Software Engineering
Lecture 2

Instructor : Conf. dr. Cristina Mîndruţă
cristina.mindruta@e-uvt.ro
Site:
http://sites.google.com/site/ingswcm

Adopted books:

Software Engineering (9th Edition) by Ian Sommerville

http://staff.fmi.uvt.ro/~cristina.mindruta

Engineering software products by Ian Sommerville

The Essentials of Modern Software Engineering by Ivar Jacobson et.all

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Perspectives on software engineering
The student

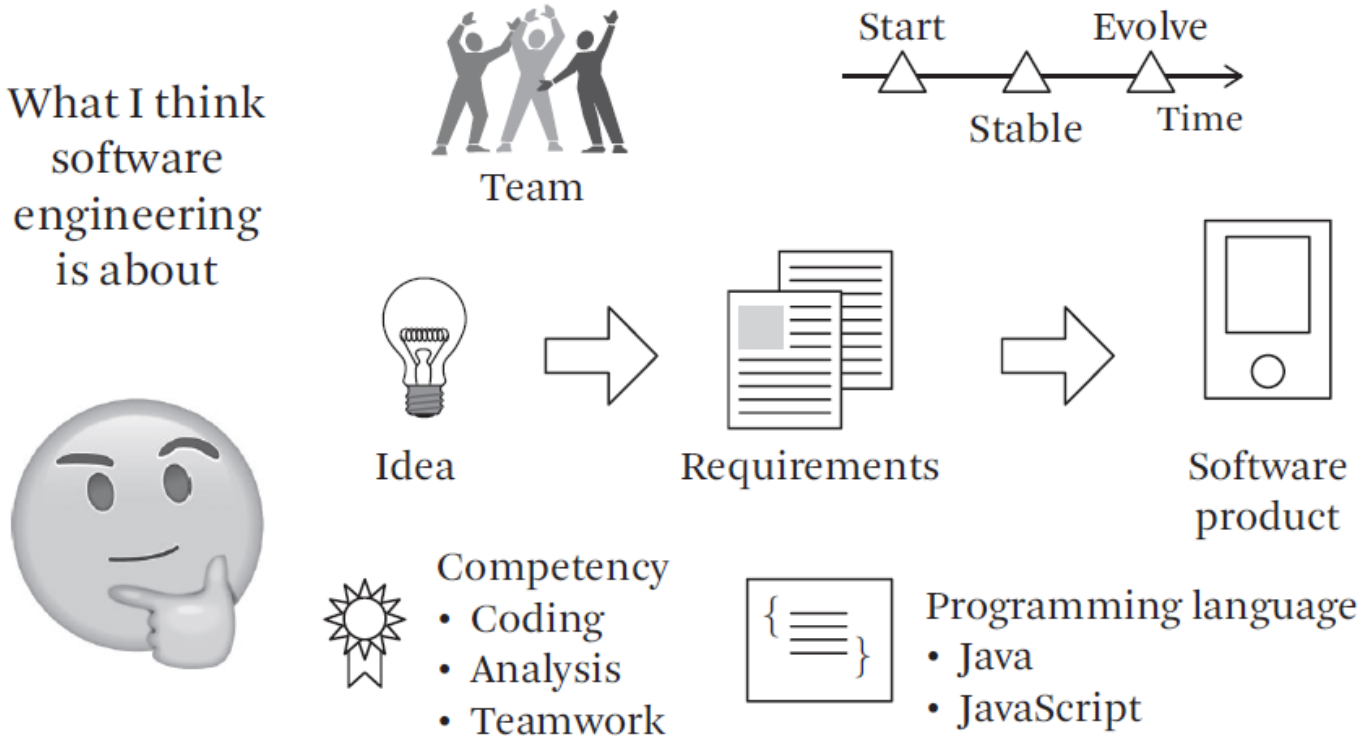


Fig. 1.2 What software engineering is from the eyes of a student.
The Essentials of Modern Software Engineering

Perspectives on software engineering
The student after internship

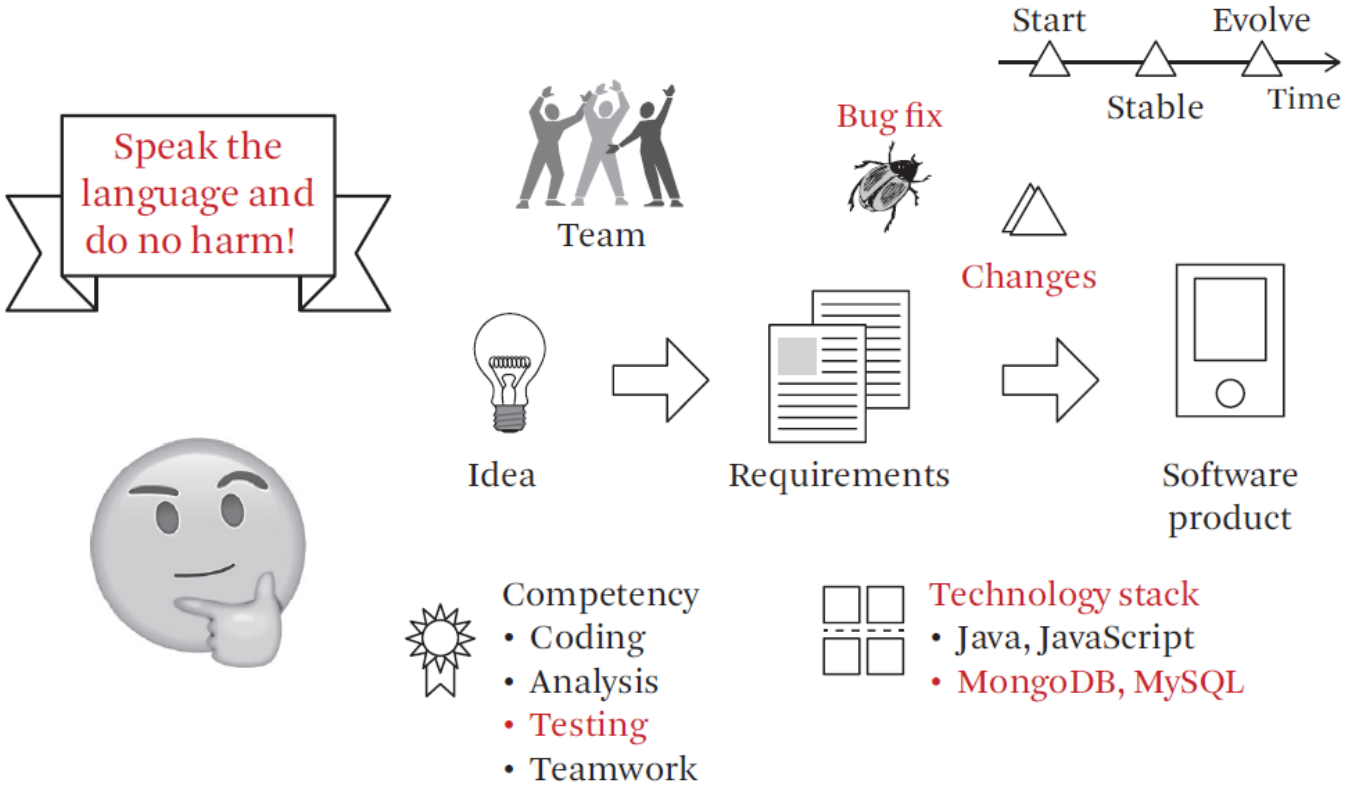


Fig. 1.4 What software engineering is from the eyes of a student after internship..
The Essentials of Modern Software Engineering

Perspectives on software engineering
Young professional

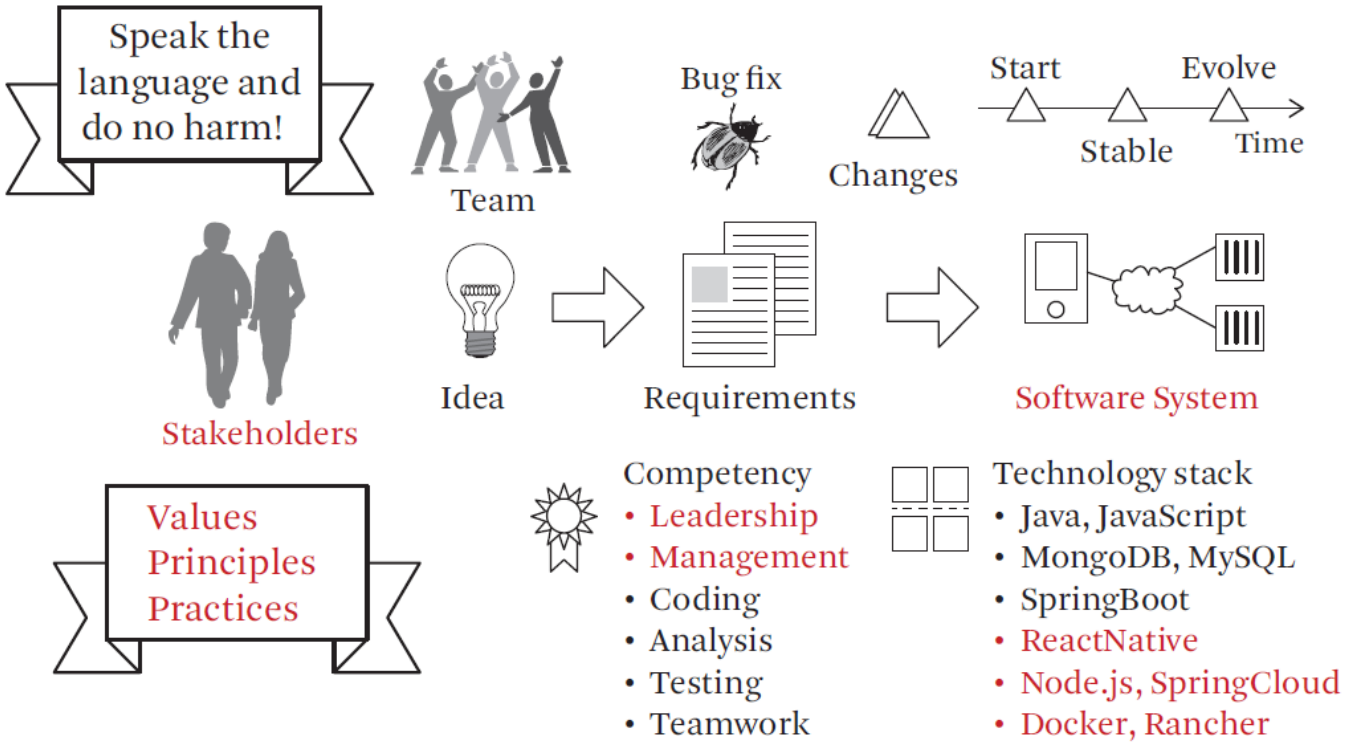


Fig. 1.5 What software engineering is from the eyes of a young professional..
The Essentials of Modern Software Engineering

Programming stands for the work related to implementation or *coding of source code*.

Software development is the larger process which, apart from programming, includes working with *requirements, design, test, etc*.

Software engineering - Moving from development to engineering

means *more reliance on science and less on craft*,

which typically manifests itself in some form of description of a *designated way of working* and *higher-level automation of work*. This allows for *repeatability* and *consistency* from project to project.

Engineering also means that *teams*, for example, *learn as they work* and *continuously improve* their way of work.

Motivations for a discipline of Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled. **?**
- 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. Suduko). **?**
- More and more needs evolve for larger and more complex software.

BUT

- Many software projects have failed.
- Defects are encountered in software systems.

Project failures

Chaos Report 1995 – only 16% completely successful projects.

Survey responses:

Mostly indicated *success factors*:

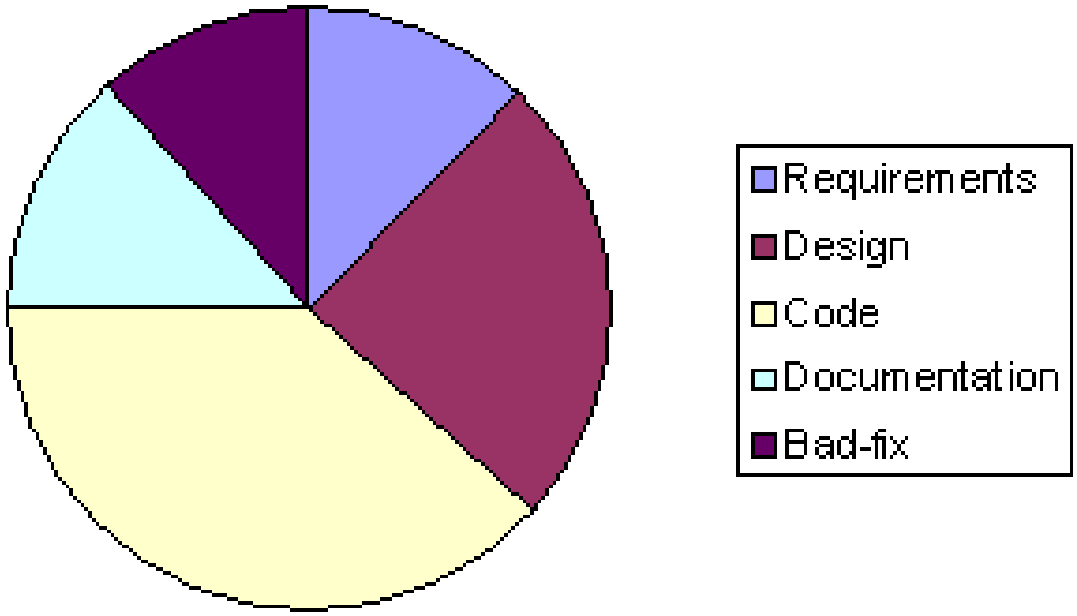
- User involvment
- Executive management support
- Clear requirement statements
- Proper planning

Mostly indicated *failure factors*:

- Lack of user input
- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of resources

Product failures

The average procent of bugs by different origins:



BUT – they have different costs, the most costly being *requirements errors*.

Product failures

Other failure sources are *bad management*, *communications* or *training*.

Basic *management strategies*:

- Focussed attention on the software development environment
- Disciplined development process
- Methodical usage of metrics to gauge cost, schedule, and performance targets.

Engineering

Definitions :

(ECPD - Engineers' Council for Professional Development) – The *creative application* of *scientific principles* to *design or develop* structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to *construct* or *operate the same with full cognizance of their design;* or to *forecast their behaviour* under specific operating conditions; all as respects an intended function, economics of operation and *safety to life and property*.

(Wikipedia) – The discipline, art, skill, profession and technology of acquiring and applying *scientific, mathematical, economic, social and practical knowledge*, in order to *design and build* structures, machines, devices, systems, materials and processes.

Software Engineering

Def. Software engineering = (1) the application of systematic, disciplined, quantifiable approach to the development, operation, maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1) . (IEEE)

Def. Software engineering = engineering discipline that is concerned with all aspects of software production, from the early stages of system specification to maintaining the system after it has gone into use. (Ian Sommerville, 2007)

Software Engineering

Def. Software engineering = the discipline of *developing and maintaining* software systems that behave reliable and efficiently, are affordable to develop and maintain, and satisfy all requirements that customers have defined for them. (Computing Curricula 2005, ACM, AIS, IEEE-CS)

Integrates *principles of mathematics* and *computer science* with the *engineering practices* developed for tangible, physical artifacts.

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

I. Jacobson, H. Lawson, Pan-Wei Ng, P.E. McMahon, M. Goedicke
The Essentials of Modern Software Engineering, 2019, ACM Books

What is fascinating about this aspect of software development is that it is *more than just programming*. Rather, it is to *learn the whole picture* and as a software engineer *to solve a problem* or *exploit an opportunity* that the *users* may have.

As a new student, understanding what software engineering is about is not easy, because there is no way we can bring its realities and complexities into the student's world.

Nevertheless, it is a student's responsibility to embark on this journey of learning and discovery into the world of software engineering.

Software Engineer

Software engineer – engineer who applies the principles of software engineering to the design, development, testing, and evaluation of the software systems.

- Software engineers should :
  - adopt a *systematic* and *organized* approach to their work
  - use *appropriate* tools and techniques depending on:
    - the problem to be solved,
    - the development constraints,
    - the resources available.

Software Engineer

Responsibilities:

- design, develop, implement and test the applications along with evaluation of the product.
- design and create the software based on the requirements of the client.
- develop easy software solutions for complex tasks.
- design and generate the software based on the interaction with programmers and clients.

Software Engineer

Software Engineer Job Responsibilities:

- Will have to understand and learn new technologies.
- Explain the client requirements to the programmers.
- Understand the tactics and line of design.
- Develop good communication and interaction with the team members for better outcome.
- Explain the restraints of technology to the business managers.
- Apply real –time computation in to practice.
- Manage and support multiple projects.
- Adapt to different technical environments.
- Upgrade with the latest technology.
- Development and testing of the applications.
- Ensure the performance of the software product.
- Conduct quality analysis.
- Will have to easily adapt to updated hardware and software fields.
- Evaluation of the developed software to ensure the quality and to check whether it meets the required standards.
- Review various business plans.
- Is responsible for full-lifecycle application growth.
- Design, code and debug applications in several software languages.

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Software Process

Def. **Software process** – structured set of ***activities*** whose goal is the development and/or evolution of software.

Generic activities in all software processes are:

- ***Specification***
  - defining what the system should do and its development constraints
- ***Development***
  - production of the software system
- ***Validation***
  - checking that the software is what the customer wants
- ***Evolution***
  - changing the software in response to changing demands.

Software Process Model

Def. A **software process model** is a simplified representation of a software process, presented from a specific perspective.

- Examples of process perspectives are
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models
  - Waterfall;
  - Iterative development;
  - Component-based software engineering;

Software Process Model

Process:

1990s - Improved (extended, complicated) by:

- More reviews, inspections, testing, meetings
- Expensive quality assurance and measurement efforts

to prevent, detect and correct problems, improve the quality of software and increase the productivity of software developers.

2000s – simplified to face new market challenges of speed and cost.

No single process fits all cases!!!

Software Engineering Methods

Def. Software engineering ***methods*** are ***structured*** approaches to software development which include system *models*, *notations*, *rules*, design *advice* and process *guidance*.

- Model descriptions

Descriptions of graphical models which should be produced;

- Rules

Constraints applied to system models;

- Recommendations

Advice on good design practice;

- Process guidance

What activities to follow.

CASE (Computer-Aided Software Engineering)

Def. CASE are software systems that are intended to provide automated support for software process activities.

CASE systems are often used for method support.

Upper-CASE

Tools to support the early process activities of requirements and design;

Lower-CASE

Tools to support later activities such as programming, debugging and testing.

Examples ?

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Professional and Ethical Responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law.

Issues of Professional Responsibility

- ***Confidentiality***
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

- ***Competence***
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

Issues of Professional Responsibility

- ***Intellectual property rights***
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

- ***Computer misuse***
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the *behaviour* of and *decisions* made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

Preamble

- Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession.

- In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Code of Ethics - Principles

PUBLIC

- Software engineers shall act consistently with the public interest.

CLIENT AND EMPLOYER

- Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

PRODUCT

- Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

JUDGMENT

- Software engineers shall maintain integrity and independence in their professional judgment.

Code of Ethics - Principles

MANAGEMENT

- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

PROFESSION

- Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

COLLEAGUES

- Software engineers shall be fair to and supportive of their colleagues.

SELF

- Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Ethical Dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Software system

Def. ***Software system*** = computer *programs* and associated *documentation* (such as requirements, design models and user manuals).

New software can be created by:

- developing new programs,
- configuring generic software systems
- reusing existing software.

Software system

Def. Software system = computer *programs* and associated *documentation* (such as requirements, design models and user manuals).

Specifics of software :

- Intangible
- Easy to reproduce – cost is in its development, not in manufacturing
- Easy to modify
- Does not “wear out”

Attributes of Good Software

The software should deliver the required *functionality* and *performance* to the user and should be *maintainable*, *dependable* and *acceptable*.

Maintainability

- Software must evolve to meet changing needs;

Dependability

- Software must be trustworthy;

Efficiency

- Software should not make wasteful use of system resources;

Acceptability

- Software must be accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

Custom software systems and software products

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.*

Software products are *generic software systems* that provide functionality that is useful to *a range of customers*.

Custom software systems

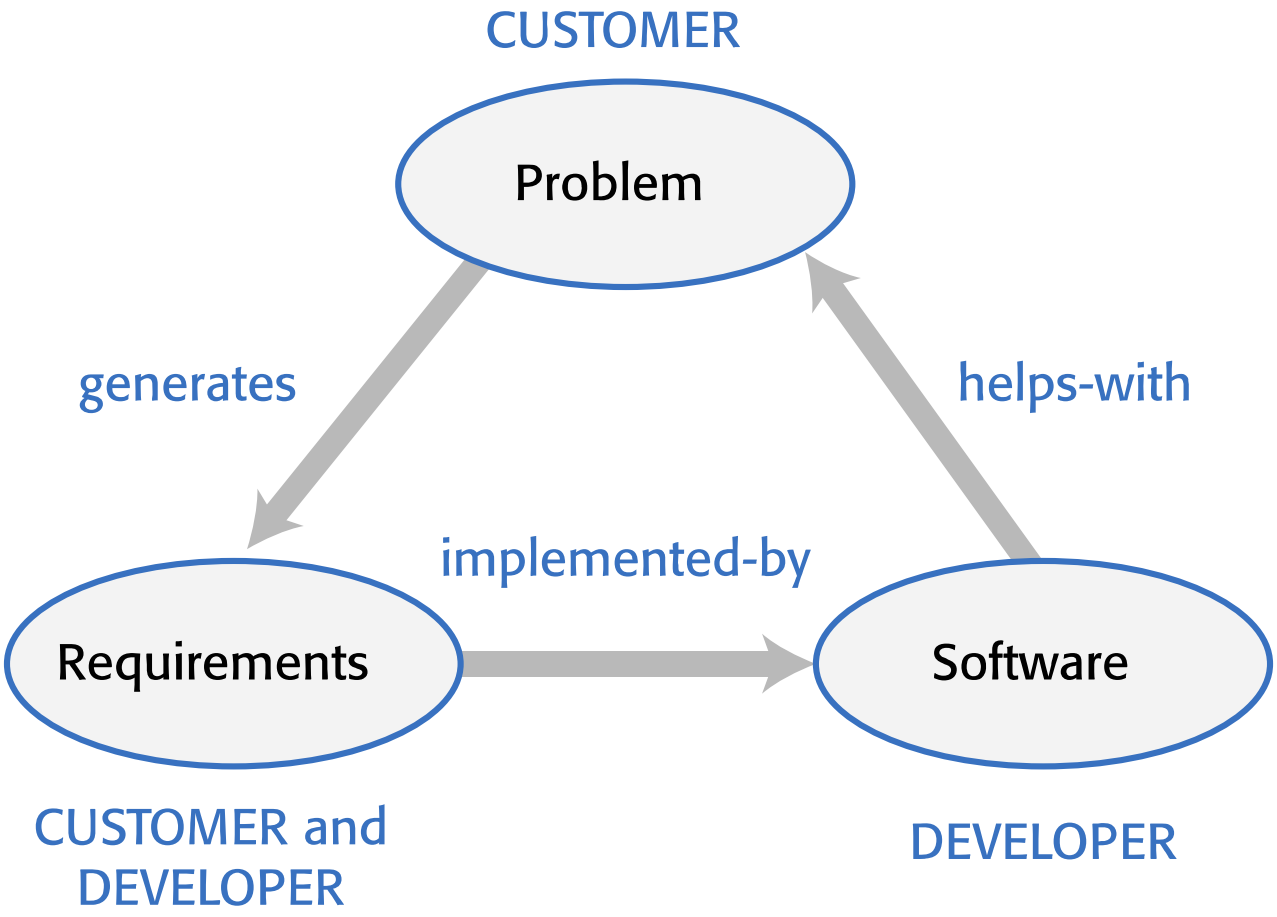
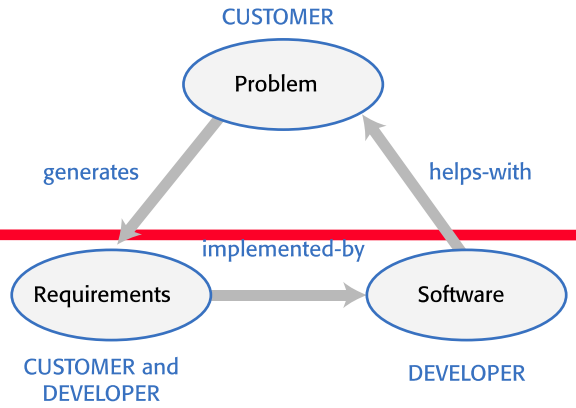


Figure 1.1 Project-based software engineering
Ian Sommerville – Engineering Software Products

Project-based software engineering



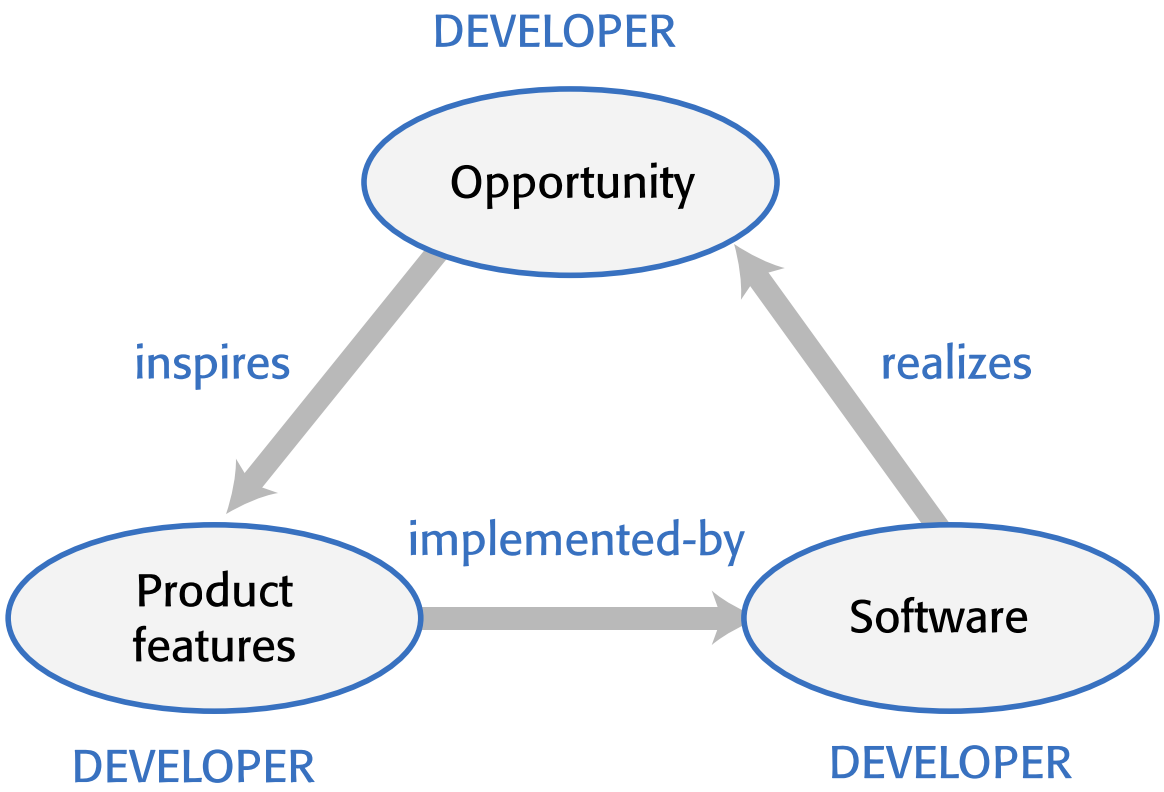
The starting point for the software development is *a set of 'software requirements'* that are owned by an *external client* and which set out what they want a software system to do to support their business processes.

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.

Custom software systems



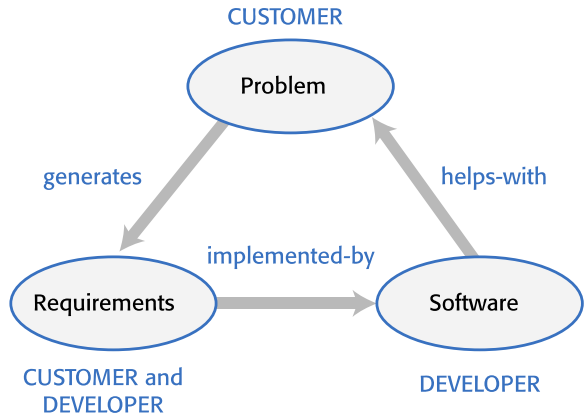
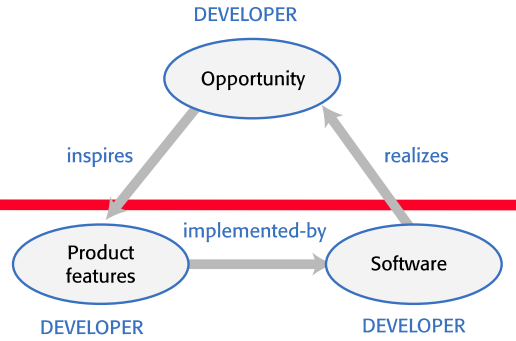


Figure 1.2 Product software engineering
Ian Sommerville – Engineering Software Products

Software product 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 com*pany 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 and 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.

Example : A company that offers communication software for emergency services. The core contains typic communication services (calls receive and logging, emergency response, transfer of information to the vehicles, etc.). The differences may appear to the radio equipment types, messages structure, etc.

Platform

A software (or software+hardware) product that includes *functionality* so that *new applications can be built on it.*

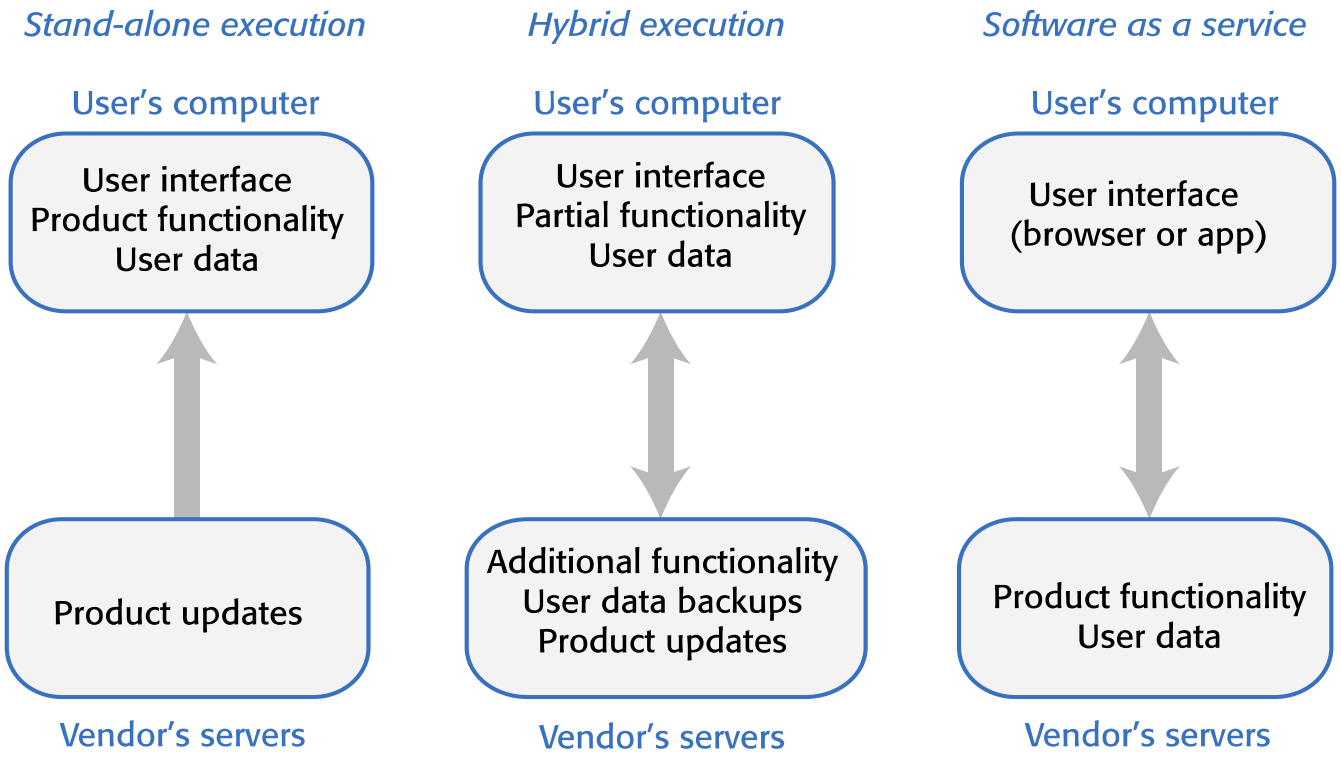
Example : Facebook. It provides an extensive set of product functionality but also provides support for creating 'Facebook apps'. These add new features that may be used by a business or a Facebook interest group.

Software execution models

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

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.

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.



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.

Formative evaluation

1. What contains a software system besides its executable code ?

2. Based on these figures, explain the differences between project-based software engineering and product software engineering.

https://forms.gle/7xrtjKF1pQHZMNYu8

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

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)

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.”

Information sources for developing a 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.

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.

Example : a vision statement for an iLearn system

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.

Specification of the benefits of the entity which will pay for the software product and needed licenses.

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.

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Software product management

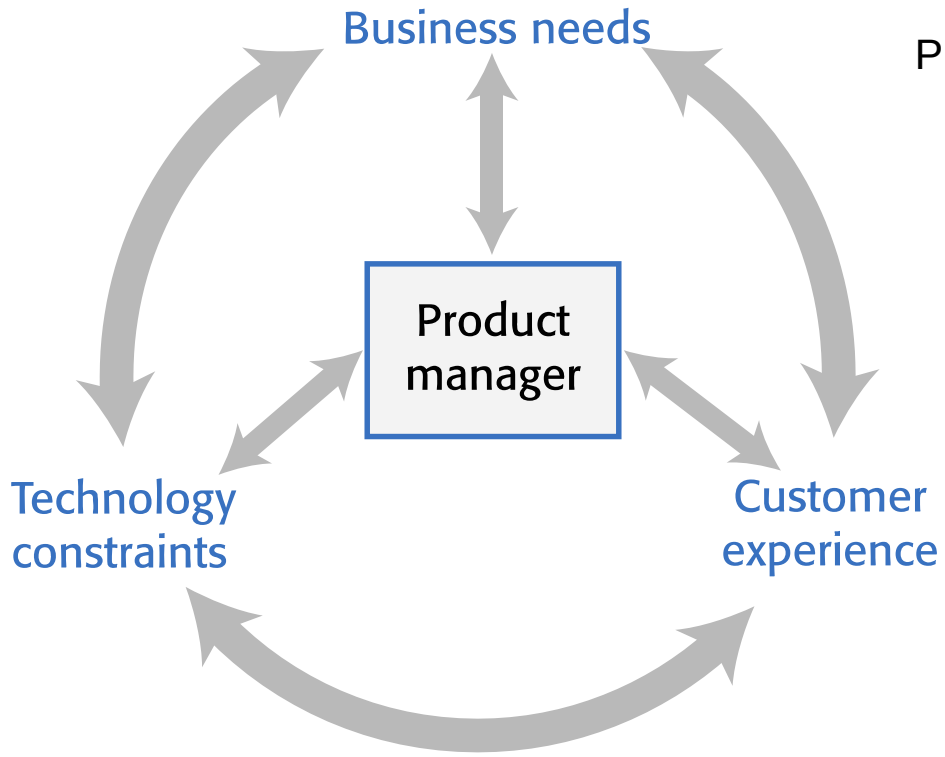
Software product management is a business activity that focuses on the software products developed and sold by the business.

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

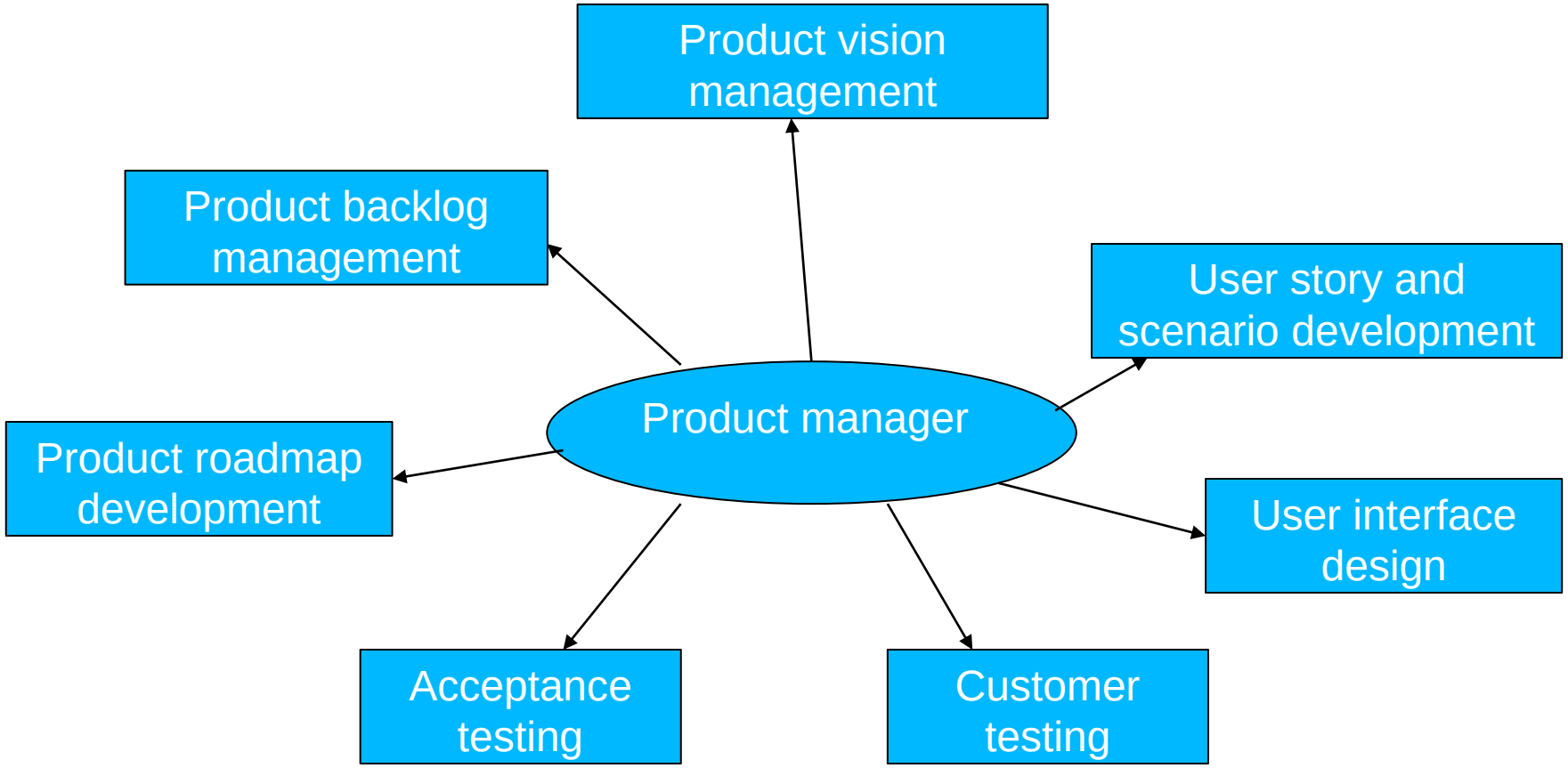


PMs have to ensure that the software being developed meets the business goals of the software development company.

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.

PMs must make developers aware of technology issues that are important to customers.

Technical interactions of product managers



Technical interactions of product managers

Product vision management

The product manager may be resposible for helping with the development of the product vision. The PM should allways be responsible for for *managing the vision,* which involves assesing and evaluating proposed changes against the product vision. The PM 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.

User story and scenario development

User stories and scenarios are used to *refine a product vision* and *identify product features*. Based on his/her knowledge of customers, the PM should lead the development of stories and scenarios.

Technical interactions of product managers

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 the software release *meets the goals* set out in the product roadmap and 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..

Customer testing

Customer testing involve taking a release of a product to customers and getting *feedback* on the product's *features, usability and its utility* for the business of the client. PMs are involved in *selecting customers* to be involved in the customer testing process and *working with them* during the process.

Technical interactions of product managers

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*.

Topics covered

Software engineering discipline

The software engineer

Software engineering process and tools

Software engineering code of ethics

Customer software and software products

The product vision

Software product management

Product prototyping

Product prototyping

Product prototyping is the process of developing an *early version* of a product to *test the ideas* and to *convince the team and the company funders* that the 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 to do when developing a software product. The aim is to have a *working version* of the software that can be used to *demonstrate its key features*.

This kind of prototype is meant to be *thrown-away* after development and the software will be *re-implemented* taking account of issues such as security and reliability, errors handling, other development details.

Two-stage prototyping

Feasibility demonstration

An *executable system* that *demonstrates the new ideas* in the software product is created.

The aims at this stage are to see if these *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

An *existing prototype* created to demonstrate feasibility is *extended* with ideas for *specific customer featur*es and how these can be realized.

Before doing this extension, some *user studies* need to be done in order to have a clearer idea of *potential users and scenarios of use*.

Formative evaluation

1. What is a prototype and why is important to develop product prototypes ?

2. Imagine you have your own software development company. Write a vision for a new software product that you intend to develop.

https://forms.gle/rprBmKLTXLn5fBRY8

Key points

Software engineering is an engineering discipline that is concerned with all aspects of software production.

Software engineer applies the principles of software engineering to the design, development, testing, and evaluation of the software systems.

Software process is a structured set of activities whose goal is the development and/or evolution of the software system.

CASE are software systems that are intended to provide automated support for software process activities.

Software engineering profession is based on a code of ethics (ACM/IEEE Code of Ethics).

In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to Eight Principles : public, client and employer, product, judgement, management, profession, colegues, self.

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.

A product vision should succinctly described what is to be developed, who are the target customers for the product and why they should buy the product to be developed.

Domain experience, product experience, customer experience and experimental software prototype may all contribute to the development of the product vision.

Key points

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.

Product managers work at the interface between the business, the software development team and the product customers. They facilitate communications between these groups.

A product prototype should always be developed in order to refine own ideas and to demonstrate the planned product features to company funders and potential customers, highlighting original product features that are better than those in competing products. The prototype may be also used to identify fundamental software components or services and to test technologies.