CHAPTER 1.2
WHAT IS SOFTWARE ENGINEERING?

BIE 33503

Special Topic of Software Engineering

TABLE OF CONTENT

WHAT IS SOFTWARE ENGINEERING? WHY DO WE NEED IT

- I already know how to code --- I am good programmer.
- I have been developing systems for years.
- I have been supporting systems and answering tough customers questions for years.
- We are using Agile process and we are just cool --- making a lot of money in the process.

WHAT IS SOFTWARE ENGINEERING?

- Software Engineering has 2 main parts:

1. Identification and Analysis of "problems"

2. Identification, Synthesis, and Construction of the "solution" to the problem.

SOFTWARE ENGINEERING

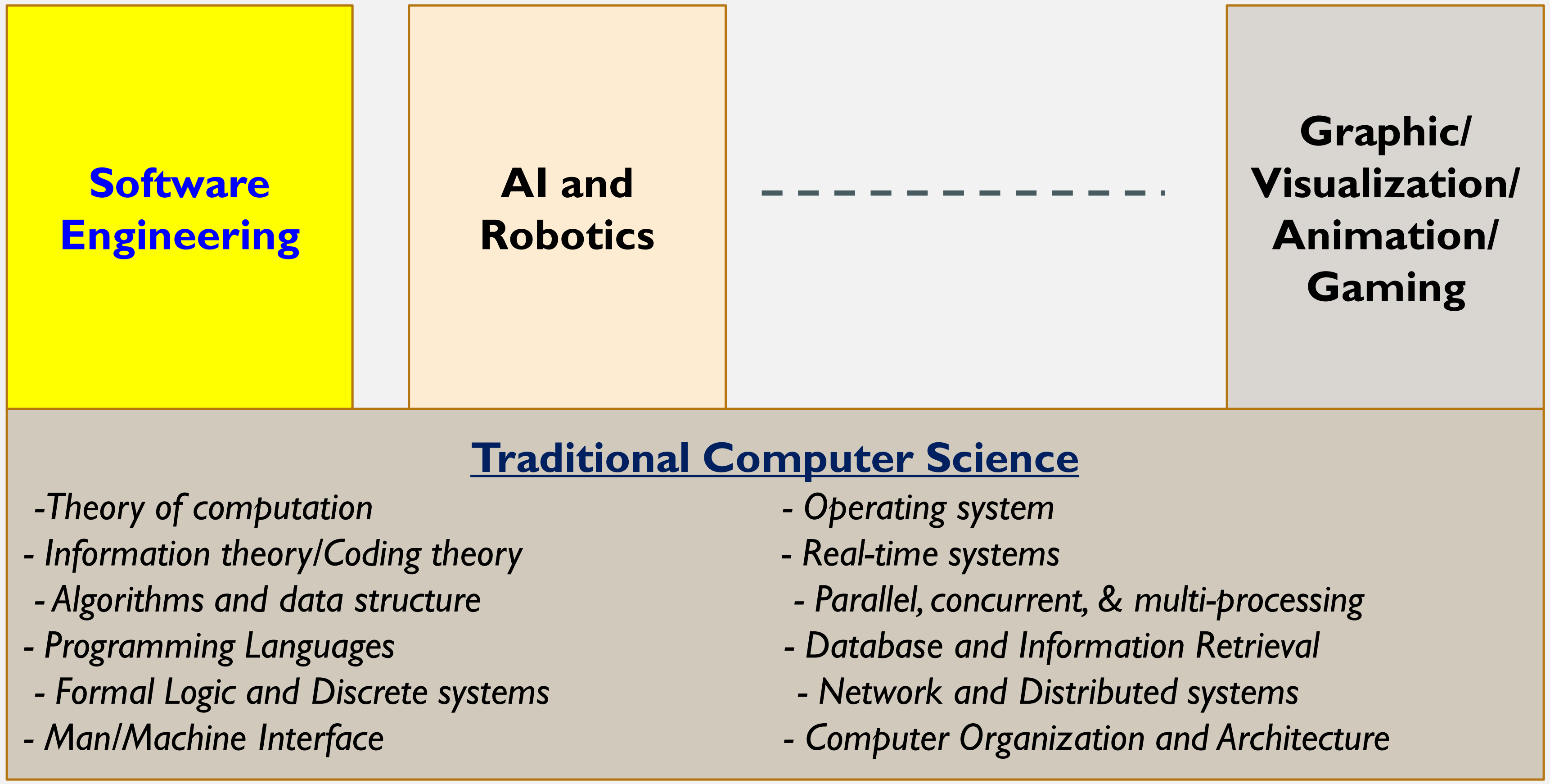
- In identifying and analyzing software problems, we use:
  - techniques: elicitation, documentation, prototyping, reviews, etc.
  - business knowledge: domain specific info., business flow, etc.

- In solving problems software engineering employs:
  - Methodology or Technique: (e.g.) designing, programming, testing, integrating, etc. which are directly related to the end product
  - Tools: (e.g.) development platform, version control, visual diagram, etc.
  - Procedure & Policies: (e.g.) inspection/review, tracking, metrics, change management, etc. which are indirectly related to the product
  - Process / Paradigm : (e.g.) Waterfall, Spiral, Incremental, etc. which are a combination of methodology, tools, and procedures

SOFTWARE ENGINEERING & OTHER DISCIPLINES

- Software Engineers uses the "computing" theories, tools, algorithms, etc. from computer science.
- Software Engineers uses procedures, techniques, and tools from other disciplines such as management science, industrial engineering, and cognitive science.
- *Software Engineers also perform in depth research in some of the above areas themselves. (my view: I differ with the text author, p.5, a bit here –)*

COMPUTING FIELDS


Software Engineering
AI and Robotics
Graphic/ Visualization/ Animation/ Gaming
Traditional Computer Science

Traditional Computer Science

- *-Theory of computation*
- *- Information theory/Coding theory*
- *- Algorithms and data structure*
- *- Programming Languages*
- *- Formal Logic and Discrete systems*
- *- Man/Machine Interface*
- *- Operating system*
- *- Real-time systems*
- *- Parallel, concurrent, & multi-processing*
- *- Database and Information Retrieval*
- *- Network and Distributed systems*
- *- Computer Organization and Architecture*

SOFTWARE ENGINEERING TOPICS

Software Development & Engineering

- *Requirements Engr.*
- *Designing*
-*Construction /coding*
- *Testing*

Management & Maintenance

- *Evolution & Support*
- *Configuration & Release mgmt*
- *Project Management*

Software Engineering Support

- *Measurement & Metrics*
- *Process & Methodology*
- *Tools*
- *Ethics*

CS Foundation:

programming; data structure; algorithms ; database; etc.

GOALS FOR SOFTWARE ENGINEER

1. Minimize Cost
2. On Schedule (& Schedule Integrity)
3. Meets Functional Requirements
4. Meets Non-Functional Requirements (some calls this "quality" requirements)
   - Performance in response time, transaction time, etc.
   - Security
   - New technology (marketability)
   - Reliability
   - Availability
   - Flexibility and Future Maintainability
   - etc.

SOFTWARE QUALITY (DEFECT FREE)

- Software Quality (mainly defects) has been an on-going issue and was the main catalyst that started Software Engineering (late 1960's) when software application grew: *(from simple programs to business systems --- today; software is managing our lives)*
  - Larger (more complex) problems and products
  - More "sophisticated" effort needed to solve more complex problems
  - More people needed to understand and solve the problem
- Focus on:
  - Product Quality
  - Process Quality
  - Quality in Business ("technical value" vs "business value"- controversial)

WE FOCUS ON BOTH DELIVERABLE/PRODUCT & PROCESS/METHODOLOGIES

- What are the *software deliverables* ?
  - documents (requirements, design, test cases, training material, etc.)
  - code (source code, executables, libraries, initialized data base, test harness, etc.)
- Will our *methodology, tools, process,* etc. that we employ *produce the deliverables* ?
  1. <u>on schedule</u>
  2. <u>within cost</u>
  3. <u>meets functional requirements</u>
  4. <u>meets non-functional requirements</u>
  5. Delight the customer!

MULTIPLE PERSPECTIVE IN SOFTWARE ENGINEERING

- *Customer* cares about *cost, schedule*, and meeting *requirements*
- *Users* care about meeting the *requirements* with emphasis on *learning, usage, & recoverability* from problems, etc.
- *Developers* care about "*meeting*": *requirements, schedule, productivity/cost, technical challenges, risks & other goals/targets*.
- Sometimes Customers and Users are the same group; sometimes the customers, users and the developers all belong to the same company.

MAJOR COMPONENT OF SOFTWARE ENGINEERING

1. Understanding the Problem (System Approach - Definition)
   - What is the "total" system (hardware; software; business; people; law; technology, timing/schedule, etc.)
   - What are the boundaries
   - What are the major components of the system and how they "inter-relate"

2. Constructing the Solution (Engineering)
   - Requirements Analysis (more of problem-analysis)
   - System Design
   - Program Design
   - Coding
   - Code Integration
   - Testing
   - Product Builds
   - Product Delivery

Not all in nice sequence;
We iterate through these
(How ?)

SOFTWARE ENGINEERING TEAM

- Applications/Business Analysts - User Requirements
- Designers - System and subsystem level solutions
- Programmers - code level solutions (a "must" skill)
- QA & Testers - design and code level defect detection
- Process and Tool Specialists - version control, configuration management, packaging, process, and metrics
- Trainers - user, customer & maintenance education
- Maintenance Support - customer support, product fixes, and future enhancements

Question: Where do UI experts fit in?

CHANGES IN COMPUTING WORLD (FORCING CONTINUOUS CHANGES TO SOFTWARE ENGINEERING)

1. Speed to market is more critical
2. Continuing drop in hardware price but increasing software development cost
3. Hardware is getting more and more powerful
4. Extensive local and wide-area network (& the web)
5. Adoption of OO (any new technology)
6. GUI is the norm
7. Waterfall process model is too conservative ; looking for new process / techniques/ tools