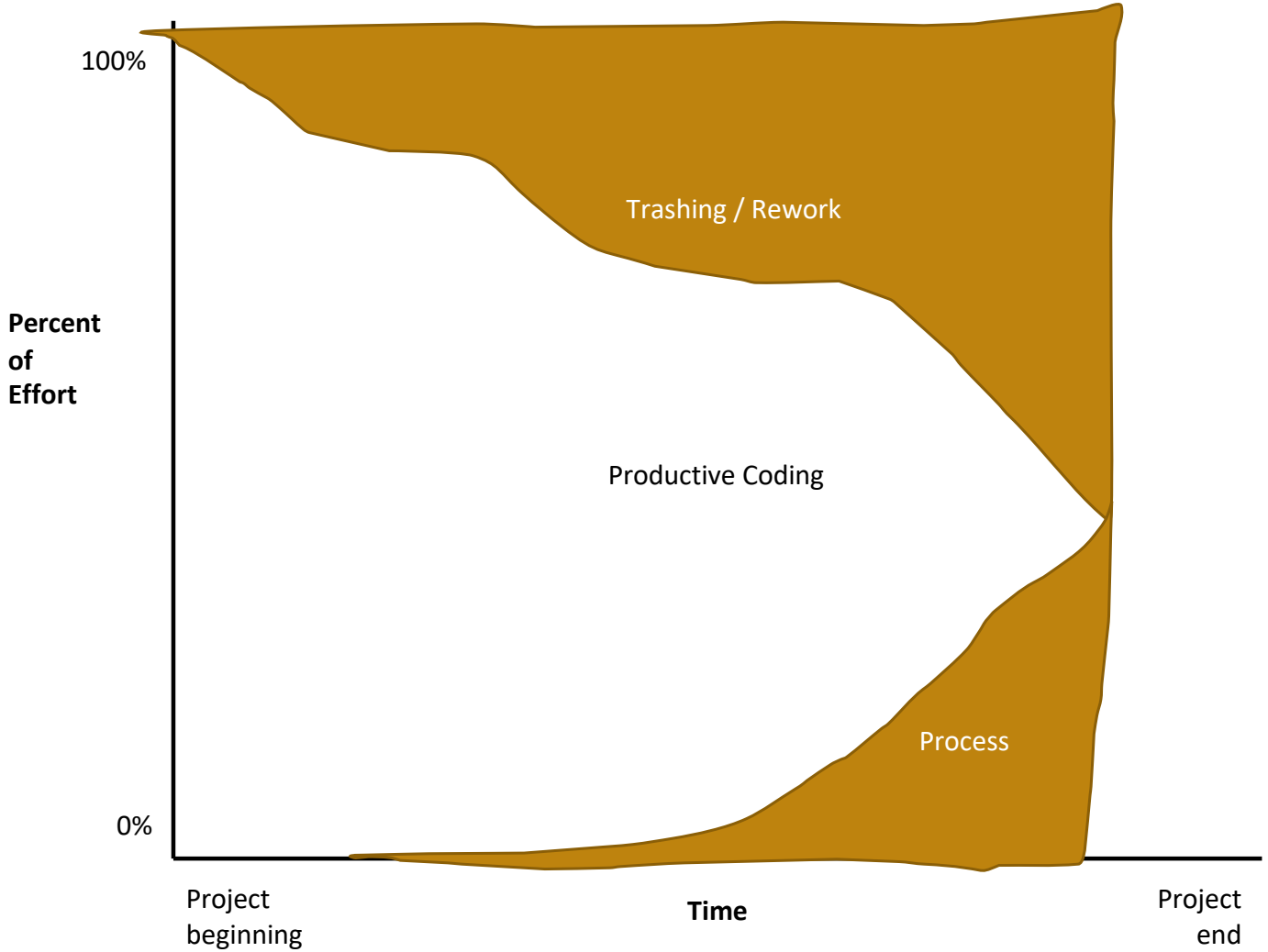

17-313
Foundations of
SOFTWARE
ENGINEERING


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

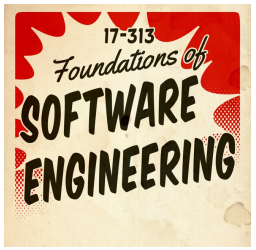

17-313
Foundations of
SOFTWARE
ENGINEERING


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


17-313
Foundations of
SOFTWARE
ENGINEERING

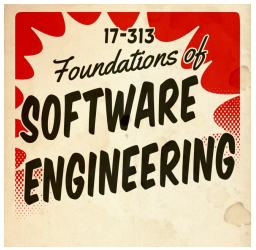

100%
Trashing / Rework
Percent
of
Effort
Productive Coding
Process
0%
Project
beginning
Time
Project
end

Example of Process Decisions

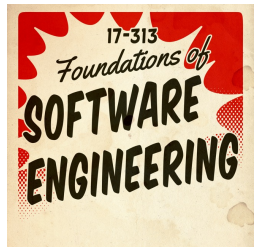


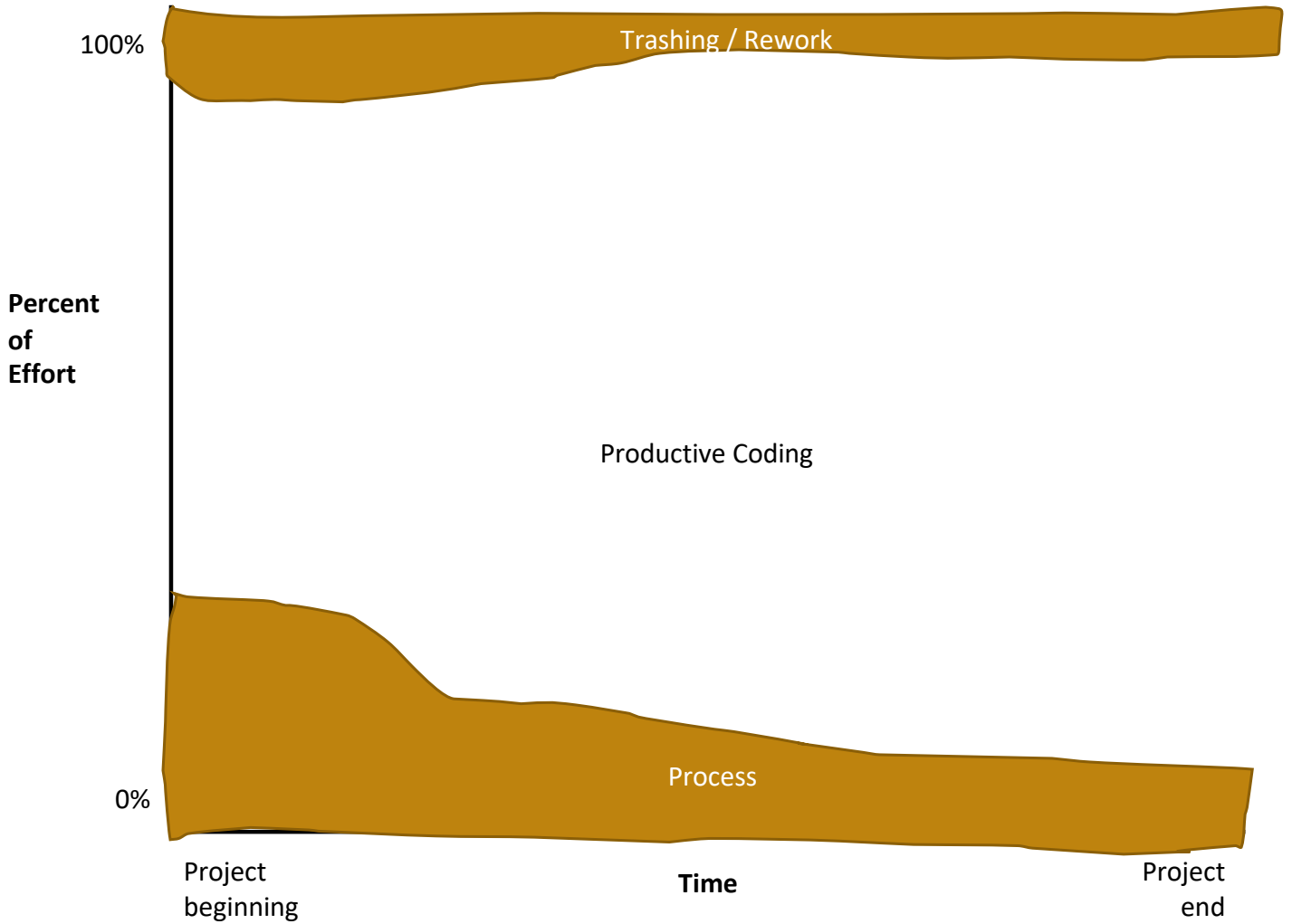
- Writing down all requirements
- Require approval for all changes to requirements
- Use version control for all changes
- Track all reported bugs
- Review requirements and code
- Break down development into smaller tasks and schedule and monitor them
- Planning and conducting quality assurance
- Have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

Example process issues

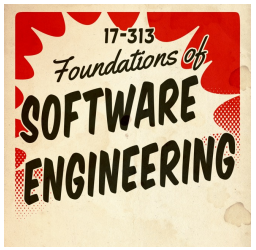


- Change Control: Mid-project informal agreement to changes suggested by customer or manager. Project scope expands 25-50%
- Quality Assurance: Late detection of requirements and design issues. Test-debug-reimplement cycle limits development of new features. Release with known defects.
- Defect Tracking: 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.

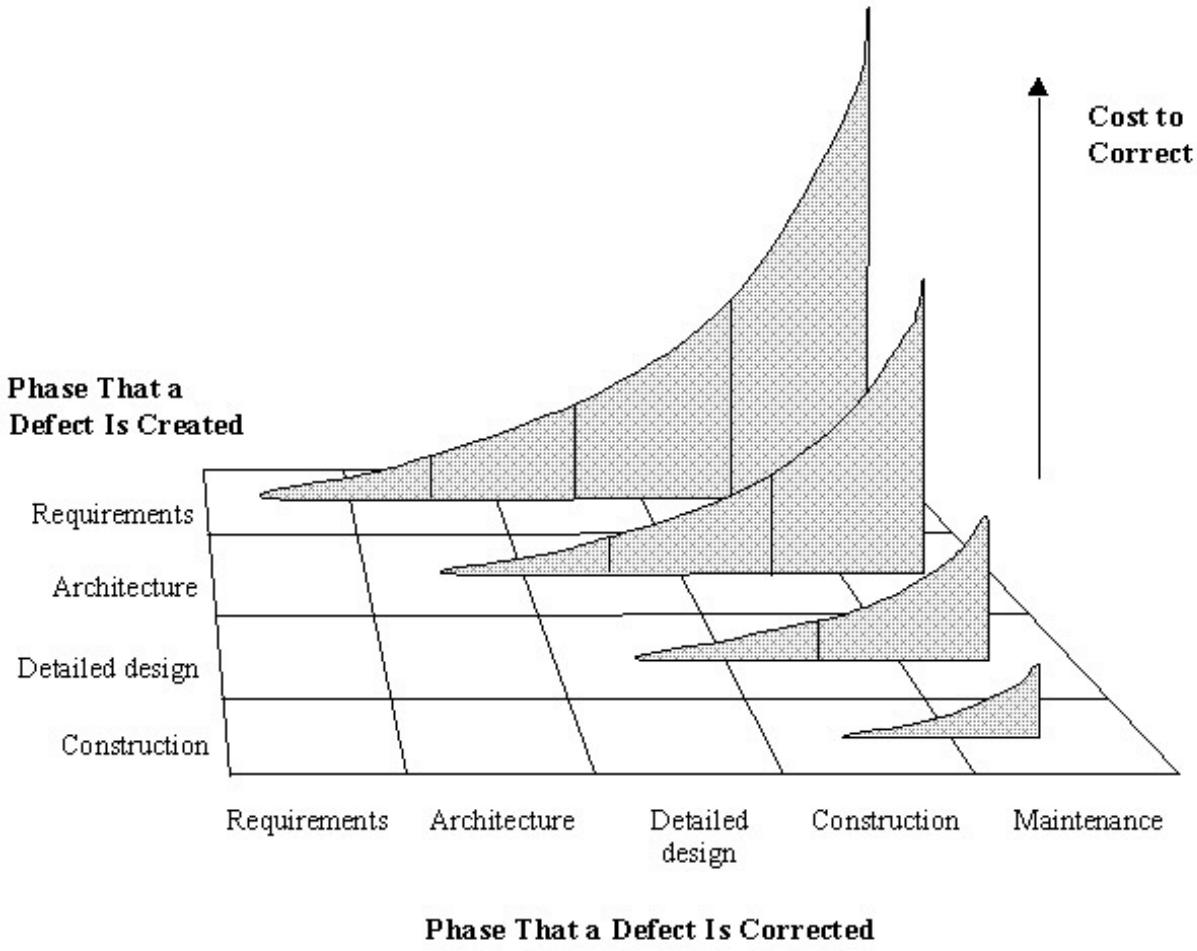

17-313
Foundations of
SOFTWARE
ENGINEERING

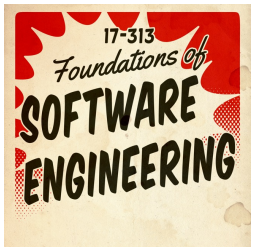

100%
Trashing / Rework
Percent
of
Effort
Productive Coding
Process
0%
Project
beginning
Time
Project
end

Hypothesis



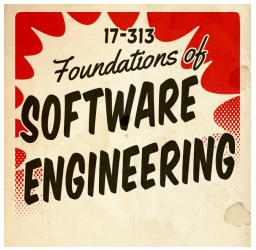
- Process increases flexibility and efficiency
- Upfront investment for later greater returns





Estimating Effort

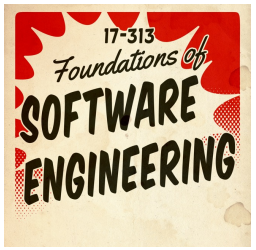
Task: Estimate Time



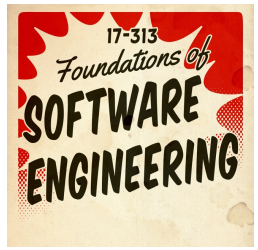
- A: Simple web version of the Monopoly boardgame with local street names
  - Team: just you
- B: Bank smartphone app
  - Team: you with team of 4 developers, one experienced with iPhone apps, one with background in security

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

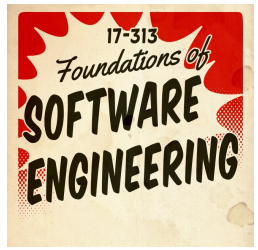
Revise Time Estimate



- 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

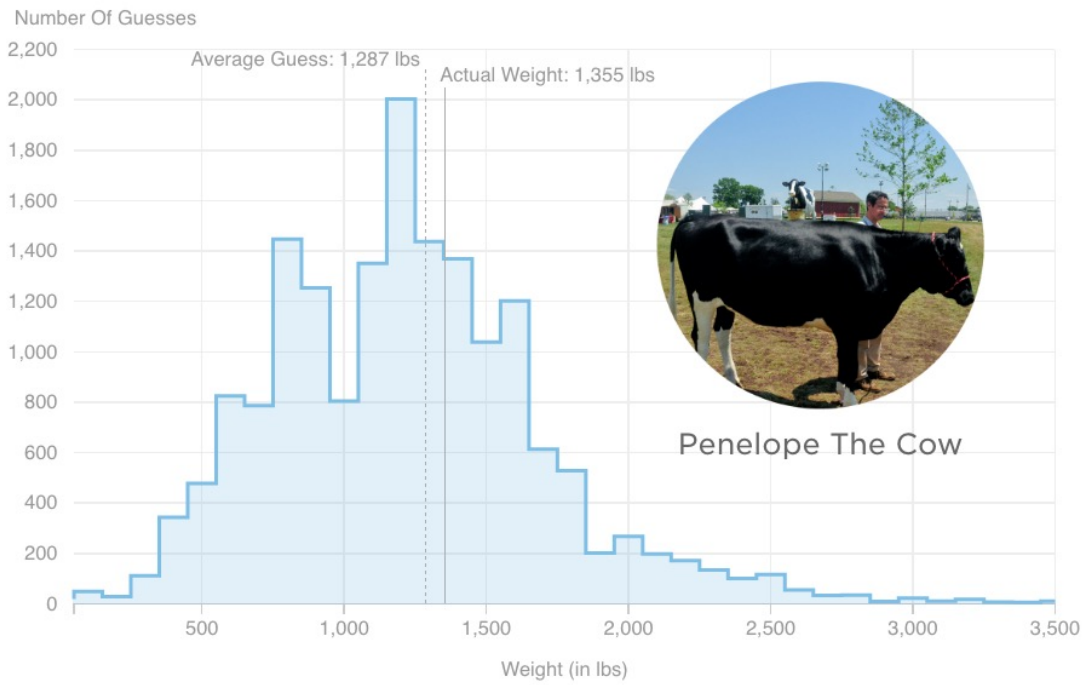


π



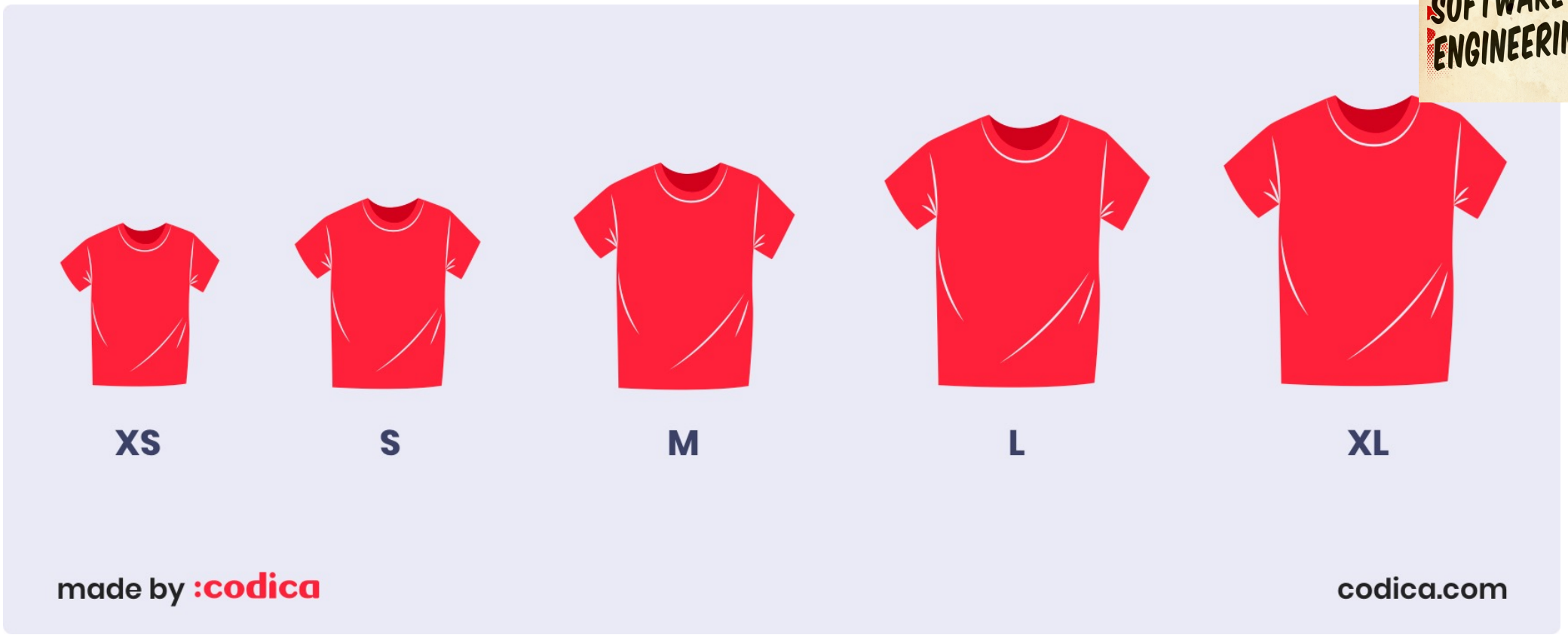
How Much Does This Cow Weigh?

(All People)

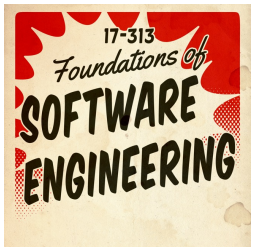


Source: The Internet.

Credit: Quoctrung Bui/NPR



Measuring Progress?

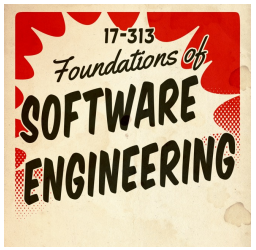


- “I’m almost done with the app. The frontend is almost fully implemented. The backend is fully finished except for the one stupid bug that keeps crashing the server. I only need to find the one stupid bug, but that can probably be done in an afternoon. We should be ready to release next week.”

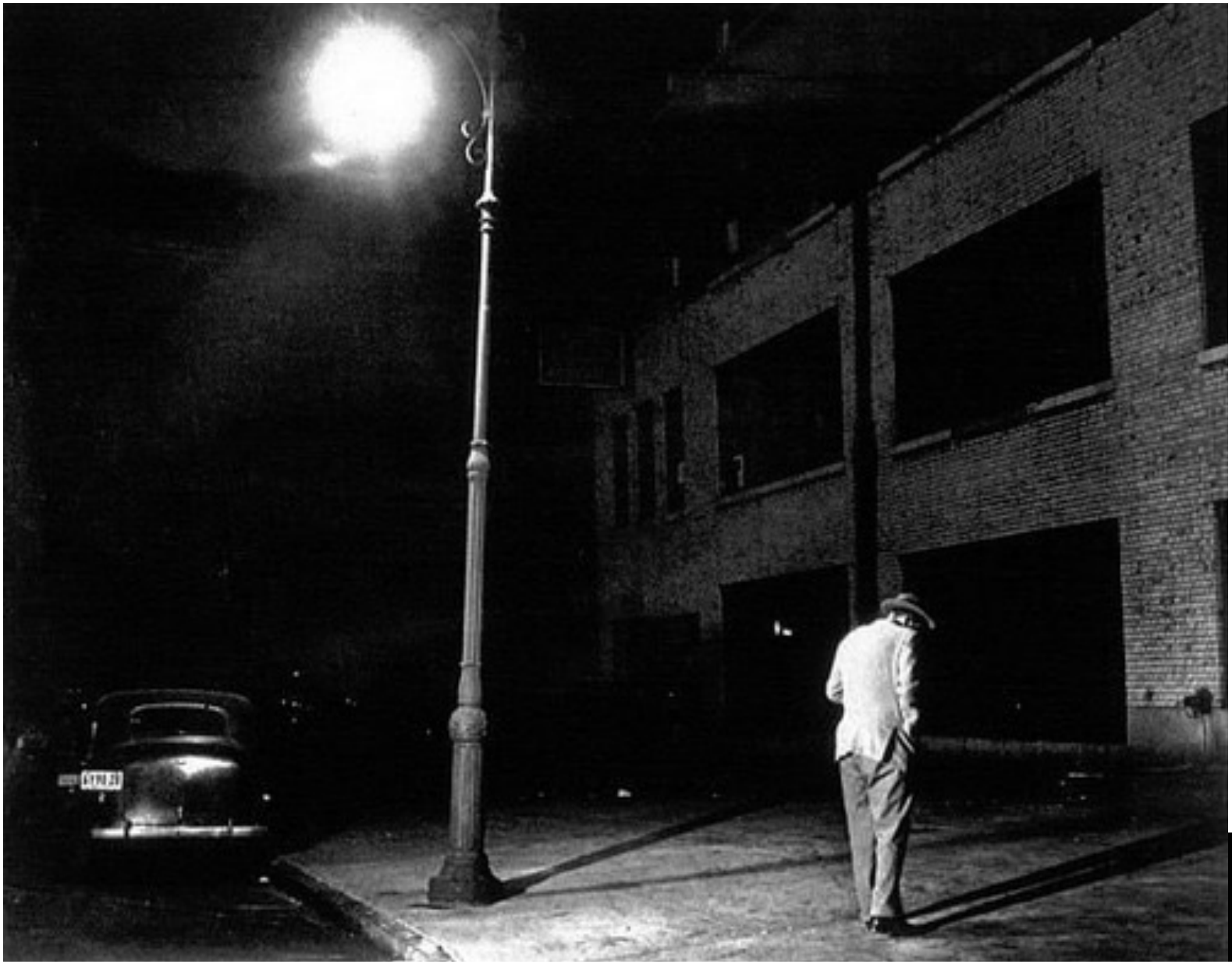
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

https://xkcd.com/612/

Measuring Progress?

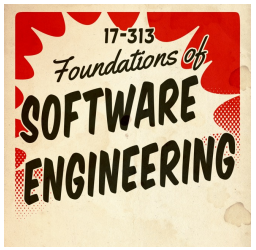


- Developer judgment: x% done
- Lines of code?
- Functionality?
- Quality?



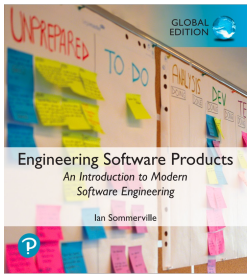
17-313
Foundations of
SOFTWARE
ENGINEERING

Milestones and deliverables



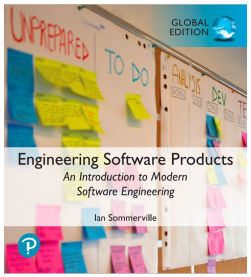
- Making progress observable, especially for software
- Milestone: clear end point of a (sub)tasks
  - For project manager
  - Reports, prototypes, completed subprojects
  - "80% done" not a suitable mile stone
- Deliverable: Result for customer
  - Similar to mile stone, but for customers
  - Reports,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

Agile Software Engineering



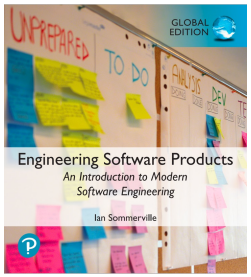
- Software products must be brought to market quickly so rapid software development and delivery is essential.
- Virtually all software products are now developed using an agile approach.
- Agile software engineering focuses on delivering functionality quickly, responding to changing product specifications 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



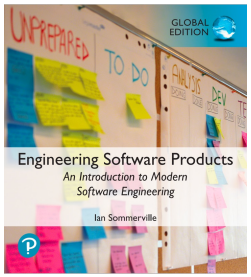
- Plan-driven development evolved to support the engineering of large, long-lifetime systems (such as aircraft control systems) where teams may be geographically dispersed and work on the software for several years.
  - This approach is based on controlled and rigorous software development processes 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



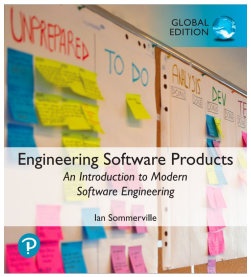
- We are uncovering better ways of developing software by doing it and helping others to do it. Through this work, we have come to value:
  - individuals and interactions over processes and tools;
  - working software over comprehensive documentation;
  - customer collaboration over contract negotiation;
  - 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.

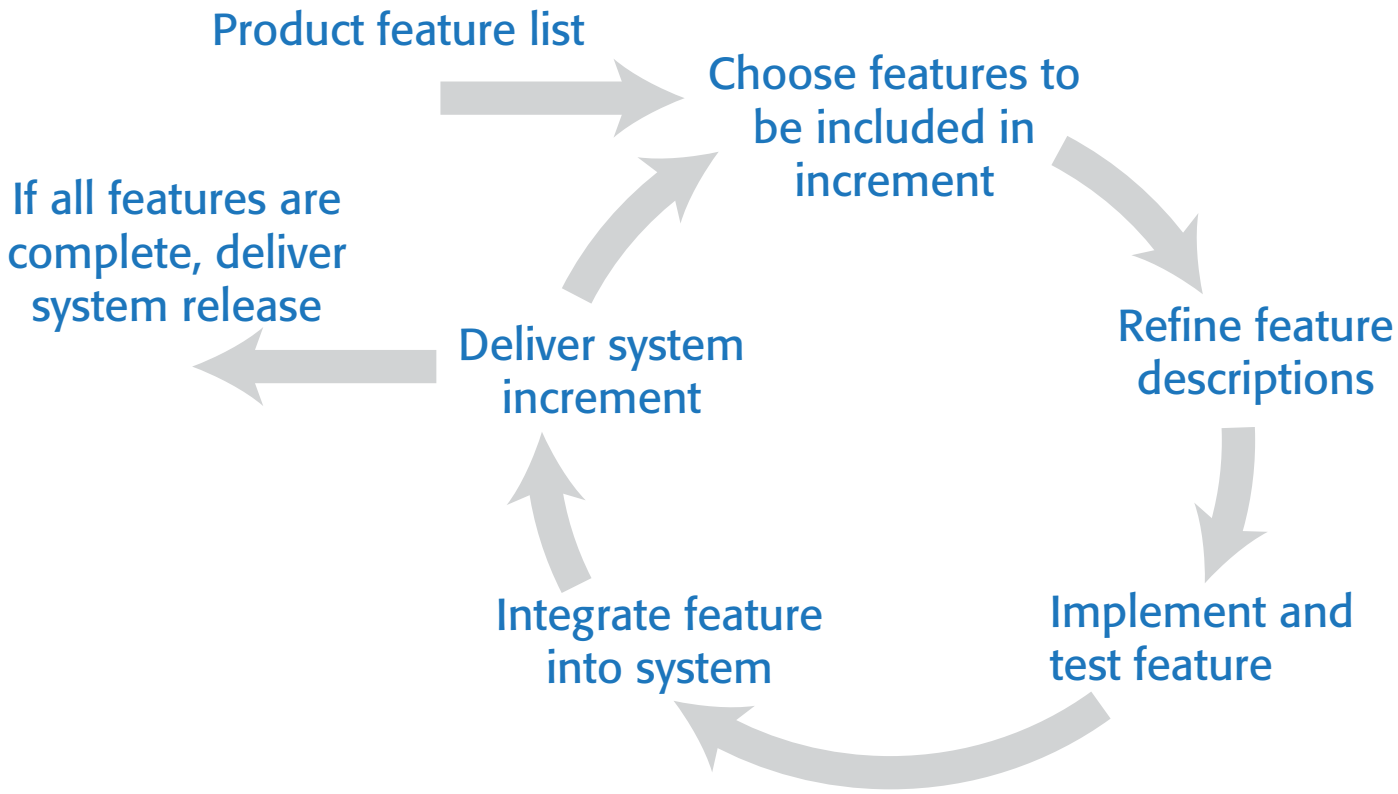
Incremental Development



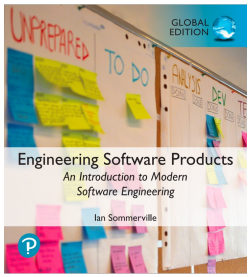
- All agile methods are based around incremental development and delivery.
- Product development focuses on the software features, where a feature does something for the software user.
- With incremental development, you start by prioritizing the features so that the most important features are implemented 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





Incremental Development Activities



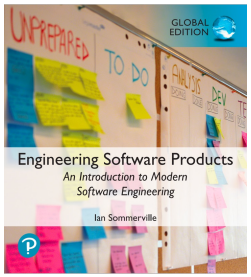
Choose features to be included in an increment

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



Implement and test

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

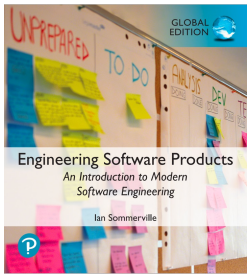
Integrate feature and test

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



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.

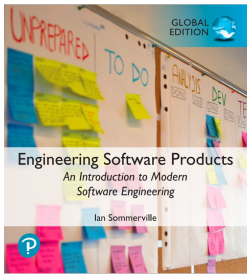
Embrace change

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



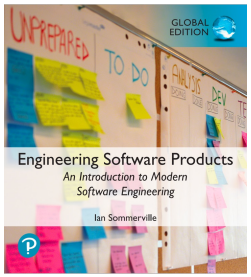
Maintain simplicity

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

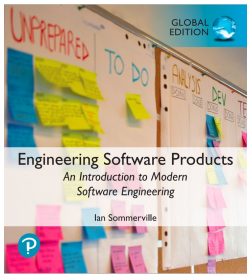
Trust the development team and do not expect everyone to always do the development process in the same way. Team members should be left to develop their own ways of working without being limited by prescriptive software processes.

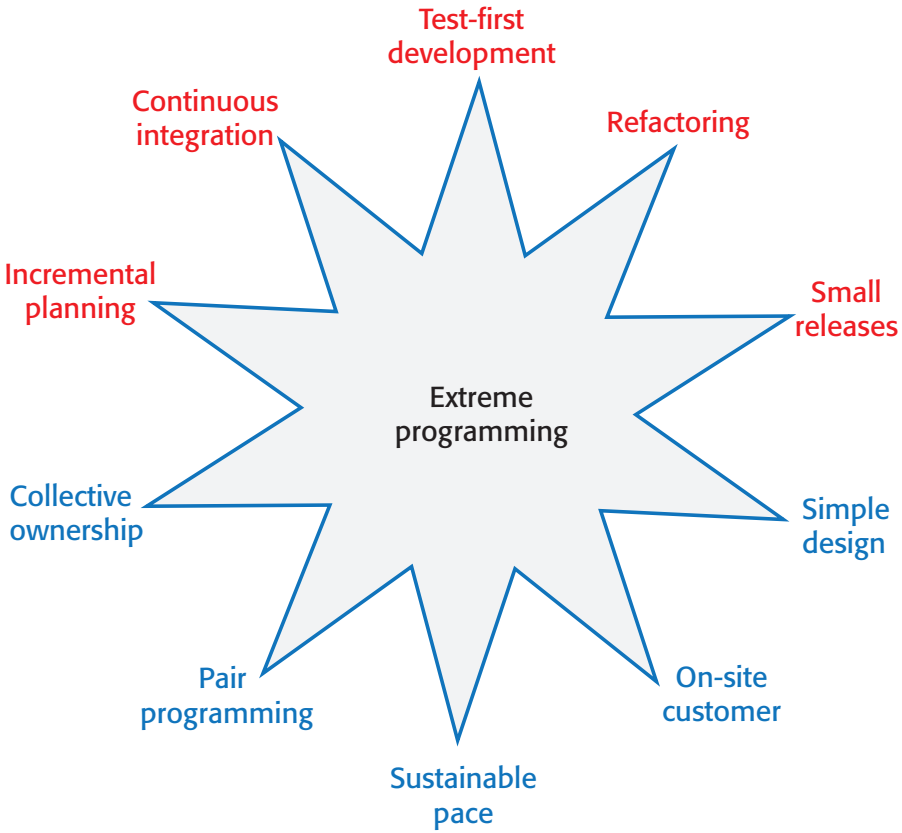
Extreme Programming (XP)



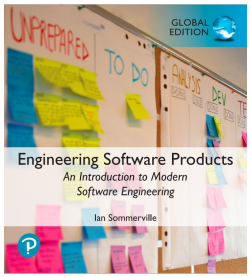
- The most influential work that has changed software development culture was the development of Extreme Programming (XP).
- The name was coined by Kent Beck in 1998 because the approach was developed by pushing recognized good practice, such as iterative development, to 'extreme' levels.
- Extreme programming 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.
- The most widely used XP techniques (highlighted in red on the following slide) are explained elsewhere in the book.

Extreme Programming Practices





Widely Adopted XP Practices



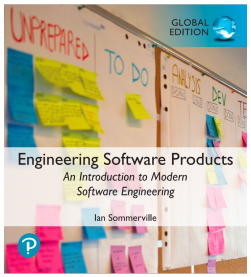
Incremental planning/user stories

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



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

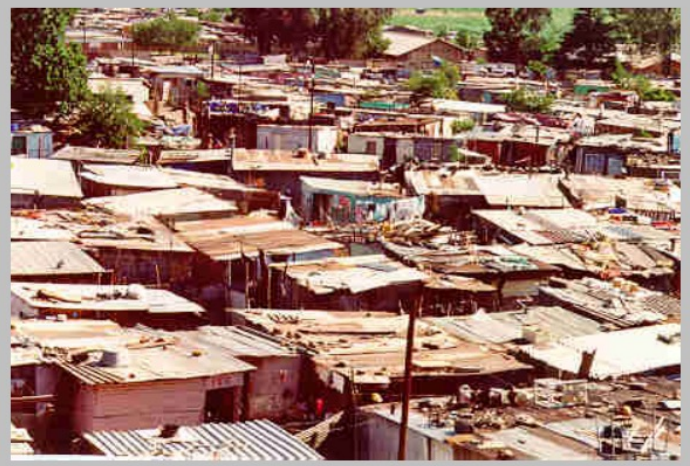
Refactoring

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

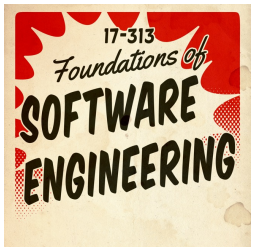
alias
SHANTYTOWN
SPAGHETTI CODE

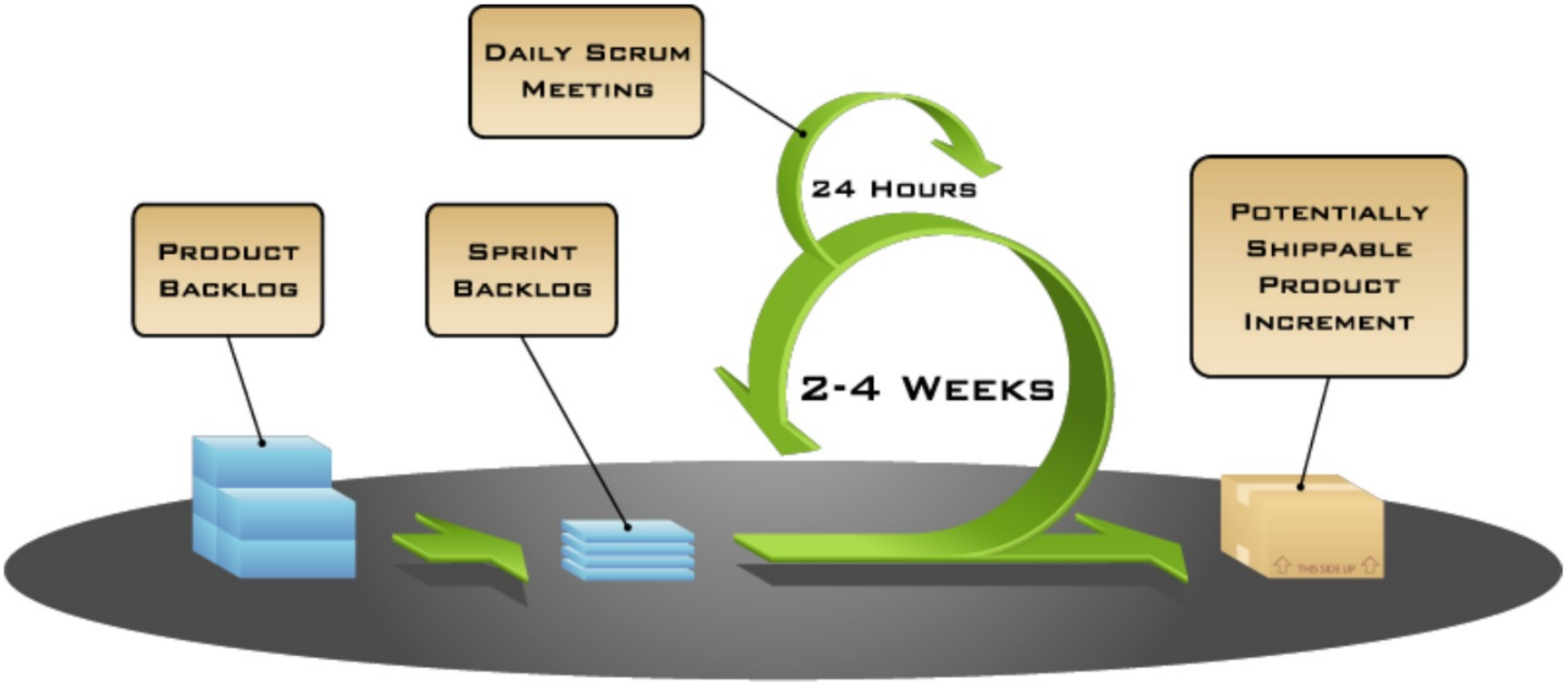


Shantytowns are squalid, sprawling slums. Everyone seems to agree they are a bad idea, but forces conspire to promote their emergence anyway. What is it that they are doing right?

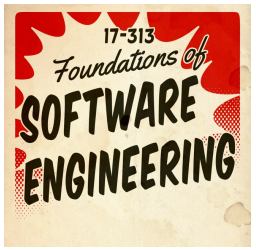
Shantytowns are usually built from common, inexpensive materials and simple tools. Shantytowns can be built using relatively unskilled labor. Even though the labor force is "unskilled" in the customary sense, the construction and maintenance of this sort of housing can be quite labor intensive. There 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.

Brief intro to Scrum





Elements of Scrum

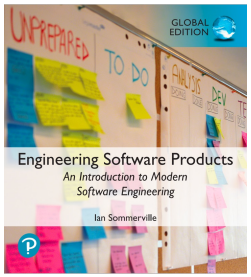


- Products:
  - Product Backlog
  - Sprint Backlog
- Process:
  - Sprint Planning Meeting
  - Daily Scrum Meeting
  - Sprint Retrospective
  - Sprint Review Meeting

Scrum

- Software company managers need information that will help them understand how much it costs to develop a software product, how long it will take and when the product can be brought to market.
- Plan-driven development provides this information through long-term development plans that identify deliverables - items the team will deliver and when these will be delivered.
- Plans always change so anything apart from short-term plans are unreliable.
- Scrum is an agile method that provides a framework for agile project organization and planning. It does not mandate any specific technical practices.

Scrum Terminology



Product

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

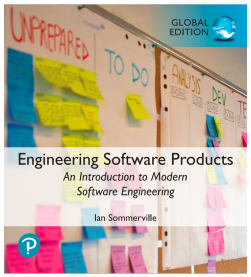
Product owner

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.

Scrum Terminology



Development team

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

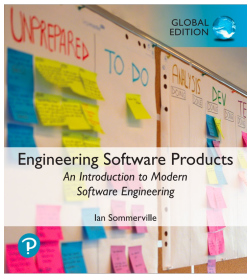
Sprint

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.

Scrum Terminology



ScrumMaster

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

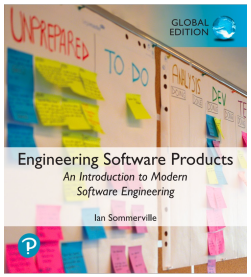
Potentially shippable product increment

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



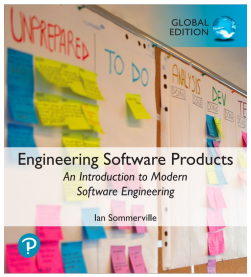
The Product Owner is responsible for ensuring that the development team are always focused on the product they are building rather than diverted into technically interesting but less relevant work.

- 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.

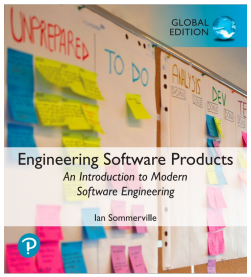
- In many companies that use Scrum, the ScrumMaster also has some project management responsibilities.

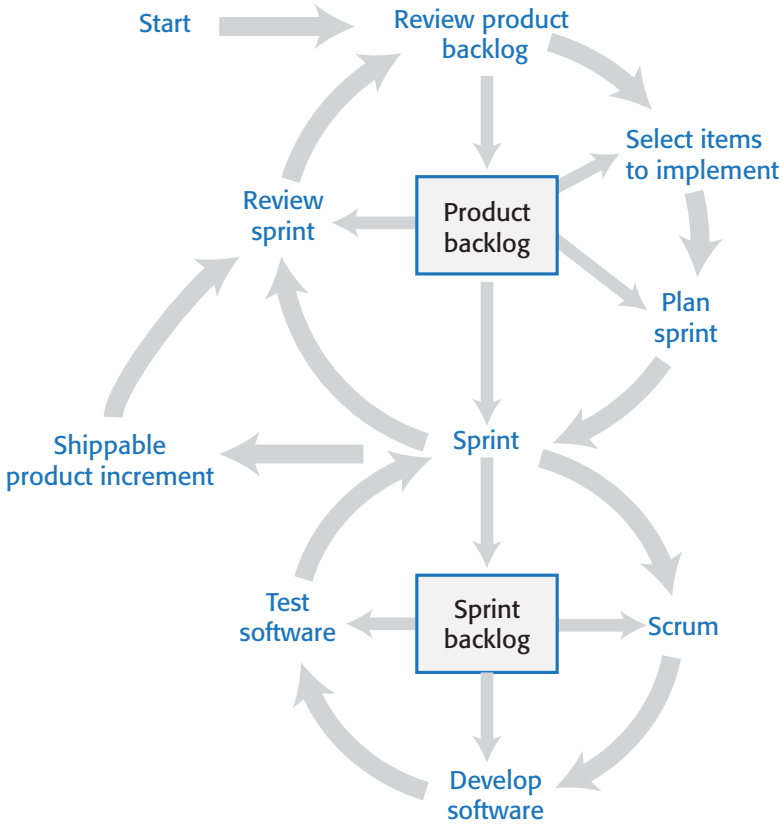
Scrum and Sprints



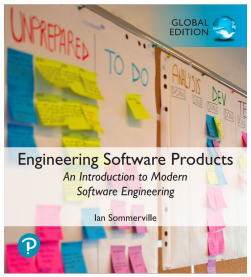
- In Scrum, software is developed in sprints, which are fixed-length periods (2 - 4 weeks) in which software features are developed and delivered.
- During a sprint, the team has daily meetings (Scrums) to review progress and to update the list of work items that are incomplete.
- Sprints should produce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

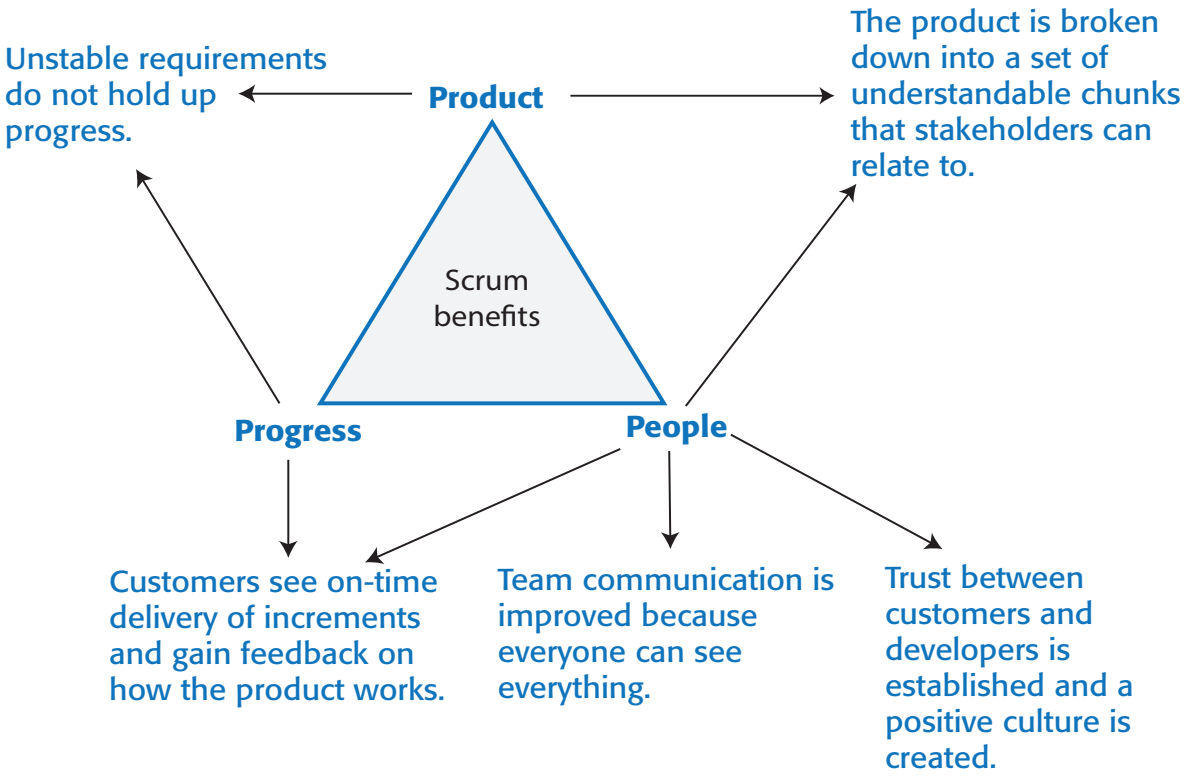
Scrum Cycles



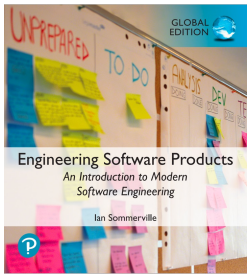


Top Five Benefits of Using Scrum





Key Scrum Practices



Product backlog

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

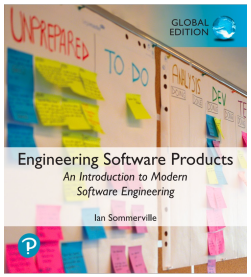
Timeboxed sprints

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

Self-organizing teams

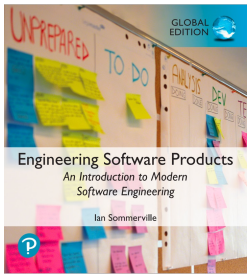
Self-organizing teams make their own decisions and work by discussing issues and making decisions by consensus.

Product Backlogs



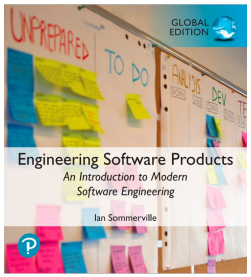
- The product backlog is a list of what needs to be done to complete the development of the product.
- The items on this list are called product backlog items (PBIs).
- The product backlog may include a variety of different items such as product features to be implemented, user requests, essential development 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



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



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

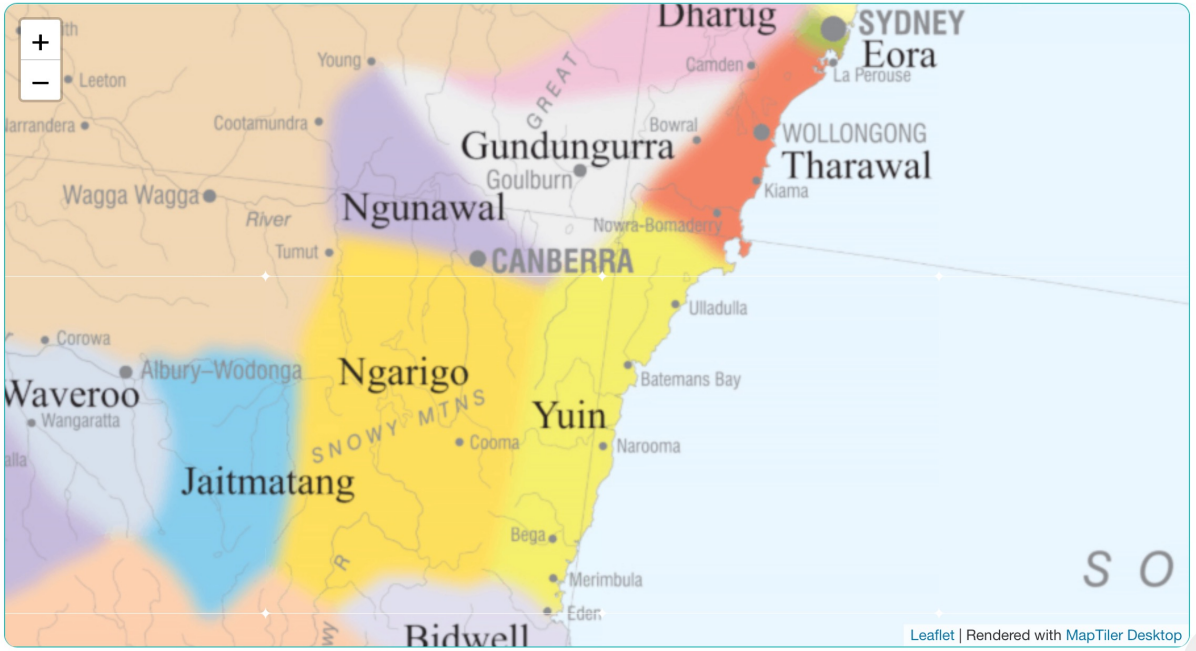
The PBI has enough detail for the team to estimate the effort involved and to implement the item. Dependencies on other items have been identified.

End of Monday Lecture/Start of Tuesday Lecture

ANU Acknowledgment of Country

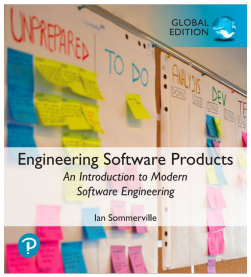
AIATSIS Explore Family history Collection Research Education What's new About Shop Search

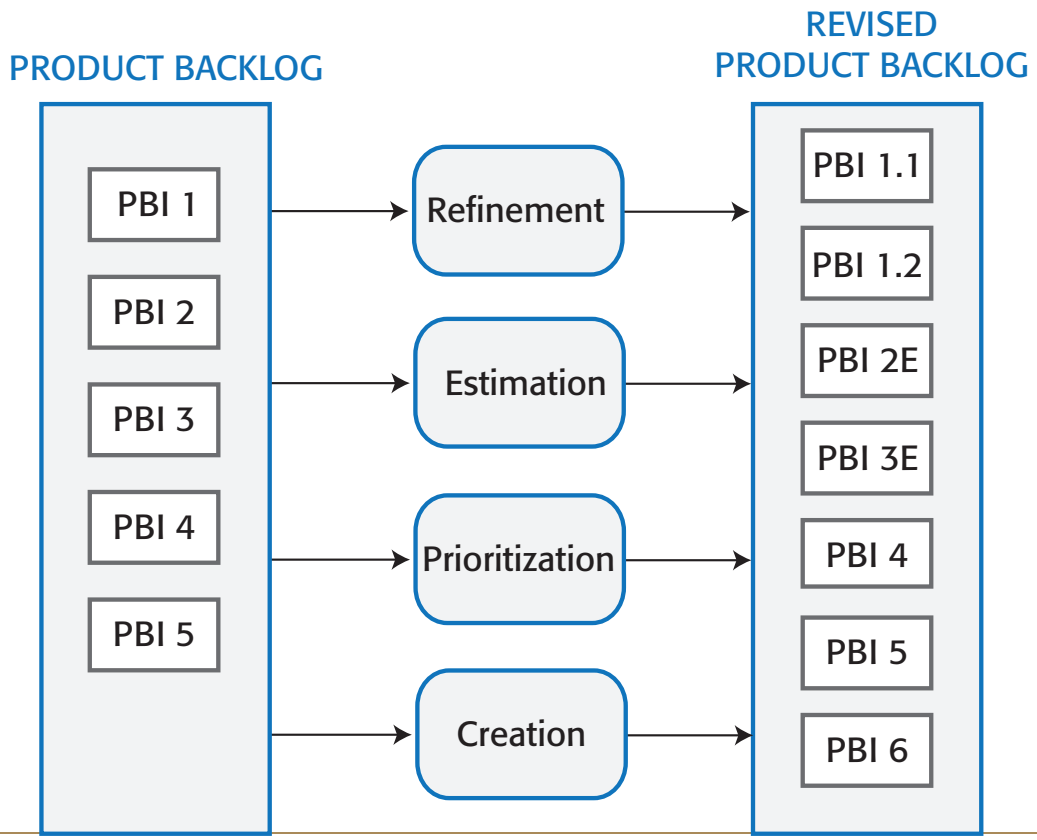
"We acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay our respect to the elders past and present."



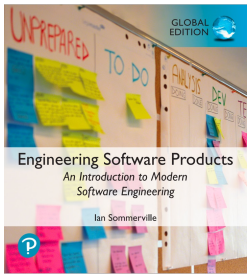
https://aiatsis.gov.au/explore/map-indigenous-australia

Product Backlog Activities





Product Backlog Activities



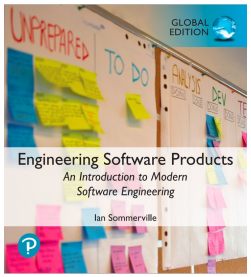
Refinement

Existing PBIs are analysed and refined to create more detailed PBIs. This may lead to the creation of new product backlog items.

Estimation

The team estimate the amount of work required to implement a PBI and add this assessment to each analysed PBI.

Product Backlog Activities



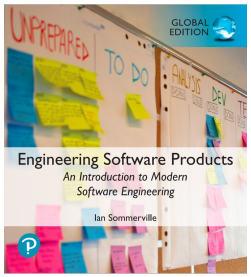
Creation

New items are added to the backlog. These may be new features suggested by the product manager, required feature changes, engineering improvements, or process activities such as the assessment of development tools that might be used.

Prioritization

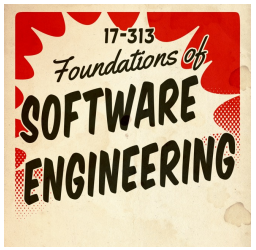
The product backlog items are reordered to take new information and changed circumstances into account.

PBI Estimation Metrics



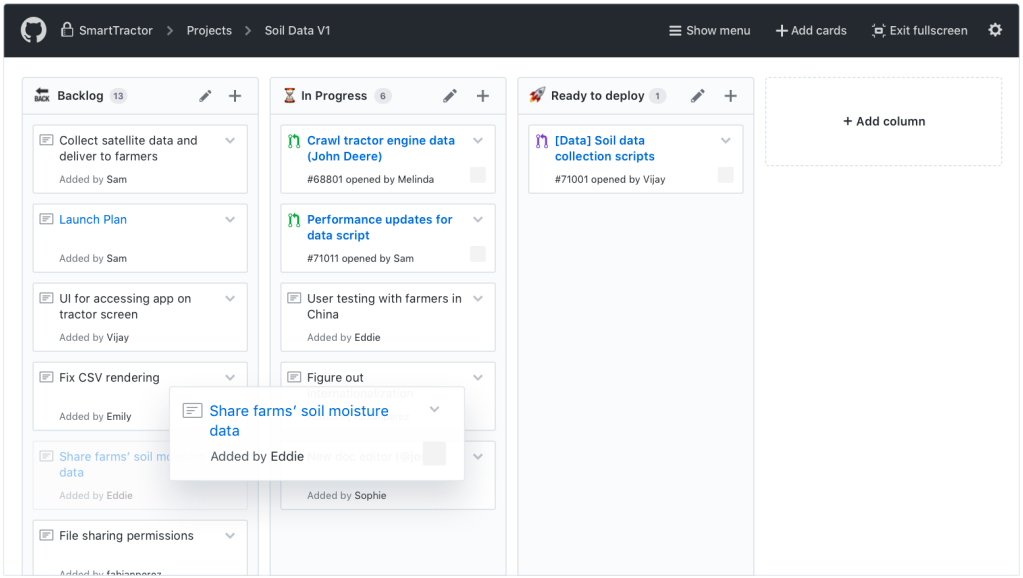
- Effort required
  - This may be expressed in person-hours or person-days i.e. the number of hours or days it would take one person to implement that PBI. This is not the same as calendar time. Several people may work on an item, which may shorten the calendar time required.
- Story points
  - Story points are an arbitrary estimate of the effort involved in implementing a PBI, taking into account the size of the task, its complexity, the technology that may be required and the 'unknown' characteristics of the work.
  - They were derived originally by comparing user stories, but they can be used for estimating any kind of PBI.
  - Story points are estimated relatively. The team agree on the story points for a baseline task and other tasks are estimated by comparison with this e.g. more/less complex, larger/smaller etc.

Product Backlog/Sprint Backlog

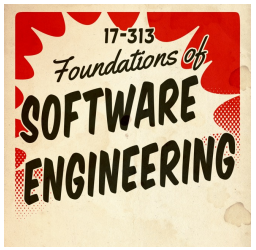


- The product backlog is all the features for the product
- The sprint backlog is all the features that will be worked on for that sprint. These should be broken down into discrete tasks:
  - Fine-grained
  - Estimated
  - Assigned to individual team members
  - Acceptance criteria should be defined
- User Stories are often used

Backlog – information radiators

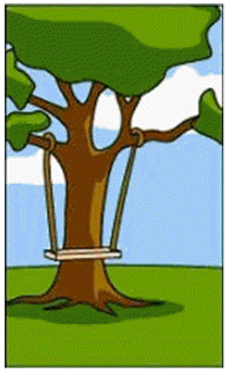


Scrum Meetings

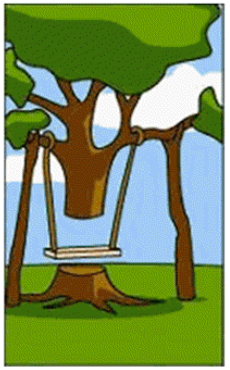


- Sprint Planning Meeting
  - Entire Team decides together what to tackle for that sprint
- Daily Scrum Meeting
  - Quick Meeting to touch base on :
    - What have I done? What am I doing next? What am I stuck on/need help?
- Sprint Retrospective
  - Review sprint process
- Sprint Review Meeting
  - Review Product

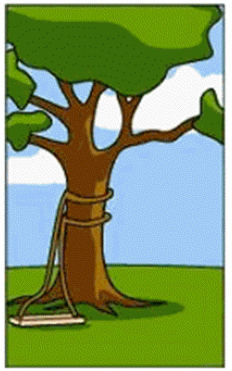
How the customer explained it



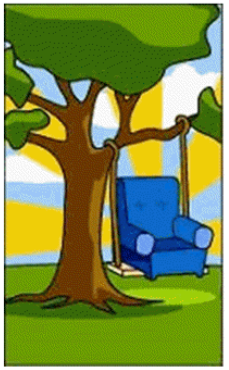
How the project leader understood it



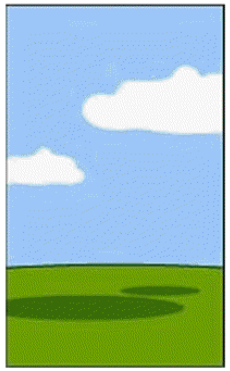
How the engineer designed it



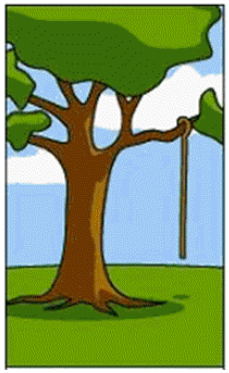
How the programmer wrote it



How the sales executive described it

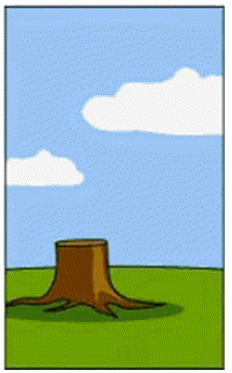


How the project was documented

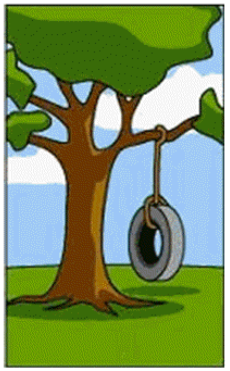


What operations installed

How the customer was billed

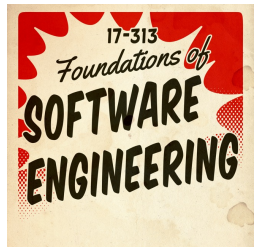


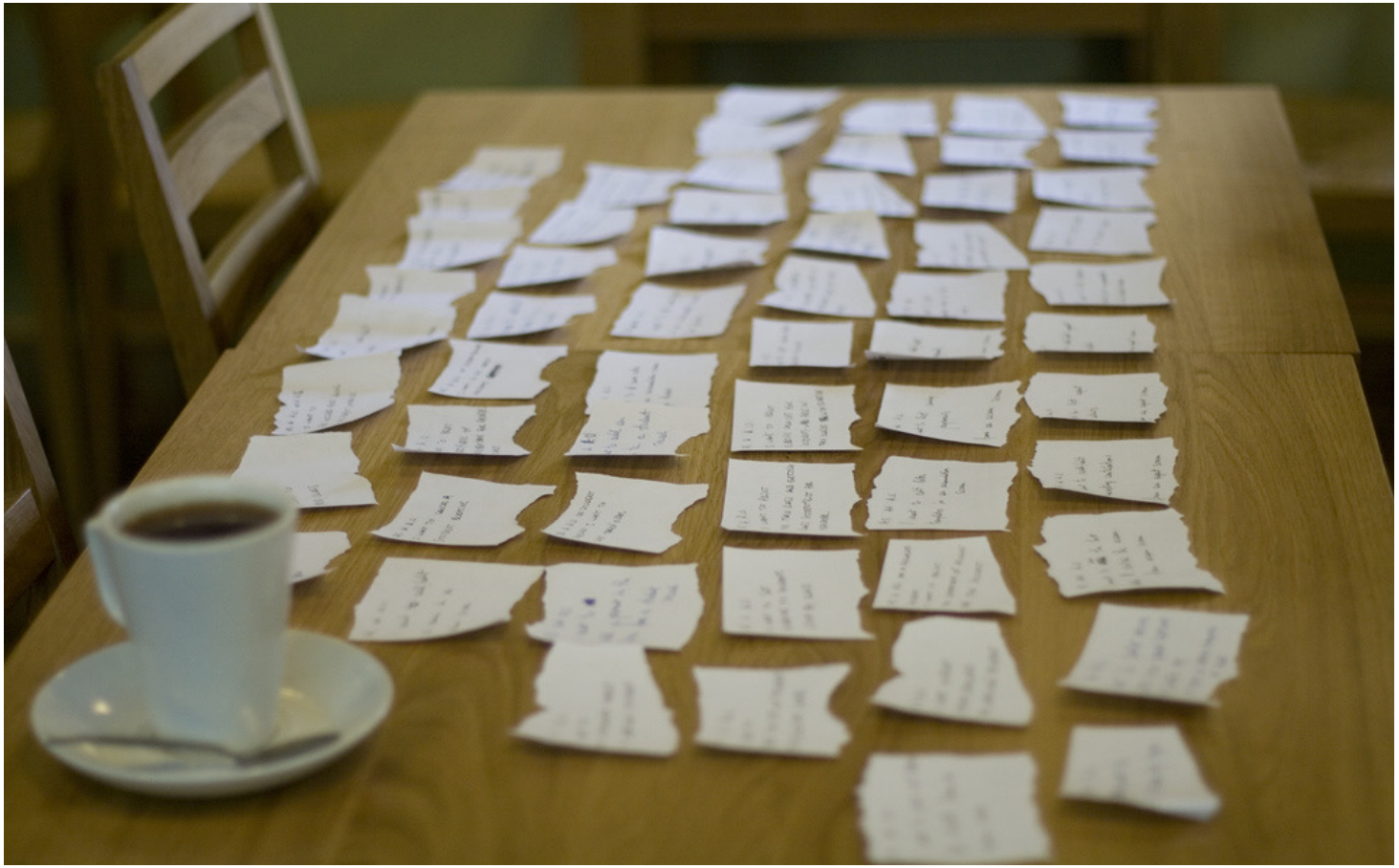
How the help desk supported it



What the customer really needed

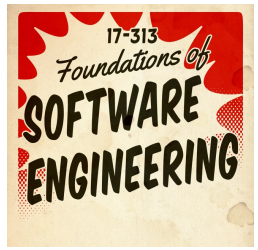
User Stories

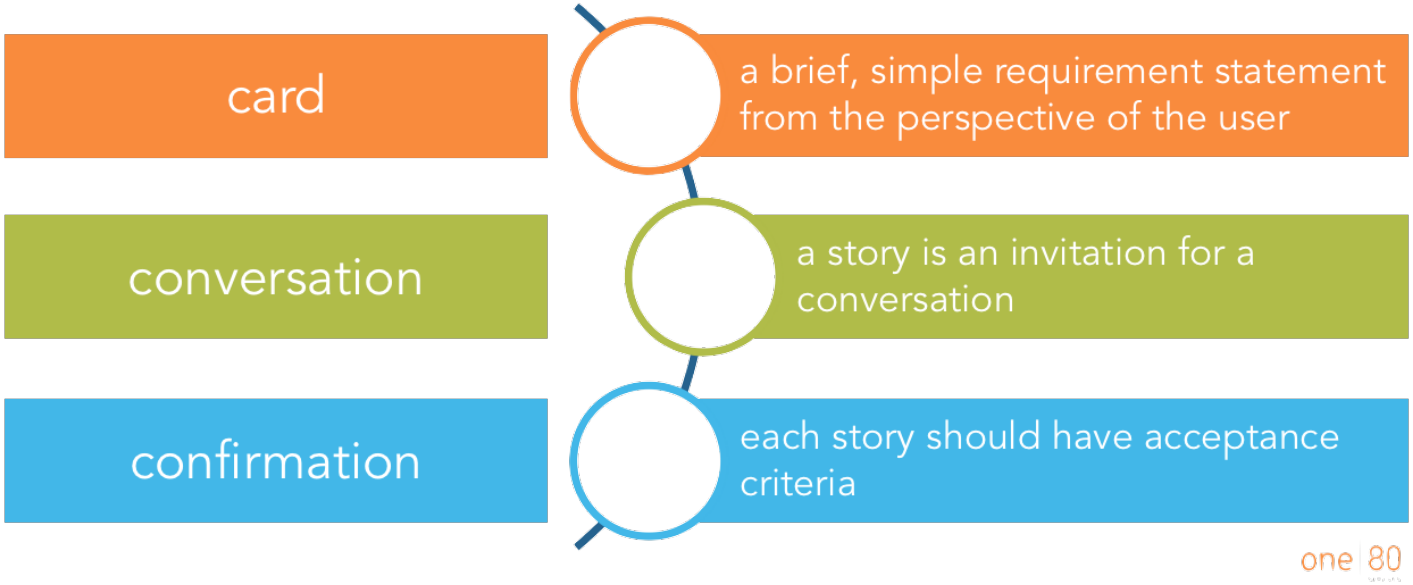




Source: https://www.flickr.com/photos/jakuza/2728096478

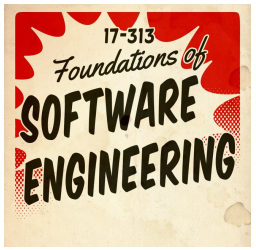
User Stories





Source: http://one80services.com/user-stories/writing-good-user-stories-isnt-its-not-about-writing/

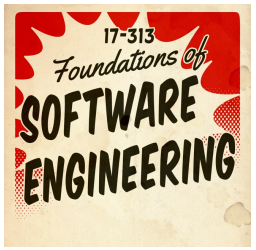
The card: user story template



"As a [role], I want [function], so that [value]"

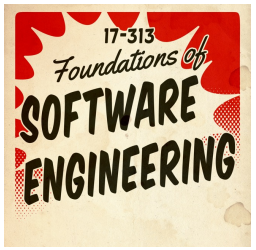
(Should fit on a 3x5 card)

The conversation



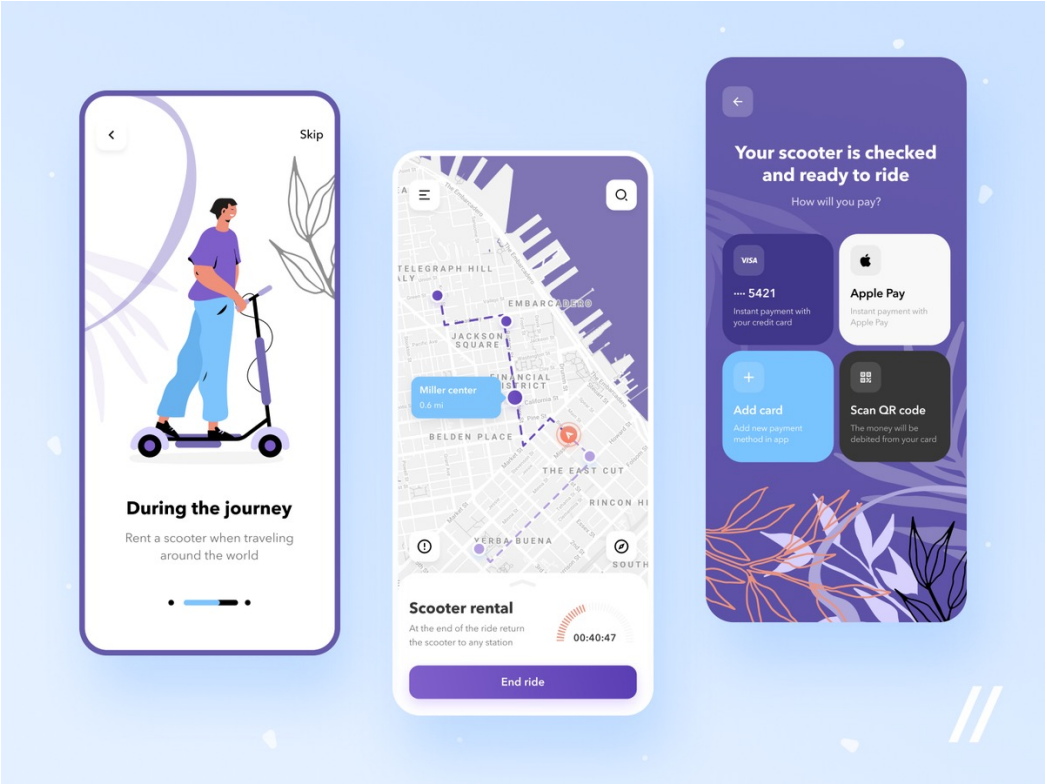
- An open dialog between everyone working on the project and the client
- Split up Epic Stories if needed

The Confirmation



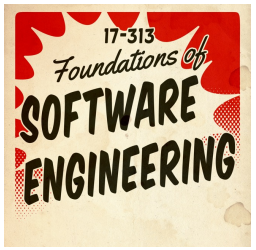
- A confirmation criterion that will show when the task is completed
- Could be automated or manual

Exercise



https://dribbble.com/shots/12512417-Scooter-Rental-App-Design

How to evaluate user story?



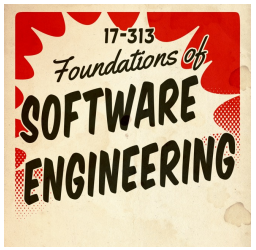
Follow the INVEST guidelines for good user stories!

Source: http://one80services.com/user-stories/writing-good-user-stories-hint-its-not-about-writing/

one 80

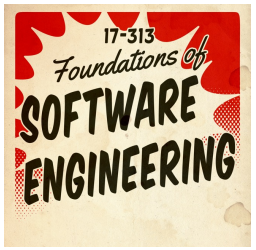
Independent

- Schedule in any order.
- Not overlapping in concept
- Not always possible



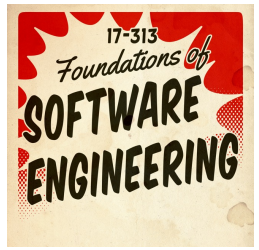
Negotiable

- Details to be negotiated during development
- Good Story captures the essence, not the details

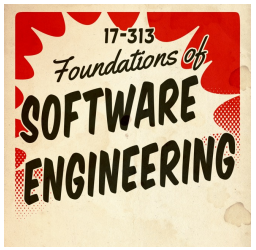


Valuable

- This story needs to have value to someone (hopefully the customer)
- Especially relevant to splitting up issues



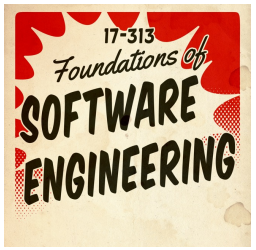
Estimable



- Helps keep the size small
- Ensure we negotiated correctly
- “Plans are nothing, planning is everything” -Dwight D. Eisenhower

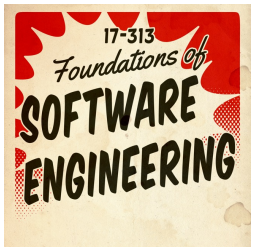
Small

- Fit on 3x5 card
- At most two person-weeks of work
- Too big == unable to estimate

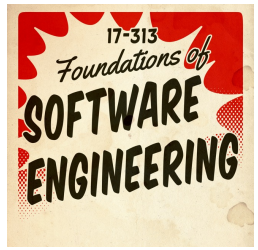


Testable

- Ensures understanding of task
- We know when we can mark task “Done”
- Unable to test == do not understand



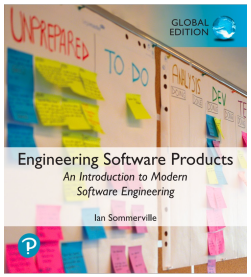
Exercise (Continued)



Follow the INVEST guidelines for good user stories!

one 80

Timeboxed Sprints

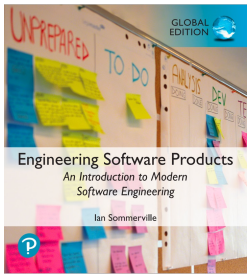


- Products are developed in a series of sprints, each of which delivers an increment of the product or supporting software.

- Sprints are short duration activities (1-4 weeks) and take place between a defined start and end date. Sprints are timeboxed, which means that development stops at the end of a sprint whether or not the work has been completed.

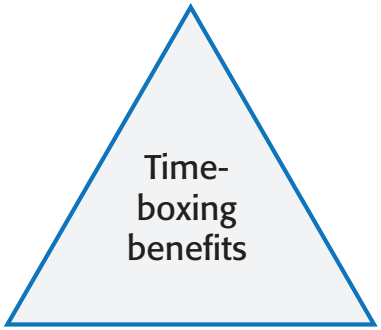
- During a sprint, the team work on the items from the product backlog.

Benefits of Using Timeboxed Sprints



There is a tangible output (usually a software demonstrator) that can be delivered at the end of every sprint.

Demonstrable progress



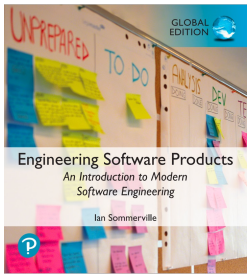
Problem discovery

If errors and omissions are discovered the rework required is limited to the duration of a sprint.

Work planning

The team develops an understanding of how much work they can do in a fixed time period.

Sprint Activities



Sprint planning

Work items to be completed in that sprint are selected and, if necessary, refined to create a sprint backlog. This should not last more than a day at the beginning of the sprint.

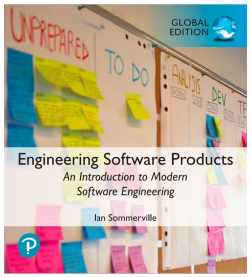
Sprint execution

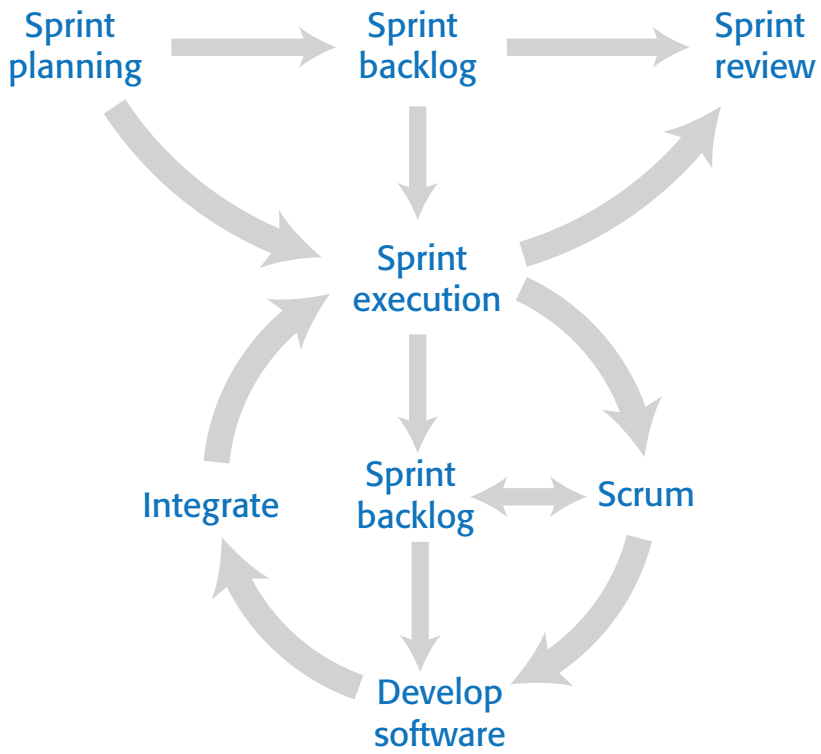
The team work to implement the sprint backlog items that have been chosen for that sprint. If it is impossible to complete all of the sprint backlog items, the sprint is not extended. The unfinished items are returned to the product backlog and queued for a future sprint.

Sprint reviewing

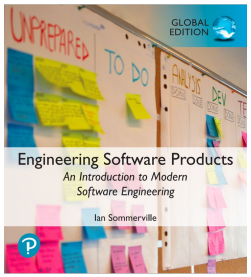
The work done in the sprint is reviewed by the team and (possibly) external stakeholders. The team reflect on what went well and what went wrong during the sprint with a view to improving their work process.

Sprint Activities



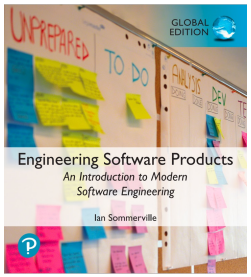


Sprint Planning



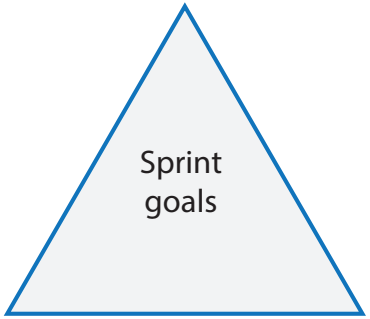
- Establish an agreed sprint goal
  - Sprint goals may be focused on software functionality, support or performance and reliability
- Decide on the list of items from the product backlog that should be implemented
- Create a sprint backlog.
  - This is a more detailed version of the product backlog that records the work to be done during the sprint

Sprint Goals



Implement user roles so that a user can select their role when they login to the system

Functional



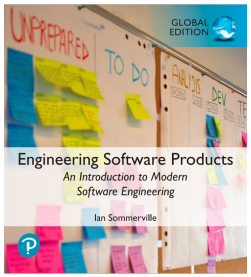
Support

Develop analytics that maintain information about the time users spend using each feature of the system.

Performance and reliability

Ensure that the login response time is less than 10 seconds for all users where there are up to 2000 simultaneous login connections.

Sprint Planning

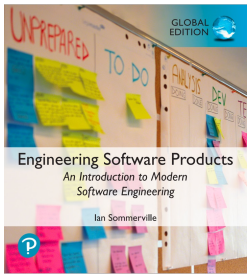


- In a sprint plan, the team decides which items in the product backlog should be implemented during that sprint.
  - Key inputs are the effort estimates associated with PBIs and the team's velocity

- The output of the sprint planning process is a sprint backlog.
  - The sprint backlog is a breakdown of PBIs to show the what is involved in implementing the PBIs chosen for that sprint.

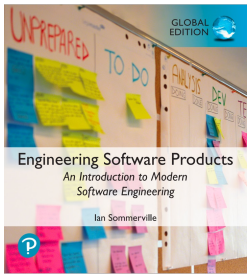
- During a sprint, the team have daily meetings (scrums) to coordinate their work.

Scrums



- A scrum is a short, daily meeting that is usually held at the beginning of the day. During a scrum, all team members share information, describe their progress since the previous day's scrum, problems that have arisen and plans for the coming day. This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.
- Scrum meetings should be short and focused. To dissuade team members from getting involved in long discussions, they are sometimes organized as 'stand-up' meetings where there are no chairs in the meeting room.
- During a scrum, the sprint backlog is reviewed. Completed items are removed from it. New items may be added to the backlog as new information emerges. The team then decide who should work on sprint backlog items that day.

Agile Activities

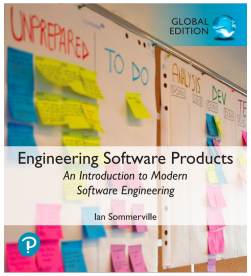


Scrum does not suggest the technical agile activities that should be used. However, I think there are two practices that should always be used in a sprint.

Test automation
As far as possible, product testing should be automated. You should develop a suite of executable tests that can be run at any time.

Continuous integration
Whenever anyone makes changes to the software components they are developing, these components should be immediately integrated with other components to create a system. This system should then be tested to check for unanticipated component interaction problems.

Code Completeness Checklist



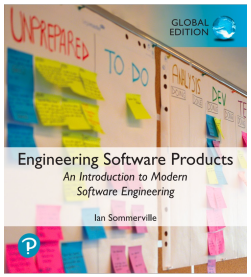
Reviewed

The code has been reviewed by another team member who has checked that it meets agreed coding standards, is understandable, includes appropriate comments, and has been refactored if necessary.

Unit tested

All unit tests have been run automatically and all tests have executed successfully.

Code Completeness Checklist



Integrated

The code has been integrated with the project codebase and no integration errors have been reported.

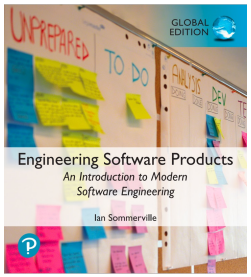
Integration tested

All integration tests have been run automatically and all tests have executed successfully.

Accepted

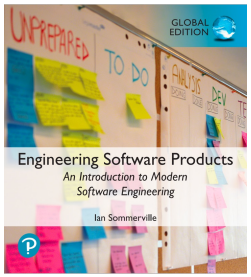
Acceptance tests have been run if appropriate and the product owner or the development team have confirmed that the product backlog item has been completed.

Sprint Reviews



- At the end of each sprint, there is a review meeting, which involves the whole team. This meeting:
  - reviews whether or not the sprint has met its goal.
  - sets out any new problems and issues that have emerged during the sprint.
  - is a way for a team to reflect on how they can improve the way they work.
- The product owner has the ultimate authority to decide whether or not the goal of the sprint has been achieved. They should confirm that the implementation of the selected product backlog items is complete.
- The sprint review should include a process review, in which the team reflects on its own way of working and how Scrum has been used.
  - The aim is to identify ways to improve and to discuss how to use Scrum more productively.

Self-Organising Teams



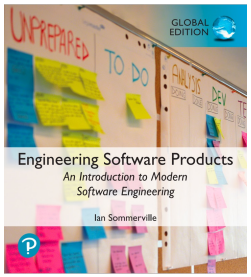
Self-organizing team

coordinates the work of the team members by discussing tasks and reaching a consensus on who should do what.

limits the involvement of engineers in external interactions with management and customers.

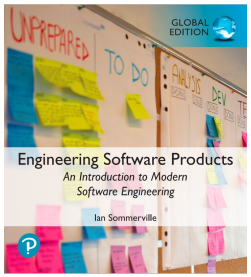
makes its own decisions on schedule and deliverables.

Team Size and Composition



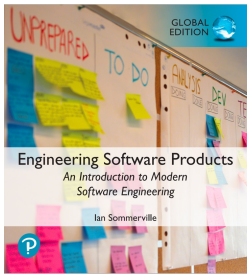
- The ideal Scrum team size is between 5 and 8 people.
  - Teams have to tackle diverse tasks and so usually require people with different skills, such as networking, user experience, database design and so on.
  - They usually involve people with different levels of experience.
  - A team of 5-8 people is large enough to be diverse yet small enough to communicate informally and effectively and to agree on the priorities of the team.
- The advantage of a self-organizing team is that it can be a cohesive team that can adapt to change.
  - Because the team rather than individuals take responsibility for the work, they can cope with people leaving and joining the team.
  - Good team communication means that team members inevitably learn something about each other's areas

Team Coordination



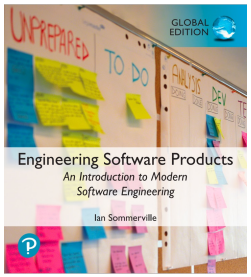
- The developers of Scrum assumed that teams would be co-located. They would work in the same room and could communicate informally.
  - Daily scrums mean that the team members know what's been done and what others are doing.
- However, the use of daily scrums as a coordination mechanism is based on two assumptions that are not always correct:
  - Scrum assumes that the team will be made up of full-time workers who share a workspace. In reality, team members may be part-time and may work in different places. For a student project team, the team members may take different classes at different times.
  - Scrum assumes that all team members can attend a morning meeting to coordinate the work for the day. However, some team members may work flexible hours (e.g. because of childcare responsibilities) or may work on several projects at the same time.

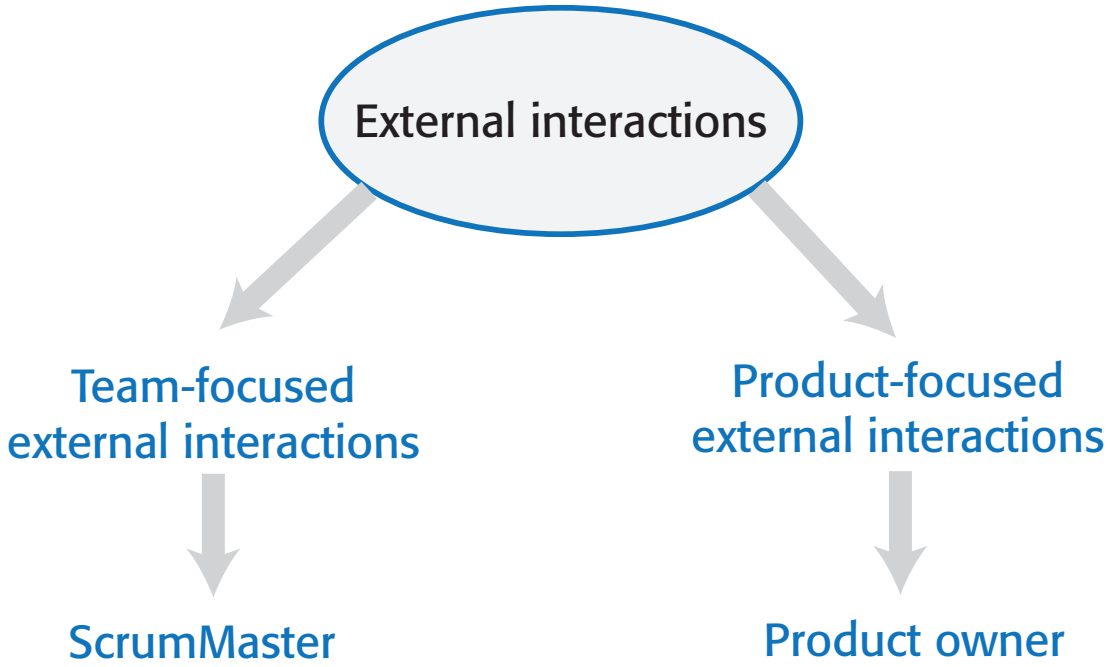
External Interactions



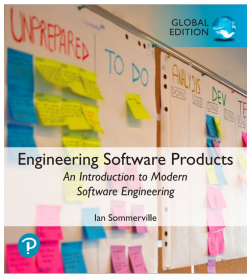
- External interactions are interactions that team members have with people outside of the team.
- In Scrum, the idea is that developers should focus on development and only the ScrumMaster and Product Owner should be involved in external interactions.
- The intention is that the team should be able to work on software development without external interference or distractions.

Managing External Interactions



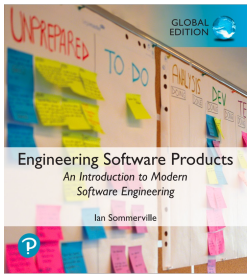


Project Management

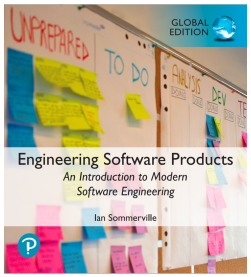


- In all but the smallest product development companies, there is a need for development teams to report on progress to company management.
- A self-organizing team has to appoint someone to take on these responsibilities.
  - Because of the need to maintain continuity of communication with people outside of the group, rotating these activities around team members is not a viable approach.
- The developers of Scrum did not envisage that the ScrumMaster should also have project management responsibilities.
  - In many companies, however, the ScrumMaster has to take on project management responsibilities.
  - They know the work going on and are in the best position to provide accurate information and project plans and progress.

Project Management Responsibilities



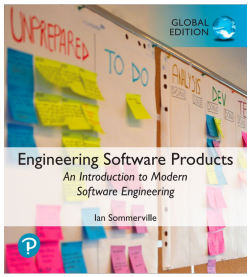
Software Products



- There are three factors that drive the design of software products
  - Business and consumer needs that are not met by current products
  - Dissatisfaction with existing business or consumer software products
  - Changes in technology that make completely new types of product possible

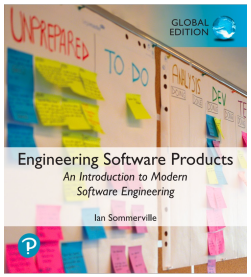
- In the early stage of product development, you are trying to understand, what product features would be useful to users, and what they like and dislike about the products that they use.

Software Features



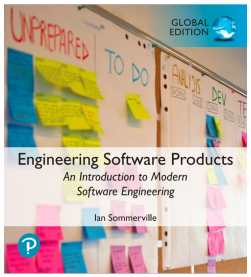
- A feature is a fragment of functionality such as a ‘print’ feature, a ‘change background feature’, a ‘new document’ feature and so on.
- Before you start programming a product, you should aim to create a list of features to be included in your product.
- The feature list should be your starting point for product design and development.

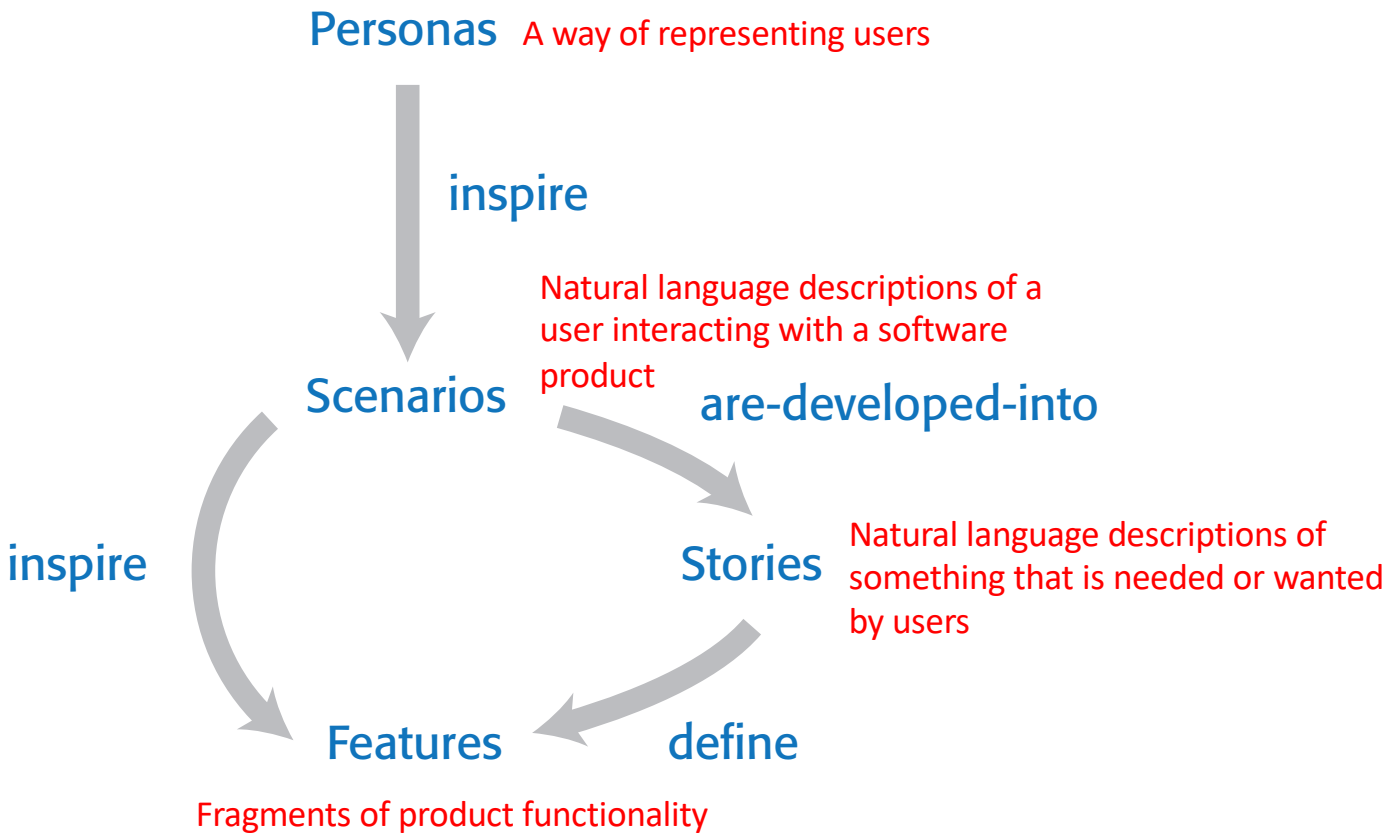
User Understanding



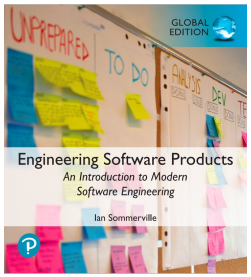
- It makes sense in any product development to spend time trying to understand the potential users and customers of your product.
- A range of techniques have been developed for understanding the ways that people work and use software.
  - These include user interviews, surveys, ethnography and task analysis.
  - Some of these techniques are expensive and unrealistic for small companies.
- Informal user analysis and discussions, which simply involve asking users about their work, the software that they use, and its strengths and weaknesses are inexpensive and very valuable.

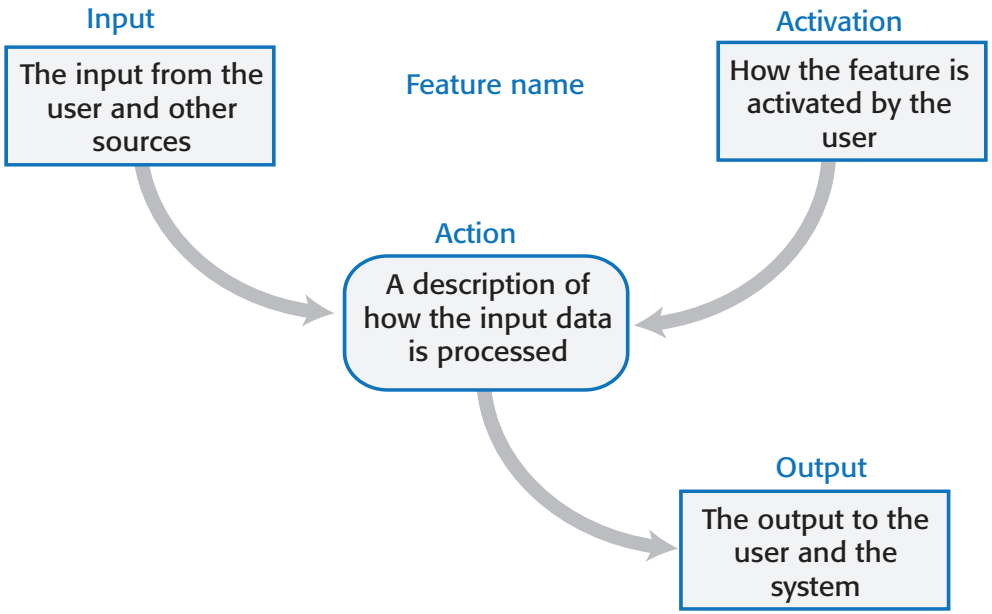
From Personas to Features



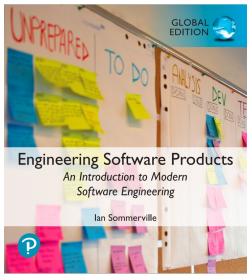


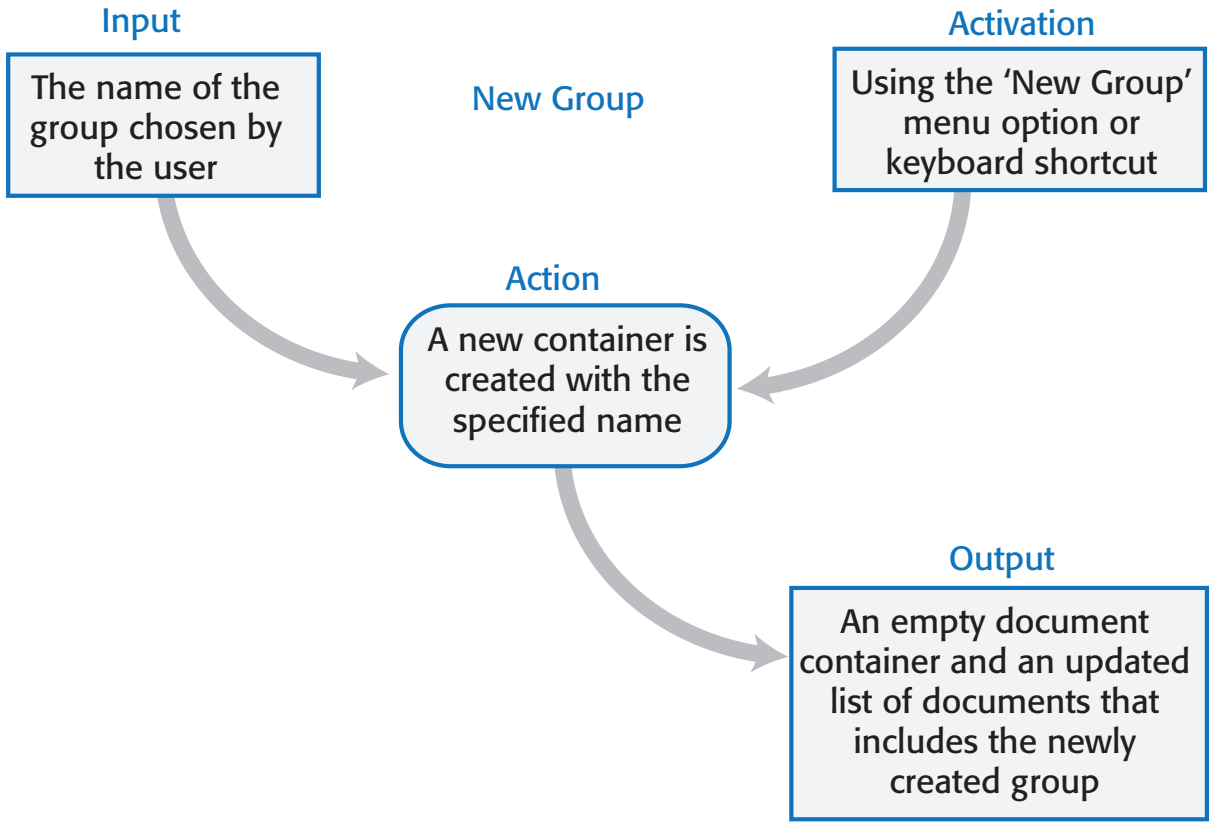
Feature Description



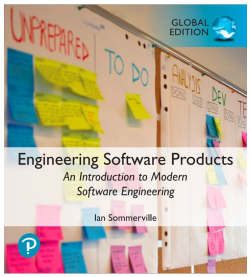


The “New Group” Feature Description



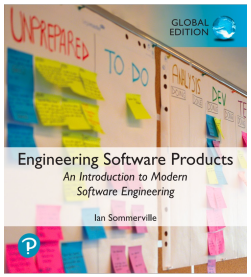


Key Points



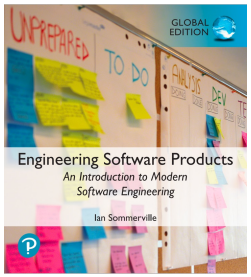
- Software products are software systems that include general functionality that is likely to be useful to a wide range of customers.
- In product software engineering, the same company is responsible for deciding on the features that should be part of the product and the implementation of these features.
- Software products may be delivered as stand-alone systems running on the customer's computers, hybrid systems or service-based systems. In hybrid systems, some features are implemented locally and others are accessed over the Internet. All product features are remotely accessed in service-based products.

Key Points



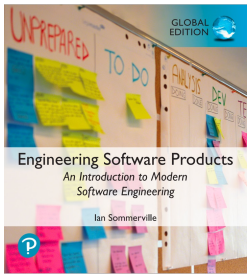
- A product vision should succinctly describe what is to be developed, who are the target customers for the product and why they should buy the product that you are developing.
- Domain experience, product experience, customer experience and an experimental software prototype may all contribute to the development of the product vision.
- Key responsibilities of product managers are product vision ownership, product roadmap development, creating user stories and the product backlog, customer and acceptance testing and user interface design.

Key Points



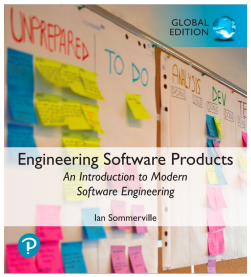
- Product managers work at the interface between the business, the software development team and the product customers. They facilitate communications between these groups.
- You should always develop a product prototype to refine your own ideas and to demonstrate the planned product features to potential customers

Key Points



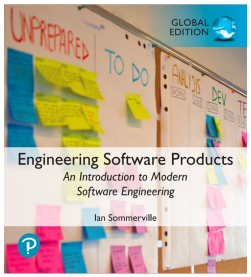
- The best way to develop software products is to use agile software engineering methods that are geared to rapid product development and delivery.
- Agile methods are based around iterative development and the minimization of overheads during the development process.
- Extreme programming (XP) is an influential agile method that introduced agile development practices such as user stories, test-first development and continuous integration. These are now mainstream software development activities.

Key Points



- Scrum is an agile method that focuses on agile planning and management. Unlike XP, it does not define the engineering practices to be used. The development team may use any technical practices that they believe are appropriate for the product being developed.
- In Scrum, work to be done is maintained in a product backlog – a list of work items to be completed. Each increment of the software implements some of the work items from the product backlog.

Key Points



- Sprints are fixed-time activities (usually 2–4 weeks) where a product increment is developed. Increments should be 'potentially shippable' i.e. they should not need further work before they are delivered.
- A self-organizing team is a development team that organizes the work to be done by discussion and agreement amongst team members.
- Scrum practices such as the product backlog, sprints and self-organizing teams can be used in any agile development process, even if other aspects of Scrum are not used.