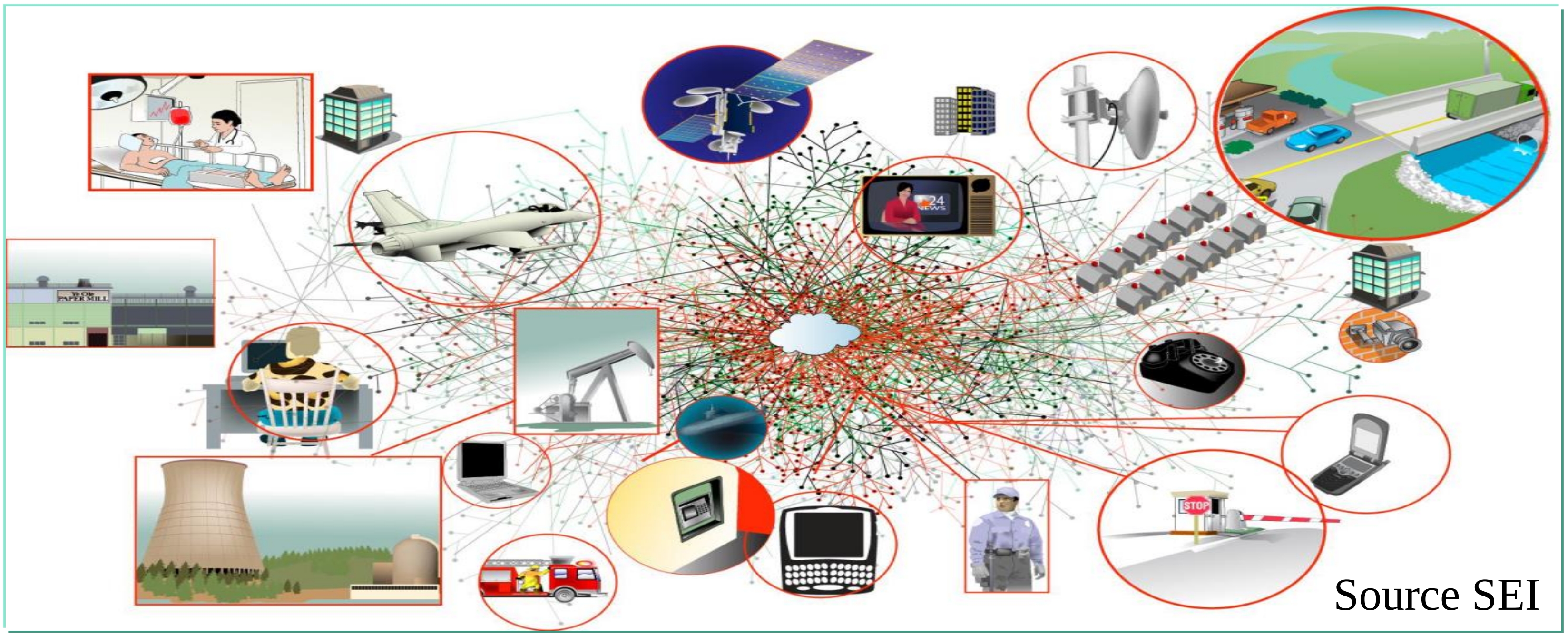
The Better Way of Managing and Leading Software Intensive Projects

Chapter 1

Introduction



Dr. Kenneth E. Nidiffer

Successfully Managing and Leading Software Projects
Software Engineering 625
Volgenau School of Information Technology and Engineering
George Mason University

Class Syllabus for SWE 625

Spring 2020

Overview of the Class Syllabus for SWE 625

- Why Take SWE 625
- Scope
- Conclusions – DIB (Defense Industrial Board) Study on Software Acquisition and Practices (SWAP) Study Biography
- Administration
- Course Text
- Major Topics
- Course Background Requirements
- Course Evaluation Procedure
- Lecture Topics and Homework Schedule
- Course Materials
- Blackboard Learn

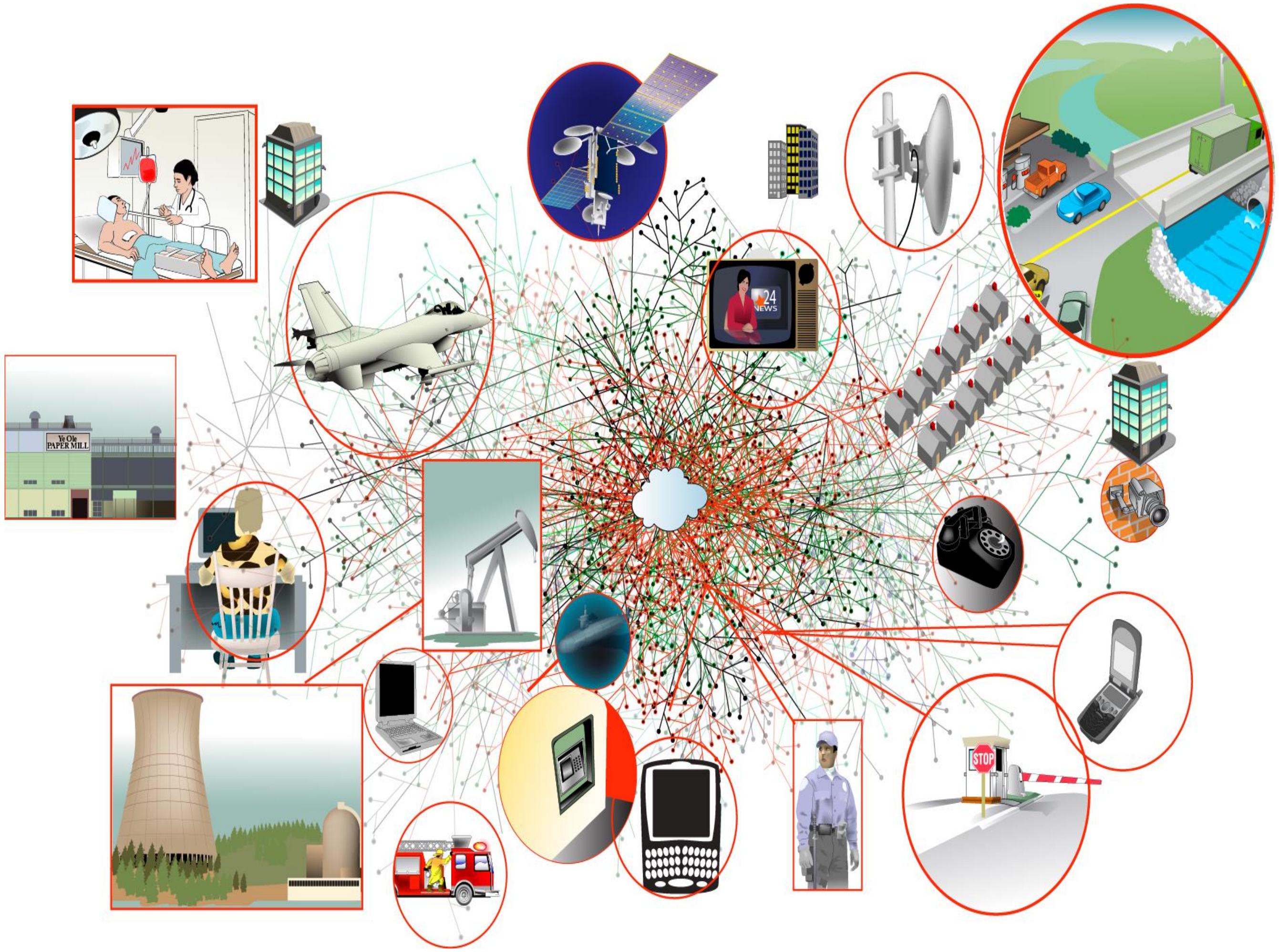
Why Take SWE 625

- Successfully managing software intensive projects is a **priority** for the industrial, government and academic organizations
- The ubiquity of software and its critical role require fundamental shifts in software engineering management and engineering to maintain competitive advantage
- The course helps participants to **rapidly deploy innovation with confidence** within this shifting landscape by:
  - Applying new principles in software **engineering management for software intensive systems**
  - Developing new practices for **enabling business/mission capability with software innovation**
- **Equips students** in applying new management techniques in today's competitive job market

Scope of Software Engineering (SwE) Management

Mission Focused

- System of Systems – (SwE, Cyber and AI/ML)
- Networked Hardware/ Platforms
- Infrastructure
- Applications
- Workforce: People who digitally connect to cyberspace



Source: SEI

Conclusions - Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) Study – 11 Jan 2019

- **Software is ubiquitous and U.S. national security relies on software.** Well-equipped and well-trained warfighters provide the capability necessary to defend the nation, but software critically enables that mission. The ability to develop, procure, assure, and deploy software is central to national defense and integrating with allies and partners.
- **Speed and cycle time are the most effective metrics for software.** Software is a critical element of the Department’s approach to executing missions, collaborating with allies, and managing its operations. DoD needs to deploy & update software at the speed of (mission) need, and execute within the OODA loop of our adversaries to maintain advantage.

The **OODA loop** is the cycle observe–orient–decide–act, developed by military strategist and United States Air Force Colonel John Boyd.

Conclusions - Defense Innovation Board Software Acquisition and Practices (SWAP) Study – 11 Jan 2019

- **Software is made by people, for people, so digital talent matters.** DoD’s current personnel processes and culture will not allow its military and civilian software capabilities to grow nearly enough. New mechanisms are needed for attracting, educating, retaining, and promoting digital talent, and providing the ecosystem that enables them to succeed.
- **Software is different than hardware (and not all software is the same)**. Hardware can be developed, procured, and maintained. Software is an enduring and evolving capability that must be supported and continuously improved throughout its lifecycle. The DoD acquisition process and culture need to be streamlined for effective delivery and oversight of multiple types of software-enabled systems, at scale, and at the speed of relevance.

Biography

DR. KENNETH E. NIDIFFER, PMP
Principal Software Engineer
Software Engineering Institute, Carnegie Mellon University

Dr. Nidiffer has over fifty-seven years of experience in the marketing, research, development, support, maintenance, and acquisition of software-intensive systems. His 24-year career in the U.S. Air Force (where he retired as a full colonel) is marked by several firsts in the area of software implementations, such as, first space-based compiler, first command-hardware in the loop simulation, a series of development/process standards, etc. From 1983-1986 he helped establish several noteworthy contributions, such as, the Software Productivity Consortium; the Software Project Management Program at the Defense Systems Management College; the George Mason Software Engineering Program and the Software Engineering Institute. At the Software Productivity Consortium he launched the Consortium’s business initiative in software process improvement, which became one of the largest programs in the world.

In 1991, Dr. Nidiffer left the Consortium to serve one of its founding members, Northrop Grumman, as Director of Systems Design and Development, Data Systems Division, and then as Director of Technical Operations, External Data Systems division, where he directed over 500 engineers and support personnel in the successful development of a variety of C4I, MIS/logistics, and high-speed computing applications.

Biography

In 1995, he joined Fidelity Investments Systems Company as Senior Vice President of Quality and Systems Assurance to lead a team of 165 professionals in implementing Total Quality Management, best-in-class software engineering processes, and the largest financial services test environment. He rejoined the Consortium in 1997 as Vice President for Business Development growing the membership from 50 to 100 members. In 2007 he joined the Software Engineering Institute to focus on promoting key software engineering technologies that support government programs. He is currently a Principal Member of the Technical Staff

Dr. Nidiffer has been widely published in the systems and software engineering community. He received his B.S. degree in Chemical Engineering in 1962 from Purdue University, Indiana, a M.S. degree in Astronautical Engineering in 1969 from the Air Force Institute of Technology, Ohio, a MBA degree from Auburn University, Alabama in 1975 and his D.Sc. degree from George Washington University, Washington D.C. in 1988.

He is a member of the Program Management Institute (PMI); the International Council on Systems Engineering (INCOSE); the Air Force Association (AFA); Senior Member of the Institute of Electrical and Electronics Engineers (IEEE) and Member of the IEEE Professional and Activities Board (PAB); the Inter-National Committee for Information Technology Standards (INCITS)/Software and Systems Engineering (INCITS/SSE) Technical Committee, Senior member of the American Institute of Aeronautics and Astronautics (AIAA); member of the National Defense Industrial Association (NDIA Systems Engineering Division); Co-Chair of the NDIA/OSD (DDR&E) Industrial Software Committee and Co-Chair of the NDIA Systems Engineering Education and Training Committee.

Biography

Ken is a certified logistician; a Professor Emeritus of the Defense Systems Management College; Industry Advisor on George Mason's Computer Science Education Committee; a Project Management Professional; and an adjunct engineering professor in graduate engineering at George Mason University for over 28 years.

Dr. Nidiffer is a man of faith and a family-oriented person. He has been married for 56 years to the former Mary Emma Walsh of Havana, Florida and they have three daughters: Sheri, Kristi and Kathi and three grandchildren. In 2002 and in 2007, he was selected as the School of Information Technology's adjunct professor of the year in Software Engineering and received special recognitions for his GMU adjunct teaching service in 2009. 2013, 2017, and 2018.

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

• COURSE TITLE:	Software Engineering Project Management (SWE 625)
• INSTRUCTOR:	Professor Kenneth E. Nidiffer
• SEMESTER CLASSES:	Spring 2020 (27 Jan to 11 May 2020, including final exam)
• SEMESTER FINAL EXAM:	11 May, IH, Room 206 *
• CLASS TIME/BLDG/ROOM:	1920 – 2200; IH, Room 206**

*Note 1: The student will be provided a reading day to prepare for the final exam 5 May

**Note 2: IH = Innovation Hall

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

- OFFICE HOURS: 1815 - 1900 Mondays

  ENGR 5309, Nguyen Engineering Building (Academic IV, Research II)
- Meeting Arrangement Mechanisms:
  - By appointment in class
  - By the Internet – knidiffe@gmu.edu – Best alternative
  - By note in my mail box – Suite 4300, Nguyen Engineering Bld.
  - By setting-up a conference call
  - By setting-up a video-teleconference (VTC)
  - Department Administration Assistant
    - Ms. Beth Posocco: 703-993-1568

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

- CONTACT INFORMATION:
  - Internet/E-mail: knidiffe@gmu.edu (Best Method)
  - Oral Communication Mechanisms
    - Method 1: (703) 455-4021(Home Phone Number) - Best Method
    - Method 2: (703) 217-0215 (Cell Phone) or Text – Good Alternative Method

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

TEXT 1:Title - Managing and Leading Software Projects
Dated: 2009*
ISBN 987-0-470-29455-0
Author: Dr. Richard E. (Dick) Fairley
Publisher: John Wiley & Sons, Inc.
Options to Obtain:

1. Can Pick-up at University Bookstore (located in the George W. Johnson Center)
2. Order on-line
3. Obtained previously owned book

* Students are expected to study and understand the contents of the course text book

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

TEXT 2:Title - Systems Engineering of Software-Enabled Systems
Dated: 2019*
ISBN 9781119535010
Author: Dr. Richard E. (Dick) Fairley
Publisher: John Wiley & Sons, Inc.

- Note 1: Students can but are not expected to buy this text book since it can be downloaded for free*
- https://wiki-int.sei.cmu.edu/confluence/download/attachments/285903764/SysE_of_SwEnabled_Sys_Fairley_2019_9781119535041.pdf?api=v2

- Note 2: A copy has been place in Blackboard – Lecture 1

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

COURSE PREREQUISITES:
Undergraduate courses or equivalent knowledge in structured programming in a high-level language, data structures, discrete mathematics, and machine organization or assembly programming.

COURSE DESCRIPTION:
This course is concerned with processes involved in project planning; organizing; staffing; estimating; measuring and controlling; communication, coordination and leadership; and risk management. Topics covered include lifecycle delivery approaches; process and engineering product development models with special emphasis on the best practices contained in the Capability Maturity Model Integrated (CMMI©) constellations and product standards. The course also stresses the Program Management Institute's Program Body of Knowledge (PMBOK©) and the Software Engineering Body of Knowledge (SWBOK).

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

COURSE OBJECTIVES:

Upon completion of this course, students will know how to develop a software project management plan for software intensive systems; how to set up monitoring and control mechanisms; how to allocate and reallocate project resources; how to track schedule, budget, quality, productivity, and progress; understand the CMMI© frameworks and how to plan for the installation and sustainment phase of the system life cycle. They will understand the importance of the work breakdown structure and its relationship to the delivery lifecycle, resource planning and execution, and progress and product measures from both a project and enterprise perspective. In addition, they will understand the relationships among quality assurance, configuration management, verification and validation, and test and evaluation. They will also gain an understanding of the key issues in costing and pricing units of effort, motivation of workers, agile development, Secure DevOps, leading project teams, machine learning, ethics and total quality management.

MAJOR TOPICS:

A taxonomy of management functions; corporate goals and objectives; system, project and product (functional and non-functional) requirements; architectural frameworks; best practice frameworks, cost estimation techniques and models; software process development models with special emphasis on the CMMI© and software systems engineering delivery models; technical methods; documentation, quality assurance, configuration management, verification and validation, test and evaluation; staffing plans; monitoring and controlling mechanisms; standards (e.g. IEEE/EIA 12207 and IEEE Std. 16326™), policies and acquisition frameworks (i.e. Defense (e.g. DODI 5000.02, Defense Acquisition Guidebook (DAG) and Commercial (e.g. Infrastructure Service Provider (ISP) /Application Server Provider (ASP) frameworks; Platform as a Service (PaaS), Software as a Service (SaaS)), and procedures; work packages, schedules, budget, accounting systems, costing and pricing units of effort; risk management; post deployment software support; leadership, ethics, team building and total quality. Also, Defense Innovation Board (DIB) and Defense Science Board (DSB) findings will be addressed.

EVALUATION PROCEDURE:

Grades will be based on student homework, class contributions, student presentation and the final exam in the following proportions:

Class Contribution (Contributions In Addition to the Six Articles*)	10 %
Homework	10 %
Six Articles*	10%
Project	15 %
Student Project Presentations**	10 %
Final Exam***	45 %

Note: Final exam is scheduled for 11 May 2020 (7:30 – 10:10 pm)

* Articles are to submitted in class. Students can submit their articles during any class period. N<u>ote: All articles will be accompanied with a one-page analysis of each article.</u> Three articles are to be from refereed sources and three can be from any source.

** 1920-2200/Innovation Hall Building; Room 206

*** 1930-2210/Innovation Hall Building; Room 206

Lecture Topics

Session	Date	Topic
1	27-Jan	Introduction to Project Management
2	3-Feb	Process Models for Software Development
3	10-Feb	Establishing Project Foundations
4	17-Feb	Plans and Planning
5	24-Feb	Project Planning Techniques
6	2-Mar	Estimating Techniques
	9-Mar	Spring Break

Lecture Topics

Session	Date	Topic
7	16-Mar	Measuring and Controlling Work Products
8	23-Mar	Measuring and Controlling Work Processes
9	30-Mar	Managing Project Risk
10	6-Apr	Teams, Teamwork, Motivation, Leadership and Communication
11	13-Apr	Organizational Issues
12	20-Apr	Furture of Software Engineering and It's Impact on Society
13	27-Apr	Student Presentations (1920 – 2200/ IH Room 206)
14	4-May	Student Presentations (1920 – 2200/ IH Room 206
	5-May	Reading Day
15	11-May	FINAL EXAM (1930-2210/ IH Room 206

GEORGE MASON UNIVERSITY
VOLGENAU SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE
COURSE OVERVIEW
SOFTWARE ENGINEERING PROJECT MANAGEMENT 625

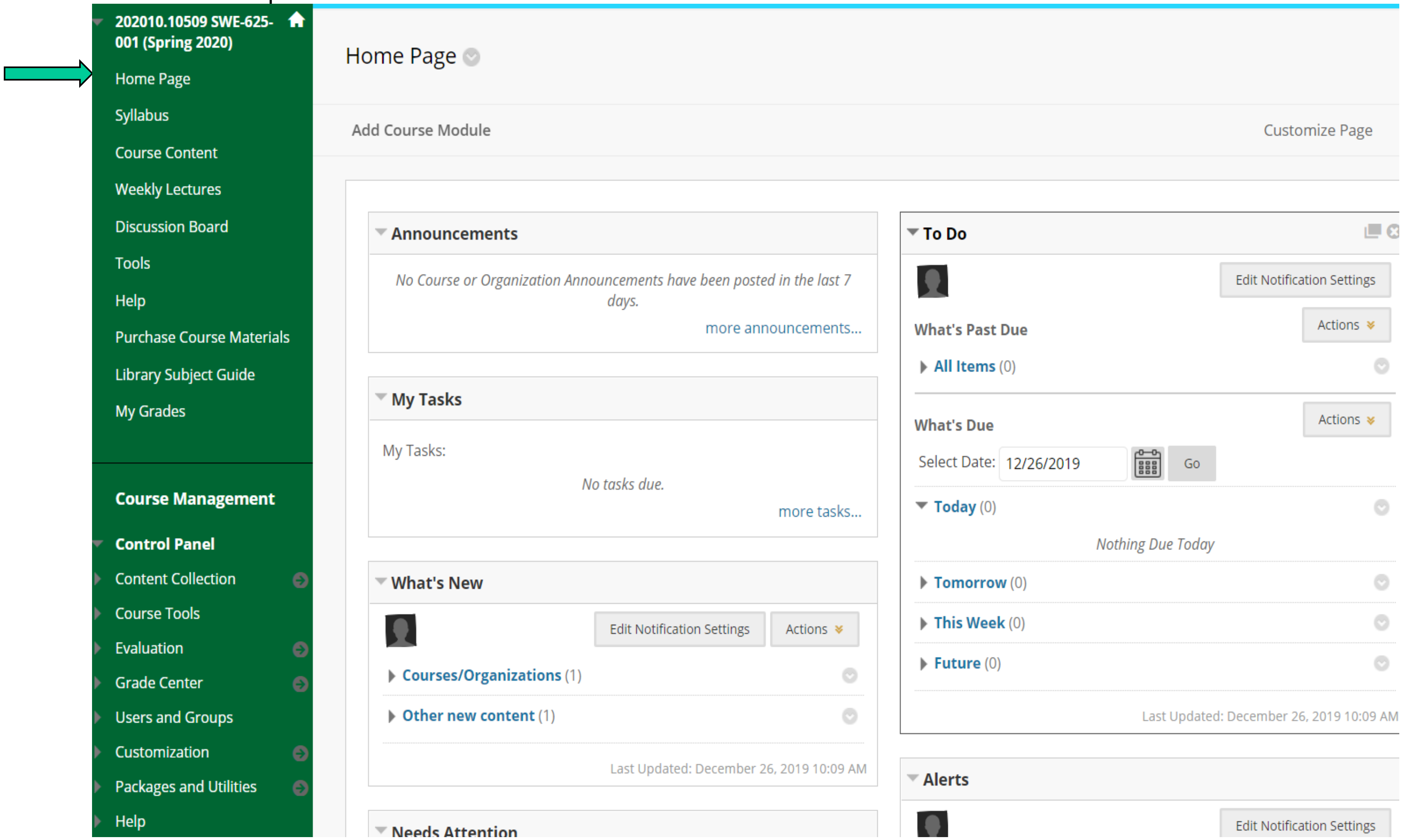
Course Materials	**Location** (Blackboard Learn)
1. Announcements	On-line Folder/In-class
2. Administrative Notes	On-line Folder/In-class
3. Presentation (Slides)	On-line Folder*
4. Student Handouts	On-line Folder/In-class
5. Student Responses	In-class
6. Graded Responses	In-class
7. Student Presentation Mat’ls	In-Class
8. Student Final Responses	In-Class
9. Class Contributions	In-Class

Blackboard Learn

- Blackboard Learn (previously the *Blackboard* Learning Management System) is a virtual learning environment and course management system developed by *Blackboard* Inc.
- Used by George Mason University
- SWE 625 Course information and assignments are contained on Blackboard Learn
- The key file is “Weekly Lectures”

Blackboard Learn

Spring 2020



202010.10509 SWE-625-001 (Spring 2020)

- Home Page
- Syllabus
- Course Content
- Weekly Lectures
- Discussion Board
- Tools
- Help
- Purchase Course Materials
- Library Subject Guide
- My Grades

Course Management

Control Panel

- Content Collection
- Course Tools
- Evaluation
- Grade Center
- Users and Groups
- Customization
- Packages and Utilities
- Help

Home Page

Add Course Module — Customize Page

Announcements

No Course or Organization Announcements have been posted in the last 7 days.

more announcements...

My Tasks

My Tasks:

No tasks due.

more tasks...

What's New

Edit Notification Settings — Actions

Courses/Organizations (1)

Other new content (1)

Last Updated: December 26, 2019 10:09 AM

Needs Attention

To Do

Edit Notification Settings

What's Past Due — Actions

All Items (0)

What's Due — Actions

Select Date: 12/26/2019 Go

Today (0)

Nothing Due Today

Tomorrow (0)

This Week (0)

Future (0)

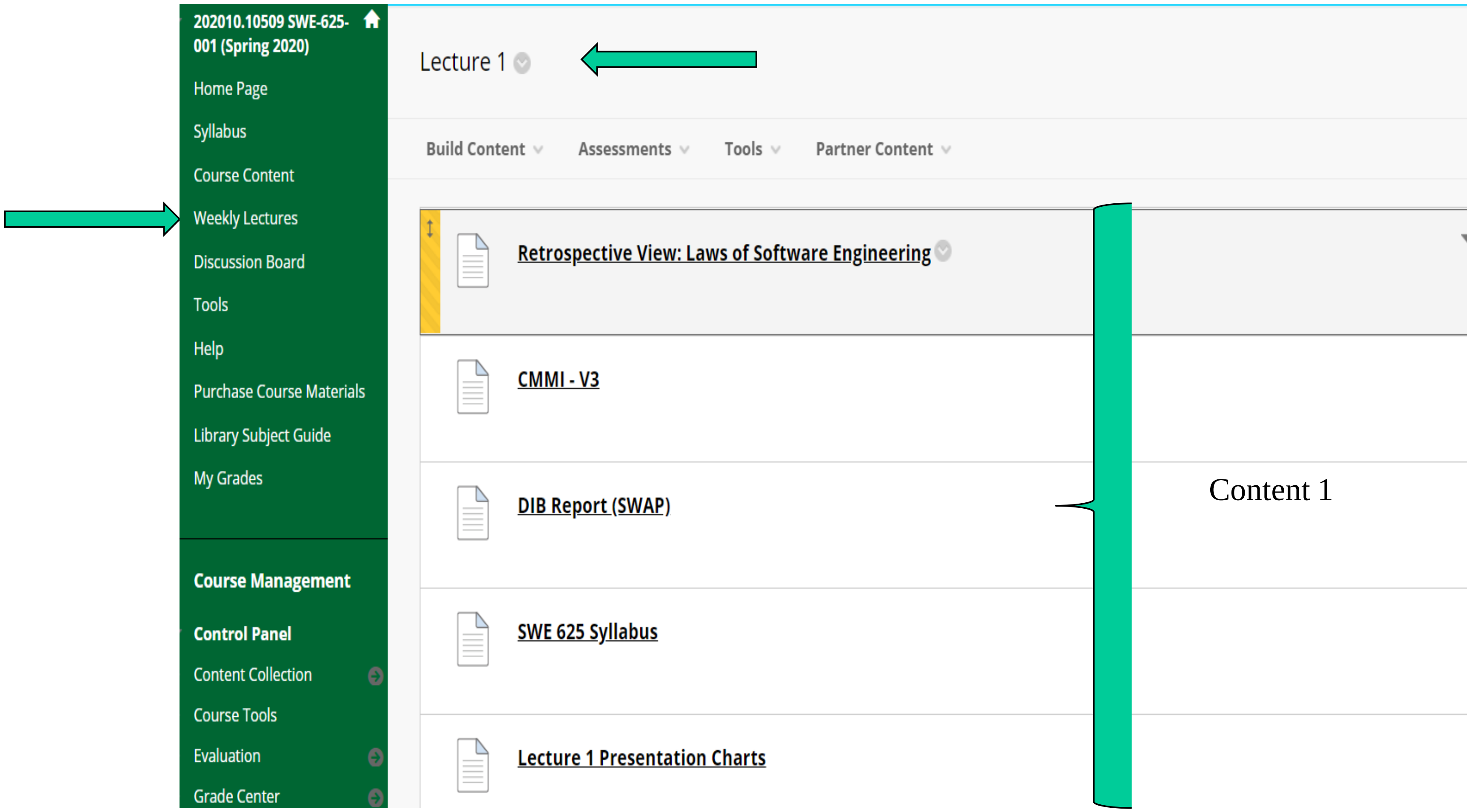
Last Updated: December 26, 2019 10:09 AM

Alerts

Edit Notification Settings

Blackboard Learn

Lecture 1



202010.10509 SWE-625-001 (Spring 2020)

Home Page

Syllabus

Course Content

Weekly Lectures

Discussion Board

Tools

Help

Purchase Course Materials

Library Subject Guide

My Grades

Course Management

Control Panel

Content Collection

Course Tools

Evaluation

Grade Center

Lecture 1

Build Content | Assessments | Tools | Partner Content

Retrospective View: Laws of Software Engineering

CMMI - V3

DIB Report (SWAP)

SWE 625 Syllabus

Lecture 1 Presentation Charts

Content 1

Questions

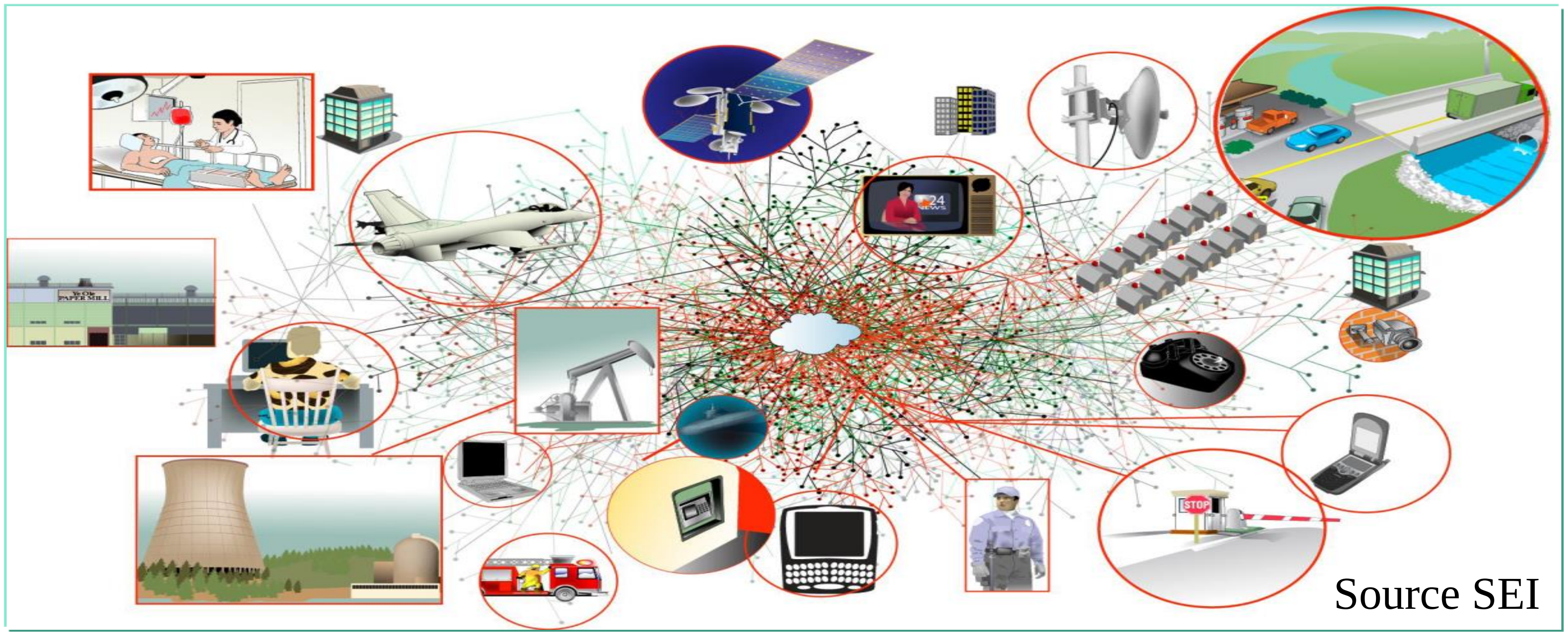
QUALITY
PRODUCTIVITY
JOB KNOWLEDGE
SKILLS
RESPONSIBILITY
ADAPTABILITY
RESOURCEFULNESS

leadership

The Better Way of Managing and Leading Software Intensive Projects

Class 1

Lecture Slides



Introductions and Expectations

- Participant Introductions
  - Name
  - Position and background
  - Experience with software project management
    - how long
    - experience with other guides or standards
- Expectations
  - What do you want to get out of this course

Source: DAU

Topics

- Fundamental Shifts in Software Management & Engineering
- Why Managing and Leading Software Projects Is Difficult
- The Nature of Project Constraints
- A Workflow Model for Managing Software Projects
- Organizational Structures for Software Projects
- Organizing the Project Team
- Maintaining the Project Vision and the Product Vision
- Frameworks, Standards, and Guidelines

Enabling Learning Objectives

- Motivation – We Need a Better Way!
- The main elements of software project management
- The influence of project constraints
- Why managing and leading software projects is difficult
- A workflow model for software projects
- The work products of software projects
- Organizational structures for software projects
- Organizing a software development team
- Maintaining the project vision and product goals
- Frameworks, standards, and guidelines

Fundamental Shifts in Software Management & Engineering

As software and systems are increasingly becoming **bigger**, **more complex**, and **intertwined**, software management and engineering and the roles people play are evolving in response.

Time →

Developers write code	Models generate code	AI/ML assists in generating models/code
Software release based on milestones (typically 12 – 24 months)	Continuous integration and continuous deployment (CI/CD)	Automated release-observe-refine
Collect data and evidence from past projects to make predictions	Moving beyond prediction to determining causality	Feedback of data and results to re-train models
Software and hardware must work together	Increasing diversity of languages, platforms, hardware & systems must be made to work together	Systems of people, policies, sensors, software, hardware, etc., continuously learn ways to work together
Developers do nearly everything	Developers determine processes and rules and create automation	Machines continually learn what to do to achieve goals
Black box test for correctness	Formal analysis of correctness	Mathematically verified enforcers watch rest of system
Human in the loop (humans invoke computers)	Humans on the loop (humans monitor computers)	Humans out of the loop (computers notify humans only when needed)

References- Building on Project Management Anchor Points Systems Engineering of Software-Enabled Systems

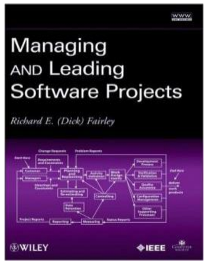
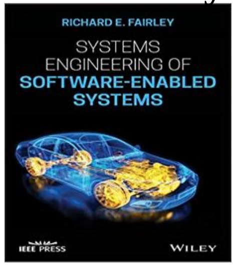
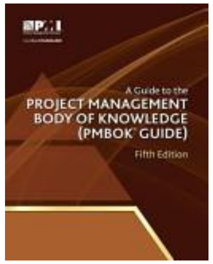
Managing and Leading Software Projects

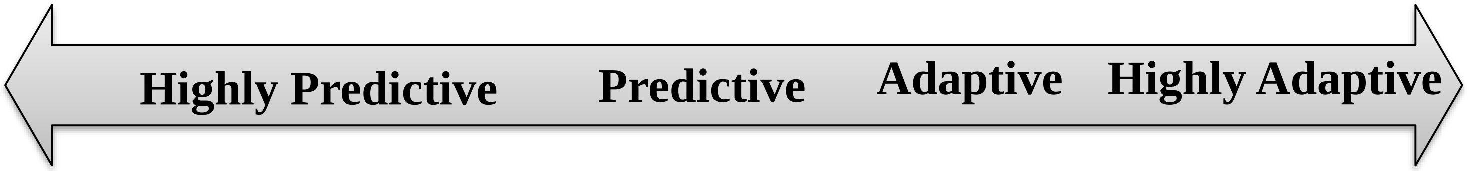
Systems Engineering of Software-Enabled Systems

PMBOK Sixth Ed

SWX PMBOK Fifth Ed

Research & Studies



Development & Acquisition Lifecycles

Accepted Standards and Process and Development Frameworks

- **IEEE Standards**
  - ➢ **16326 - Project Management Plan (PMP)**
  - ➢ **12207 - Systems and software engineering — Software life cycle processes**
- **Capability Maturity Model Integration (CMMI) Version 3**

Workforce Frameworks

- **IT BOK- Skills for the Information Age (SFIA)**

Successfully Managing and Leading Software Projects,

Assignments Due for Next Period

- Assignments:
  1. Study Chapters 1 & 2 in course Text 1
  2. Read and Prepare Comments on Article: A Retrospective View of the Laws of Software Engineering, Capers Jones, 2017
  3. Answer questions: 1.1; 1.3; 1:17

- All assignments are to be turned in at the beginning of the next class period.
- All articles should be accompanied with approximately a one-page analysis (i.e. 50% on the content and 50% on your view of the article).

Why We Need a Better Way !

Discussion

Why do you think software-intensive projects are hard to manage and lead?

Seventeen Reasons Why Software-Intensive Projects Are Hard to Manage and Lead?*

1. Software requirements often change during a software project as knowledge is gained and the scope of the project and the product emerge.
2. Requirements for new and modified software often influence, and are influenced by, an organization's business processes and the workflow processes of employees.
3. Intellectual capital of software personnel is the primary capital asset for software projects and software development organizations because software is a direct product of human cognitive processes.
4. Communication and coordination within software teams and with project stakeholders often lack clarity. Many of the tools and techniques used in software engineering are intended to improve communication and coordination.

*SWX PMBOK Fifth Ed

Seventeen Reasons Why Software-Intensive Projects Are Hard to Manage and Lead?*

5. Creation of software requires innovative problem solving to create unique solutions. Most software projects develop unique products because replication of existing software is a simple process, as compared to replication of physical artifacts. Software projects are more akin to research and development projects than to construction or manufacturing projects.
6. Exhaustive testing of software is impractical because of the time that would be required to test all logical paths and interfaces under all combinations of input data and other input stimuli.
7. Software development often involves inclusion of different vendor products and development of interfaces to other software; this may result in integration and performance issues.

* SWX PMBOK Fifth Ed

Seventeen Reasons Why Software-Intensive Projects Are Hard to Manage and Lead?*

8. Software projects involve risk and uncertainty because they require innovation, the product is intangible, and stakeholders may not effectively articulate, or agree on, the needs to be satisfied by the software product.
9. Initial planning and estimation for software projects is challenging because they depend on requirements, which are often imprecise and on historical data, which is often missing or inapplicable. Preparation of accurate estimates is also challenging because the efficiency and effectiveness of software developers is widely variable.
10. Product complexity makes development and modification of software challenging because of the enormous number of logical paths within program modules combined with data values that exercise the paths, and the combinations of interface details among program modules.

* SWX PMBOK Fifth Ed

Seventeen Reasons Why Software-Intensive Projects Are Hard to Manage and Lead?*

11. Because most software is interconnected, information security techniques are necessary. Software security is a large and growing challenge.
12. Objective quantification and measurement of software quality is difficult because of the intangible nature of software.
13. Software developers use processes, methods, and tools that are constantly evolving and are frequently updated.
14. Software is often the element of a system that is changed when functionality, behavior, or quality attributes must be changed.
15. A software product may be required to operate on a variety of hardware platforms and infrastructure software

* SWX PMBOK Fifth Ed

Seventeen Reasons Why Software-Intensive Projects Are Hard to Manage and Lead?*

16. Executable software is not a standalone product. It is executed on computing hardware and is often an element of a system consisting of diverse hardware, other software, and manual procedures.
17. Platform technologies, infrastructure software, and vendor-supplied software are frequently changed or updated, which can necessitate changes to the software being developed.

* SWX PMBOK Fifth Ed

Why We Need a Better Way

- Software is a part of the very fabric of civilization, living in its interstitial spaces
  - The complexity of software-intensive systems continues to grow; this complexity impacts its users as well as the stakeholders who develop, deploy, operate, and evolve them
  - There is a plethora of articles on the problems associated with software systems
  - We know a lot about classical project management
  - Unfortunately we know very little about successfully managing an leading software intensive projects

What is a Project?

A project is characterized as follows:

- a one-time effort is planned
- starting and ending dates are prescribed
- a project team is assembled
- schedule and budget are allocated
- well-defined objectives are established
- roles are identified, responsibilities are assigned, and authority is delegated

Software projects are temporary organizational units

What is Management?

- Management is concerned with planning and coordinating the work activities of others so that they can achieve goals that cannot be achieved by each individual acting alone

Synergy: the combined effect is greater than the sum of the individual effects

What is Software Project Management (SPM)?

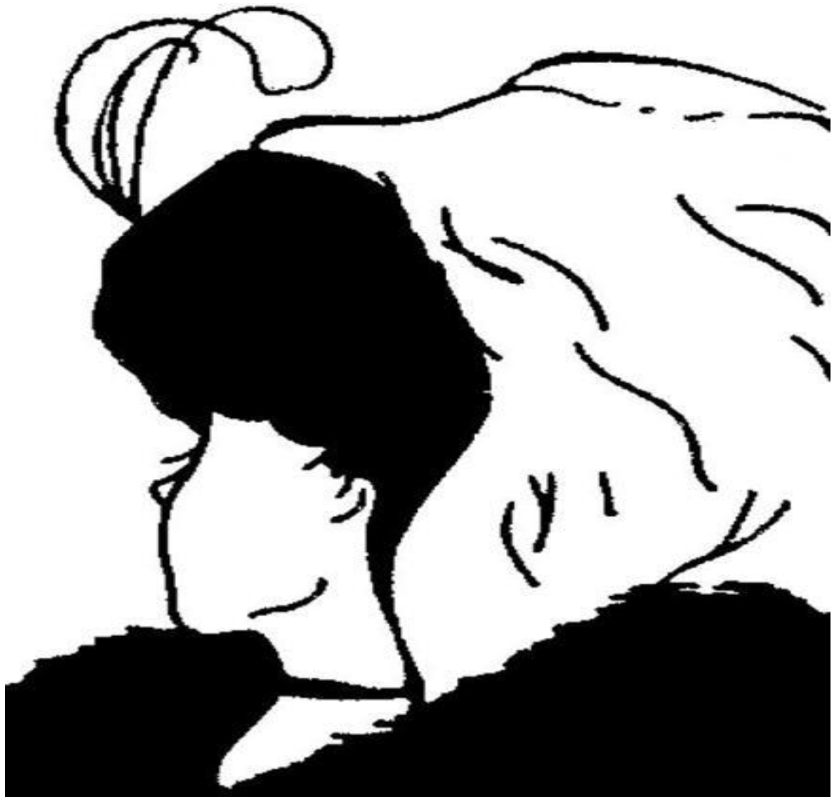
Software Project Management (SPM) is the art and science of

- planning and coordinating the work of software developers and other personnel
- to develop and modify software artifacts using as appropriate abstraction design techniques
- that are pleasing to users and customers
- that are developed and modified in an economical and timely manner
- and that can be sustained/continuos engineered efficiently and effectively

Abstraction: Software and Systems Engineering deal more than other forms of engineering with ideas that reduce and factor out details so that one can focus on a few concepts at a time

Please Close Your Eyes

Abstractions Can Lead to Unintended Perspectives



Why are multiple views important?

The Ten Major Activities of Software Project Management

1. Planning, Organizing, Staffing and Estimating
   - o identify work activities
   - o determine organizational structure
   - o prepare a schedule
   - o prepare a budget
2. Measuring and Controlling Processes and Product
   - o requirements
   - o quality and productivity
   - o schedule and budget
   - o product evolution
3. Leading, Coordination and Communicating
   - o motivating / coaching / educating project members
   - o communicating with management, customers, subcontractors, other projects
4. Managing Risk
   - o identifying and confronting potential problems

Managing versus Leading

- Managing is concerned with the quantitative aspects of SPM:
  - Planning, organizing and estimating
  - measuring and controlling
  - quantitative risk management
- Leading is concerned with the qualitative aspects of SPM:
  - communicating and coordinating
  - inspiring and maintaining morale
  - qualitative risk management

An effective project manager is both a manager and a leader

You cannot manage men into battle. You manage things; you lead people.

Grace Murray Hopper

“First Lady of Software”

Grace Brewster Murray Hopper; December 9, 1906 – January 1, 1992)

Project Success Criteria

- The primary goal of software engineering is to develop and modify software so that:
  - the product is delivered on time & within budget
  - the product satisfies technical requirements, user needs, and customer expectations
  - the product is easy to modify and maintain
  - development milestones are achieved on time & within budget
  - staff morale is high throughout project
  - work instills pride in the developers

Q1: What are your personal success criteria?
Q2: What are most organizations' main success criteria?
Q3: What are most customers' main success criteria?

Project Manager's Success Criteria

A project manager's success criteria include, or should include:

- delivery of an acceptable product on time and within budget
    - within the limits imposed by *project constraints**
- maintaining good relations with customers, suppliers, managers, and other organizational units
- maintaining a motivated project team
- advancing the career of each project member
- advancing his or her career
- Other criteria?

*A *constraint* is an externally imposed limitation

Why Are Software Projects Difficult?

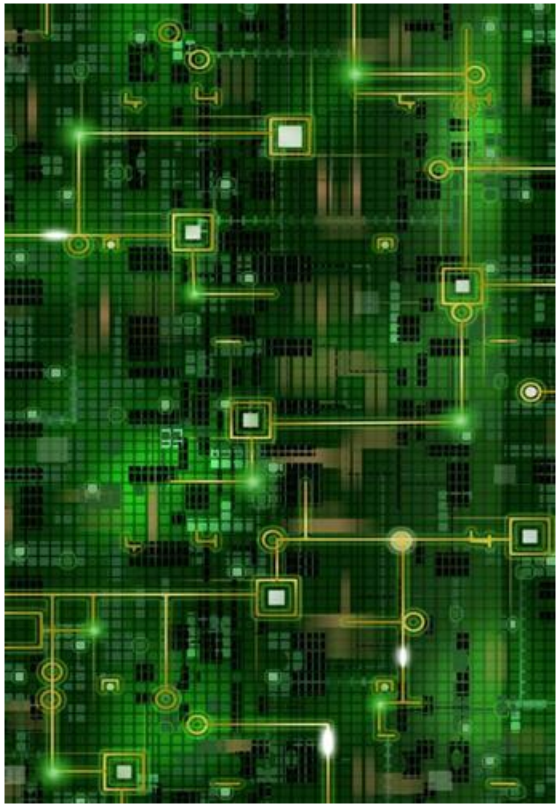
- According to Fred Brooks* software projects are difficult because of accidental and essential difficulties
  - accidental difficulties are caused by the current state of our understanding
    - of methods, tools, and techniques
    - of the underlying technology base
  - essential difficulties are caused by the inherent nature of software
    - invisibility - lack of physical properties
    - complexity
    - conformity
    - changeability

* *The Mythical Man-Month* by Fred Brooks, Addison Wesley, 1995

Example: Why IT Software-Intensive Projects Are Hard to Manage and Lead?

complexity: Cyber Physical Systems - Components

- Due to interaction of components, number of possible states grows.
- Hardware is complex but we usually know that for a known input, what to expect output should be given we are only looking at the hardware
- For its size, software is very complex compared to even hardware.



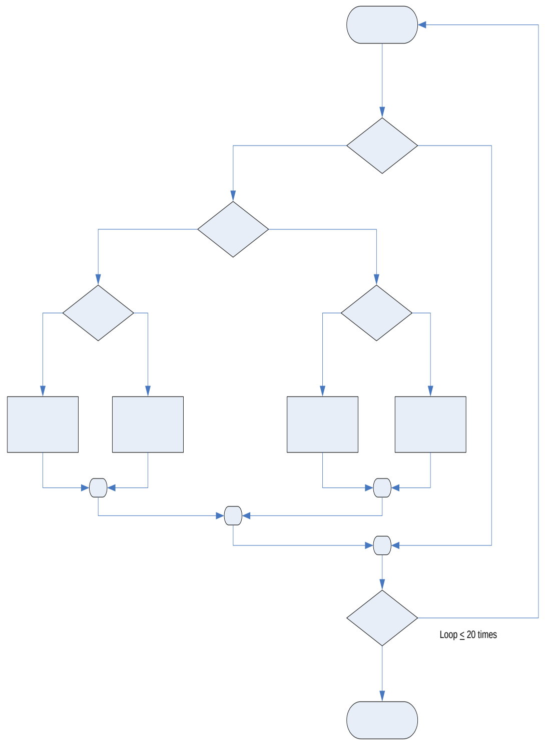
Source SEI

Complexity - Example

The flowchart might correspond to a 100 LOC module with a single loop that may be executed no more than 20 times.

There are approximately 10^{14} possible paths that may be executed!

For any but the smallest programs, complete path coverage for defect detection is impractical.



Lehman Laws:

1. The Law of Continuing Change – programs must change to be useful
2. The Law of Increasing Complexity – programs that change become more complex

Adapted from Pressman, R.S., *Software Engineering: A Practitioner's Approach, Third Edition*, McGraw Hil, 1992

Additional Difficulties

- Additional reasons software projects are difficult are:
  - intellect-intensive, small team-oriented nature of the work
  - externally imposed constraints
- Software is developed by:
  - teams of individuals
  - engaged in closely coordinated intellectual work activities
  - to produce various written work products

An Observation

- As Michael Jackson has observed, the entire description of a software system or product is usually too complex for the entire description to be written directly in a programming language, so we must prepare different descriptions at different levels of abstraction, and for different purposes*.
- Also, note that each of the work products listed on the following slide is a document
  - Software developers and software project managers produce physical artifacts (e.g. code) and documents, which may exist in printed or electronic form.

* M. Jackson, "Descriptions in Software Development," *Lecture Notes in Computer Science*, Springer Verlag GmbH, Volume 2460, 2002.

Some Work Products of Software Projects

Document	**Content of the document**
Project plan	Roadmap for conducting the project
Status reports	Visibility of progress, cost, schedule, and quality
Memos and meeting minutes	Issues, problems, recommendations, resolutions
e-mail messages	On-going communications
Operational requirements	User needs, desires, and expectations
Technical specification	Product features and quality attributes
Architectural design documents	Components and interfaces – many different views
Detailed design specification	Algorithms, data structures, and interface details of individual modules
Source code	Product implementation
Test plan	Product validation criteria and test scenarios
Reference manual	Product encyclopedia
Help messages	Guidance for users
Installation instructions	Guidance for operators
Release notes	Known issues; hints and guidelines
Maintenance guide	Guidance for maintainers

Note

- Note that the work products generated by software engineers exist in graphical, iconic, and textual forms
  - software engineers do not design or fabricate artifacts made of physical materials
  - our work products are generated from our thought processes

A Simile

- A team that writes software together is like a team that writes a book together
  - the team may pursue a “plan-driven” approach
  - or an “agile” approach

Plan-Driven Development

- Plan-driven development involves:
  - o defining the product requirements
  - o developing an architectural structure for the product
  - o allocating the work among teams
  - o measuring progress and making corrections
  - o refining and revising the work products as necessary
    - preferably in an iterative manner

Agile Development

- When pursuing an agile approach, the team members must:
  - develop an understanding of the nature of the desired product to be delivered,
  - develop continuous, ongoing relationship with a knowledgeable user representative
  - establish a shared design metaphor,
  - adopting a version of agile development, and
  - determine the constraints on schedule, budget, resources, and technology that must be observed.

Most successful software projects incorporate aspects of both plan-driven and agility

Additional Difficulties

- Additional reasons software projects are difficult are:
    - intellect-intensive nature of the work
    - externally imposed constraints

Engineering Constraints

- Engineering is concerned with applying science and technology to develop products for use by society within the *constraints* of:
  - product requirements: features and quality attributes
  - project scope: work activities to be accomplished
  - time: scheduled dates for progress
  - resources: assets available to conduct a project
  - budget: money used to acquire resources

Additional Constraints

- Additional limitations imposed on software projects include:
  - platform technology: software tools and hardware/software base
  - domain technology: the realm of the user domain
  - process standards: ways of conducting work activities
  - scientific knowledge: solution methods
  - business considerations: profit, stability, growth
  - mission needs: safety and security of citizens
  - ethical considerations: serving best interests of humans and society
- Others?

Useful Constraints and Inhibiting Constraints

- Useful constraints provide guidance:
  - for example, well-defined requirements are the basis for planning, estimating, and establishing success criteria
- Inhibiting constraints inhibit the ability to achieve success criteria:
  - for example, excessive schedule pressure may inhibit the ability to delivery a product of high quality

The Challenges of Software Project Management

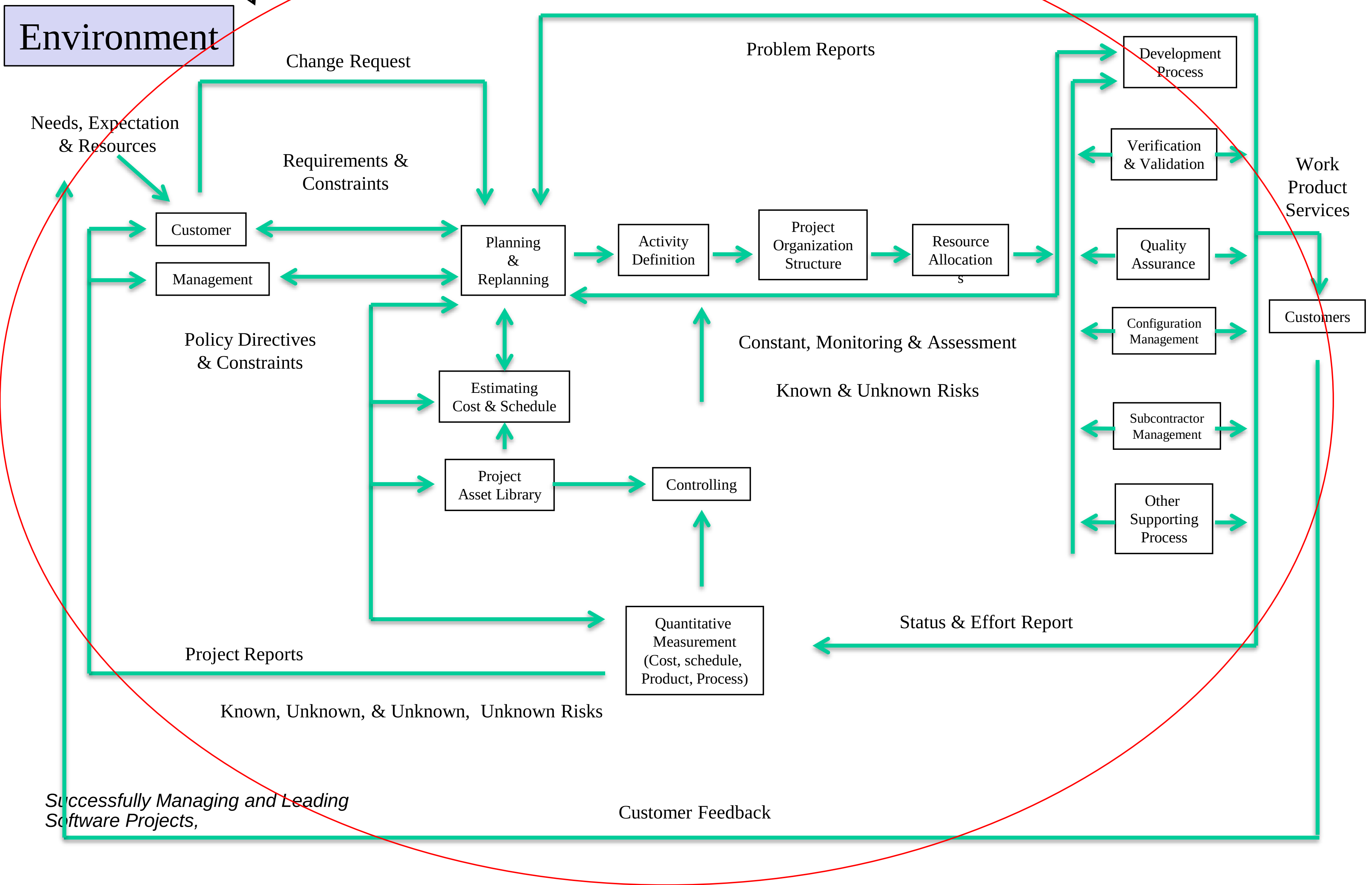
- Some of the most difficult problems you will encounter in managing software projects arise from establishing and maintaining a balance among the constraints on project scope, budget, resources, technology, and the scheduled delivery date:
  - scope: the work to be done,
  - budget: the money to acquire resources,
  - resources: the assets available to do the job,
  - technology: methods and tools to be used, and
  - delivery date: the date on which the system must be ready for delivery.

Topics

- Fundamental Shifts in Software Management & Engineering
- Why Managing and Leading Software Projects Is Difficult
- The Nature of Project Constraints
- A Workflow Model for Managing Software Projects
- Organizational Structures for Software Projects
- Organizing the Project Team
- Maintaining the Project Vision and the Product Vision
- Frameworks, Standards, and Guidelines

You are Here in the Front End of the Project – Note: Some of Work May Be Done Before Project is Officially Formed

A Workflow Model for Software Projects



Some Elements of the Model

- Customers and managers
- Requirements
- Directives and constraints
- Planning and re-planning
- Estimating
- Identifying the work activities and work assignments
- Conducting the work activities
- Measuring and reporting status
- Controlling the project
- Retaining status data
- Handling change requests and problem reports
- Supporting processes

Some Supporting Processes for Software Projects

Supporting Process	**Purpose**
Configuration Management	Change control; baseline management; product audits; Status Reporting
Verification	Determining the degree to which work products satisfy the conditions placed on them by other work products and work processes
Validation	Determining the degree of fitness of work products for their intended use in their intended environments
Quality Assurance	1. Assuring conformance of work processes and work products to policies, plans, and procedures 2. Engineering-in quality over the life of the product
Documentation	Preparation and updating of intermediate and deliverable work products
Developer Training	Maintaining adequate and appropriate skills
User and Operator Training	Imparting skills needed to effectively use and operate systems

Eight Supporting Processes in ISO & IEEE Standards 12207

- Documentation
- Configuration management
- Quality assurance
- Verification
- Validation
- Joint review
- Audit
- Problem resolution

A Note on Terminology

- In many organizations the term “software quality assurance (SQA)” is used to mean independent testing
- In the 12207 standards quality assurance is concerned with:

  “providing adequate assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans.”
- Testing is in the realm of Verification and Validation
  - Independent testing should not be termed “QA” or “SQA”

CMMI-DEV-v1.3 uses terminology similar to 12207
more later

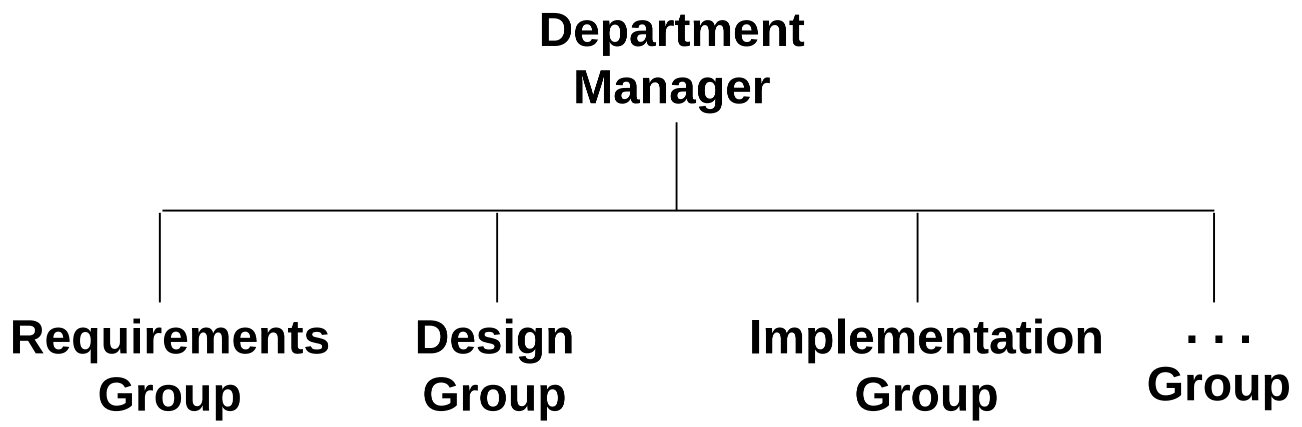
Chapter 1 Topics

- Fundamental Shifts in Software Management & Engineering
- Why Managing and Leading Software Projects Is Difficult
- The Nature of Project Constraints
- A Workflow Model for Managing Software Projects
- *Organizational Structures for Software Projects*
- Organizing the Project Team
- Maintaining the Project Vision and the Product Vision
- Frameworks, Standards, and Guidelines

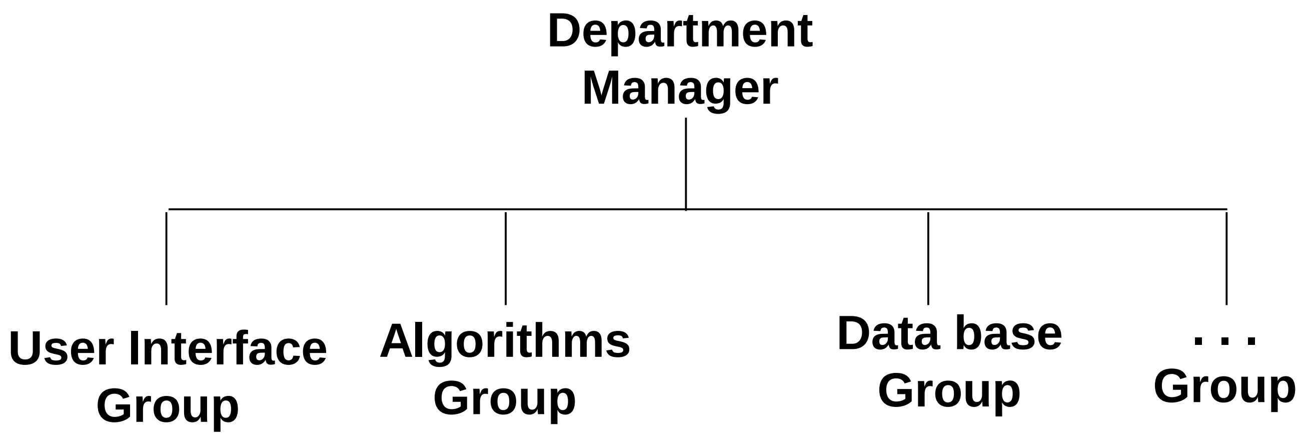
Organizational Structures

- Organizations that conduct engineering projects, including software projects, are typically organized in one of four ways:
  - functional structure,
  - project structure,
  - matrix structure, or
  - hybrid structure.

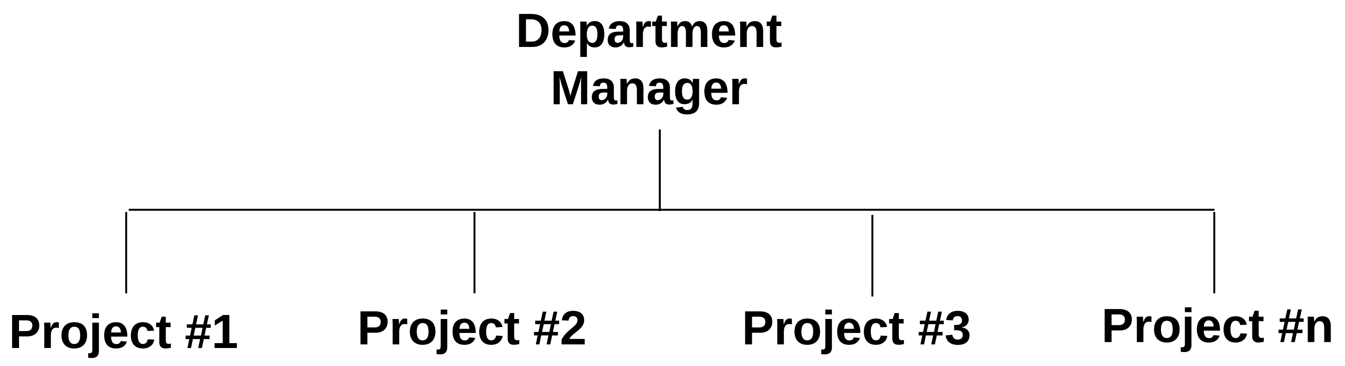
A Process-Structured Functional Organization



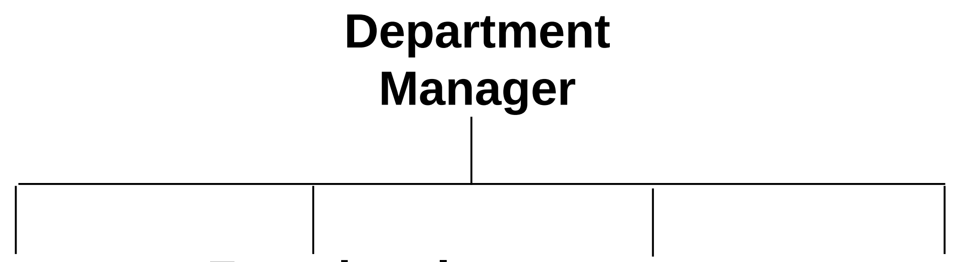
A Product-Structured Functional Organization



A Project-Structured Organization

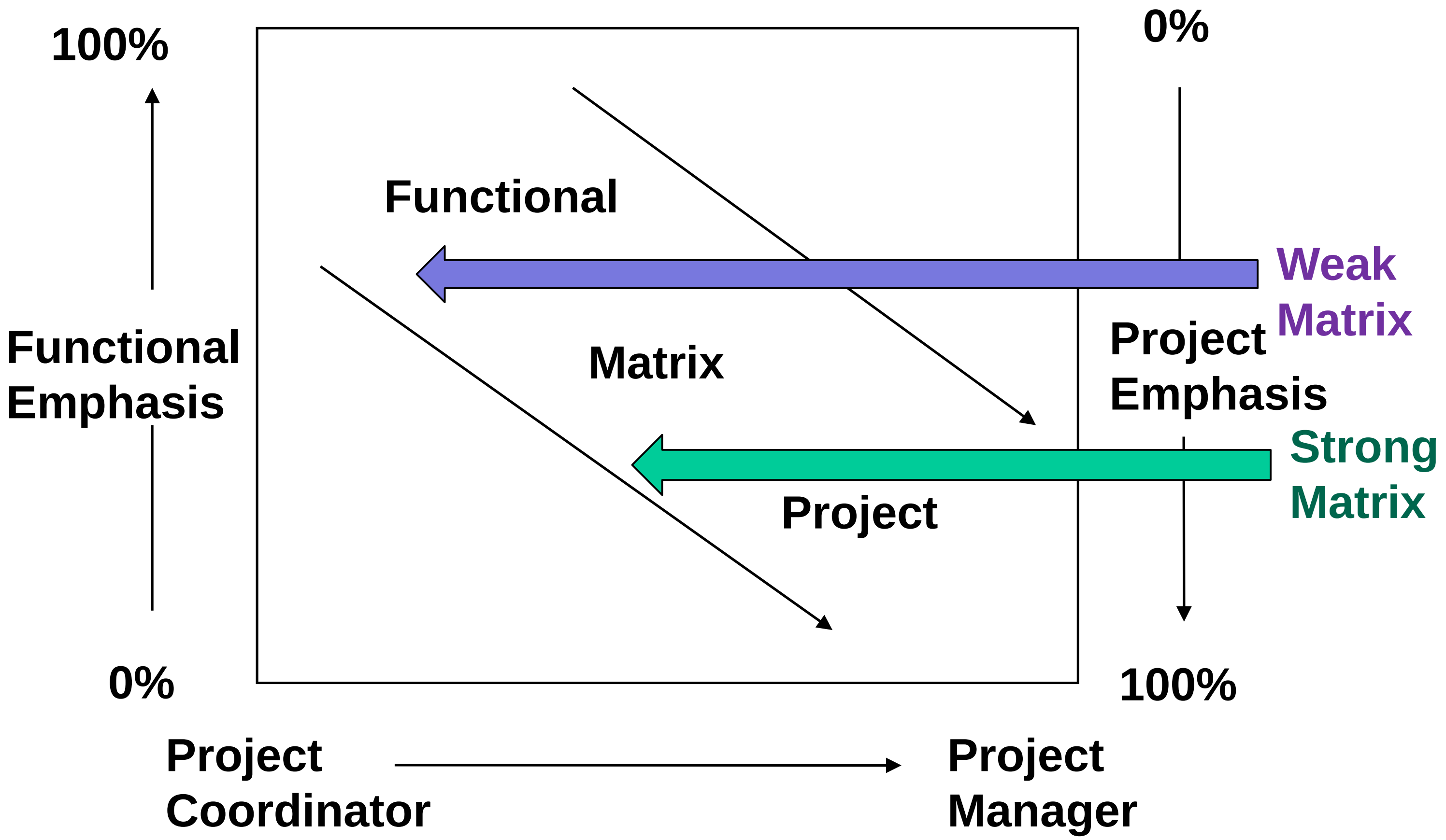


A Matrix-Structured Organization

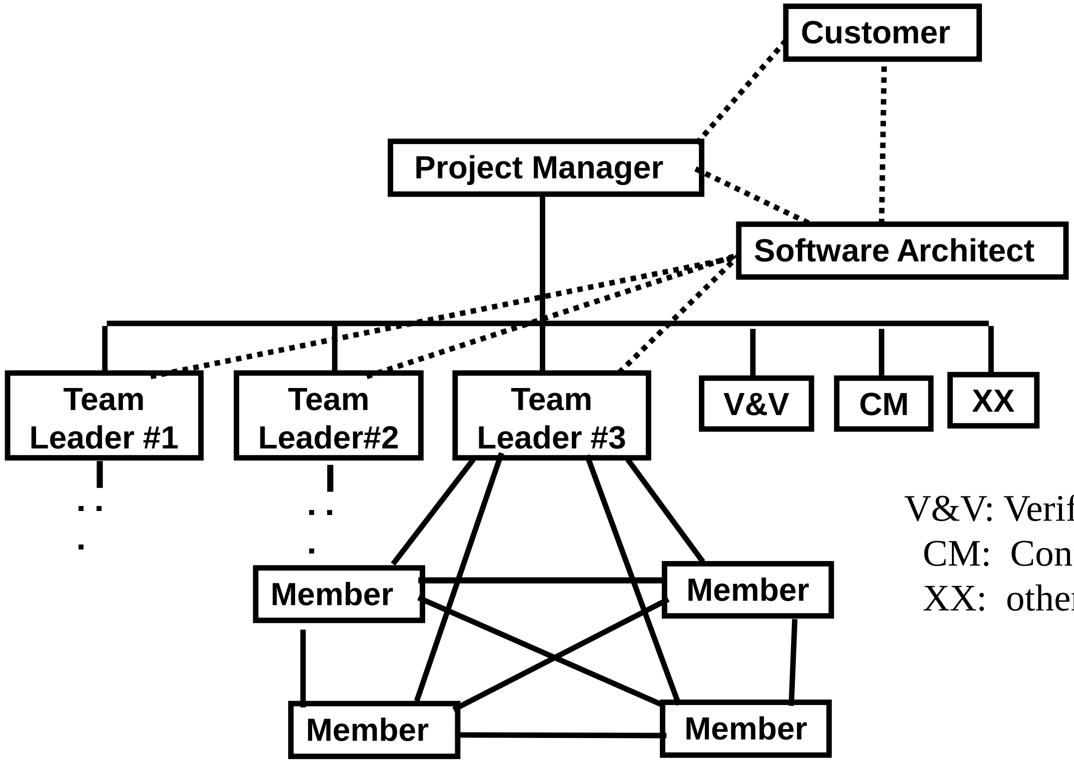


	Functional Manager #1	Functional Manager #2	Functional Manager #3	Functional Manager #4
Project Manager #1	2		5	3
Project Manager #2	3	4	7	
Project Manager #3	6	8	3	4
Project Manager #m	1	2		6

The Organizational Continuum



An Organizational Model for Software Projects



V&V: Verification and Validation
CM: Configuration Management
XX: other supporting processes

Each team has 2 to 5 members plus a team leader

A Note

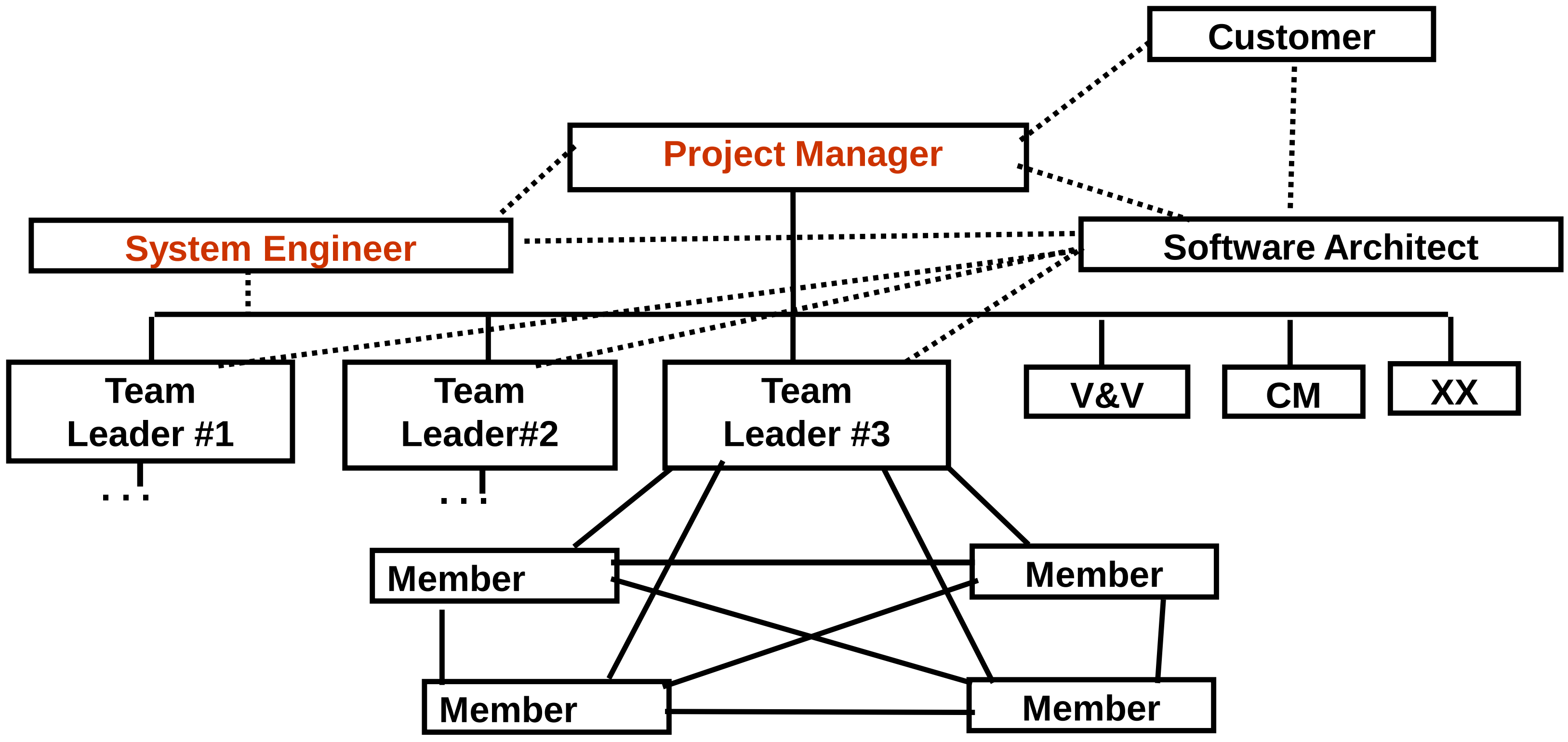
- A complex system is composed of:
  - o hardware (computers and others)
  - o software (newly developed and reused, COTS)
  - o people (operators, maintainers)
- A software program may be one of a collection of projects
  - o under the technical direction of a system engineering team

The System Engineering Team

- The responsibilities of systems engineers include:
  - defining operational requirements,
  - specifying system requirements,
  - developing the system design,
  - allocating system requirements to system components,
  - integrating the system components as they become available,
  - verifying that the system to be delivered is correct, complete, and consistent with respect to its technical specifications, and
  - validating operation of the system with its intended users in its intended operational environment.

for “software only” projects the people who perform these functions are termed “software system engineers”

An Organizational Structure for Software Projects



V&V:	Verification and Validation
CM:	Configuration Management
XX:	other supporting processes

Each team has 2 to 5 members plus a team leader

Chapter 1 Topics

- Fundamental Shifts in Software Management & Engineering
- Why Managing and Leading Software Projects Is Difficult
- The Nature of Project Constraints
- A Workflow Model for Managing Software Projects
- Organizational Structures for Software Projects
- Organizing the Project Team
- *Maintaining the Project Vision and the Product Vision*
- Frameworks, Standards, and Guidelines

Maintaining the Project and Product Visions

- The project manager is the keeper of the process vision
  - o which is documented in the project plan
    - and is updated as the project evolves
- The system engineer/software architect is the keeper of the product vision,
  - o which is documented in the requirements and architectural design specifications
    - and is updated as the product evolves

Another Simile

- The project manager is like a movie producer and the software architect to a movie director.
    - The producer (project manager) has overall responsibility for schedules, budgets, resources, customer relations, and delivery of a satisfactory product on time and within budget.
- The director (software architect) is responsible for the content of the product.

Producer and director must work together to maintain and constantly communicate the process vision and the product vision to the cast of developers and supporting personnel as well as other project stakeholders

Chapter 1 Topics

- Fundamental Shifts in Software Management & Engineering
- Why Managing and Leading Software Projects Is Difficult
- The Nature of Project Constraints
- A Workflow Model for Managing Software Projects
- Organizational Structures for Software Projects
- Organizing the Project Team
- Maintaining the Project Vision and the Product Vision
- *Frameworks, Standards, and Guidelines*

Frameworks, Standards, and Guidelines (1)

- A *process framework* is a generic process model that can be tailored and adapted to fit the needs of particular projects and organizations.
- An *engineering standard* is a codification of methods, practices, and procedures that is usually developed and endorsed by a professional society or independent agency.
- *Guidelines* are pragmatic statements of practices that have been found to be effective in many practical situations.

Frameworks, Standards, and Guidelines (2)

- Some well known frameworks, standards, and guidelines for software engineering and the associated URLs are:
  - Capability Maturity Model® Integration for development (CMMI-DEV-v1.3)
  - 12207-2017 - ISO/IEC/IEEE Standard for Systems and Software Engineering - Software Life Cycle Processes
  - 16326-2009 - ISO/IEC/IEEE International Standard Systems and Software Engineering--Life Cycle Processes--Project Management
  - Project Management Body of Knowledge (PMBOK®) – Sixth Edition[www.pmibookstore.org]

Terminal Learning Objectives for Module 1 (1)

- A project is a coordinated set of activities that occur within a specific timeframe to achieve specific objectives
- The primary activities of software project management are planning and estimating; measuring and controlling; leading, communicating, and coordinating; and managing risk
- Software projects are inherently difficult because software is complex, changeable, conformable, and invisible
- Software projects are conducted by teams of individuals who engage in intellect-intensive teamwork
- Project constraints are limitations imposed by external agents on some or all of the operational domain, operational requirements, product requirements, project scope, budget, resources, completion date, and platform technology
- A workflow model indicates the work activities and the flow of work products among work activities in a software project

Terminal Learning Objectives for Module 1 (2)

- The entire description of a software system or product is usually too complex for the entire description to be written directly in a programming language, so we must prepare different descriptions at different levels of abstraction, and for different purposes
- Organizations that conduct software projects use functional, project, weak matrix, and strong matrix structures
- Software projects organized in a hierarchical manner provide well-defined work activities, roles, authorities, and responsibilities at each level in the hierarchy; hierarchies can expand and shrink to fit the needs of each project
- Requirements must be allocated and the design structured so that the work of each small team can proceed concurrently with the work of other teams

Terminal Learning Objectives for Module 1 (3)

- The project manager maintains the project vision, as documented in the project plan, and the software architect maintains the product goals, as documented in the requirements and architectural design
- A software process framework is a generic process model that can be tailored and adapted to fit the needs of particular projects and organizations.
- A software engineering standard is a codification of methods, practices, and procedures, usually developed and endorsed by a professional society or independent agency.
- Guidelines are pragmatic statements of practices that have been found to be effective in many practical situations.

Questions

QUALITY
PRODUCTIVITY
JOB KNOWLEDGE
SKILLS
RESPONSIBILITY
ADAPTABILITY
RESOURCEFULNESS

leadership