CTIS359
Principles of Software Engineering

Software Engineering Myths & Principles

Today

- The well-known SWE Myths
  - Management Myths
  - Customer Myths
  - Practitioner's Myths
- Highlight a set of SWE principles mentioned in *Alan Davis*'s book
  - He gathered 201 principles

Software Myths

- Software myths—**erroneous beliefs** about software & the process that is used to build it—can be traced to the earliest days of computing.
- Myths have a number of attributes that make them insidious (=sinsi).
  - They appear to be reasonable statements of fact (sometimes containing elements of truth), they have an intuitive feel, and they are often promulgated (=ilan etmek, resmen duyurmak) by experienced practitioners who "know the score."
- Today, most knowledgeable SWE professionals recognize myths for what they are—misleading attitudes that have caused serious problems for managers and practitioners alike.
- However, old attitudes and habits are difficult to modify, and remnants (=kalıntı) of software myths remain.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths - Management Myths

- Managers with software responsibility, like managers in most disciplines, are often under pressure
  - to maintain budgets
  - to keep schedules from slipping
  - to improve quality.
- Like a drowning (=suda boğulmak) person who grasps at a straw, a SW manager often grasps at belief in a SW myth, if that belief will lessen the pressure (even temporarily).

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths - Management Myths

- **Myth:** *We already have a book that's full of standards & procedures for building software.*

  Won't that provide my people with everything they need to know?

- **Reality:**

Software Myths - Management Myths

- **Myth:** *We already have a book that's full of standards & procedures for building software.*

  Won't that provide my people with everything they need to know?

- **Reality:**
  - The book of standards may very well exist, but is it used?
  - Are software practitioners aware of its existence? Does it reflect modern SWE practice?
  - Is it complete? Is it adaptable?
  - Is it streamlined to improve time-to-delivery while still maintaining a focus on quality?
  - **In many cases, the answer to all of these questions is "no."**

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths - Management Myths

- **Myth:** *If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).*
- **Reality:**

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths - Management Myths

- **Myth:** *If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).*
- **Reality:**
- Software development is NOT a mechanistic (linear) process like manufacturing.
- In the words of Brooks [Bro95]: "`adding people to a late software project makes it later`."
- At first, this statement may seem counterintuitive. However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned an well-coordinated manner.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths - Management Myths

- **Myth:** *If I decide to outsource the software project to a third party, I can just relax and let that firm build it.*
- **Reality:**

Software Myths - Management Myths

- **Myth:** *If I decide to outsource the software project to a third party, I can just relax and let that firm build it.*
- **Reality:**
- If an organization does NOT understand how to manage & control software projects internally, it will invariably struggle when it outsources software projects.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Customer Myths

- A customer who requests computer software may be
  - a person at the next desk
  - a technical group down the hall
  - the marketing/sales department
  - an outside company

that has requested software under contract.

- In many cases, the customer believes myths about software because software managers & practitioners do little to correct misinformation.
- Myths lead to false expectations (by the customer) and, ultimately, dissatisfaction with the developer.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Customer Myths

- **Myth:** *A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.*
- **Reality:**

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Customer Myths

- **Myth:** *A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.*
- **Reality:**
- Although a comprehensive & stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster!!!.
- Unambiguous requirements (usually derived iteratively) are developed **only through** effective and continuous **communication** between customer and developer.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Customer Myths

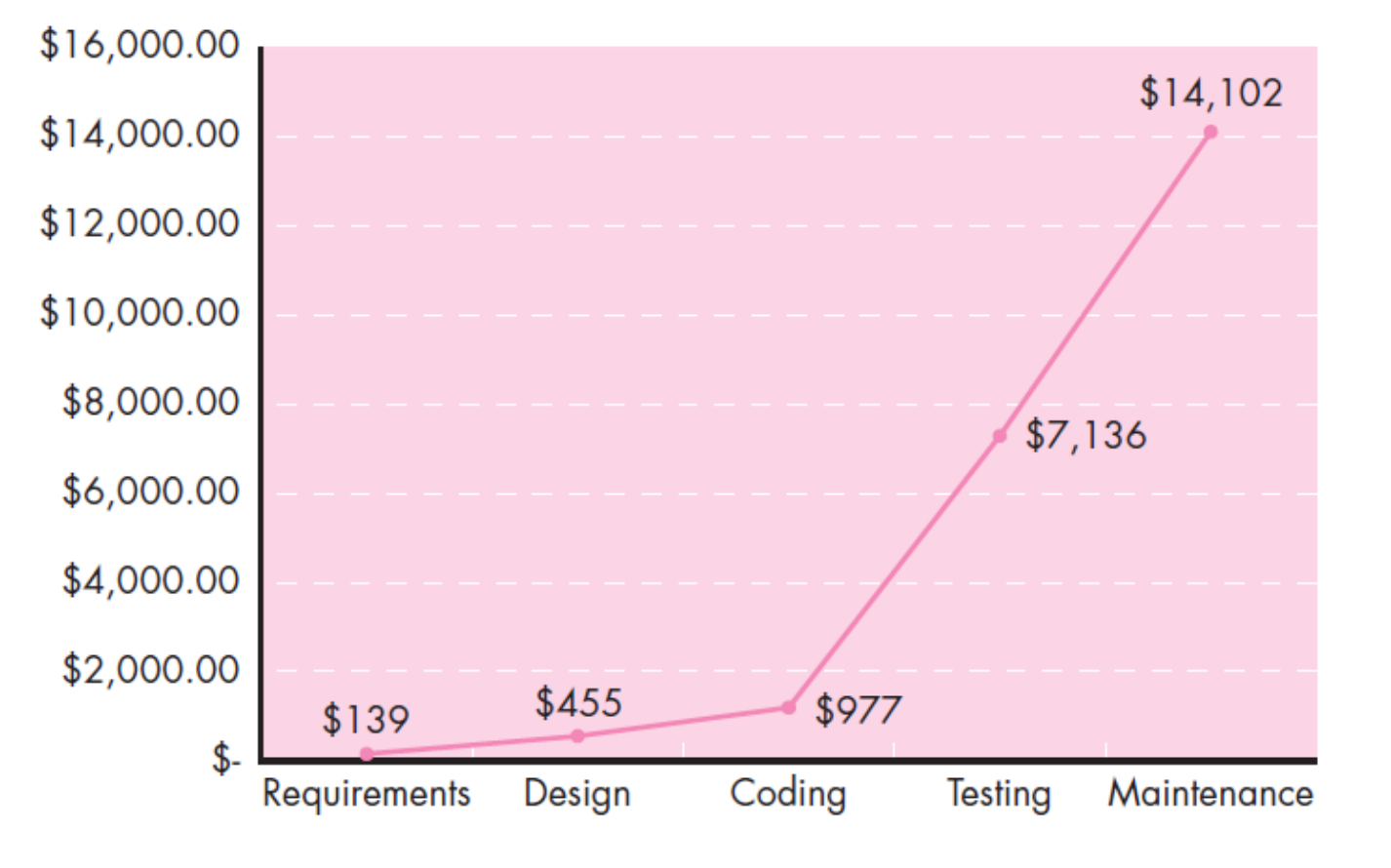
- **Myth:** *Software requirements continually change, but change can be easily accommodated because software is flexible.*
- **Reality:**

Software Myths -Customer Myths

- **Myth:** *Software requirements continually change, but change can be easily accommodated because software is flexible.*
- **Reality:**
- It is true that software requirements **change**, but the impact of change varies with the time @ which it is introduced.
- When requirements changes are requested early (before design or code has been started), the cost impact is relatively small.
- However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources & major design modification.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Customer Myths



Relative cost of correcting errors & defects
Industry average data

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- Myths that are still believed by software practitioners have been fostered by over 50 years of programming culture.
- During the early days, programming was viewed as an art form.
- Old ways & attitudes die hard.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *Once we write the program and get it to work, our job is done*.
- **Reality:**

Software Myths -Practitioner's Myths

- **Myth:** *Once we write the program and get it to work, our job is done*.
- **Reality:**
- Someone once said that "the sooner you begin 'writing code,' the longer it'll take you to get done."
- Industry data indicate that between 60 and 80 % of ALL effort expended on software will be expended **after** it is delivered to the customer for the first time.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *Until I get the program "running" I have no way of assessing its quality.*

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *Until I get the program "running" I have no way of assessing its quality.*
- **Reality:**
- One of the most effective software quality assurance mechanisms can be applied from the inception of a project—*the* ***technical review***.
- Software **reviews** are a "quality filter" that have been found to be more effective than testing for finding certain classes of software defects.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *The only deliverable work product for a successful project is the working program.*
- **Reality:**

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *The only deliverable work product for a successful project is the working program.*
- **Reality:**
- A working program is only one part of a software configuration that includes many elements.
  - A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software **support**.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.*
- **Reality:**

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

Software Myths -Practitioner's Myths

- **Myth:** *Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.*
- **Reality:**
- SWE is NOT about creating documents.
- It is about creating a **quality product**.
- Better quality leads to **reduced rework**.
- And reduced rework results in **faster delivery times**.

Source: Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim

SWE Principles

SWE Principles

- The field of SWE has matured greatly since 1970s.
- Throughout this time practitioners have **learned valuable lessons** that contribute to the **best practices** of today.
  - Some have become outdated, but many are still very relevant and widely implemented today.
- In *Alan Davis*'s book, he gathered 201 principles that form the foundation of SWE.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

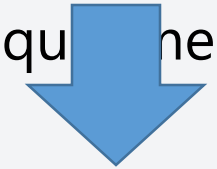
SWE Principles

- Make Quality Number 1
- High-Quality Software Is Possible
- Give Products to Customers Early
- Use an Appropriate Software Process
- Minimize Intellectual Distance
- Inspect ~~Code~~ Artifacts
- People Are the Key to Success
- … etc.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - Make Quality Number 1

- There is nothing more important than delivering a quality product to customers.
- However, different people have different ideas of what quality means, and it therefore must be specified and measured.
  - how closely software meets the customer's requirements?
  - how many (or few) defects it has?
  - how much it costs to produce?
- Quality measures need to be **specified in advance** to ensure the correct targets are being pursued and met.



Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - High-Quality SW Is Possible

- Although it may be difficult to produce high-quality software, following modern SWE methods and techniques has proven to meet reasonable quality goals.
- Examples:
  - involving the customer
  - prototyping
  - conducting inspections
  - employing incremental software processes
  - ...etc.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - Give Products to Customers Early

- Many SW projects fail because customers are given their first look at SW too late in the SDLC.
  - This was a major motivation for the introduction of agile methods.
- It's virtually impossible to know ALL the requirements in advance, and involving customers as early as possible is critical to getting the requirements right.
  - Their early involvement in helping to specify requirements is very important, but giving them working SW and having them use it is critical to understanding what they really need.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - Give Products to Customers Early

- Customers may think
  - they want a particular feature OR
  - they want a user interface to look a certain way,

but UNTIL they get a version of software to work with you can NEVER be sure.

- Employing techniques such as agile processes, prototyping, or incremental processes allow customers to get SW into their hands early in SDLC.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

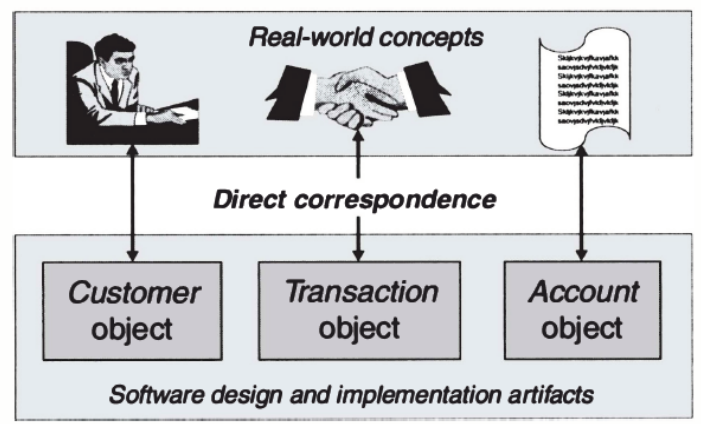
SWE Principles - Use an Appropriate Software Process

- There are many software process models, and NO single one is appropriate for EVERY TYPE of project.
  - **Ex:** The waterfall process works well for projects where all of the requirements are well known up front.
  - **Ex:** Conversely, agile and other iterative processes are called for when few requirements are known in advance.
- Good SW engineers & project leaders take the time to understand the type of project being undertaken and use an appropriate model.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - Minimize Intellectual Distance

- For any SW solution to a real-world problem, the structures of both the software solution and real-world problem should be as similar as possible.
- The closer the structures are to each other, the easier it is to develop and maintain the SW.



OO approach achieves this objective.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - Inspect ~~Code~~ Artifacts

- This should be extended to read "Inspect All Artifacts".
- Artifacts are defined as any product of the software development process including
  - technical specifications
  - test plans
  - documentation
  - code, and etc.
- Inspections have been proven to find errors as early as possible, increase quality, and decrease overall project cost.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

SWE Principles - People Are the Key to Success

- Highly skilled, motivated people are probably the most important factor contributing to the success.
  - Good people can make up for a variety of obstacles including poor tools, insufficient processes, and unforeseen problems.
  - Good people will figure out a way to overcome these obstacles and make the project a success. Poor performers without *any* of these obstacles will probably still fail.
  - Hiring and retaining the best people is critical to producing high-quality and successful software.

Source: Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011

References

1. Software Engineering: A Practitioner's Approach 8th Edition by R. Pressman, B. Maxim
2. Software Engineering: Modern Approaches 2nd Edt. E. Braude, E. Bernstein, 2011