Embedded System

Lecture 2
Layered Architecture

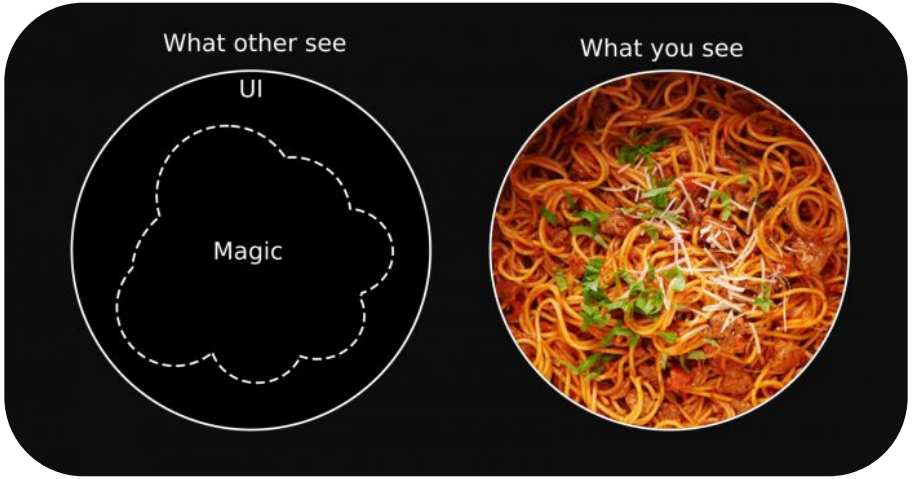
IMT SCHOOL

Software Architecture

Software Architecture

Imagine having a source code file contains 10 thousands lines of code. If you have a ***bug*** in this code, how complex is finding the ***bug*** … ?

This is what's called the **Spaghetti Code**, which is the code that ***unstructured*** and ***difficult to maintain***.



What is SW Architecture ?

Software architecture is, simply, the organization of a system. This organization includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software. In many cases, it can also include the evolution of the software into the future. In Embedded Systems we use a major type of software architecture called Layered Architecture.

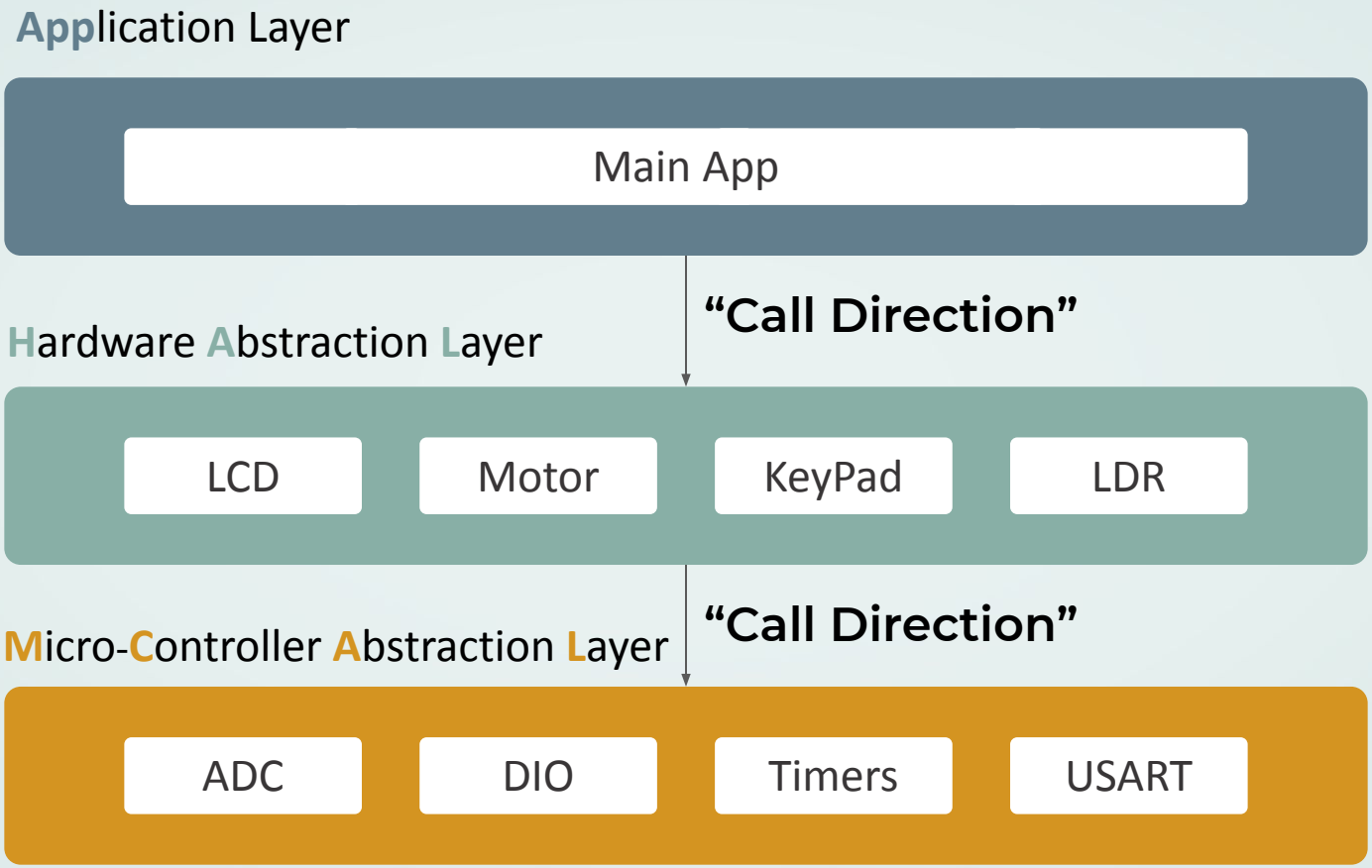
In the layered architecture the software is divided into small parts called ***software components*** (SWC). Software components related to each other are organized in a horizontal layer. Each layer is performing a specific role

IMT SCHOOL

IMT SCHOOL

Layered Architecture

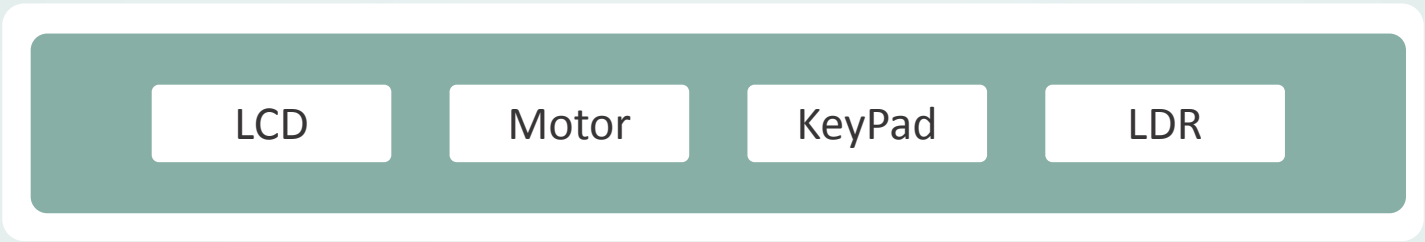
Layered Architecture



MCAL

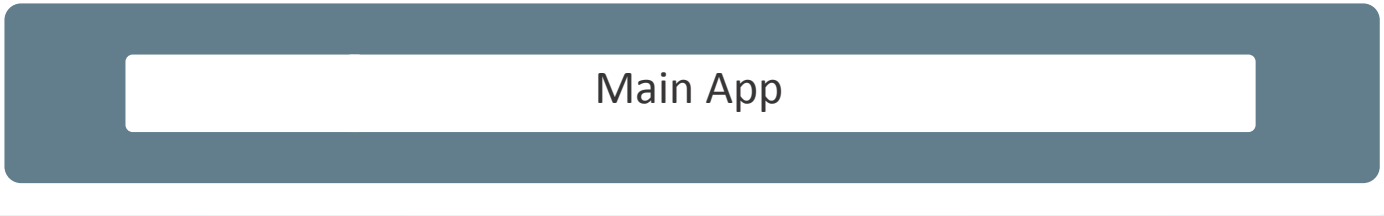
MCAL: Micro-**C**ontroller **A**bstraction **L**ayer
Software related to any peripheral inside the microcontroller

HAL



HAL: Hardware **A**bstraction **L**ayer
Software related to any on board hardware element

APP



App: Application Layer
System application

Advantages of Layered Architecture

01

Modularity

In a Layered architecture we separate the user application from the hardware drivers from the microcontroller specific drivers.

02

Portability

Changing any part of the software part would change its layer only. For example, if we need the same application with a new microcontroller, we shall only change the MCAL.

03

Reusability

Code could be easily ***reused*** in different applications and systems.

04

Maintainability

Debugging and ***Testing*** is now much easier in small parts of the software instead of having a very long and complex one.

Building Drivers

Building DIO Driver

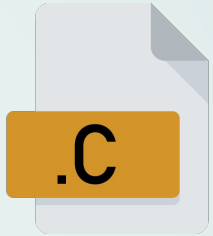
The Simplest driver consists of only 2 files

Program File

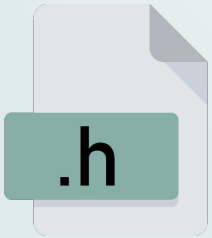
01

C file contains the implementations of the functions provided by the driver

ex: DIO_prg.c



Interface File

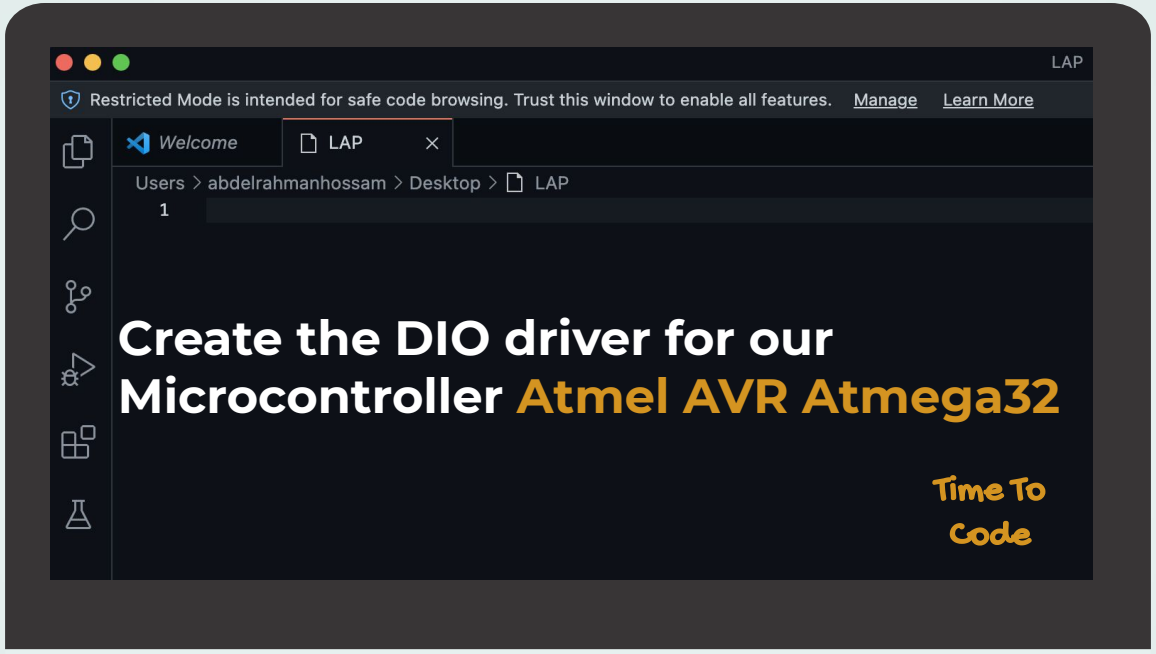


Header file contains the prototypes of the functions provided by the driver to be used by other SWCs that need to use this driver

ex: DIO_int.h

02

LAb 1



IMT SCHOOL

Software Engineering

IMT SCHOOL

What is Software Engineering

Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.

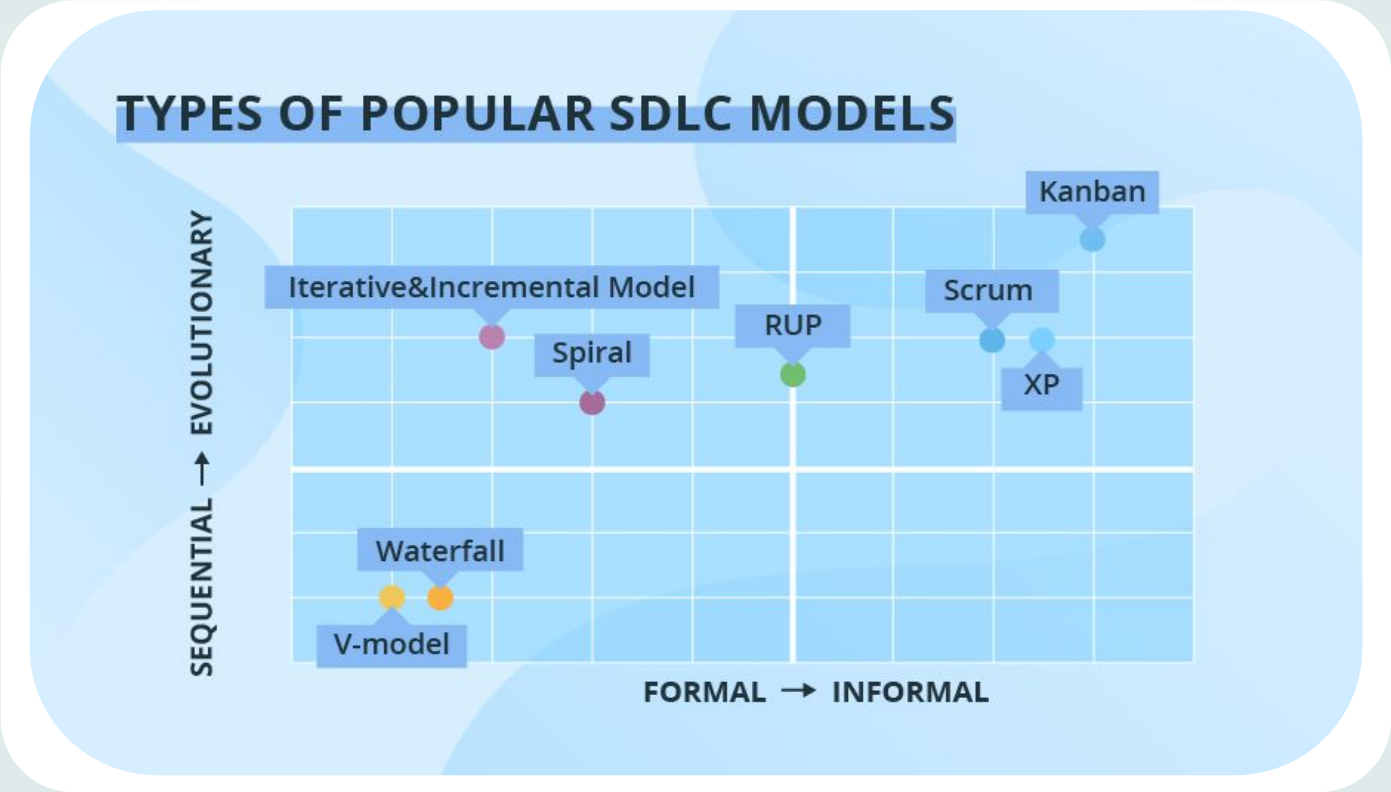
Software Development Models

Software development life cycle (SDLC) models show the ways to navigate through the complex and demanding process of software building. A project's quality, timeframes, budget, and ability to meet the stakeholders' expectations largely depend on the chosen model.

Today, there are more than **50** recognized SDLC models in use. None of them is perfect, and each brings its favorable aspects and disadvantages for a specific software development project or a team. According to our 32 years of experience in software development, we've chosen 8 most popular models to look into their essence and compare for core features.

IMT SCHOOL

The outline of popular SDLC models



The outline of popular SDLC models

All SDLC models can be structured into several groups depending on how they approach workflow organization – linearly or iteratively – and what kind of relationships are established between the development team and the customer.

The types in the lower quadrants of the chart take the sequential flow. They are easy to implement, use and manage. As you move higher, the process becomes less rigid and offers more flexibility when it comes to changes in the requirements for future software.

The models on the left side of the chart imply low customer involvement; as you move toward the right side, the models grow more ‘cooperative’ and include customers into different stages of software development life cycle more intensively.

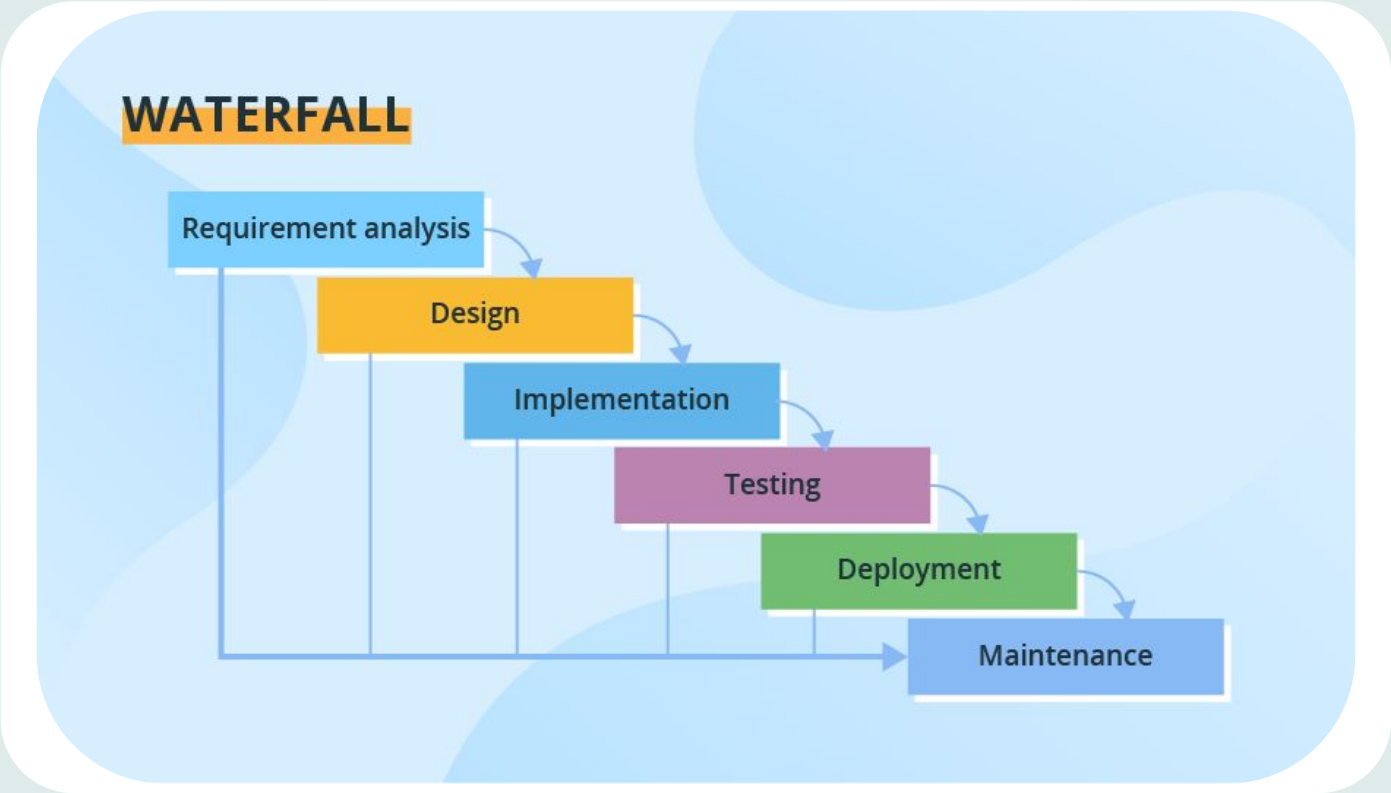
IMT SCHOOL

SDLC Models

Waterfall Model

Through all development stages (analysis, design, coding, testing, deployment), the process moves in a cascade mode. Each stage has concrete deliverables and is strictly documented. The next stage cannot start before the previous one is fully completed. Thus, for example, software requirements cannot be re-evaluated further in development. There is also no ability to see and try software until the last development stage is finished, which results in high project risks and unpredictable project results. Testing is often rushed, and errors are costly to fix.

Waterfall Model



When to Use

01

Simple small or mid-sized projects with clearly defined and unchanging requirements (small company website development).

02

Projects with the need for stricter control, predictable budget and timelines (e.g., governmental projects).

03

Projects that must adhere to multiple rules and regulations (healthcare projects).

04

Projects where a well-known technology stack and tools are used.

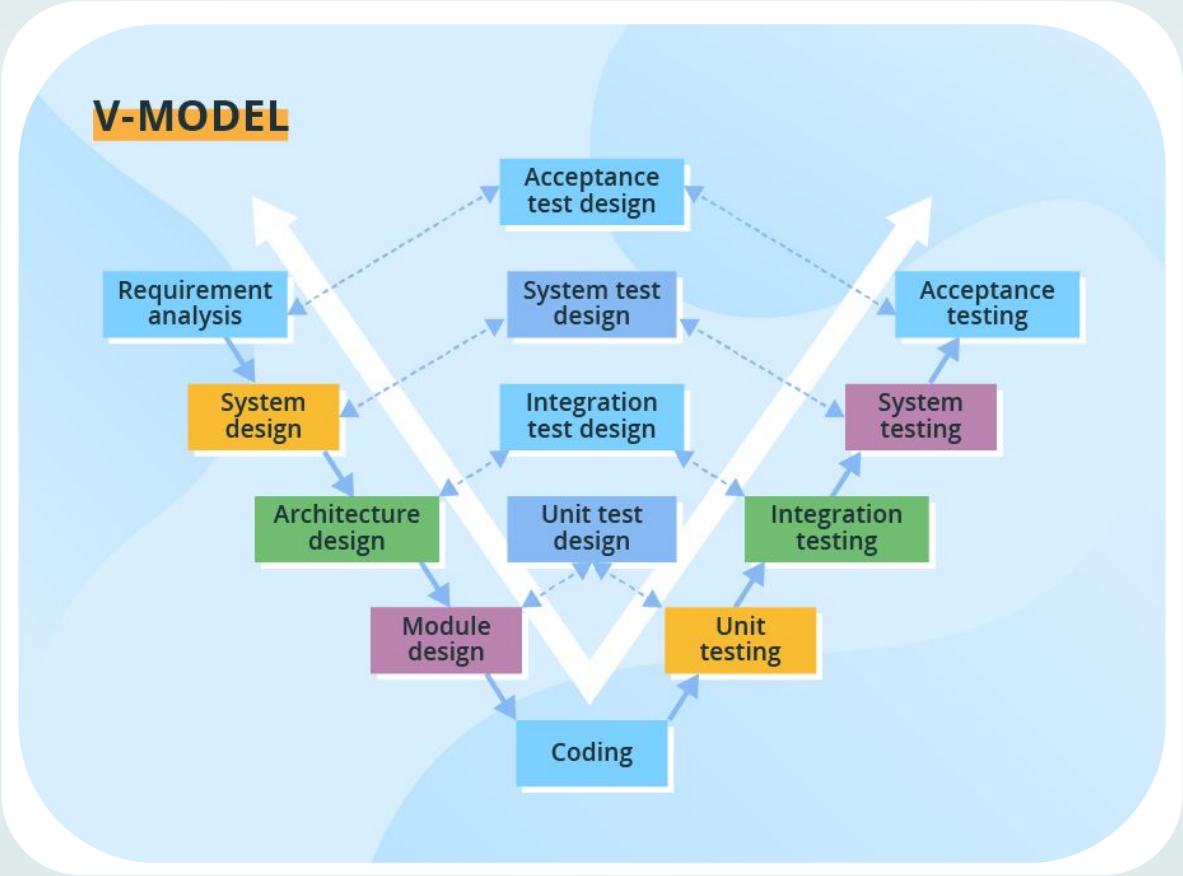
V-Model
Validation and Verification Model

The V-model is another linear model with each stage having a corresponding testing activity. Such workflow organization implies exceptional quality control, but at the same time, it makes the V-model one of the most expensive and time-consuming models. Moreover, even though mistakes in requirements specifications, code and architecture errors can be detected early, changes during development are still expensive and difficult to implement. As in the Waterfall case, all requirements are gathered at the start and cannot be changed.

Use cases:Projects where failures and downtimes are unacceptable (e.g., medical software, aviation fleet management software).

IMT SCHOOL

V Model


V-MODEL
Acceptance test design
Requirement analysis
System test design
Acceptance testing
System design
Integration test design
System testing
Architecture design
Unit test design
Integration testing
Module design
Unit testing
Coding

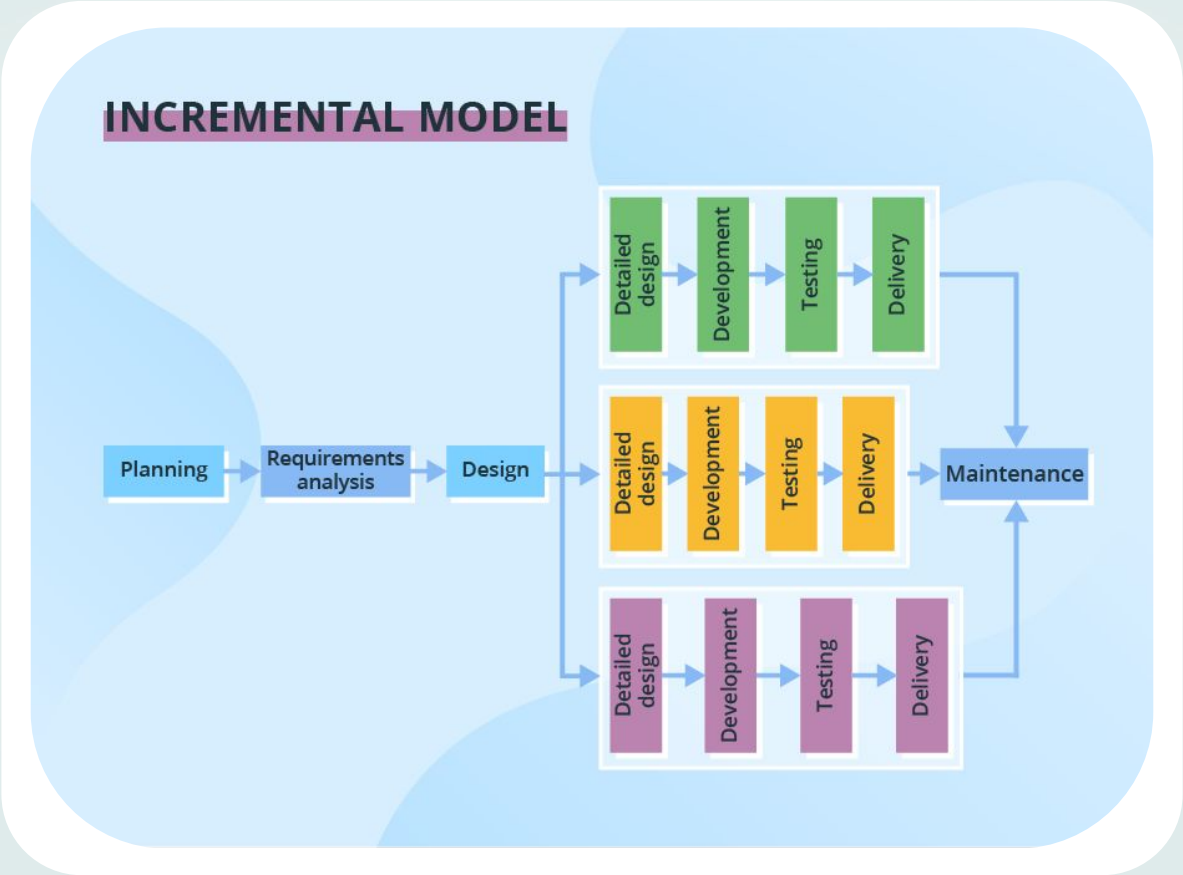
Incremental Model

The development process based on the **Incremental model** is split into several iterations (“Lego-style” modular software design is required!). New software modules are added in each iteration with no or little change in earlier added modules. The development process can go either sequentially or in parallel. Parallel development adds to the speed of delivery, while many repeated cycles of sequential development can make the project long and costly.

Use cases: Large, mission-critical enterprise applications that preferably consist of loosely coupled parts, such as microservices or web services.

IMT SCHOOL

Incremental Model



Iterative Model

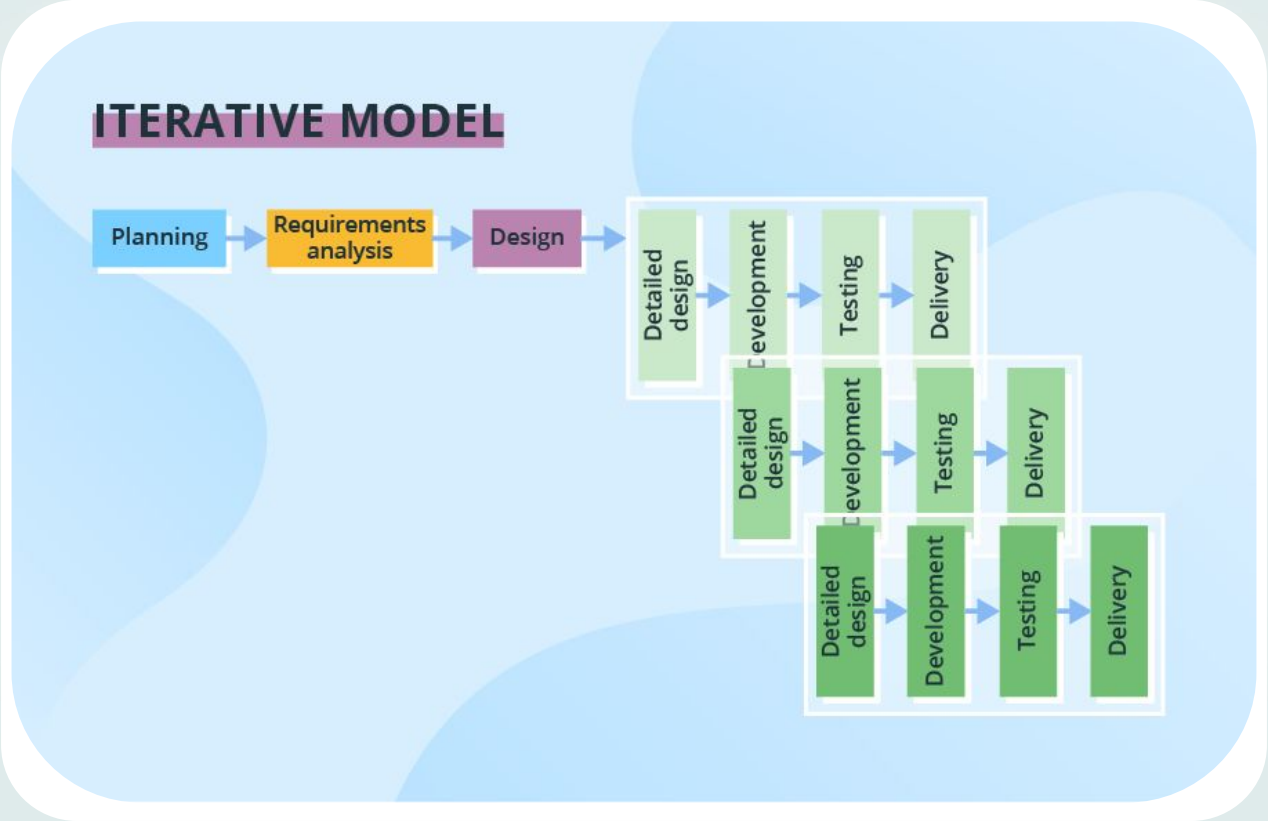
With **Iterative development** software changes on each iteration, evolves and grows. As each iteration builds on the previous one, software design remains consistent.

As software is delivered in parts, there is no need for a full specification from the project's start and small changes to requirements are possible in the course of the development process. However, the requirements can't change radically – major ones must be defined in the beginning, especially those for system design in case of Incremental development as further integration of the delivered software parts can become an issue.

This SDLC model typically entails some customer involvement because of the possible need in small requirements amendments during the development process.

IMT SCHOOL

Iterative Model



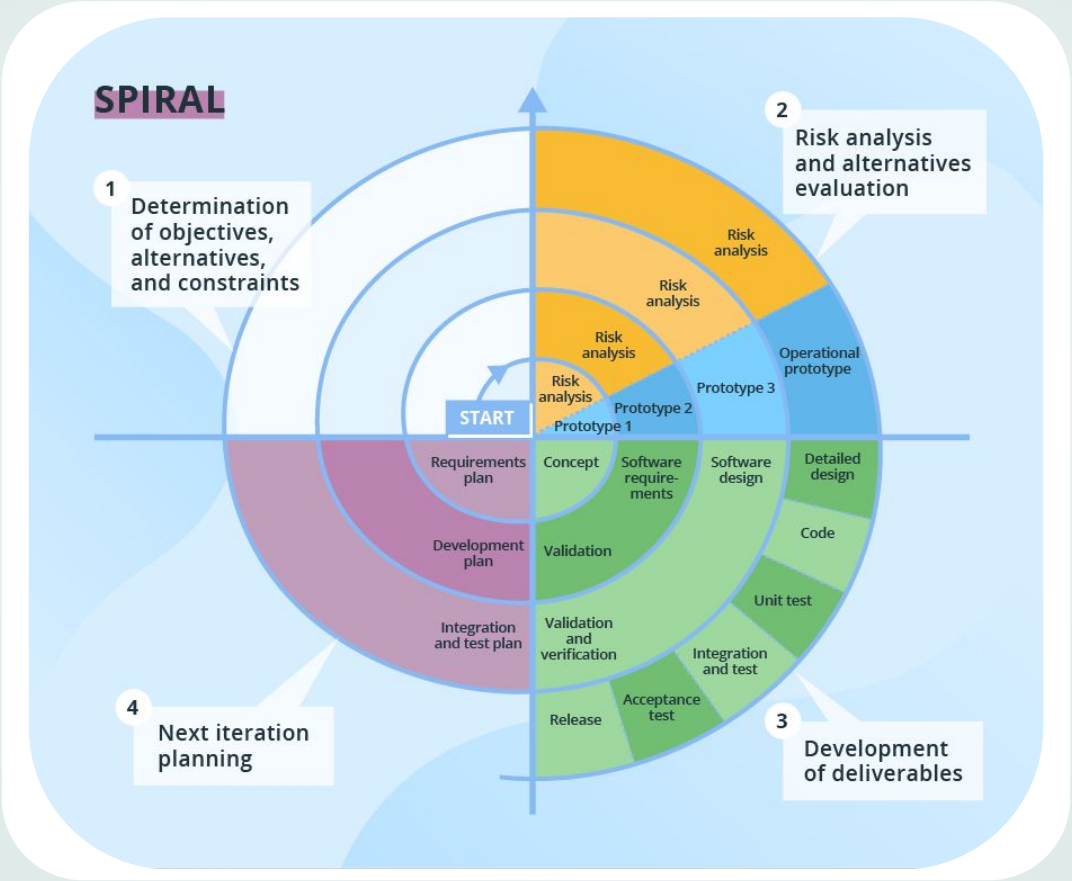
Spiral Model

The Spiral model puts focus on thorough risk assessment. Thus, to reap the benefits of the model to the fullest, you'll need to engage people with a strong background in risk evaluation. A typical Spiral iteration lasts around 6 months and starts with 4 important activities - thorough planning, risk analysis, prototypes creation, and evaluation of the previously delivered part. Repeated spiral cycles seriously extend project timeframes.

This is the model where intensive customer involvement appears. They can be involved in the exploration and review stages of each cycle. At the development stage, the customer's amendments are not acceptable.

IMT SCHOOL

Spiral Model



When to Use

01

Projects with unclear business needs or too ambitious/innovative requirements.

02

Projects that are large and complicated.

03

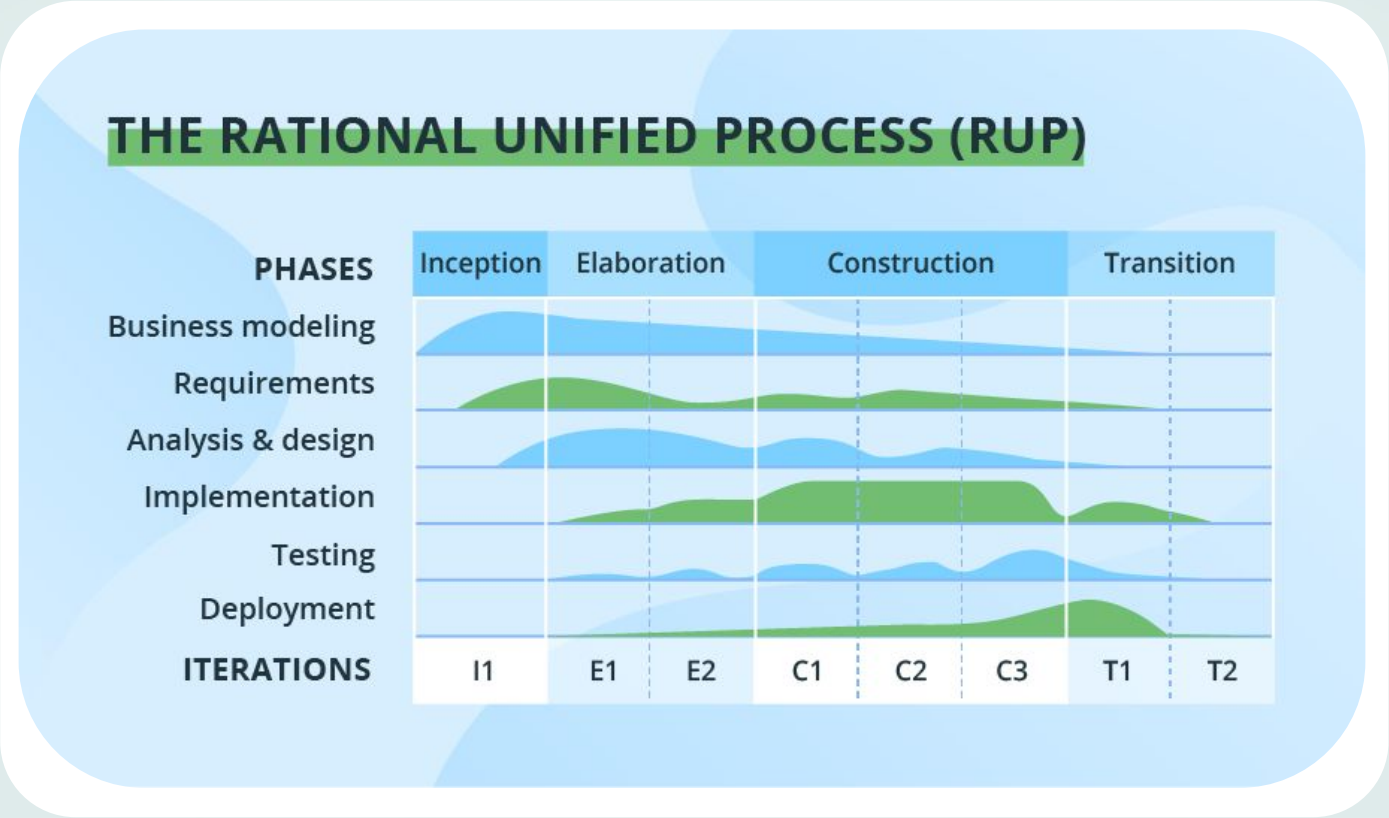
Research and development (R&D) activity or the introduction of a new service or a product.

The Rational Unified Process (RUP)

The Rational Unified Process (RUP) is also a combination of linear and iterative frameworks. The model divides the software development process into 4 phases – inception, elaboration, construction, and transition. Each phase but Inception is usually done in several iterations. All basic activities (requirements, design, etc.) of the development process are done in parallel across these 4 RUP phases, though with different intensity.

RUP helps to build stable and, at the same time, flexible solutions, but still, this model is not as quick and adaptable as the pure Agile group (Scrum, Kanban, XP, etc.). The degree of customer involvement, documentation intensity, and iteration length may vary depending on the project needs.

The Rational Unified Process (RUP)



When to Use

01

Use cases: Large and high-risk projects, especially, use-case based development and fast development of high-quality software.

The Agile Group

The rest of the SDLC models we've chosen fall under the umbrella of Agile. Nowadays, more than 70% of organizations employ this or that Agile approach in their IT projects. In general, at the heart of Agile are iterative development, intensive communication, and early customer feedback.

Each Agile iteration usually takes several weeks and delivers a complete working software version. The models of this group put more focus on delivering a functioning part of the application quickly. They pay less attention to detailed software documentation (detailed requirement specification, detailed architecture description), and more to software testing activities. This fosters quick development but considerably prolongs software transfer to the support team as well as makes its maintenance more complicated as more time is spent to find the problem when there's no detailed software description.

The Agile Group

Agile is about working in close collaboration both across the team and with the customers. At the end of each iteration, stakeholders review the development progress and re-evaluate the priority of tasks for the future iteration to increase the return on investment (ROI) and ensure alignment with user needs and business goals.

Accordingly, frequent releases are characteristic to the Agile models. They also allow for continuous software improvement with easy fixes and changes, quick updates, and feature addition, and help to deliver applications that satisfy users' needs better. However, the lack of detailed planning and openness to changes make it difficult to accurately estimate budget, time and people required for the project.

When to Use

01

Practically any startup initiatives, when end users' early feedback is required.

02

Most of mid-sized projects in custom software development where business requirements cannot be confidently translated to detailed software requirements.

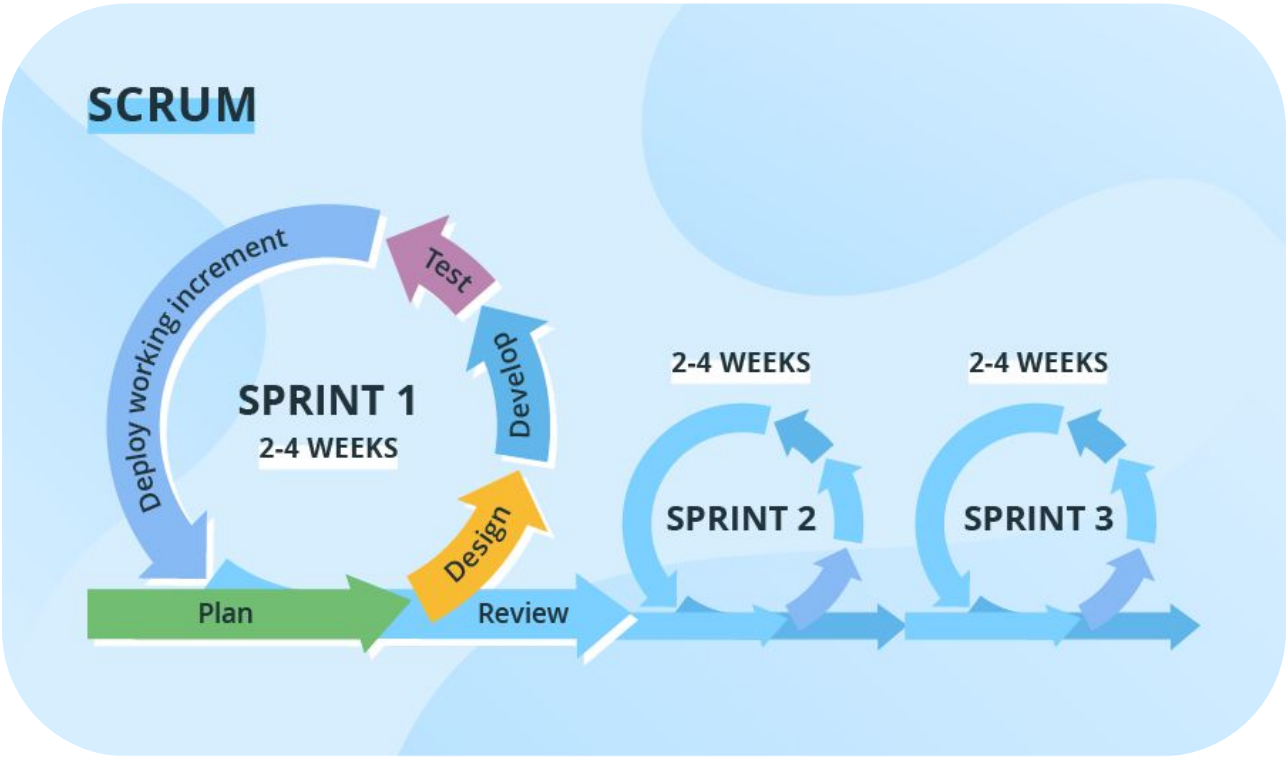
03

Large projects that are easy to divide into small functional parts and can be developed incrementally over each iteration.

Scrum

Scrum is probably the most popular Agile model. The iterations (‘sprints’) are usually 2-4 weeks long and they are preceded with thorough planning and previous sprint assessment. No changes are allowed after the sprint activities have been defined.

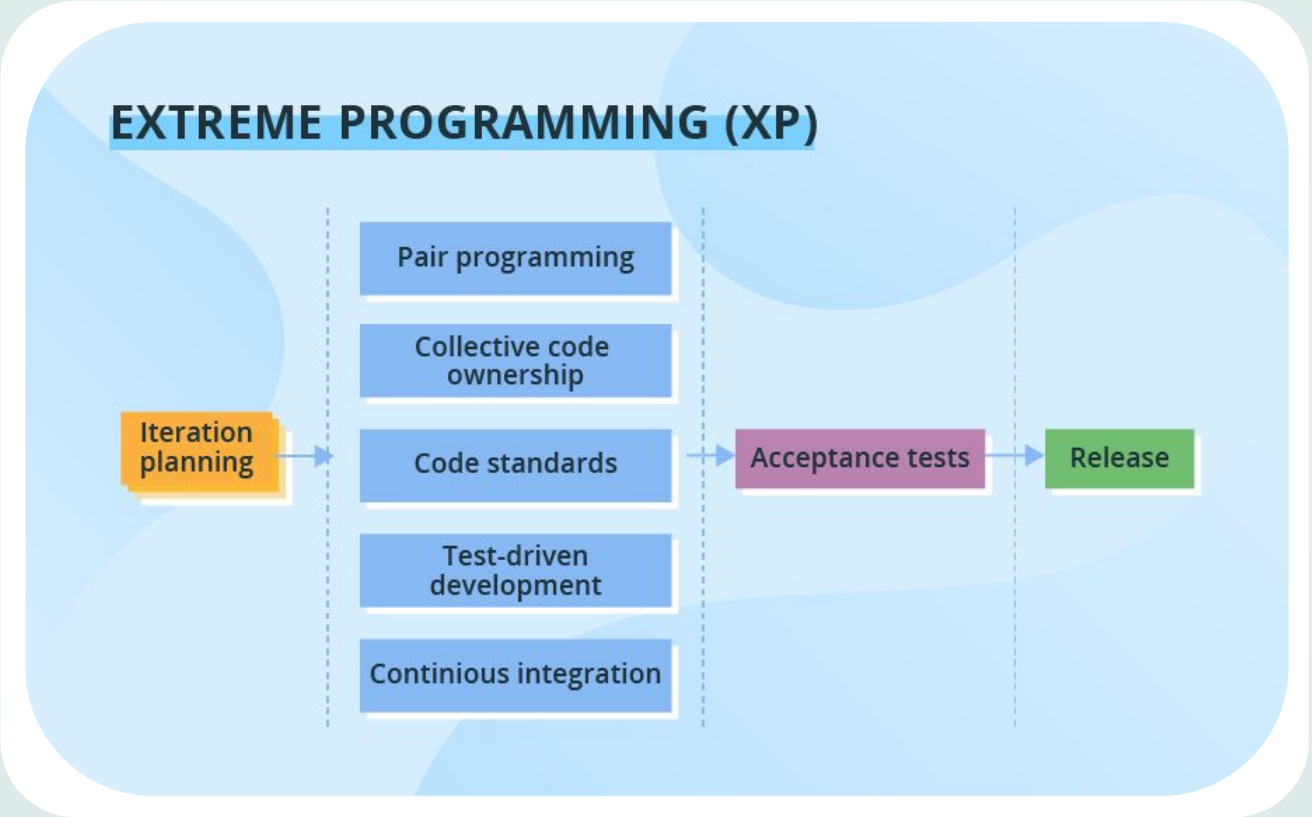
Scrum


SCRUM
Deploy working increment
Test
Develop
SPRINT 1
2-4 WEEKS
Design
Plan
Review
2-4 WEEKS
SPRINT 2
2-4 WEEKS
SPRINT 3

Extreme Programming (XP)

With ***Extreme Programming (XP),*** a typical iteration lasts 1-2 weeks. The model allows changes to be introduced even after the iteration launch if the team hasn't started to work with the relevant software piece yet. Such flexibility significantly complicates the delivery of quality software. To mitigate the problem, XP requires the use of pair programming, test-driven development and test automation, continuous integration (CI), small releases, simple software design and prescribes to follow the coding standards.

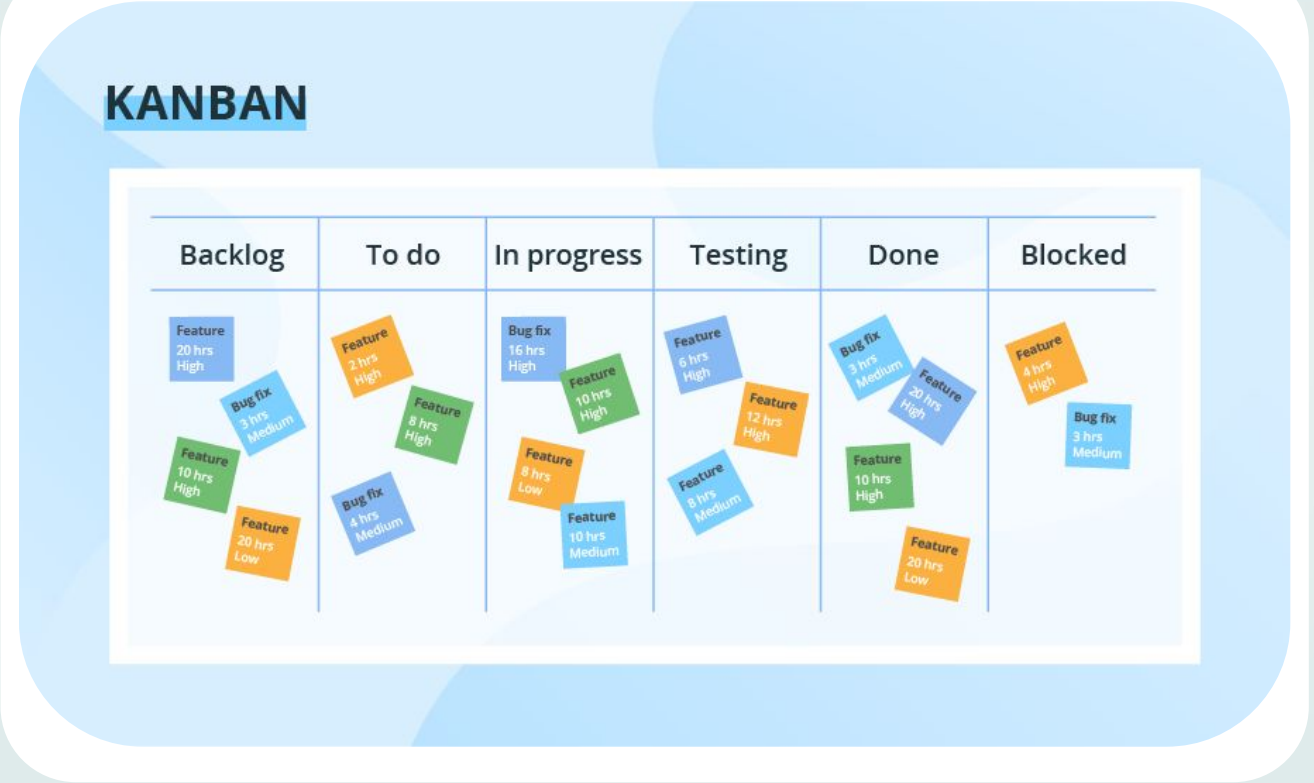
Extreme Programming (XP)

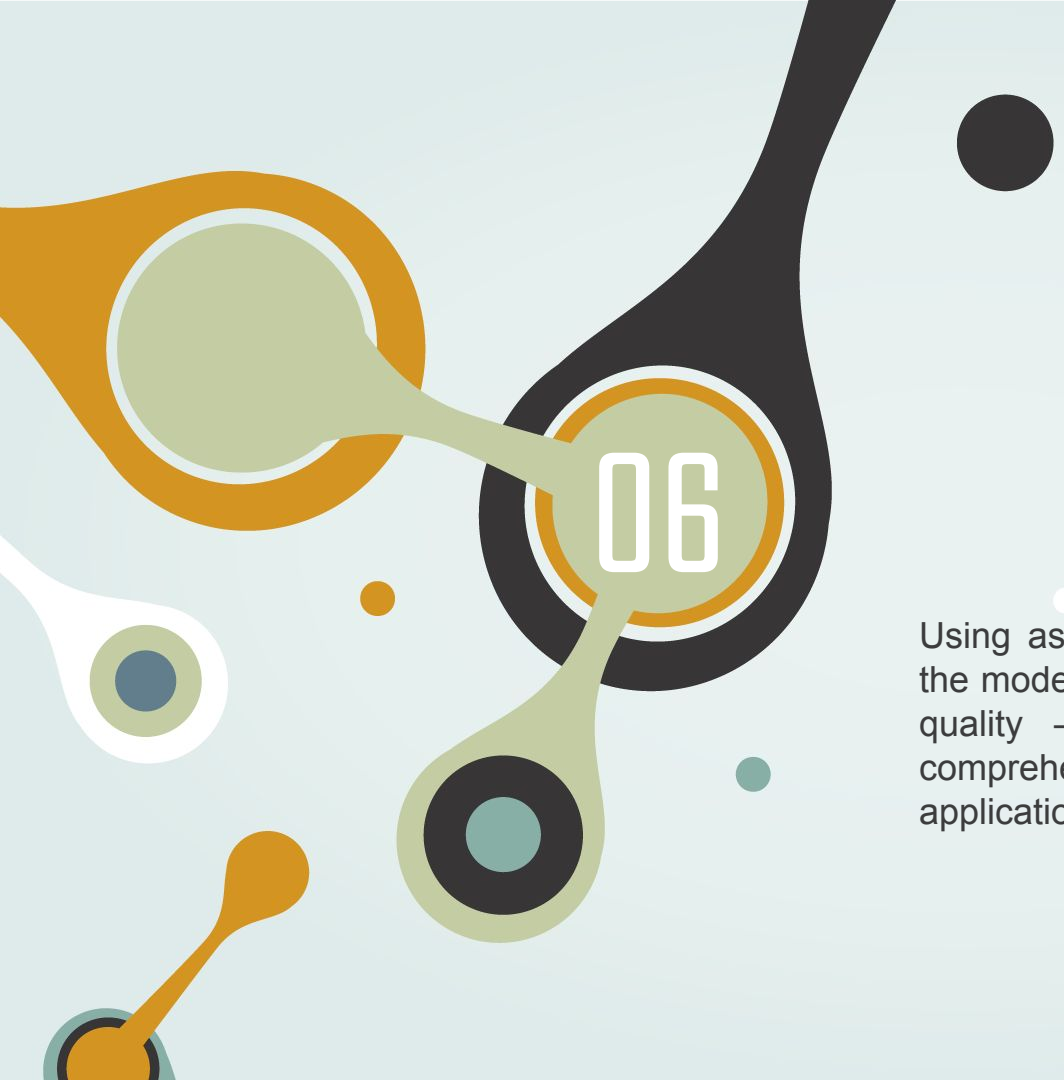


Kanban

As for **Kanban,** its key distinguishing feature is the absence of pronounced iterations. If used, they are kept extremely short (‘daily sprints’). Instead, the emphasis is placed on plan visualization. The team uses the **Kanban Board** tool that provides a clear representation of all project activities, their number, responsible persons, and progress. Such increased transparency helps to estimate the most urgent tasks more accurately. Also, the model has no separate planning stage, so a new change request can be introduced at any time. Communication with the customer is ongoing, they can check the work results whenever they like, and the meetings with the project team can happen even daily. Due to its nature, the model is frequently used in *projects on software support and evolution*.

Kanban


KANBAN
Backlog
To do
In progress
Testing
Done
Blocked
Feature 20 hrs High
Bug fix 3 hrs Medium
Feature 10 hrs High
Feature 20 hrs Low
Feature 2 hrs High
Feature 8 hrs High
Bug fix 4 hrs Medium
Bug fix 16 hrs High
Feature 10 hrs High
Feature 8 hrs Low
Feature 10 hrs Medium
Feature 6 hrs High
Feature 12 hrs High
Feature 8 hrs Medium
Bug fix 3 hrs Medium
Feature 20 hrs High
Feature 10 hrs High
Feature 20 hrs Low
Feature 4 hrs High
Bug fix 3 hrs Medium

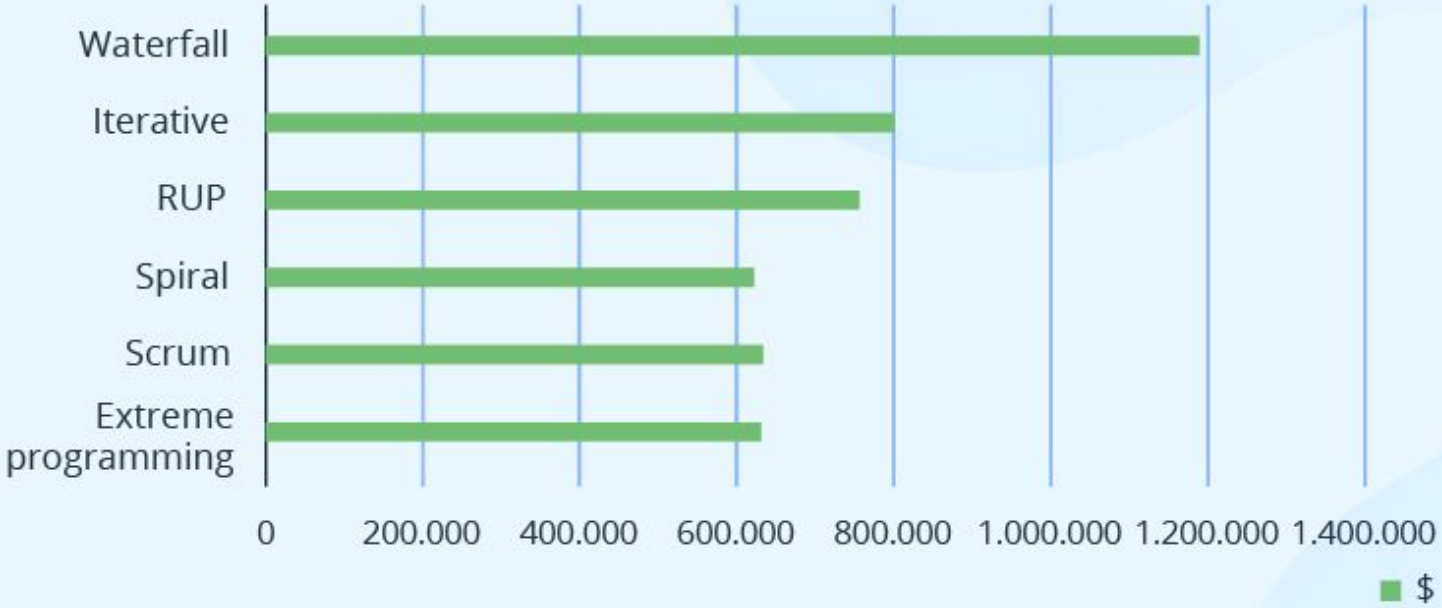


IMT SCHOOL

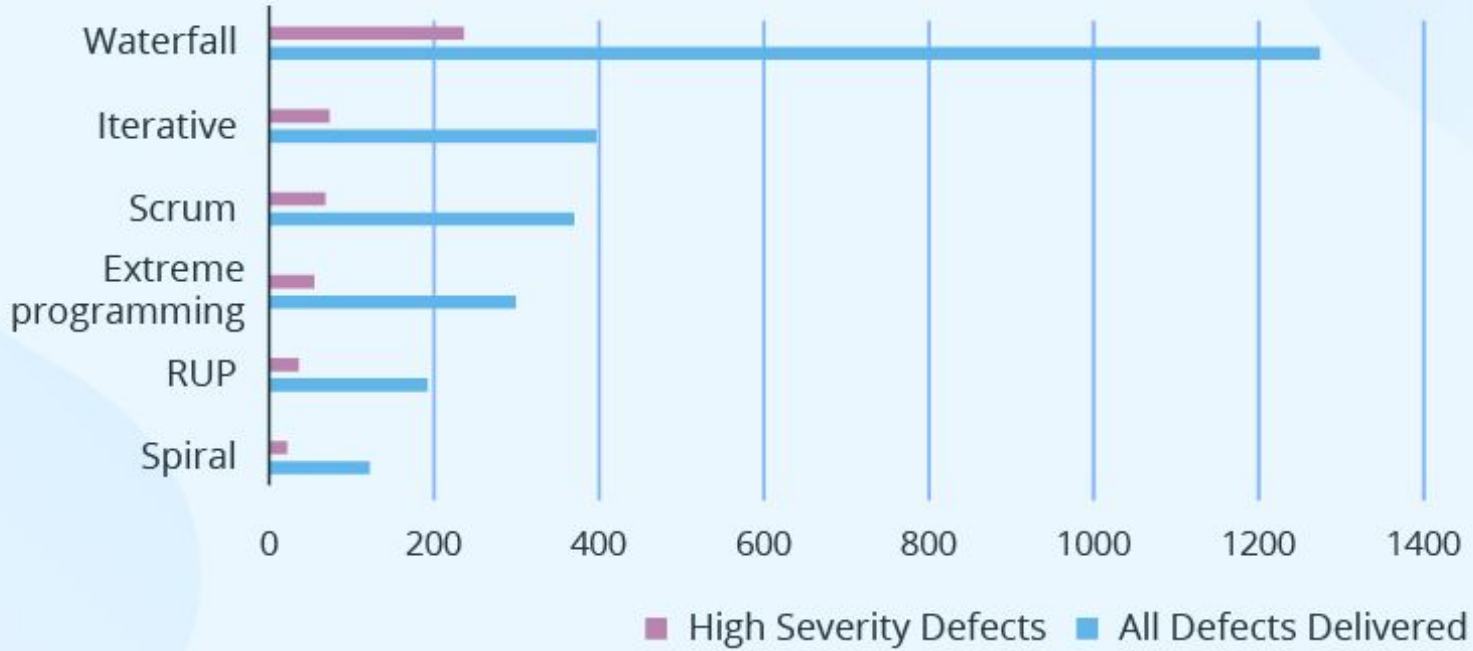
Models Charts

Using as basis the research data, we've compared the models in terms of core features – time, cost and quality – to make them easier to digest and comprehend. All estimates are relevant to small applications with code consisting of 1,000 functions.

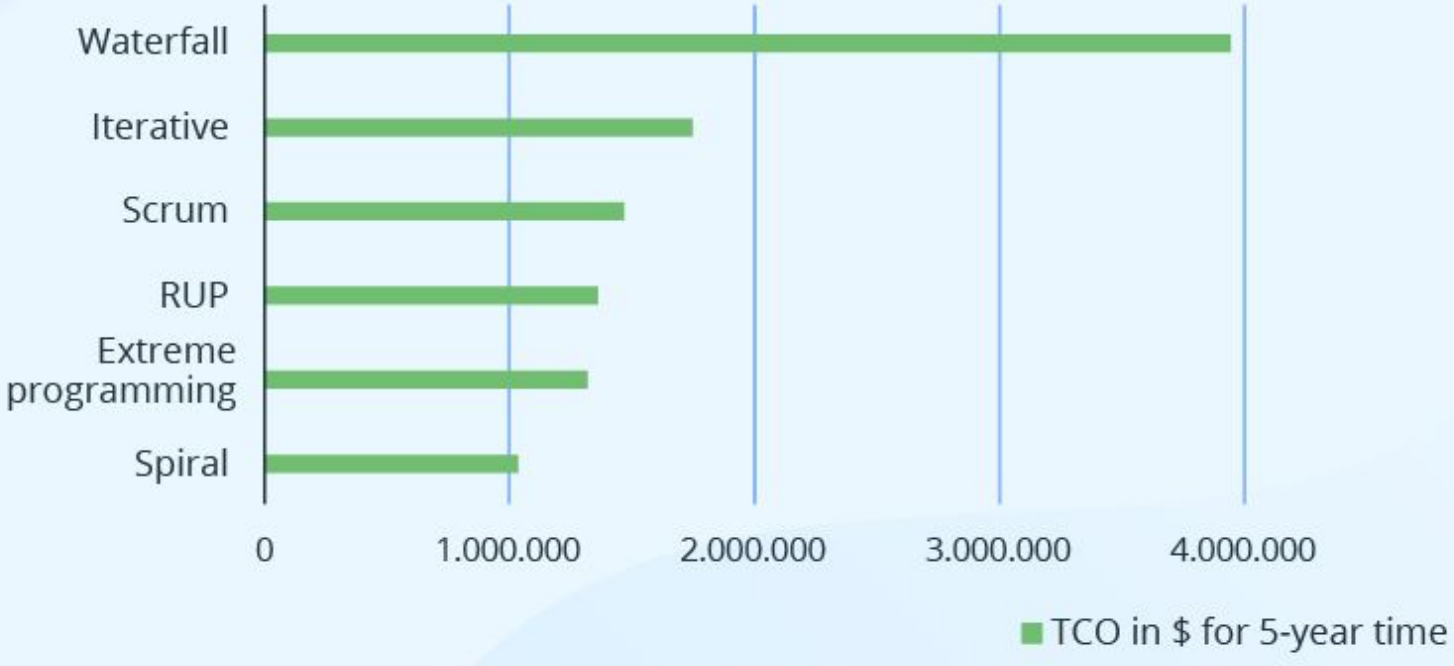
COMPARING COSTS



COMPARING QUALITY



COMPARING TCO FOR 5-YEAR TIME



The End

IMT SCHOOL

www.imtschool.com

www.facebook.com/imaketechnologyschool/