Програмна Інженерія

Software Engineering

SE

Prehistory of SE

- **The early 1940s** -- first digital computers and main computing abstractions:
  - design was not flexible -- instructions were wired into the machine
  - came up with the "stored program architecture" or von Neumann architecture
  - division between "hardware" and "software" began with abstraction being used to deal with the complexity of computing
- **The 1950s** -- programming languages -- another major step in abstraction
  - Fortran, ALGOL, and COBOL were released to deal with scientific, algorithmic, and business problems respectively

SE history: maturity period

- **1968** -- term "**software engineering**“ -- Anthony Oettinger, Margaret Hamilton
  - The same principles used in engineering can be applied to software
  - E.W. Dijkstra - **goto** statement considered harmful [1968]
  - David Parnas - concept of modularity and information hiding [1972] to deal with the ever increasing complexity of software
  - Barry Boehm - SE Economics [1981]

    PM=A*KSLOC^(1+x) -- software versus hardware costs
    - **PM** - software development effort for a program, in Person-Mans
    - **KSLOC** -thousand source lines of code, **A** - calibration constant (based on project data)
    - **(1+x)** - an exponent for the software diseconomy of scale
- **1984** --Software Engineering Institute (SEI) (Carnegie Mellon University in Pittsburgh, US)
- **SWEBOK** (**Software Engineering Body of Knowledge**) -- modern, generally accepted best-practices for software engineering have been collected by the ISO/IEC JTC 1/SC7 subcommittee

A Computing Curricula Series Report

Computing Curricula 2020

CC2020

Paradigms for Global Computing Education

encompassing undergraduate programs in

Computer Engineering

Computer Science

Cybersecurity

Information Systems

Information Technology

Software Engineering

with data science

Association for Computing Machinery (ACM)
IEEE Computer Society (IEEE-CS)

https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

Landscape of computing education CC2020

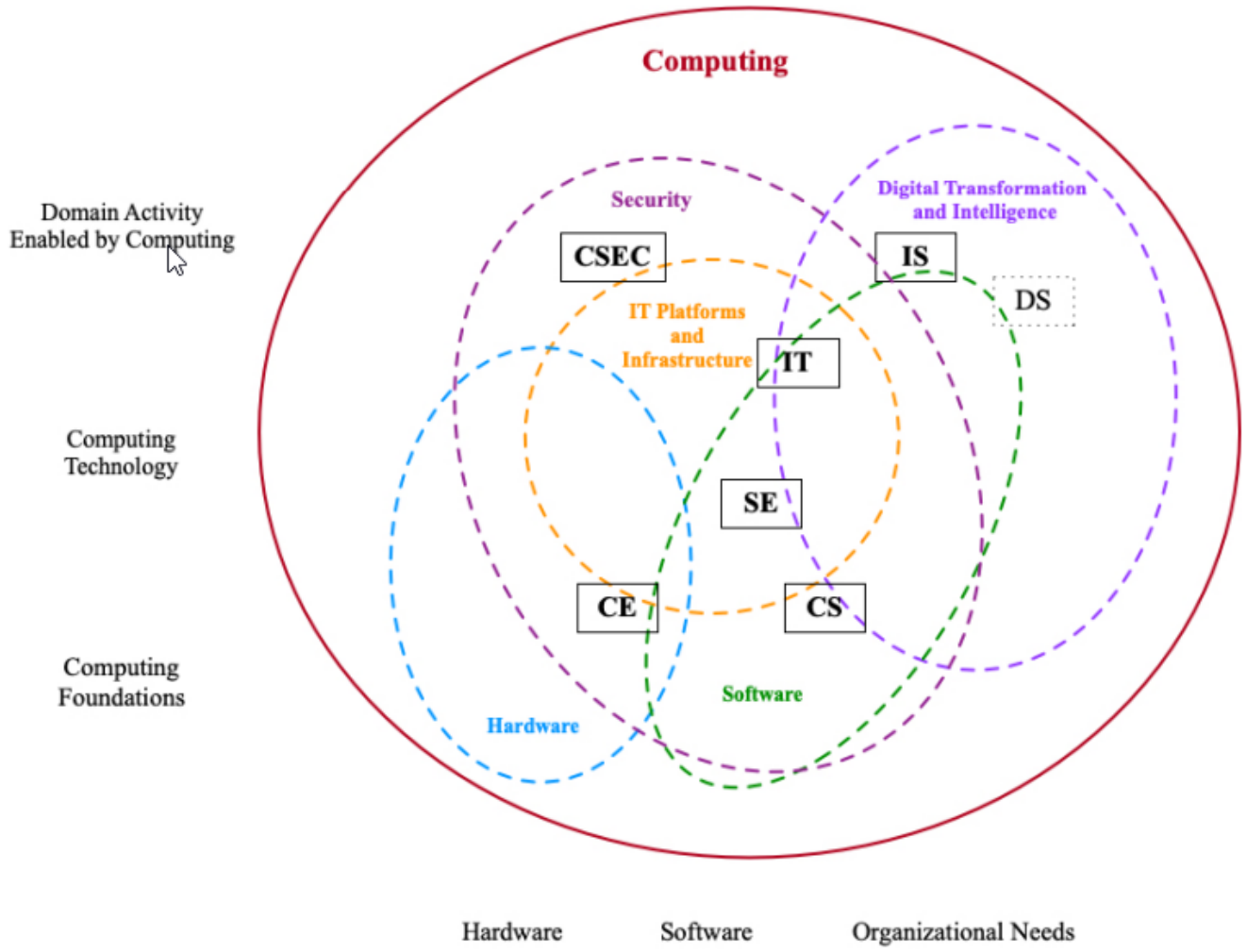


Figure 2.2. A contemporary view of the landscape of computing education

Legend: Curricular reports: CE=computer engineering; CS=computer science; CSEC=cybersecurity; IS=information systems; IT=information technology; SE=software engineering; DS=data science (under development).

Computing Interrelationships

- Foundations, technology, domain activity -- **three levels of computing** as related to hardware, software, and organizational needs
- Information technology platforms and infrastructure capture the **integration** of hardware and software **into technology solutions** that enable computing-based solutions having capabilities associated with data storage, processing, artificial intelligence, and visualization
- CE, CS, SE **provide the components** required for these computing technology capabilities to exist
- IT focuses on making and keeping components available for individual and organizational users
- Digital intelligence and transformation covers the capture, management, and analysis of data enabling individuals, organizations, and societies to conduct their activities in a way that helps them better achieve their goals
- IS and DS enable digital intelligence and transformation
- Security permeates the entire space of computing

SE components

- **Requirements**: elicitation, analysis, specification, and validation of requirements
- **Design**: defining the architecture, components, interfaces, and other characteristics of a system or component
- **Construction**: creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging
- **Testing**: empirical, technical investigation conducted to provide stakeholders with information about the quality of the product or service under test
- **Maintenance**: activities required to provide cost-effective support to software
- **Configuration management**: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle
- **Management**: planning, coordinating, measuring, controlling, and reporting
- **Quality management**: degree to which a set of inherent characteristics fulfills requirements
- **Process**: definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself
- **Tools and methods**: making the activity systematic and ultimately more likely to be successful

SE in Computer Science

Computer science is the scientific and practical approach to computation and its applications.

- **Theoretical computer science**
  - Theory of computation
  - Information and coding theory
  - Algorithms and data structures
  - Programming language theory
  - Formal methods
- **Applied computer science**
  - Artificial intelligence
  - Computer architecture and engineering
  - Computer performance analysis
  - Computer graphics and visualization
  - Computer security and cryptography
  - Computational science
  - Computer networks
  - Concurrent, parallel and distributed systems
  - Databases
  - **Software engineering**

Computer Science Curricula 2013(XX)

- SE is the discipline concerned with the application of theory, knowledge, and practice to effectively build reliable software systems that satisfy the requirements of customers and users
- Elements of SE are applicable in all areas of computing by a wide variety of engineering methods, processes, techniques, and measurements
- Professionalism, quality, schedule, and cost are critical to producing software systems in every computing application domain
- SE is applicable to small, medium, and large-scale systems
- The strong focus of SE is on the design of reliable, trustworthy, secure, and usable software systems

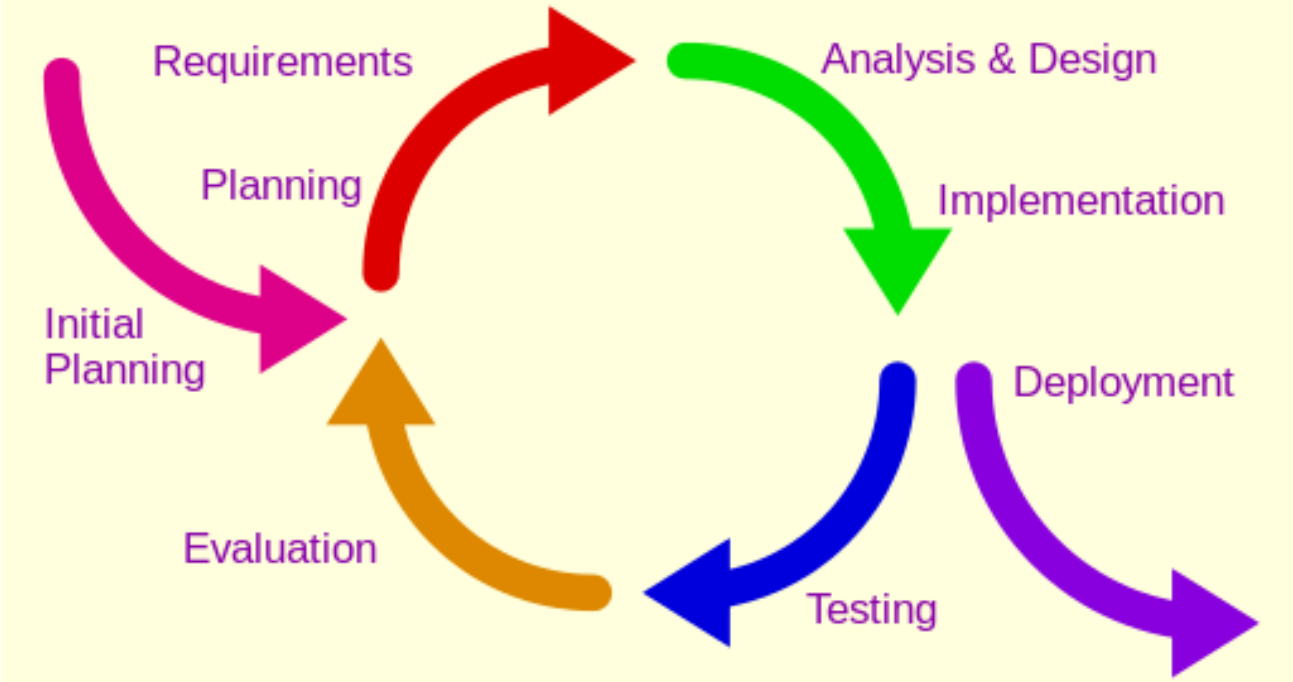
OO A&D as SE technical approach

- for analyzing, designing an application, system, or business
- by applying the object-oriented paradigm and visual modeling
- throughout the development life cycles
- to foster better stakeholder communication and product quality

Software Development Processes

- is a division of software development work into distinct phases (or stages) containing activities with the intent of better planning and management
- also known as :
  - software process
  - software development life cycle
  - software development methodology
  - system development methodology

Iterative development model



RUP Model

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.