# **LEVEL 1**

## **Waterfall model**

Pros

1. **Each stage has a clear verifiable outcome** (Each step must be completed, before moving onto the next one, so half-finished projects less likely to be, giving in the end more completed and polish product)
2. **Phases are processed and completed one at a time** (almost the same as the first one)
3. **Works well for smaller projects where requirements are clearly defined and very well understood** (If tasks or projects are small, they make a clear goal from the beginning, with less potential for getting lost in the details)

Cons

1. **Very difficult to adapt project with changes in requirements** (A sudden change to the requirements of the project could create a lot of work that was already implemented – useless, and you need to redo part of your project (even if not whole), with new changes for requirement)
2. **Late creation of a working product** (late creation of product can bring an outdated version of desired product or reality could not met with expectations of user/client)
3. **High amounts of risk and uncertainty** (Once the product is developed and if any failure occurs then the cost of fixing such issues is very high, because we need to update everything from document till the logic.)
4. **Participation of software users almost absent** (it either does not provide for it at all, or it provides only indirectly at the stage of a single requirements collection. Only once the product is ready then only it can be demonstrated to the end users.

This model is used when the requirements are very well known, clear and fixed and it can be intuitively applied to relatively simple tasks and this model.

In a slightly improved form, is used on large projects in which the requirementsare very stable and can be well formulated at the beginning of a project (aerospace, medical software, etc.).

## **Incremental-iterative model**

Pros

1. **Pretty early prototyping** (Early prototyping can help with problem, if we created a right product for our customer. And spot some inconsistencies in it before adding new features)
2. **Decomposition of the project into manageable iterations** (It helps to iterate development of product in small chunks, that more clear and easily checkable than the whole project)
3. **Less cost to change scope and requirements** (Due to its nature, we can add changes before next iteration, that helps to decrease a cost for edits)
4. **Customer can respond to each built** (Communication with customer helps to define, if general direction of project implementation goes as he seen)
5. **Easier to manage risk (**because risky pieces are identified and handled during each iteration)

Cons

1. **Lack of flexibility within iterations** (Changes cannot be done, while iteration would not end. It can create a prototype, without feature, that needed for this change)
2. **Difficulty in troubleshooting, missed on early stages project development**
3. **Needs good planning and design**
4. **Needs a clear and complete definition of the whole system before it can be broken down and built incrementally**

This model can be used, when the requirements of the system are clearly defined and understood, however, some details can be changed with time. And if there is a need to get a product to the market early with some high-risk features.

## **V-model**

Pros

1. **At each stage there is a clear verifiable result** (like for a waterfall, we got full coverage of our step, without blockers)
2. **Attention is paid to testing with the very first stage** (At the earliest stages of project development, testing appears, which allows minimizing risks, and also detect and fix many potential problems before they happened)
3. **Testing activities like planning, test designing happens well before coding (**This saves a lot of time. Hence higher chance of success over the waterfall model)

Cons

1. **Lack of flexibility and adaptability** (due to its adaptation from waterfall model, it still lacks the ability of flexible changing and changing in requirement’s creates an increased cost of a project completion)
2. **Missing early prototyping** (product is developed during the implementation phase, so no early prototypes of the are produced)
3. **Difficulty in troubleshooting, missed on early stages project development**

The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed. High confidence of customer is required for choosing the V-Shaped model approach, since no prototypes are produced, there is a very high risk involved in meeting customer expectations.

## **Agile model**

Pros

1. **Maximum customer involvement** (More precise end-product will be, due to involvement of party of interest)
2. **Tight integration testing and development** (possible and already spotted problems can be fixed with a help of tight integration of testing and development, that can improve speed and quality of product creation for customer)
3. **People and interactions are emphasized rather than process and tools** (Customers, developers, and testers constantly interact with each other)
4. **Working software is delivered frequently** (Early working builds could help define if we created a right product for our customer. And with this, constantly increase a value of product by adding and fixing new features)
5. **Close, daily cooperation between business and developers** (Daily operation between business and developers help to have one's finger on the pulse with ongoing developing of the product and the expectation and remarks from business)
6. **Regular adaptation to changing circumstances** (due to fast paced life and economic situations, sometimes late changes must be done to keep a product of the customer more competitive)

Cons

1. **Complexity of implementation for large projects** (a lot of changes can deviate what we plan and what we have. So, the bigger project, the less possibility to predict overhaul project result)
2. **Difficulty in building stable processes** (a lot of changes, that can be made up, means the less stable process for a team)
3. **There is lack of emphasis on necessary designing and documentation**
4. **The project can easily get taken off track if the customer representative is not clear what outcome that they want.**

Agile can be chosen when new changes should be implemented. It can be done with a very little cost because of the frequency of new builds that are produced.

Changes can be discussed and features can be added or removed based on feedback.

Having options gives the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

## **Spiral model**

Pros

1. **Suitable for large projects** (With a help of risk analysis and it`s spiral repeat, it helps to control direction of the project and minimize rash decisions)
2. **Early prototyping**
3. **High amount of risk analysis** (As for first one)
4. **Good for large and mission-critical projects** (As for first one)
5. **Strong approval and documentation control** (As for first one)

Cons

1. **High overhead expenses.**
2. **Difficult to apply for small projects.**
3. **High dependence of success on the quality of analysis risks.**
4. **Risk analysis requires highly specific expertise.**
5. **Project’s success is highly dependent on the risk analysis phase.**

Spiral model can be used for long-term project costs and risk evaluation is important and commitment to it is unwise because of potential changes to economic. Also, when users are unsure of their needs and his requirements can be significantly changing

# **LEVEL 2**

**На твою думку, чому з’явився Agile маніфест?**

The manifesto was created as a solution to the raising problem with the traditional development processes, due to increased market and competition on it. With fast-paced changing society and its requests to companies and their products, a bunch of developers had tried to describe a new way for project leading

**Які виклики він мав вирішити і чи це вдалося?**

The manifesto was designed to empower developers, speed up processes, and help encourage working practices that focus more directly on the user, and not on documentation.

The lag time between business needs and solutions being developed was with major delay, and the standard processes were unwieldy, unsatisfactory, and overburdened with documentation and oversight.

Agile allows to be adaptive, to respond quickly and to be in a state of constant change by customer feedback.

Constant communication and trust in people, allows gathering an important information in the most efficient and effective way, and use it to create a working product, that will be, first of all, user-oriented, and not just “job to be done”

And **yes**, part of these ideas and principles are using. Not them all, because there 12 principals of it. But 4 main postulates, clearly using by companies, that chosen Agile methodology for their process.

# **LEVEL 3**

**Ти – засновник стартапу і плануєш випустити на ринок мобільний застосунок для обміну фотографіями котиків. Яку методологію ти обереш для процесу розробки і чому? Відповідь текстово обгрунтуй.**

For mobile development and for this type of application, it's best to use Agile methodology. If to be clearer, use Kanban or Scrum.

By dividing our application on small block, team can develop/test/release main framework of application, with minimum acceptable features for it, early as possible. And after that, constantly add new content/features/improvements not only from our backlog list of required features, but also from users feedback, that will be constantly created from rating and comments.

Add to it constantly changing requests from people's needs, that grows on the internet culture. Especially with pet’s segments.

Today we share cat`s photos. Tomorrow – cat’s videos. After a week – we create dating application, that matches people based on if they like cats on person's profile 

And we can easily start to monetize the content and features from our application almost after the first prototype release if we would launch this application as an early access product.

So, for maximum value and flexibility, Agile would be right way to start application