ILLINOIS INSTITUTE OF TECHNOLOGY

Transforming Lives. Inventing the Future.

www.iit.edu

SOFTWARE ENGINEERING
CS 487

Prof. Dennis Hood

Computer Science

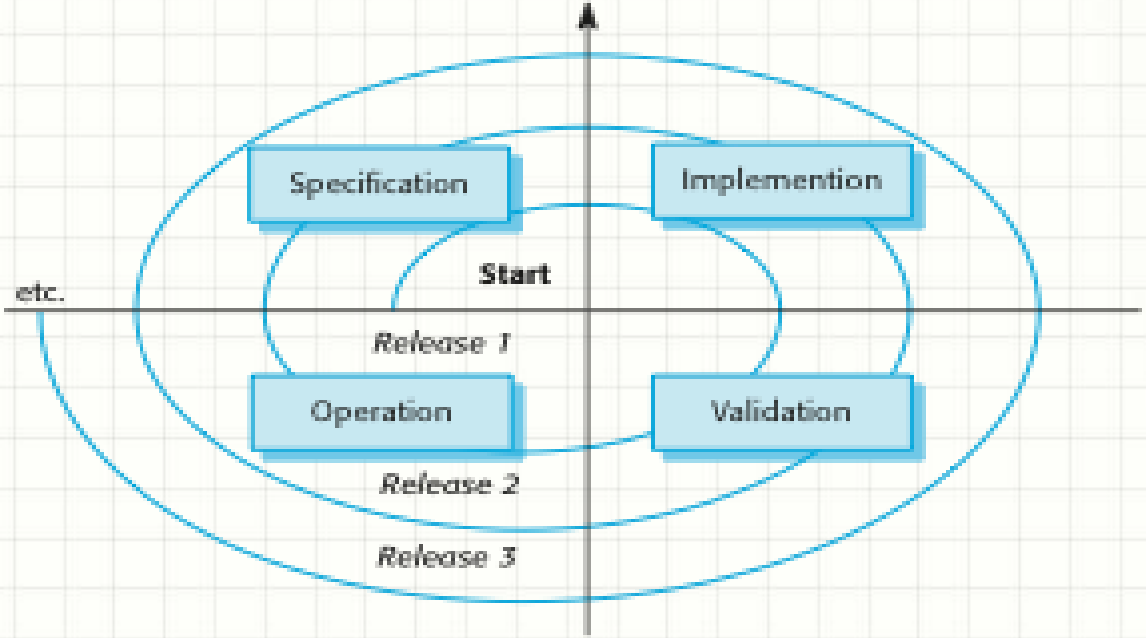
Week 9
Software Evolution and Resilience

Lesson Overview

- Evolution and Resilience
- Reading
  - Ch. 9 – Software Evolution
  - Ch. 14 – Resilience Engineering
- Objectives
  - Look at challenges associated with evolving software systems and approaches for dealing with them
  - Discuss system reengineering as a means for gaining insight into the design of legacy systems
  - Consider the concepts of resiliency and resilient engineering
  - Discuss cybersecurity in this context

Software Evolution

- Manage the inevitable change
  - Support and facilitate business growth
  - Take advantage of technology innovation
- Evolve-release-evolve-release-…



Evolution Dynamics

- Grow or progressively lose value (e.g., user satisfaction, perceived quality, etc.)
- Evolution tends to increase complexity
- Bigger systems tend to resist evolution
- Organizational bureaucracy dampens evolution
- The bigger the evolution the greater the number of associated problems

Managed Change

- Formal change requests/proposals
  - Purpose and priority
  - Cost and effort
  - Risk assessment
- Change review and authorization
  - Change control board
- Change implementation
  - Software engineering
  - Planning and management
- Change release
  - Communication
  - Rollback planning

Program Evolution Dynamics

- Change is inevitable
  - As long as the system is used there will be demand to correct imperfections, improve shortcomings and add functionality
- Change increases complexity
  - Adding to an existing structure tends to degrade stability
  - New functionality may bring new defects
- Change tends to be regulated by factors such as system and organization size
  - Big systems are more difficult to change
  - Bureaucracies impede change

Maintenance

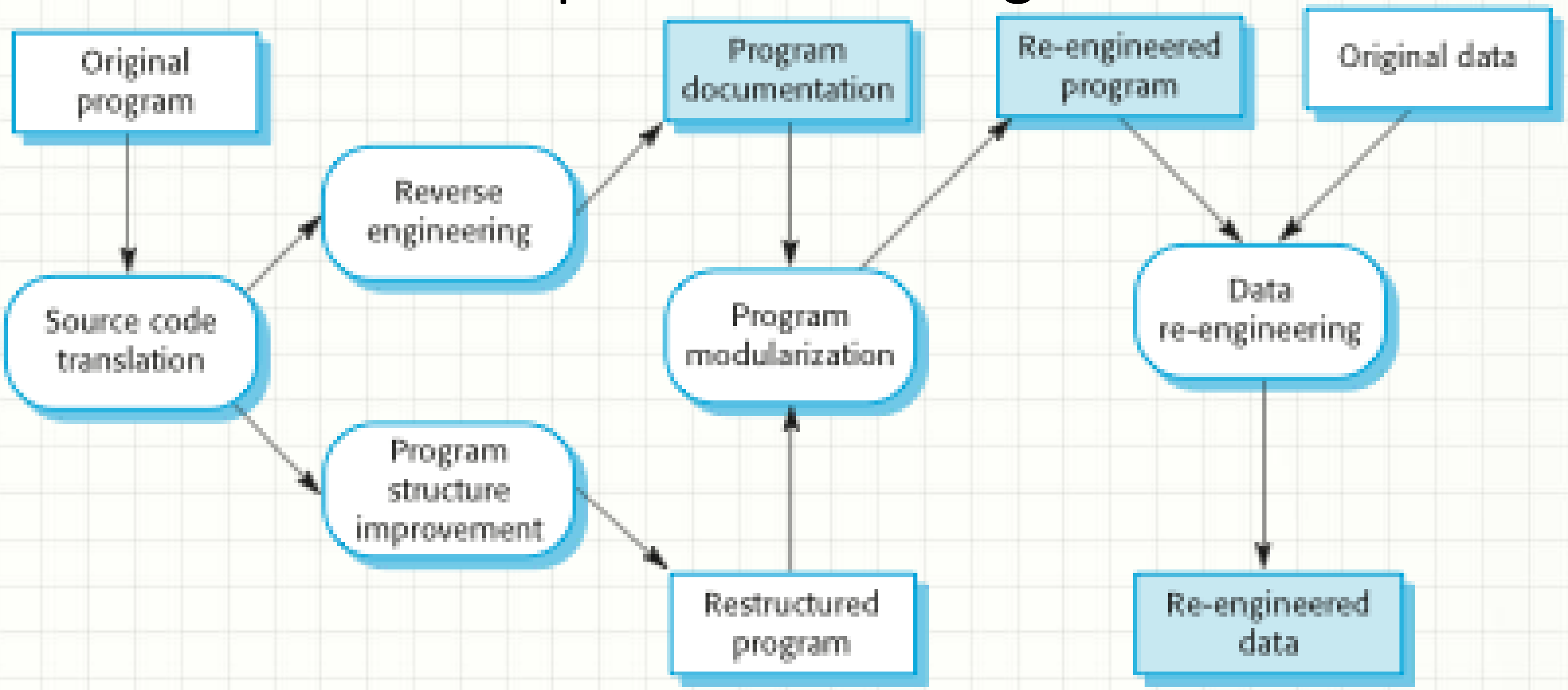
- Development effort and discipline should reduce maintenance effort and cost
  - Better analysis will result in a better alignment with user needs
  - Better design and implementation will reduce defects (including user confusion, etc.)
  - More thorough QA will minimize defects in production
- The evolutionary approach can be result in higher maintenance costs
  - Adding functionality to an existing system is more difficult
  - This issue should be countered by planning evolutions well in advance where possible
- Change should be managed with discipline
  - Defect removal
  - Enhancements

Drivers of Maintenance Activity

- Interfaces
  - Number and complexity matter
  - User and system interfaces
- Information
  - Number of data sources
  - Data structure complexity
- Volatile requirements
  - Policies and procedures
  - Business rules
  - Technology
- Processes utilizing the system
  - The more users, the more demand for change

Reengineering to Gain Understanding

- Benefits
  - Creates a newer, more maintainable version
  - Faster and cheaper than building brand new



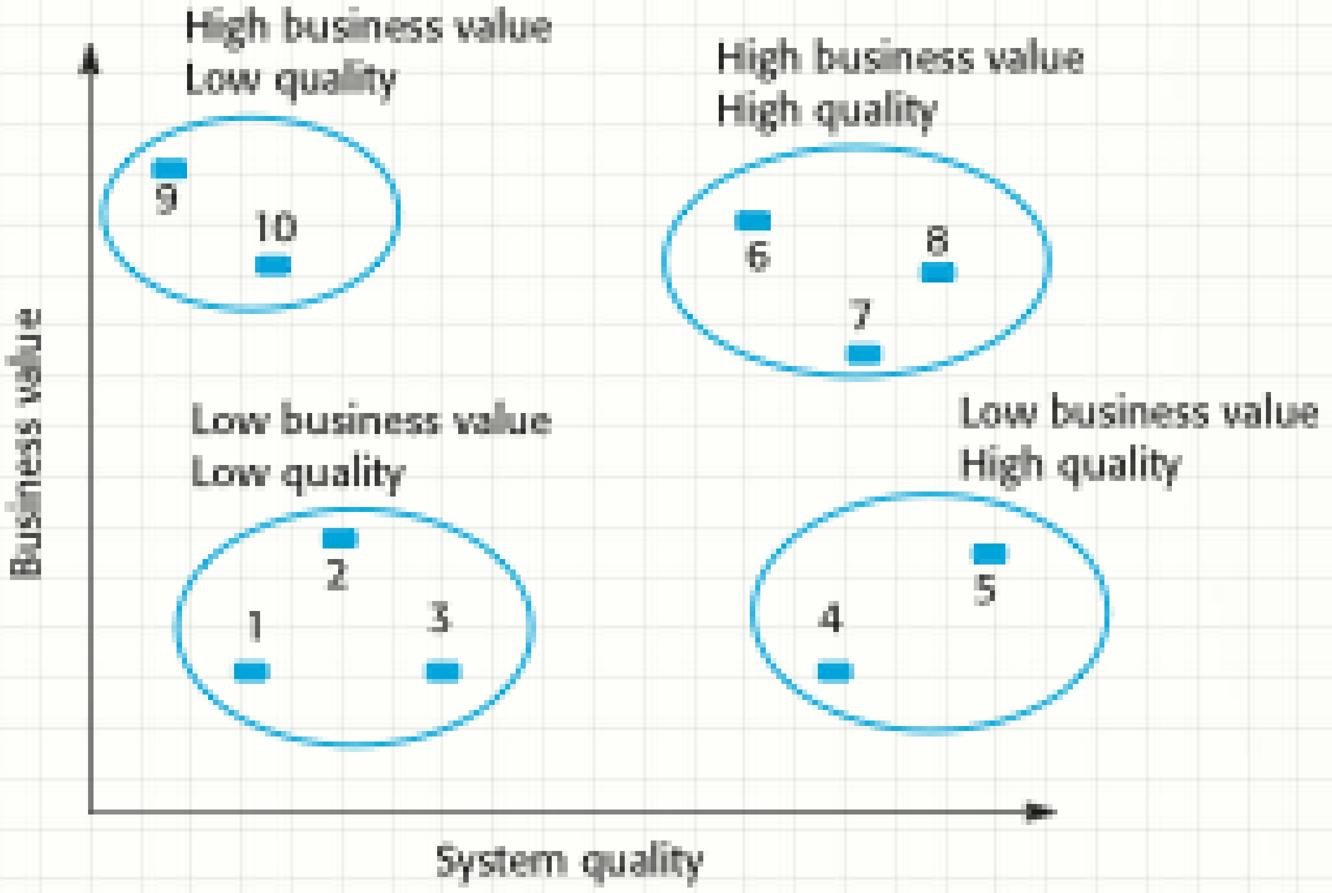
Refactoring

- Preventive maintenance thru occasional touching up to stave off degradation
  - Improve structure and performance
  - Reduce complexity
  - Improve understandability, etc.
- Improve what's already there, don't add new functionality
- Targets for refactoring improvement
  - Removal of duplicate code
  - Decomposing long methods
  - Simplify or replace "switch" statements
  - Encapsulate recurring "clumps" of data
  - Remove speculative generality

Legacy Systems

- Systems supporting critical business systems may hang around for a while
  - Change is risky
  - Need downtime to switch over
  - Domain knowledge seeps away over time
- Possible next steps
  - Scrap it
  - Leave it as is and maintain
  - Reengineer it to improve maintainability
  - Replace all or at least some of it
- Assess business value vs. system quality

Legacy System Evaluation



Sociotechnical Systems

- Systems consist of both software and hardware, and operate within an environment which includes users, partners and other systems
- The sociotechnical systems stack
  - Social
  - Organizational
  - Business process
  - Application
  - Communications and data management
  - Operating system
  - Equipment

Resilience

- The ability to withstand stress to the point of complete recovery
- The resiliency of a system therefore refers to its ability to continue to provide services as specified, even under the stress of disruptive events (e.g., equipment failures, user errors, cyberattacks, etc.)
  - Failure happens – best be able to handle it
  - There is an implied priority given to critical services
- We would benefit from an extension which includes resilience in the face of change

Exception Handling

- Resilient engineering is focused on the detection of, response to, and recovery from exceptional conditions
- Resilient activities
  - Recognition – an adequate degree of awareness
  - Resistance – early detection may allow for "fighting off" the condition
  - Recovery – timely restoration of (especially critical) systems to minimize disruption
  - Reinstatement – return to normal operation

Cybersecurity

- Our current massively interconnected environment offers many paths for opportunistic would-be invaders
- Adequately securing system functions and sensitive data has been always been a (non-functional) requirement
- Increasingly this has extended to cybersecurity – maintaining adequate security in a networked environment

Security Controls

- Authentication
  - Forcing users to prove that they are authorized to access assets
- Encryption
  - Altering the form of sensitive data to the point that unauthorized users would not be able to understand or make use of it
- Firewalls
  - Protective checkpoints for restricting traffic flow to only trusted sources