I. Introduction

Who am I?

Andrea Polini

Software Engineering Laboratory

Istituto di Scienza e Tecnologie dell'Informazione “Alessandro Faedo”

Area della ricerca del CNR di Pisa

Via Moruzzi, 1

56124 Pisa (ITALY)

e-mail: andrea.polini@isti.cnr.it

web: http://www1.isti.cnr.it/~polini

Objectives of today lesson

- Introduce the course:
  - Organisation and Contents
  - Exam organisation
- SE, a bit of history
- SE definitions and key concepts
- SE, basic concepts
  - Important factors of SE
  - product vs. process
  - Quality in SE

Organisation

- Lessons:
  - Until November 17: Monday 4pm-6pm – Tuesday 11am-1pm
- SE Laboratory:
  - Starting December (??) – Oliviero Riganelli
  - Monday 6pm-7pm – Tuesday 3pm-5pm
- When you can find me:
  - Tuesday 10am-11am (??)
- Where
  - ??

Exam

- **1st option**
  - Two parts (project and questions)
    - Define groups of student (at least 3 at most 5 students for each group)
    - Assign projects to each group (by mid of November)
- 2nd option
  - Report on a specific subject related to SE (group of 2 or 3) and final exam with questions on the whole program

Course Structure 1/2

- Introduction
- Software Development Processes
- Requirements
  - Types of requirements
  - Elicitation, Analysis, Validation, Management
  - Requirements Documentation and Modeling
- Design
  - Decomposition and Modularity
  - Software Architecture
  - Distributed systems architecture
  - Object oriented design
  - Design by Contract
  - Design Patterns

Course Structure 2/2

- Coding
  - Programming Guidelines
  - Agile methodologies
  - Component Based Software Development
- Verification and Validation
  - Code inspection
  - Testing
  - Code-Based Testing
  - Model-Based Testing
  - Concepts of Formal Testing (??)

What will be your skill at the end

- You should be able to:
  - Understand the basic concept and motivation of software engineering
  - Understand difficulties and the issues affecting the development of complex software systems
  - Select and judge different techniques and tools to carry on the development of a complex system
- Your analysis and synthesis skills will have been enriched with methodologies, techniques and tools allowing you to approach more complex problems

Course Materials

- Reference book:

  Ian Sommerville: "*Software Engineering*" - Addison Wesley

  (Italian Edition: "Ingegneria del Software" - Pearson Education)

  Chapter: 1-4, 6-14, 17-24

- Additional material provided during the course covering specific subjects (mainly in english)
- Web:
  - http://www1.isti.cnr.it/~polini/SEcamerino0607.html
- Blog:
  - http://softengcamerino.blogspot.com
- **Comments on any aspect of the course are really welcome!!!**

A bit of history

- Why we need Software Engineering?
- Late '50s higher level programming languages (COBOL)
  - Programs started to be more complex
  - Programming become a job
- Software crisis – 60's/70's
  - Developer were not anymore able to develop software taking advantage of hardware improvements
  - Development methodologies did not scale up to produce larger software
- Many solution were proposed
  - *different team organisations*
  - *definition of standards*
  - *application of more rigorous and formal approaches to programming*

A bit of history

- Emergence of a new discipline called Software Engineering
  - This term was first used in 1968 at a NATO conference in Garmisch - Germany
- Building software systems is a complex task that asks for an engineered approach
  - Methodologies
  - Tools
  - Organization
  - Theories
  - Etc...

Still a complex discipline

- Software Complexity continued to increase
- Many software projects fail or are canceled after starting
- Often software projects are late!
- Some famous software failures
  - Ariane 5
  - Therac25
  - London ambulance service
  - Denver Airport

Software Engineering definitions

- IEEE
  - *Application of systematic, disciplined, **quantifiable** approach to the **development, operation, and maintenance of software***
- Sommerville
  - *Software engineering is an engineering discipline that is 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.*
- *Ghezzi, Jazayeri, Mandrioli*
  - *Software Engineering is the field of computer science that deals with the **building of software systems that are so large or so complex** that they are built **by a team or teams of engineers***

Software Engineering definitions

- Emmerich
  - Software engineering is the branch of systems engineering concerned with the development of **large and complex software intensive systems**. It focuses on: the real-world goals for, services provided by, and constraints on such systems; the precise specification of system structure and behaviour, and the implementation of these specifications; the activities required in order to develop an assurance that the specifications and real-world goals have been met; the evolution of such systems over time and across system families. **It is also concerned with the processes, methods and tools for the development of software intensive systems in an economic and timely manner**

Key points

- SE methodologies to be applied for the development of **mid to large systems**
- **Disciplined** and **systematic** development
- **Quantifiable** – methods to objectively judge different solutions are necessary
- Involve more than one person
- Does not only concern programming
- Maintenance and evolution
- **Development cost and time are key issues**
- Asks for management responsibilities and abilities

Important Factors in Today SE Practice

- Development Process should be highly iterative
- Object-oriented technology
- Powerful Desktop computing
- Reduced Time to market
- Networking and distribution – mobile devices
- GUI (windows, menus, icons, pointers)
- Reduced hardware cost and increased software cost

Some Relevant “tools” for SE practice

- Abstraction
- Analysis and Design Methods and Notations
- Software Process
- Measurement
- Tools and Integrated Environments
- ...

Abstraction

- Abstraction: description of a problem at a level of generalization that **allow to concentrate on relevant key aspects hiding details** which we do not consider important to find a solution.
  - Generalization is one of the typical tool useful to abstract away details
- **Subjective process** if your abstraction is not correct (with respect your target) neither can be your solution
  - Consider the develoment of a mechanisms passing data among processes on different machine on which different protocols are available
    - At a first step you will probably ignore the type of the data focusing on understanding how to transfer it and making some performance consideration on the base of the available bandwidth.
    - Nevertheless probably you cannot ignore some protocol characteristics such as **synchronous or asynchronous behavior**. Instead you should group protocols on the base of this kind of behavior
- It is really difficult (if possible) to “teach” abstraction

Analysis and Design Methods and Notations

- SE applies to team work that need to **exchange infos**
- We need **mechanisms to describe our abstractions** to **reason about** them and to **communicate** them to other people in a way that they can be correctly interpreted.
- Sharing requires common languages and agreed meaning
- Reasoning requires formal descriptions
- Many solution proposed in different context with different level of formality to describe systems:
  - UML, SDL, Graph based languages, Z, PetriNet, etc..

Software Process

- What is a software process? (S*ommerville)*
  - set of activities whose goal is the development or evolution of software.
  - Generic activities in all software processes are:
    - Specification - 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.

Measurements

- *"Formally a measure is a mapping from a set of entities and attributes in the real empirical world to a representation or model in the mathematical world ... in order to obtain more information and understanding about the real world"* *Shari Pfleeger*
- It is important to define **mechanisms** that allow us **to compare** different solution on a quantitative basis and that allow us to **predict software qualities**
- We would like to answer to question such as:
  - Is the system X more reliable then system Y?
  - Is it convenient to enlarge a development team?
  - Which kind of guarantees give me the execution of a test suite?
  - ...

Tools and Integrated environments

- As in other engineering disciplines we need tools that drives us and assist in the development of software (e.g. AutoCAD in other area)
- Many different tools have been defined generally focusing only on some activities (design, testing etc..)
- **Interoperability issues** using different tools
- Computer Aided Software Engineering (**CASE**)

- **Eclipse** framework an interesting proposal integrating many different tools thanks to the **Plug-in based architecture**

Product vs. Process

- SE concerns two main elements:
  - Product – **what** we want to develop and deliver to a client
  - Process – **how** we develop, deliver, and maintain a product

Quality of Software

- Another important concept in SE is “Quality”
  - Main objective is to provide techniques and tools to increase the quality of the delivered system
  - i.e. definition of programing languages reducing number of errors introduced by to the programmer
  - how we can evaluate properties and develop product with given properties?
- Assumption: improvement to the development process will cause positive feedback on the product quality
  - i.e. introduce a verification step during development should reduce the number of errors in the code

Quality

- Is it quality a precisely defined concept?
  - Intuitively quality evaluation is generally subjective and strongly related to the context!!
- Qualities can be related both to products and to processes
- In our context a quality is meaningful only if can be measured
  - Definition of measures providing objective judgments of qualities really important and complex topic in SE (not treated in this course)
- SE methodologies should lead us to the development of systems anticipating and satisfying relevant qualities
  - Following quality list taken from Ghezzi, Jazayeri, Mandrioli book “Fundamentals of SE”

Correctness

- A program is functionally correct if it behaves according to its stated functional specification.
  - We need a spec suitable for comparing
  - Testing or formal proof can be used to asses such quality depending on how the spec has been described

Correctness

- Spec: develop a mathematical library supporting calculation on natural number

```
public class math1{
        public double sum(double x, double y) {
                return x+y;
        }
        public double subtract(double x, double y) {
                return x-y;
        }
        public double abs(double x) {
                if (x>0) {return x;}
                else {return x;}
        }
....
}
```

```
public class math2{
        public int sum(int x, int y) {
                return x+y;
        }
        public int subtract(int x, int y) {
                return x-y;
        }
        public int abs(int x) {
                if (x>0) {return x;}
                else {return -x;}
        }
....
}
```

- Which one is a correct implementation of the Spec?

Reliability

- **Probability** that a software will **operate as expected** over a specified time interval
- This quality is strongly dependent on the usage of the system.
- In some context a fault in the implementation can be hidden by the usage
  - When integer are considered Math1 is probably more reliable. But consider the case when it is used with a spec talking of real numbers
- Almost every time software is released with bugs. Different user will experiment different reliability!

Availability

- **Probability** that a system at a certain point in time **will be operational and able to deliver the requested service**
- Some systems can have critical requirements on availability but more relaxed on reliability
- e.g. telephone switching system

Robusteness

- A program is robust if it behaves **reasonably even in circumstances that were not anticipated in the requirements specification** – for example when it encounters incorrect input data or some hardware malfunction
- If the specification defines the behavior also for “incorrect” data we will check correctness.

- Spec: ...on non natural number the program should raise an exception

```
public int abs(int x) throws Exception {
    if (x>0) {return x;}
    else {System.out.println(“WARNING negative number”); throw new Exception(); }
}
```

Performance

- Performance in general refer to the number of request that can be served in a fixed time (**Throughput**)
- At the same time can refer to the time that it is necessary to deliver the service (**Latency**)
- Performance is not the same as **efficiency** that relates to the usage of computational resources (memory, cpu, devices, etc..) to provide the service

Usability

- This properties is related to how much a **human user can easily interact with the system**.
- Usability in general affects performance and efficiency given that more resources are required

Verifiability

- A software system is verifiable if its properties can be verified easily
- Consider for instance a system that provides information concerning performance qualities (such as time required to process the request)
- **Testability** related quality and assess how much it is easy to test the system
  - For example a class could **add specific methods for testing purpose**

Maintainability

- Refers to modifications that become necessary after the system has been released.
- It includes in general three different type of modifications:
  - **Corrective** (removal of bugs)
  - **Adaptive** (to adapt functionalities to new context, as a new Operating System)
  - **Perfective** (requires the introduction of new functionalities or improve performance)

Repairability

- Refers to the time necessary to repair the system after a failure
  - It include the time to **diagnose the problem, localise the fault, and make the necessary modifications**

Evolvability

- System evolution refers to the ability of **introducing new functionality** and making them to correctly interact with the already existing features.

- In general evolvability reduces after a number of successful evolution

- Systems with a **Plug-In** architecture are good example of evolvable systems

Reusability

- Refers to the ability of reusing a system or sub-system in a different context, possibly after some minor change

Portability

- A **Portable system can be run on different platforms**. This quality refers to the complexity and modification necessary to make a system runnable on a different environment
- From a OS point of view software developed in Java are really portable!

Understability

- Answer to the question:

  is it easy to understand the system implementation?

- In complex organisation really important quality that affect for instance evolvability or maintainability
- Clearly related also to the documentation provided with the system

Interoperability

- Refers to the ability of a system to **coexist and cooperate with other systems**
- Really relevant factor getting more and more importance given the increasing request for integration
- **XML** tool that drastically improved interoperability among systems exchanging data

Productivity

- Related to the **process** and not to the product
- Refer to the efficiency of the process resulting in faster delivery of the product

Timeliness

- Relates to the ability of the process of letting you to **deliver the product on time**.
- Refers then to the techniques available for planning and assessing the status of development

- Often software projects are late with respect to the schedule!!

Visibility

- Relates to the process and judge **how much clear the status of the development is documented.**
- Clearly relevant for manager working on the development of complex systems
- At the same time quite costly property

Other “-ilities”

- The listed “-ilities” are general and common to all application domain
- Different domains can have more **specific quality attributes**
  - In a database system it is certainly relevant the number of transaction that a system can carry on in a quantity of tile
- Moreover quality attributes are not a finite set. New “-ilities” given the emergence of new technical possibilities and paradigms
- New entries:
  - Mobility
  - Adaptability
  - ...

Critical Systems

- Critical systems are technical or socio-technical systems that people or business depends on.
- Three main types of critical systems
  - **Safety critical**
  - **Mission critical**
  - **Business critical**
- Dependability term used to identify different qualities that are generally **relevant for critical systems** (Reliability, Availability, Security and Safety)
- How we can increase dependability:
  - Testing, Formal Verification, Monitoring, Fault tolerance...

Safety

- Safety is a property of a system that reflects the system's **ability to operate, normally or abnormally, without danger of causing human injury or death** and without damage to the system's environment
- It is increasingly important to consider software safety as more and more devices incorporate software-based control systems
- Related to robustness – injuring people is certainly a not reasonable behaviour

Safety and Reliability

- Safety and reliability are related but distinct
  - In general, reliability and availability are necessary but not sufficient conditions for system safety
- Reliability is **concerned with conformance to a given specification** and delivery of service
- Safety is concerned with ensuring system **cannot cause damage irrespective of whether or not it conforms to its specification**

Unsafe Reliable Systems

- Specification errors
  - If the system specification is incorrect then the system can behave as specified but still cause an accident
- Hardware failures generating spurious inputs
  - Hard to anticipate in the specification
- Context-sensitive commands i.e. issuing the right command at the wrong time
  - Often the result of operator error

- Complex systems are not 100% safe. Society decide if it is worthy or not to risk.

Security

- The security of a system is a system property that reflects the system's ability **to protect itself from accidental or deliberate external attack**
- Security is becoming increasingly important as **systems are networked** so that external access to the system through the Internet is possible
- Security is an essential pre-requisite for availability, reliability and safety

Security Terminology

Term	Definition
Exposure	Possible loss or harm in a computing system. This can be loss or damage to data or can be a loss of time and effort if recovery is necessary after a security breach.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.
Attack	An exploitation of a system vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
Threats	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Control	A protective measure that reduces a system vulnerability. Encryption would be an example of a control that reduced a vulnerability of a weak access control system.

Damage from insecurity

- Denial of service (DoS attack)
  - The system is forced into a state where normal services are unavailable or where **service provision is significantly degraded**
- Corruption of programs or data
  - The programs or data in the system may be **modified in an unauthorised way**
- Disclosure of confidential information
  - **Information** that is managed by the system **may be exposed to people who are not authorised** to read or use that information

Key Points

- SE key characteristics
- SE challenges
- Product vs. Process
- Qualities in Software Artifacts and software process
- Critical System and dependability