

COURSE INTRODUCTION AND OVERVIEW BRIEF RECAP OF SOFTWARE ENGINEERING FUNDAMENTALS SOFTWARE DESIGN CONCEPTS

Dr. Tony D. Barber / Prof. Sahana Kini
Fall, 2022



TODAY'S CLASS OBJECTIVES AND OUTCOME

- Identify the objectives and content of the course
- Define the skills acquired during the course
- Identify topics, assessments, and schedule for the ENPM613 class
- Identify the type of activities for this course
- Explain prerequisites and expectations
- Identify university and course rules
- Identify supporting resources and develop a strategy for achieving those goals
- Get familiar with your classmates
- Answer fundamental questions about software engineering
- Define what is ethics in software engineering
- Apply critical thinking principles
- Introduce main software design concepts



OUTLINE

- Introductions
 - Instructor
 - Students
 - ENPM 613 course
- Critical thinking
- Ethics in software engineering
- Brief recap of software engineering
- Software design concepts
- This week's assignments



OUTLINE

- Introductions



- Instructor

- Students
 - ENPM 613 course
 - Ethics in software engineering
 - Brief recap of software engineering
 - Software design concepts
 - Requirements analysis
 - This week's assignments



YOUR INSTRUCTOR: DR. TONY D. BARBER



BS in Engineering (Computer Engineering)

- University of South Carolina – Columbia (USC)



MS in Systems Engineering & Engineering Management

- George Washington University (GWU)



MBA (Management Consulting)

- American University (AU)



Post Grad Cert in Org Development

- Georgetown University (GT)



Doctorate in Engineering (Systems Engineering)

- George Washington University (GWU)



Post Grad Cert in Data Analytics

- Cornell University



Post Grad Cert in Python Development

- Columbia University

Professional Certs: PMI PMP, PMI RMP, PMI PBA, INCOSE CSEP, IIBA CBAP, PEM, ScrumMaster, etc.

▪ ***Current Work:***



US Army Office of the Chief Information Officer

- Principal Technical Advisor
- System Branch Chief
- Computer Engineer



University of Maryland School of Engineering

- Lecturer / Professor



TD Watts Consulting / TD Strategy

- President & Chief Executive Officer (CEO)



Acquired Data Solutions

- Corporate Advisor / Chief Growth Officer (CGO)



NucoreVision Corporation

- Chief Growth Officer (CGO)



InfoSymmetry Technology & NPO

- President and CEO (Tech)
- Board Chair (NPO)



AccuStrata LLC

- Corporate Advisor

Previous Work:

Booz Allen Hamilton, BAE Systems, General Dynamics, Raytheon, Northrop Grumman, BearingPoint, Deloitte Consulting, US Census Bureau Decennial Directorate



YOUR INSTRUCTOR: PROFESSOR SAHANA KINI

Education



Bachelor of Engineering (BE) (Computer Science)

- Visvesvaraya Technological University



Master of Science, Computer Science (Computer Systems)

- Georgia Institute of Technology (Georgia Tech)

Professional Certs: Scrum Master Certified (SMC)

Current Work



KACE Company

- Senior Software Engineer



University of Maryland School of Engineering

- Lecturer / Professor

Previous Work



MVM, Inc.

- Senior Software Developer



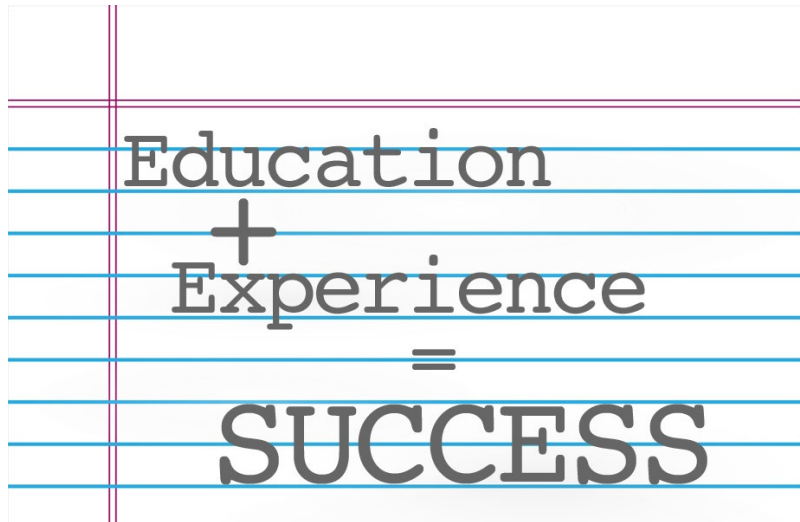
Harmonia Holdings Group, LLC

- Software Developer



Hewlett-Packard

- Software Engineer



OUTLINE

- Introductions

- Instructor



- Students

- ENPM 613 course

- Ethics in software engineering
- Brief recap of software engineering
 - Software design concepts
 - Requirements analysis
- This week's assignments



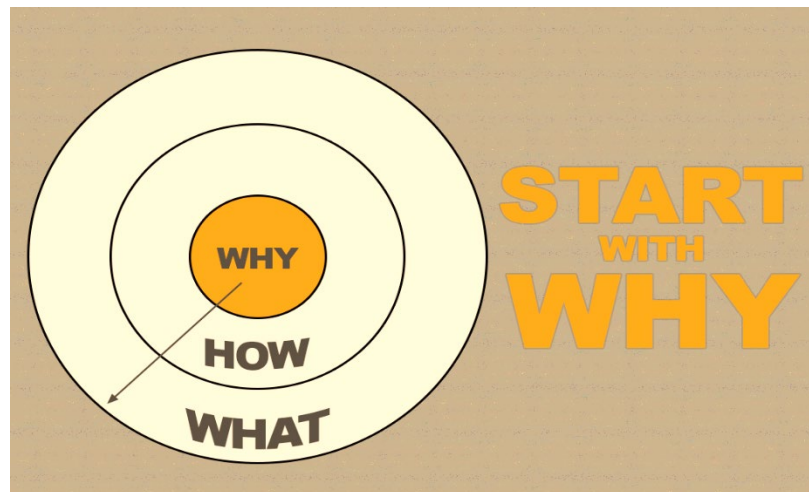
STUDENTS INTRODUCTION

- **WHY** are you taking this class
- **WHAT**
 - Is your objective for the class
 - Do you expect the outcome of this class to be for you
 - Is your definition of “success”
- **HOW** will you achieve your objective
- **Answer these questions in the ELMS *Discussion* forum *Students Introduction***
- Throughout the semester and at the end of the class
 - Ask your self these questions again and again
 - Check how close you are to achieving your goals
 - What does it take to get closer to your goals
 - Frequent retrospection and introspection -> frequent course correction



“START WITH WHY”

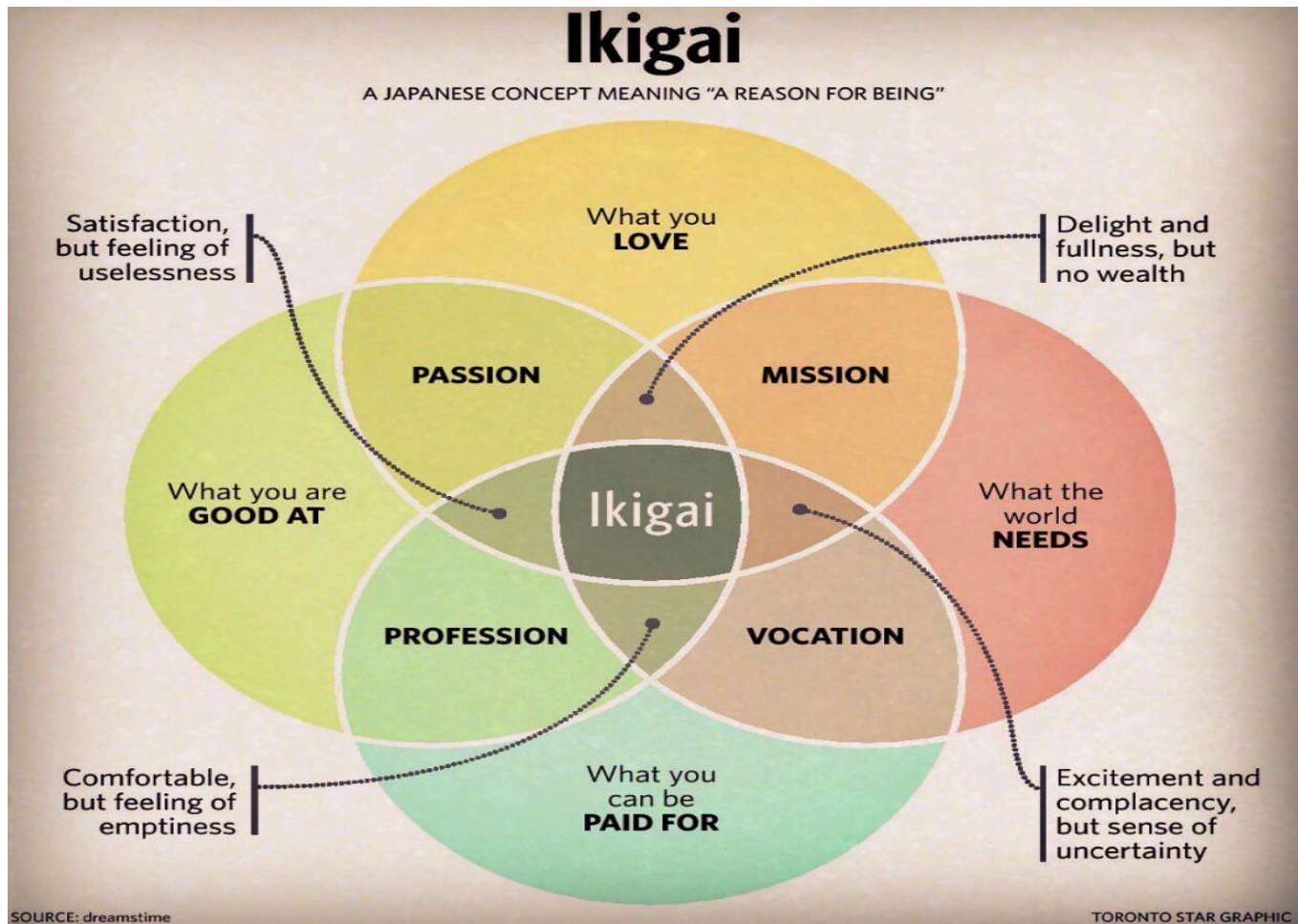
- *Start with Why: How Great Leaders Inspire Everyone to Take Action*, by Simon Sinek
- https://www.ted.com/talks/simon_sinek_how_great_leaders_inspire_action (~18 mins)



- “The 5 WHYs”, **root cause analysis technique** - by Sakichi Toyoda, Toyota Motor Corporation)



WHAT IS **YOUR** REASON FOR BEING?




OUTLINE

- Introductions

- Instructor

- Students

 ENPM 613 course

- Ethics in software engineering
- Brief recap of software engineering
 - Software design concepts
 - Requirements analysis
- This week's assignments



ENPM 613 IN THE MAGE

- Objective of the ENPM program (“WHY”): prepare students for the software engineering profession
- Core ENPM in Software Engineering Courses:
 - ENPM 611 *Software Engineering* – **prerequisite for 613**
 - ENPM 612 – *Software Requirements* recommended before ENPM 613
 - **ENPM 613 – *Software Design and Implementation***
 - ENPM 614 – *Software Testing and Maintenance*



ENPM 613 COURSE INTRODUCTION

- **WHY:** Teach critical design skills that will help you in your professional lives as software engineers
- **WHAT** (outcome): Be able to effectively select and use *principles, methods, techniques, notations, tools, practices, and processes*, to develop, document, verify and validate a “good-enough” **architecture** and **detailed design** of a complex and high-quality **software product**.
- **HOW:** Interactive classes and a variety of work assignments and assessments



CRITICAL DESIGN SKILLS

- Technical skills
- “Soft” skills

IN JUST FIVE YEARS, 35 PERCENT OF THE SKILLS DEEMED ESSENTIAL TODAY WILL CHANGE, ACCORDING TO THE WORLD ECONOMIC FORUM.



TECHNICAL SKILLS

- What technical skills are needed for designing software?



TECHNICAL SKILLS

- Modeling and abstraction
- Analysis
 - Trade-off analysis
- Use of design methods, principles, practices
- Proper use of a modeling language
- Problem solving
- Research
- Basic Project Management
- ...



“SOFT” SKILLS

- What are “soft” skills?
- What “soft” skills are needed for designing software?



“SOFT” SKILLS

- What are “soft” skills?
 - More intangible and non-technical abilities
 - Sometimes referred to as **transferable skills** or **professional skills**



“SOFT” SKILLS FOR DESIGNING SOFTWARE

- **Critical thinking** – peer/team reviews; identify problems; understand different perspectives, evaluate info sources; be aware of and document your assumptions
- **Communication**
 - Technical writing – develop project documentation
 - Verbal communication – class presentations
 - **Provide and receive feedback** – review, class presentations feedback; provide feedback to instructor through online discussion and course evaluation
- **Teamwork and collaboration** – class team project (“safe” experimentation laboratory)
- **Ethics** – practice academic integrity in class and understand software engineering/computing professionals code of ethics



“SOFT” SKILLS

- **Independent learning** – not everything you need to know can be delivered in class as lecture; there is an amount of independent/outside class studying required
- **Leadership** – is a mindset; be a leader in your class project
- **Negotiation and conflict resolution** – within the class project (e.g., re-scope), NOT with your grades
- **Responsibility** – own your work and results
- **Time management and ability to work under pressure** – manage multiple assignments; coordinate time with peers for the project



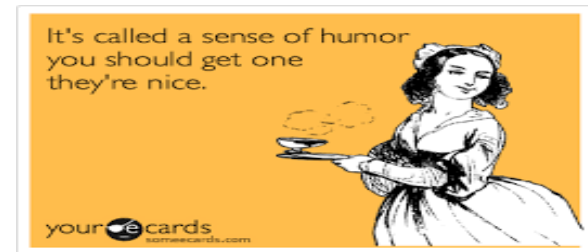
“SOFT” SKILLS

- **Resilience/”grit”** – work will be challenging; if you don’t always reach your goals (aka “fail”), regroup and recover, learn your lessons and become stronger
- **Respect, consideration**, environment that fosters goals’ achievement, growth, development, learning, effectiveness, productivity – **practiced throughout class, in all interactions with peers and instructor(s)**



destress.com

- **Relax**
- **Have a sense of humor**



<https://www.driveactiondigital.com/a-sense-of-humor-and-social-media>



COURSE TOPICS

- Software requirements analysis
- Software architecture and detailed design
 - Methods for designing and evaluation
 - User centric design and engineering design
 - Documentation: views, notations (UML)
 - Styles, patterns, tactics
 - Process, Artifacts, Role
 - Design and implementation
 - Design in new product development and maintenance/evolution
- Software implementation considerations



PRE-REQUISITES

- Software engineering (ENPM611 or equivalent)
- Object Oriented software development/ programming

This is NOT a programming course !



PRE-CLASS QUIZ

- Objective: quick screening of your knowledge
 - Help you identify background gaps that you need to close (and even whether you should take the class this semester)
 - Help me assess the knowledge level of the class
- In ELMS
- Not graded, but **mandatory**



CLASS ACTIVITIES

- Lectures
- Hands-on exercises
 - Including reviews and discussions
- Student presentations
 - Software development project
- Breaks 😊
- Outside class: individual study, quizzes, project



CLASS ACTIVITY TYPES

The Cone of Learning

sparkinsight.com

*I see and I forget.
I hear and I remember.
I do and I understand.*
— Confucius



After 2 weeks,
we tend to remember ...

- 10% of what we READ
- 20% of what we HEAR
- 30% of what we SEE
- 50% of what we SEE & HEAR
- 70% of what we SAY
- 90% of what we SAY & DO

P
a
s
s
i
v
e

A
c
t
i
v
e

Source: Edgar Dale (1969)



ASSESSMENTS

- Demonstrate that students can:
 - Identify the characteristics of a complex, high quality software product
 - Identify and solve problems
 - Analyze requirements
 - Select appropriate design practices
 - Apply principles and selected practices to
 - create a “good-enough” architecture
 - document the architecture
 - assess the architecture
 - create a good detailed design
 - document the detailed design
 - assess the detailed design
 - implement part of the design
 - Work as a team
 - Communicate, collaborate, support each other



TOPICS, ASSESSMENTS, ASSIGNMENTS

- See syllabus for list of assessments, points, weights, and schedule



QUIZZES

- Take home
- Quick and frequent feedback
- Low stakes



SOFTWARE DEVELOPMENT PROJECTS

- Team project – everyone needs to contribute to all activities and deliverables
- Team size: 4-6 students
- Teams will be formed by the students
- Several project phases and deliverables
- Apply what you learn in class
- Implementation - your choice of language/technology
- Project submissions are strictly due on their due date
- Peer review – each team's deliverables will be reviewed by another team
 - Review is also graded!

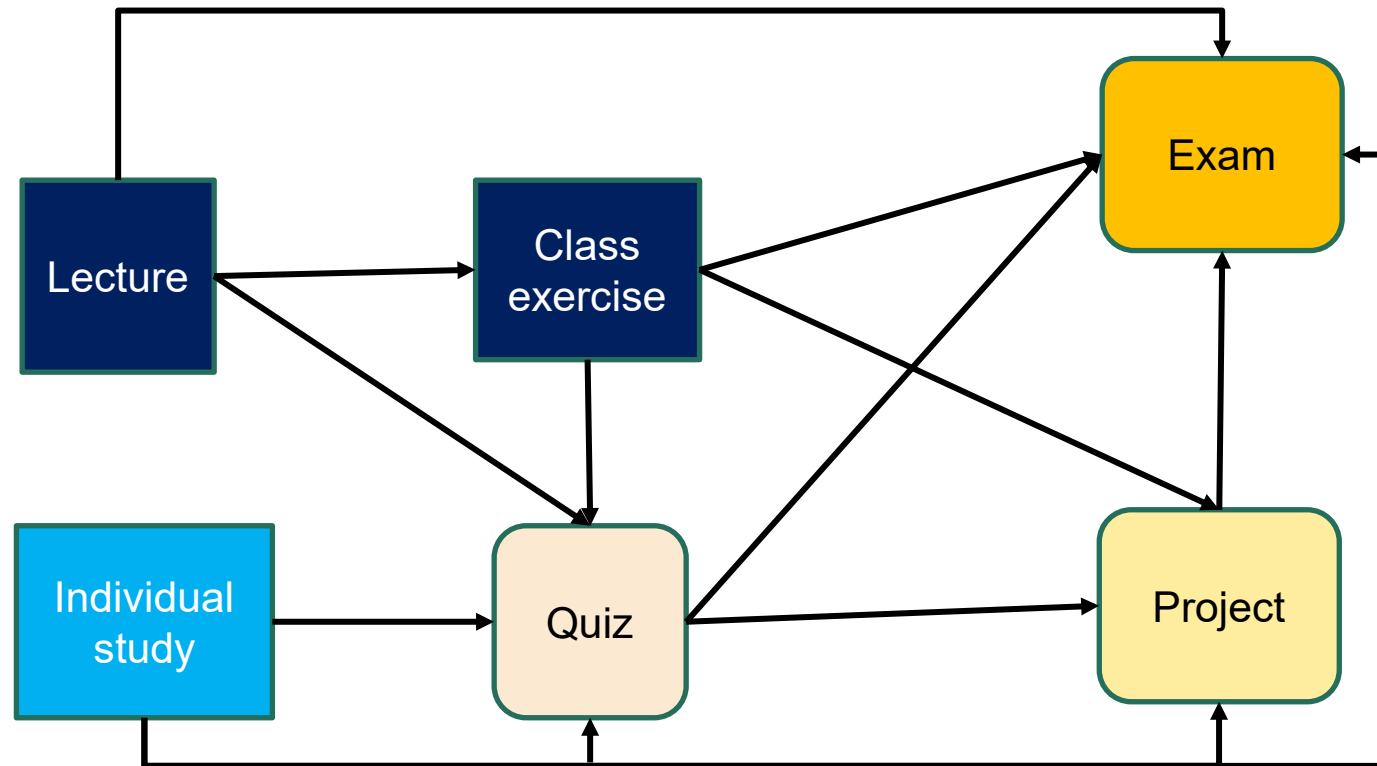


EXAMS

- Closed book, notes, neighbor
 - Midterm
 - For date see schedule in ELMS
 - Final exam
 - For date see schedule in ELMS
 - Comprehensive (everything covered)



COURSE ACTIVITIES AND ASSESSMENTS



TYPICAL WEEK*

***Note:** exceptions will occur
(1st week, exams, break, or
major project presentations)

Week/topic “n” class preparation

- Read material
- Watch lecture video
- Take quiz
- Answer questions
- Do individual exercise

4:00 pm

*Where n = Requirements Analysis, UI design,
Architecture Design, Architecture Documentation,
Detailed Design, Implementation*

Project

Perform project tasks and
produce artifacts/deliverables
(topic “n-1”)

Project

Perform project tasks and
produce artifacts/deliverables
(topic “n”)

Wednesday

Project Drafts or
Deliverables submission

Thursday

Class

- Discuss main lecture points
- Answer questions
- Group exercise OR
- Project presentation

Friday

Project (after deliverables for activity “n-1”):

- Team-to-team review
- Update Individual contribution
- Update Project Management

Sunday



SUBMISSIONS

- Submissions for project(s) deliverables will be made online, using ELMS (Canvas), by the date specified in this system.
- Late submissions are **not** accepted and will not be graded
 - Exceptions might be **occasionally** granted, **if and only if** permission is obtained from the instructor **before** the submission deadline



GRADING

- Grading scale – see syllabus
- Absolute grading
 - Your grades depend on your performance alone and not that of the rest of the class
- You earn and own your grade
 - Your grade is based on your **results**, not anything else, e.g., not on your amount of effort, and not on the grade you need for a certain GPA
- **Passing grade is C- or higher, i.e., $\geq 70\%$**
- If your midterm exam grade is lower than C- (70%), you will need to immediately discuss causes and a remediation plan with your instructor



ACADEMIC INTEGRITY POLICY

- You are
 - Encouraged to collaborate and discuss among yourselves
 - Not allowed to copy code and solutions from other students
- If you use code available on-line, acknowledge it
- Mid-terms and finals will be closed-book, closed notes, closed-neighbor
- Standard policy for University academic honesty policy applies (see Syllabus and <https://www.president.umd.edu/sites/president.umd.edu/files/documents/policies/III-100A.pdf>)
- Pledge of honor: *“I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.”* – on assignments and exams



COMMUNICATIONS

- Read regularly class **ELMS Announcements** and your **email**
- Do not hesitate to communicate with your instructor, TA/ grader, and with fellow students
- Use email to communicate with the instructor and TA/grader, **through ELMS (Canvas) email ONLY**
- Office hours – see syllabus
- **ELMS Discussions** – also a great way to post questions/answers, thoughts, and ideas that are relevant not only to you, but also to other students
- Be explicit, never make assumptions
- Think forward, anticipate risks and manage them
- Do not let small problems become big problems

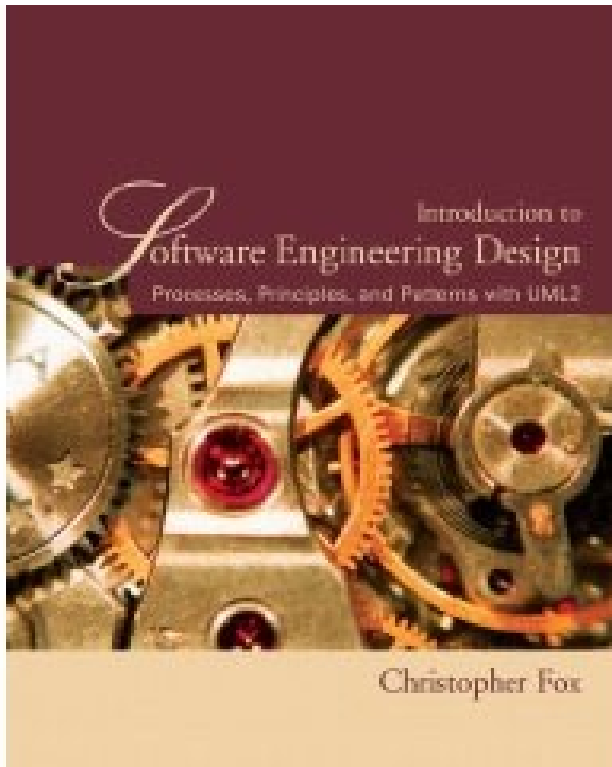


FOR MORE INFORMATION AND DETAILS...

- ELMS (Canvas) online course support:
 - E.g., **Syllabus** – there is a **Quiz on Syllabus in ELMS**



COURSE TEXTBOOK



*Introduction to Software
Engineering Design
Processes, Principles and Patterns with UML 2*

By Christopher Fox

Addison Wesley

Book Web resources:

<https://users.cs.jmu.edu/foxcj/Public/ISED/index.htm>

On reserve at the UMD Library, together with other material



OTHER RESOURCES - BOOKS

Software Engineering

- *Software Engineering: Theory and Practice* by Shari Lawrence Pfleeger, Joanne M. Atlee
- *Software Engineering* by Ian Sommerville

Architecture and Design

- *Software Architecture in Practice*, by Len Bass and Paul Clements
- *Documenting Software Architectures: Views and Beyond* by Paul Clements and Felix Bachmann
- *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives* by Nick Rozanski and Eóin Woods
- *Evaluating Software Architectures: Methods and Case Studies* by Paul Clements and Rick Kazman
- *Software Design From Programming to Architecture* by Eric Braude - see Wiley [book site](#)
- *Real Time Software Design* by Hasan Gomaa
- *Software Architecture for Developers - Technical leadership and communication* by Simon Brown <https://leanpub.com/b/software-architecture>



OTHER RESOURCES - BOOKS

User Centered Design

- *The Elements of User Experience: User-Centered Design for the Web and Beyond* by Jesse James Garrett
- *User-Centered Design: A Developer's Guide to Building User-Friendly Applications* by Travis Lowdermilk
- *Sketching User Experiences* by Bill Buxton

Design Patterns

- *Head First Design Patterns* by Bert Bates et al
- *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma and Richard Helm

UML

- *UML Distilled* by Martin Fowler.
- *UML 2 For Dummies* by Michael Jesse Chonoles, James A. Schardt



OTHER RESOURCES

Journals

- IEEE Software
 - Computer
 - Journal of Systems and Software
 - Communications of the ACM
 - IEEE Transactions on Software Engineering
-
- Others - provided in class (or to use on your own)
 - Papers
 - Tutorials and other material on the web
 - Youtube, SlideShare
 - IEEE Computer Society Youtube channel:
<https://www.youtube.com/user/ieeeComputerSociety/videos>
 - Software Engineering radio: <http://www.se-radio.net>
 - Software Engineering Daily podcast:
<https://softwareengineeringdaily.com>



MY GOAL AND APPROACH FOR THIS CLASS

- To inspire and coach you in learning software design practices and developing skills that will support your growth as a software engineer, and enable you to move towards your dream job.

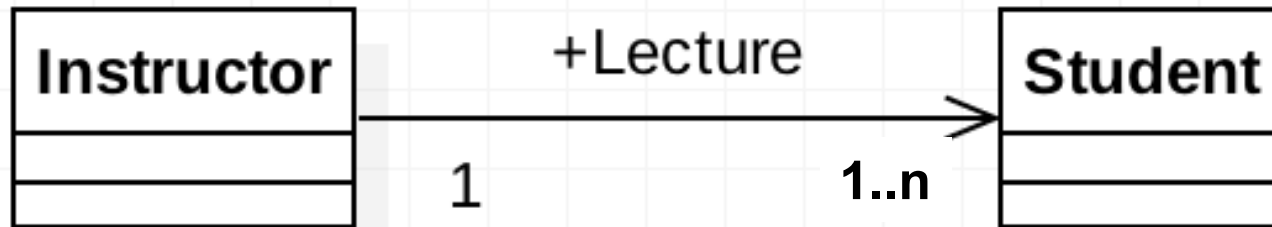


MY TEACHING PHILOSOPHY

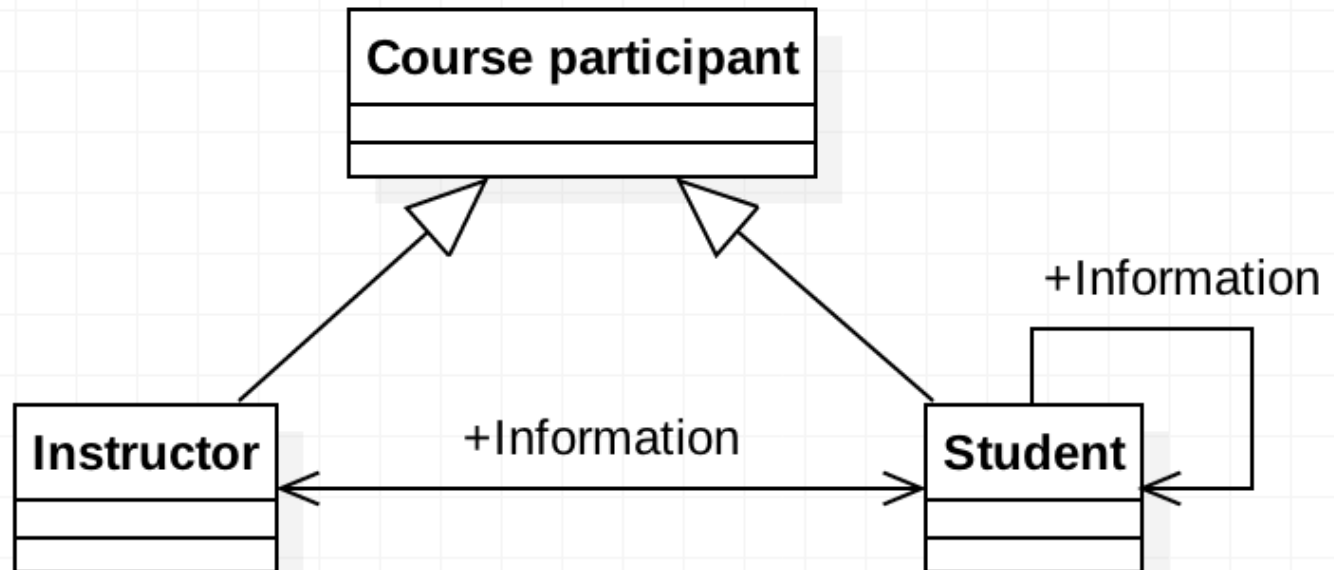
- Student centered vs. teacher centered
- Instructor as a coach and class as a team
 - Everyone's contribution counts (experience, opinions, thoughts)
 - Cooperation vs. competition (Do and be your best)
 - Reciprocal teaching and support
 - In a professional manner!
- Continuous learning, feedback and growth
 - Practice included in class, assessments, reviews/discussions
 - Early and frequent feedback to students (Quizzes and other assignments)
 - Feedback to instructor (Assessments, Discussions in ELMS)
- Result: “No surprise” exams and final grades



COURSE INTERACTION – TRADITIONAL MODEL



COURSE INTERACTION – NEW MODEL



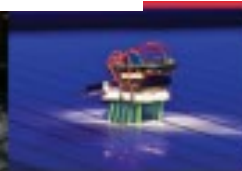
CLASS PARTICIPATION

- **YES** – laptops
- **NO** – cellphones
- Be present, engaged, active, curious, committed, professional
- ALWAYS be thoughtful, respectful, considerate
- Collaborate, support each other, learn from each other
- Use this class as a safe experimentation lab
- Avoid waste (time, \$, opportunities)
- Speak up, so everyone can hear you
- Remote site(s) students – do remind us that you are in class
- **Let me know of any special accommodation needs or special circumstances**
- Effort budget: ~8-10 hours/week (average)





www.shutterstock.com · 127880048



OUTLINE

- Introductions
 - Instructor
 - Students
 - ENPM 613 course



Critical thinking

- Ethics in software engineering
- Brief recap of software engineering
 - Software design concepts
 - This week's assignments



CRITICAL THINKING

“Critical thinking is the *intellectually disciplined* **process** of *actively and skillfully* **conceptualizing, applying, analyzing, synthesizing, and/or evaluating** information gathered from, or generated by, observation, experience, reflection, reasoning, or communication, as a **guide to belief and action.**”

[\[https://www.criticalthinking.org/pages/defining-critical-thinking/766 \]](https://www.criticalthinking.org/pages/defining-critical-thinking/766)



CRITICAL THINKING (CONTINUED)

Who	<ul style="list-style-type: none"> ... benefits from this? ... is this harmful to? ... makes decisions about this? ... is most directly affected? 	<ul style="list-style-type: none"> ... have you also heard discuss this? ... would be the best person to consult? ... will be the key people in this? ... deserves recognition for this?
What	<ul style="list-style-type: none"> ... are the strengths/weaknesses? ... is another perspective? ... is another alternative? ... would be a counter-argument? 	<ul style="list-style-type: none"> ... is the best/worst case scenario? ... is most/least important? ... can we do to make a positive change? ... is getting in the way of our action?
Where	<ul style="list-style-type: none"> ... would we see this in the real world? ... are there similar concepts/situations? ... is there the most need for this? ... in the world would this be a problem? 	<ul style="list-style-type: none"> ... can we get more information? ... do we go for help with this? ... will this idea take us? ... are the areas for improvement?
When	<ul style="list-style-type: none"> ... is this acceptable/unacceptable? ... would this benefit our society? ... would this cause a problem? ... is the best time to take action? 	<ul style="list-style-type: none"> ... will we know we've succeeded? ... has this played a part in our history? ... can we expect this to change? ... should we ask for help with this?
Why	<ul style="list-style-type: none"> ... is this a problem/challenge? ... is it relevant to me/others? ... is this the best/worst scenario? ... are people influenced by this? 	<ul style="list-style-type: none"> ... should people know about this? ... has it been this way for so long? ... have we allowed this to happen? ... is there a need for this today?
How	<ul style="list-style-type: none"> ... is this similar to _____? ... does this disrupt things? ... do we know the truth about this? ... will we approach this safely? 	<ul style="list-style-type: none"> ... does this benefit us/others? ... does this harm us/others? ... do we see this in the future? ... can we change this for our good?



FOOD FOR THOUGHT



Barbara Kruger, *Belief + Doubt*, Hirshorn Gallery, Washington DC



CRITICAL THINKING



Watch the videos below for understanding the elements of critical thinking:

- Purpose <https://www.youtube.com/watch?v=tAPgRHPCQEA>
- Question-at-issue <https://www.youtube.com/watch?v=zZvJPbQRnQ0>
- Information https://www.youtube.com/watch?v=l3j_QRAHnrw&feature=youtu.be
- Concepts and Ideas <https://www.youtube.com/watch?v=iGkBychY-0&feature=youtu.be>
- Intellectual standards <https://www.youtube.com/watch?v=gcxhalSk73k&feature=youtu.be>
- Assumptions <https://www.youtube.com/watch?v=2qLwTPq-5ul>
- Inference and Interpretation <https://www.youtube.com/watch?v=xzD3Xsi3luM&feature=youtu.be>
- Point of view <https://www.youtube.com/watch?v=VuXf6w3uxKg&feature=youtu.be>
- Implications and Consequences <https://www.youtube.com/watch?v=wrDX-1rAHOA&feature=youtu.be>



OUTLINE

- Introductions
 - Instructor
 - Students
 - ENPM 613 course
- Critical thinking
- ➔ **Ethics in software engineering**
 - Brief recap of software engineering
 - Software design concepts
 - This week's assignments



WHAT IS *ETHICS*?



WHAT IS *ETHICS*?

“moral principles that govern a person's behavior or the conducting of an activity.”

<https://www.lexico.com/en/definition/ethics>



ETHICS

- 2018 - ACM Code of Ethics and Professional Conduct
<https://www.acm.org/code-of-ethics>
- Older Code (2007) - Software Engineering Code of Ethics and Professional Practice <https://www.acm.org/code-of-ethics>
- **WHY**
 - “Because of their roles in developing software systems, **software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm.**”
 - “To ensure, as much as possible, that their efforts will be used for good, **software engineers must commit themselves to making software engineering a beneficial and respected profession.** In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.”
- Quiz on Ethics in ELMS



OUTLINE

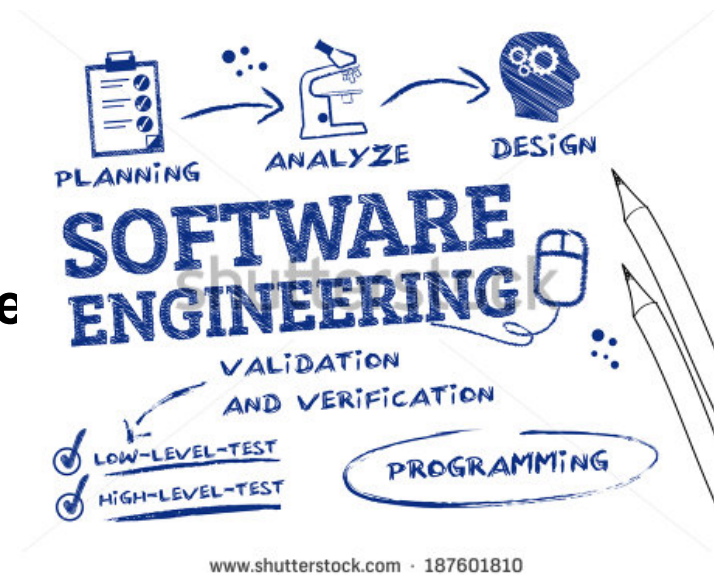
- Introductions
 - Instructor
 - Students
 - ENPM 613 course
- Critical thinking
- Ethics in software engineering
- ➔ **Brief recap of software engineering**
 - Software design concepts
 - This week's assignments



WHAT IS SOFTWARE ENGINEERING?

(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, **the application of engineering to software** (IEEE 1990)

(2) The study of approaches as in (1).
(IEEE 1990)



[http://sebokwiki.org/wiki/Software_Engineering_\(glossary\)](http://sebokwiki.org/wiki/Software_Engineering_(glossary))

IEEE Standard Glossary of Software Engineering Terminology. IEEE 610.12-1990



IS ALL SOFTWARE THE SAME?

- Why? or Why not?
- Why do we study “software” engineering (software design)?



IS ALL SOFTWARE THE SAME?

- Why? or Why not?
- Why do we study “software” engineering?
 - **General** principles and practices that *usually* apply to *most* software
 - Tailoring/customization might be needed, depending on the context



WHY DO WE NEED SOFTWARE *ENGINEERING*?

- SW development is expensive, takes (too) long
- SW is large, complex, pervasive, can fail, and cause harm
 - Question: How much software is in a car (lines of code)?



HOW LARGE IS SOFTWARE TODAY?

- A Chevy Volt uses **10 million lines of code**.
- The control software to run a U.S. military drone uses **3.5 million lines of code**.
- A Boeing 787 has **6.5 million lines** behind its avionics and online support systems.
- The Android operating system runs on 12-15 million lines.
- The Large Hadron Collider uses 50 million lines.
- Not including backend code, Facebook runs on 62 million lines of code.
- With the advent of sophisticated, cloud-connected infotainment systems, the car software in a modern vehicle apparently uses 100 million lines of code. (according to Wired magazine)
- Google Chrome (browser) runs on 6.7 million lines of code (upper estimate).
- All Google services combine for a whopping 2 billion lines.

<http://www.visualcapitalist.com/millions-lines-of-code/>



SOFTWARE CAN ENABLE SAFETY INCIDENTS

- ROCKET LAUNCH ERRORS - Explosion of rocket Ariane 5, 1996 https://www.youtube.com/watch?v=PK_yguLapgA
 - Conversion of a 64 bit integer into a 16 bit signed integer lead to an overflow
- DEADLY RADIATION THERAPY - Therac-25 radiation overdoses ('85 – '87)
 - Poor system design; poor development process; insufficient system test
- FLIGHT CRASHES
- LOST IN SPACE
- [How the Boeing 737 Max Disaster Looks to a Software Developer](#)



SOFTWARE CAN ENABLE SECURITY INCIDENTS (1)

- SQL injection – due to failure to separate data and control instructions and the co-mingling of them in a string
 - Is the “network infiltration method for 12 of 17 corporations that were fleeced of roughly 160 million credit card numbers in a seven-year-long hacking campaign that ended last year.” (2013)
- Vulnerabilities in operating systems or application software
 - Malware (Worms/trojans) attacks
 - [Stuxnet](#) – attacked Iranian nuclear facility in 2010, exploited Siemens WinCC SCADA (Supervisory Control And Data Acquisition) system
- [Car hacking](#)
- [Ransomware](#) attacks



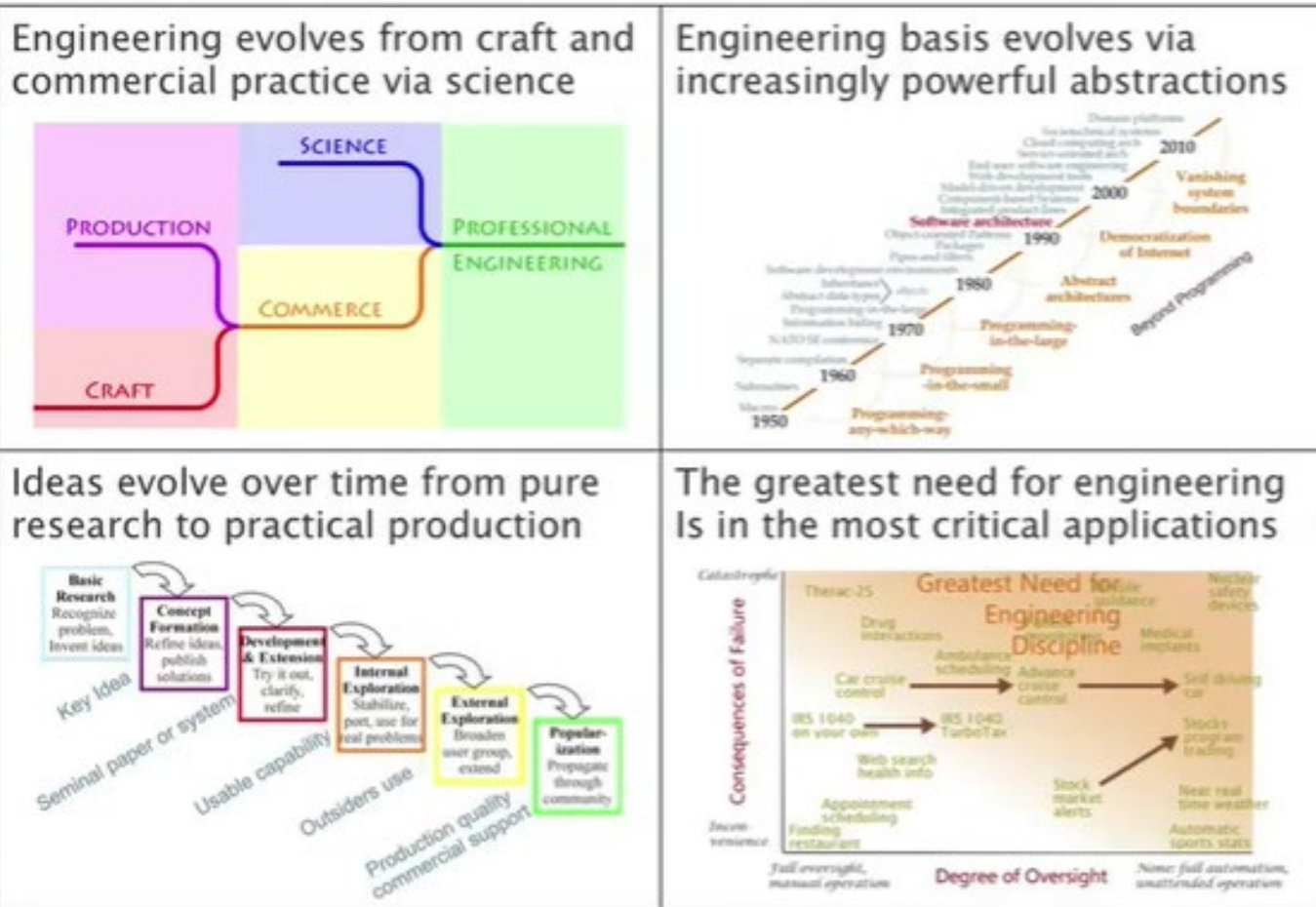
SOFTWARE CAN ENABLE SECURITY INCIDENTS (2)

- Target [data breach](#) - a design flaw leading to a hack
 - It was "easy to get to the middle where all the data was stored" once the attackers had compromised the point-of-sale system.
 - "The **design** of the communications and storage... were [poorly] done. Often a bug provides a foothold into a system to be exploited because of bad security **design**."
 - (from Garry McGraw)



SOFTWARE ENGINEERING - ARE WE THERE YET?

Recapitulation



<https://www.youtube.com/watch?v=S03bsjs2YnQ> (~1hr)



SOFTWARE DEVELOPMENT ACTIVITIES

- Requirements engineering and management
- Design
 - Architecture
 - Detailed design
- Implementation
- Verification and validation
- Deployment
- Operation
- Maintenance and evolution

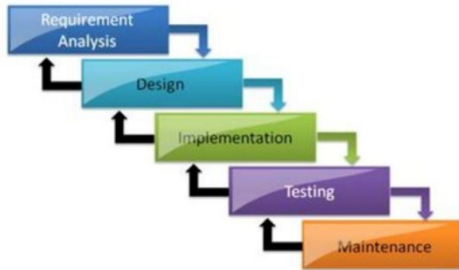
Project and
process
Management

Configuration
and Change
Management



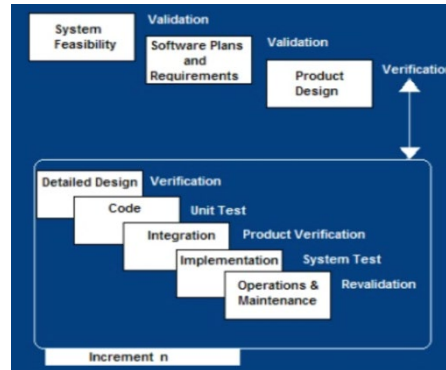
SOFTWARE DEVELOPMENT LIFECYCLE MODELS

Waterfall

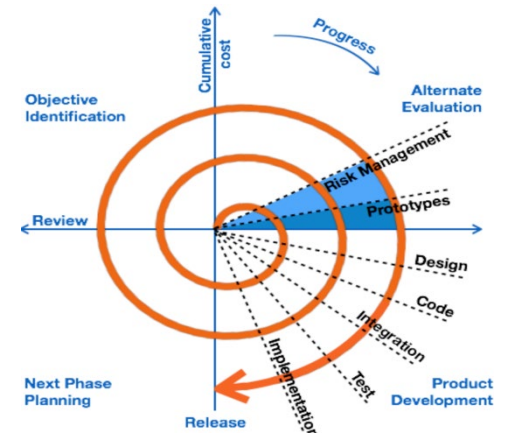


© Presentation-Process.com

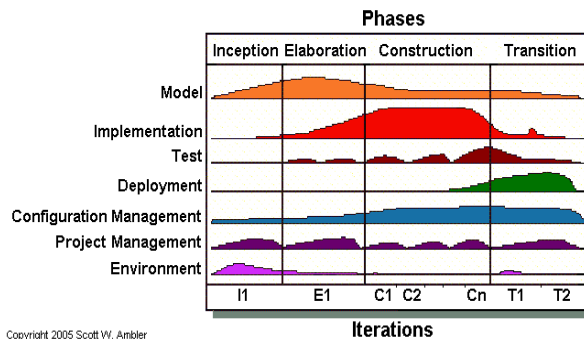
Incremental



Spiral

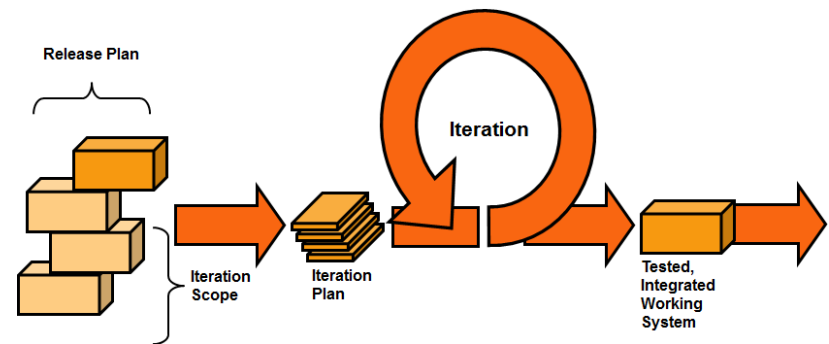


RUP

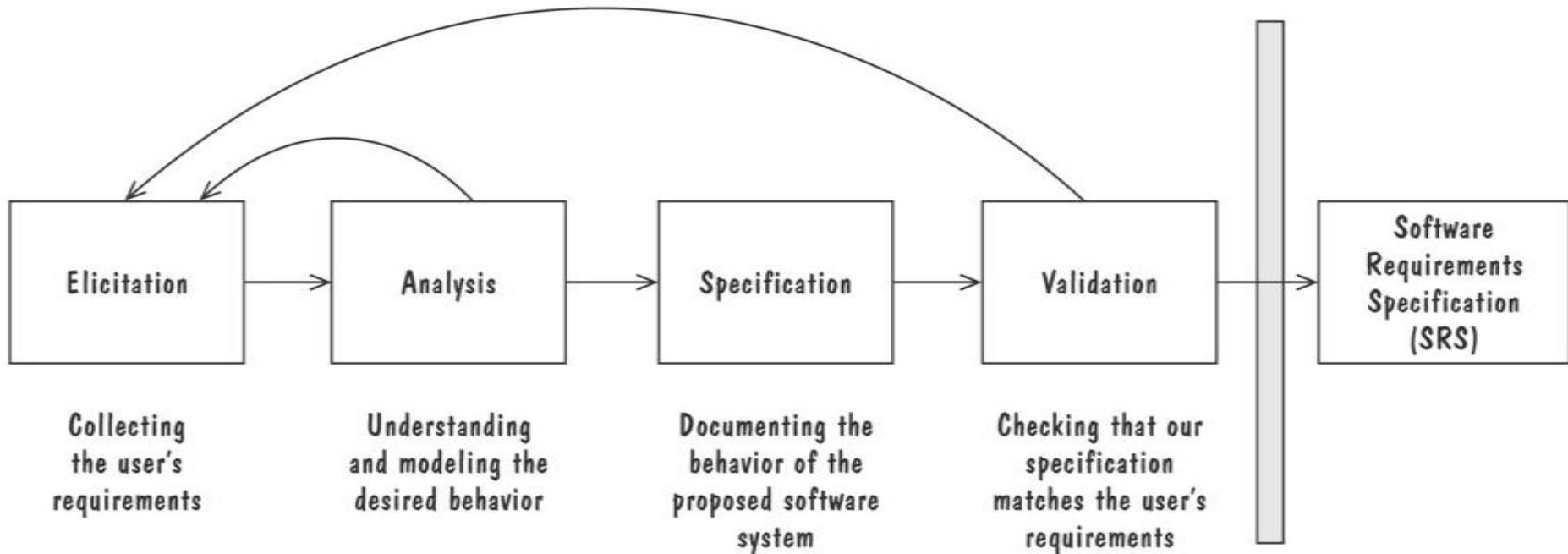


Copyright 2005 Scott W. Ambler

Agile



SOFTWARE REQUIREMENTS



[Source: Pfleeger & Atlee]



SOFTWARE DESIGN

- “Software engineering design is the activity of *specifying programs* and **sub-systems**, and **their constituent parts and workings**, to meet **software product specifications**.” (C. Fox)
- *Deriving* a **solution** which satisfies **software requirements**

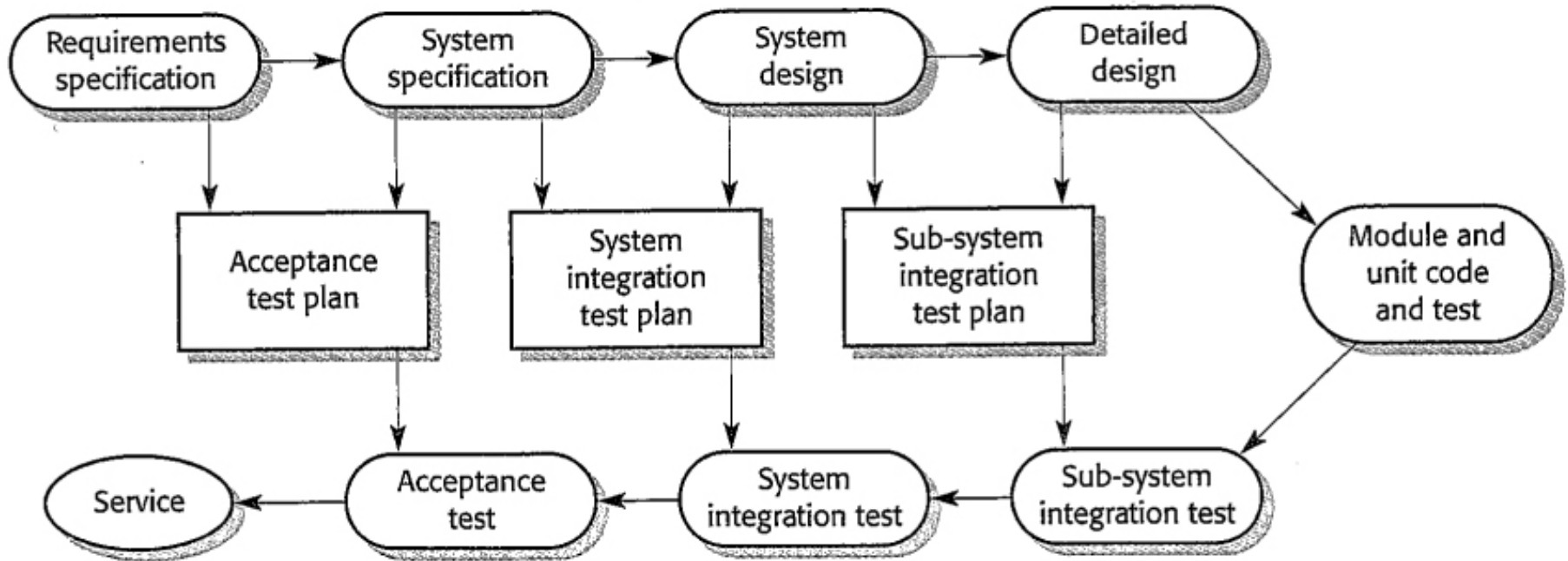


IMPLEMENTATION

- Given a set of requirements and a design, implement the system that satisfies both
 - Coding
 - Debugging
 - Verification and validation
 - Version/configuration control
 - Tracking issues
 - Managing developers



VERIFICATION AND VALIDATION



From: Ian Sommerville, Software Engineering

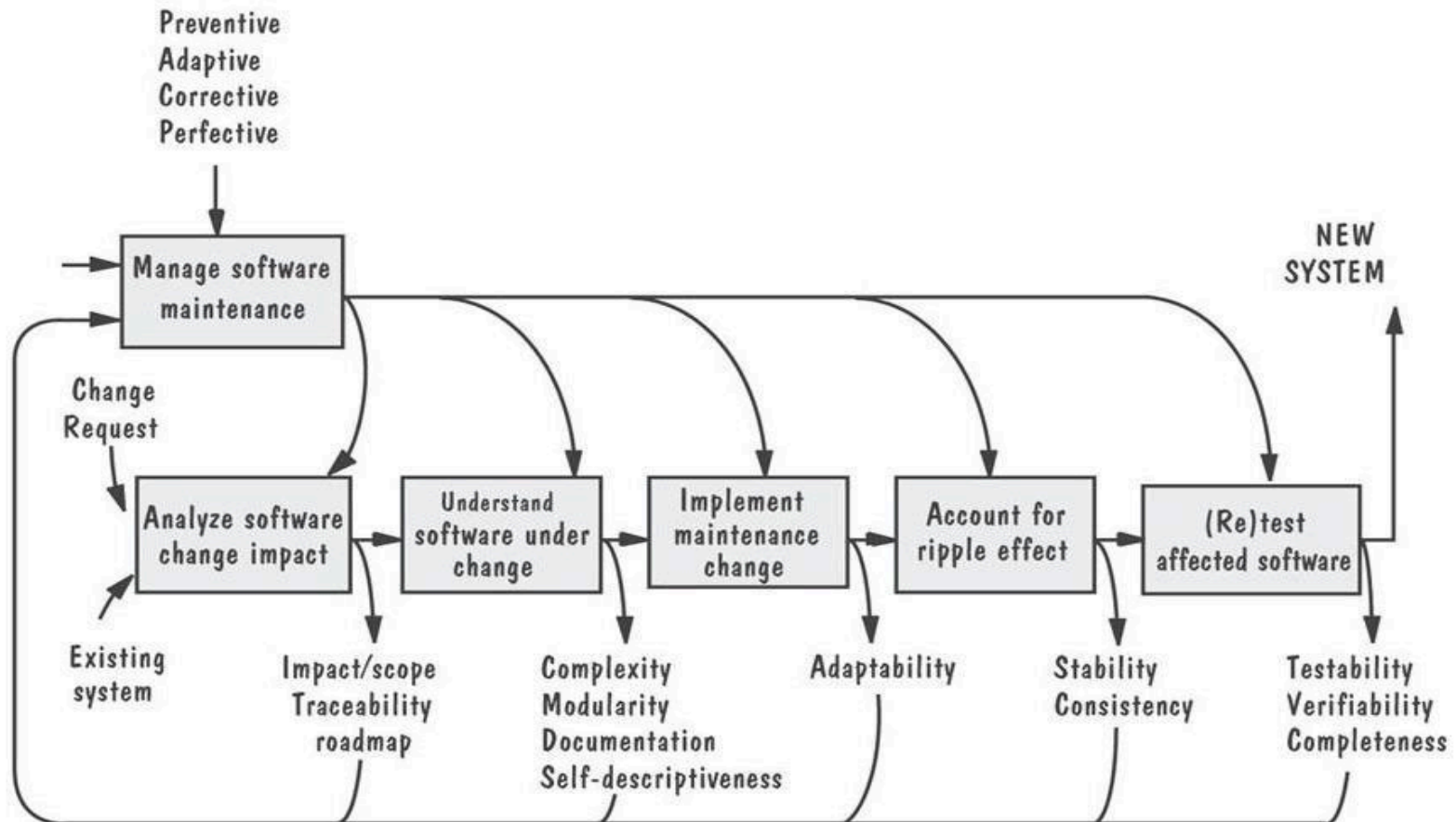


DELIVERY/DEPLOYMENT AND BEYOND

- Installation
- Training
- Operation/Support/Troubleshooting
- Maintenance/evolution
- Retirement

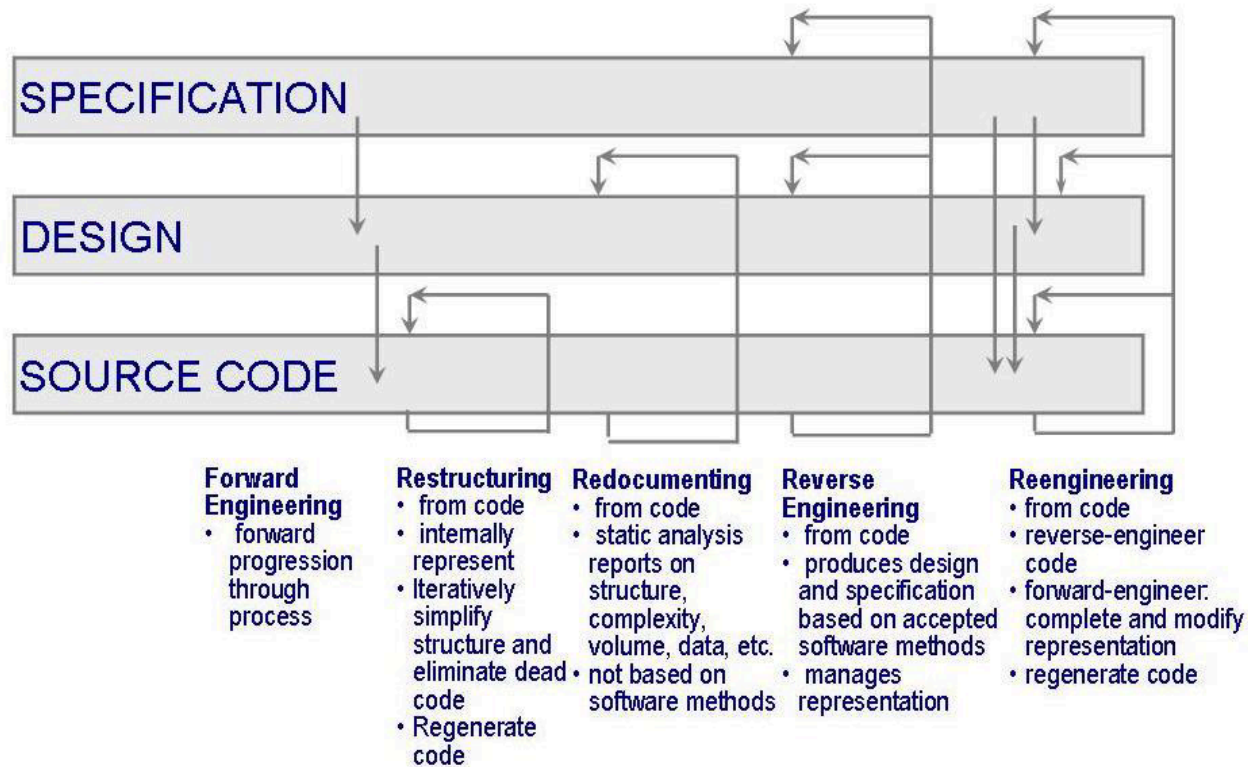


MAINTENANCE AND EVOLUTION



SOFTWARE “REJUVENATION”

- The concept of periodically stopping the running of software, cleaning its internal state through garbage collection, defragmentation, flushing operating system kernel tables, reinitializing data structures, and restarting it.



[From Pfleeger & Atlee - ENPM611]



OUTLINE

- Introductions
 - Instructor
 - Students
 - ENPM 613 course
- Critical thinking
- Ethics in software engineering
- Brief recap of software engineering
- ➔ **Software design concepts**
- This week's assignments



WHAT IS SOFTWARE DESIGN?

- “Software engineering design is the activity of *specifying* **programs** and **sub-systems**, and **their constituent parts and workings**, to meet **software product specifications**.”
(C. Fox)
- *Deriving* a **solution** which satisfies **software requirements**
- The software to be developed is meant to “solve a problem”
 - Requirements engineering - we *specify* **the problem (need)**;
 - Design - we *develop/select/specify* a **solution to the problem (need)**

- **Question:** are there more solutions?

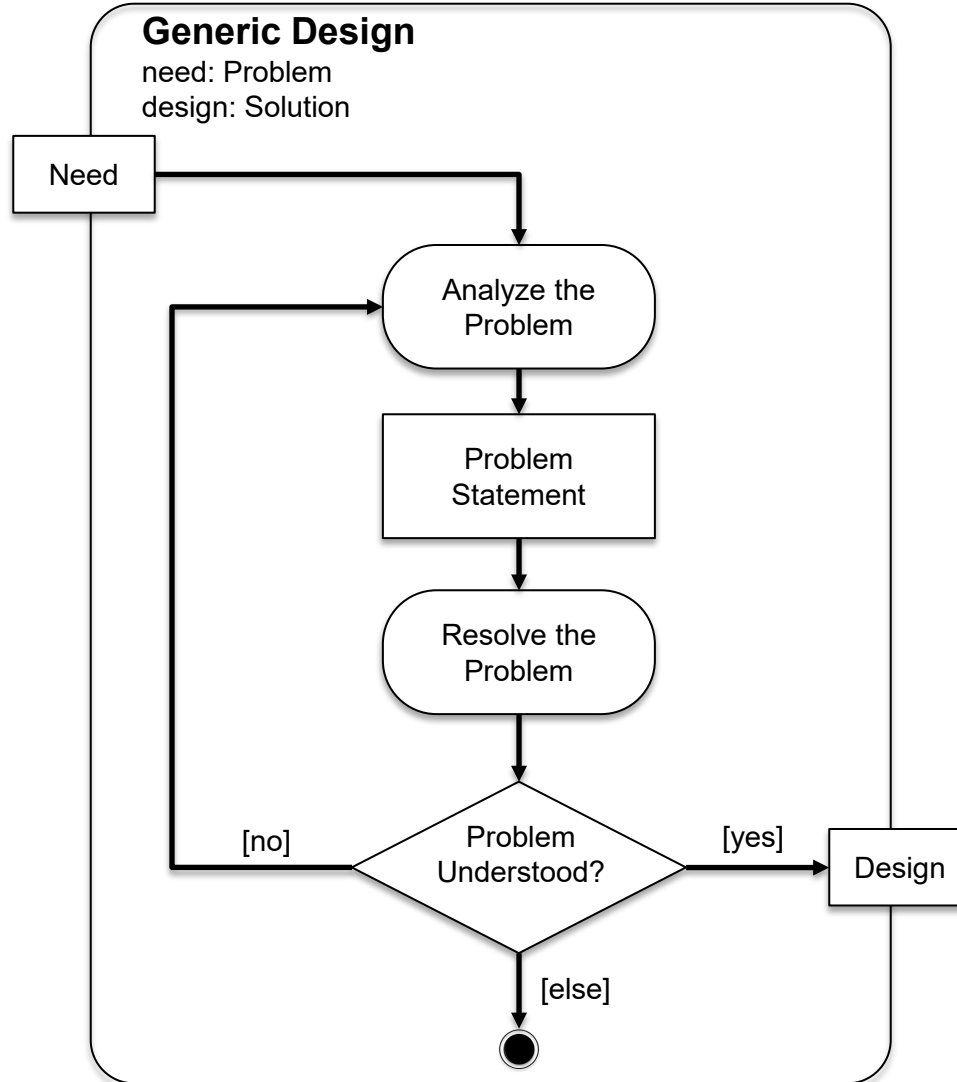


ELEMENTS OF DESIGN AS ACTIVITY

- *Design Process*—A collection of related tasks that transforms a set of inputs into a set of outputs
- *Design principles* - Characteristics of design that make them better
- *Design Heuristics*—Rules providing guidance, but no guarantee, for achieving some end
- *Design Notations*—A symbolic representational system
 - Modeling language

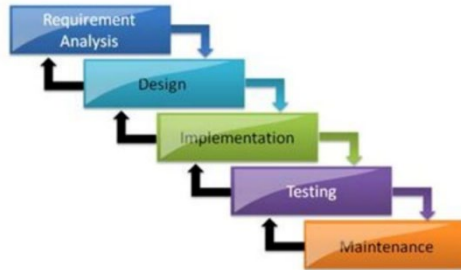


A GENERIC DESIGN PROCESS



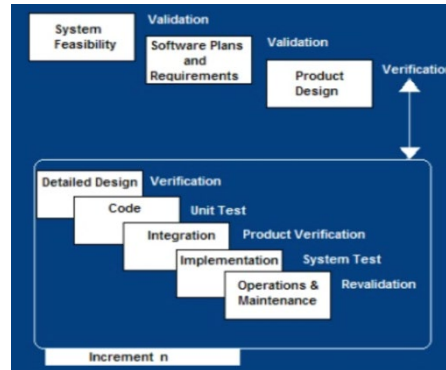
SOFTWARE DEVELOPMENT LIFECYCLE MODELS

Waterfall

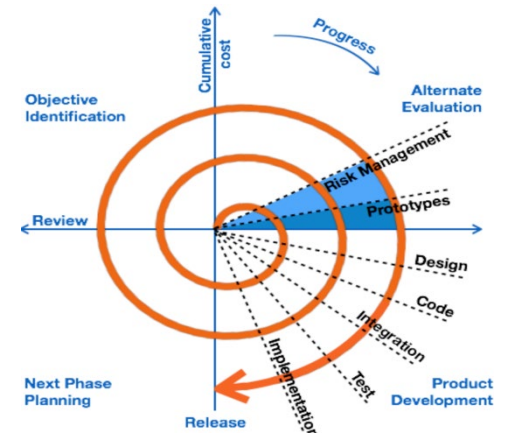


© Presentation-Process.com

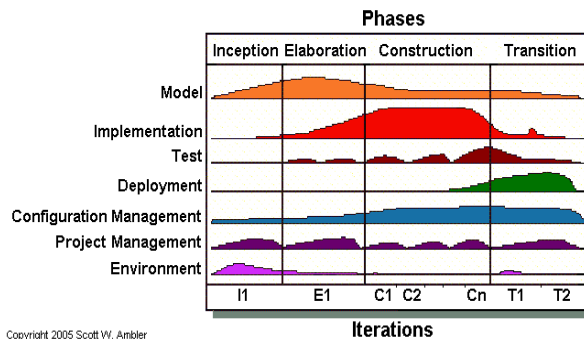
Incremental



Spiral

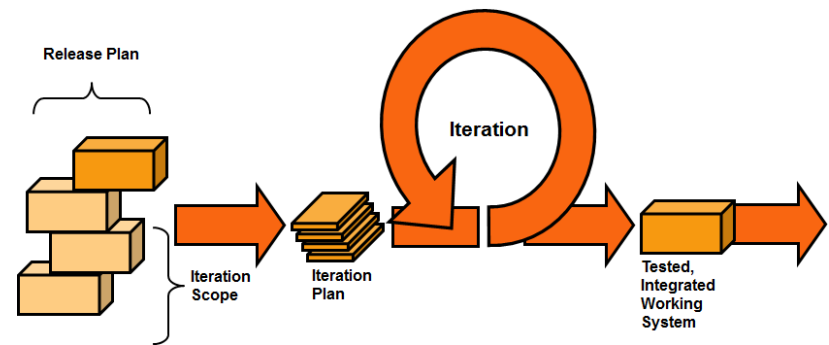


RUP



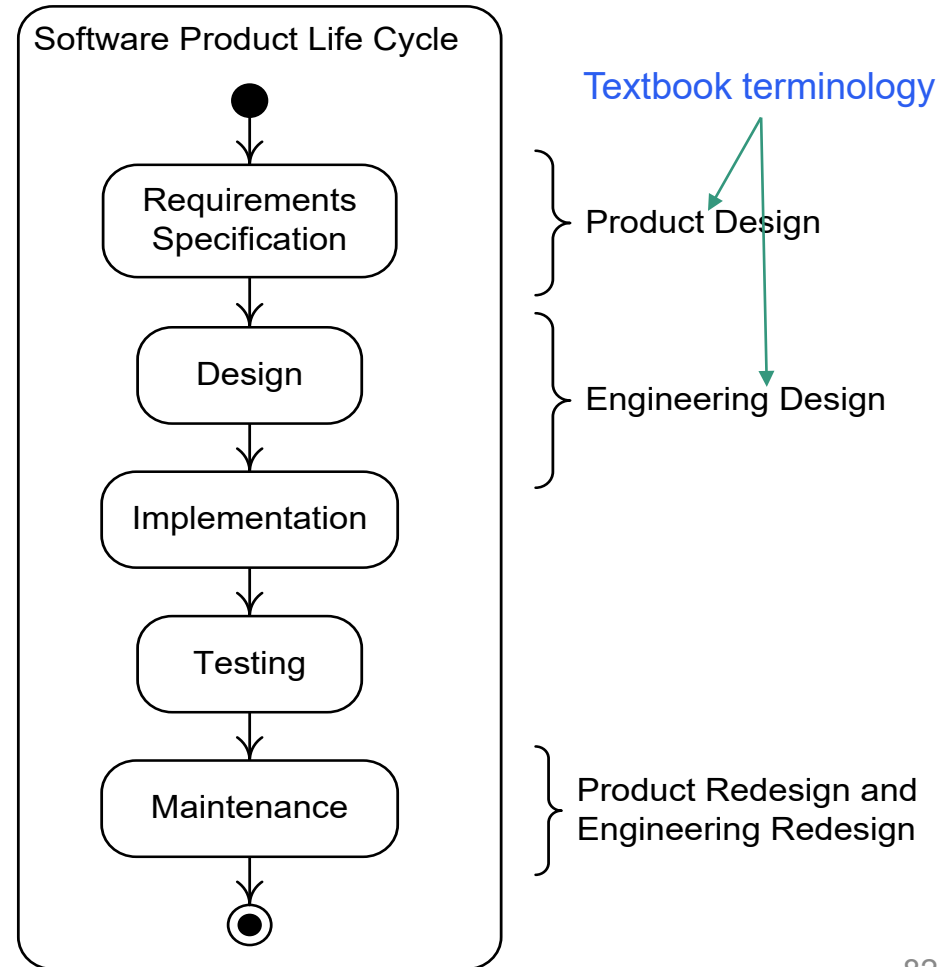
Copyright 2005 Scott W. Ambler

Agile



DESIGN IN THE SOFTWARE LIFECYCLE – C. FOX TEXTBOOK TERMINOLOGY

- C. Fox: “Software design comprises both product and engineering design.”
- **Software product design** is the activity of specifying software product features, capabilities, and interfaces to satisfy client needs and desires.
- **Software engineering design** is the activity of specifying programs and sub-systems, and their constituent parts and workings, to meet software product specifications.
- Product design occurs mainly in the requirements specification phase; engineering design mainly in the design and implementation phases.

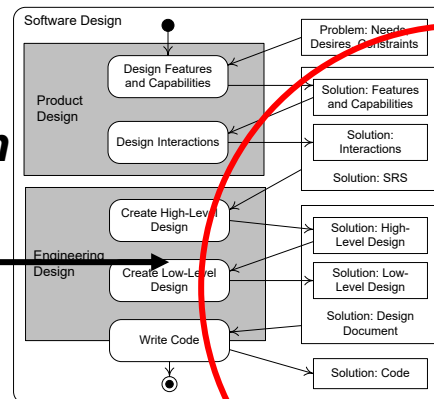


DESIGN IN THE SOFTWARE LIFECYCLE – C. FOX TEXTBOOK

TERMINOLOGY (CONTINUED)

The topic of **Requirements Engineering** is the subject of ENPM612

ENPM613 focuses on **Design Interactions** (a.k.a. *User experience/User centered design*) and **Engineering design**



DO WE NEED DESIGN? WHY? WHEN?



WHY DO WE NEED SOFTWARE DESIGN?

- To reason about solutions
- To document solutions
- To communicate solution(s) to stakeholders and reach agreement

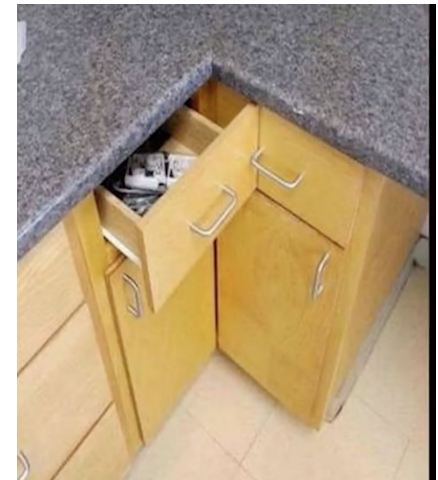


WHY DO WE NEED SOFTWARE DESIGN?

- “You can use an eraser on the drafting table, or a sledge hammer on the construction site” –Frank Lloyd Wright



LACK OF DESIGN OR BAD DESIGN



87



BAD ARCHITECTURE



**"If you think good architecture is expensive, try bad architecture."
- Brian Foote and Joseph Yoder**



WHEN DO WE NEED SOFTWARE DESIGN (AND HOW MUCH)?



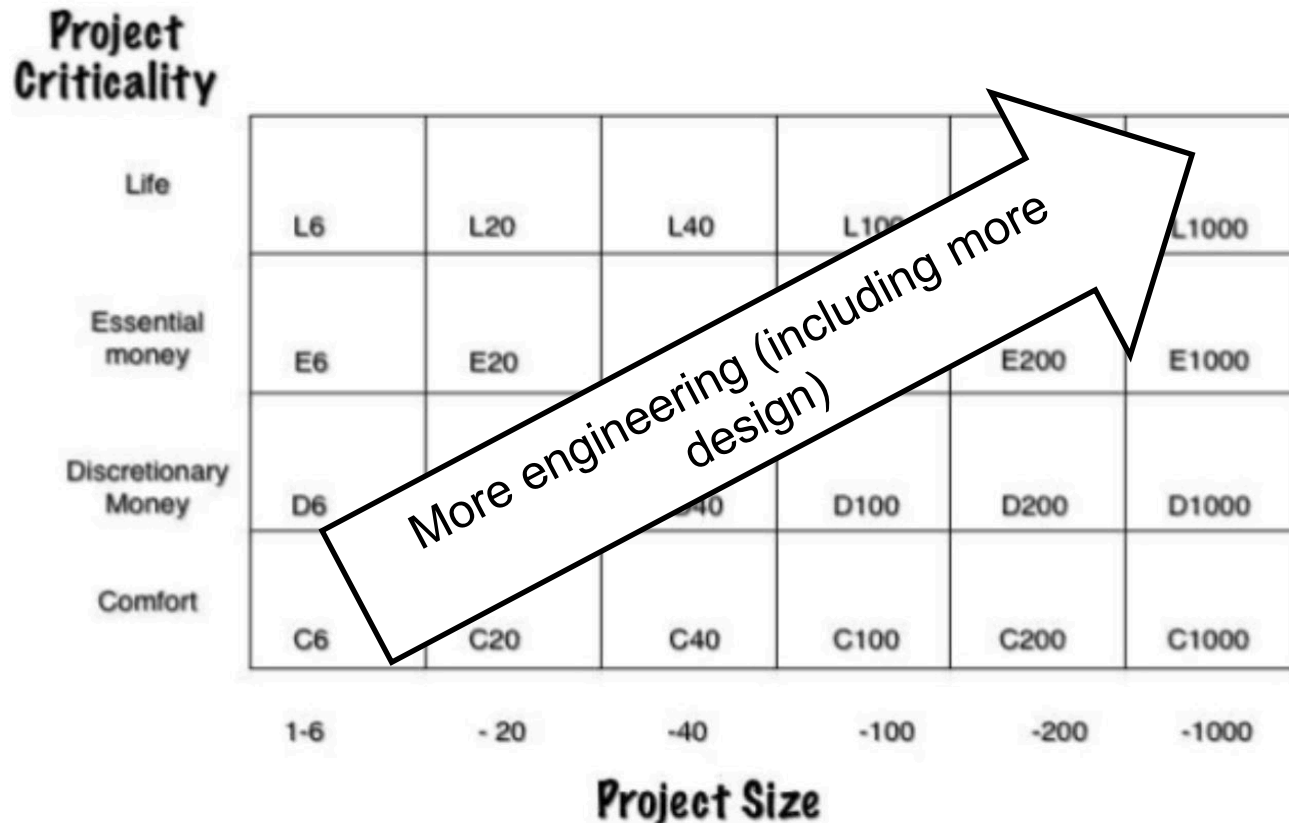
COMPARE AND CONTRAST



COMPARE AND CONTRAST (CONTINUED)



WHEN DO WE NEED SOFTWARE DESIGN (AND HOW MUCH)?



Alistair Cockburn's project characteristics grid

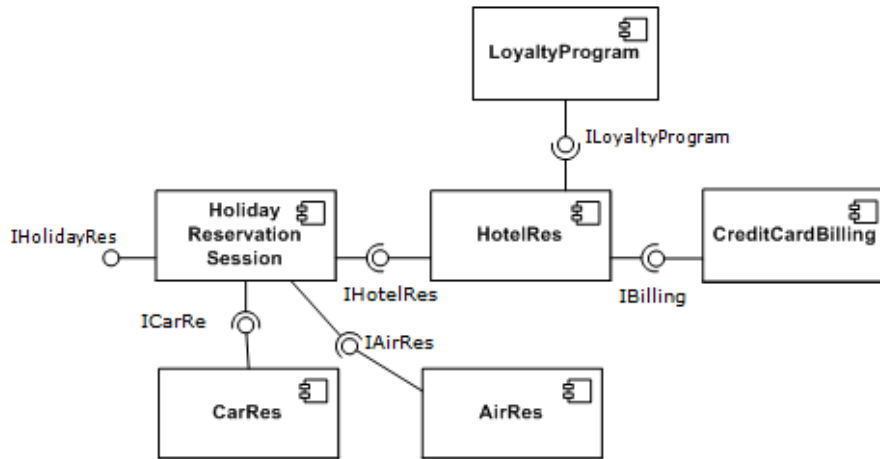


WHAT DO WE DESIGN (IN SOFTWARE DEVELOPMENT)?

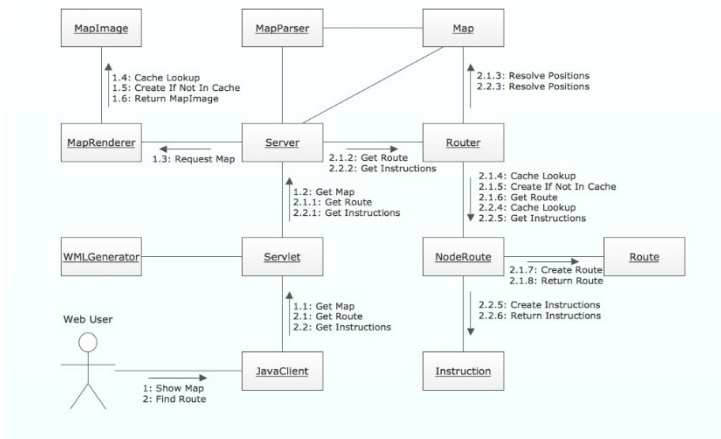
- Structure
 - Behavior
 - Data/Information (structured and unstructured)
 - User interaction
- Engineering design
 - User centered design



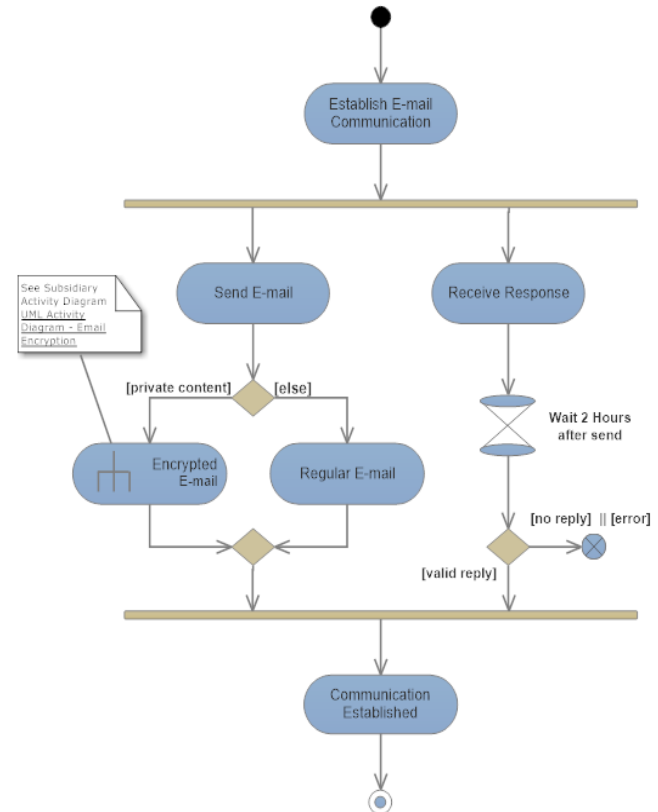
ENGINEERING DESIGN MODELS – EXAMPLES



Client Server Access



UML Activity Diagram: Email Connection



USER CENTER DESIGN MODELS – EXAMPLES

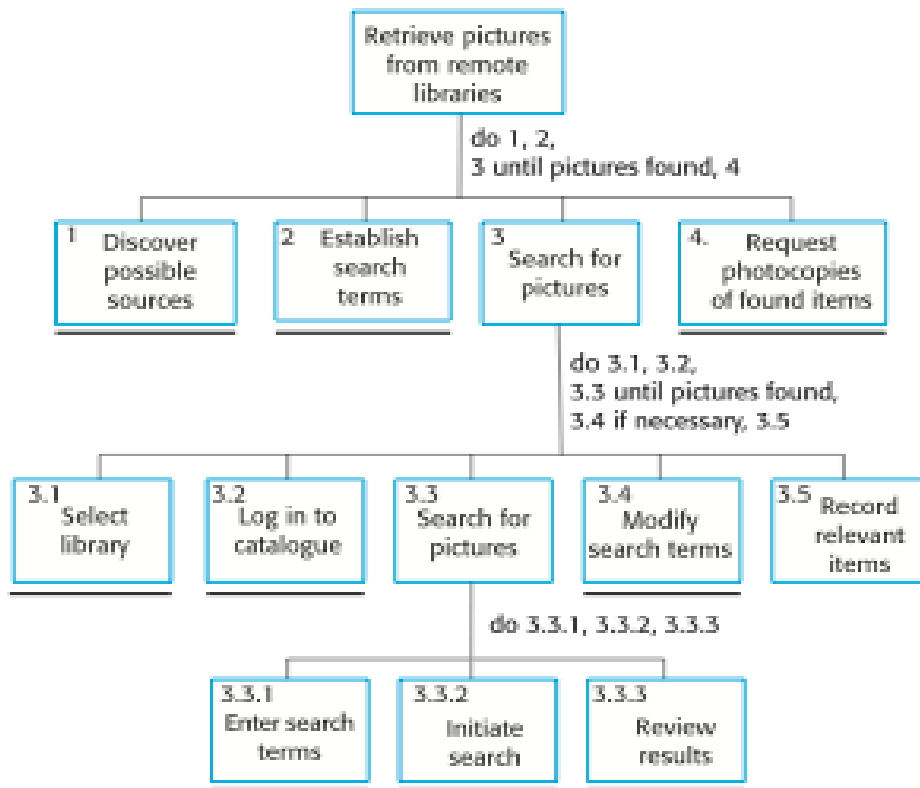


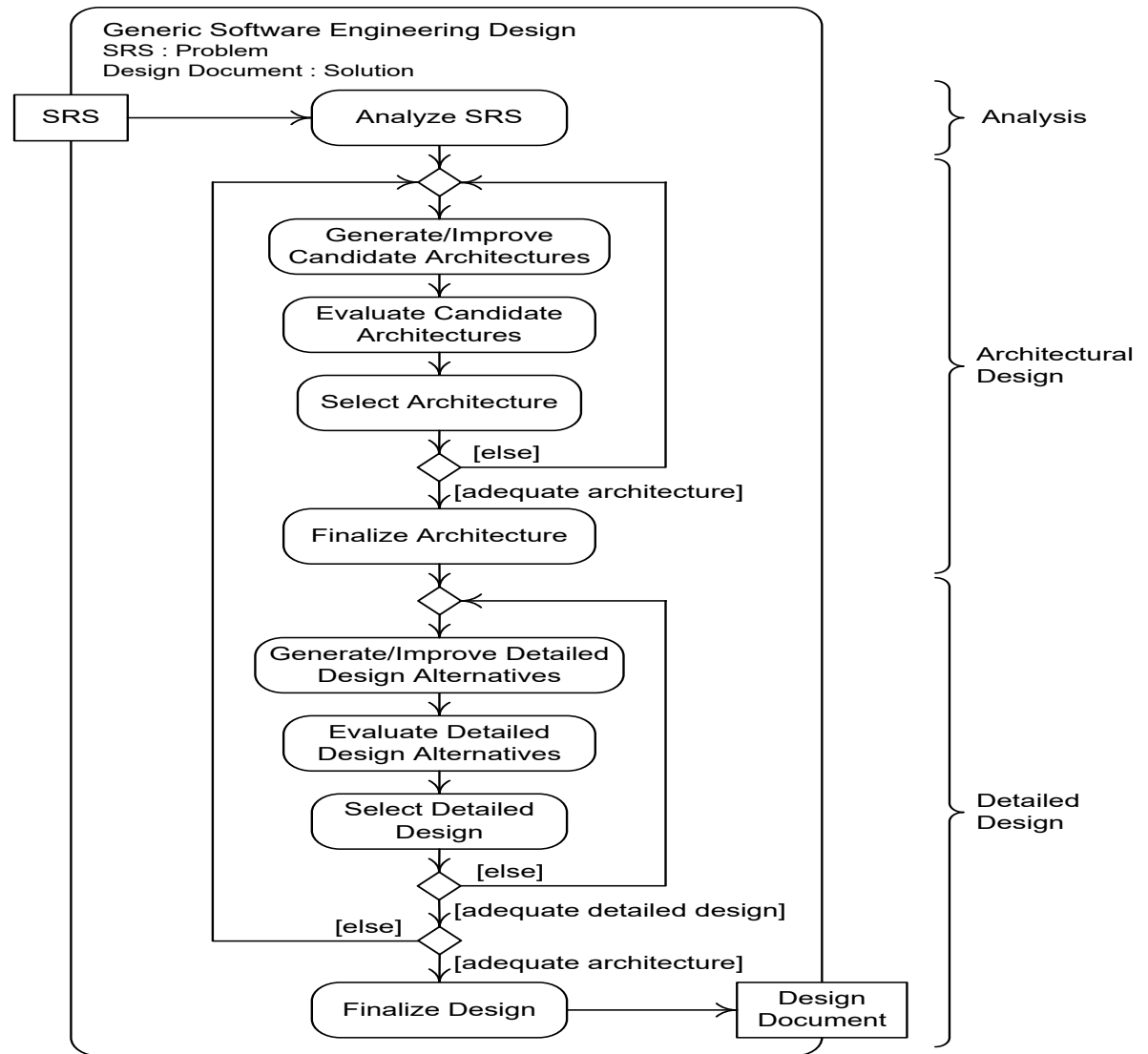
Figure 99: Sketches of PDA Agenda Screens

The sequence of images sketches out a potential design for interacting with a PDA-based agenda. Each image is like a key frame in an animation. Notice the **state transition** diagram at the bottom of each image, which shows its context relative to the others, as well as the overall page hierarchy.

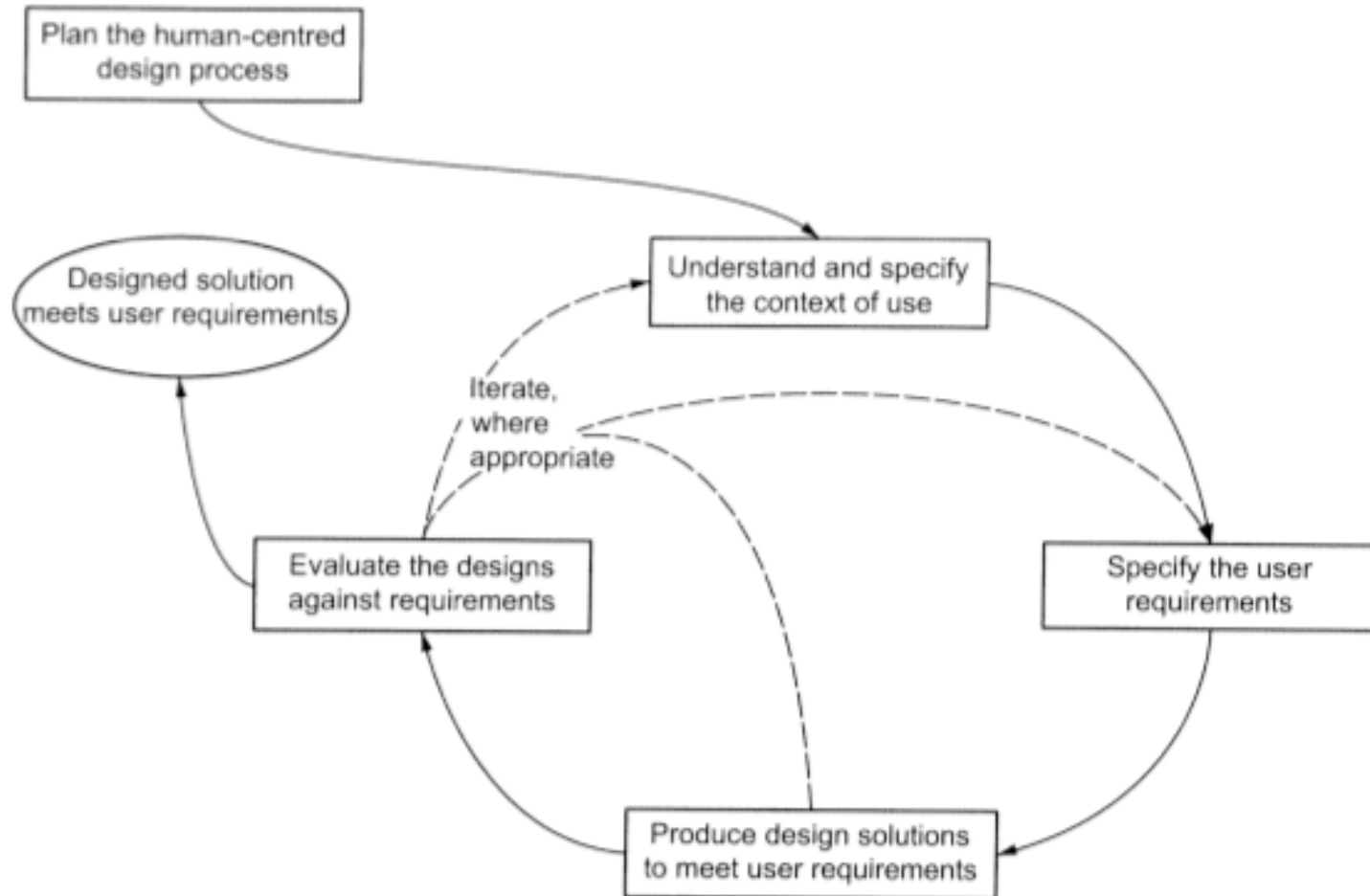
Images: Ron Bird



SOFTWARE ENGINEERING DESIGN PROCESS



USER CENTERED DESIGN (UCD) PROCESS



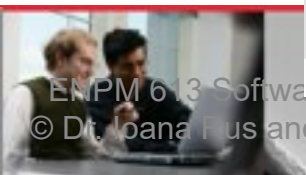
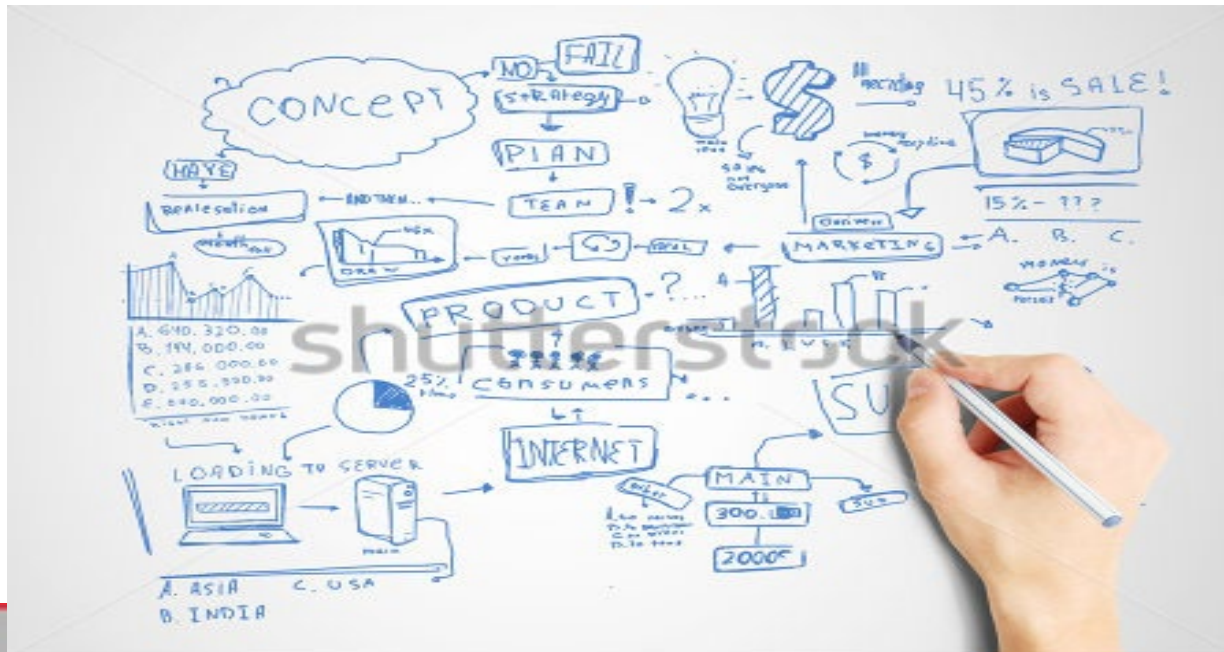
97

[From ISO 9241 Part 210]



MODELING IN SOFTWARE DEVELOPMENT

- “A schematic description of a system [...] that accounts for its known or inferred properties and may be used for further study of its characteristics.”
 - The American Heritage® Dictionary of the English Language, Fourth Edition, (Noun, def. 3)
- A model is an **abstraction** of something

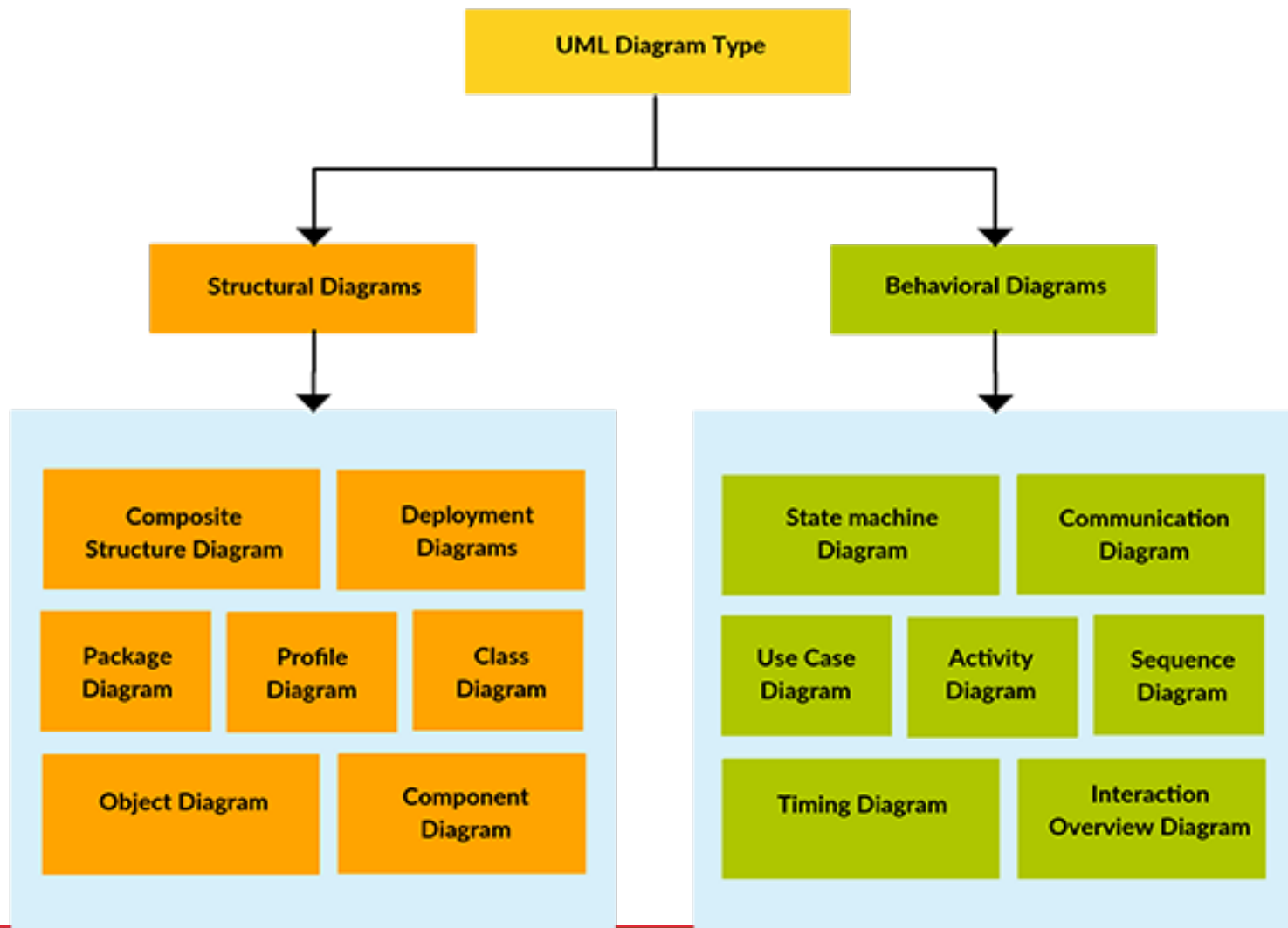


UNIFIED MODELING LANGUAGE (UML)

- A nonproprietary, standard, general purpose, graphical language for modeling software systems
- Used for visualizing, specifying, constructing, and documenting the artifacts of software-intensive systems



UML DIAGRAMS



OUTLINE

- Introductions
 - Instructor
 - Students
 - ENPM 613 course
- Critical thinking
- Ethics in software engineering
- Brief recap of software engineering
 - Software design concepts
 - Requirements analysis

 This week's assignments



ASSIGNMENTS

- Pre-requisite quiz – in ELMS
- Syllabus quiz – in ELMS
- Ethics quiz – in ELMS
- Discussion – Students Introduction – in ELMS

- Revisit/refresh your knowledge of:
 - Requirements engineering
 - Functional and non-functional requirements
 - UML class diagram, use case diagram, activity diagram, and sequence diagram





www.shutterstock.com · 127880048

