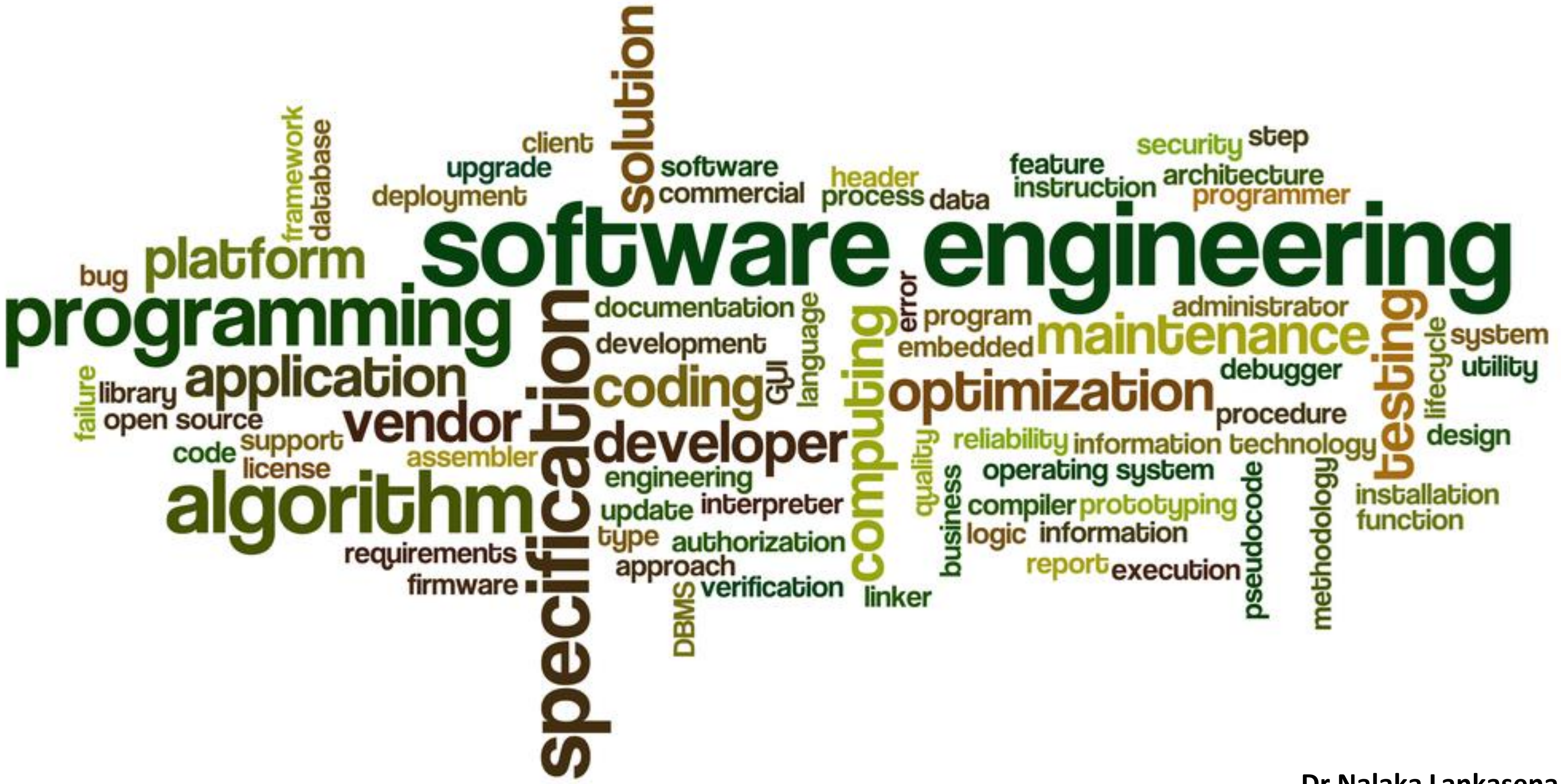


Dr Nalaka Lankasena
Faculty of Technology
University of Sri Jayewardenepura

Table of Content

Essential attributes of good software/ software quality

Essential attributes of good software/ software quality

Functionality: This refers to the software's ability to perform the tasks and functions it is designed for. It includes the features and capabilities that the software provides to users. Functionality is a crucial aspect of software as it determines whether the software can meet the user's requirements.

Performance: Performance involves the software's responsiveness and efficiency in terms of speed, processing power, and resource utilization. Well-performing software should execute tasks in a timely manner and manage system resources efficiently.

Reliability: Reliability relates to the software's ability to perform consistently and accurately under various conditions. Reliable software should deliver consistent results and minimize errors. It's important that software behaves as expected without unexpected crashes or failures.

Contd…

..contd.. Essential attributes of good software/**software quality**

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 unavoidable requirement** of a changing business environment.

Usability: Usability refers to how easily and effectively users can interact with the software to achieve their goals. User interfaces should be intuitive, user-friendly, and efficient, allowing users to perform tasks with minimal confusion and frustration.

Scalability: Scalability is the software's ability to handle increased workloads or growing numbers of users without a significant decrease in performance. Scalable software can adapt to changing demands without compromising its effectiveness.

..contd.. Essential attributes of good software/**software quality**

Portability: Portability is the software's ability to function across d**ifferent platforms and environments without significant modifications**. Portability allows software to be deployed on various **operating systems and devices** with minimal effort.

Security: Security is a critical attribute that ensures the protection of data and prevents unauthorized access or malicious attacks. Software should have robust security mechanisms to safeguard sensitive information and maintain user privacy.

..contd.. Essential attributes of good software/**software quality**

Interoperability: Interoperability is the ability of software to communicate and work seamlessly with other software systems or components. Interoperable software can exchange data and services effectively, enabling integration within complex ecosystems.

Compliance: Compliance refers to the software's adherence to industry standards, regulations, and legal requirements. Depending on the domain, software may need to meet specific standards to ensure its legality, safety, and effectiveness.

Testability: Testability is the ease with which software can be tested to identify defects, errors, and vulnerabilities. Testable software should have clear requirements, well-defined behavior, and a design that facilitates comprehensive testing.

Fundamental software engineering activities/processes?

- **Software specification:** Customers and engineers define the software that is to be produced and the constraints on its operation
- **Software design & development:** Software is designed and programmed
- **Software validation:** Software is checked to ensure that it is what the customer requires
- **Software evolution**: Software is modified to reflect changing customer and market requirements

General issues that affect software: Challenges in the Software Industry

General issues that affect software: Challenges in the Software Industry

- **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**.

- **Heterogeneity**
  - Increasingly, systems are required to operate as distributed systems across networks that include **different types of computer and mobile devices**.

.. Contd..

..contd.. General issues that affect software: Challenges in the Software Industry

- **Security and trust**
  - As software is connected very closely 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 engineering diversity and software techniques

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

Application types

Examples

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.

… contd…

..contd Application types

Industrial Robots
GPS
GPS Receivers
Digital Cameras
DVD Players
Embedded Systems
Wireless Routers
MP3 Players
Set top Boxes
Gaming Consoles
Photocopiers
Microwave Ovens

Embedded control systems

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

Embedded systems are used in navigation tools like global positioning system (GPS), automated teller machines (ATMs), networking equipment, digital video cameras, mobile phones, aerospace applications, telecom applications, etc

… contd…

..contd Application types

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.

- Example: *end-of-day (EOD)* jobs in Banks
  - This include interest calculation, generation of reports and data sets to other systems, printing statements, and payment processing. This coincides with the concept of cutover, where transaction and data are cut off for a particular day's batch activity and any further data is contributed to the next following day's batch activity (this is the reason for messages like "deposits after 3pm will be processed the next day").

… contd…

..contd… Application types

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.

… contd…

..contd. Application types

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.

Systems of systems

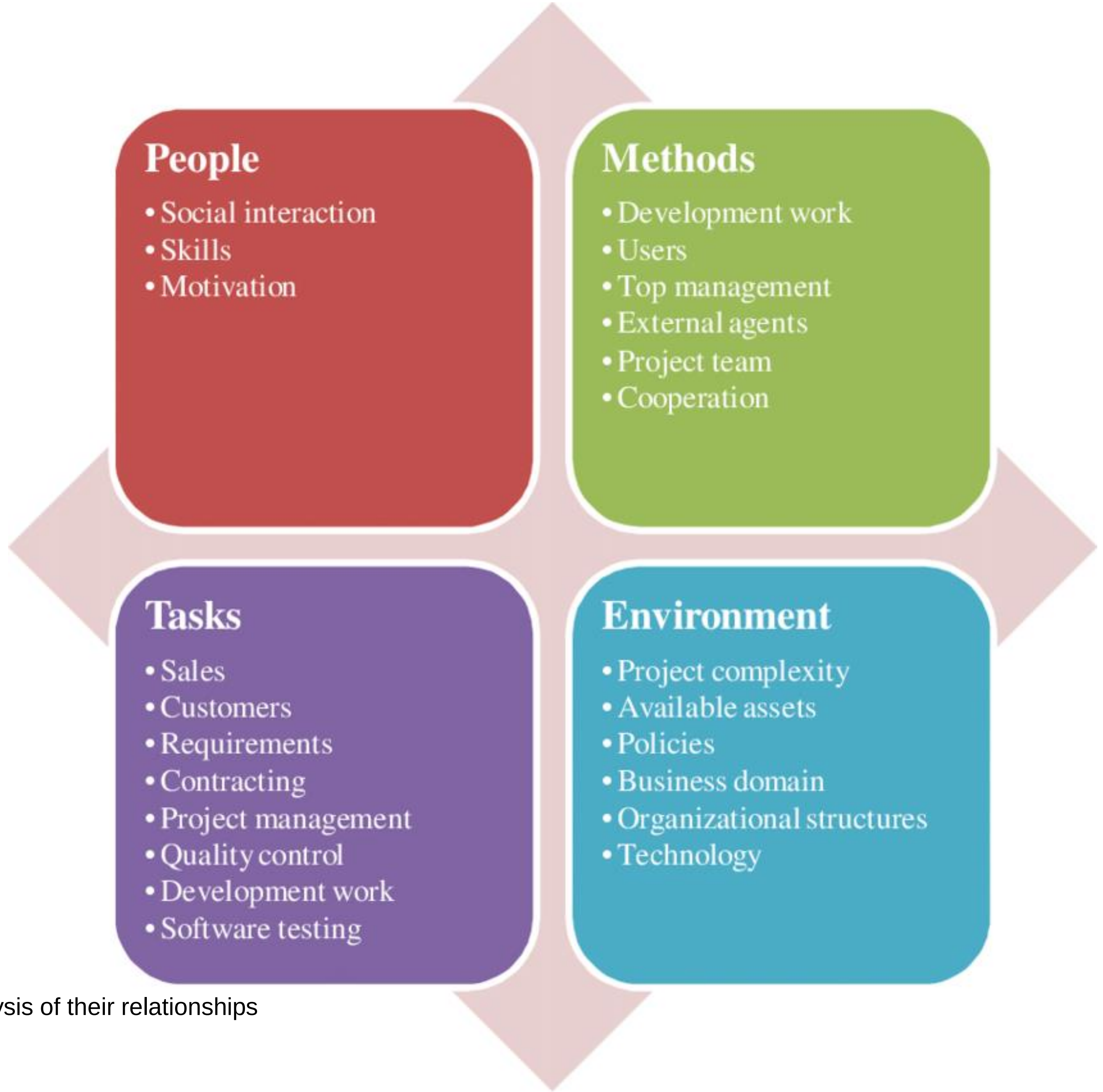
- These are systems that are composed of a number of other software systems.
- Some of these may be generic software products, such as a **spreadsheet program**. Other systems in the assembly may be specially written for that environment.

Software Project failures

- The discipline of software engineering (SE) was born in 1968 due to software project failures.
- **Preventing software project failures** is the main objective of software process improvement as it aims at lowering the costs of development work, shortening the time to market, and improving product quality.
- There are still many reports of software projects going wrong and ‘software failures’
- Software engineering is criticized as inadequate for modern software development.

..contd..

Summary of the common causes of software project failures



Reference: Perceived causes of software project failures: An analysis of their relationships
Juha Itkonen, Vanhanen Jari-Pekka, Timo Lehtinen

Software Project failures

- examples?

…. Contd.. Software Project failures

A few examples

- Incomplete requirements
- Lack of user involvement
- Lack of resources
- Unrealistic expectations
- Lack of executive support
- Changing requirements and specifications

- The reasons for software project failure have to do with processes for determining **What is desired?, What is needed?, What is possible?, Who knows?, Who decides?, What is lacking?** among these often does not become evident until the system is implemented.
- So the critical factors for a successful project are methods and mechanisms that can draw out this information at the beginning of a project, and throughout it.

(Ref Barry Boehm, IEEE Software, September/October 2002, page 97)

..contd..

Software engineering 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.
- Adhering to software engineering ethics helps **maintain the trust of users, promotes responsible technological advancement, and contributes to a more ethical and sustainable digital landscape**.

Why ethics are important for Software Engineers?

Why ethics are important for Software Engineers?

User Safety and Well-being: Ethical conduct ensures software is safe, reliable, and doesn't harm users.

Public Trust: Ethical behavior maintains user trust and fosters positive relationships.

Social Responsibility: Engineers must consider societal implications and avoid harm to marginalized groups.

Avoiding Harm: Ethical guidelines prevent the creation of harmful software and unintended consequences.

Legal Compliance: Ethics ensure adherence to laws and regulations, reducing legal risks.

Long-Term Viability: Ethical decisions lead to sustainable solutions, preventing negative long-term impacts.

Reputation and Professionalism: Ethical behavior enhances professionalism and industry reputation.

Collaboration: Ethics encourage teamwork, improving software solutions and addressing diverse needs.

Global Impact: Ethical choices affect worldwide cultures, economies, and societies.

Ethical Principles for Software Engineers …contd..

User-Centricity: Prioritize the needs and well-being of users in software design and development.

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 without their competence.

Transparency: Communicate openly about software capabilities, limitations, and potential risks.

…Contd..

Ethical Principles for Software Engineers ...contd..

- **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 importance (game playing on an employer's machine) to extremely serious (dissemination of viruses).**
- **Avoid Harm:** Develop software that avoids causing harm, whether physical, financial, or psychological.

Ethical Principles for Software Engineers

Accountability: Take responsibility for software quality, security, and the impact on users and society.

Professionalism: Maintain competence, continuously learn, and uphold the reputation of the profession.

Q&A
Questions
Answers

THANK
YOU