SOFTWARE DESIGN
SOFTWARE QUALITY ASSESSMENT

Assoc.Prof. Feza BUZLUCA
Istanbul Technical University
Computer Engineering Department

http://akademi.itu.edu.tr/en/buzluca/
http://www.buzluca.info

About me:

Undergraduate courses:

- Digital Circuits
- Computer Architecture

Hardware-based

- Object-Oriented Programming
- Object-Oriented Modeling and Design (*elective*)

Software-based

Graduate course (PhD):

- Software Design Quality Assessment

Research Area:

- Object-Oriented Software Modeling and Design
- Software quality measurement and evaluation (assessment)
- Computer Networks

Why is software important?

Today, almost each electronic device includes a computer system (microprocessor, memory) controlled by software.

Software is playing a very important role in our daily lives.

- Phones, cameras, watches, TVs, washing machines

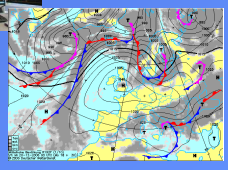


- Cars, railway systems, air traffic control systems



- Banking, e-commerce systems


- Weather forecast, predicting the path of a hurricane



- Student administration software
- Social media, manufacturing systems, robots, etc.

Objective (software development):

The ability to deliver a software system

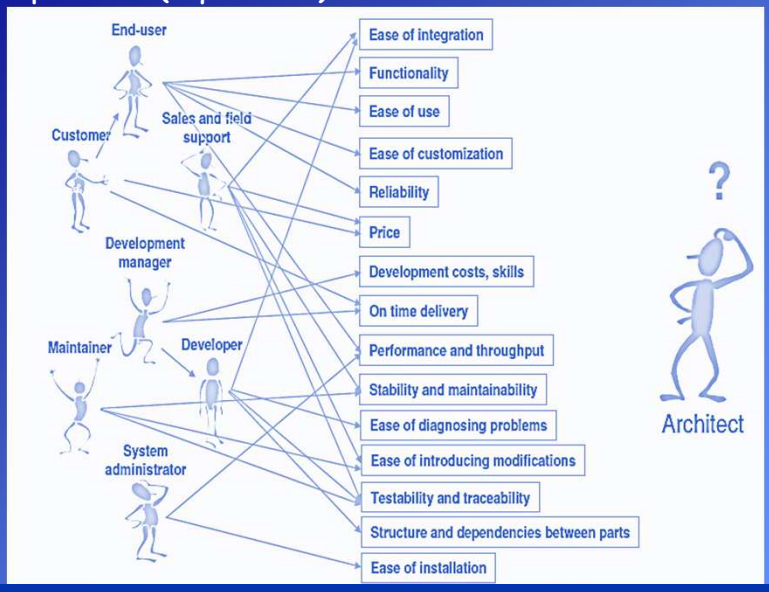
- that meets **quality** needs of different stakeholders (user, developer, customer …)
  - functionality
  - performance (speed, accuracy, etc.)
  - efficiency (processor, memory, network etc.)
  - reliability (error free)
  - security
  - maintainability
  - …
- on time,
- within budget.

Just writing a code that runs somehow is not sufficient!

In case of failure?

- People may lose their lives (life-critical systems).
- Software company may lose money (time).
- Other stakeholders (direct/indirect users) may lose money (time).

Expectations (requirements) and the Software Architect



Source: D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design," *ACM Computing Surveys*, vol. 43, pp. 1-28, Oct. 2011.

What are the quality characteristics of a software system?

ISO (the International Organization for Standardization) and **IEC** (the International Electrotechnical Commission) prepared standards for quality models.

ISO/IEC 25010 : 2011
Systems and software engineering —
Systems and software Quality Requirements and Evaluation (SQuaRE)-
System and software quality models

This standard includes 2 quality models.

A) Quality in use model:

This is the external quality of the system; impact on stakeholders in specific contexts of use.

Quality in use model

Effectiveness	Efficiency	Satisfaction	Freedom from risk	Context coverage
Effectiveness	Efficiency	Usefulness Trust Pleasure Comfort	Economic risk mitigation Health and safety risk mit. Environmental risk mitigation	Context completeness Flexibility

What are the quality characteristics of a software system?

B) Product Quality:

These characteristics relate to the software development team.

Product Quality Model

1. Functional suitability.
   - Functional completeness
   - Functional correctness
   - Functional appropriateness
2. Performance efficiency
   - Time behavior
   - Resource utilization
   - Capacity
3. Compatibility
   - Co-existence
   - Interoperability
4. Usability
   - Learnability
   - Operability
   - User error protection
   - User interface aesthetics
   - …
5. Reliability
   - Availability
   - Recoverability
   - …
6. Security
   - …
7. Maintainability
   - Modularity
   - Reusability
   - Analysability
   - Modifiability
   - Testability
8. Portability
   - Adaptability
   - Installability
   - Replaceability

The software crisis

Project Success Ratio:

	2015	2020
Successful	29%	31%
Challenged	52%	50%
Failed	19%	19%

https://standishgroup.com/

According to the Standish Group in 2020:

- Only 31% of all projects succeeding by delivered on time, on budget, with required features and functions (with a satisfactory result).
- 50% of software projects were late, over budget, and/or with less than the required features and functions.
- 19% of projects were failed and were cancelled prior to completion, or delivered and never used.

Software maintenance cost and Technical debt

Software maintenance costs are around between 50% and 90% of total software life-cycle cost.

Maintenance: Changes (requirement changes or bug fixes) that have to be made to software system after it has been delivered to the customer.

Technical debt (design debt, or code debt): Concept in programming that reflects the cost of extra rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.

Key Terms:

- Analyzability: Easy to understand
- Flexibility: Easy to change (modify), extend, correct, and adapt
- Testability: Easy to find faults
- Reusability: Easy to use in new modules or new projects

Why is software development difficult?

"Programming is fun, but developing quality software is hard." (*Philippe Kruchten*)

We focus on the challenges of developing "industrial-strength" software.

They have a very rich set of behaviors, include a lot of components, which cooperate with each other to fulfill some functionalities.

Complexity:

- This type of software systems are developed to solve problems in **complex** real-world systems.

  For example; banking systems, air or railway traffic control systems, a cellular phone switching system.
- Software inherits **complexity** of the problem domain.
- Today software products are often more complex than other engineering artifacts such as buildings, bridges or vehicles.

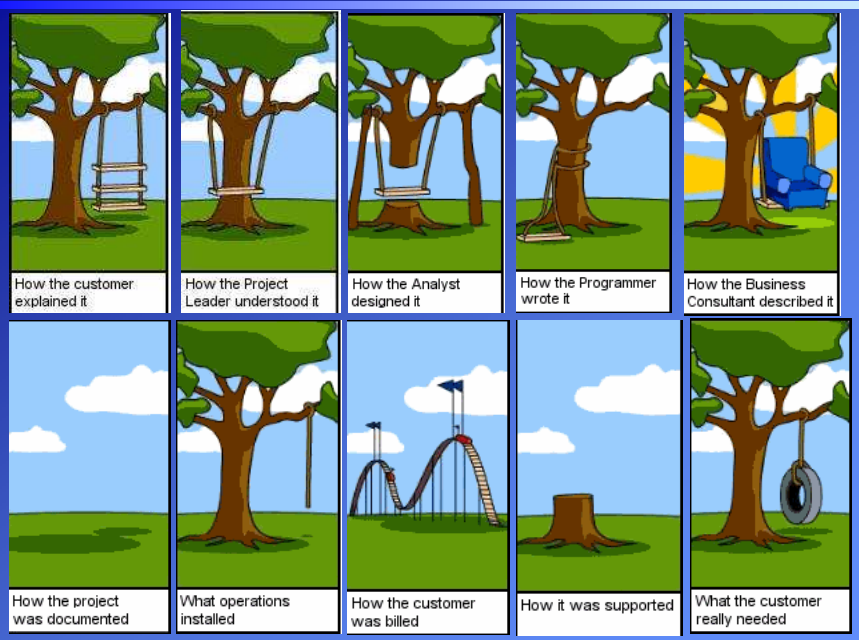
Why is software development difficult?

Many Components:

- Large software systems include many components, and they are developed by teams including a lot of members.
- **Communication** (interaction) and **cohesion** (harmony) between components play an important role.

Changes:

- Software systems tend to have a long life span. **Requirements change**.
- They must be **flexible to be adapted** to new needs.
- They must be **reusable**.



Why is the job of a software engineer difficult?

If you are a civil engineer building bridges then all you need to know is about bridges.

Unlike this, if you are developing software you need to know

a) about ***software domain*** (because that is what you are building)

and

b) you need to know about the ***problem domain*** (because that is what you are building a solution for).

"Progress is possible only if we train ourselves to think about programs without thinking of them as pieces of executable code."
Edsger W. Djikstra (1930-2002)

We cannot handle software systems as just long texts.
We must consider software systems as complex machines that consist of many components and layers.
Sometimes we need to change, replace, fix, or reuse these components.

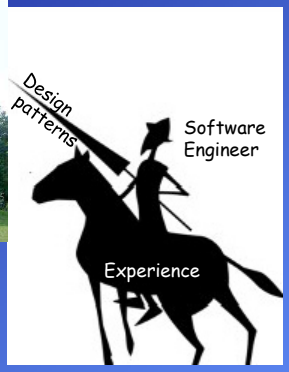
```
class ProductSpecification
{
  private:
     ItemID id;
     Money price;
     string specification;
  public:
ProductSpecification( const ItemID &id, const Money &price, const
string &spec )   {
       this->id = id;
       this->price = price;
       specification = spec;
                }
   const ItemID & getItemId() { return id; }
   const Money & getPrice() { return price; }
   const string & getSpecification() { return specification; }
};

class Sale
{
.....
.....
```

From Instagram @whatchandlove

The Goal

(1)



(2)

Besides meeting functional requirements of the stakeholders,

our ***objective*** is to learn how to deal with complexity,

handle changes,

build ***extensible***, ***flexible***, ***reusable***, ***error-free*** software systems,

and as a consequence **reduce the** (maintenance) **cost**.

For this reason, we should learn **object-oriented design principles** and **software design patterns**.

(1) From: http://www.photoeverywhere.co.uk
(2) From: http://www.cafepress.co.uk/

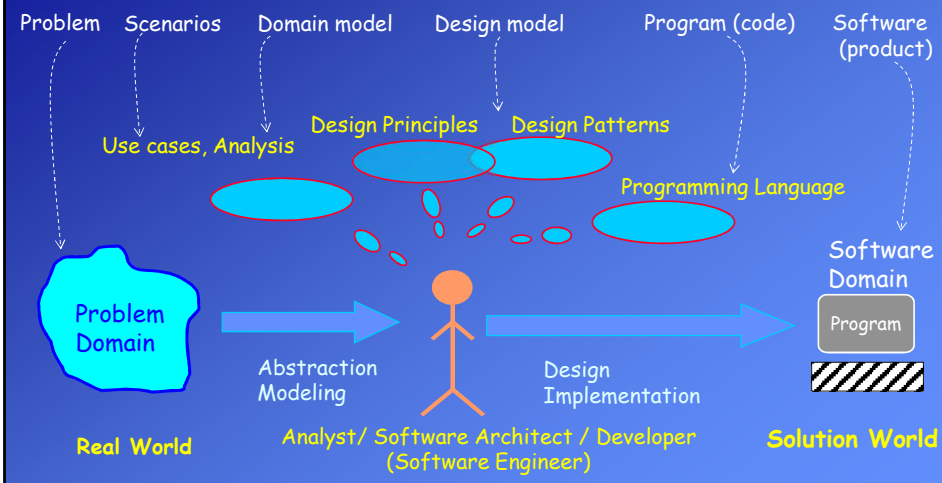
Our Tools:

- Software development is both an art and an engineering.
- There isn't any magic formula or any silver bullet (~~un~~fortunately).
  Intuition and **experiences** play important roles.
- Bjarne Stroustrup: "There are no 'cookbook' methods that can replace intelligence, experience, and **good taste in design** and programming."

Some helpful tools:

- Knowledge of Object-Oriented Programming (OOP course)
- Software development process: (*SwEng. course*)
  The Unified Process (UP): Iterative and evolutionary development
- Use case methodology (*SwEng. course*)
- **Object-oriented design principles** (BLG 468E)
- **Software design patterns** (BLG 468E)
- The Unified Modeling Language (UML) (OOP and BLG 468E)
- Software testing (BLG 475E)
- Software quality measurement and assessment (*PhD Course*)

The World of a Software Engineer



What should we learn/know?

"Progress is possible only if we train ourselves to think about programs without thinking of them as pieces of executable code."

Edsger W. Djikstra (1930-2002)

Learning of programming languages, platforms and popular tools is important and necessary but not sufficient.

They change, some of them disappear, some new products are created.

Focus on general concepts and do not get lost in language-technical details.

What should we learn/know?

- Efficient programming/coding (not just coding or just programming languages)
  (Efficient algorithms, efficient data structures)
  - Time efficiency
  - Space efficiency
  - Energy efficiency
- High-quality design (Design principles (OOP) and design patterns)
  - Extensible, modifiable
  - Reusable
  - Maintainable
- Hardware knowledge
  - The computer is not just a black box that executes programs by magic.
  - You need to understand computer architecture to develop programs that can achieve high performance.
- Try and gain experience
  - Understanding of development of high-quality software systems comes with time and practice.

A possible road in your professional life

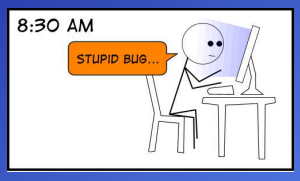
If you will work in the world of software development

Writes the code.

Designs the architecture.
Coaches the team.
Decides.

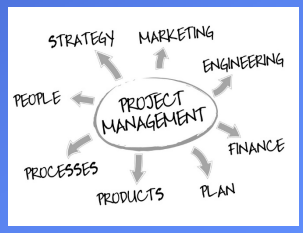
Management duties.
Not only about software

Programmer/ Developer → **Software Architect** → Project Leader/Manager



Source:
http://www.smashingapps.com/

Source:
http://www.planetgeek.ch



Source:
http://www.businessadministrationinformation.com/

An Option:

Start your own software company.

- Investment costs are not too high.
- They are inducements (KOSGEB, TÜBİTAK, İTÜ Arı Çekirdek)
- To build your own business you need experience.
- It is a good idea to work in the industry and gather experience and then to start your own company .

The Software Engineering Code of Ethics and Professional Practice

https://ethics.acm.org/code-of-ethics/software-engineering-code/

"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.

In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice."

It contains 8 Principles.

Principles contain Clauses.

The Software Engineering Code of Ethics and Professional Practice

Examples of Clauses:

1.01. Accept full responsibility for their own work.

1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.

2.02. Not knowingly use software that is obtained or retained either illegally or unethically.

2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.

3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.

The Software Engineering Code of Ethics and Professional Practice

Examples of Clauses:

6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.

7.03. Credit fully the work of others and refrain from taking undue credit.

8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.