Introduction to Software Engineering

Chapter 1

PowerPoint

Instructor

Lecturer: Dr. Samer Zein

Room: Masri 417

E-mail: szain@birzeit.edu

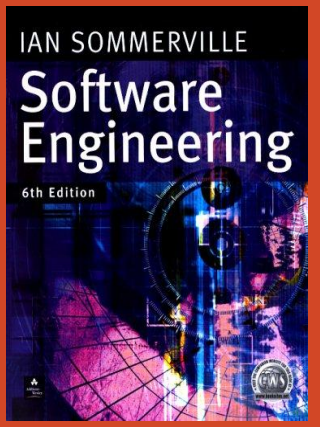
Linkedin: https://www.linkedin.com/in/samer-m-zain-phd/

General Class Regulations

- Eating is prohibited
- Make sure you arrive on time.
- Ask for permission before leaving the room.
- Talking is not allowed during lecture.
- Feel free to ask questions all the time!

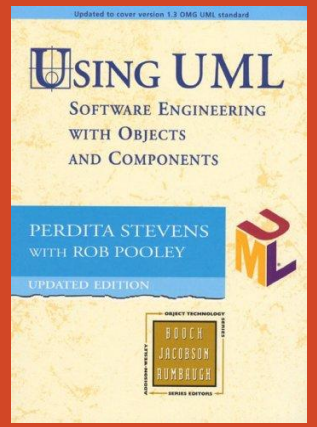
Recommended Text Books

- Bruegge and Dutoit, ***Object-Oriented Software Engineering Using UML, Patterns, and Java***, Prentice Hall **3rd** Edition



- Sommerville I. (2010)
  Software Engineering 9th Edition, Addison-Wesley, Harlow, Essex,UK (6th ,7th,or 8th would suffice)

- Stevens P. with Pooley, R. (2005)
  Using UML: Software Engineering with Objects and Components,
  2nd Ed., Addison-Wesley, Harlow, Essex, UK



- Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich. (2005)
  Modern System Analysis and Design 4th - 6th Edition,
  Prentice Hall.

- *Roger Pressman (2014),* ***Software Engineering: A Practitioner's Approach*** *6-8th Edition, McGraw-Hill.*

Chapter Objectives

- What is Software Engineering and why we need it?
- Why Software Development is complex?
- How Software projects are different than traditional engineering projects?
- What is professional and large scale software projects?
- What are different types of software applications?
- What is meant by software development process and why it is important?

Motivation

- We can't run the modern world without software.
  - National infrastructure
  - Electric products.
  - Industrial manufacturing
  - Goods distribution.
  - Education, Banks
  - Organizations
  - Mobile apps, the new player!

- However, **building and maintaining software is hard and getting harder**

Complexity:

- **Software development is Complex**!
- Abstract and intangible.
- **Important to distinguish "small" systems** (*one developer, one user, experimental use only*) **from "Complex" systems** (*multiple developers, multiple users, products*)
- **Experience with "small" systems is misleading**
  - *One person techniques do not scale up*
- **Analogy with bridge building:**
  - Over a stream = easy, one person job
  - Over a River … ? (*the techniques do not scale)*

Different Approaches!

- It is pointless to look for universal notations, methods, or techniques for software engineering because different types of software require different approaches.
- All of these applications need software engineering; they do not all need the same software engineering techniques.

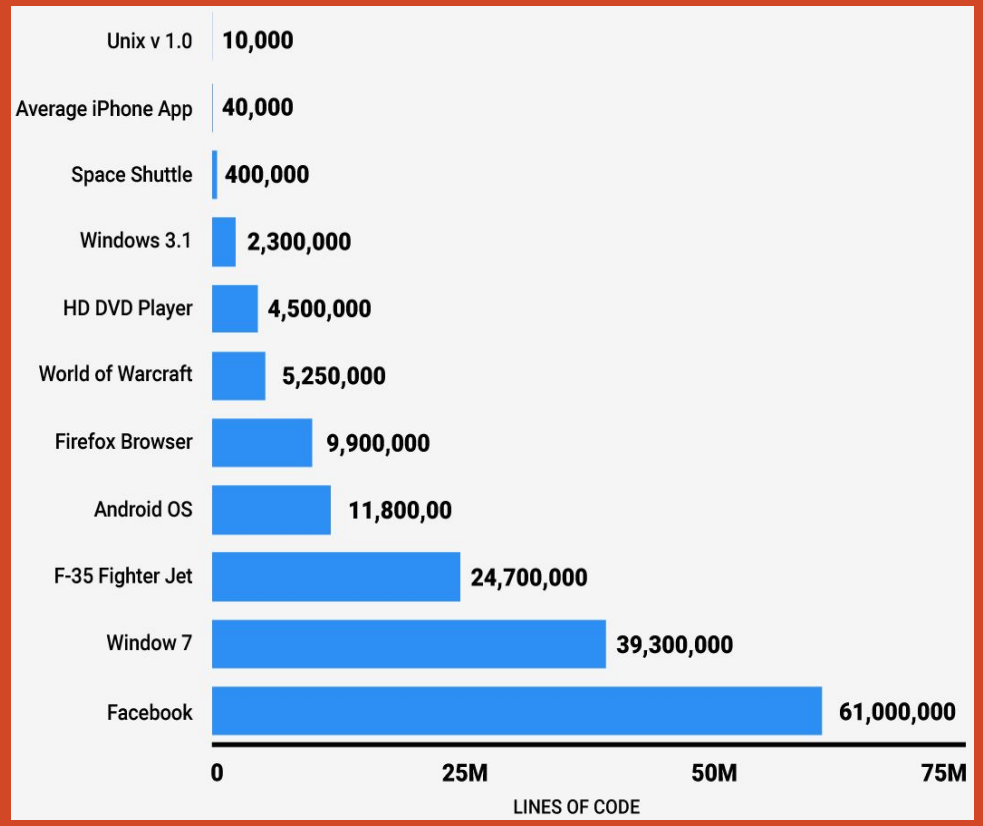
Industry and Software Engineering

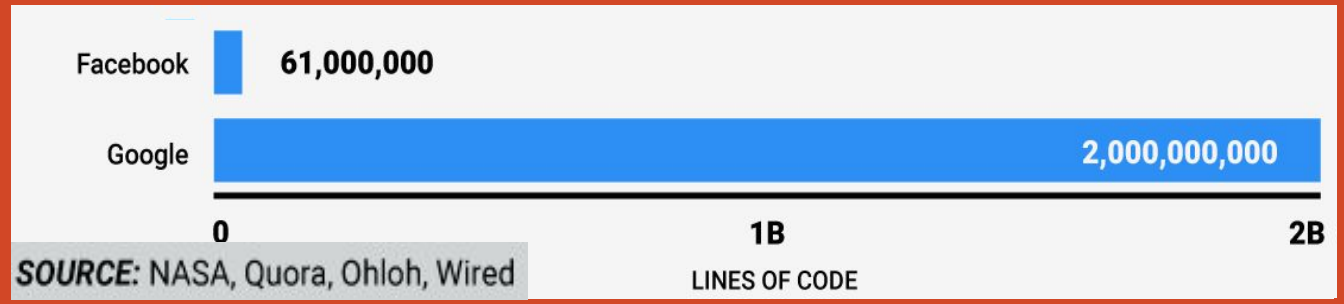
- **Increasing demands:** Systems have to be built and delivered more quickly; larger, even more complex systems are required

- **Low expectations:** It is relatively easy to write computer programs without using software engineering methods and techniques.
  - Many companies have drifted into software development as their products and services have evolved.
  - They do not use software engineering methods in their everyday work.
  - Consequently, their software is often more expensive and less reliable than it should be.

Why Software Engineering ?

Complexity increases as Size **increases!**





Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation, and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering. Software engineering is part of this more general process.

What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the Web made to software engineering?	The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

What is the difference between software engineering and computer science?

Computer Science	Software Engineering
is concerned with	
theory fundamentals Algorithms, date structures, complexity theory, numerical methods	the practicalities of developing and delivering useful software SE deals with practical problems in complex software products

Computer science theories are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering

Different than regular engineers?

- Yes, normal engineers for buildings and bridges for example, have faced similar examples, risks, solutions.
- But we are not so lucky
- So what is the problem, really?
  - **Complexity**
  - **Change**

SE history

- SE introduced first in 1968 – conference about "software crisis" when the introduction of third generation computer hardware led to more complex software systems than before
- Early approaches based on informal methodologies led to
  - Delays in software delivery
  - Higher costs than initially estimated
  - Unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software.

Professional Software Development

- The vast majority of software development is a professional activity:
  - Special team of developers.
  - For specific business purpose.
  - Formal and systematic (Process).
  - Software will be maintained and changed throughout its life
- Software Engineering supports professional SW development, not amateur development.
- Two types of Software:
  - Generic products: MS Office
  - Customized Products: Ramallah Library System, Jawwal سابي

Attributes of a Good Software

- Good software:
  - deliver the required functionality
  - Acceptable performance to the user
  - and should be maintainable,
  - dependable,
  - and usable
- Therefore, **a banking system** must be **secure**,
- An interactive **game** must be **responsive**,
- a **medical software** system must be **reliable**, and so on

Attributes of a Good Software 2

Product characteristics	Description
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 inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

Software myths (For Discussion)

- **Management myths**
  - *Standards and procedures for building software exist*
  - *Add more programmers if behind schedule*
- **Customer myths**
  - *A general description of objectives enough to start coding*
  - *Requirements may change as software is flexible*
- **Practitioner myths**
  - *Task accomplished when the program works*
  - *"Working program" the only project deliverable*

Software failures

- **Therac-25 (1985-1987)**: six people overexposed during treatments for cancer
- **Taurus (1993)**: the planned automatic transaction settlement system for London Stock Exchange cancelled after five years of development
- **Ariane 5 (1996)**: rocket exploded soon after its launch due error conversion (16 floating point into 16-bit integer)
- **The Mars Climate Orbiter** assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

More Software failures

- Passport System delays cause backlog (1999, UK)
- Ferry Company left thousands of lorries stranded for 12 hours (back up also failed, 1999, UK)
- Inland Revenue (IR) 'losing tax records' (2000, UK)

  => IR spokesman said 'All major IT initiatives have some kind of teething problems ....'

  => Guardian (20 July 2000) 'At the centre of the crisis are two computer systems .... Files appear to have gone missing somewhere between the two'

- General Motors Ford Cars (2016, USA + Worldwide): A "software bug" that may cause human safety, 4.5M cars recalled.

Even More Software failures

In 1995 annual US spending on software projects reached 250 billion dollars

This involved some 175,000 projects

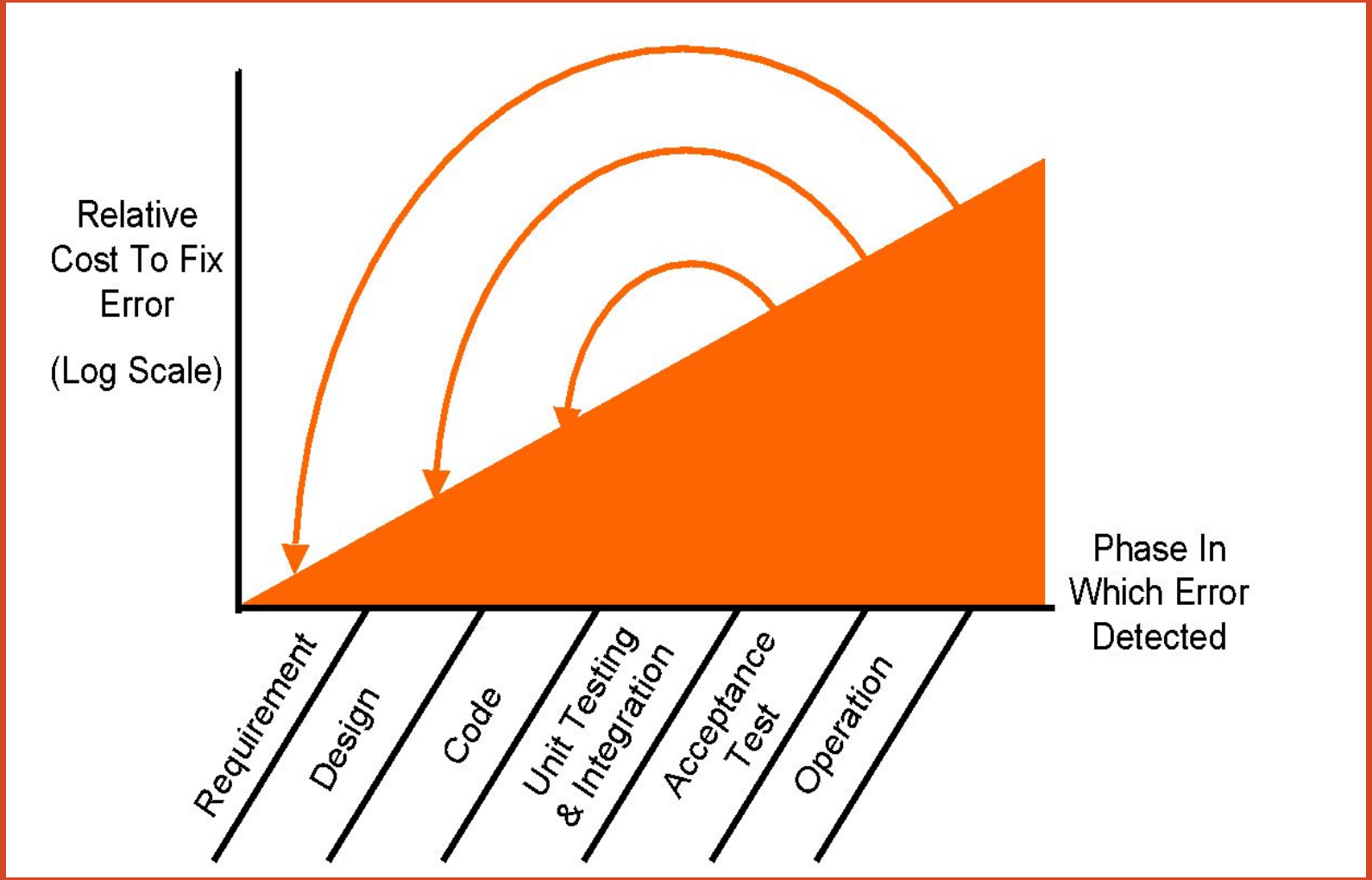
Of this spend:

- Overspend cost 59 billion dollars
- Cancelled projects cost 81 billion dollars

Causes of Software Failures?

Causes:

Cost of delayed error detection



Why software projects fail?

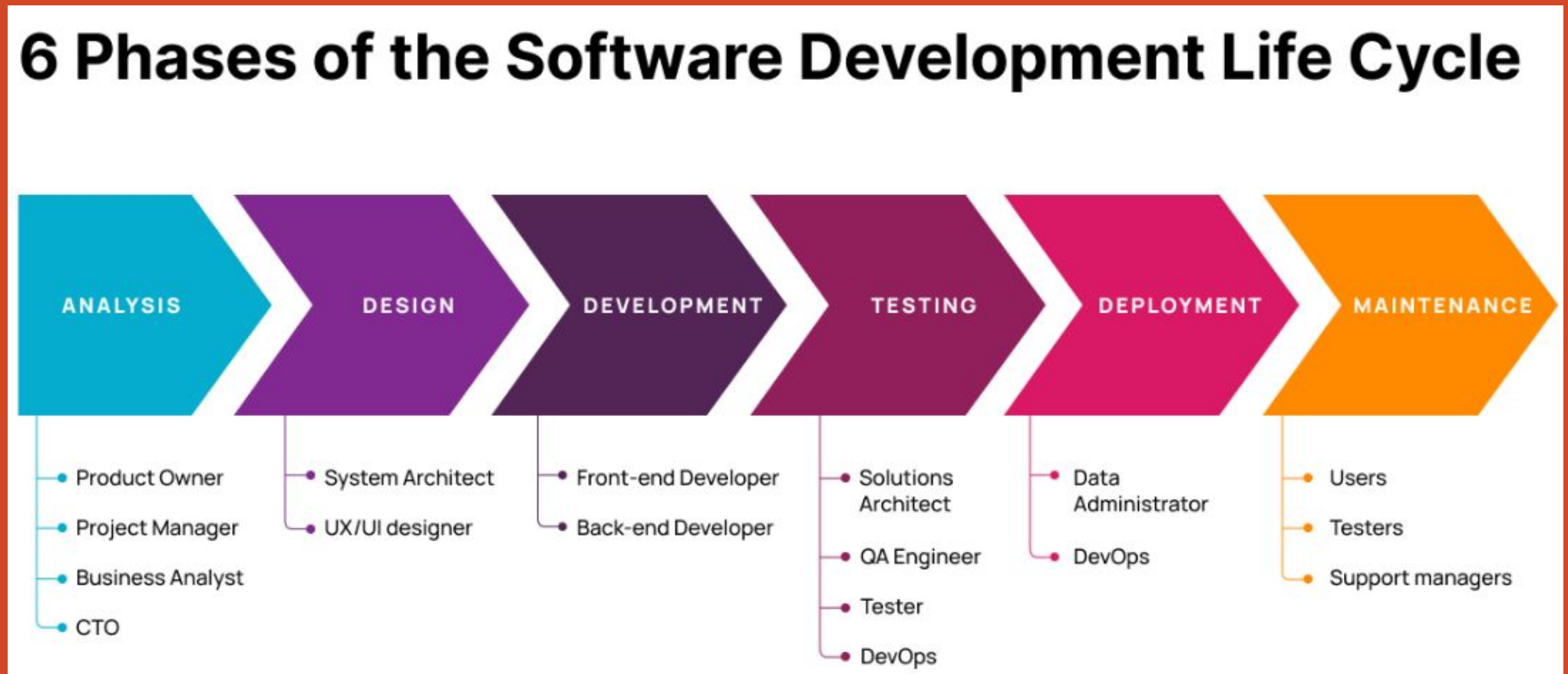
- Inaccurate understanding of customer needs
- Inability to deal with changing requirements
- Modules that do not fit together
- Software that are hard to maintain/extend
- Poor Quality
- Testing….normally should cost 40%
- Unacceptable performance
- Technology change and team-member change over time in long period projects

So what is a successful software project?

- Good software should:
  - Deliver the required functionality
  - Efficient: does not waste voluble resources, response time
  - Usable
  - Dependable: reliable, secure, and safe.
  - Maintainable
  - **Within budget and time**

Software Development Process

- A systematic approach for software development.
- It is a sequence of activities that leads to the production of software.



Software Development Process..2

- **Different** types of systems need **different** development processes.
- E-commerce and MIS systems **Incremental**

- While real time (**critical**) system for an aircraft must be **completely specified** in details before design & implementation

Different Types of Software Different Types of Approaches

- Different types of software:
  - **Stand-Alone: Office products**
  - **Interactive transaction based: Ritaj**
  - **Embedded systems: car software systems**
  - **Entertainment: games**
  - **Data collection: Mars rovers**

Software Engineering fundamentals that Apply to all Types of Software

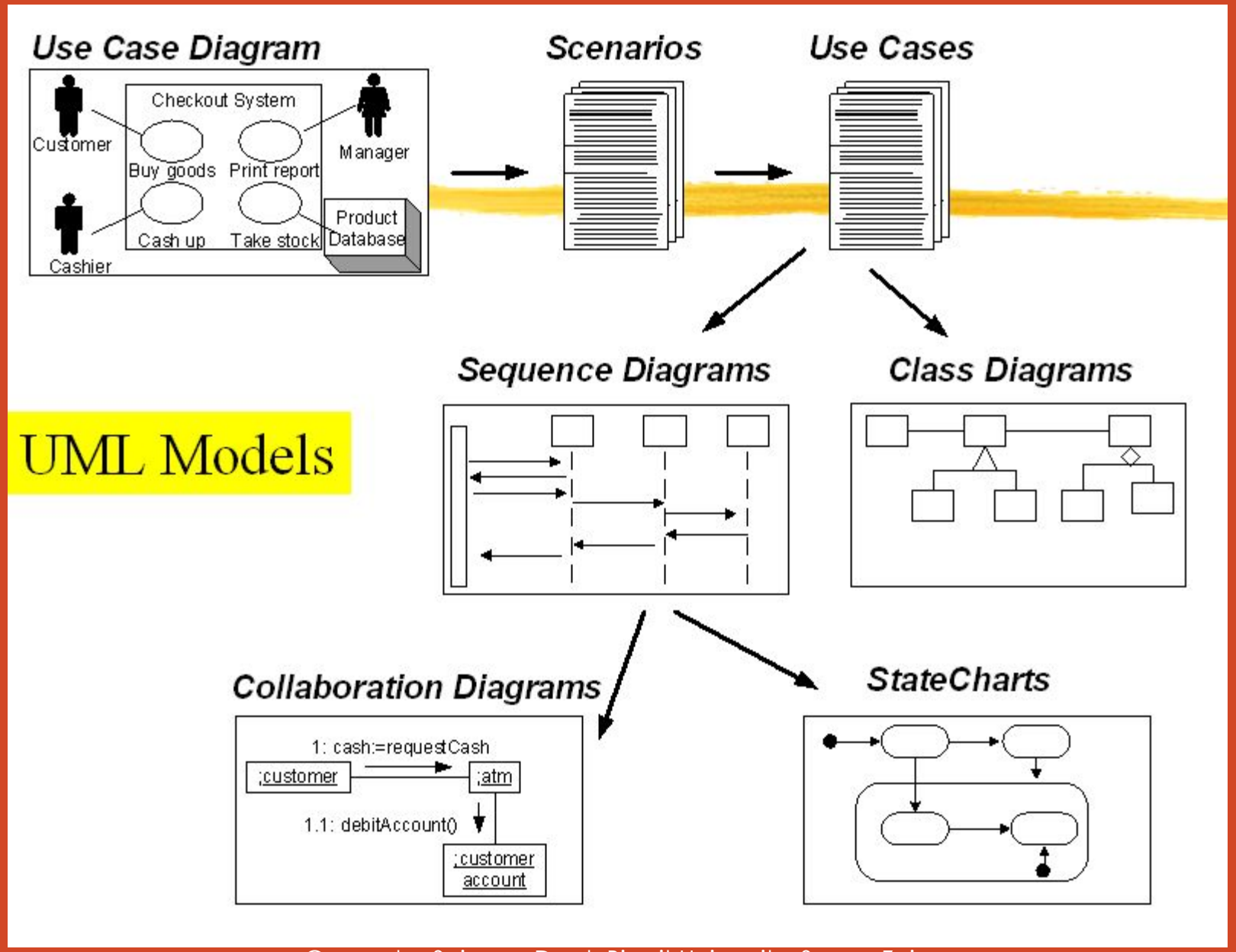
Clear Development Process

Performance

Clear Requirements

Dependability

Reusability



Requirements

- Requirements specify a set of features that the system must have.
- A **functional requirement** is a specification of a function that the system must support,

- **Nonfunctional requirement:** is a constraint on the operation of the system that is not related directly to a function of the system.

Example

1. The user must be able to purchase tickets
2. The user must be able to access traffic information
3. The system must be provided feedback in less than one second
4. The colors used in the interface should be consistent with the company LOGO
5. System should be easy to use since users could be of different ages

- **Other nonfunctional requirements may include:**

1. using specific hardware platform for the system,
2. security requirements,
3. how the system should deal with failures and faults,
4. and how to provide backward compatibility with an old system that the client is unwilling to retire.

Generic Technical Terms:

- **Notation**: a graphical or textual set of rules for representing a model (UML)
- **Method:** a repeatable technique that specifies the steps involved in solving a specific problem (Sorting Algorithm)
- **Methodology:** a collection of methods for solving a specific set of problems.

What is CASE ?
(Computer-Aided Software Engineering)

Software systems which are intended to provide automated support for software process activities, such as requirements analysis, system modelling, debugging and testing

Upper-CASE

Tools to support the early process requirements and design

Lower-CASE

Tools to support later activities such as programming, debugging and testing

What are the key challenges facing software engineering?

Software engineering in the 22st century faces three key challenges:

- **Legacy systems**
  - Old, valuable systems must be maintained and updated
- **Increasing Diversity and Heterogeneity**
  - Systems are distributed and include a mix of different hardware and software
- **Dependability and Delivery**
  - Having trustworthy software with faster delivery of software (time-to-market)