Western

UNIVERSITY · CANADA

Topic 1

Software and Software Engineering (Chapter 1)

Computer Science 2212b
Introduction to Software Engineering
Winter 2014

Jeff Shantz
jeff@csd.uwo

Agenda / Announcements

- Friday's class
  - Meet your team members

“The real exam – that is, the real measure of what you have learned – doesn’t happen, indeed can’t happen, in the classroom. Evidence of what you have learned really happens ‘out there’ in daily practice, and the exam typically lasts not for two hours at the end of a semester, but for three to four decades over the course of a lifetime.”

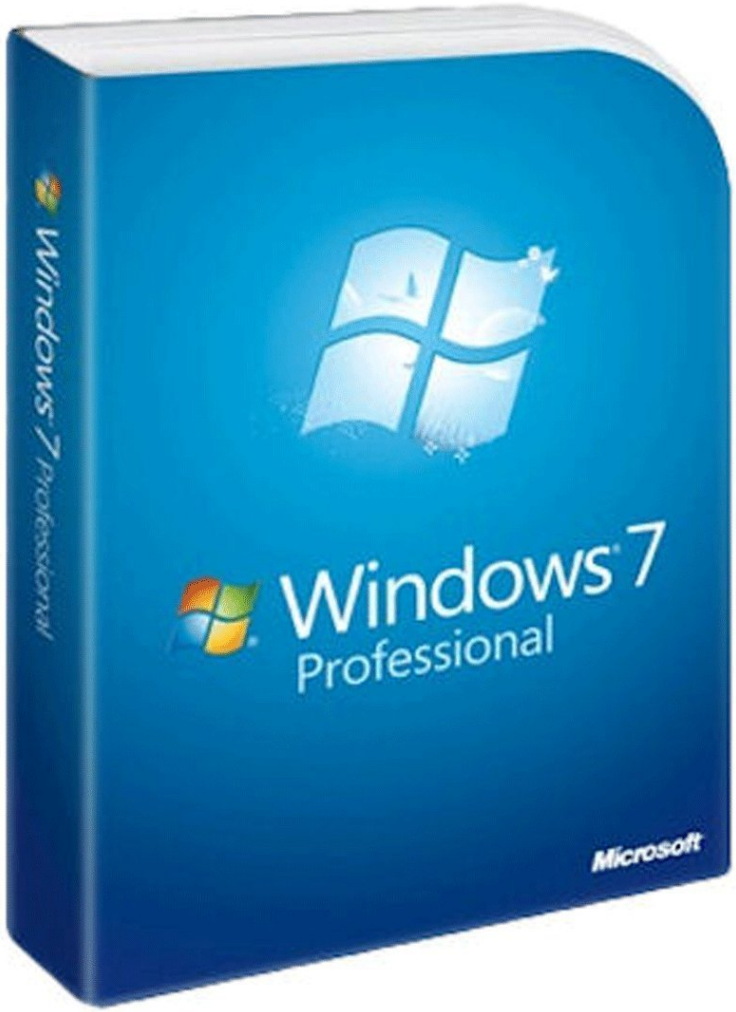
-- Reg Litz, Professor, Asper School of Business

Differences?

vs.

Differences?

vs.



1.1 – The Nature of Software

- **Software is intangible**
  - Hard to understand development effort
- **Software is easy to reproduce**
  - Cost is in its *development*
- **The industry is labour-intensive**
  - Hard to automate

The Nature of Software

- **Untrained people can hack something together**
  - Quality problems are hard to notice

- **Software is easy to modify**
  - People make changes without fully understanding it

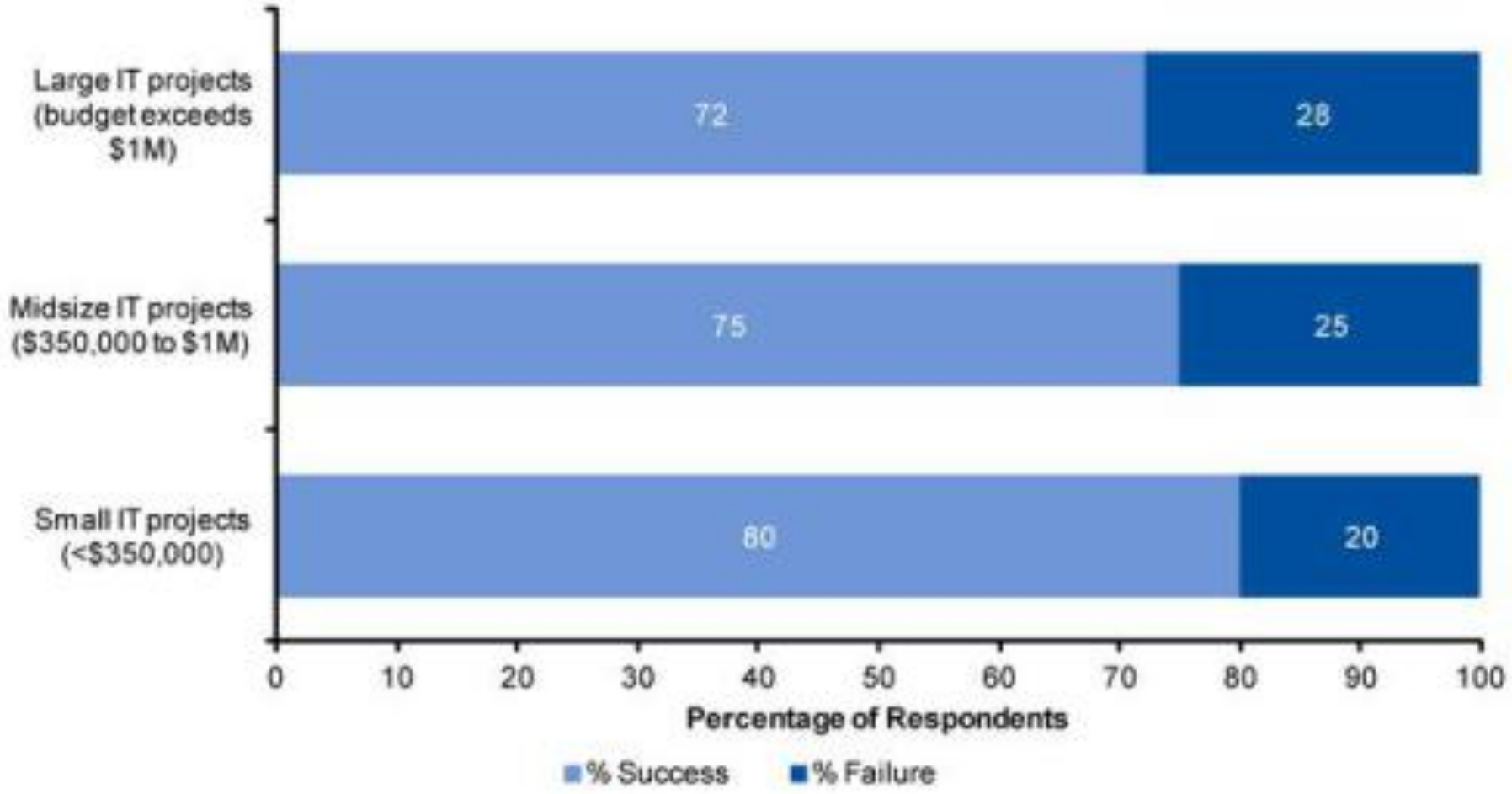
- **Software does not *wear out***
  - It deteriorates by having its design changed:
    - erroneously, or
    - in ways that were not anticipated, thus making it complex

The Nature of Software

- **Conclusions**
  - Much software has poor design and is getting worse
  - Demand for software is high and rising
  - We are in a perpetual *software crisis*
  - We have to learn to *engineer* software

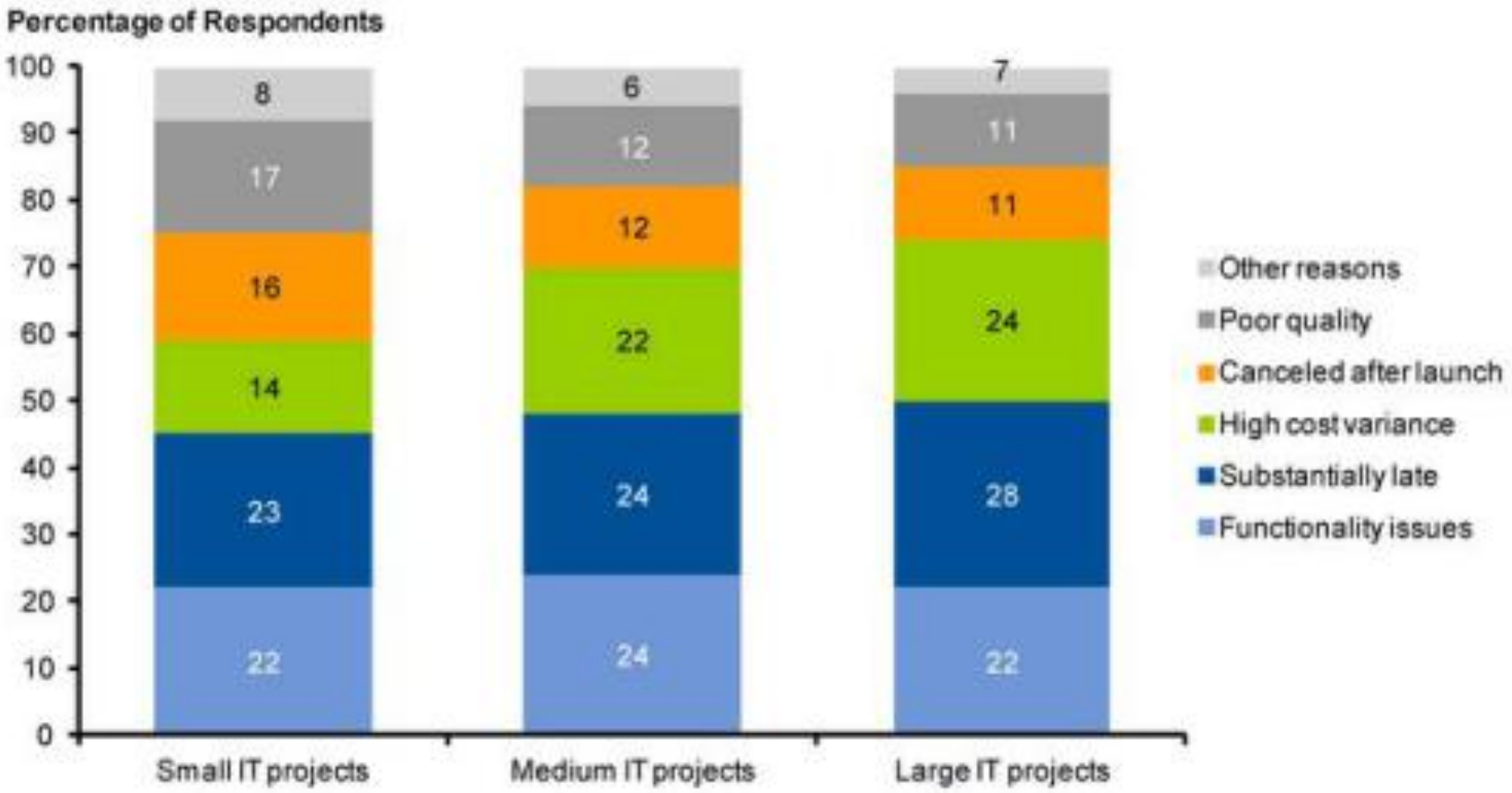
The Nature of Software

Distribution of Success and Failure Rates Across Project Sizes (Gartner, 2012)



The Nature of Software

Why Projects Fail (Gartner, 2012)



Types of Software

- **Custom**
  - For a specific customer
  - Usually developed in-house
- **Generic**
  - Sold on open market
  - Also known as
    - COTS (Commercial Off The Shelf)
    - Shrink-wrapped
- **Embedded**
  - Built into hardware
  - Hard to change

Types of Software

Custom vs. Generic vs. Embedded Software			
	Custom	**Generic**	**Embedded**
Copies in use	Low	Medium	High
Processing power devoted to running this type of software	Low	High	Medium
Annual development effort	High	Medium	Medium

Types of Software

- Real time software
  - e.g. control and monitoring systems
  - Must react immediately
  - Safety often a concern
  - Soft vs. hard real-time

- Data processing software
  - Used to run businesses
  - Accuracy and security of data are key

Some software has both aspects

Food for Thought

What do you think?

- Does everyone need to learn code?
  - Are there disadvantages to having everyone know how to code?
- Is all software development *engineering*?
  - If not, what is the difference to you?

1.2 – What is Software Engineering?

The process of solving customers' problems by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints

- Textbook definition

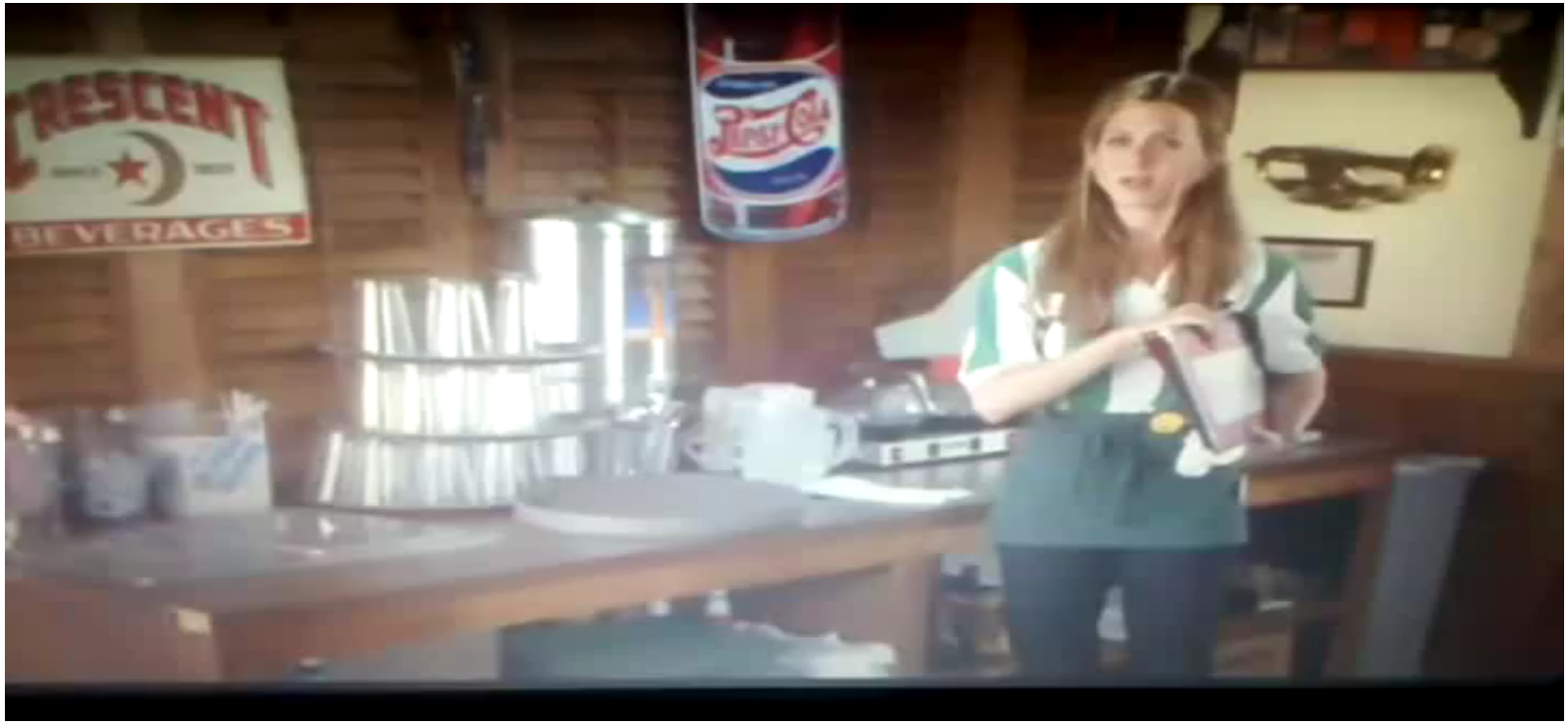
What is Software Engineering?

Solving customers' problems

- This is the goal of software engineering
- Sometimes the solution is to buy, not build
- Adding unnecessary features does not help solve the problem
- Software engineers must communicate effectively to identify and understand the problem

What is Software Engineering?

Solving customers' problems



What is Software Engineering?

Systematic development and evolution

- An engineering process involves applying well understood techniques in a organized and disciplined way
- Many well-accepted practices have been formally standardized
  - e.g. by the IEEE or ISO
- Most development work is *evolution*

What is Software Engineering?

Large, high quality software systems

- Software engineering techniques are needed because large systems cannot be completely understood by one person
- Teamwork and co-ordination are required
- Key challenge: Dividing up the work and ensuring that the parts of the system work properly together
- The end-product that is produced must be of sufficient quality

What is Software Engineering?

Cost, time, and other constraints

- Finite resources
- The benefit must outweigh the cost
- Others are competing to do the job cheaper and faster
- Inaccurate estimates of cost and time have caused many project failures

1.3 – Software Engineering and the Engineering Profession

- **"Software Engineering"**
  - Coined in 1968
  - People began to realize that the principles of engineering should be applied to software development

- Engineering is a licensed profession
  - In order to protect the public
  - Engineers design artifacts following well accepted practices which involve the application of science, mathematics and economics
  - Ethical practice is also a key tenet of the profession
  - Take personal responsibility for the work created

1.4 – Stakeholders in Software Engineering

1. **Users**
   - Those who use the software

2. **Customers**
   - Those who pay for the software

3. **Software developers**

4. **Development Managers**

All four roles can be fulfilled by the same person

1.5 – Software Quality

Usability

Users can learn it and fast and get their job done easily

Efficiency

It doesn't waste resources such as CPU time and memory

Reliability

It does what it is required to do without failing

Maintainability

It can be easily changed

Reusability

Parts can be used elsewhere; no need to reinvent the wheel

Software Quality

Of the 5 most important attributes of software quality – *Usability, Maintainability, Reliability, Reusability and Efficiency* – which ones do **you** think would be **most** and **least** important to each of the four stakeholders?

Stakeholder	Most Important	Least Important
User		
Customer		
Developer		
Development manager		

Software Quality

Customer:
solves problems at
an acceptable cost in
terms of money paid and
resources used

User:
easy to learn;
efficient to use;
helps get work done

Developer:
easy to design;
easy to maintain;
easy to reuse its parts

Development manager:
sells more and
pleases customers
while costing less
to develop and maintain

Software Quality

- **The various qualities often conflict**
  - Improving one comes at the expense of another
  - Have to make trade-offs

If we improve efficiency, what might suffer? _____

If we improve reliability, what might suffer? _____

Software Quality

For each of the following systems, which attributes of quality would be most and least important? Justify your answer.

- a web based banking system, enabling the user to do all aspects of banking on line

- an air traffic control system

- a program that will enable users to view digital images or movies stored in all known formats

- a system to manage the work schedule of nurses that respects all the constraints and regulations in force at a particular hospital

Internal Quality Criteria

- So far, we've talked about **external quality**
  - Observed by stakeholders; have direct impact on them

- Development teams must also consider **internal quality**
  - Characterize aspects of the design of the software
  - Have an effect on the external quality attributes
    - The amount of commenting of the code (SLOC)
    - The complexity of the code
    - The coupling of the code

Short-Term vs. Long-Term Quality

- **Short term**
  - Does the software meet the customer's immediate needs?
  - Is it sufficiently efficient for the volume of data we have today?

- **Long term**
  - Maintainability
  - Customer's future needs

1.6 – Software Engineering Projects

- **Evolutionary**
- **Greenfield**
- **Building on a framework or set of existing components**

Software Engineering Projects

- **Evolutionary (maintenance)**
  - Most projects of this type
  - May have to work on *legacy* systems
  - Types:
    - **Corrective projects**
      - fixing defects
    - **Adaptive projects**
      - changing the system in response to changes in operating system, laws, etc.
    - **Enhancement projects**
      - adding new features for users
    - **Reengineering or perfective projects**
      - changing the system internally so it is more maintainable

Software Engineering Projects

- **Greenfield**
  - New development
  - The minority of projects
  - Often the most enjoyable
  - Take a lot of work

What is the one area where you might be working on a greenfield project (hint: we have a minor in it)?

Software Engineering Projects

- **Projects that involve building on a framework / set of existing components**
  - The framework is an application that is missing some important details
    - e.g. specific rules of this organization
  - Such projects:
    - Involve plugging together *components* that are:
      - Already developed
      - Provide significant functionality
    - Benefit from reusing reliable software
    - Provide much of the same freedom to innovate found in green field development

1.7 – Activities Common to Software Projects

- **Requirements and specification**
  - Domain analysis
  - Defining the problem
  - Requirements gathering
    - Obtaining input from as many sources as possible
  - Requirements analysis
    - Organizing the information
  - Requirements specification
    - Writing detailed instructions about how the software should behave

1.7 – Activities Common to Software Projects

- **Design**
  - Deciding how the requirements should be implemented, using the available technology
  - Includes:
    - **Systems engineering**: Deciding what should be in hardware and what in software
    - **Software architecture**: Dividing the system into subsystems and deciding how the subsystems will interact
    - Detailed design of the internals of a subsystem
    - User interface design
    - Determining how data will be stored

1.7 – Activities Common to Software Projects

- **Modeling**
  - Creating representations of the domain or the software
    - Use case modeling
    - Structural modeling
    - Dynamic and behavioural modeling
- **Programming**
- **Quality assurance**
  - Reviews and inspections
  - Testing
- **Deployment**
- **Managing the process**

1.9 – Difficulties/Risks in Software Engineering

- Complexity and large numbers of details
- Uncertainty about technology
- Uncertainty about requirements
- Uncertainty about software engineering skills
- Constant change
- Deterioration of software design
- Political risks

Western

UNIVERSITY · CANADA