



SWE 205: Introduction to Software Engineering

## Lecture 9

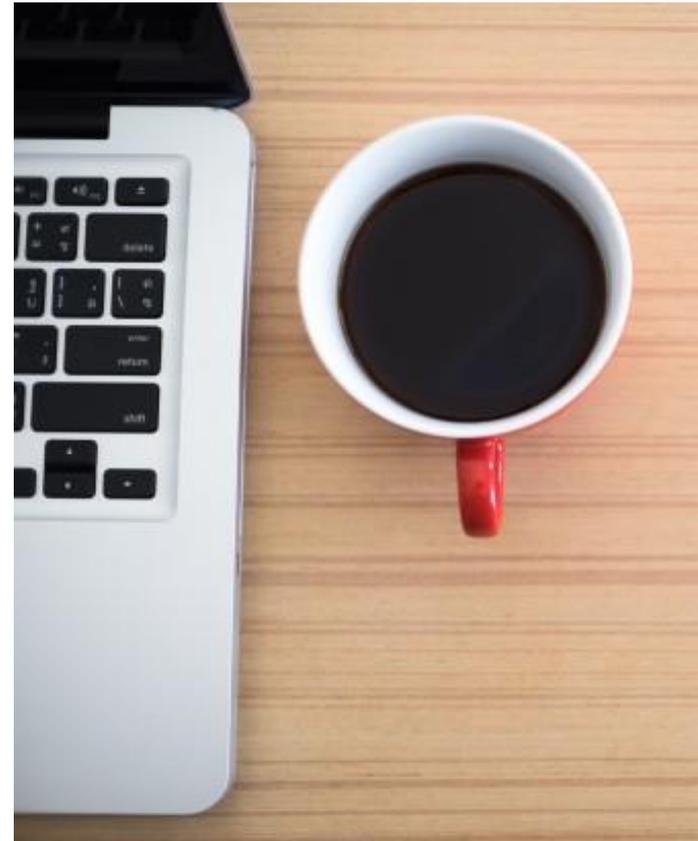
# Requirements Engineering IV

# Course Topics

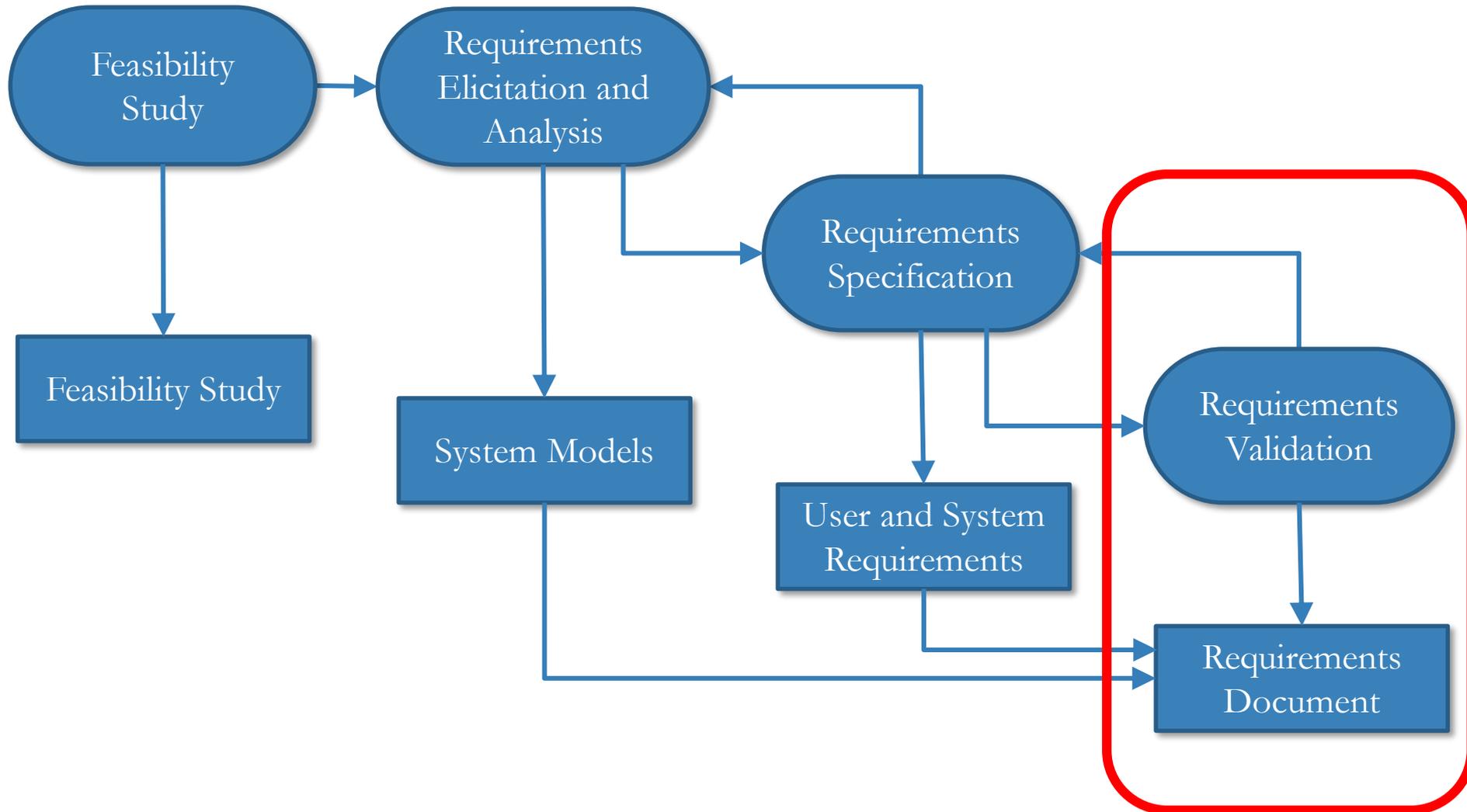
- ~~Introduction~~
- ~~Software Process Models~~
- Requirements Engineering
- Modeling
- Programming Languages
- Software Construction Techniques
- Testing
- Project Management
- Refactoring
- Ethical Issues

# Lecture Objectives

- ✓ Validation techniques
- ✓ Requirements management



# The requirements engineering process



# Requirements validation



- Concerned with demonstrating that requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

# Requirements checking

- **Validity**
  - Does the system provide the functions which best support the customer's needs?
- **Consistency**
  - Are there any requirements conflicts?
- **Completeness**
  - Are all functions required by the customer included?
- **Realism**
  - Can the requirements be implemented given available budget and technology
- **Verifiability**
  - Can the requirements be checked?

# Requirements validation techniques



## 1 - Requirements reviews

- Systematic manual analysis of the requirements.

## 2 - Prototyping

- Using an executable model of the system to check requirements.

## 3 - Test-case generation

- Requirements should be testable.
  - If a test is difficult or impossible to design, this usually means that the requirement will be difficult to implement and should be reconsidered.
- 

# Requirements Management

- It is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes.
  - You need to establish a formal process for making change proposals and linking these to system requirements.

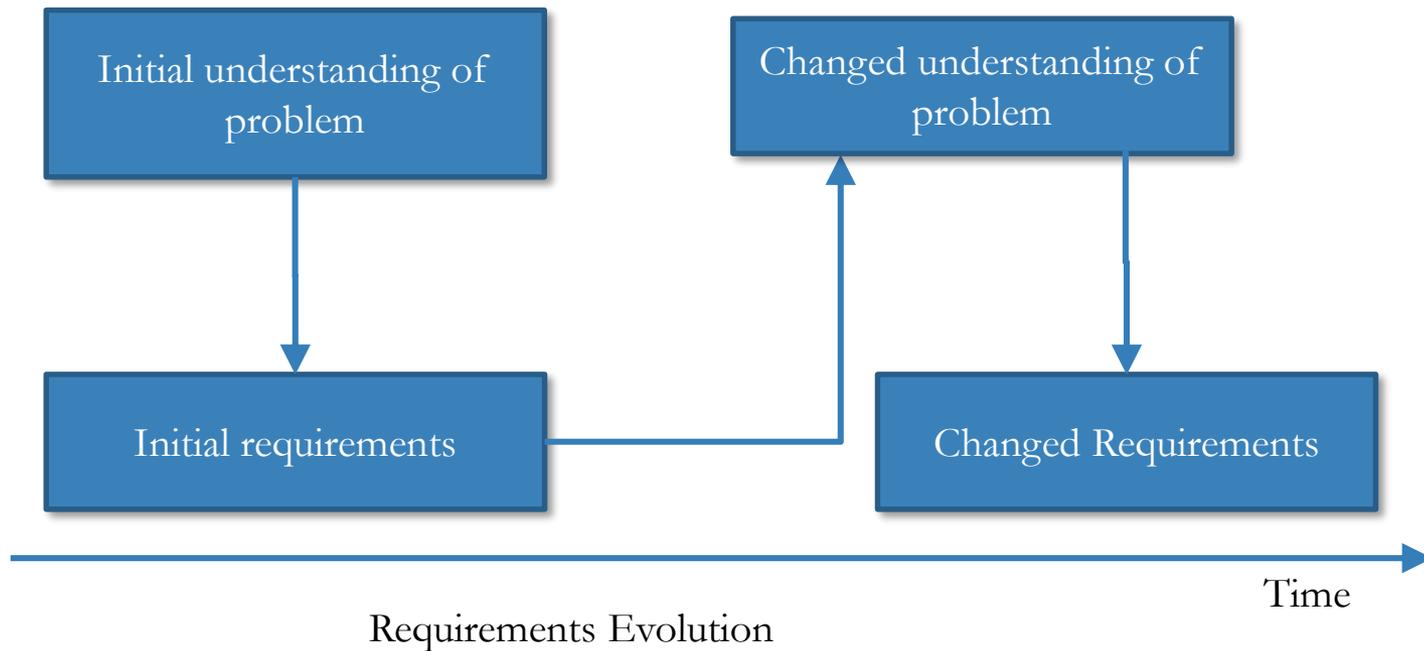


**Programming Wisdom** @CodeWisdom · Aug 26



"Walking on water and developing software from a specification are easy if both are frozen." - Edward V Berard

# Requirements evolution



# Changing requirements



- The business and technical environment of the system always change after installation.
  - New hardware may be introduced,
  - It may be necessary to interface the system with other systems,
  - Business priorities may change (with consequent changes in the system support required), and
  - New legislation and regulations may be introduced that the system must necessarily abide by.

# Changing requirements



- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints.
  - These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

# Changing requirements



- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them and, with experience,
  - It is often discovered that the balance of support given to different users has to be changed.

# Requirements management planning

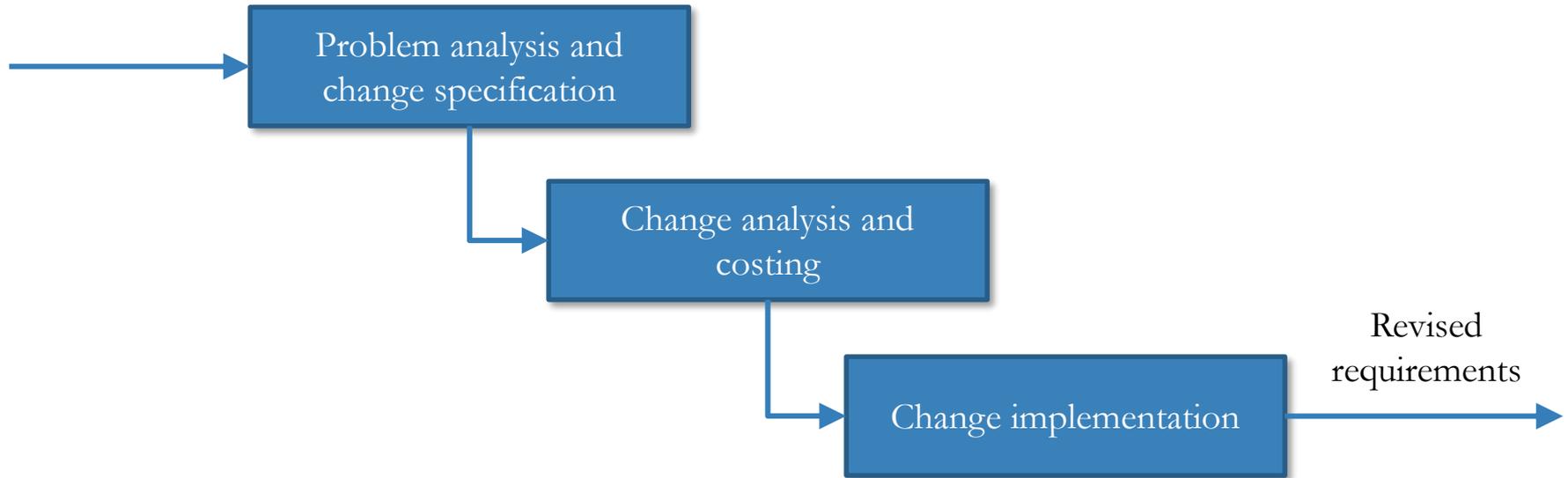
- Requirements management decisions:
  - *Requirements identification*: Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - *A change management process*: This is the set of activities that assess the impact and cost of changes (Discussed in the next slides).
  - *Traceability policies*: These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
  - *Tool support*: Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

# Requirements change management

- Deciding if a requirements change should be accepted
  - Problem analysis and change specification
    - During this stage, the problem or the change proposal is analyzed to check its validity. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.
  - Change analysis and cost
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

# Requirements change management

Identified problem



Requirements change management

# Unofficial requirement change

- Some requirement changes are "official" external changes, representing customer requests made through the appropriate channels of communications
- But many are "unofficial" (also known as "requirements leakage").  
Examples:
  - Direct customer requests to programmers
  - Functionality inserted by programmers with "careful consideration" of what's good for the customer
- Up to half of the total work product of the system are invested in requirements leakage!

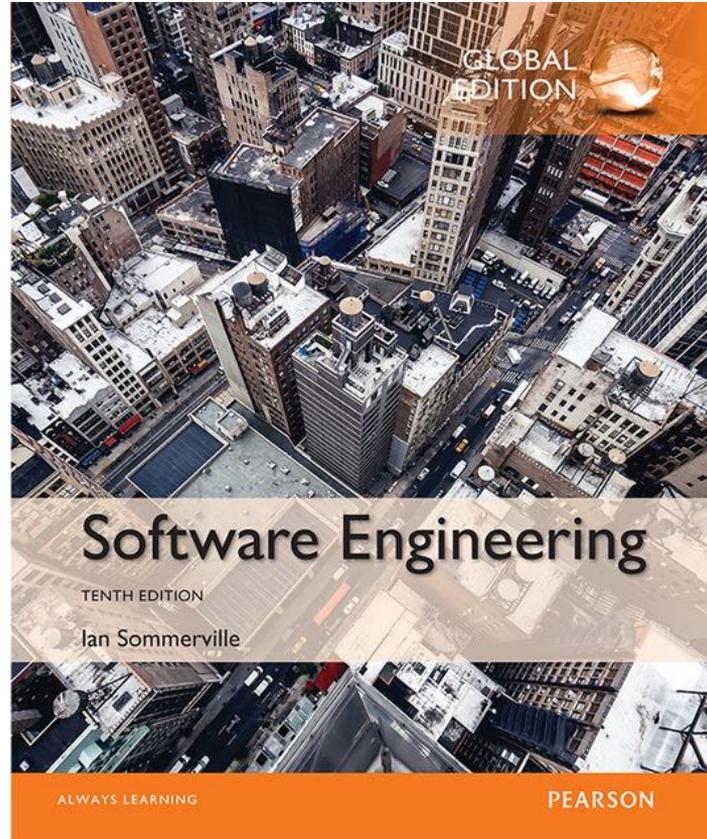
# Key points



- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
  - Business, organizational and technical changes inevitably lead to changes to the requirements for a software system.
  - Requirements management is the process of managing and controlling these changes.
- 

# Read

## Chapter 4



# References



- Ian Sommerville, “Software Engineering”, 10<sup>th</sup> Edition, Addison-Wesley, 2015.
- Timothy C. Lethbridge and Robert Laganière, “Object-Oriented Software Engineering: Practical Software Development using UML and Java”, 2<sup>nd</sup> Edition, McGraw Hill, 2001.
- R. S. Pressman, Software Engineering: A Practitioner’s Approach, 10th Edition, McGraw-Hill, 2005.

# Next



## Chapter 5 Modeling



# Course Topics

- ~~Introduction~~
- ~~Software Process Models~~
- ~~Requirements Engineering~~
- Modeling
- Programming Languages
- Software Construction Techniques
- Testing
- Project Management
- Refactoring
- Ethical Issues