Skoltech

Skolkovo Institute of Science and Technology

Software Development Life Cycle

Foundations of Software Engineering

FSE v2020.1, Block 1 Module 1

Alexey Artemov, Fall 2020

Lecture Outline

Skoltech
Skolkovo Institute of Science and Technology

§1. Software engineering [15 min]

1.1. Why learn about the software development methodology?

1.2. What software development life cycle looks like

1.3. Challenges in software development

§2. Software development life cycle [15 min]

2.1. Requirements, specification, architecture, design, implementation, testing, deployment, maintenance

§1. Software engineering

Why learn about software engineering?

The Goal:
Think Before You Code

Image Credit: Alexey Artemov

Image Credit: Alexey Artemov

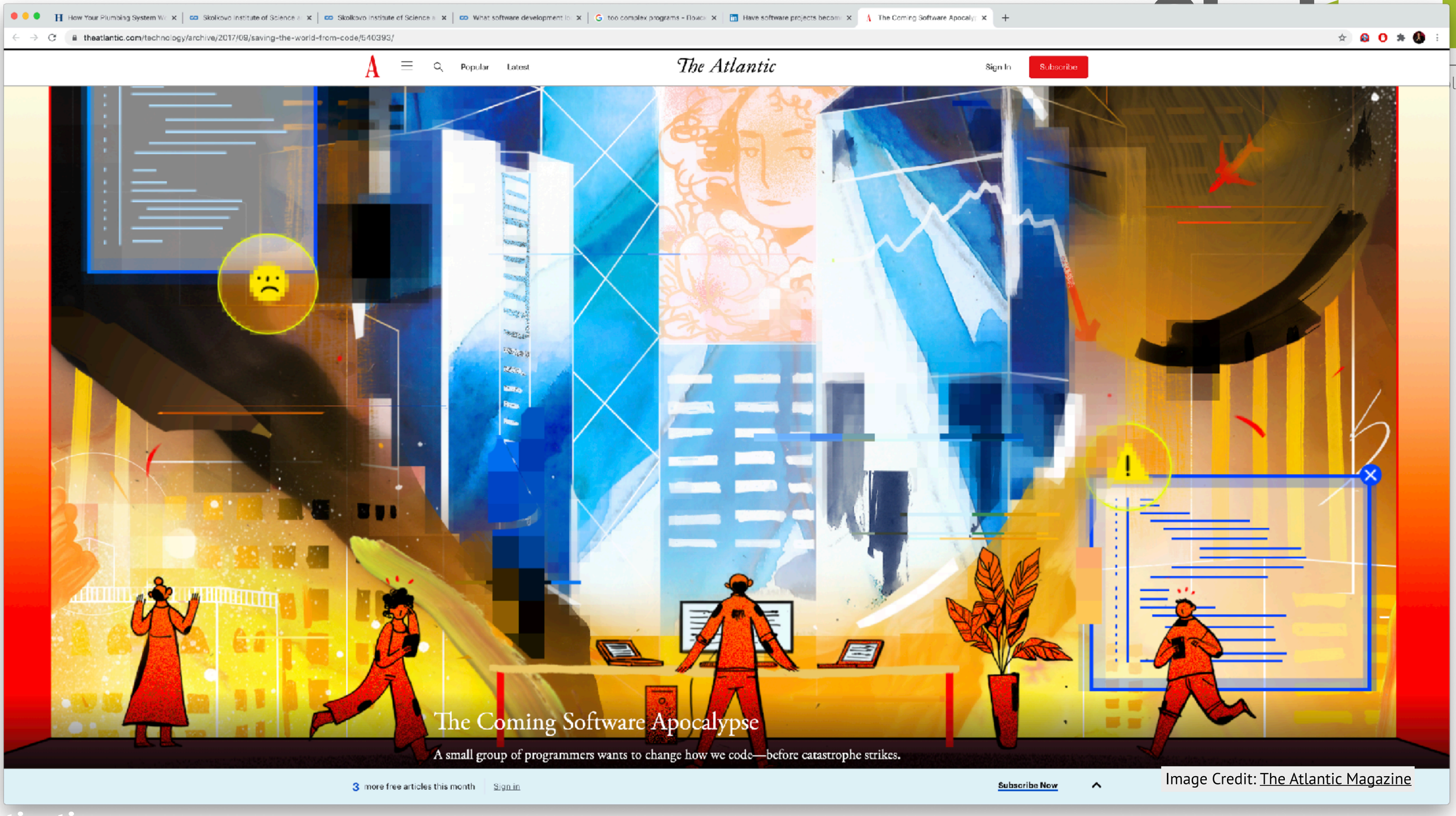
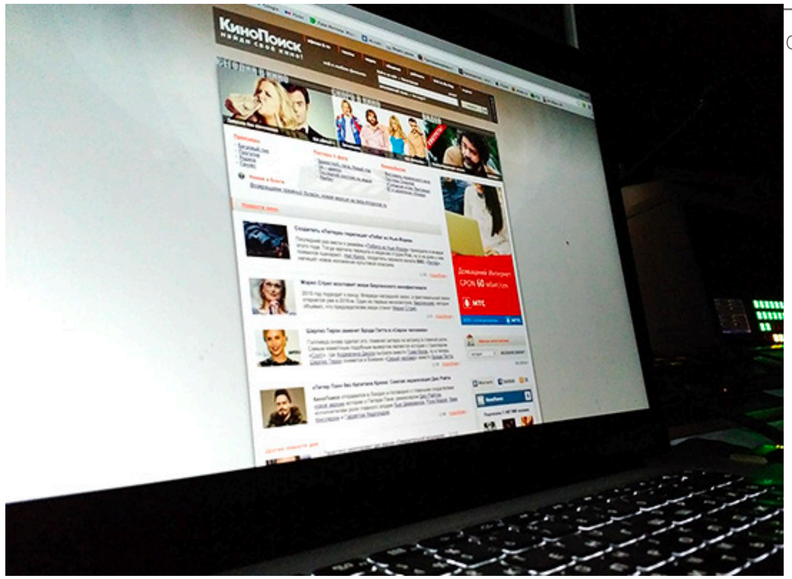


Image Credit: The Atlantic Magazine

Инженеры Яндекса разговаривают с эталонным пользователем



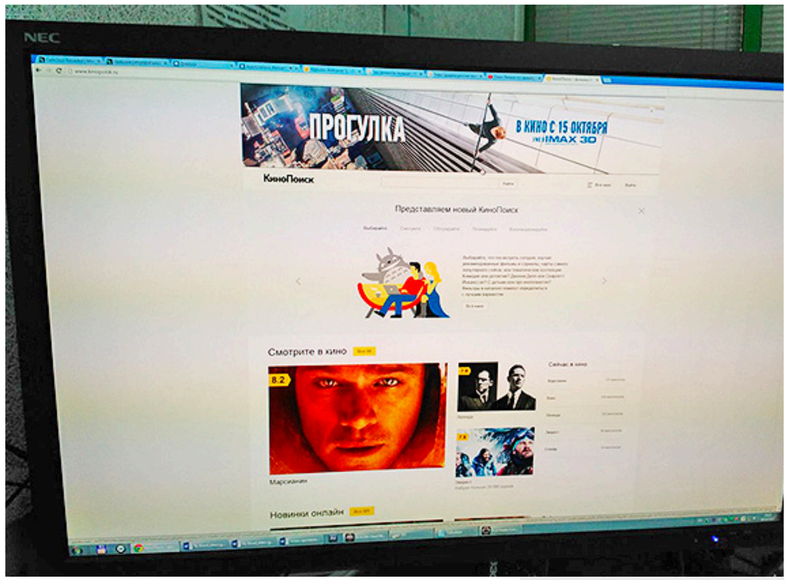
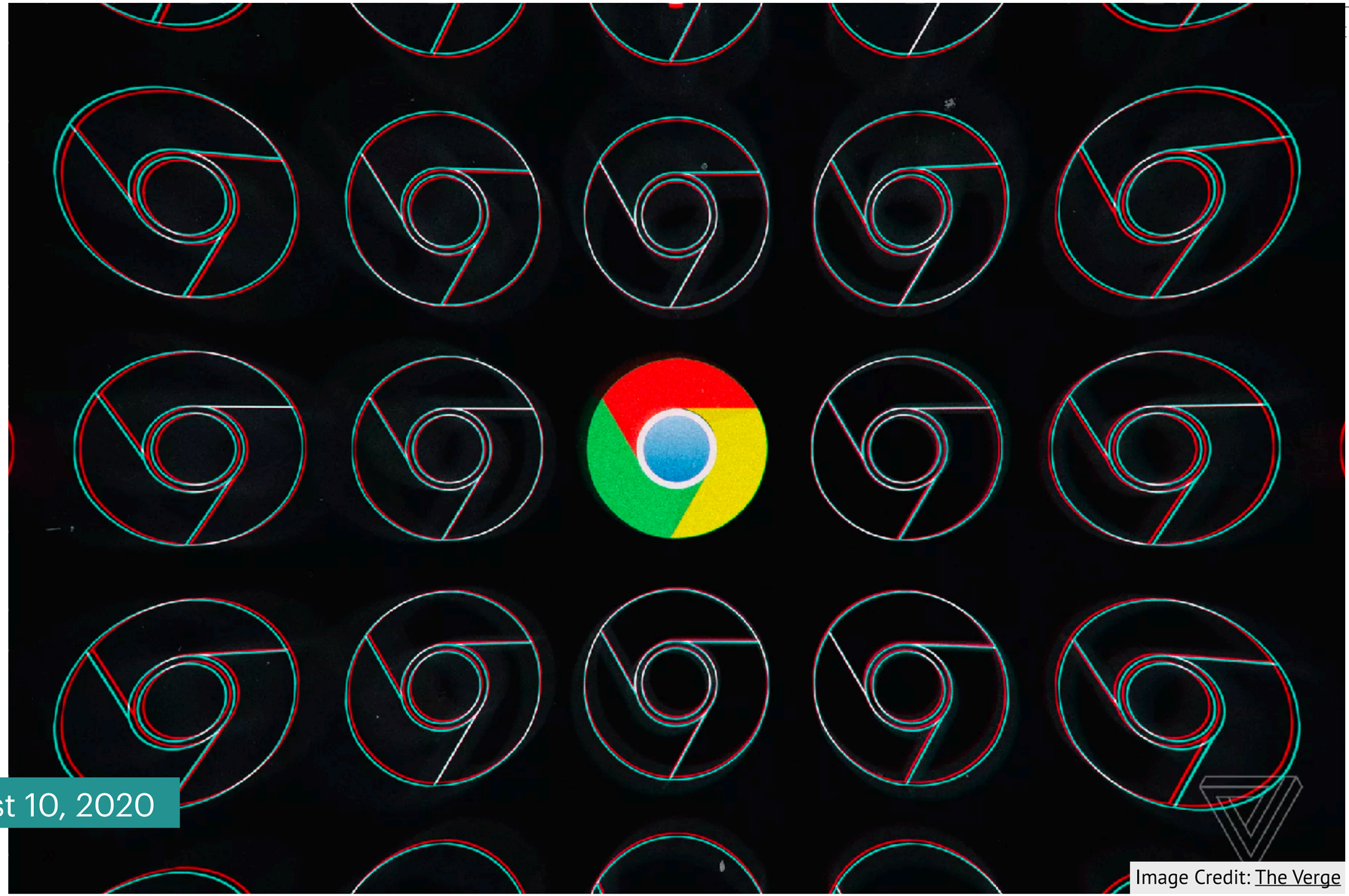


Image Credit: Lenta.ru

Image Credit: bbvaopenmind.com

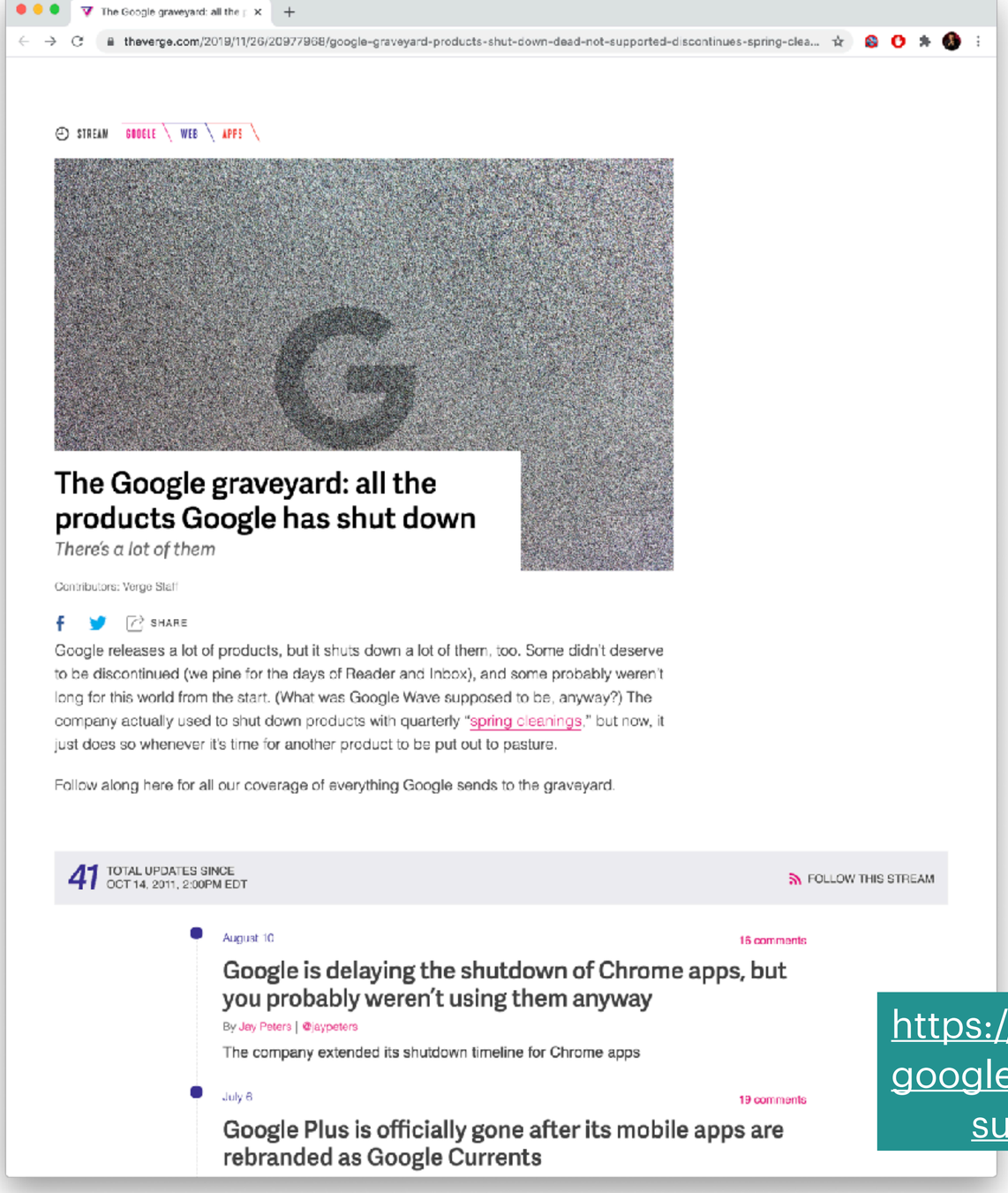
370 million dollars worth of fireworks because of a software bug. (Source: ESA)

The greater horizontal acceleration caused a data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow and cause a hardware exception. Efficiency considerations had omitted range checks for this particular variable, though conversions of other variables in the code were protected. The exception halted the reference platforms, resulting in the destruction of the flight.



August 10, 2020

Image Credit: The Verge



https://www.theverge.com/2019/11/26/20977968/google-graveyard-products-shut-down-dead-not-supported-discontinues-spring-cleaning

Image Credit: The Verge

What is software engineering?

§1. Software engineering

Skoltech
Skolkovo Institute of Science and Technology

1.2. What software engineering looks like

- “An organized, analytical approach to the design, development, use, and maintenance of software.” [1]
- Methodology with clearly defined processes for creating high-quality software
- Goal: produce software with the highest quality and lowest cost in the shortest time possible
- Ensure effectiveness, usability, and maintainability
- Meet schedule and budget
- Make changes to meet unexpected demands

Image Credit: Paul Brennan (CC0 Public Domain, publicdomainpictures.net)

Customer needs

Image Credit: maxpixel.com

High-level design

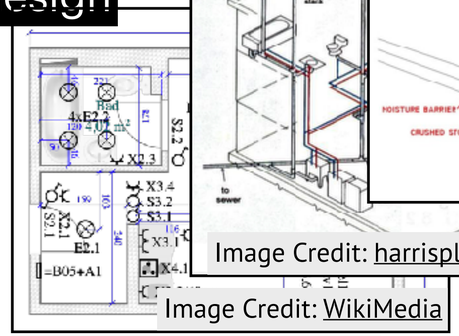


Image Credit: WikiMedia

Image Credit: harrisplumbing.ca

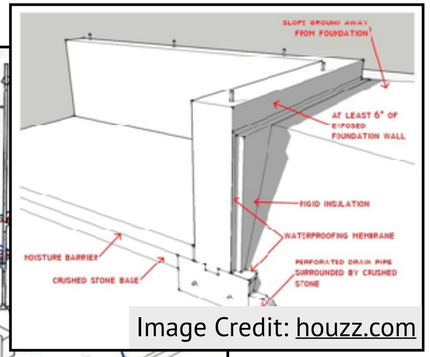


Image Credit: houzz.com

Low-level designs

Image Credit: makaan.com

Customer satisfaction?

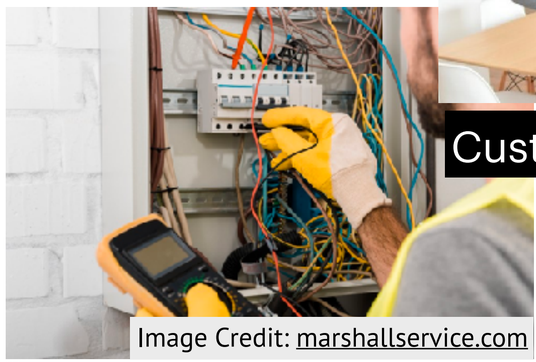
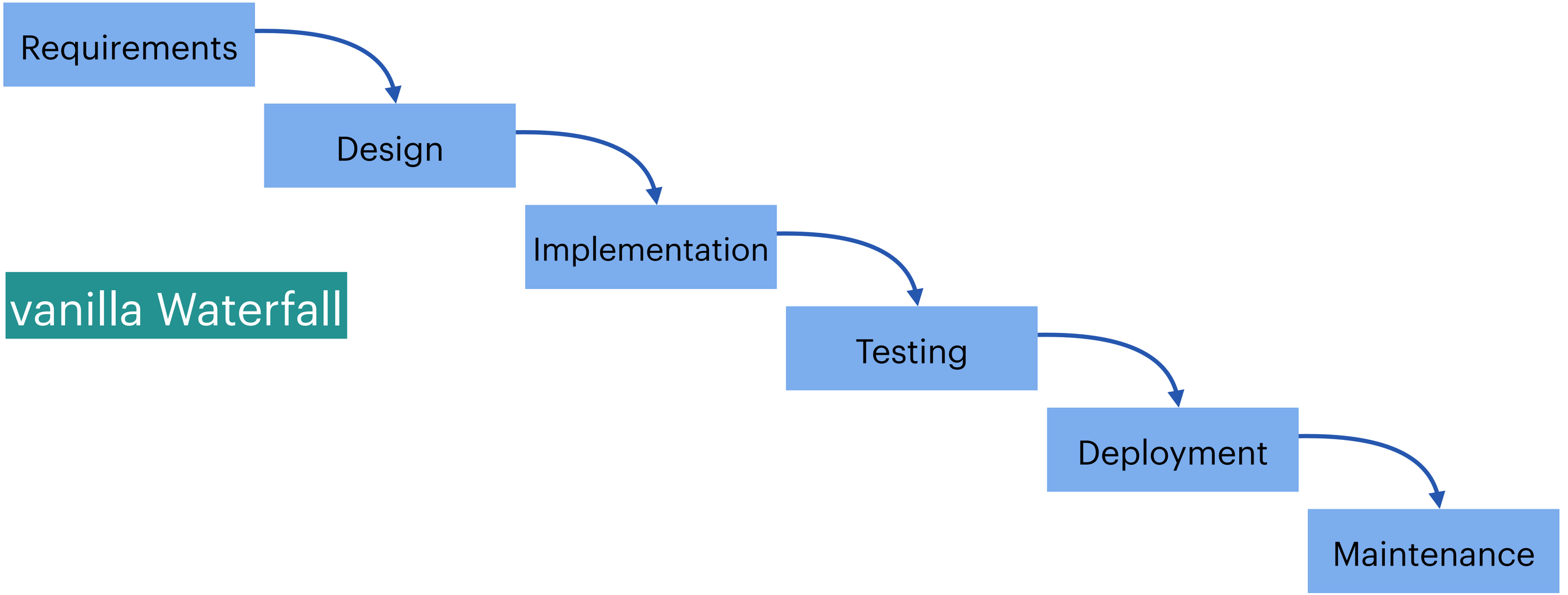


Image Credit: marshallservice.com

Validation and approval

Image Credit: publicdomainpictures.net

Construction



vanilla Waterfall

§1. Software engineering

Skoltech
Skolkovo Institute of Science and Technology

1.3. Challenges in software development

- Uncertain customer needs
- Changing customer requirements
- Rapid evolution of technology
- Changing market trends
- Unestablished project infrastructure
- Integration between complex systems
- Quality assurance
- Requirements misinterpretation

§1. Software engineering

Skoltech
Skolkovo Institute of Science and Technology

1.3. Challenges in software development

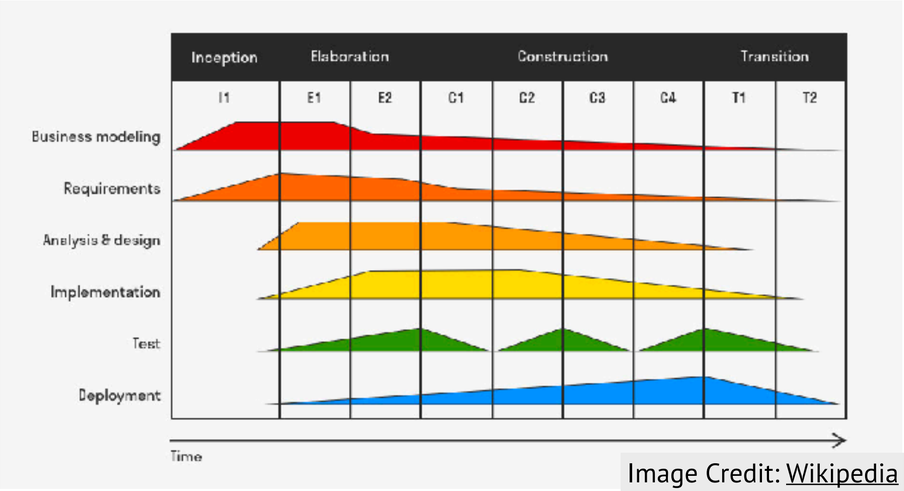


Image Credit: Wikipedia

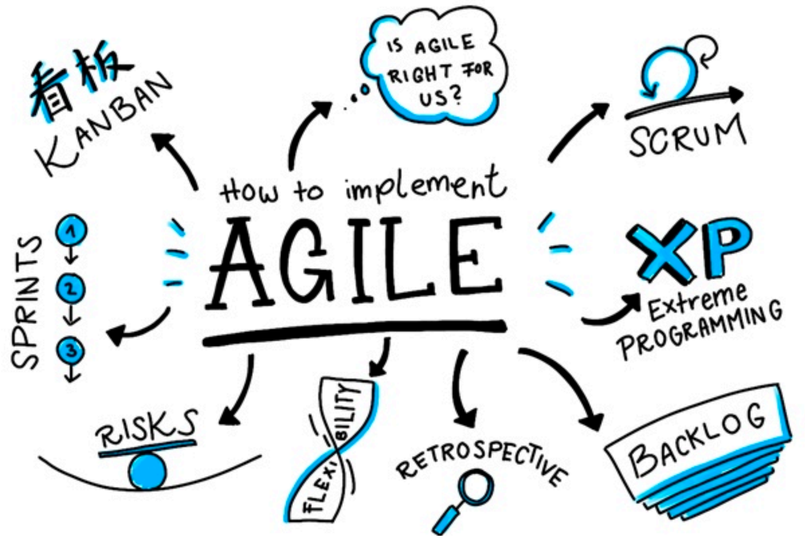


Image Credit: blog.molodost.bz

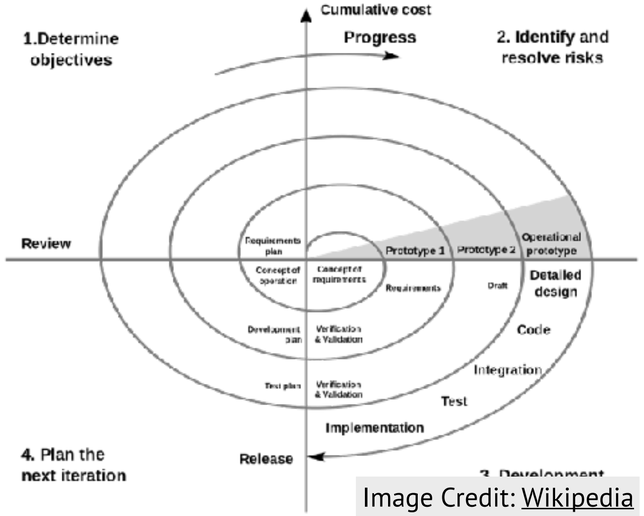


Image Credit: Wikipedia

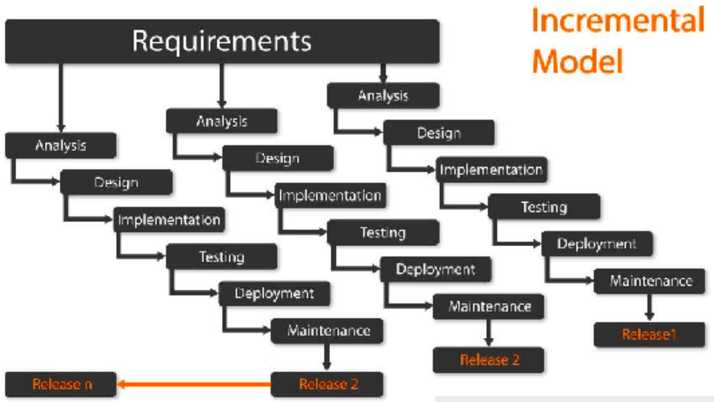


Image Credit: newline.tech

§1. Software engineering

Skoltech
Skolkovo Institute of Science and Technology

1.3. Challenges in software development

Uncertain customer needs	Solid requirements engineering processes
Changing customer requirements	Risk mitigation strategies
Rapid evolution of technology	Agile development processes
Changing market trends	Integrate development and learning new tech

Unestablished project infrastructure	Create dev, test, and prod environments
Integration between complex systems	Continuous integration
Quality assurance	Formal QA, automated testing
Requirements misinterpretation	Incremental development

§1. Software engineering

Skoltech
Skolkovo Institute of Science and Technology

1.3. Challenges in software development

- Software Engineering is one of the most complex things that humans have ever done
  - Uncertain/changing user needs, volatile market trends, rapid technological advancements, infrastructure and integration issues
- ~20% projects fail According to the Catalogue of Catastrophe: http://calleam.com/WTPF/?page_id=3 (and other sources)
- Software engineering: mitigating risks and keeping on track by learning from experience gained throughout human history
  - Solid processes, incremental and iterative practices, risk mitigation strategies
  - On the technical side: QA, testing&CI, deployment, …

§2. Software development life cycle

Why do we need a software development life cycle?


Image Credit: maxpixel.com

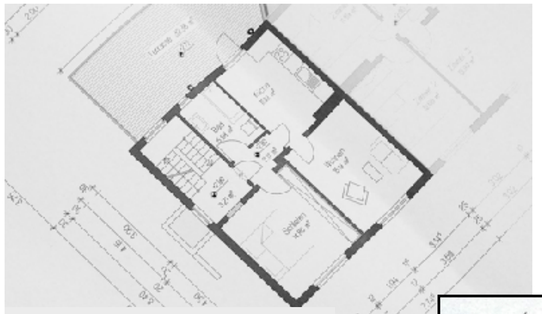


Image Credit: maxpixel.com

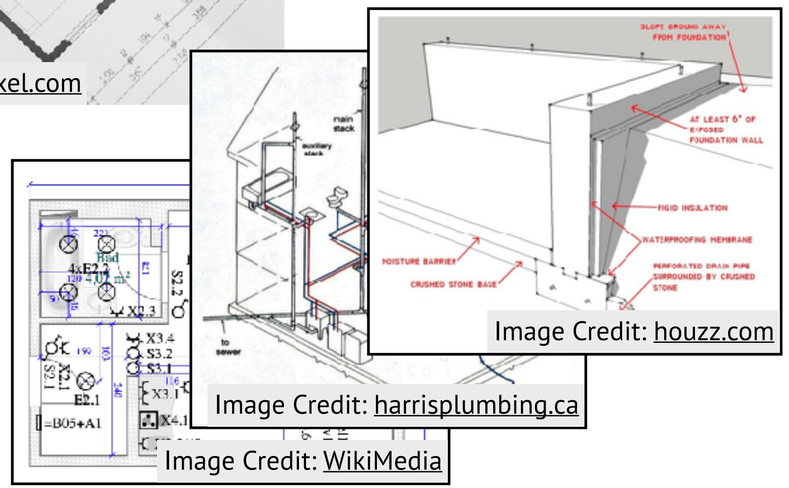


Image Credit: houzz.com
Image Credit: harrisplumbing.ca
Image Credit: WikiMedia


Image Credit: makaan.com

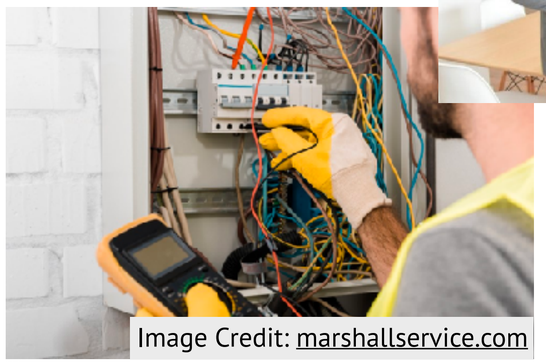


Image Credit: marshallservice.com


Image Credit: publicdomainpictures.net

What is the software development life cycle?

§2. Software development life cycle

2.1. Definitions

- SLDC: useful for managing a planned and controlled development effort
  - Solution consistent with requirements
  - Design/testing processes are sound
  - Repeatable

- Limited in handling uncertainty
- Creates some overhead/bureaucracy

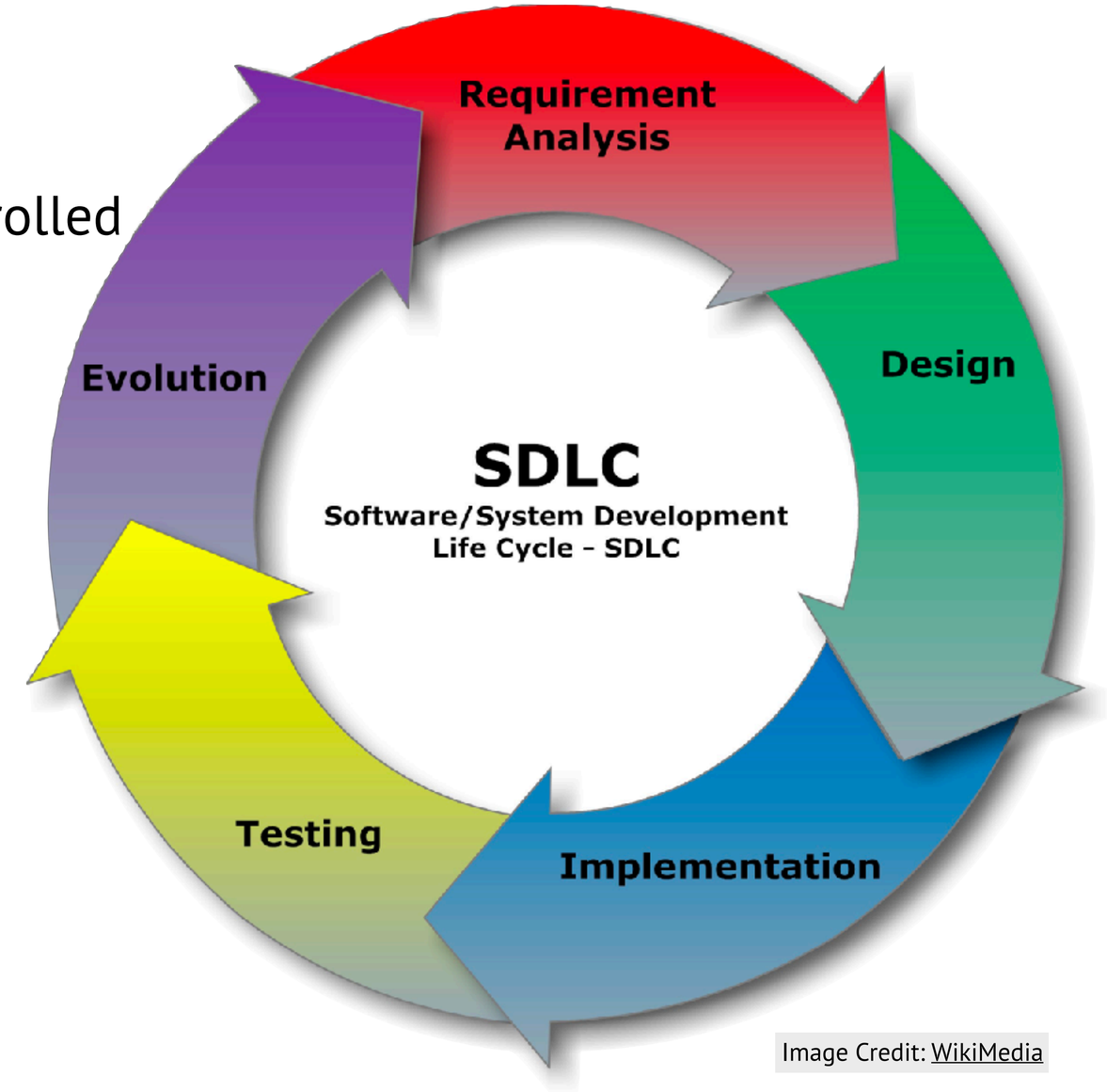


Image Credit: WikiMedia

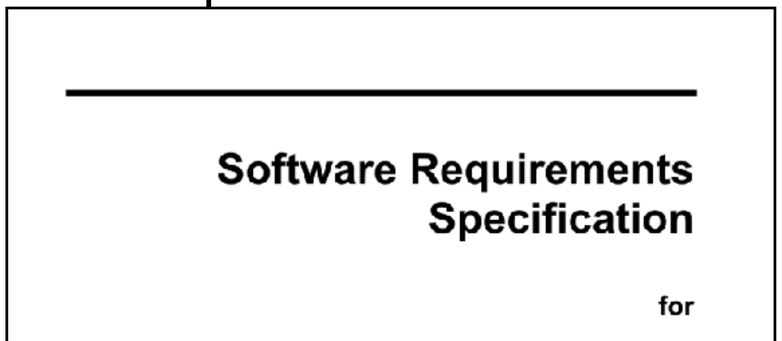
§2. Software development life cycle

2.1. Requirements and specification

- Who's the customer?

- What the customers want vs. need?

- Software requirements documentation



Software Requirements
Specification

for

§2. Software development life cycle

Skoltech
Skolkovo Institute of Science and Technology

2.1. Architecture and high-level design

- What platform to use? (hardware, OS, data design, interfaces, …)


Image Credit: pxhere.com

Image Credit: pexels.com

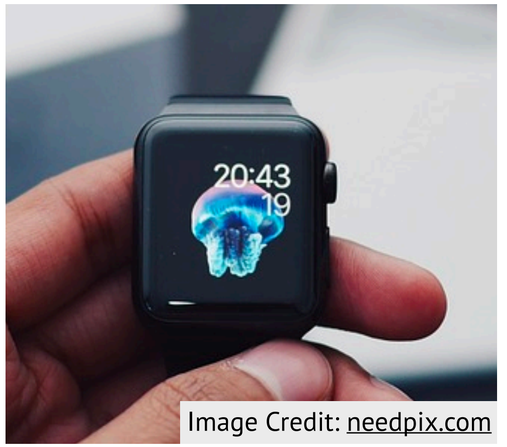


Image Credit: needpix.com

- Components (identification, decomposition, work dispatching, …)

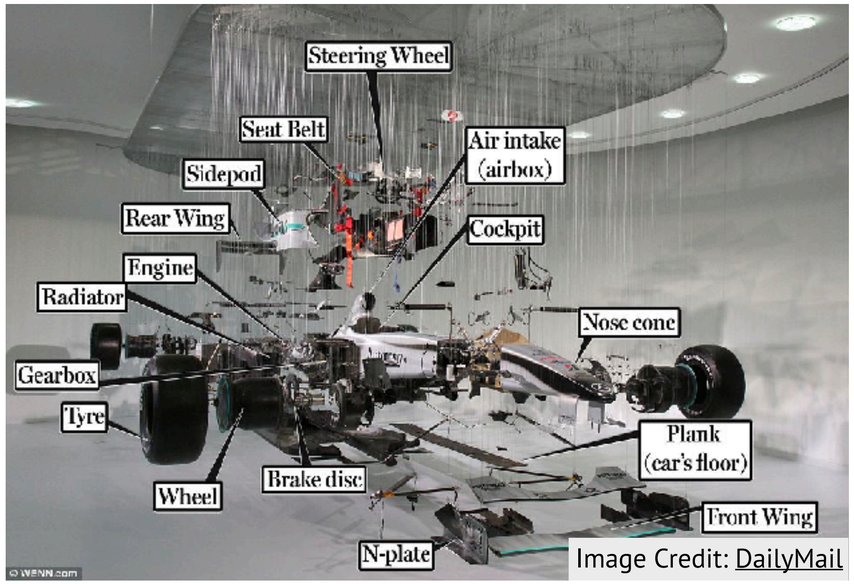


Image Credit: DailyMail

- Project architecture (component interfaces, …)

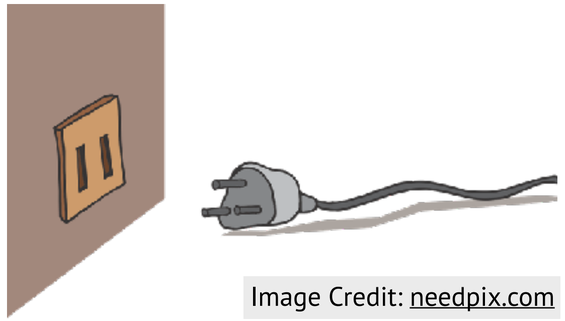


Image Credit: needpix.com

§2. Software development life cycle

Skoltech
Skolkovo Institute of Science and Technology

2.1. Low-level design, implementation, testing

- Proceed in groups
- Low-level design
  (guide developers, refine architecture)
- Implementation
  (refine design until ready to code)
- Testing (unit-testing, integration, regression, …)

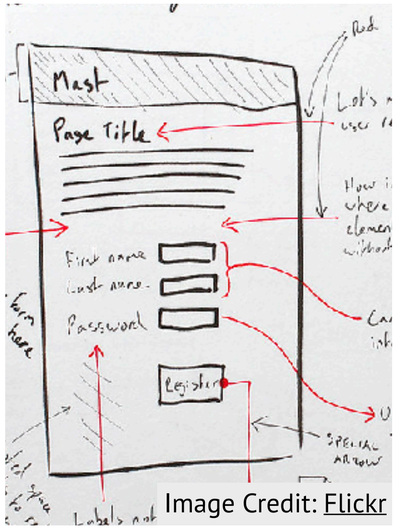


Image Credit: Flickr

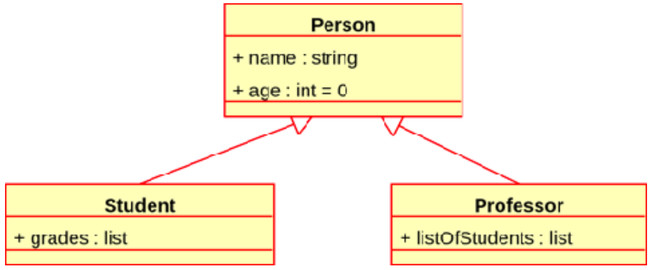


Image Credit: Wikipedia

Image Credit: wallpaperflare.com


Image Credit: WikiPedia

§2. Software development life cycle

2.1. Properties of various design phases

The longer a bug remains undetected, the harder it is to fix.

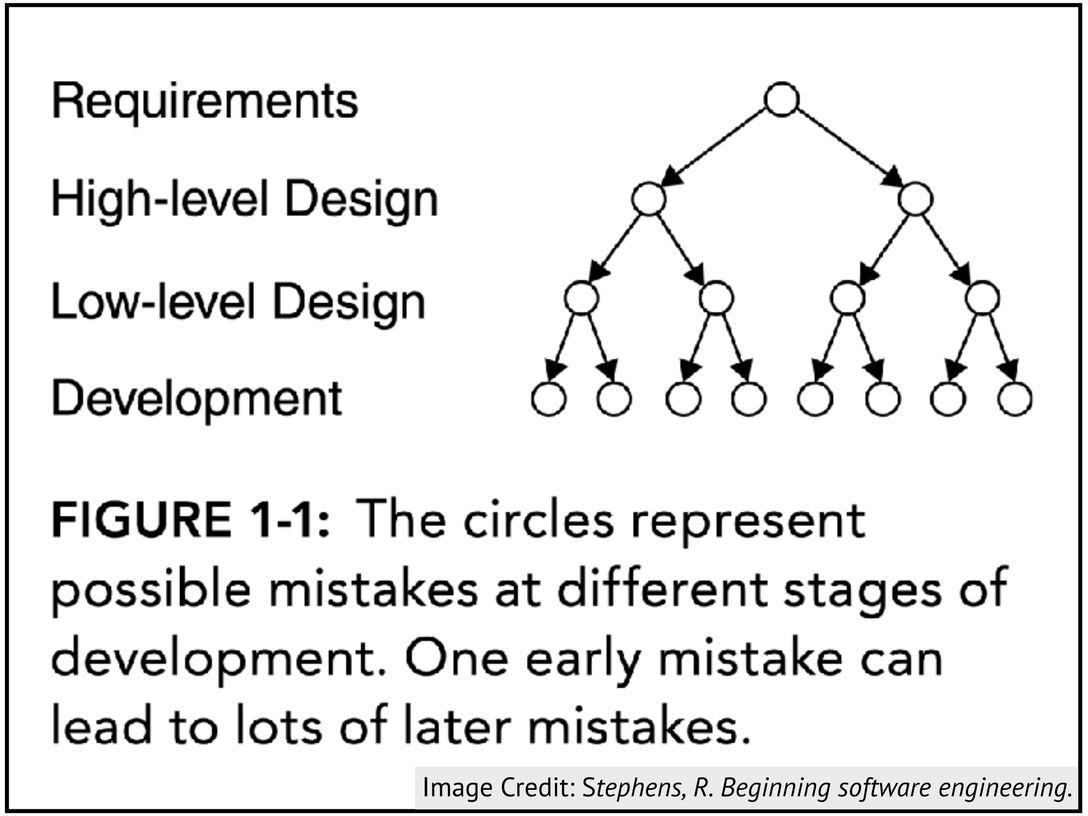


FIGURE 1-1: The circles represent possible mistakes at different stages of development. One early mistake can lead to lots of later mistakes.

Image Credit: *Stephens, R. Beginning software engineering.*

§2. Software development life cycle

Skoltech
Skolkovo Institute of Science and Technology

2.1. Deployment, maintenance, and transition

- Deploy software from testing setup into production (cutover strategies, …)

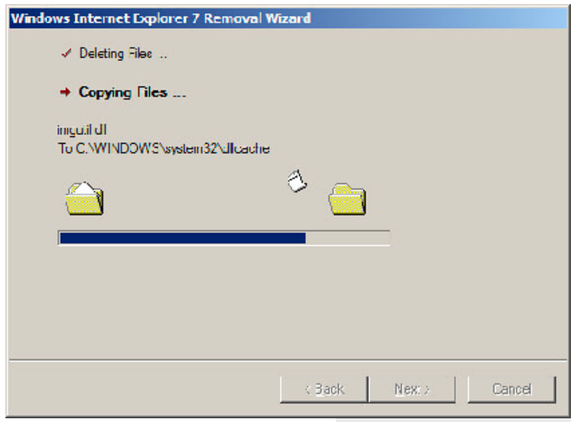


Image Credit: Flickr

Image Credit: public domain

- Maintenance (hotfixes, more if popular)

- Wrap-up (post-mortem)

§2. Software development life cycle

Skoltech
Skolkovo Institute of Science and Technology

2.1. Summary

- All projects perform the same basic tasks:
  - Requirements Gathering
  - High–level (Architectural) Design
  - Low–level Design
  - Development
  - Testing
  - Deployment
  - Maintenance

§2. Software development life cycle

Skoltech
Skolkovo Institute of Science and Technology

2.1. Summary

- Different development models handle the basic tasks in different ways, such as making some less formal or repeating tasks many times.
- The basic tasks often occur at the same time, with some developers working on one task while other developers work on other tasks.
- Work sometimes flows backward with later tasks requiring changes to earlier tasks.
- Fixing a bug can lead to other bugs.
- The longer a mistake remains undetected, the harder it is to fix.
- Surprises are inevitable, so you should allow some extra time to handle them.

References

1. Stephens, R. (2015). Beginning software engineering. John Wiley & Sons.
2. Zhu, H. (2005). Software design methodology: From principles to architectural styles. Elsevier.
3. *Coursera course:* Software Development Lifecycle. University of Minnesota