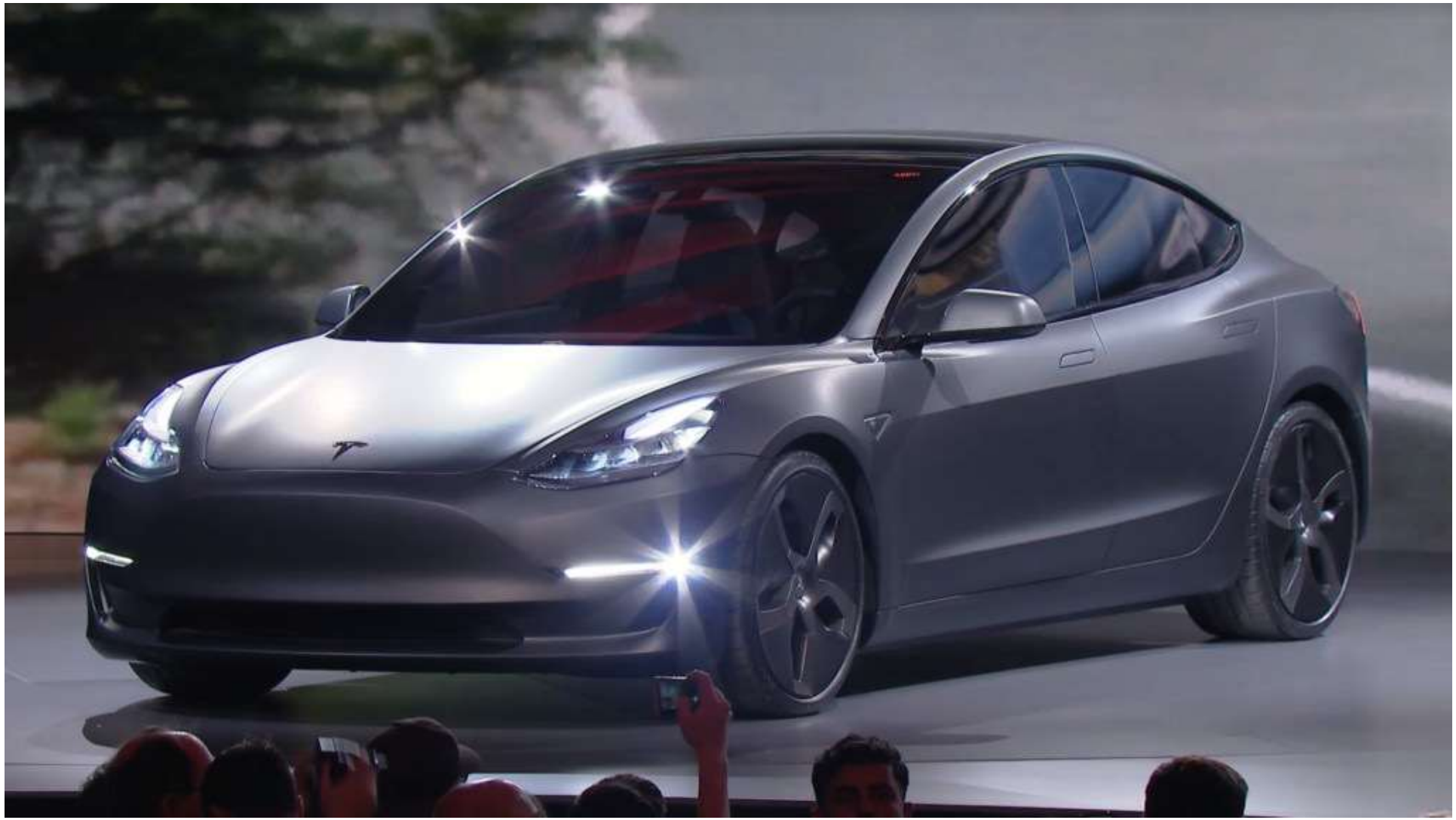
Introduction to Software Development

Software Development for Robotics and Automation Engineering

we want to build great things

THAT’S WHY YOU PICKED ENGINEERING (HOPEFULLY)

MEET TESLA MODEL 3



THE TESLA MODEL S IS A QUIET MASTERPIECE

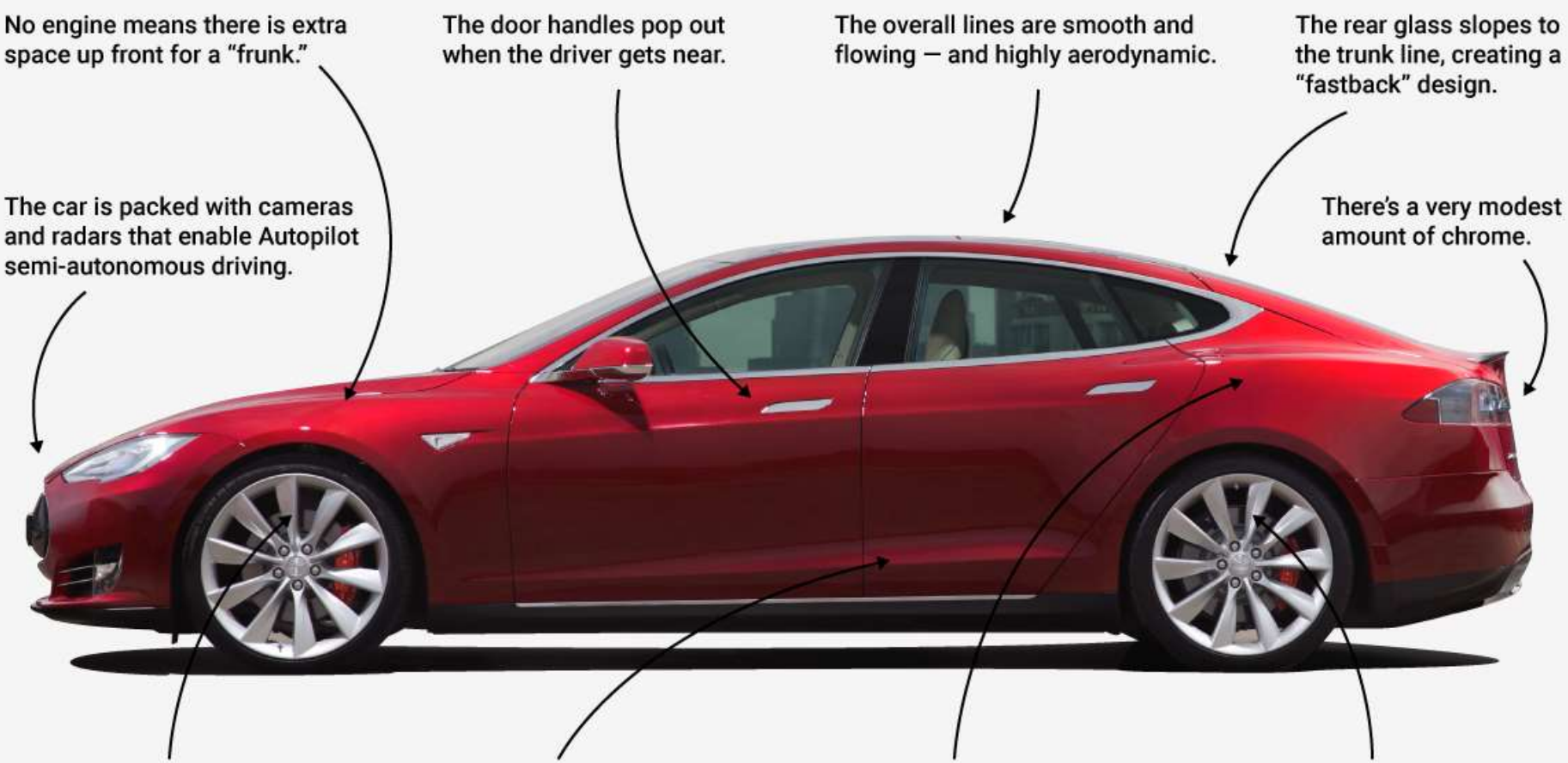


IMAGE: Tesla

BUSINESS INSIDER

- Deliver real value
- Simple, intuitive, and anticipates needs
- Exceed expectations
- Emotionally resonates
- Change users' life for the better

Great things are built in team

how to build things that people want?

How to Build Things People Want

- Know exactly what people want (how hard it is to come up with things people want?)
- Assemble a great team.
- Have a plan. A solid plan!
- Build that thing.
- Test whether the thing you built is what people actually want.

Importance of Software Engineering

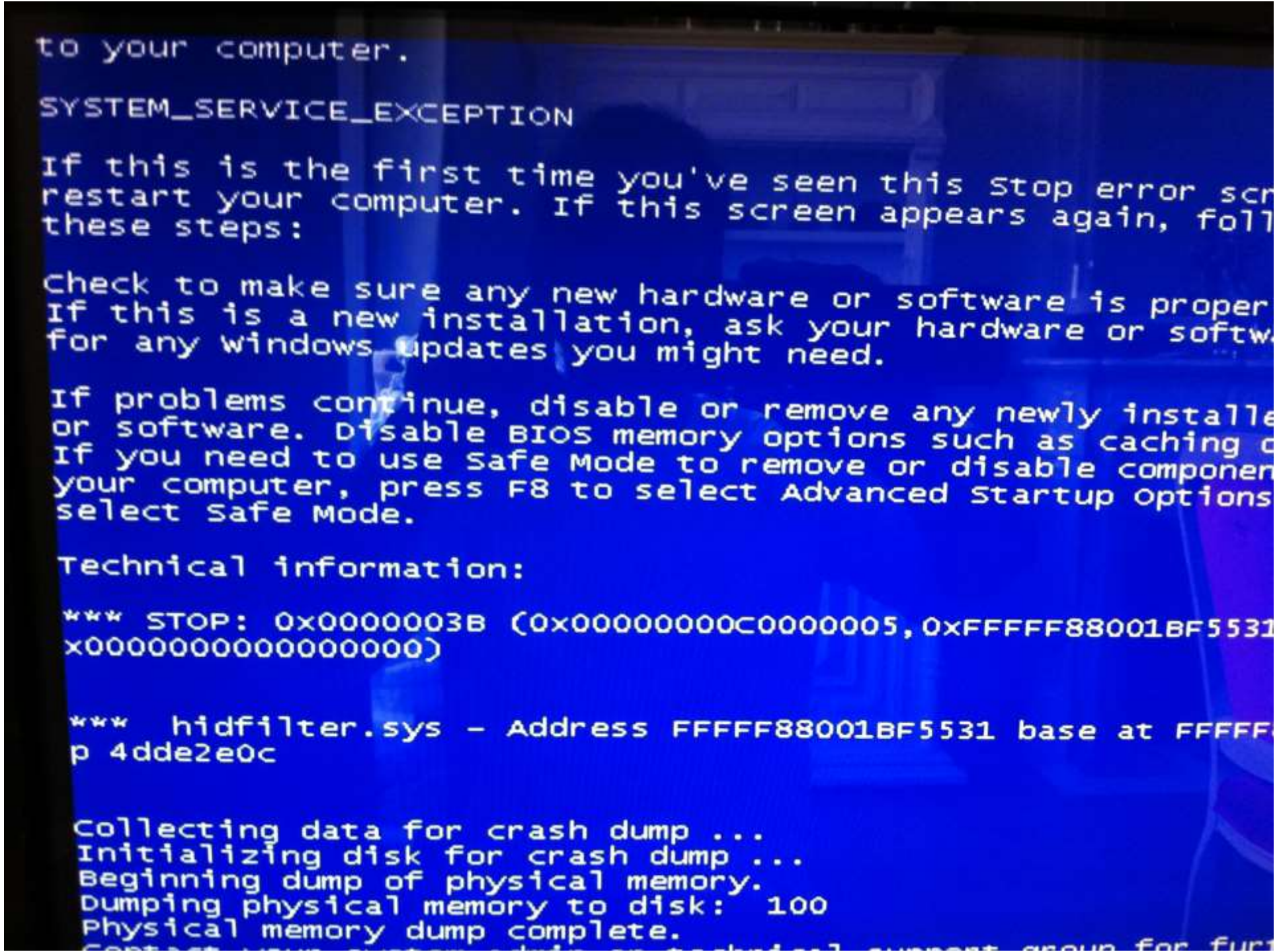
CODING

- Implementing software
- Getting computer to do what you want
- Between you and computer
- Ensuring correct function

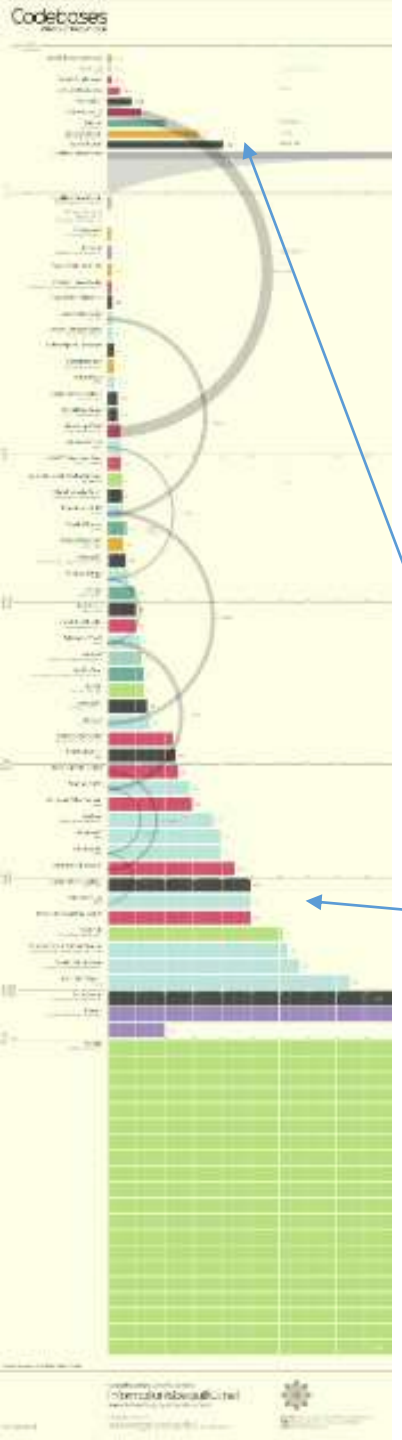
ENGINEERING

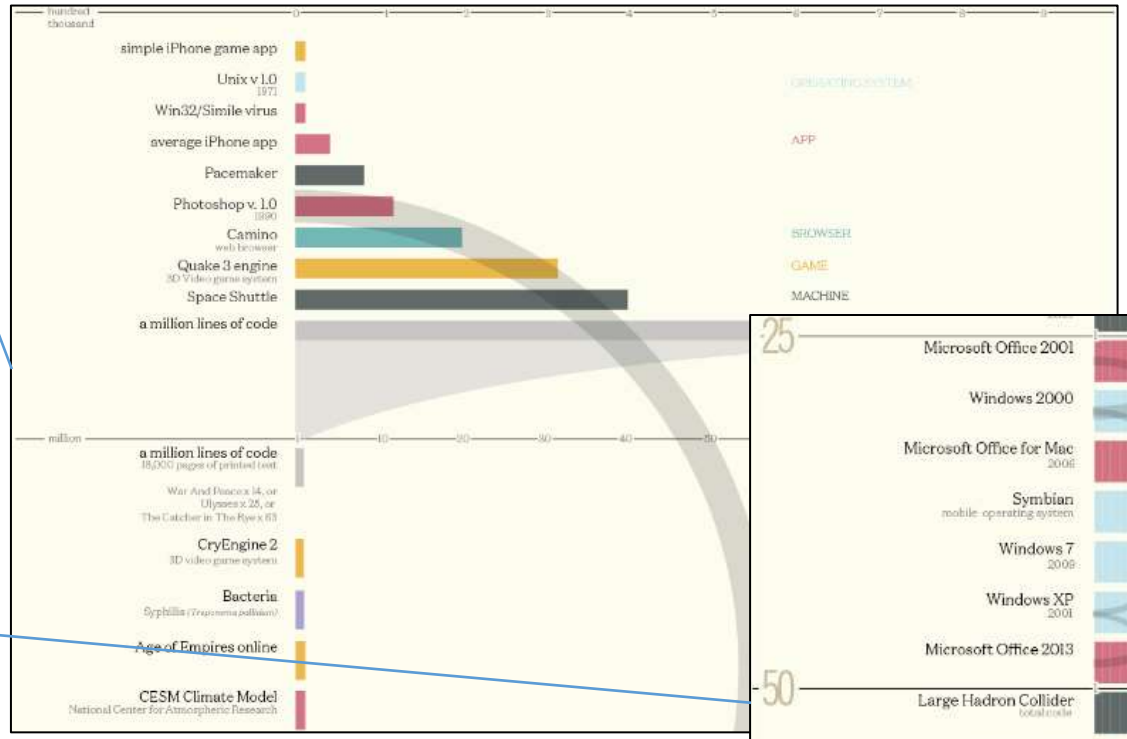
- Software life cycle
- Quality of what you created
- Between you and teammates
- Ensuring successful system

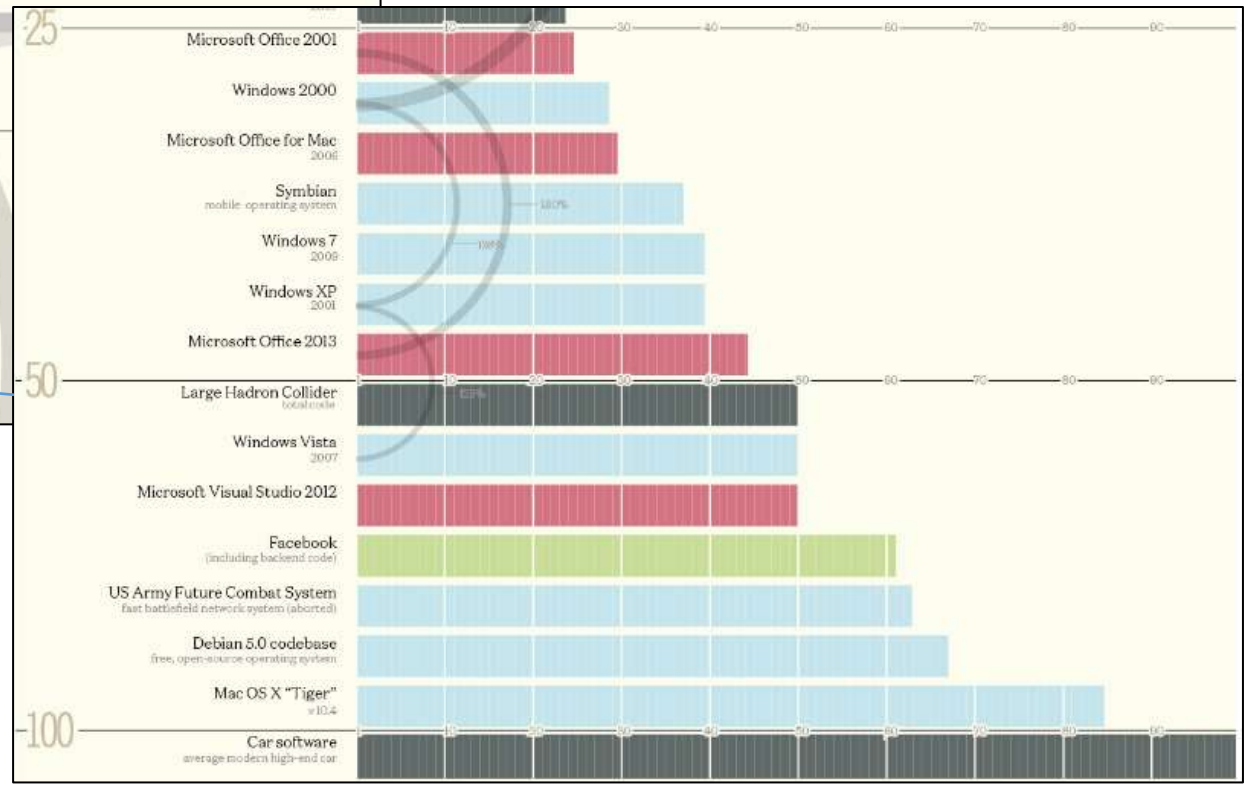
Coding is hard,
software engineering is harder


to your computer.
SYSTEM_SERVICE_EXCEPTION
If this is the first time you've seen this Stop error scr
restart your computer. If this screen appears again, foll
these steps:
Check to make sure any new hardware or software is proper
If this is a new installation, ask your hardware or softw
for any Windows updates you might need.
If problems continue, disable or remove any newly installe
or software. Disable BIOS memory options such as caching o
If you need to use Safe Mode to remove or disable componen
your computer, press F8 to select Advanced Startup Options
select Safe Mode.
Technical information:
*** STOP: 0x0000003B (0x00000000C0000005,0xFFFFF88001BF5531
x0000000000000000)
*** hidfilter.sys - Address FFFFF88001BF5531 base at FFFFF
p 4dde2e0c
Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk: 100
Physical memory dump complete.

How many line of code?







https://informationisbeautiful.net/visualizations/million-lines-of-code/

SOFTWARE DEVELOPMENT STUDY

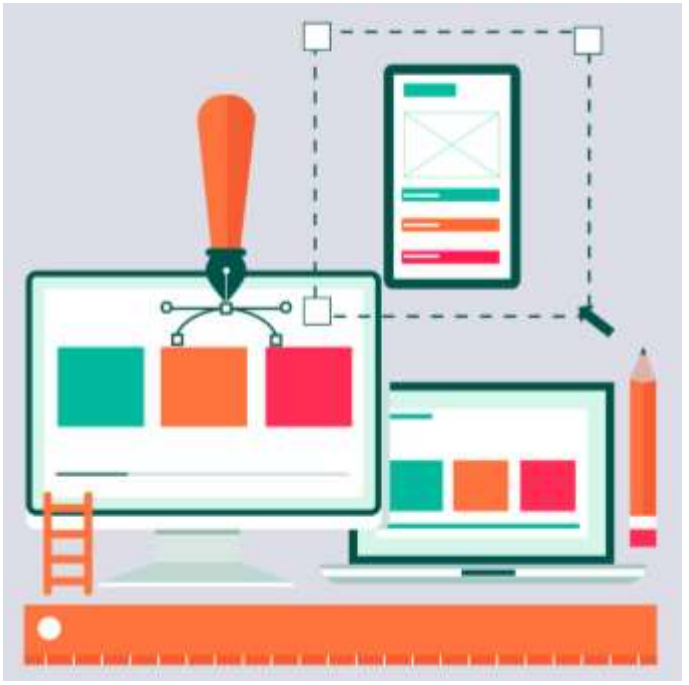
Davis, 1990

30% software usable as delivered
software usable after some changes

70% software usable after extensive modifications
software delivered, but never successfully used
software not delivered

you will get to learn how to manage large code base

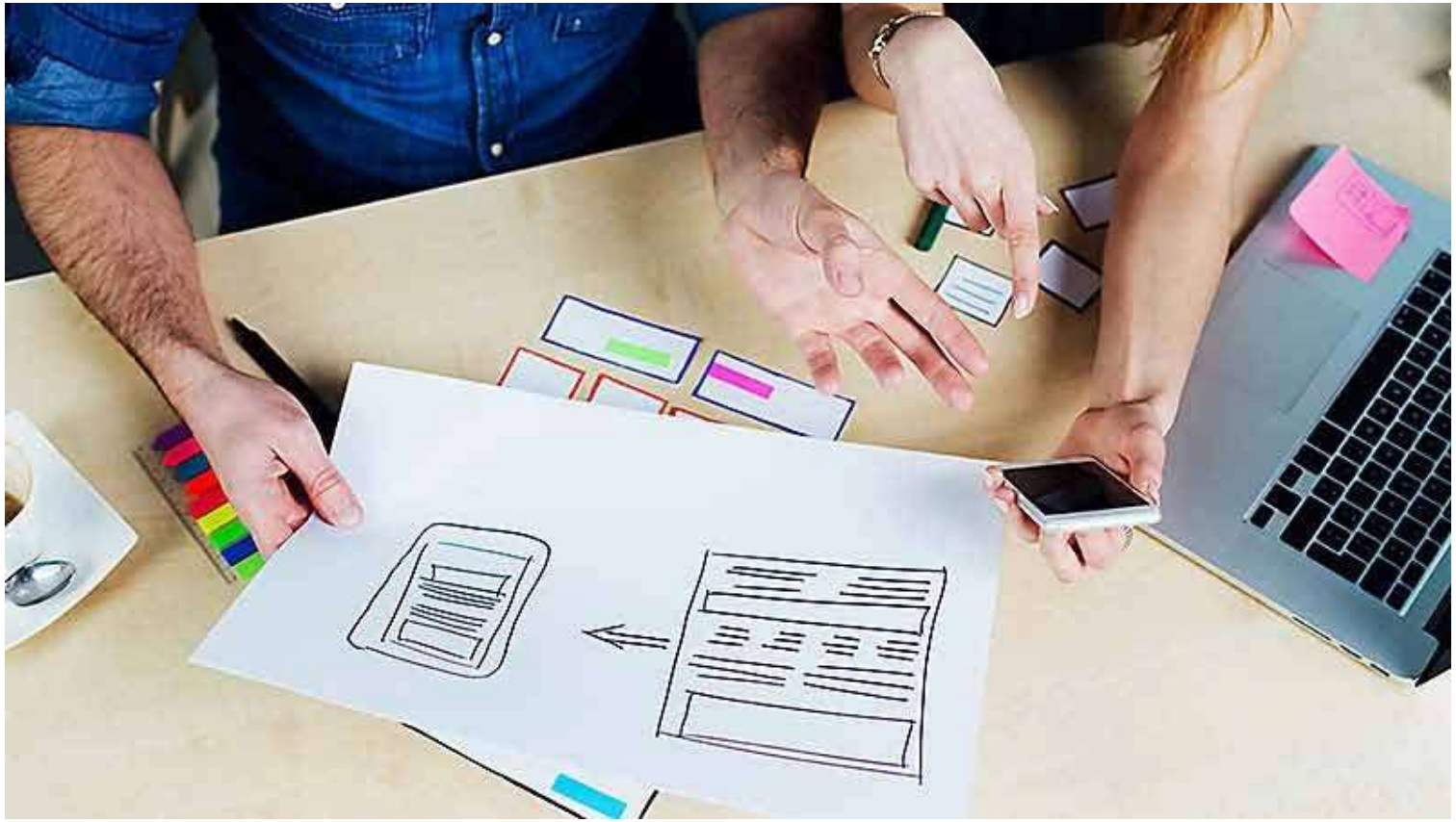
(And get to attempt to be the top 30%).



IDEA

SYSTEM

SYSTEMATIC and FORMAL way
to go from idea to workable system



FROM IDEAS TO DESIGN

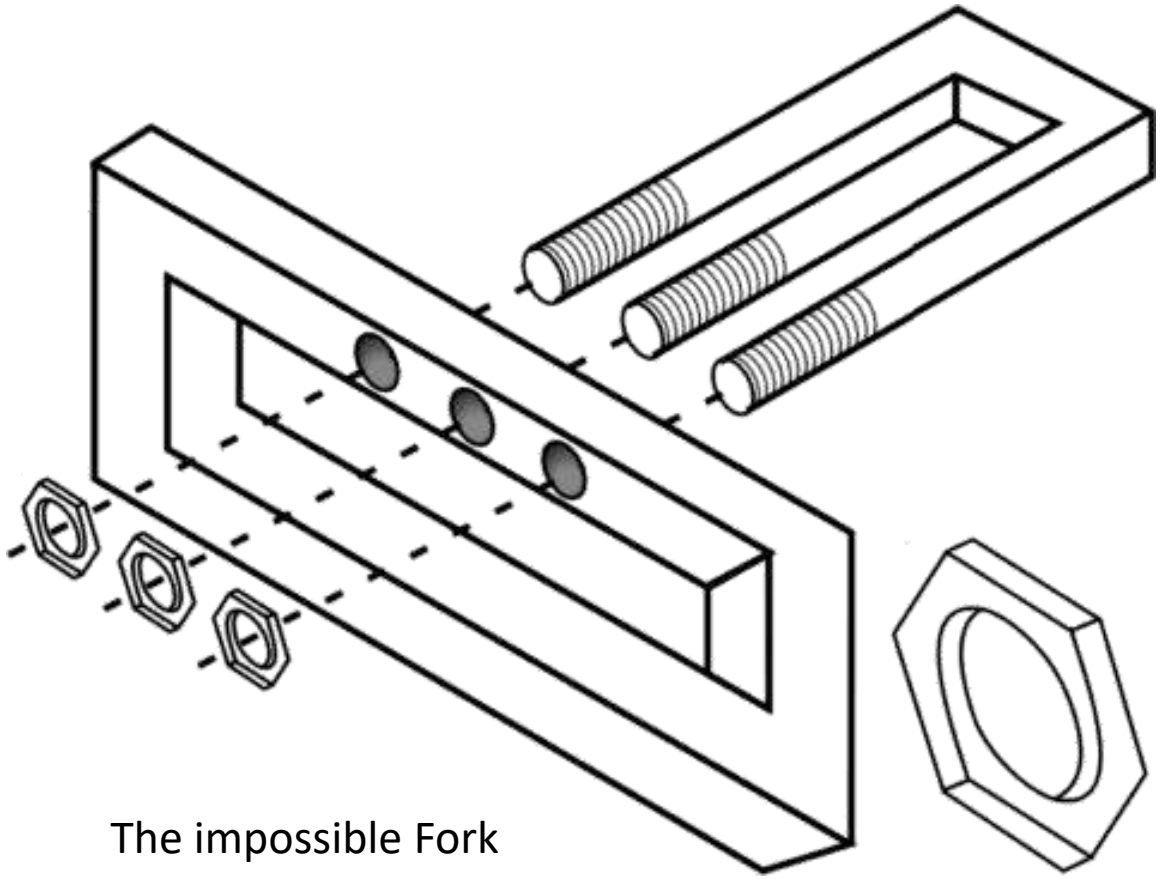
You will learn all those processes

(And try implementing some).

Software Development Life Cycle

Software Development for Robotics and Automation Engineering

Can you develop this?



The impossible Fork

http://www.moillusions.com/assembly-required-optical-illusion/

Software Engineering: A Working Definition

Software Engineering is a collection of techniques,
methodologies and tools that help with the production of

A **high quality** software system developed with a given **budget**
before a given **deadline** while **change** occurs

Challenge: Dealing with complexity and change

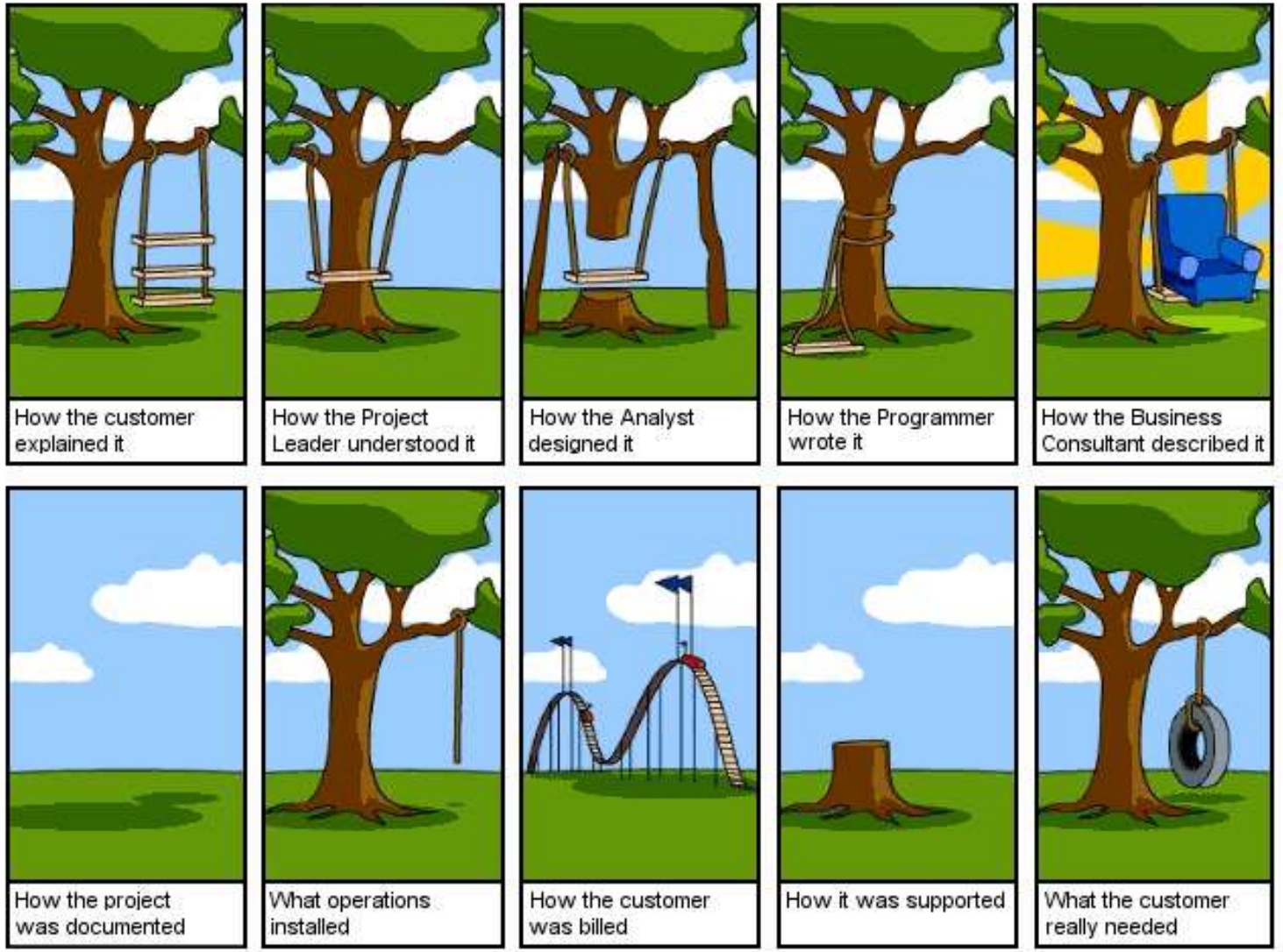
Software Development

- According to IBM Research: “Software development refers to a set of computer science activities dedicated to the process of creating, designing, deploying and supporting software.”
- Software development is a vast field and one that is creative, demanding and extremely rewarding. A career as a software developer gives you the opportunity to learn about the business as well as technology.
    1. Problem Solving
    2. Creative
    3. Project-Based
    4. Continuous Learning
    5. Collaboration

https://www.fdmgroup.com/5-reasons-why-you-should-be-a-software-developer/

Software Engineering - Phases

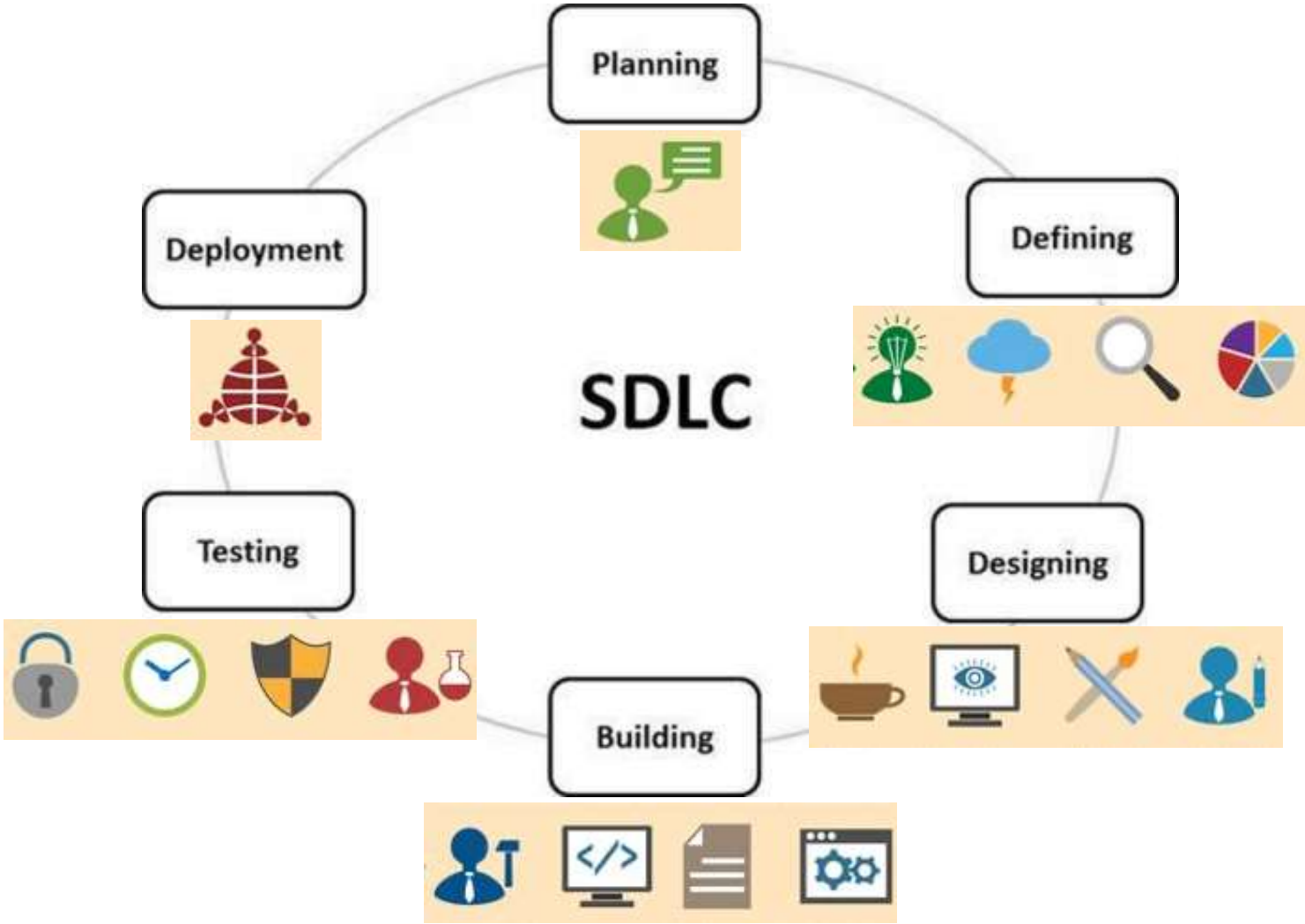
- Requirements analysis
- Systems Modeling
- Functional Modeling
- Implementation
- Evaluation
- Deployment
- Maintenance
- Documentation



ref: https://serlab.wordpress.com/2010/02/03/how-software-engineers-work/

Software Development Life Cycle (SDLC)

- The **SDLC** is a process used by the software industry to design, develop and test high quality software.
- The **SDLC** aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- It is also called as Software Development Process.
- ISO/IEC 12207 is an international standard for software life-cycle processes.

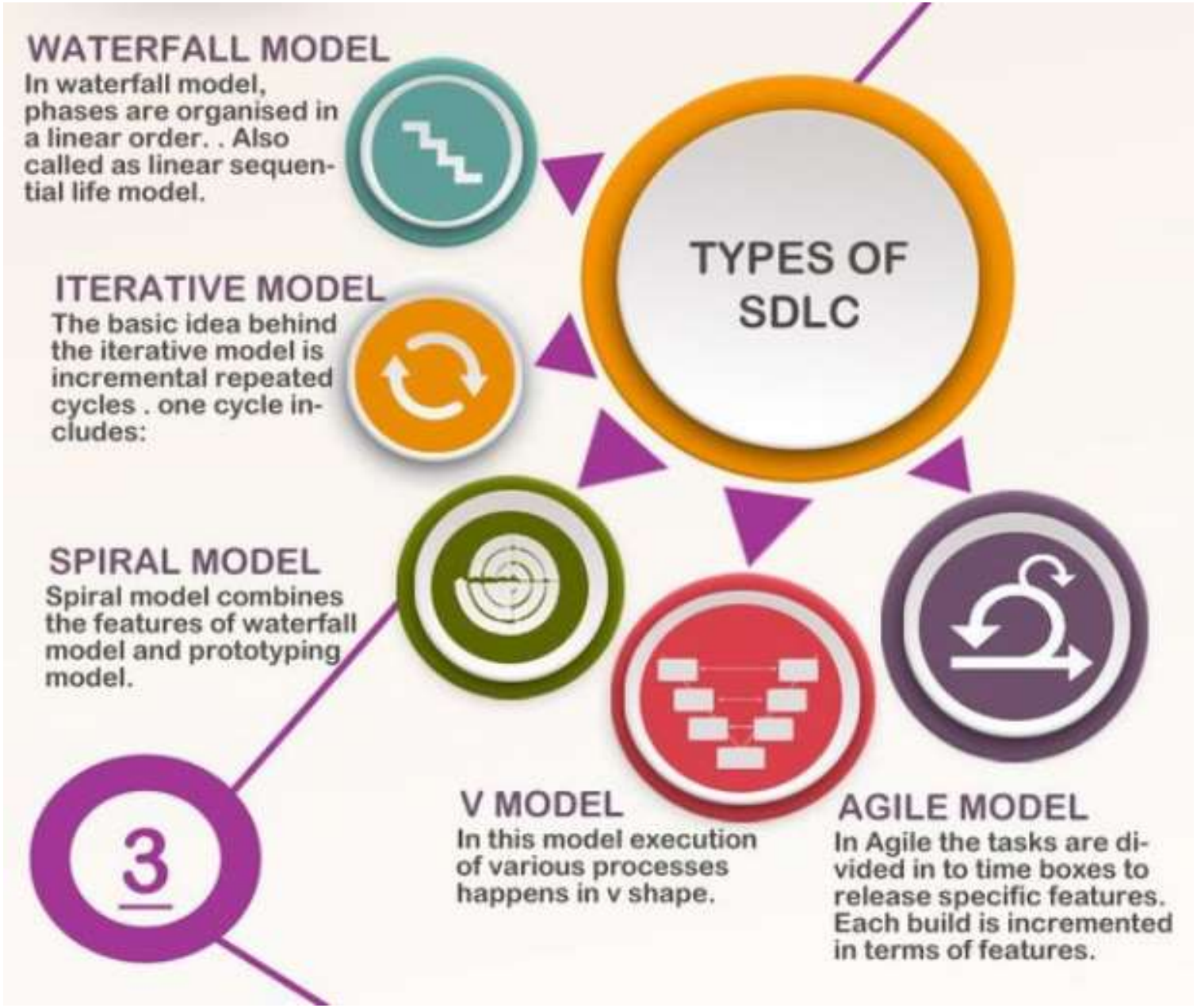


https://www.tutorialspoint.com/sdlc/sdlc_overview.htm

https://i.pinimg.com/564x/e6/ce/77/e6ce77b70fd1d1e3932c1a6254f02370.jpg

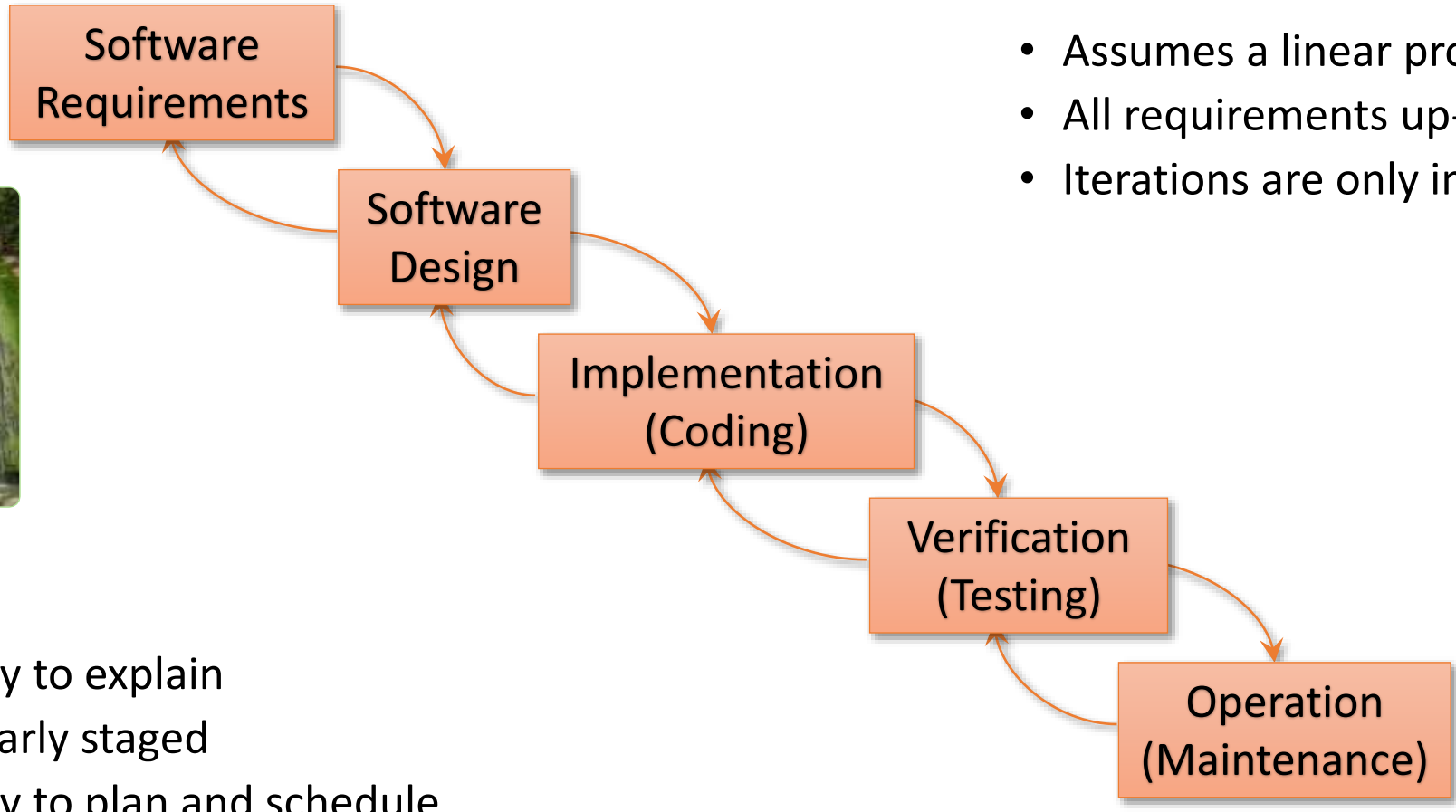
SDLC Models

- Waterfall Model
- Iterative Model
- Prototype Model
- Spiral Model
- V-Model
- Rational Unified Model
- Agile Model



https://i.pinimg.com/564x/e6/ce/77/e6ce77b70fd1d1e3932c1a6254f02370.jpg

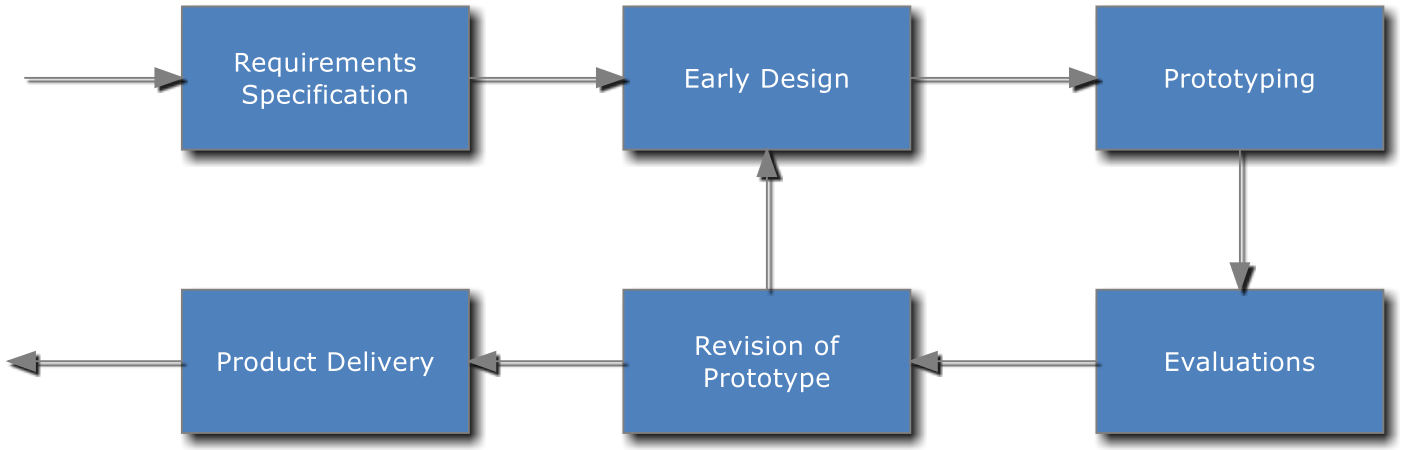
Waterfall Process



- Pros
  - Easy to explain
  - Clearly staged
  - Easy to plan and schedule
  - Clear Deliverables / milestones

- Cons
  - Assumes a linear project progression
  - All requirements up-front
  - Iterations are only implicit

Evolutionary Prototyping Process

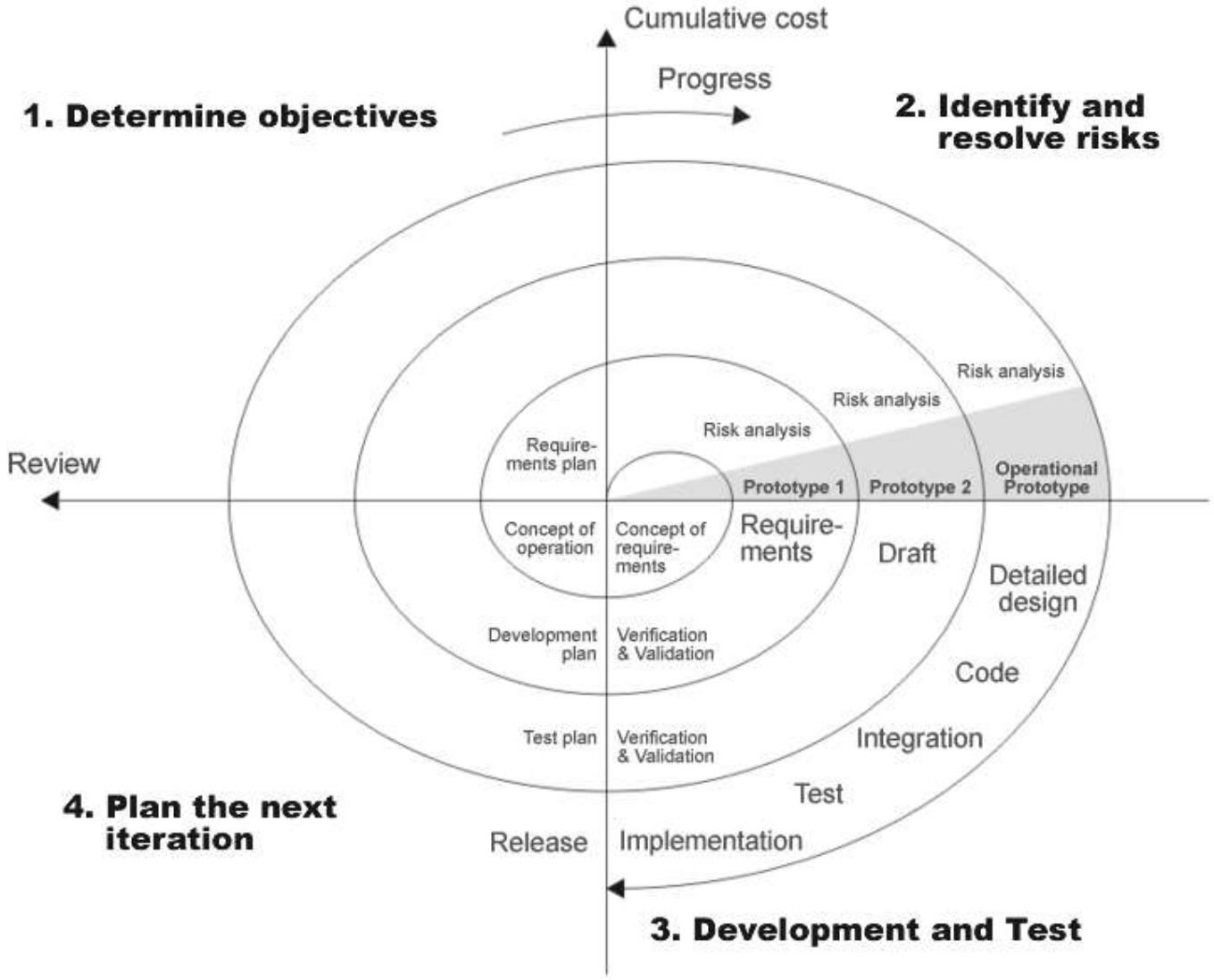


- Pros
  - Direct user involvement
  - Good fit to user expectations
  - Errors detected early
  - Solution as user buy in
- Cons
  - Often a patched design
  - Difficult to plan

Spiral Process

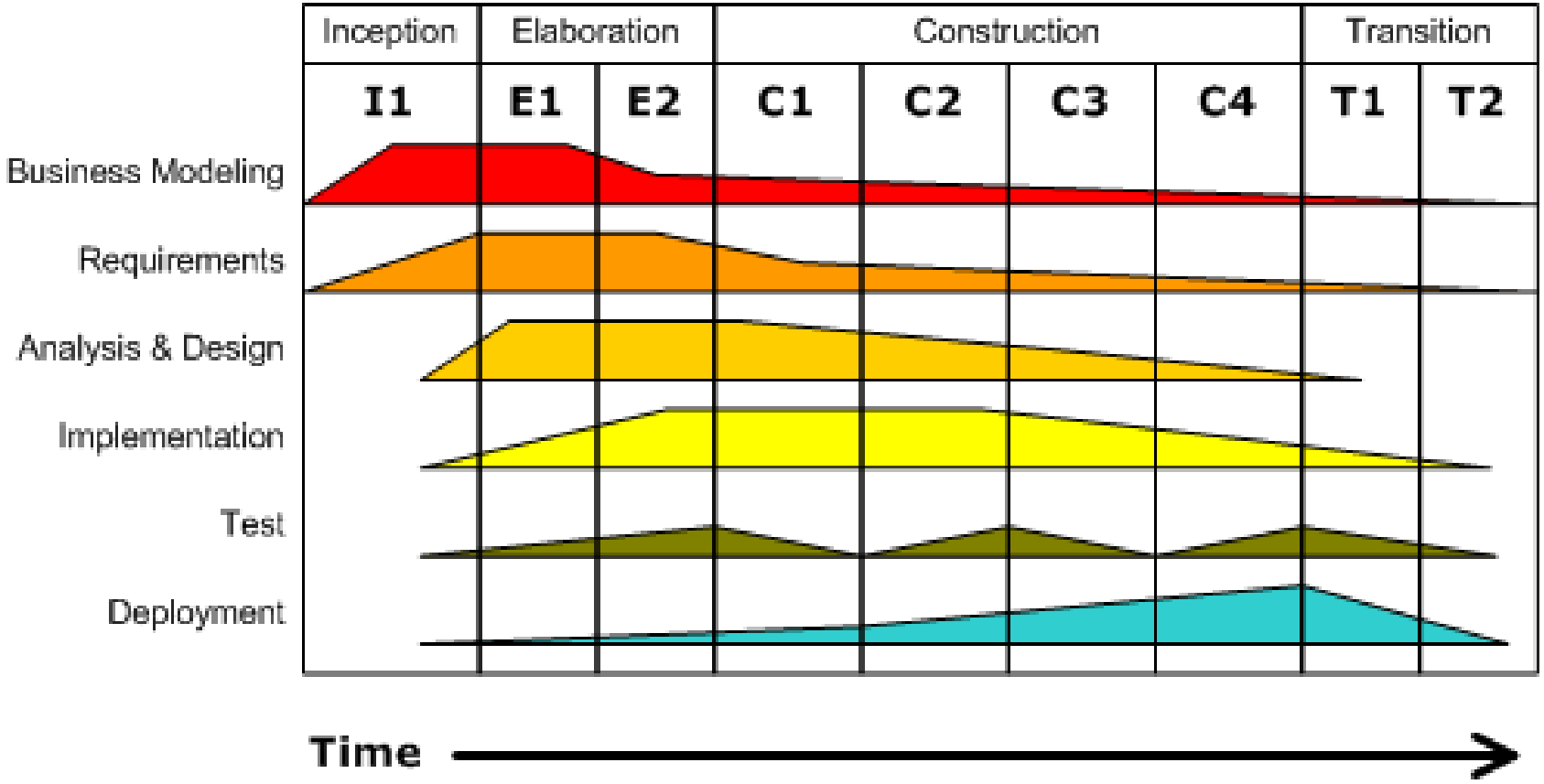
- Pro
  - Clear division of activities
  - Explicit deliverables

- Cons
  - Has potential to be costly
  - Could be slow to implement



Rational Unified Process (RUP)

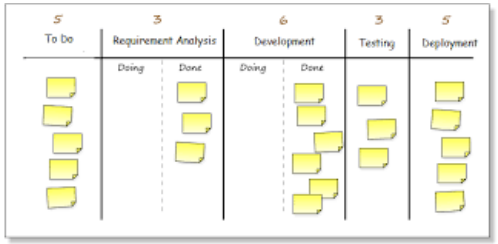
- Based on UML (Unified Modeling Language)
- OOP approach



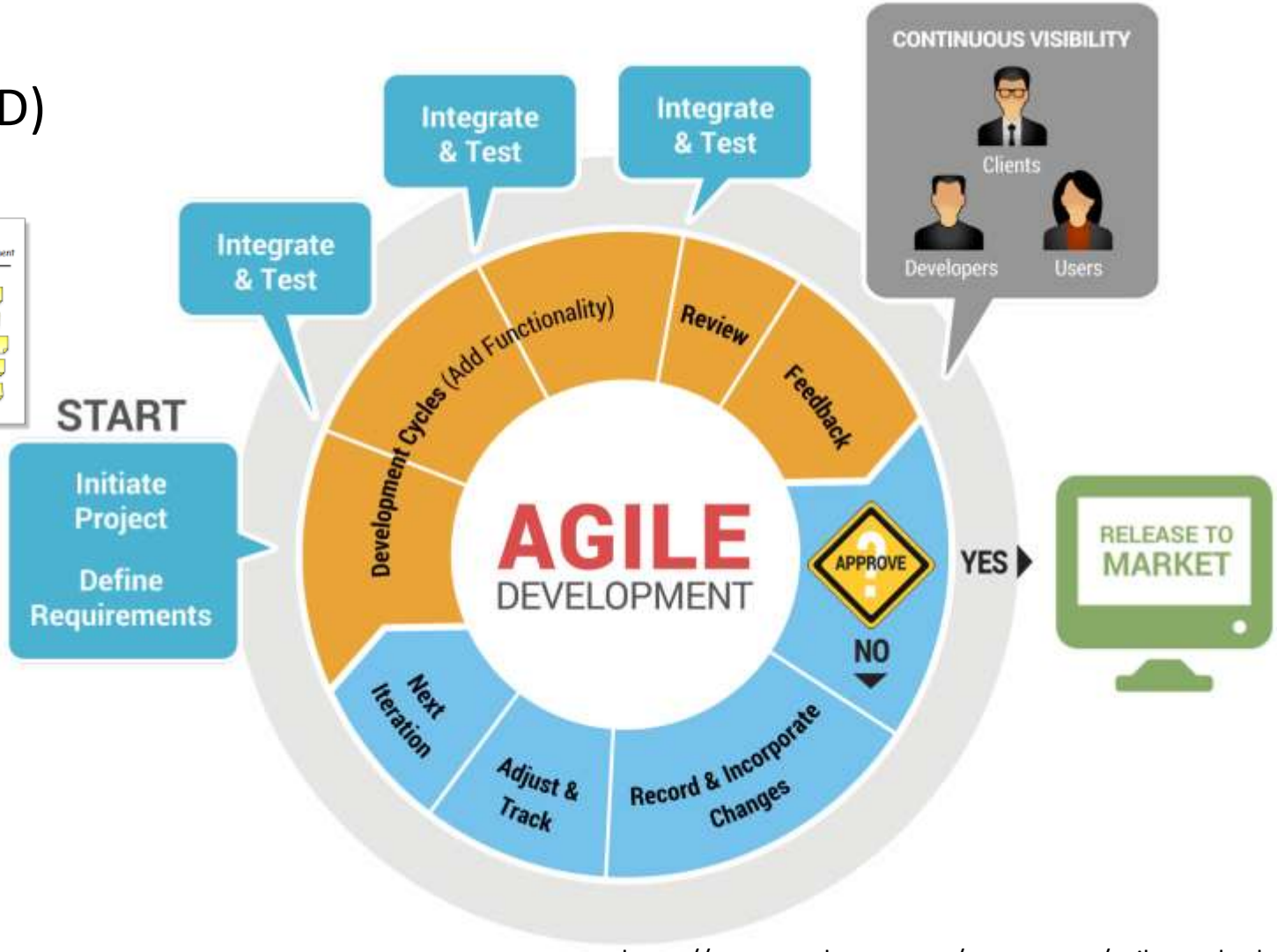
https://en.wikipedia.org/wiki/Rational_Unified_Process

Agile Process

- Test Driven Development (TDD)
  - Extreme programming (XP)
  - Scrum
  - Kanban

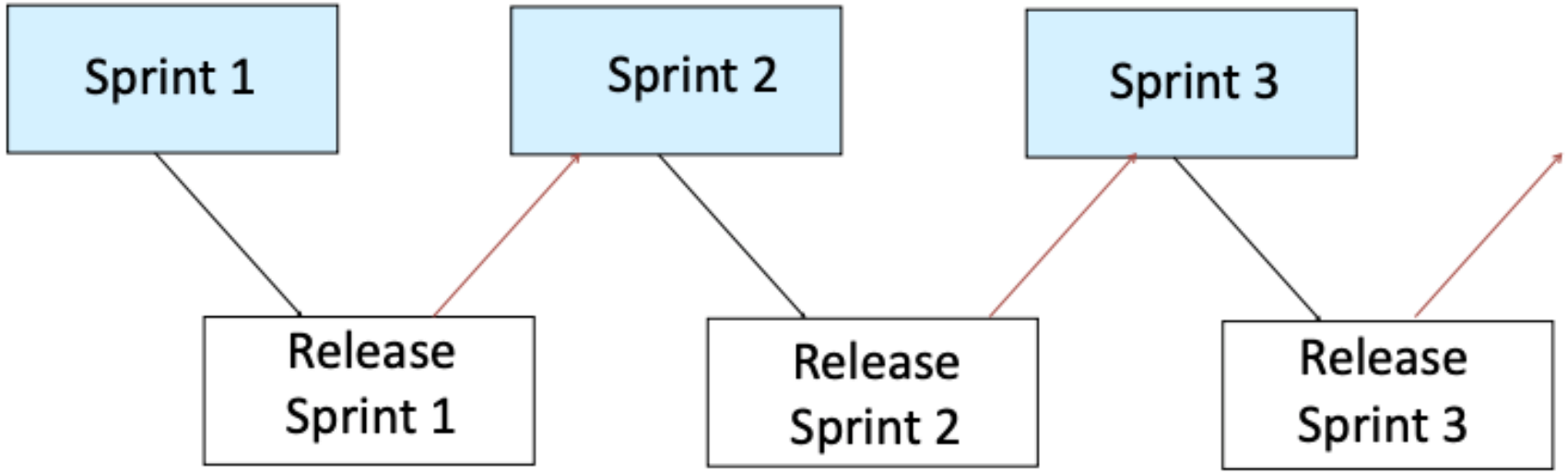


- Philosophy
  - Staff is competent
  - Simplicity
  - Agility
  - Focus on high-value
  - Tool independence



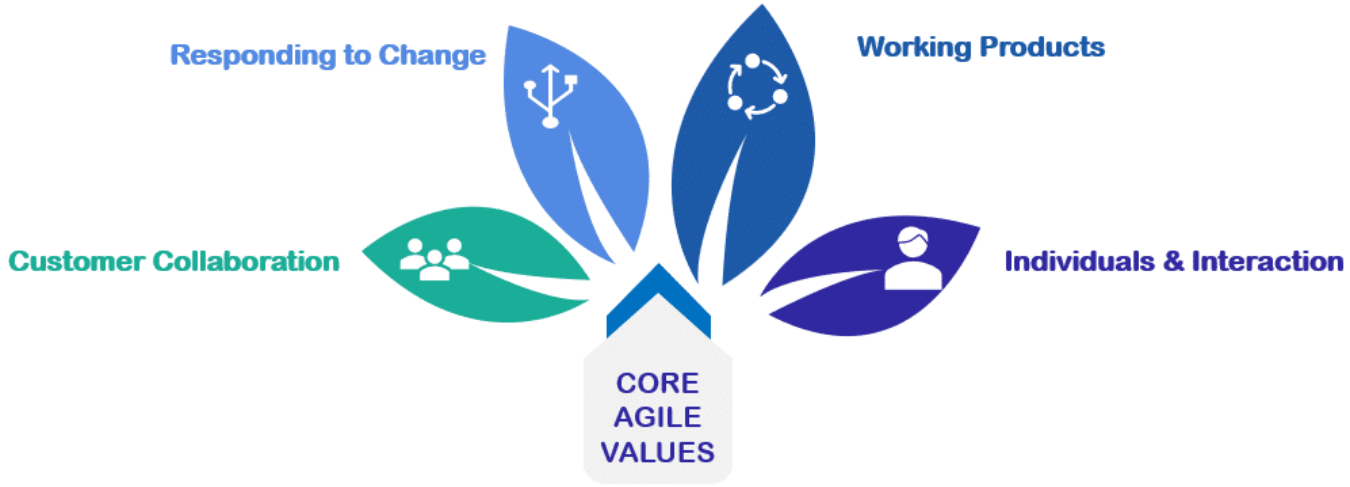
https://www.xandermar.com/our-process/agile-methodology

Agile Process



- The project is divided into a large number of small tasks, known as sprints.
- For each sprint, a team works through a full software development cycle including planning, requirements analysis, design, coding, testing, and acceptance testing, and release.
- Each sprint is completed in a fixed time period, e.g., four weeks.
- The size of a sprint is based on team size, e.g., 5-10 people.

http://www.cs.cornell.edu/courses/cs5150/2018sp/slides/3-process-examples.pdf



Agile
4 Values

Agile
12 Principles

- Satisfy The Customer
- Welcome Changing Requirements
- Deliver Working Software Frequently
- Collaborate Daily
- Motivated Individuals
- Face-to-face Conversation
- Measure Of Progress Through Working Product
- Promote Sustainable Development
- Continuous Attention To Technical Excellence
- Simplicity Is Essential
- Self-organizing Teams
- Regularity Reflect On Continuously Improving

https://www.toolsqa.com/agile/agile-manifesto/

Example Use of Processes

Large software development organizations have their own internal processes that are designed for their needs. For example:

- Amazon.com (Internet commerce) makes extensive use of sprints. Most software development is divided into increments of about four weeks elapsed time.
- Lockheed Martin (government contractor) follows a sequential process that fits with the way that the US government manages software contracts.
- SAP (business software) emphasizes the functionality that is seen by their business customers. Much of the development is suitable for a sequential process.
- Microsoft (PC software) places great emphasis on testing with a very wide variety of equipment and backward compatibility. Much of the development uses a spiral process.

http://www.cs.cornell.edu/courses/cs5150/2018sp/slides/3-process-examples.pdf

How to Choosing a Model

Appropriate model vs Wrong model

- Requirement understanding
- Expected lifetime
- Risk
- Schedule Constraints
- Interaction with management / customers
- Expertise

How to Choosing a Model

Changes during the software development process are expensive.

- If the requirements are poorly understood, or expected to change, select a process that keeps flexibility. Iterative refinement, agile sprints, phased implementation.
- If a big software system has many inter-related components, avoid major changes to the design of a system during development. Sequential process, such as the modified waterfall model.
- If the market for the software is poorly understood, use a process that gets operational software in front of customers as quickly as possible. Incremental, agile sprints.

http://www.cs.cornell.edu/courses/cs5150/2018sp/slides/3-process-examples.pdf

Software Processes

Completed projects should have included all the basic process steps but ... the development process is always partly evolutionary.

Risk is lowered by:

- Prototyping key components
- Frequent releases, or dividing large projects into phases
- Early and repeated testing with users and customers
- Following a visible software process
- Making use of reusable components

It is never possible to complete each step without provision for revision. This is known as throwing it over the wall.

http://www.cs.cornell.edu/courses/cs5150/2018sp/slides/3-process-examples.pdf

Classic Mistakes

- **People**
  - One people cannot do all
  - Work environment
  - People management
- **Process**
  - Scheduling issues
  - Planning issues
  - Failures
- **Product**
  - Lot of requirement is not necessary
  - Feature not in the plan
  - Research is not Development
- **Technology**
  - Not rely to much on technology
  - Changing tools in the middle of project
  - No version control

Summary of Classic Mistakes

People-Related Mistakes	Process-Related Mistakes	Product-Related Mistakes	Technology-Related Mistakes
1. Undermined motivation 2. Weak personnel 3. Uncontrolled problem employees 4. Heroics 5. Adding people to a late project 6. Noisy, crowded offices 7. Friction between developers and customers 8. Unrealistic expectations 9. Lack of effective project sponsorship 10. Lack of stakeholder buy-in 11. Lack of user input 12. Politics placed over substance 13. Wishful thinking	14. Overly optimistic schedules 15. Insufficient risk management 16. Contractor failure 17. Insufficient planning 18. Abandonment of planning under pressure 19. Wasted time during the fuzzy front end 20. Shortchanged upstream activities 21. Inadequate design 22. Shortchanged quality assurance 23. Insufficient management controls 24. Premature or too frequent convergence 25. Omitting necessary tasks from estimates 26. Planning to catch up later 27. Code-like-hell programming	28. Requirements gold-plating 29. Feature creep 30. Developer gold-plating 31. Push me, pull me negotiation 32. Research-oriented development	33. Silver-bullet syndrome 34. Overestimated savings from new tools or methods 35. Switching tools in the middle of a project 36. Lack of automated source-code control

http://www.stevemcconnell.com/rdenum.htm