Osnovi softverskog inženjerstva

P-01: Uvod

2024

Topics covered

- **Professional software development**
  - What is meant by software engineering.
- **Software engineering ethics**
  - A brief introduction to ethical issues that affect software engineering.

Software engineering

- **Software engineering** is an **engineering discipline** that is **concerned with all aspects** of **software production** **from the early stages** of system specification through **to maintaining** the system after it has gone into use.
- **Engineering discipline**
  - **Using appropriate theories** and **methods** **to solve problems** **bearing in mind constraints** (organizational and financial).
- **All aspects of software production**
  - Not just technical process of development.
  - Also project management, and
  - Development of tools, methods etc. to support software production.

Alternative definitions of SE

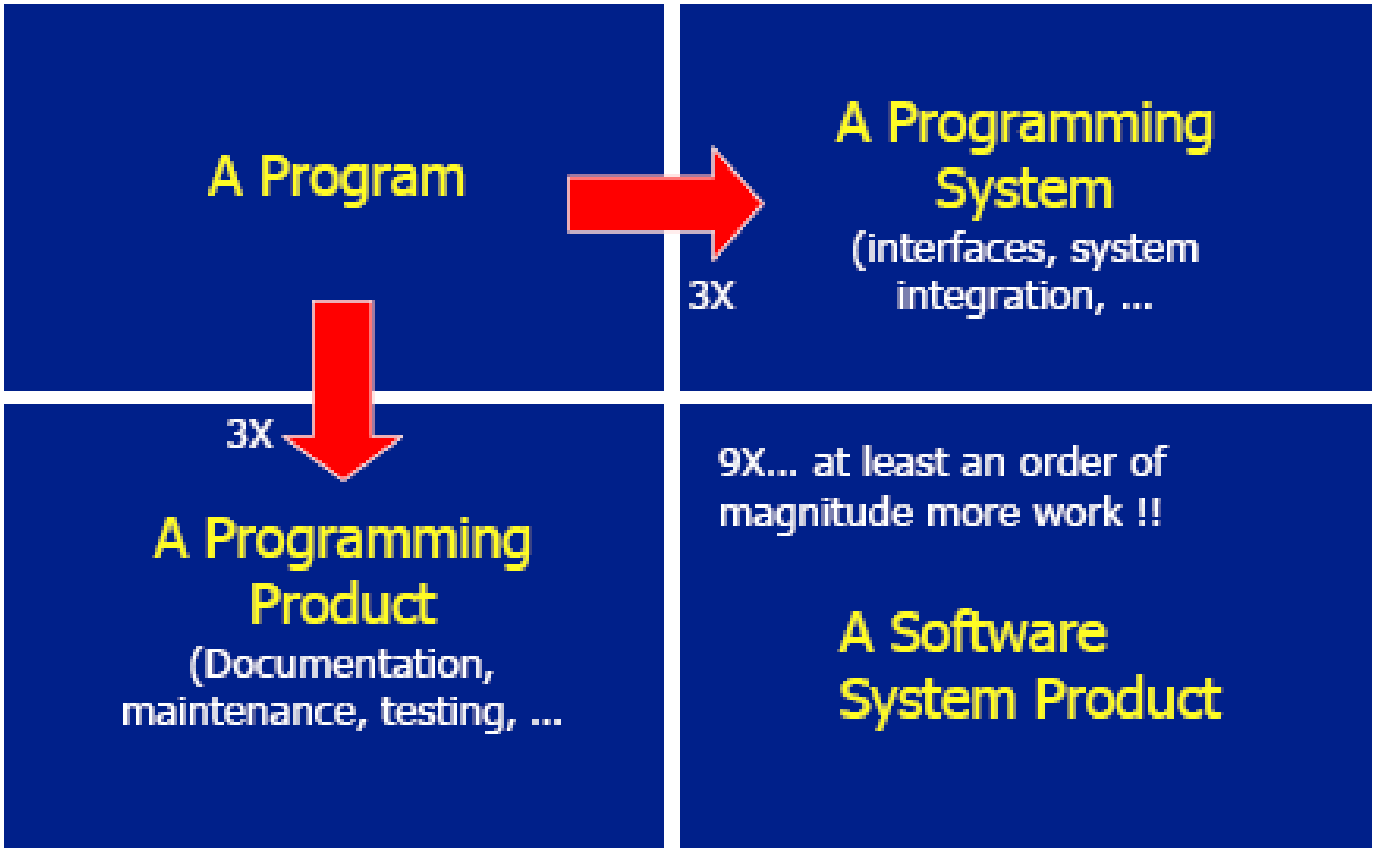
- **Systematic approach for developing software**
- **Methods and techniques to develop and maintain quality software to solve problems** (*Pfleeger*)
- **Study of the principles and methodologies for developing and maintaining software systems** (*Zelkowitz*)
- **Practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them** (*Boehm*)
- **Deals with establishment of sound engineering principles and methods in order to economically obtain software that is reliable and works on real machines** (*Bauer*)

SE ≠ Programming

- Programming without SE is just hacking

Software product ≠ Program

- **Software product** = **Computer programs** + **associated documentation** (requirements specification, design models, user manuals).



SE vs Computer Science

- **Computer science** is **concerned with theory and fundamentals**
- **Software engineering** is **concerned with the practicalities of developing and delivering useful software**
- **Software engineering** is **closer to good business practices than science**
- **Computer science theories** are **still insufficient to act as a complete underpinning for software engineering** (unlike e.g. physics and electrical engineering)

SE vs System Engineering

- **System engineering** is **concerned with all aspects of systems development** including hardware, software and process engineering.
- **Software engineering** is **part of system engineering**, concerned with developing the software infrastructure, control, applications and databases in the system.
- **System engineers** are involved in system specification, architectural design, integration and deployment.

Software engineering (SE)

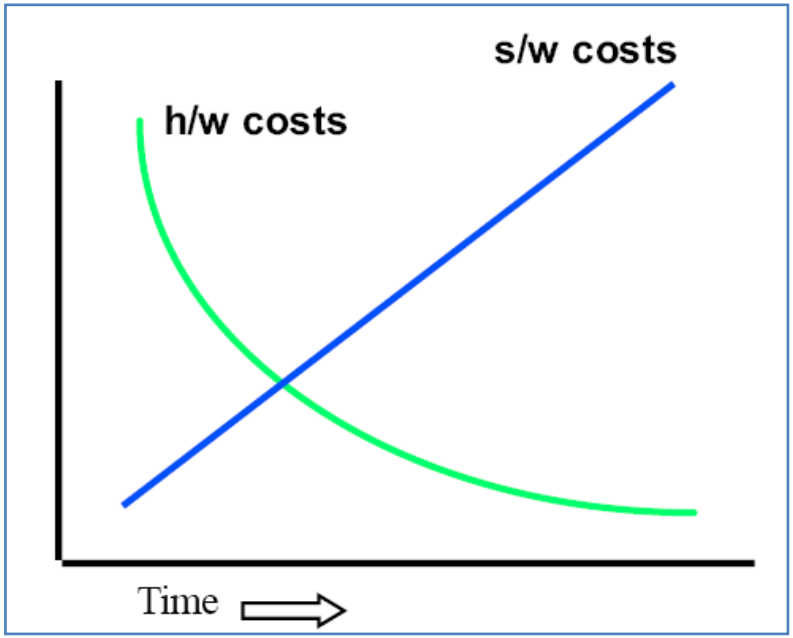
- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

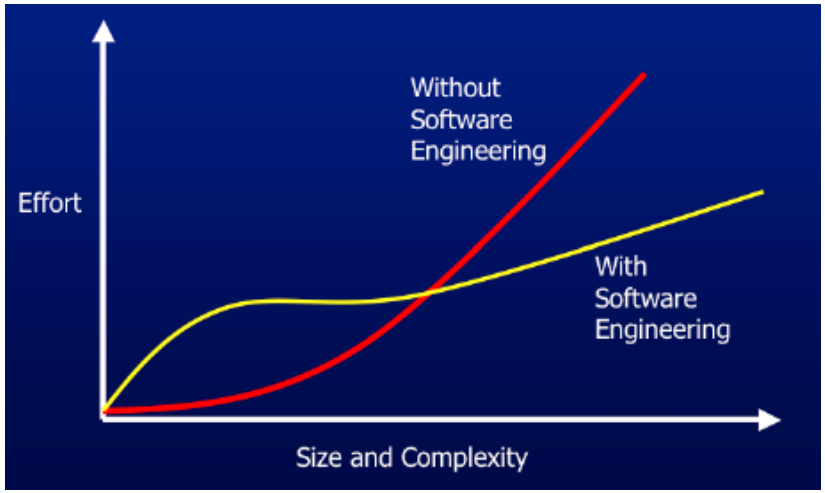
Elements of SE

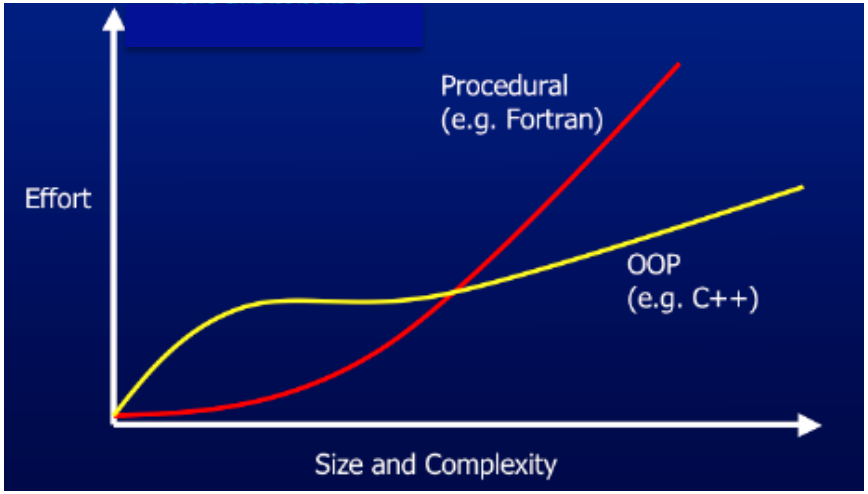
- Professionalism, economics, ethics
- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering processes
- Software engineering tools and methods
- Software quality

Software costs

- **Software costs often dominate computer system costs**.
  The costs of software on a PC are often greater than the hardware cost.
- **Software costs more to maintain than it does to develop**. For systems with a long life, maintenance costs may be several times development costs.
- **SE is concerned with cost-effective software development**.





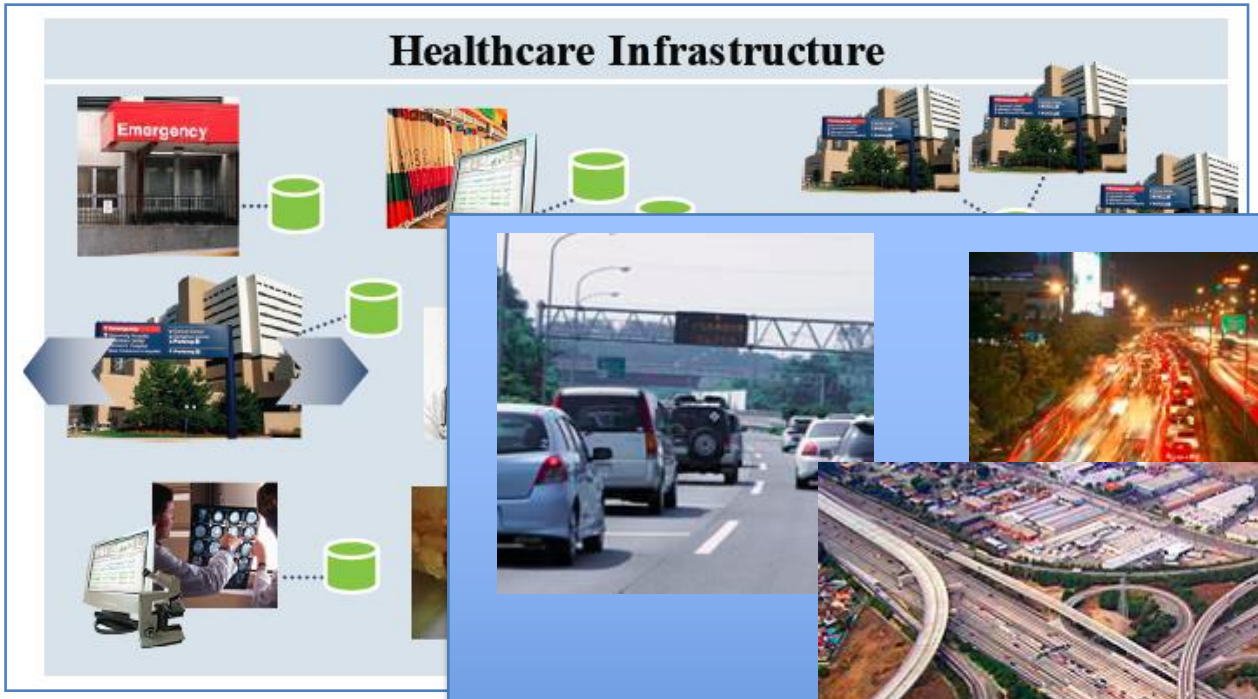


Size of programs continues to grow

- **Trivial: 1 month, 1 programmer, 500 LOC**
  - Intro programming assignments
- **Very small: 4 months, 1 programmer, 2000 LOC**
  - Course project
- **Small: 2 years, 3 programmers, 50K LOC**
  - Pace maker
- **Medium: 3 years, 10s of programmers, 100K LOC**
  - Optimizing compiler
- **Large: 5 years, 100s of programmers, 1M LOC**
  - MS Word, Excel
- **Very large: 10 years, 1000s of programmers, 10M LOC**
  - Air traffic control,
  - Telecommunications, space shuttle
- **Very, Very Large: 15+ years, 1000s programmers, 35M LOC**
  - Windows OS
- **Ultra-Large Scale: ? years, ? developers distributed,**
  - 1000s of sensors, decision units,
  - heterogeneous platforms, decentralized control
  - Intelligent transportation systems; healthcare systems

New Scale:

Healthcare Infrastructure
ULS SW-Intensive Systems



Intelligent Transportation and Vehicle Systems

Software project failure

✧ Increasing system complexity

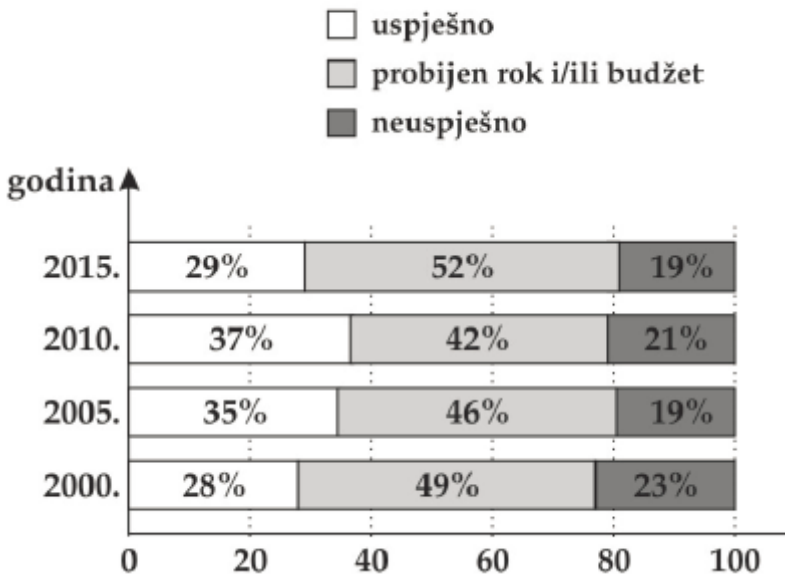
- As new SE techniques help us to build larger, more complex systems, the demands change.
- Systems have to be: built and delivered more quickly; larger, even more complex systems are required;
- Systems have to have new capabilities that were previously thought to be impossible.

✧ Failure to use SE methods

- It is fairly easy to write computer programs without using SE methods and techniques.
- Many companies have drifted into software development as their products and services have evolved.
  They do not use SE methods in their everyday work.
  Consequently, their software is often more expensive and less reliable than it should be.

✧ Too many projects fail

- Arianne Missile 501 (1996), $370 M
- FBI system, (2005), $170 M
- Denver Airport Baggage System, $560 M over budget
- ...



CHAOS Reports

Why is SE needed?

- To predict time, effort, and cost
- To improve software quality
- To improve maintainability
- To meet increasing demands
- To lower software costs
- To successfully build large, complex software systems
- To facilitate group effort in developing software

Frequently asked questions about SE

Question	Answer
What is software?	**Computer programs and associated documentation**. Software products may be developed **for a particular customer** or may be developed **for a general market**.
What are the attributes of good software?	Good software **should deliver the required functionality** and **performance** to the user and should be **maintainable**, **dependable** and **usable**.
What is software engineering?	Software engineering is an **engineering discipline that is concerned with all aspects of software production**.
What are the fundamental software engineering activities?	Software **specification**, software **development**, software **validation** and software **evolution**.
What is the difference between software engineering and computer science?	**CS focuses on theory and fundamentals**. **SE is concerned with the practicalities of developing and delivering useful software.**
What is the difference between software engineering and system engineering?	**System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering.** **Software engineering is part** of this more general process.

Frequently asked questions about SE

Question	Answer
What are the key challenges facing software engineering?	Coping with **increasing diversity**, demands for **reduced delivery times** and **developing trustworthy software**.
What are the costs of software engineering?	Roughly **60% of software costs are development costs**, **40% are testing costs**. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While **all software projects have to be professionally managed and developed**, **different techniques are appropriate for different types of system**. For example, games should always be developed using a series of prototypes, whereas safety critical control systems require a complete and analyzable specification to be developed. **We can not say that one method is better than another.**
What differences has the web made to software engineering?	The web has led to the **availability of software services** and the **possibility of developing highly distributed service-based systems**. Web-based systems development has led to important **advances in programming languages and software reuse**.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that **it can evolve to meet the changing needs of customers**. This is a **critical attribute** because **software change is an inevitable requirement** of a changing business environment.
Dependability and security	Software dependability includes a **range of characteristics including reliability, security and safety**. Dependable software **should not cause physical or economic damage in the event of system failure**. **Malicious users** should not be able to access or damage the system.
Efficiency	Software **should not make wasteful use of system resources** such as memory and processor cycles. Efficiency therefore includes **responsiveness, processing time, memory utilisation**, etc.
Acceptability	Software **must be acceptable** to the type of users for which it is designed. This means that it **must be understandable, usable and compatible** with other systems that they use.

Software products

Generic products

- **Stand-alone systems that are marketed and sold to any customer who wishes to buy them**.
- Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

Customized products

- **Software that is commissioned by a specific customer to meet their own needs.**
- Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Product specification

Generic products

- **The specification** of what the software should do **is owned by the software developer** and decisions on software change are made by the developer.

Customized products

- **The specification** of what the software should do **is owned by the customer** for the software and they make decisions on software changes that are required.

Questions addressed by SE

- How do we ensure the quality of the software that we produce?
- How do we meet growing demand and still maintain budget control?
- How do we avoid disastrous time delays?

Importance of SE

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Historical Perspective

- 1940s: computers invented
- 1950s: assembly language, Fortran
- 1960s: COBOL, ALGOL, PL/1, operating systems
- **1969: First conference (NATO) on Software Engineering**
- 1970s: multi-user systems, databases, structured programming
- 1980s: networking, personal computing, embedded systems, parallel architectures
- 1990s: information superhighway, distributed systems, OO in widespread use
- 2000s: virtual reality, voice recognition, video conferencing, global computing, pervasive computing...
- 2010s: EMRs, autonomous vehicles, new security awareness, ...

General issues that affect software

- **Heterogeneity**
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- **Business and social change**
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available.
  - They need to be able to change their existing software and to rapidly develop new software.
- **Security and trust**
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
- **Scale**
  - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

Software process activities

- **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation. (**WHAT?**)
- **Software development**, where the software is designed and programmed. (**HOW?**)
- **Software validation**, where the software is checked to ensure that it is what the customer requires.
- **Software evolution** (**maintenance**), where the software is modified to reflect changing customer and market requirements.
- **Umbrella Activities**: Throughout lifecycle

Software specification

- **Requirements definition and analysis**
  - Developer must understand
    - Application domain
    - Required functionality
    - Required performance
    - User interface
- **Project planning**
  - Allocate resources
  - Estimate costs
  - Define work tasks
  - Define schedule
- **System analysis**
  - Allocate system resources to
    - Hardware
    - Software
    - Users

Software process activities (continued)

Software development

- **Software design**
  - – User interface design
  - – High-level design
    - • Define modular components
    - • Define major data structures
  - – Detailed design
    - • Define algorithms and procedural detail
- **Coding**
  - – Develop code for each module
  - – Unit testing
- **Integration**
  - – Combine modules
  - – System testing

Maintenance

- **Correction**
  - – Fix software defects
- **Adaptation**
  - – Accommodate changes
    - • New hardware & new policies
- **Enhancement**
  - – Add functionality
- **Prevention**
  - – Make more maintainable

Umbrella Activities

- **Reviews**
  - – assure quality
- **Documentation**
  - – improve maintainability
- **Version control**
  - – track changes
- **Configuration management**
  - – integrity of collection of components

Software Process

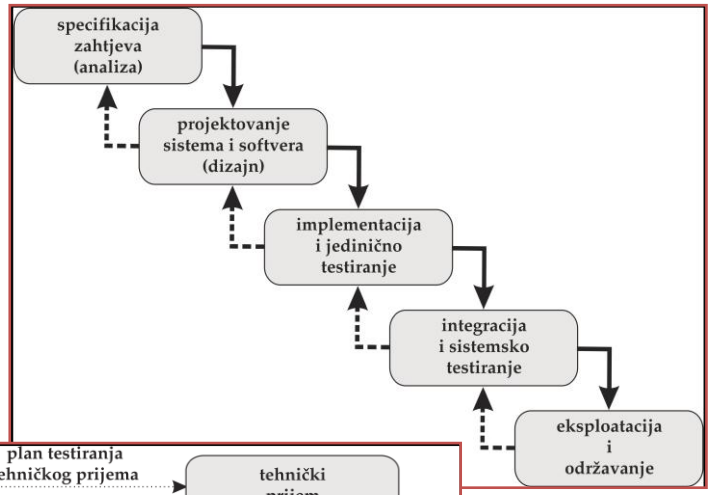
Step-by-step procedure to develop software

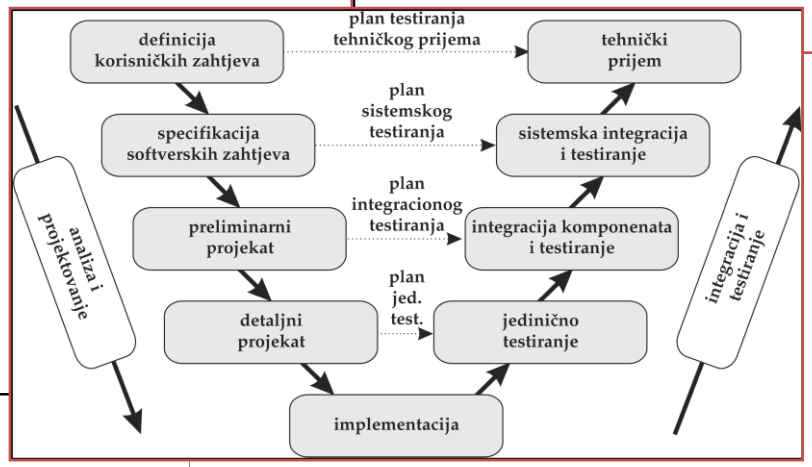
Typically involves the major phases:

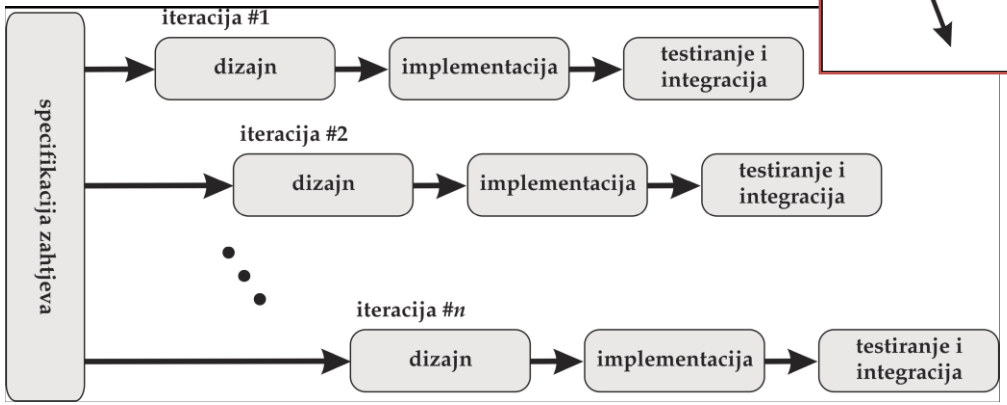
- analysis
- design
- coding
- testing

Different development process models:

- sequential (waterfall, V)
- iterative-incremental (RUP)
- spiral
- agile (SCRUM)







. . .

SE diversity

- There are **many different types of software system.**
- There is **no universal set of software techniques** that is applicable to all of types of software systems.
- The **software engineering methods and tools used depend on:**
  - **type of application** being developed,
  - requirements of the customer,
  - background of the development team.

SE fundamentals

- **Some fundamental principles apply to all types of software system,** irrespective of the development techniques used:
  - **Systems should be developed using a managed and understood development process**.
    Of course, different processes are used for different types of software.
  - **Dependability and performance** are important for all types of system.
  - Understanding and managing the **software specification and requirements** (what the software should do) **are important**.
  - Where appropriate, **we should reuse software** that has already been developed **rather than write new software**.

Application types

Stand-alone applications

- These are application systems that **run on a local computer**, such as a PC.
- They include all necessary functionality and do not need to be connected to a network.

Interactive transaction-based applications

- Applications that **execute on a remote computer** and **are accessed by users from their own PCs or terminals**.
- These include web applications such as e-commerce applications.

Embedded control systems

- These are software **control systems that control and manage hardware devices**.
- There are probably more embedded systems than any other type of system.

Data collection systems

- These are systems that **collect data from their environment** using a set of sensors and **send that data to other systems** for processing.

Batch processing systems

- These are **business systems** that are designed to **process data in large batches**.
- They **process large numbers of individual inputs** to **create corresponding outputs**.

Entertainment systems

- These are systems that are **primarily for personal use** and which are **intended to entertain the user**.

Systems for modeling and simulation

- These are systems that are **developed by scientists and engineers** to **model physical processes or situations**, which include many, separate, interacting objects.

Systems of systems

- These are systems that are **composed of a number of other software systems**.

Why is software development so difficult?

Communication

- **Between customer and developer**
  - Poor problem definition is largest cause of failed software projects
- **Within development team**
  - More people = more communication
  - New programmers need training

Project characteristics

- **Novelty**
- **Changing requirements**
  - 10 x cost during development
  - up to 100 x cost during maintenance
- **Hardware/software configuration**
- **Security requirements**
- **Real time requirements**
- **Reliability requirements**

Personnel characteristics

- **Ability**
- **Prior experience**
- **Communication skills**
- **Team cooperation**
- **Training**

Management issues

- **Realistic goals**
- **Cost estimation**
- **Scheduling**
- **Resource allocation**
- **Quality assurance**
- **Version control**
- **Contracts**

. . .

Software engineering ethics

Ethics vs Law

- Ethics are personal code of behavior
- Law is minimum standard imposed by society
- Law represents will of majority (violation punishable by government)
- Ethics are suggested, not mandated (violation can result in malpractice suit, loss of job, …)

SE ethics

- 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 but involves following a set of principles that are morally correct.

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.

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**.

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) 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.

Rationale for the code of ethics

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

The ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

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

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

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

Ethical principles

1. **PUBLIC** - Software engineers shall act consistently with the public interest.

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

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

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

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

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

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

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

Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- The high-level activities of specification, development, validation and evolution are part of all software processes.
- The fundamental notions of software engineering are universally applicable to all types of system development.
- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- The fundamental ideas of software engineering are applicable to all types of software system.
- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.