Software Engineering I

Introduction

Dr. Elham Mahmoudzadeh
Isfahan University of Technology
mahmoudzadeh@iut.ac.ir
2022

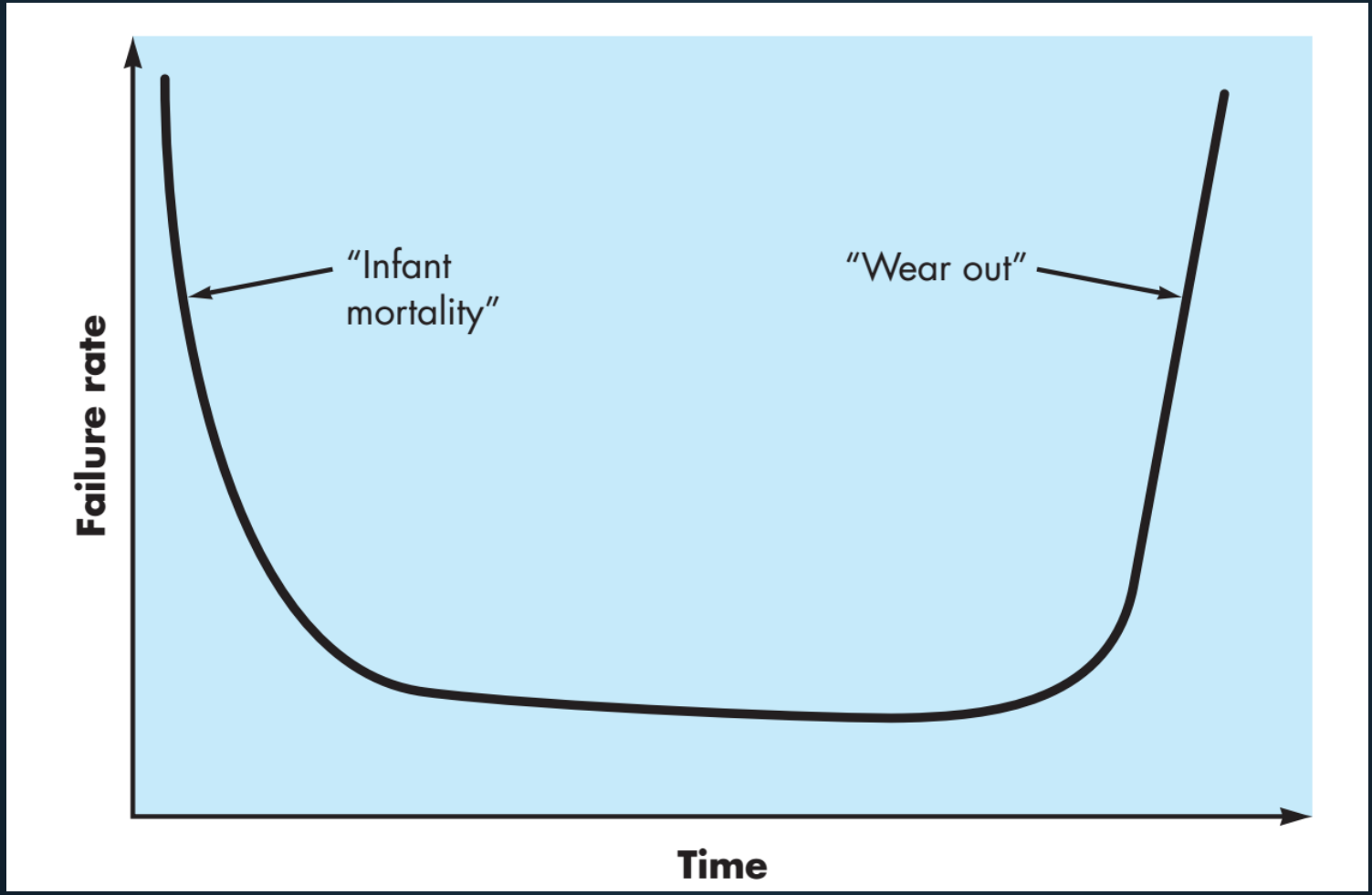
No one could foresee that software would become embedded in systems of all kinds: transportation, medical, telecommunications, military, industrial, entertainment, . . .
the list is almost endless.

What is software?

- Instructions (computer programs) that when executed provide desired features, function, and performance;
- Data structures that enable the programs to adequately manipulate information,
- Descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

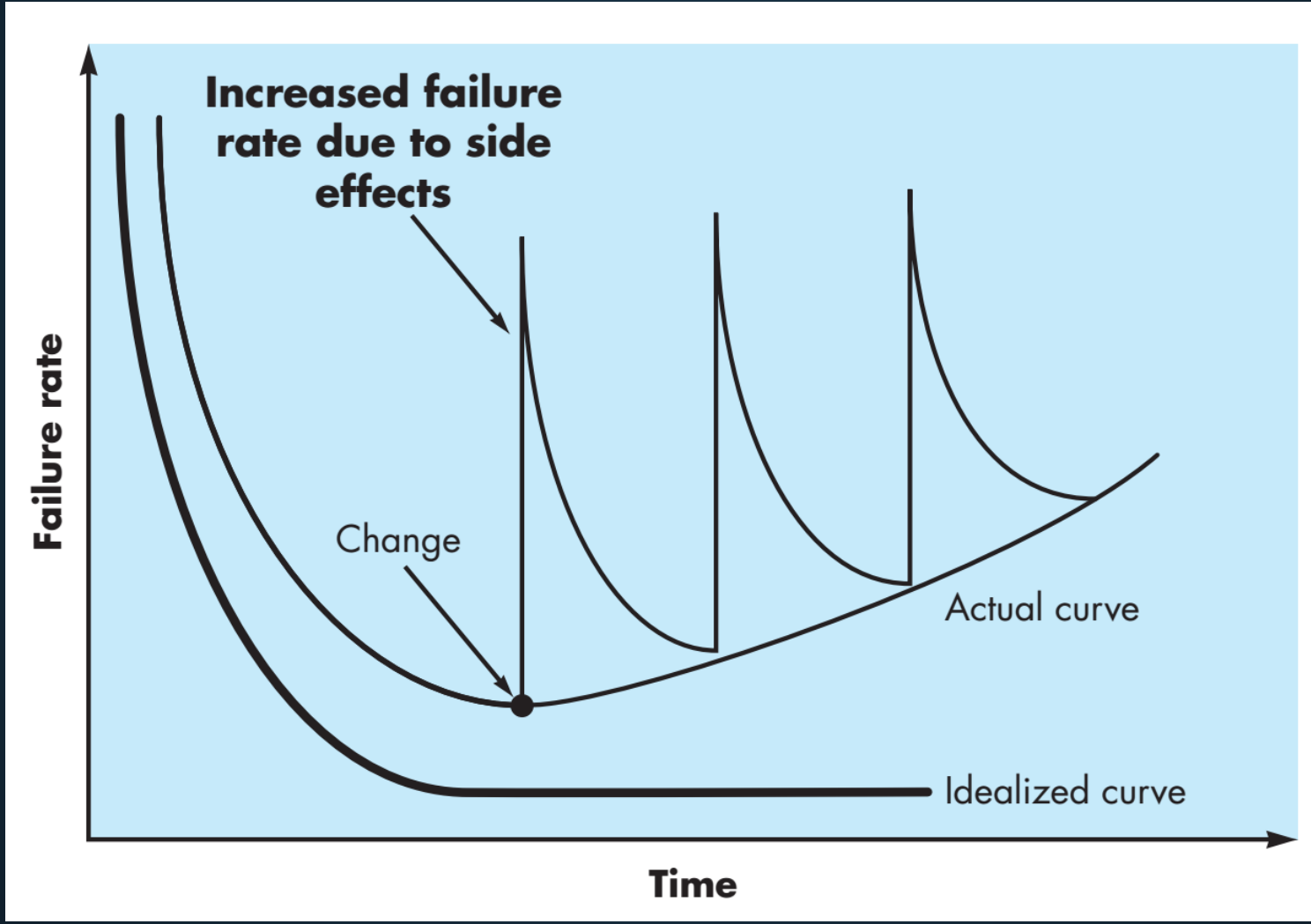
Software is a logical rather than a physical system element.

Hardware failure rate as a function of time



میزان خرابی سخت افزار به عنوان تابعی از زمان
اگر بخوایم نرم افزار با سخت افزار مقایسه بکنیم:
failure rate
اوایل سخت افزار که می خواد تولید بشه failure rate زیادی داره دیوایس ها و همینطور که اون ساخته میشه تبدیل میشه به یک دیوایسی و وقتی که ساخته میشه و خطاهاش درمیاد و از یک زمانی به بعد failure rate کمی داریم ولی بخاطر ماهیت فیزیکی اون سخت افزار از یه جایی به بعد دیگه فرسوده میشه و قابل استفاده نیست
اما نرم افزار: صفحه بعدی میشه

Software failure rate as a function of time



نرخ شکست نرم افزار به عنوان تابعی از زمان:
نرم افزار موجودیت منطقی داره و مثل سخت افزار نیست ینی قسمت فرسودگی و قابل استفاده نیست رو اینجا نداریم و ایده الش اینه که بخاطر موجودیت منطقی بودنش
اولش که داریم کار میکنیم خطاها است و حالت ایده الش اینه که اون نرم افزار تا بی نهایت کار بکنه ما در عمل و در واقعیت در مقایسه با سخت افزار کهنه و فرسوده نمیشه ولی در عمل اون اتفاقی که می افته اینه که بخاطر اینکه این نرم افزار داره کار میکنه طبیعتا یکسری تغییراتی توش به وجود میاد پس تغییراتی داخلش هست - این تغییرات باید اپلای بشه روی سیستم و این اپلای شدنه یک جایی اثر خودشو می ذاره ینی این اپلای شدنه باید تست بشه یا ...
اگر این تغییرات زیاد باشه ینی شوک هایی که به نرم افزار بیاد زیاد باشه در بی نهایت نرم افزار مارو از سناریوی اصلی خودش باز میکنه ینی دیگه اصل سیستم می تونه نباشه
پس نرم افزار در حالت ایده الش اینه که هیچ فرسودگی نداشته باشه ولی ماهیت تغییر اینو داره که باعث بشه نرم افزار ما فرسوده بشه ینی نرم افزار ما به سمتی می ره که دیگه اون اصلیه نیست تا حد ممکن باید شیبمون کند باشه ینی این curve ما با شیب کند اتفاق بیوفته ینی در زمان بیشتری نرم افزار پاسخگوی نیاز باشه حول یک محوری ینی رسالت اصلی خودشو داشته باشه
نکته: ولی وقتی شوک خیلی زیادی به سیستم وارد میکنیم احتمال اینکه از رسالت اصلی خودش دور بشه خیلی زیاده پس باید سعی کنیم تا حد ممکن این شیب کند باشه

Challenges

- Characteristics of software
  - Intangible
  - Changeable

- Characteristics of software development
  - Human-intensive
  - Multi-disciplinary

-
چالش هایی که داریم اینجا:
یکیش ماهیت خود نرم افزار است و یکی هم رویکرد تولید نرم افزار
درباره خود نرم افزار --> تولیدش نامحسوس است بالا تر گفتیم نرم افزار منطقی است و فیزیکی نیست : رشد تولید نرم افزار نامحسوس است ینی ممکنه 1000 خط کد زده باشیم ولی نرم افزار کار نکنه و از دید مشتری این ینی هیچ
قابل تغییر بودن: همه چیز در حال تغییر است و تغییرات خیلی زیاده توی نرم افزار پس باید قابل تغییر باشه
رویکرد تولید نرم افزار:
از طرفی نرم افزار را واسه یک مشتری میخوایم پس جنبه های انسانی رو باید براش در نظر بگیریم - خود رویکرد تولید نرم افزار یک کار تیمی می خواد - انسان رو عامل مهم انجام کار در نظر می گیرن
بین رشته ای بودن مثلا سیستمی می خوایم که ارتباط بین پزشکان را فراهم بکنه پس بین رشته ای بودن و تک بعدی نبودن هم از چالش های نرم افزار است مثلا در رابطه با مثال بالا باید خیلی چیزا از بیمارستان بودنیم که بتونیم تولید بکنیم

Software Failures – Main Reasons

- ***Increasing demands***
  - Systems have to be built and delivered more quickly;
  - Larger, even more complex systems are required;
  - Systems have to have new capabilities that were previously thought to be impossible.
  - Existing software engineering methods cannot cope and new software engineering techniques have to be developed to meet these new demands.

- ***Low expectations***
  - It is relatively easy to write computer programs without using software engineering methods and techniques.
  - Many companies do not use software engineering methods. Consequently, their software is often more expensive and less reliable than it should be.

We need better software engineering education and training to address this problem.

Software Engineering is NOT Programming!

Programming Vs. Engineering

Programming	Software Engineering
Personal activity (instrument)	Team activity (orchestra)
One aspect of software development	Large systems must be developed similar to other engineering practices
Concerned about accomplishing the objective of the program itself	Concerned about the entire solution, its feasibility, and future use

This slide is brought from “SOFTWARE ENGINEERING course, Dr. Garousi and Dr. Tarhan, 2017.

Professional Software Development

- Professional software, intended for use by someone apart from its developer, is usually developed by teams rather than individuals. It is maintained and changed throughout its life.

- Software engineering is intended to support professional software development, rather than individual programming. It includes techniques that support program specification, design, and evolution.

Professional Software Development(Cnt'd)

- Many people think that software is simply another word for computer programs. However, software is not just the programs themselves but also all associated documentation and configuration data that is required to make these programs operate correctly.
  - A professionally developed software system usually consists of system documentation, which describes the structure of the system; user documentation, which explains how to use the system.
- This is one of the important differences between professional and amateur software development.

Software Products

- Software engineers are concerned with developing software products.
- Kinds of software products
  - *Generic products: The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.*
  - *Customized products: The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.*

Software Engineering(I)

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
  - *Engineering discipline - engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work to organizational and financial constraints so they look for solutions within these constraints.*
  - *All aspects of software production - software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software production.*

Software Engineering(II)

- Doing the right thing
  - Software that users want and need
  - Software that benefits society
- Doing the thing right
  - Following a good software process
  - Developing your programming skills

Software Engineering(III)

IEEE Computer Society Definition:
“Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.”

Engineering Discipline

Engineering is about getting repeatable results of the required quality within the schedule and budget.

- This often involves making compromises—engineers cannot be perfectionists.
- People writing programs for themselves, however, can spend as much time as they wish on program development.

In general, software engineers adopt a systematic and organized approach to their work, as this is often the most effective way to produce high-quality software.

- However, engineering is all about selecting the most appropriate method for a set of circumstances so a more creative, less formal approach to development may be effective in some circumstances.

General Issues of software

- *Heterogeneity* increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices. As well as running on general-purpose computers, software may also have to execute on mobile phones.
- You often have to integrate new software with older legacy systems written in different programming languages.
- The challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity.

General Issues of software(Cnt'd)

- *Business and social change* business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
  - Many traditional software engineering techniques are time consuming and delivery of new systems often takes longer than planned. They need to evolve so that the time required for software to deliver value to its customers is reduced.
- *Security and trust* as software is intertwined with all aspects of our lives, it is essential that we can trust that software. This is especially true for remote software systems accessed through a web page or web service interface.
  - We have to make sure that malicious users cannot attack our software and that information security is maintained.

Software Engineering Diversity

- Software engineering is a systematic approach to the production of software that takes into account practical cost, schedule, and dependability issues, as well as the needs of software customers and producers.
  - How this systematic approach is actually implemented varies dramatically depending on the organization developing the software, the type of software, and the people involved in the development process.
  - There are no universal software engineering methods and techniques that are suitable for all systems and all companies. Rather, a diverse set of software engineering methods and tools has evolved over the past 50 years.

Software Engineering Diversity(Cnt'd)

- You use different software engineering techniques for each type of system because the software has quite different characteristics.
  - For example, an embedded control system in an automobile is safety-critical and is burned into rom when installed in the vehicle. It is therefore very expensive to change. Such a system needs very extensive verification and validation so that the chances of having to recall cars after sale to fix software problems are minimized. User interaction is minimal (or perhaps nonexistent) so there is no need to use a development process that relies on user interface prototyping.
  - For a web-based system, an approach based on iterative development and delivery may be appropriate, with the system being composed of reusable components.

Software Engineering Diversity(Cnt'd)

- There are software engineering fundamentals that apply to all types of software systems.
  - They should be developed using a managed and understood development process. The organization developing the software should plan the development process and have clear ideas of what will be produced and when it will be completed. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of systems. Software should behave as expected, without failures and should be available for use when it is required. It should be safe in its operation and, as far as possible, should be secure against external attack. The system should perform efficiently and should not waste resources.

Software Engineering Diversity(Cnt'd)

- Understanding and managing the software specification and requirements (what the software should do) are important.
  - You have to know what different customers and users of the system expect from it and you have to manage their expectations so that a useful system can be delivered within budget and to schedule.

- You should make as effective use as possible of existing resources. This means that, where appropriate, you should reuse software that has already been developed rather than write new software.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

This slide is brought from Somerville, I., “Software Engineering”, 2014

ویژگی های اساسی نرم افزار خوب
ویژگی محصول یا Product characteristic Description:
1- قابلیت نگهداری
Description یا شرح:
نرم افزار باید به گونه ای نوشته شود که بتواند برای پاسخگویی به نیازهای متغیر مشتریان تکامل یابد. این یک ویژگی حیاتی است زیرا تغییر نرم افزار یک نیاز اجتناب ناپذیر یک محیط تجاری در حال تغییر است.
2- قابل اعتماد بودن و امنیت:
قابلیت اطمینان نرم افزار شامل طیف وسیعی از ویژگی ها از جمله قابلیت اطمینان، امنیت و ایمنی است. نرم افزار قابل اعتماد نباید در صورت خرابی سیستم باعث آسیب فیزیکی یا اقتصادی شود. کاربران مخرب نباید بتوانند به سیستم دسترسی داشته باشند یا به آن آسیب برسانند.
3- کارایی:
نرم افزار نباید از منابع سیستمی مانند چرخه های حافظه و پردازنده بیهوده استفاده کند. بنابراین کارایی شامل پاسخگویی، زمان پردازش، استفاده از حافظه و غیره است.
4- مقبولیت:
نرم افزار باید برای نوع کاربرانی که برای آن طراحی شده است قابل قبول باشد. این بدان معنی است که باید قابل درک، قابل استفاده و سازگار با سایر سیستم هایی باشد که آنها استفاده می کنند.

Safety and reliability

- Safety and reliability are related but distinct
  - In general, reliability is necessary but not sufficient conditions for system safety.
- Reliability is concerned with conformance to a given specification and delivery of service.
  - The probability of failure-free system operation over a specified time in a given environment for a given purpose
- Safety is concerned with ensuring system cannot cause damage irrespective of whether or not it conforms to its specification.
  - System reliability is essential for safety but is not enough

General principles

1. *The Reason It All Exists*
   - A software system exists for one reason: *to provide value to its users*.
2. *Keep It Simple, Stupid!*
   - *All design should be as simple as possible, but no simpler*.
3. *Maintain the Vision*
   - *A clear vision is essential to the success of a software project*.
4. *What You Produce, Others Will Consume*
   - Specify with an eye to the users. Design, keeping the implementers in mind. Code with concern for those that must maintain and extend the system.

General principles(Cnt’d)

5. *Be Open to the Future*
   - A system with a long lifetime has more value.
   - These systems must be ready to adapt to the changes.
6. *Plan Ahead for Reuse*
   - *It reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated*.
7. *Think!*
   - *Placing clear, complete thought before action almost always produces better results*.

References

- **Dennis, Wixon, Tegarden, “System Analysis and Design, An Object Oriented Approach with UML”, 5th Edition, 2015.**
- **https://www.vgarousi.com, accessed on 28th September, 2020.**
- Sommerville, I., “Software Engineering”, 10th Edition, 2015.

What we will talk about next...

- How to write a proposal.
- Introduction about System, Software Development Life Cycle(SDLC).