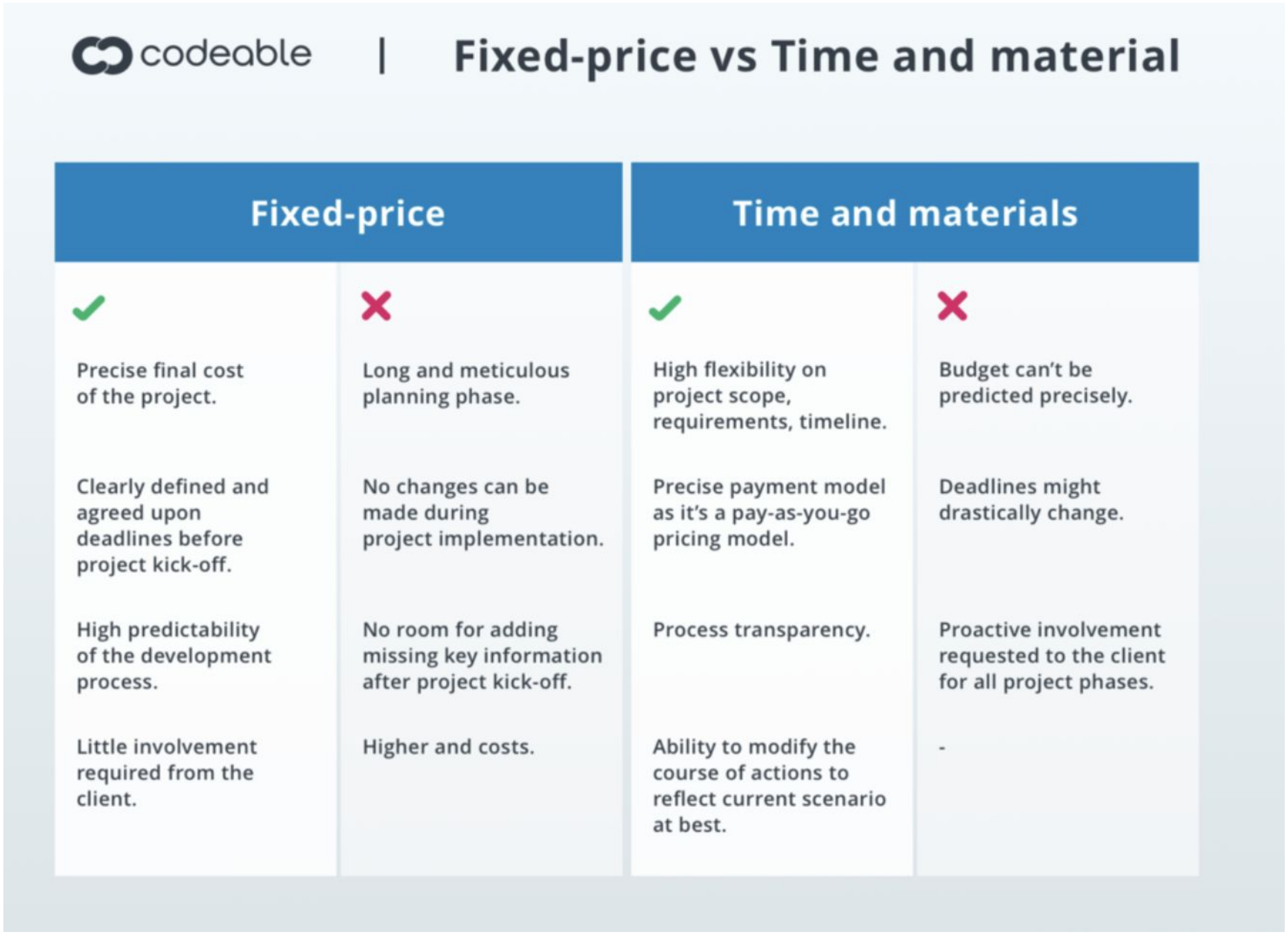
SDLC and Waterfall

Krasner Stanislau, 10 September 2020



codeable | Fixed-price vs Time and material

Fixed-price		Time and materials	
✓	✗	✓	✗
Precise final cost of the project.	Long and meticulous planning phase.	High flexibility on project scope, requirements, timeline.	Budget can't be predicted precisely.
Clearly defined and agreed upon deadlines before project kick-off.	No changes can be made during project implementation.	Precise payment model as it's a pay-as-you-go pricing model.	Deadlines might drastically change.
High predictability of the development process.	No room for adding missing key information after project kick-off.	Process transparency.	Proactive involvement requested to the client for all project phases.
Little involvement required from the client.	Higher and costs.	Ability to modify the course of actions to reflect current scenario at best.	-

SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)

is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase.

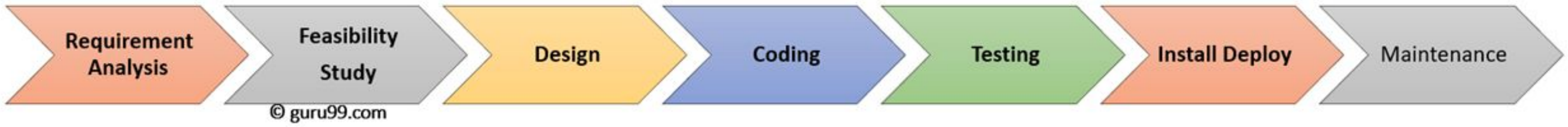
Why SLDS

Here, are prime reasons why SDLC is important for developing a software system.

- It offers a basis for project planning, scheduling, and estimating
- Provides a framework for a standard set of activities and deliverables
- It is a mechanism for project tracking and control
- Increases visibility of project planning to all involved stakeholders of the development process
- Increased and enhance development speed
- Improved client relations

SDLS Phases

Фазы



SDLS Phases

Фазы

Phase 1: Requirement collection and analysis:
The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.
This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.
Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

Phase 2: Feasibility study:
Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.
There are mainly five types of feasibilities checks:

- Economic: Can we complete the project within the budget or not?
- Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
- Operation feasibility: Can we create operations which is expected by the client?
- Technical: Need to check whether the current computer system can support the software
- Schedule: Decide that the project can be completed within the given schedule or not

Phase 3: Design:
In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.
This design phase serves as input for the next phase of the model.
There are two kinds of design documents developed in this phase:
High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Low-Level Design(LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues

Phase 4: Coding:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

Phase 5: Testing:
Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.
During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

Phase 6: Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

Phase 7: Maintenance:
Once the system is deployed, and customers start using the developed system, following 3 activities occur

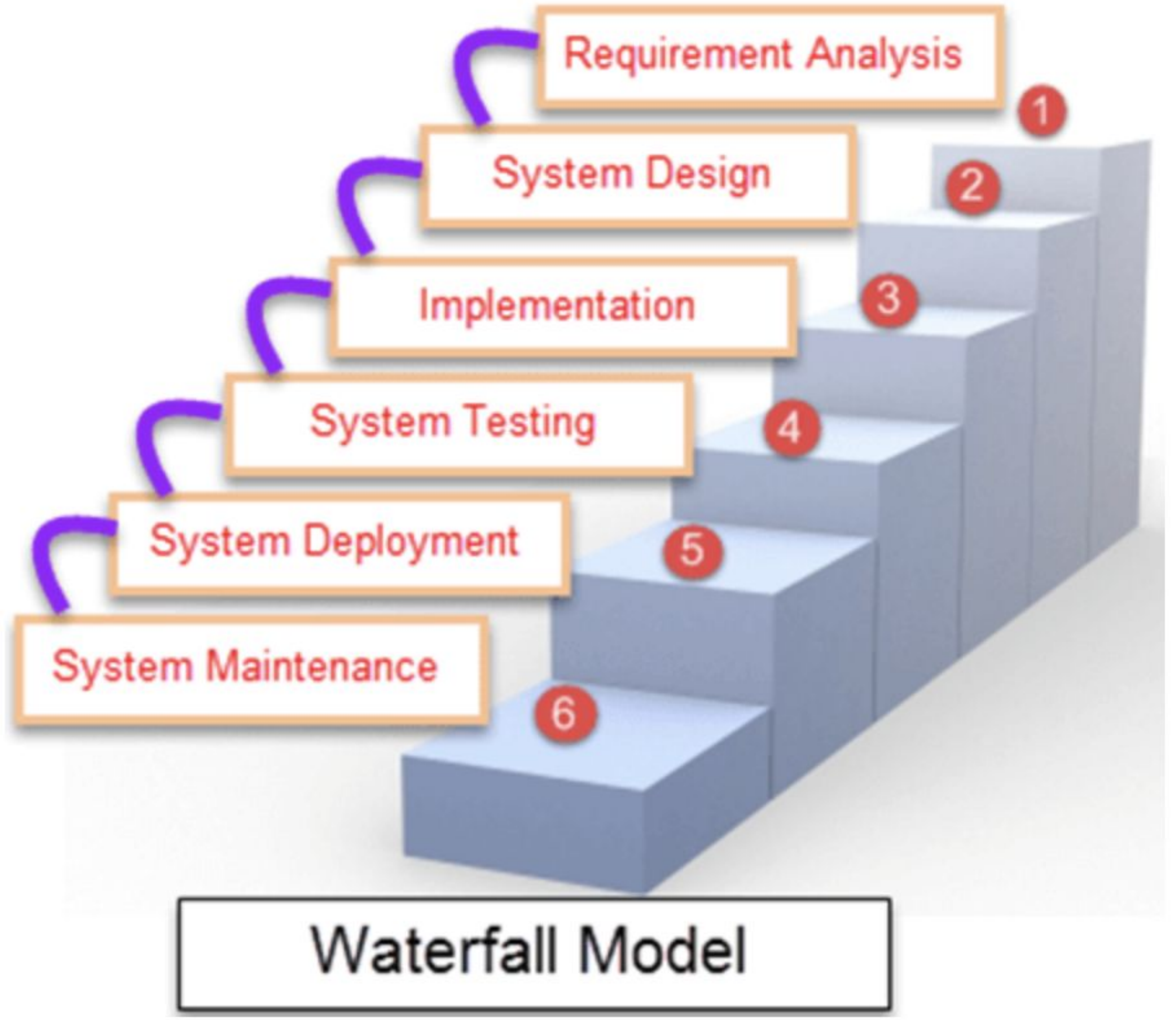
- Bug fixing - bugs are reported because of some scenarios which are not tested at all
- Upgrade - Upgrading the application to the newer versions of the Software
- Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

What is The Waterfall Model?

What is The Waterfall Model?
WATERFALL MODEL is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase. It was introduced in 1970 by Winston Royce.

Waterfall Model



Different Phases of Waterfall Model in Software Engineering

Different phases	Activities performed in each stage
Requirement Gathering stage	● During this phase, detailed requirements of the software system to be developed are gathered from client
Design Stage	● Plan the programming language, for Example Java, PHP, .net ● or database like Oracle, MySQL, etc. ● Or other high-level technical details of the project
Built Stage	● After design stage, it is built stage, that is nothing but coding the software
Test Stage	● In this phase, you test the software to verify that it is built as per the specifications given by the client.
Deployment stage	● Deploy the application in the respective environment
Maintenance stage	● Once your system is ready to use, you may later require change the code as per customer request

When to use SDLC Waterfall Model
Waterfall model can be used when

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantages and Disadvantages of Waterfall-Model

Advantages	Dis-Advantages
● Before the next phase of development, each phase must be completed	● Error can be fixed only during the phase
● Suited for smaller projects where requirements are well defined	● It is not desirable for complex project where requirement changes frequently
● They should perform quality assurance test (Verification and Validation) before completing each stage	● Testing period comes quite late in the developmental process
● Elaborate documentation is done at every phase of the software's development cycle	● Documentation occupies a lot of time of developers and testers
● Project is completely dependent on project team with minimum client intervention	● Clients valuable feedback cannot be included with ongoing development phase
● Any changes in software is made during the process of the development	● Small changes or errors that arise in the completed software may cause a lot of problems

Verification in Software Testing
Verification in Software Testing is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

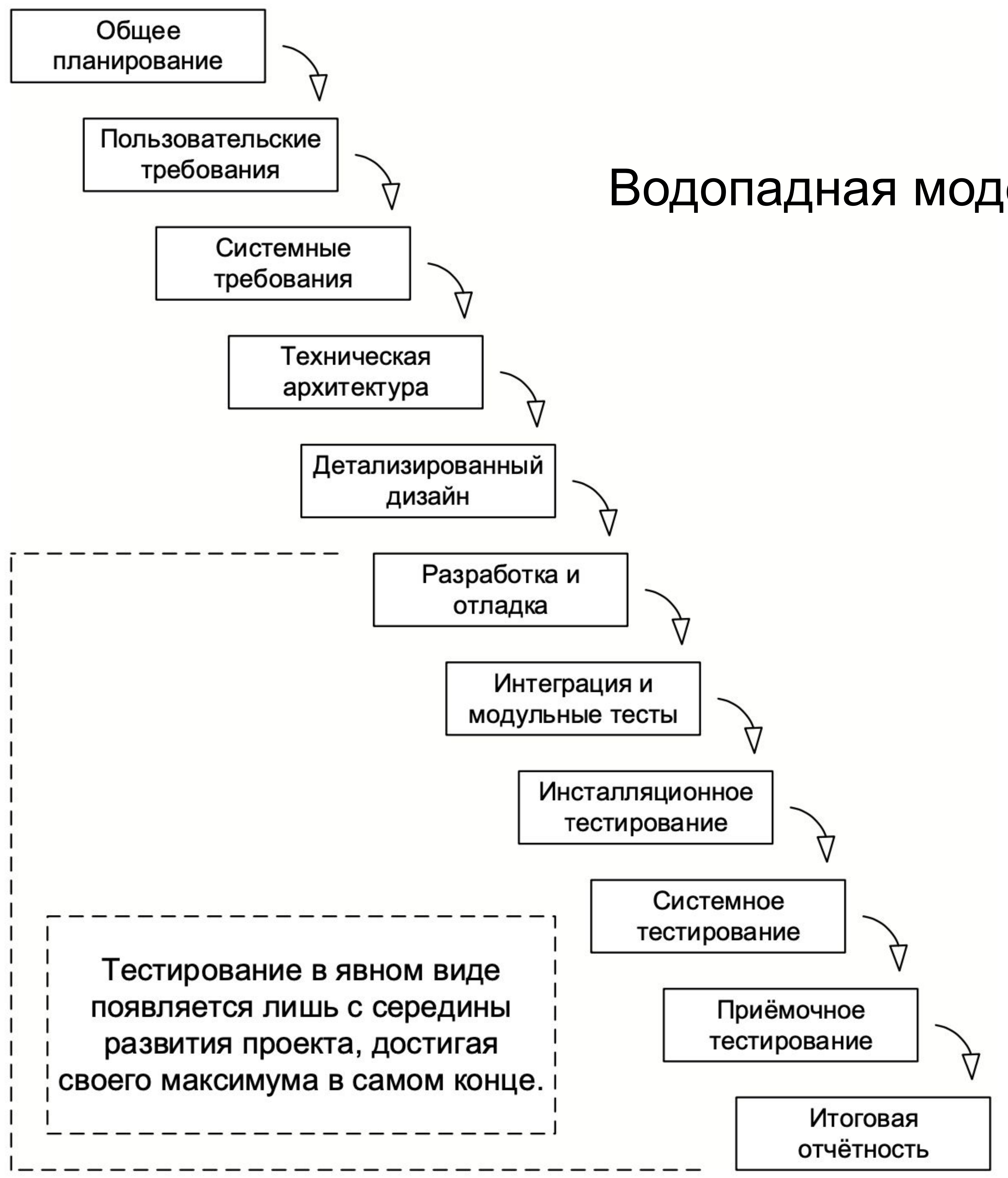
Validation in Software Testing
Validation in Software Testing is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

KEY DIFFERENCE

- Verification process includes checking of documents, design, code and program whereas Validation process includes testing and validation of the actual product.
- Verification does not involve code execution while Validation involves code execution.
- Verification uses methods like reviews, walkthroughs, inspections and desk-checking whereas Validation uses methods like black box testing, white box testing and non-functional testing.
- Verification checks whether the software confirms a specification whereas Validation checks whether the software meets the requirements and expectations.
- Verification finds the bugs early in the development cycle whereas Validation finds the bugs that verification can not catch.
- Verification process targets on software architecture, design, database, etc. while Validation process targets the actual software product.
- Verification is done by the QA team while Validation is done by the involvement of testing team with QA team.
- Verification process comes before validation whereas Validation process comes after verification.

fall

Водопадная модель разработки ПО


Общее планирование
Пользовательские требования
Системные требования
Техническая архитектура
Детализированный дизайн
Разработка и отладка
Интеграция и модульные тесты
Инсталляционное тестирование
Системное тестирование
Приёмочное тестирование
Итоговая отчётность
Тестирование в явном виде появляется лишь с середины развития проекта, достигая своего максимума в самом конце.

Плюсы и минусы

В модели Waterfall легко управлять проектом.
Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены.
Но это палка о двух концах.
Каскадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями и способами их реализации.
Нет возможности сделать шаг назад, тестирование начинается только после того, как разработка завершена или почти завершена.
Продукты, разработанные по данной модели без обоснованного ее выбора, могут иметь недочеты (список требований нельзя скорректировать в любой момент), о которых становится известно лишь в конце из-за строгой последовательности действий.
Стоимость внесения изменений высока, так как для ее инициализации приходится ждать завершения всего проекта. Тем не менее, фиксированная стоимость часто перевешивает минусы подхода.
Исправление осознанных в процессе создания недостатков возможно, и, по нашему опыту, требует от одного до трех дополнительных соглашений к контракту с небольшим ТЗ.

С помощью каскадной модели мы создали множество проектов «с нуля», включая разработку только ТЗ.