CSE344
Software Engineering (SWE)

Week – 1
Introduction

Historical Definition

- The term *Software Engineering* first introduced in the first NATO conference in 1968.
- It was *defined then* as follows [2]:
  - The establishment and use of *sound* engineering principles in order to *economically* obtain *reliable* software that works *efficiently* on *real machines*.

About Historical Definition

- The definition was very modern since it is still valid.
- Software engineering
  - is *disciplined* engineering work,
  - offers *means to* build *high-quality*, *efficient* software at *affordable prices*, and
  - offers *task allocation* and *tools* for all software building phases.

Preliminary Questions on SWE

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- What is the difference between software engineering and system engineering?
- What is a software process?
- What is a software process model?
- What are the costs of software engineering?
- What are software engineering methods?
- What is CASE (Computer-Aided Software Engineering)
- What are the attributes of good software?
- What are the key challenges facing software engineering?

What is SW?

- SW is an *interconnected set of computer progams and associated documentation* such as
  - data and configuration files, requirements, design models, test plans, cases and results, user and technical manuals
- Two types of SW products
  - *generic*: developed to sell to multiple customers
  - *bespoke* or *custom*: developed for a single customer

What is SWE?

- SWE is an *engineering discipline* concerned with all aspects of *software production*.
- Software engineers should
  - adopt a *systematic and organised approach* to their work, and
  - use *appropriate tools and techniques* depending on
    - the problem to be solved,
    - the development constraints and
    - the resources available.

What is the difference between SWE and computer science (CS)?

- CS ↔ *theory and fundamentals*
- SWE ↔ *practicalities of developing and delivering useful software*.
- CS theories still insufficient as a complete foundation for SWE unlike e.g. physics and electrical engineering.

What is the difference between SWE and system engineering (SyE)?

- *A computer-based system* ↔ a set of interrelated components
  - *hardware* and *software* components,
  - users and use environments.
- That is, *SW components are a part of the computer-based system*.
- SyE ↔ computer-based systems development including
  - hardware,
  - software and
  - process engineering.
- SWE is part of this process concerned with developing
  - the SW infrastructure,
  - control,
  - applications and
  - databases in the system.

What is a SW process?

- A set of activities to develop/evolve SW.
- *Generic activities* in all SW processes are:
  - ***Specification*** - what the system should do and its development constraints
  - ***Development*** – how to produce the SW system
  - ***Validation & Verification (V&V)*** - checking that
    - the SW product is what the customer really wants, and
    - the SW product does what it is supposed to.
  - ***Evolution*** - changing the software in response to changing demands.

What is a SW process model?

- A *simplified representation* of a SW process
  - presented *from a specific perspective*.
- Examples of *process perspectives*
  - *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***.

What are the costs of SWE?

- Roughly,
  - ***60% ↔ development costs***,
  - ***40% ↔ testing costs***.
- For ***custom SW***, *evolution costs often exceed development costs*.
- Costs vary depending on
  - the *type of system being developed* and
  - the *requirements of system attributes* such as *performance and system reliability*.
- Distribution of costs depends on *process model* used.

What are SWE methods?

- *Structured approaches to SW development* such as
  - system models, notations, rules,
  - design advice and process guidance.
- *Model descriptions*
  - Descriptions of graphical models. They *should be* produced;
- *Rules*
  - Constraints applied to system models;
- *Recommendations*
  - Advice on good design practice;
- *Process guidance*
  - What activities to follow.

What is CASE (Computer-Aided Software Engineering)?

- Software systems that *provide automated support* for SW 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*.

What are the attributes of good SW?

- SW should deliver the *required functionality* and *performance* to the user and should be *maintainable*, *dependable and acceptable*.
- ***Maintainability***
  - Software must evolve (i.e., be changed) to meet changing needs;
- ***Dependability (Reliability-Safety-Security)***
  - Software must be trustworthy;
- ***Efficiency***
  - Software should not make wasteful use of system resources;
- ***Acceptability***
  - Software must accepted by the users for which it was designed. This means it must be *understandable, usable and compatible* with other systems.

Supplementary slide – 1... Reliability

Reliability (Güvenilirlik): The ability of the SW product to function within a prespecified span of time to fulfill a specific purpose under certain conditions as expected.

Supplementary slide – 2... Safety

Safety (Güvenlik): The capability of the SW product to end the functioning of the system it controls/manages without ever damaging the system's environment even if the SW product itself fails.

Supplementary slide – 3... Security

Security (Güvenlik): The ability of the SW product to protect itself (and so the system it controls) from any external attacks or unauthorized use/access of system services that be either accidental or deliberate.

What are the key challenges facing SWE?

- *Heterogeneity, delivery and trust*.
- ***Heterogeneity***
  - Developing techniques for building software that can cope with heterogeneous platforms and execution environments;
- ***Delivery***
  - Developing techniques that lead to faster delivery of SW;
- ***Trust***
  - Developing techniques that demonstrate that SW can be trusted by its users.

A Brief History of SWE

1950's... First Commercial SW

- First commercial SW appeared in 1951 in England, made by the company J. Lyons.
- Then,
  - developing SW was
    - *straightforward and*
    - *uncontrolled*.
  - SW could be relatively *versatile* (i.e., adaptable, multi-purpose).

Early 1960's... SW Crisis

- Early 1960's... First appearance of terms such as
  - *testability*
  - *interfaces*.
- *SW development still not structured*.
- Average size of SW grew and problems got worse.
- Projects started to miss their budget and schedule.
- Unstructured SW was not good enough any more.
- ***The SW crisis of 1960's!!!***

Late 1960's... First SWE Conferences

- 1968... NATO organized the first SWE conference
- The term "***Software Engineering***" was introduced.
- 1969... The second SWE conference by NATO.
- The *basis of "Software Engineering"* was defined in these two conferences.

Results of the first Conferences

1. These conferences did not solve the SW crisis.
2. The crisis seems not to be solved even today. (Why? How to see this?)
    1. Req's not well established (why?)
    2. SW testing still a problem (why?)
    3. Many SW projects still go unfinished (miss their budget and schedule) (how to see?)
3. However, ideas emerged then brought about a solid foundation for "*modern Software Engineering*."

1970’s... SW Process Models

- Early 1970’s... The first *SW process model* ***waterfall*** model was defined.

- The *waterfall* model defined a *systems engineering standard* and it was originally used for *hardware engineering* fields.

1970's and 1980's...More on Process Models

- The systems engineering model was good enough for other engineering branches, but not for SWE. (Why?) Most importantly product intangibility
- Thus, in 70's and 80's several *new software process models* were introduced

Programming Languages

- 1950's... Machine language was employed to develop the first programs, but *first* high-level programming languages were already known at.
- Early 1960's... ***Cobol and PL/1*** were introduced.
- ***Late 1960's ...*** *Introduction of* ***C***.

1980's... Object-Oriented Programming (OOP)

- 1980's... *OOP* is proposed.
- 1986... The *first conference OOPSLA* on OOPs.
- OO philosophy was long since discussed, but the first OOP languages started to solidly use the principles.
- OO philosophy spread slowly from languages to all SWE fields.

Personal Computing

- Early 1990's... SWE quite stable...
- The term ***personal computing*** emerged and changed needs for SW but not the techniques of building SW.
- *Computer-Aided Software Engineering (CASE)* tools or ***CASE-tools*** appeared and evolved fast.

1999... Agile Process Models

- 1999... Introduction of ***Extreme Programming (XP)***
- *XP*, an *agile process model*, was the most recent change in SWE
- Agile process models soon became popular among programmers and small/medium companies because XP is a suitable model to use to develop small/medium-scale SW projects for both design and programming issues.

Current... Distributed SW systems

- Currently, the term ***distributed SW architecture*** is an essential concept to explain today's SW products.
- Although the basic principles of SWE have not changed much, companies develop their own way of handling SW process. They use their ***specialized*** process models, methods and tools.

Current... Specialized SW Products

- SW is specialized from company to company.
- For any-scale SW products ranging from
  - *small-scale SW* (e.g., cell phone SW and other mobile equipment) to
  - *large-scale SW* (e.g., jumbo airplane (BOEING) navigation systems, complex weapon control systems, nuclear reactor systems),
- the *need for specialization is to remain in the market and keep or enhance their share in the market*.

Future...?

- Three potential challenges in the future:
  - Heterogeneity challenge.
  - Delivery challenge.
  - Trust challenge.
- They are not independent; may even conflict with each other.

Heterogeneity Challenge

- Software-based systems have to work in *various environments cooperating with different types of systems*.
- Software has to be built to *work both with current and legacy (old) systems*.
- Legacy systems often need new functionality.
- The life span of old SW is often expanded beyond natural limits.

Delivery Challenge

- SW development takes unexpectedly much time. Time is an expensive resource.
- Business is an extremely dynamic domain and requires any system it involves be ready for changes.
- So, fast development of maintainable/evolvable SW is inevitable and the maintenance to SW should not compromise from quality.

Trust Challenge

- Software systems affect more and more our lives.
- The more they control, the more we need to trust them.
- Trust does not come easily. One has to be able to see from software that it works as planned.
- Currently such negative phenomena as spam, viruses and worms damage trust.

References

[1] Ian Sommerville, Software Engineering, 8th ed. 2007

[2] P. Naur, R.Randell (eds.): Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee, 1968