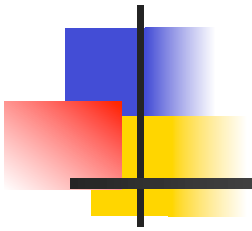
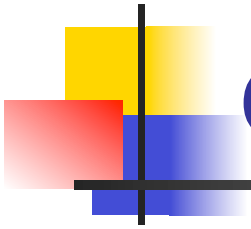


Software Requirements

Gruia-Catalin Roman

October 2014

Department of Computer Science
University of New Mexico



Course Overview

[1] Software Crisis

[2] Life-Cycle Perspective

[3] Software Requirements

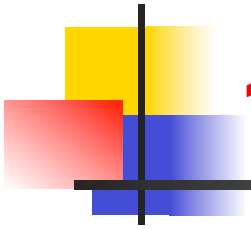
[4] Processes

[5] Products

[6] Basic Methods

[7] Complex Models

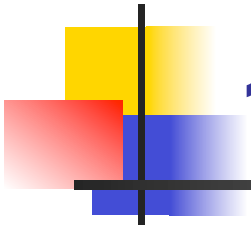
[8] Reviews



1. Software Crisis

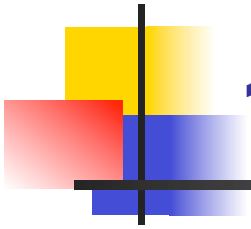
Overview

- 1.1 Symptoms
- 1.2 Causes
- 1.3 Costs
- 1.4 Response



1.1 Symptoms

- Systems are being delivered late and over budget
  - Denver Airport (losses of $1.1 million/day)
  - FAA ($1 billion over budget, five years late)
- Systems under development are canceled due to emerging gaps between the projected capabilities and the understood needs
  - two out six major projects
- Completed systems meet only in part the needs of the user
- Completed systems are never installed
- These problems are not limited to large and complex systems

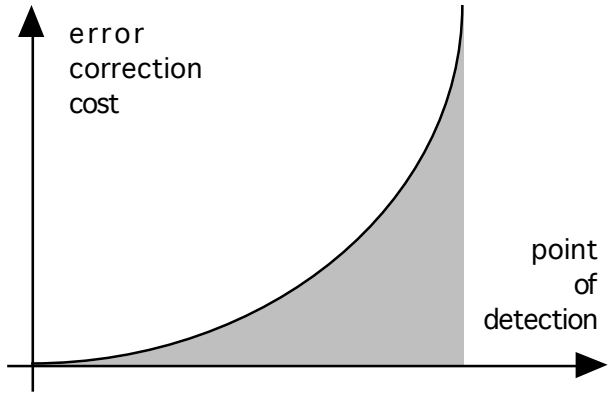


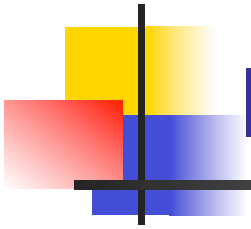
1.2 Causes

- Complexity
- System requirements
- Weak management controls
- Lack of technical maturity

1.3 Costs

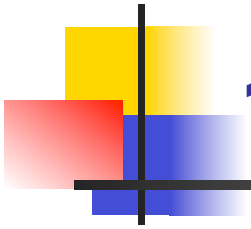
- The cost of correcting an error grows very rapidly as the project moves along its life-cycle
- This observation argues for early error detection and provides the motivation for technical reviews
- The highest cost errors are those involving the systems requirements formulation





Implications

- Problems relating to the identification and documentation of system requirements present the highest risk for a project
- Investments in other areas of the software development process can be easily undercut by problems with the requirements
- Meaningful measurement and evaluation must take into consideration the relation between the error introduction and error detection points
  - effectiveness of the quality assurance
  - weaknesses in the development process

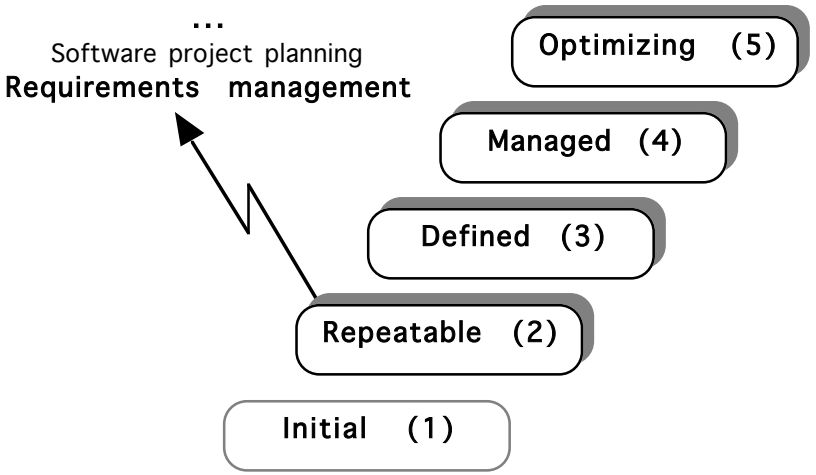


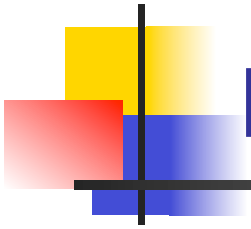
1.4 Response

- There is no silver bullet for the very difficult task of requirements definition and management
- The state of the art, however, is very much ahead of the state of the practice
- A standardized framework can be the conduit for bridging the gap
  - increased awareness
  - common terminology
  - assimilation of very basic practices

Capability Maturity Model

- Capability Maturity Model (CMM) is a framework designed to facilitate the introduction of basic sound practices across the industry
- CMM does not solve the technical problem
- CMM facilitates the adoption of sound technical and managerial practices

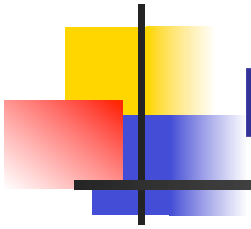




Repeatable Level

Key process areas

- Requirements management
- Software project management
- Software project tracking and oversight
- Software subcontract management
- Software quality assurance
- Software configuration management



Requirements Management

Goals

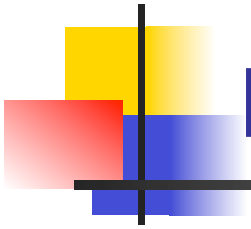
- Requirements are controlled to establish a baseline for technical and managerial use
- Plans, products, and activities are kept consistent with the requirements

Commitment to perform

- The project follows a written organizational policy for managing system requirements

Ability to perform

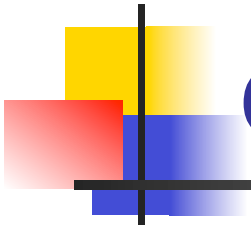
- Responsibility is established for analyzing and allocating the requirements
- Requirements are documented
- Resources are provided for managing the requirements
- Training is provided



Performance

Activities performed

- Requirements are reviewed prior to use
- Requirements are used as a basis for plans, products, and activities
- Changes to allocated requirements are reviewed



Controls

Measurement and analysis

- Measurements are made and used to determine the status of the activities for managing the requirements

Verifying implementation

- Activities for managing requirements are reviewed with senior management
- Activities for managing requirements are reviewed with the project manager
- The software quality assurance group reviews activities and products