What is Software Engineering Anyway?

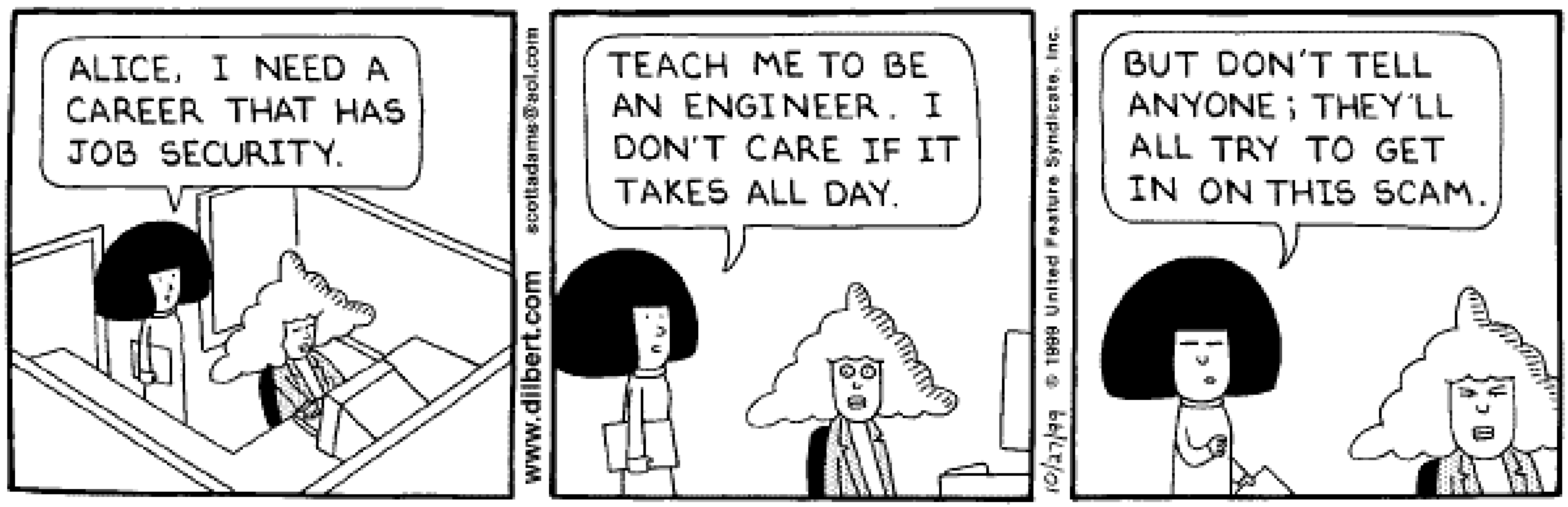
A Reflection of 50 years of Software Engineering & The Road Ahead!

Dr. Sridhar Chimalakonda
Department of Computer Science & Engineering
Indian Institute of Technology, Tirupati, India

RISHA →

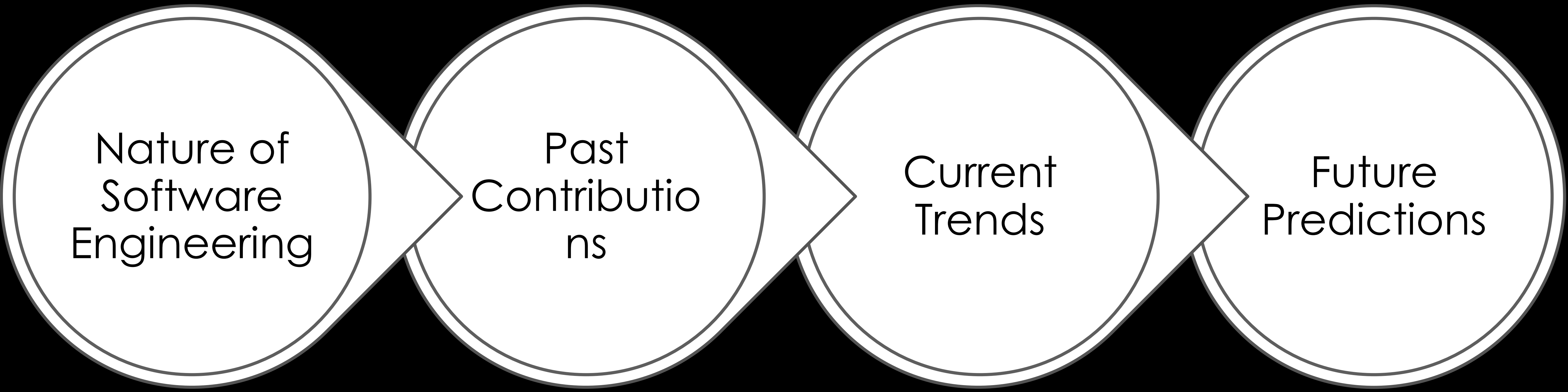
Research in Intelligent Software & Human Analytics Lab

Why are you here?


ALICE, I NEED A CAREER THAT HAS JOB SECURITY.
TEACH ME TO BE AN ENGINEER. I DON'T CARE IF IT TAKES ALL DAY.
BUT DON'T TELL ANYONE; THEY'LL ALL TRY TO GET IN ON THIS SCAM.
scottadams@aol.com
www.dilbert.com
© 1999 United Feature Syndicate, Inc.
10/27/99

What's your goal?

What we will do?



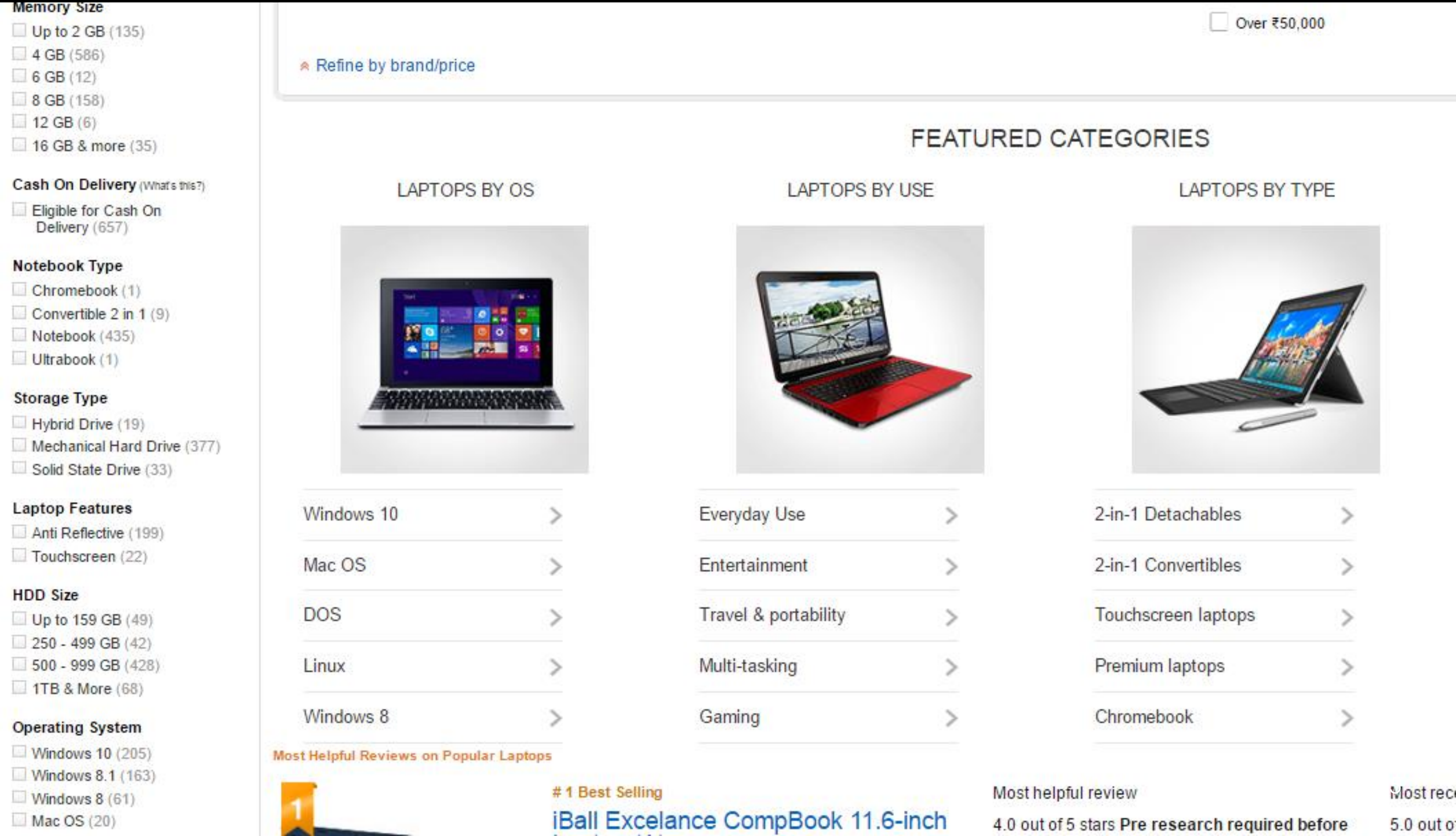
Nature of Software Engineering

Past Contributions

Current Trends

Future Predictions

Which laptop will you buy?



Memory Size

- □ Up to 2 GB (135)
- □ 4 GB (586)
- □ 6 GB (12)
- □ 8 GB (158)
- □ 12 GB (6)
- □ 16 GB & more (35)

Cash On Delivery (What's this?)

- □ Eligible for Cash On Delivery (657)

Notebook Type

- □ Chromebook (1)
- □ Convertible 2 in 1 (9)
- □ Notebook (435)
- □ Ultrabook (1)

Storage Type

- □ Hybrid Drive (19)
- □ Mechanical Hard Drive (377)
- □ Solid State Drive (33)

Laptop Features

- □ Anti Reflective (199)
- □ Touchscreen (22)

HDD Size

- □ Up to 159 GB (49)
- □ 250 - 499 GB (42)
- □ 500 - 999 GB (428)
- □ 1TB & More (68)

Operating System

- □ Windows 10 (205)
- □ Windows 8.1 (163)
- □ Windows 8 (61)
- □ Mac OS (20)

□ Over ₹50,000

Refine by brand/price

FEATURED CATEGORIES

LAPTOPS BY OS	LAPTOPS BY USE	LAPTOPS BY TYPE
Windows 10 ›	Everyday Use ›	2-in-1 Detachables ›
Mac OS ›	Entertainment ›	2-in-1 Convertibles ›
DOS ›	Travel & portability ›	Touchscreen laptops ›
Linux ›	Multi-tasking ›	Premium laptops ›
Windows 8 ›	Gaming ›	Chromebook ›

Most Helpful Reviews on Popular Laptops

1 Best Selling

iBall Excelance CompBook 11.6-inch

Most helpful review

4.0 out of 5 stars **Pre research required before**

Most rec

5.0 out o

Which mobile/speakers will you buy?

Beats

Bose

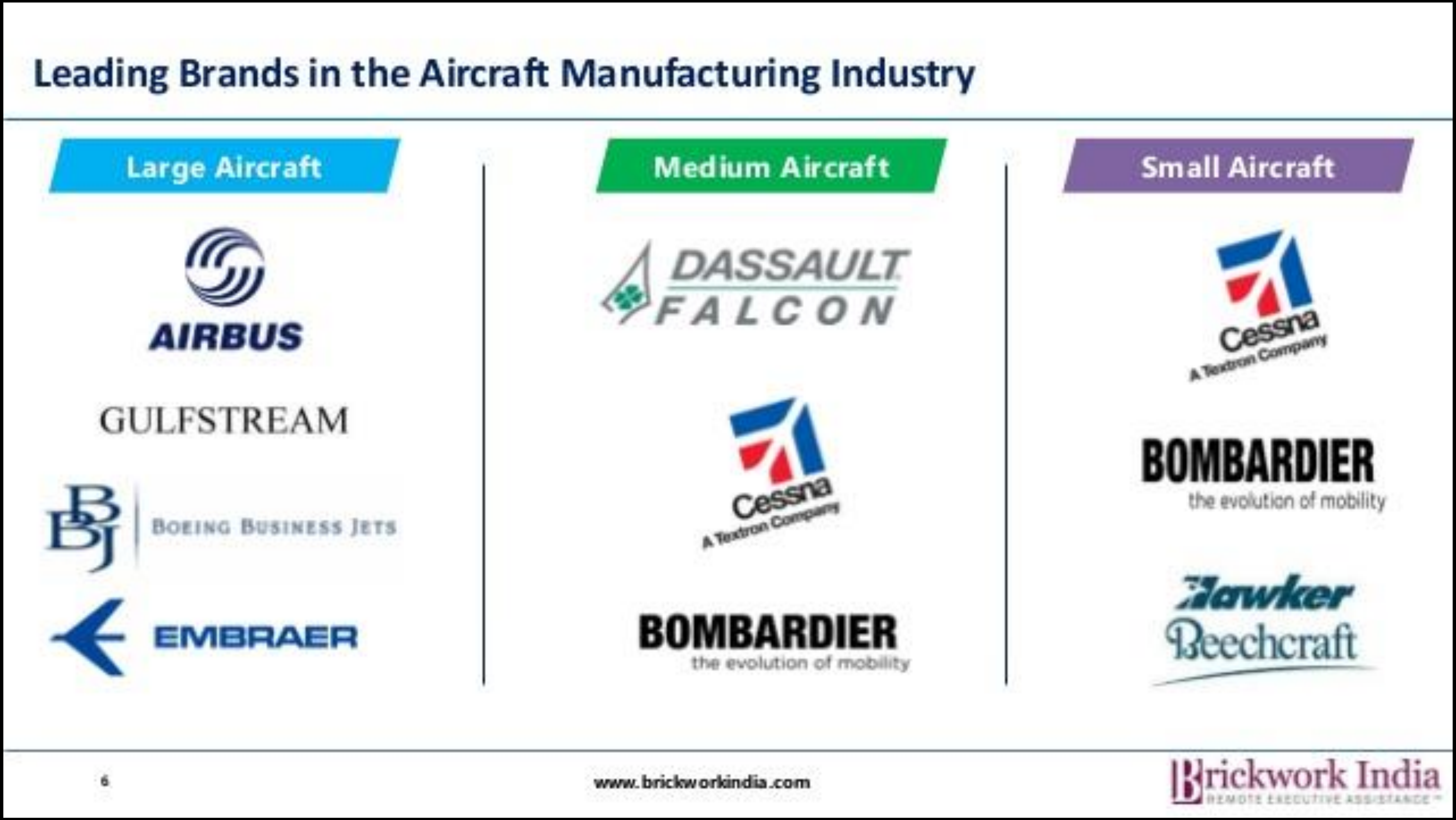
LG

Sony

[redacted]

JVC

Which plane do you fly?



Leading Brands in the Aircraft Manufacturing Industry

Large Aircraft	Medium Aircraft	Small Aircraft
AIRBUS	DASSAULT FALCON	Cessna A Textron Company
GULFSTREAM	Cessna A Textron Company	BOMBARDIER the evolution of mobility
BBJ BOEING BUSINESS JETS	BOMBARDIER the evolution of mobility	Hawker Beechcraft
EMBRAER		

Warranty

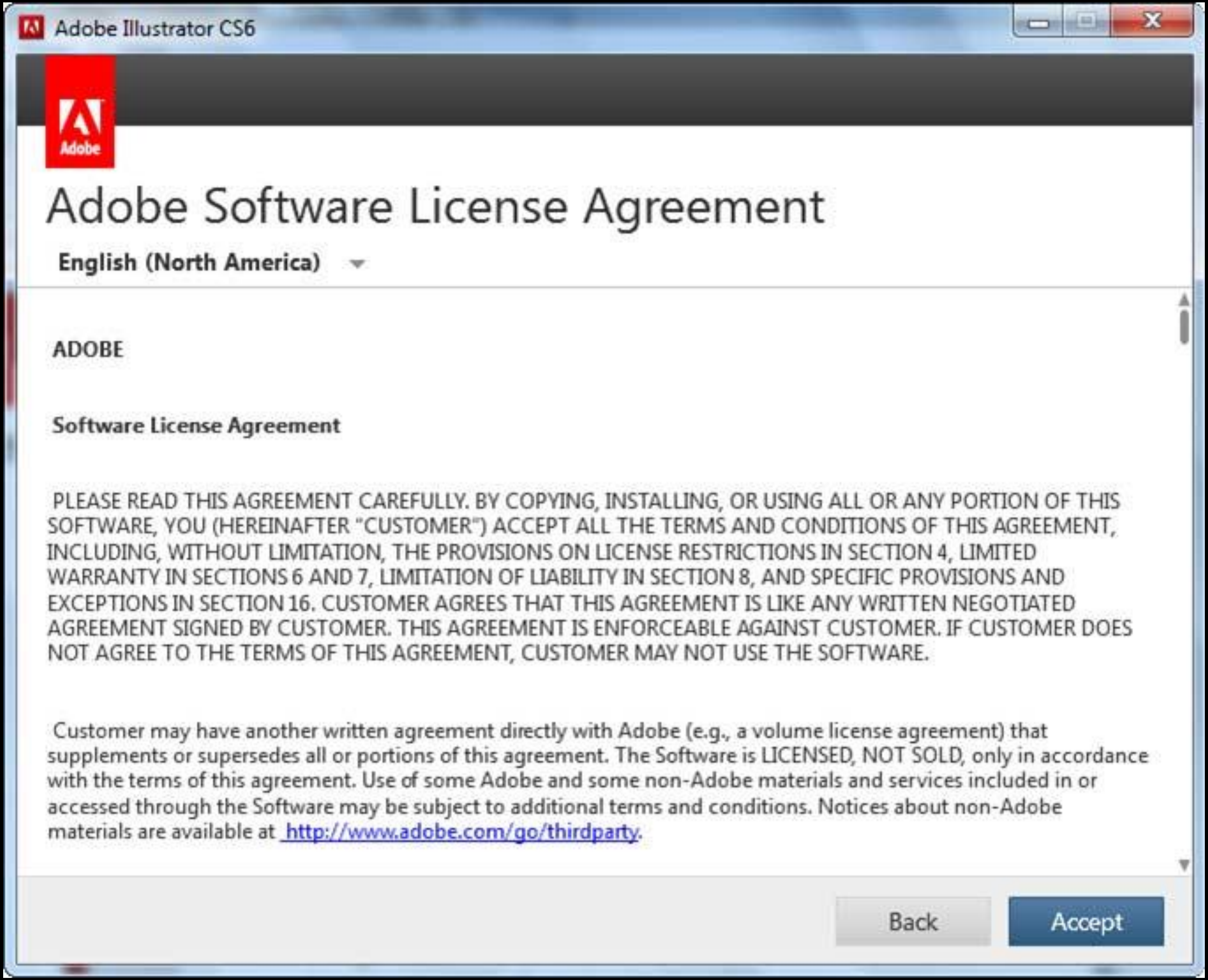
5 year
TOYOTA
WARRANTY

TOYOTA
GENUINE PARTS
TWO YEAR
WARRANTY
40,000 KM

BRANDS WE CARRY
GE
PHILIPS
SIEMENS
TOSHIBA

Quality

{*Software Quality*}



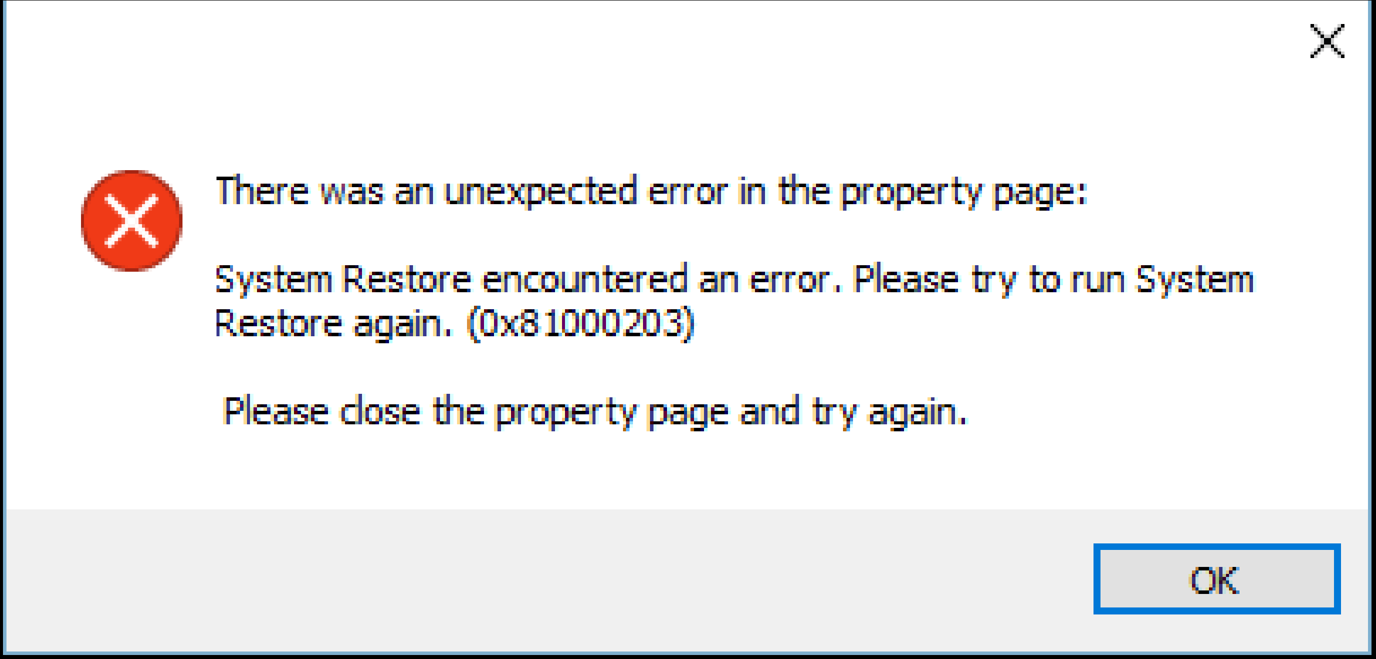
Adobe Software License Agreement

ADOBE

Software License Agreement

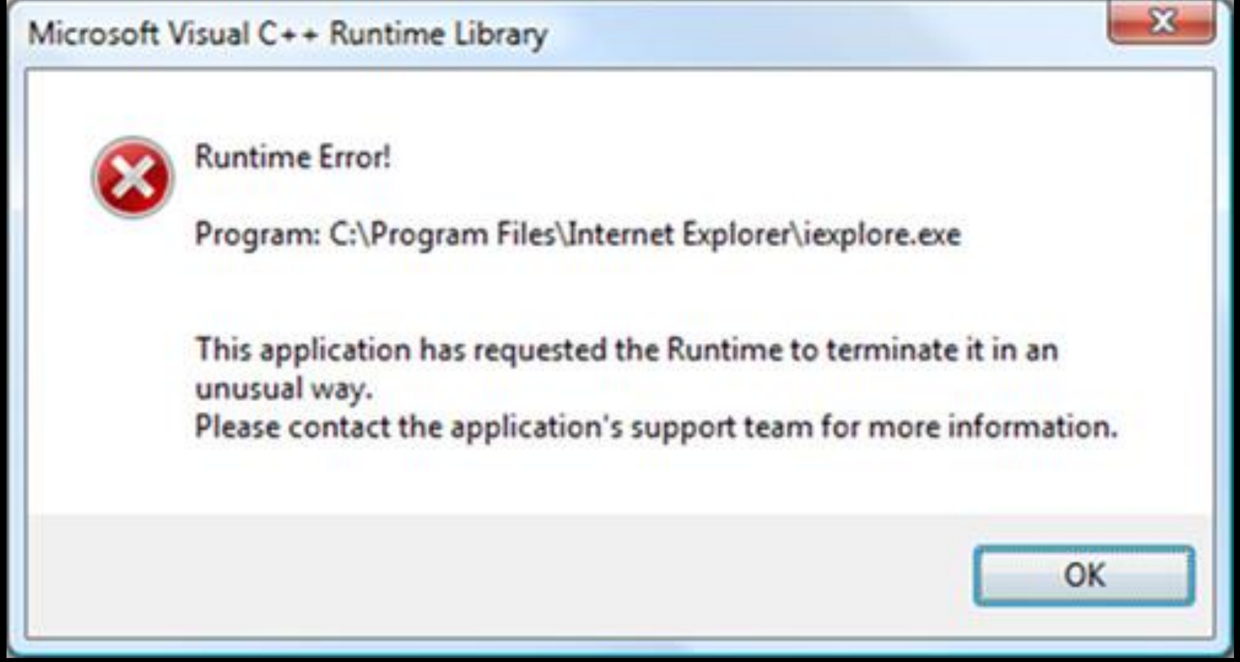
PLEASE READ THIS AGREEMENT CAREFULLY. BY COPYING, INSTALLING, OR USING ALL OR ANY PORTION OF THIS SOFTWARE, YOU (HEREINAFTER "CUSTOMER") ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT, INCLUDING, WITHOUT LIMITATION, THE PROVISIONS ON LICENSE RESTRICTIONS IN SECTION 4, LIMITED WARRANTY IN SECTIONS 6 AND 7, LIMITATION OF LIABILITY IN SECTION 8, AND SPECIFIC PROVISIONS AND EXCEPTIONS IN SECTION 16. CUSTOMER AGREES THAT THIS AGREEMENT IS LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY CUSTOMER. THIS AGREEMENT IS ENFORCEABLE AGAINST CUSTOMER. IF CUSTOMER DOES NOT AGREE TO THE TERMS OF THIS AGREEMENT, CUSTOMER MAY NOT USE THE SOFTWARE.

Customer may have another written agreement directly with Adobe (e.g., a volume license agreement) that supplements or supersedes all or portions of this agreement. The Software is LICENSED, NOT SOLD, only in accordance with the terms of this agreement. Use of some Adobe and some non-Adobe materials and services included in or accessed through the Software may be subject to additional terms and conditions. Notices about non-Adobe materials are available at http://www.adobe.com/go/thirdparty.


There was an unexpected error in the property page:

System Restore encountered an error. Please try to run System Restore again. (0x81000203)

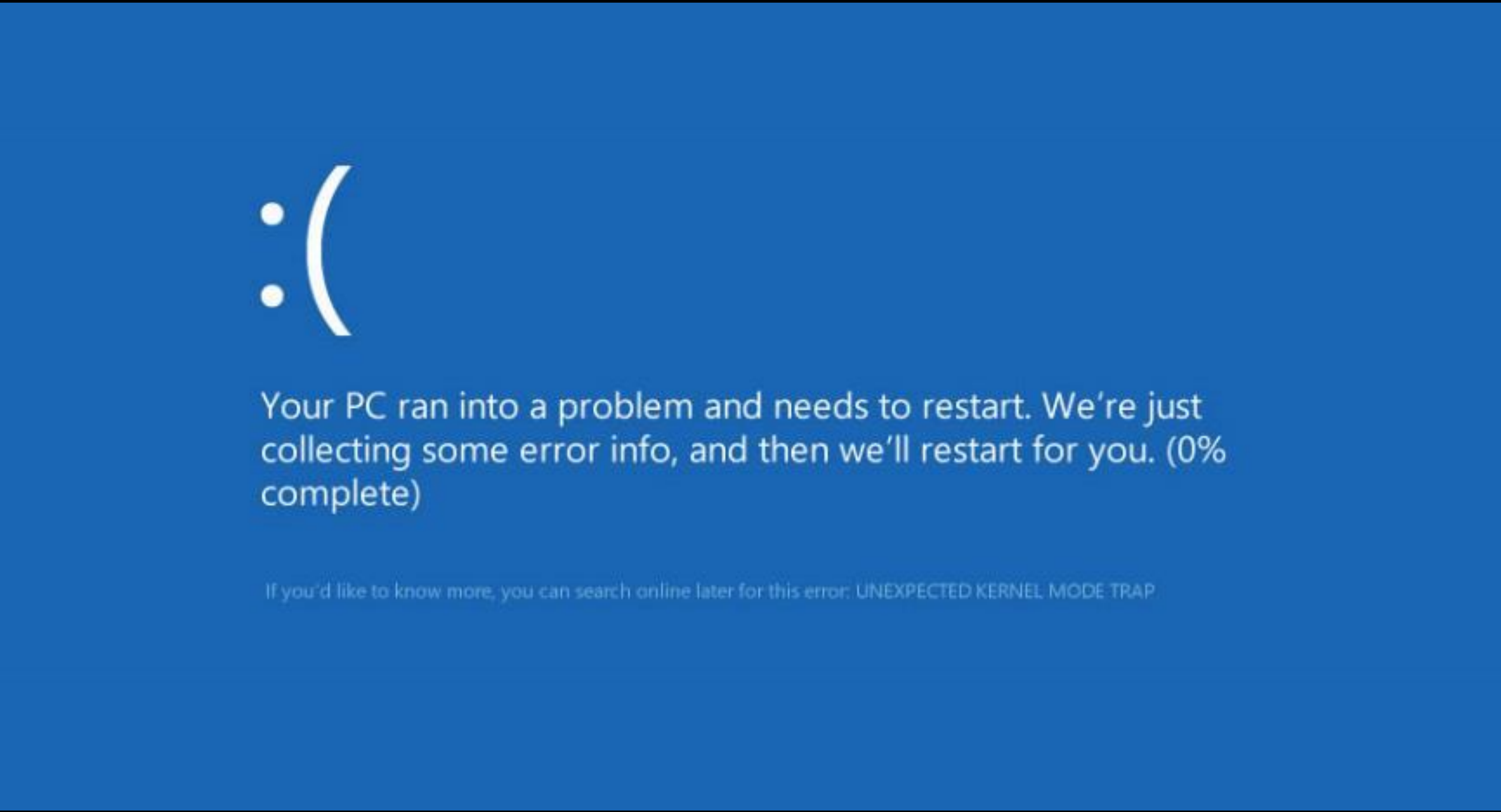
Please close the property page and try again.


Microsoft Visual C++ Runtime Library

Runtime Error!

Program: C:\Program Files\Internet Explorer\iexplore.exe

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.


:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: UNEXPECTED KERNEL MODE TRAP

Software is Ubiquitous
Pervasive!

State of software today?

- Size of software industry - $466 billion dollars?
- Size of software? – millions of lines of code
- Number of organizations that use software?
- Number of software companies?
- Number of software developers?
- Volume of data that is generated for software?

Source: https://www.statista.com/forecasts/963597/software-revenue-in-the-world

Slide Dec 1:
Q Key Diff between SE and Programmer(Table) + engineer/masion example
1)Inputs and Outputs known vs unkown
2)Builds programms for personal use. vs builds large scale non trivial software intensive systems for industy.
3)Work gets finished when goals are fullfilled vs ever changing new requirements of the user and meeting their ever increasing expectations.
4) Can follow exploratory adhock approach vs following a well organized systematic approach.
4)Poor Documentation vs Well documented + Code is written with mentainablity in mind.
5) Designed by single person for personal use. vs Designed and Developed by a team of develepors for large scale use.

Q List 3 signficant contributions in SE from(Nato 1968 1969 discussion) which are challanges today

1)Software Crisis and Complexity: The conferences highlighted the growing complexity of software systems and the "software crisis," where projects were often delivered late, over budget, and with defects. This challenge persists today as software systems become increasingly complex and integrated.

2)Software Engineering as a Discipline: The conferences advocated for treating software development as an engineering discipline, emphasizing systematic approaches, rigorous methodologies, and best practices. Despite advances, the challenge of ensuring consistent, high-quality software development processes remains.

3)Proving correctness of programms: Djisktra Highlighted the importance of making reliable software that could be proven correct to ensure reliability of critical used software

a new outlook to Human Computer Interaction

Q List 3 Future trends of SE
1) We expect to see programming languages becoming more close to natural language (CMU prof project shown in lectures)
2) We will see increasing work done in data security as big data becomes bigger & peoples need and expectation becomes bigger.
3) AI for software engineering.
4) Quantum Computing.
What is Software Engineering anyway? -> write 3 definitions of SE Include one byChimalakonda

1)is a systematic approach for the design of non-trivial software-intensive systems with desired qualities under given constraints
2) It is a discipline that deals with design of fault free software delivered on time which satisifieses user needs and within user budget.
3)Multi-person construction of multi-version software.”

What are the key decesions for android project which must be avoided and are absolutely wrong.
1) Not preffering cross platform development.-> for a dontaion app we focussed more on performance and not on approaching a large audiance therby we preffered native development
2) We made the app too complex by adding relativly complex features for in-experienced old aged groups making it harder for less tech driven people reducing its accesablilty.
3) We did not handle sms recipt generation after transaction was complete since it costed money which is a big no for Real life dev.
3 Coding activities where chatgpt fails horribly
What skills of a SE are automated / will be automated by AI tools and why??

Slide Dec 2:
2 Primary differences between Verification and Validation give example for each
Explain the Kano Model of qualities for a software chat application
Draw table to list Garvins 5 perspectives of Software Quality, explain each through example of bankinng software.
Explain various kind of classification and categories of software qualities? Write an example for each for bookmyshow platform.

1968

The NATO Conferences

"The major cause [of the software crisis] is … that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."

SOFTWARE ENGINEERING

[illegible]

[illegible]

REPORT ON A CONFERENCE SPONSORED BY THE
NATO SCIENCE COMMITTEE
ROME, ITALY, 27th to 31st OCTOBER 1969

SOFTWARE ENGINEERING
TECHNIQUES

CHAIRMAN : PROFESSOR P. ERCOLI
CO-CHAIRMAN : PROFESSOR DR. F.L. BAUER

EDITORS : J.N. BUXTON and B. RANDELL

APRIL 1970

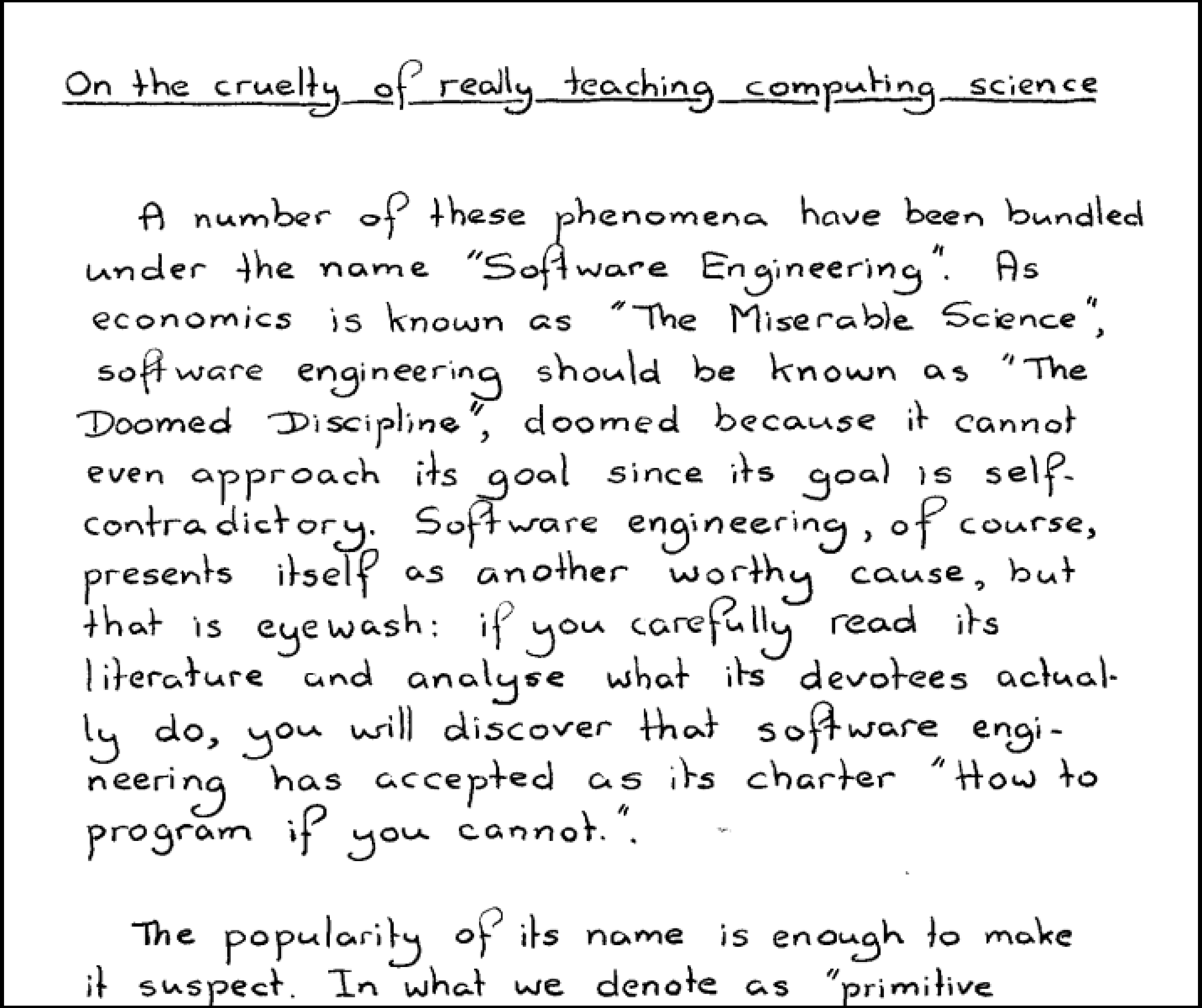
Memoirs from NATO

- Doug McIlroy on mass-produced software components
- “We build systems like the Wright brothers built airplanes — build the whole thing, push it off the cliff, let it crash, and start over again”
- “We are starting gradually, and building up. My motto is ‘do something small, useful, now.”

- Computer science should focus on making it a precise mathematical science, and leave the rest to others
    - correctness concerns and efficiency concerns
    - the design and use of notations, tailored to one’s manipulative needs
- “Testing can only reveal the presence of bugs, not their absence”

“Software Engineering as It Should Be” to “My Hopes for Computing Science.”
Edsger Dijkstra

The Software Engineering Discipline



On the cruelty of really teaching computing science

A number of these phenomena have been bundled under the name "Software Engineering". As economics is known as "The Miserable Science", software engineering should be known as "The Doomed Discipline", doomed because it cannot even approach its goal since its goal is self-contradictory. Software engineering, of course, presents itself as another worthy cause, but that is eyewash: if you carefully read its literature and analyse what its devotees actually do, you will discover that software engineering has accepted as its charter "How to program if you cannot.".

The popularity of its name is enough to make it suspect. In what we denote as "primitive

What is this Discipline?

- Craftsmanship?
- Engineering?
- Science?
- Manufacturing?
- Discipline?

- Not Programming?
- Not Computer Science?
- Not Management?
- Not Science?
- Not Mathematics?

Why is Software Engineering Challenging?

- *Ill*-formed Vs Well Formed problems
- No physical artifacts
- Lack of clarity
- Mind boggling complexity
- Failures often but not tolerable
- Change is expected rapidly

- Explicit Versus Implicit
- Tangible Versus Intangible
- Manageable Complexity Versus Unmanageable Complexity
- Changeable Environment Versus Unchangeable Environment
- No Major Changes Versus Major Changes

What is Software Engineering anyway?

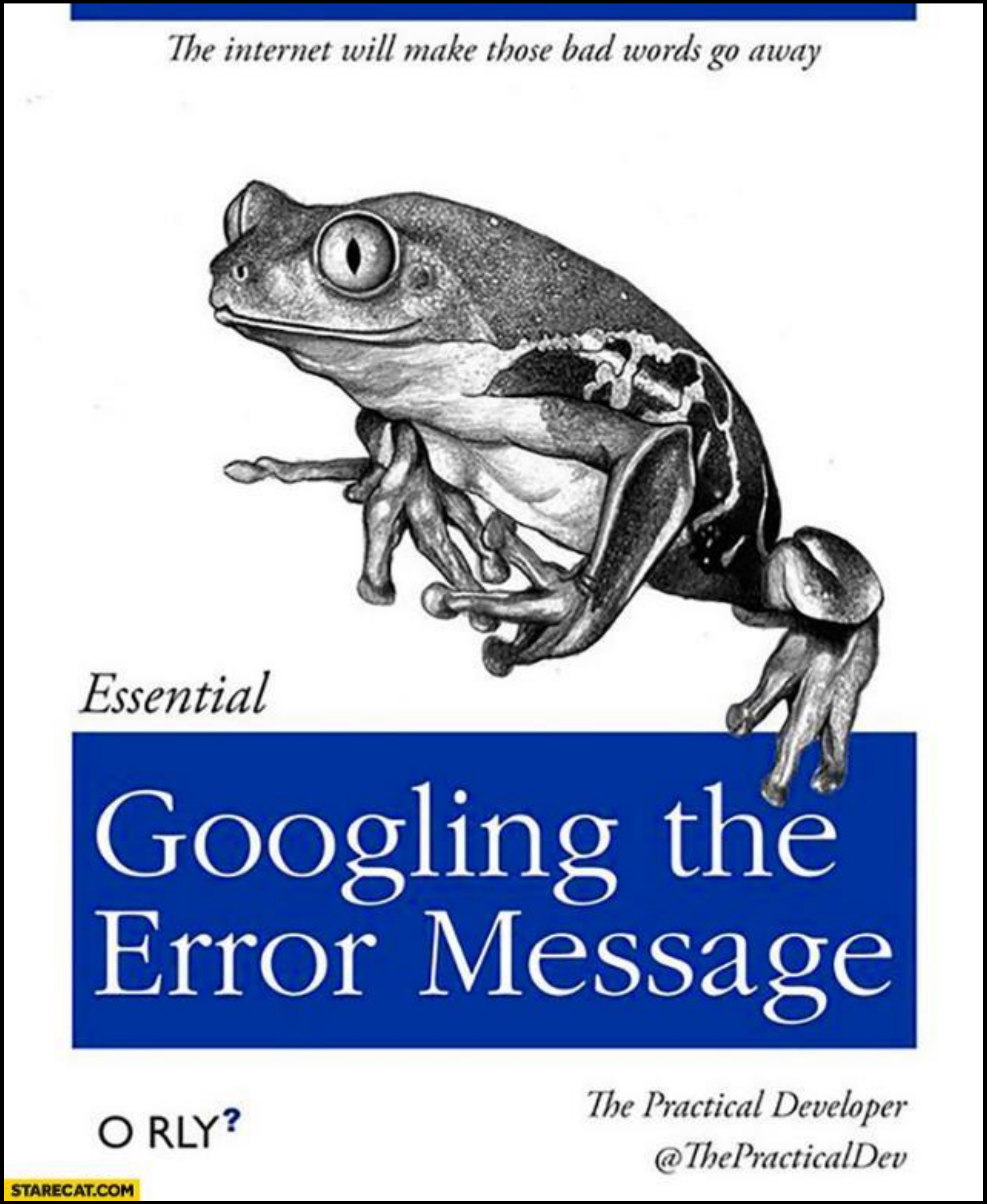
- “A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers.” [Ghezzi, Jazayeri, Mandrioli]

- “Multi-person construction of multi-version software.” [Parnas]

- “A discipline whose aim is the production of fault-free software, delivered on-time and within budget, that satisfies the user’s needs.” [Schach]

- “is a systematic approach for the design of non-trivial software-intensive systems with desired qualities under given constraints” [Chimalakonda]

What is Software Engineering
A Programmer's View



The internet will make those bad words go away

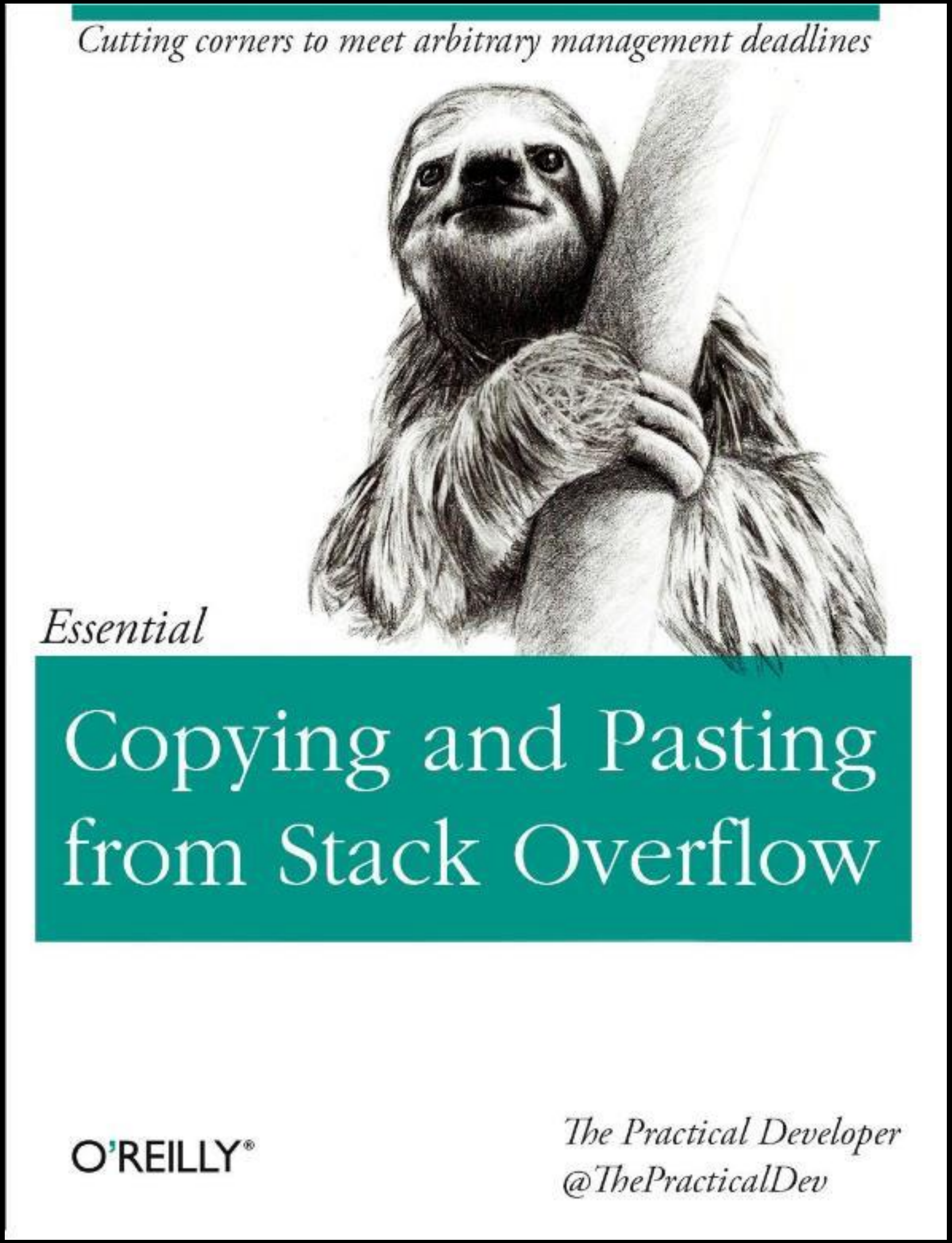
Essential

Googling the Error Message

O RLY?

The Practical Developer
@ThePracticalDev

STARECAT.COM



Cutting corners to meet arbitrary management deadlines

Essential

Copying and Pasting from Stack Overflow

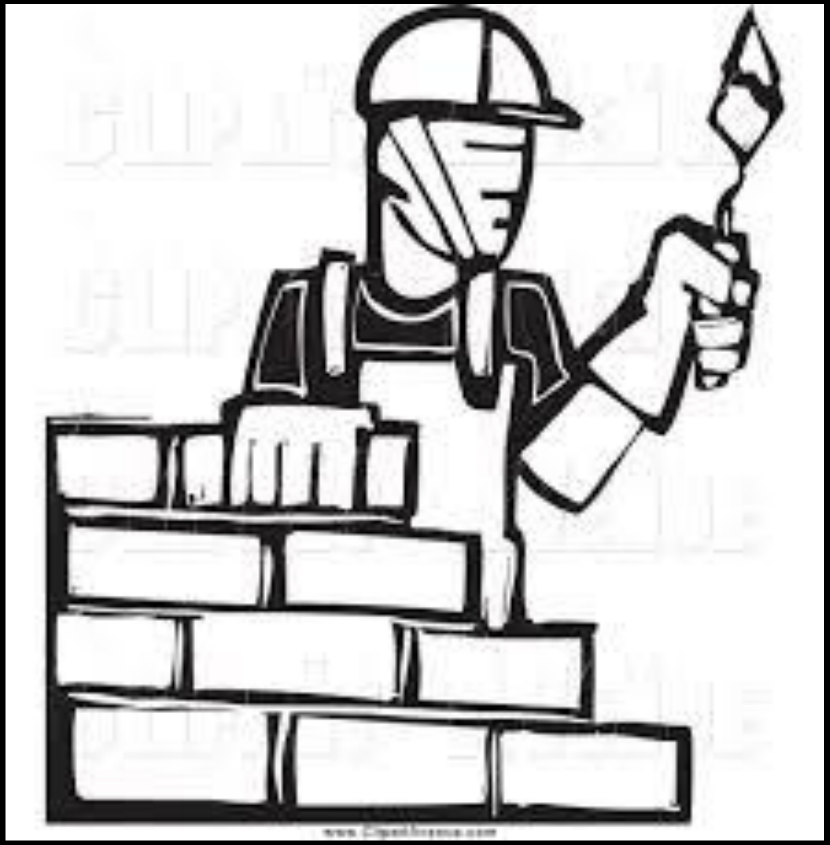
O'REILLY®

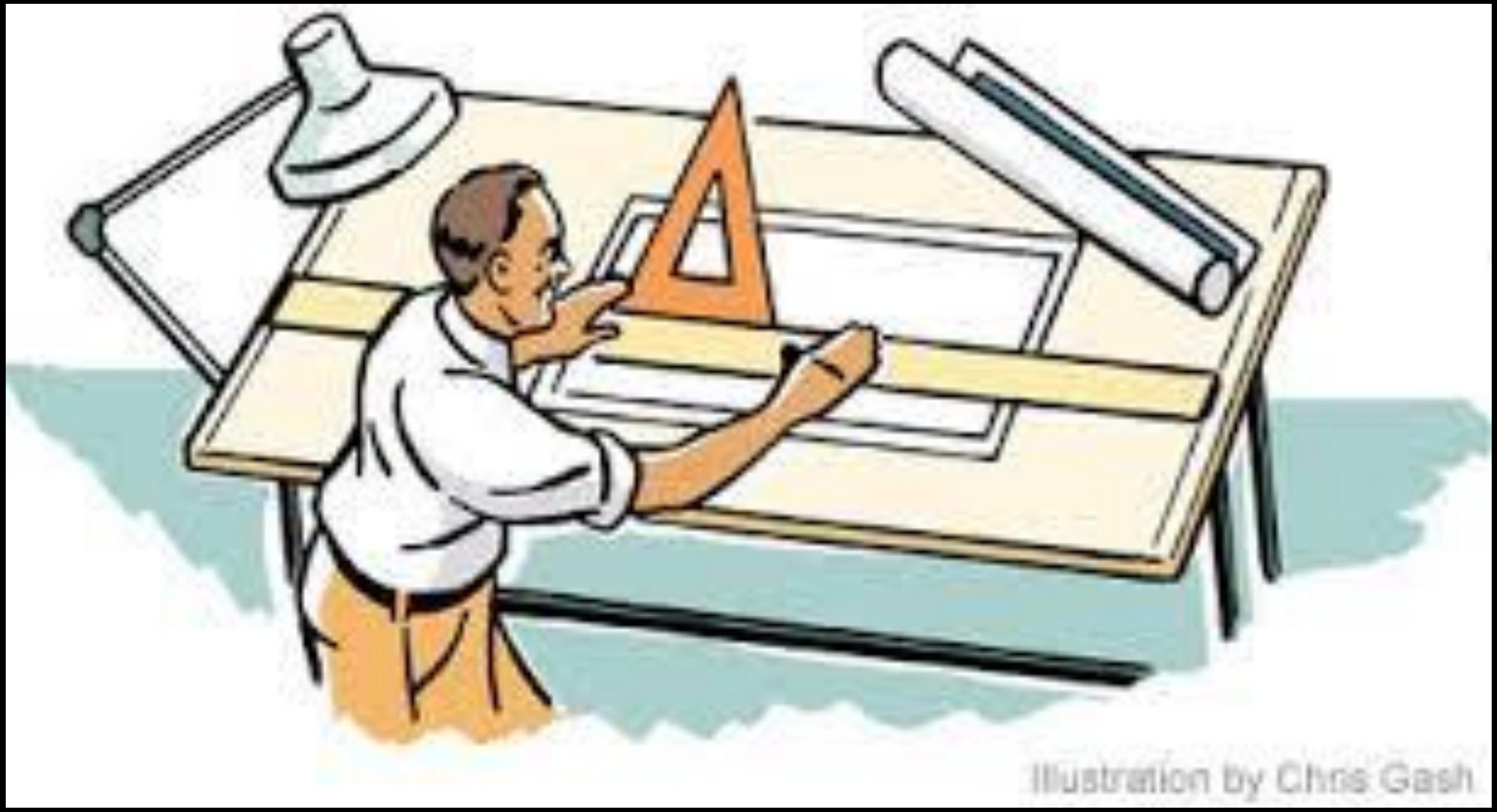
The Practical Developer
@ThePracticalDev

Software Versus Other disciplines

- testing software quality is hard
- lower barrier to entry
- immaturity of the discipline
- customer expectations: quality, delivery timeline, etc.
- fast pace of technological change
- software is easier to copy
- software isn't always "soft"
- change is not easy, yet requirements do change
- change often forces a rewriting of major parts of the software
- developers still need to plan, execute, test, and sell – the discipline is still in its infancy

Can you tell me the difference?





‘Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson

Programs Vs Software

• Usually small in size	• Large
• Author himself is sole user	• Large number of users
• Single developer	• Team of developers
• Lacks proper user interface	• Well-designed interface
• Lacks proper documentation	• Well documented & user-manual prepared
• Ad hoc development	• Systematic development

What do Software Engineers do?

Interaction

Maintenance

Agile, SCRUM

Desktop, embedded, mobile, web-based

Networks

Open source

Extreme programming

Concurrency

Teams

Data flow

Accessibility

Testing

Computer games

SVN, CVS

Functions, Methods

Security

Graphics

Websites

Ruby, PHP

Webservers

Hardware

User-centered

GUI

Software architecture

Meetings

Linux, .NET, OS X

AJAX

Financial systems

SQL

UML

Databases

Requirements scenarios

Design patterns

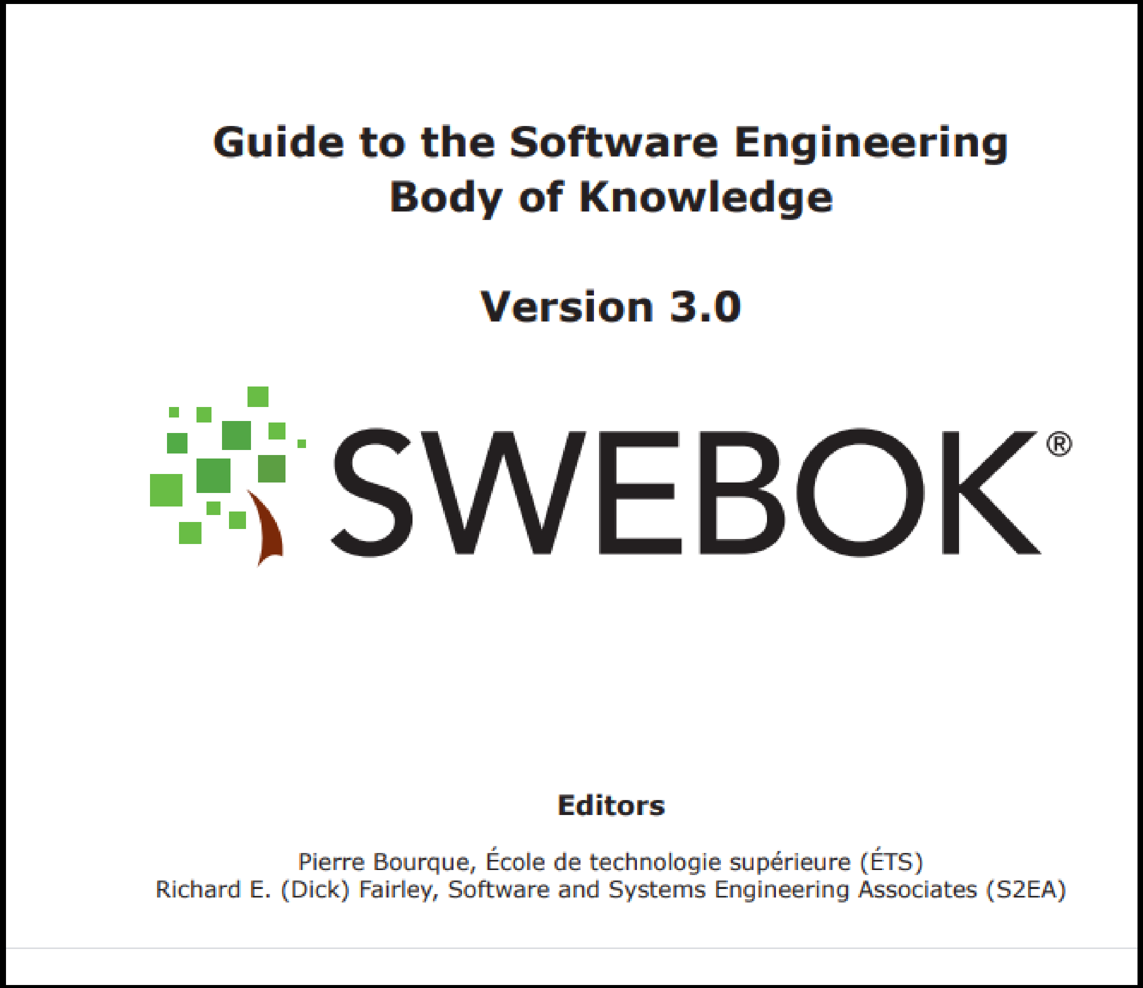
Java, C++, Python

Objects, classes

Software models

What is Software Engineering? Again?

‘‘(1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e. the application of engineering to software. (2) The study of approaches as in (1)’’ (IEEE 610.12, 1991)”



Guide to the Software Engineering Body of Knowledge

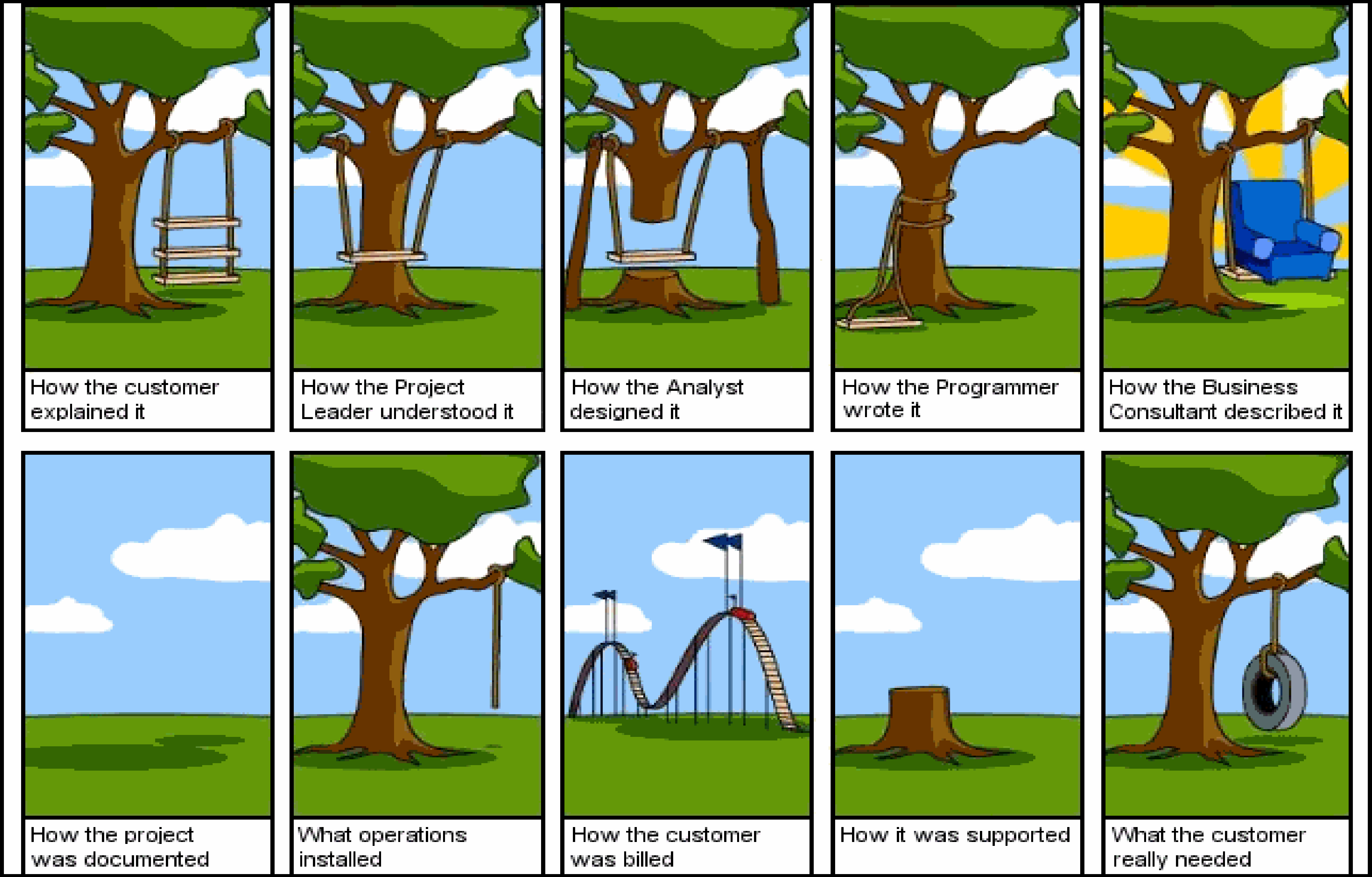
Version 3.0

SWEBOK®

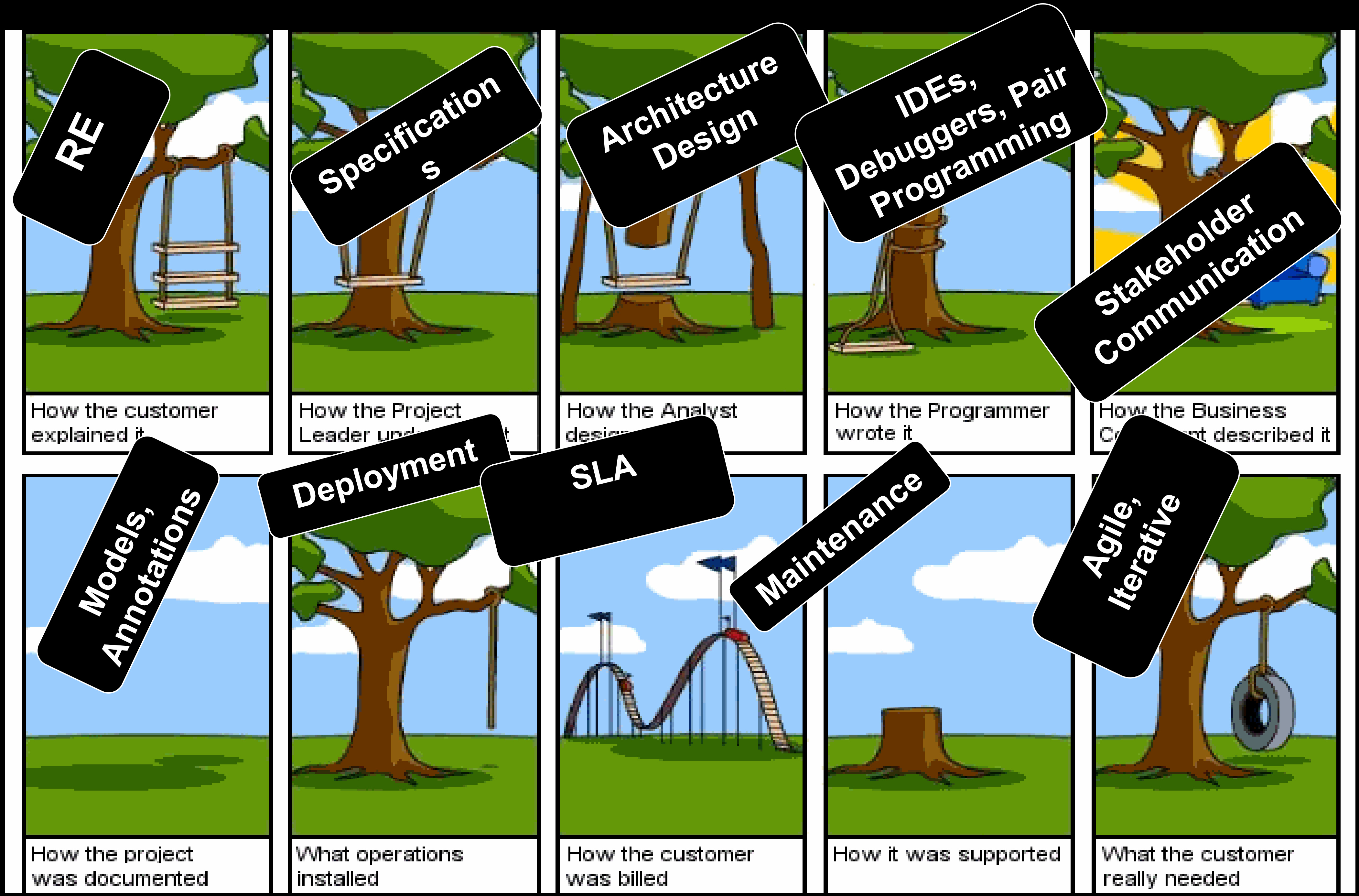
Editors

Pierre Bourque, École de technologie supérieure (ÉTS)
Richard E. (Dick) Fairley, Software and Systems Engineering Associates (S2EA)

A Practitioner's View


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Software Engineering is still an young and emerging discipline but has made significant progress in the last 50 years!



RE
Specifications
Architecture Design
IDEs, Debuggers, Pair Programming
Stakeholder Communication
How the customer explained it
How the Project Leader und
How the Analyst desig
How the Programmer wrote it
How the Business C nt described it
Models, Annotations
Deployment
SLA
Maintenance
Agile, Iterative
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Source: Great image courtesy of Coding Horror

Who is a Software Professional?

- Lead Software Engineer
- Software Architect
- Testing Engineer
- Quality Analyst
- Software Engineer
- Domain Expert
- Software Developer
- Requirements Engineer
- Usability Expert
- Support Engineer
- Systems Analyst
- Business Analyst

What the Errors Tell Us

"With a preventative paradigm, most errors aren't allowed into a system in the first place, just by the way the system is defined. With such an approach, the more reliable the system, the higher the productivity in its lifecycle."

→ Margaret Hamilton

PAST CONTRIBUTIONS

“Those who cannot remember the past are condemned to repeat it” → George Santayana

1960s to 1980s

- Modular programming, *cohesion* and coupling as a means for *decomposition*
- Formal approaches to software engineering
  - structured programming
  - formal ways to express and reason about programs
  - Proving programs

- Birth of object-oriented rather than algorithmic in nature
- The waterfall model as a means for a formal software development process
  - iterative development
  - prototyping
  - Software artifacts beyond source code itself

1960s to 1980s

- Barbara Liskov’s Abstract Data Types
- Entity–relationship modeling
- Structured analysis and design methods
- software inspections
- functional programming
- best practices for distributed computing
- debugging

1980s

- software quality
- the rise of ultra-large software-intensive systems
- the globalization of software
- shift from programs to distributed systems

- Rise of object-oriented programing and languages: Smalltalk, C with Classes, Ada, and many others
- The Unified Modeling Language
- Philippe Kruchten's 4+1 View Model of software architecture

1980s

- Barry Boehm's work in software economics, together with his spiral model
- Vic Basili's empirical software engineering
- Software metrics
- Clean-room software engineering

- Donald Knuth's literate programming
- Watts Humphrey's Capability Maturity Model
- Cox's bazaar model of component-based engineering
- COM (Component Object Model), which were the predecessors of today's microservice architecture

1990s

- The rise of internet
  - users were measured in the billions
  - billions of devices
  - non trustworthy
- Building systems from components

- Continuous integration with incremental and iterative development was becoming the norm.
- Design patterns as next level of abstraction by The Gang of Four

2000s

- Software architecture styles
- Legal framework for open source
- The software outsourcing contract model – Kiran Karnik
- Service-based architectures and microservice architectures as Web's technical infrastructure
- The idea of refactoring
- The Agile Manifesto, 2001
- Datacenter and Cloud as infrastructure
- Company-specific ecosystems rose like walled cathedrals: Amazon, Google, Microsoft, Facebook, Salesforce, IBM

2000s

- The domination of Java, JavaScript, Python, C#, PHP, and Swift
- Git and GitHub
- Version Control
- Stack Overflow
- Computational thinking
- Metrics

- Full stack development
- DevOps
- Internet of Things
- End User Driven – Everybody codes!
- The era of open source frameworks: Bootstrap, jQuery, Apache, NodeJS, MongoDB, Brew, Cocoa, Caffe, Flutter

“One of the major challenges facing project software system managers and maintainers in the 1980’s is how to upgrade large, complex, embedded systems, written a decade or more ago in unstructured languages according to designs that make modification difficult.”

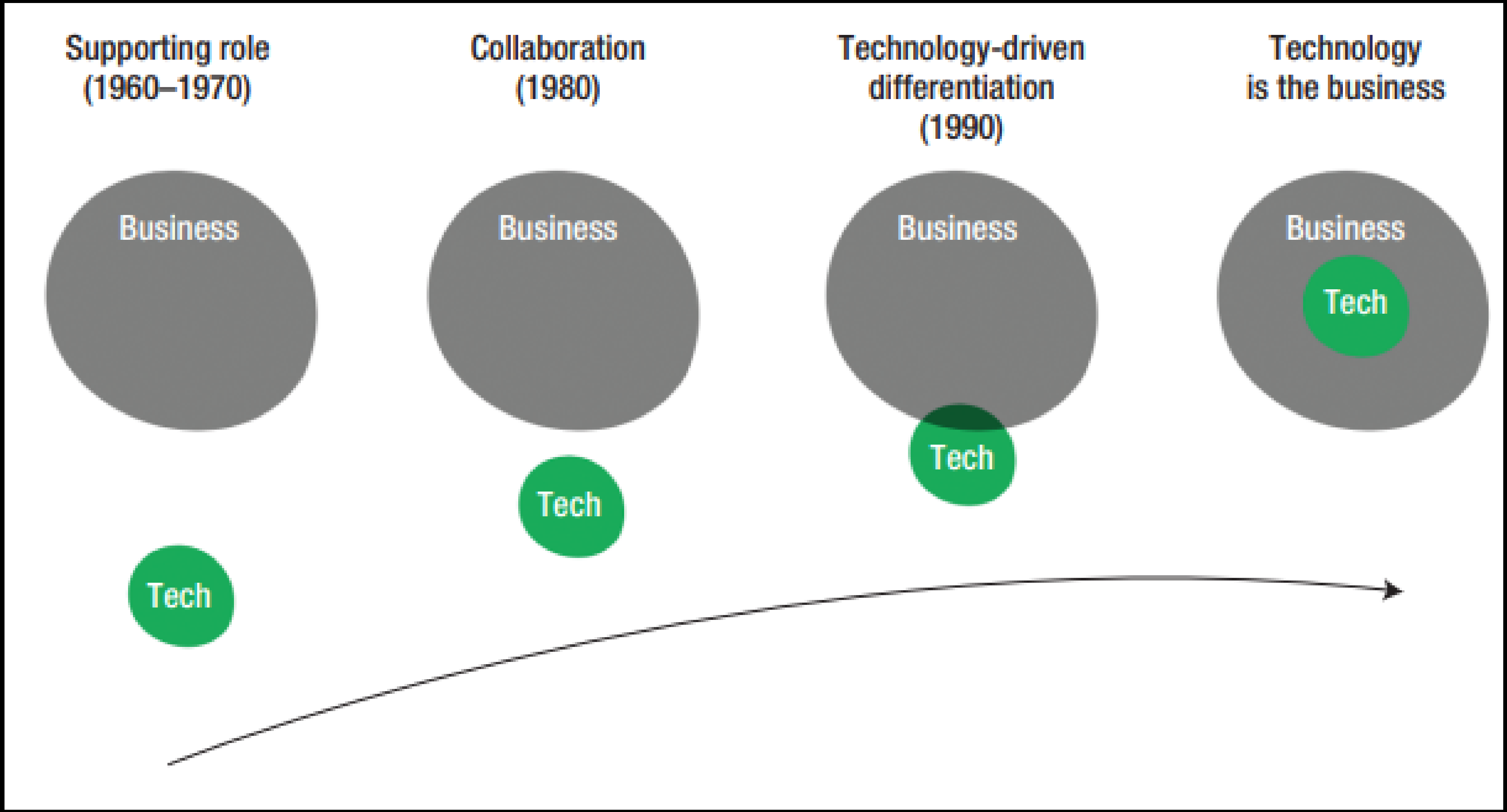
→ *R.N. Britcher and J.J. Craig*

CURRENT TRENDS

Yesterday's the past, tomorrow's the future, but today is a GIFT.
That's why it's called the present.

→ *Bill Keane*

Changing role of Technology



Source: Dörnenburg, E. (2018). The path to DevOps. *IEEE Software*, *35*(5), 71-75.

Achievements

- Software has become part of physical reality and is ubiquitous – client server architectures, cloud, services, automation… serving billions of devices and users!

- Design, development and deployment of software systems with millions to hundreds of millions of code lines. (modern software's size and complexity)

- Software engineering has coevolved with computer hardware – advanced system architectures

- powerful abstractions such as design and architecture patterns, frameworks that increase software construction's efficiency, maintainability and reliability

Advances 1

- Adaptive Systems
- Agile Software Development
- Architecture and Design
- Aspect-orientation and feature interaction
- Computer-supported collaborative work
- Configuration Mgmt and Deployment
- Dependability, Safety and Reliability
- Distributed, web-based and internet-scale SE
- Embedded/Real-Time Software
- Empirical SE
- End User Software Engineering
- Engineering Secure Software
- Human-Computer Interaction
- Mobile, Ubiquitous, Pervasive Systems
- Model Driven Engineering

Advances 2

- Parallel/Distributed Systems
- Patterns and Frameworks
- Processes and workflow
- Program Comprehension and Visualization
- Programming languages
- Requirements Engineering
- Rev Eng/Maintenance
- Search-based Software Engineering
- Software Components and Reuse
- Software Economics and Metrics
- Software Specifications
- Software Testing and Analysis
- SW Processes and Workflows
- Testing and Analysis
- Theory and Formal Methods
- Tools/Development Environments
- Validation and Verification

Challenges

- Programming languages that do not reflect the needs of today's programming of cyber-physical systems

- Our programming languages are no good for handling real-time interaction and distribution.

- Coding is error prone, not sufficiently abstract, and still done too close to the machine level or at least to the OS level, mixing application logic

- Best Practice versus Everyday Practice

Open Challenges from NATO

- getting requirements right
- designing adequate and appropriate architectures
- doing implementation effectively and correctly
- verifying the quality of the result
- maintaining software systems with targeted functionality and high code quality over long time periods

The Unanswered 1

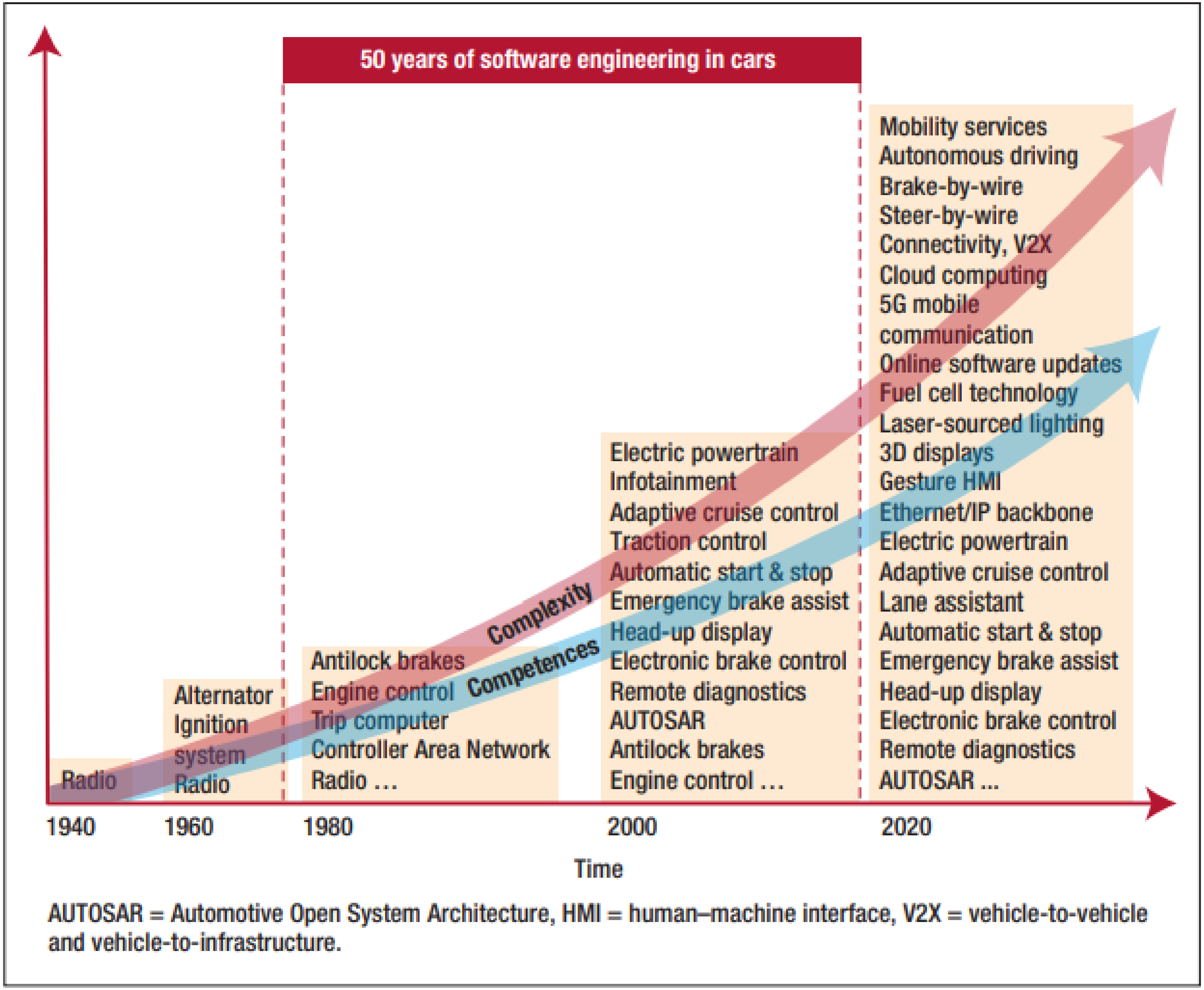
- Is agile software development better than carefully planned development, and if so, under what circumstances?
- Are formal methods essential, or even useful, or are they just an intellectual exercise that gets in the way of building real-world systems?
- Should a system's architecture be designed, or does it emerge organically during development? If the former is the case, how, when, and to what extent is architectural design appropriate?
- Does success depend primarily on technical skills or peopleware?

The Unanswered 2

- Are standards critical to software engineering’s maturation as a discipline, or are they just an impediment to pragmatic development?
- Should software development rely on up-front planning or on a more adaptive approach in which you strive to decide at the last responsible moment?
- • What constitutes good software: software that appears to work most of the time and does the job? Or delights users in special ways? Or never crashes? Or is easy to understand and maintain? Or has an elegant design? Or is thoroughly tested? Or is proven to be correct? Or is a combination of some or all of these criteria?

FUTURE PREDICTIONS

"The best way to predict the future is to create it."
→ Abraham Lincoln and Peter Drucker?



AUTOSAR = Automotive Open System Architecture, HMI = human–machine interface, V2X = vehicle-to-vehicle and vehicle-to-infrastructure.

FIGURE 1. Automotive electronics as an example of 50 years of software engineering.

Ebert, C. (2018). 50 years of software engineering: Progress and perils. *IEEE Software*, *35*(5), 94-101.

Open Challenges & Trends

- Software complexity and Specialization
- Effective collaboration with software stakeholders
- Maintaining legacy software
- Context-driven software engineering
- Social Patterns in SE
- Reproducibility
- Innovations in Design, Develop, Deployment
- Cross pollination of ideas
  - Software as Key competitor
  - Specialized software stacks
  - Beyond code
- Data privacy

What can we analyze? – The new era! [AI/ML/NLP]

open source initiative®

Google Play

3.5 million apps
Billions of app reviews

GitHub

50 million developers
100+ million repositories

Google
Issue Tracker

10 million+ issues

Jira

stackoverflow

16 million+ questions
25 million+ answers
Q every 11 minutes
A every 8 minutes

THE APACHE® SOFTWARE FOUNDATION

Image Credits: All icons/logos used in this presentation are taken from Google Search labelled with Reuse

Evidence-based software engineering

- “Pre and post release failures are not connected”
- “The language construct GOTO is rarely considered harmful”
- “Strongly typed languages are not associated with successful projects”
- “Test-driven development is not any better than “test last”
- “Most “bad smells” should not be fixed”

Concerns with Software Analytics

- The Invisible Software Artifacts
- Better evidence for software experiments
- Scaling software experiments
- Does it work in another context?
- Is there actionable evidence?
- Can we reply on AI/ML or NLP models?

Trends

- Software engineering ↠ Systems engineering, cyber-physical systems
- Programming has to be much closer to the real world
- Architecture abstractions
- Beyond von Neumann–style!
- Self-adaptive software systems
- Technical challenges such as autonomous driving pose questions related to societal, ethical, and legal issues

The Need for Better Tooling

- New Programming Languages
- Coding at higher level of abstractions
- Debugging
- Beyond UML
- APIs, Frameworks, Cloud, Microservices
- AI driven tools

The Future of Agile

- How will agile practices enable AI-based software engineering?
- Can agile improve data analytics and data sciences practices in the way it has improved software engineering?
- How can agile processes support the development of safety critical systems in increasingly software-intensive autonomous vehicles, software-defined networking, and robotics development and integration?
- End user software development and agile?

Not Industry Ready Yet!

- Gap between academic research and industry!
- Documenting Architectural Decisions
- Automatic Bug Assignment
- Software Quality Improvement
- Requirements Based Testing

Two Questions!

- *What can AI/NLP do for Software Engineering?*
- *What can Software Engineering do for AI/NLP?*

Energy-Aware Software Engineering

- Energy is no longer “free” and unlimited. Mobile devices’ energy efficiency affects us all, and large corporations are increasingly concerned with their server farms’ energy efficiency.

- Green Computing

Summary

- Software Engineering is a unique and emerging discipline. The goal is Quality with less effort and time!
- Abstractions, notations, design patterns, frameworks, component based, open source, advanced architectures and networks
- Systems engineering, AI for SE, SE for AI, New Programming Languages, Agile, Evidence-based SE, Energy-aware SE, AR/VR