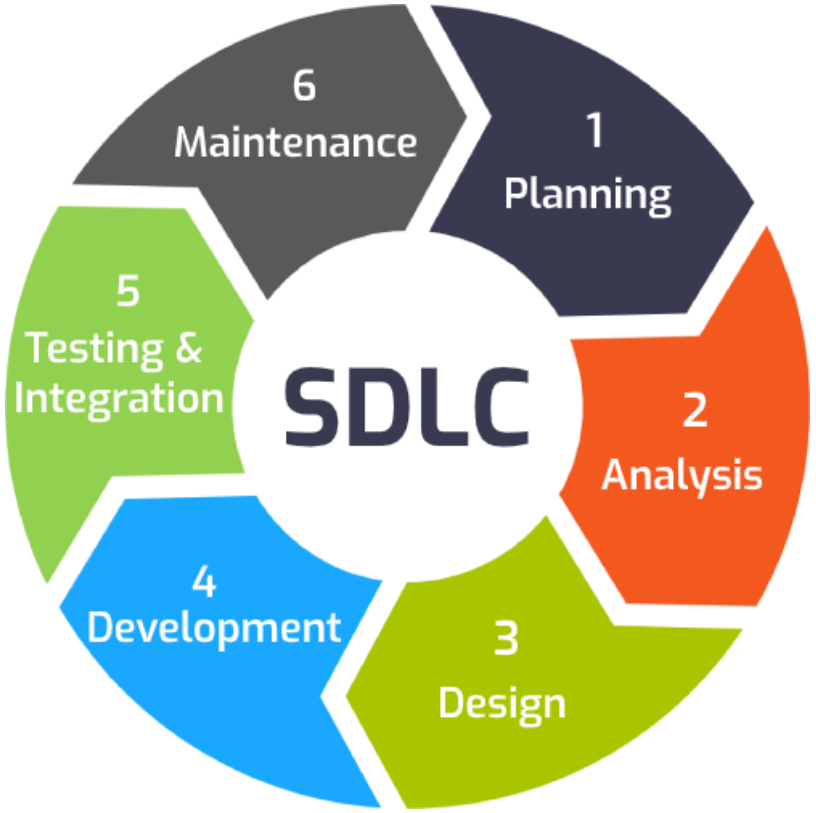
UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT LIFE CYCLE

SOFTWARE DEVELOPMENT LIFE CYCLE



Objective: Provide a predictable framework of procedures designed to identify all requirements about functionality, cost, reliability, and delivery schedule and ensure that each are met in the final solution.

SOFTWARE DEVELOPMENT LIFE CYCLE

Planning

Determine whether information involved in the project requires protection,
Any information that is handled by the application needs a value assigned by its owner

- Define the scope
- Determine solutions
- Cost
- Time

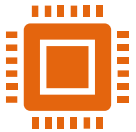
SOFTWARE DEVELOPMENT LIFE CYCLE

Analysis

Requirements could be derived from a variety of sources such as evaluating competitor products for a commercial product to surveying the needs of users for an internal solution. In some cases these requirements could come from a direct request from a current customer.

- Information is collected from the customer
- Software Requirement Specification document (SRS)
- Purpose of the product.
- Meetings

SOFTWARE DEVELOPMENT LIFE CYCLE



Design

In this process question like how the software will satisfy all functional and security goals.

- internal behavior and operations
- Software to specific requirements
- Hardware

SOFTWARE DEVELOPMENT LIFE CYCLE

Development

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase. The Develop phase involves writing the code or instructions that make the software work.

- Research and Study
- Tickets
- Start of production / Coding

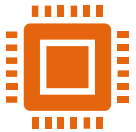
SOFTWARE DEVELOPMENT LIFE CYCLE

Testing & Integration/Deployment

In the Test/Validate phase, several types of testing should occur, including ways to identify both functional errors and security issues. Once the product is tested, it is deployed in the production environment

- Software is tested thoroughly and any defects found are assigned to developers to get them fixed.kets
- Retesting, regression testing is done until the point at which the software is as per the customer's expectation.

SOFTWARE DEVELOPMENT LIFE CYCLE



Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

VIDEOS

REFERENCES

- https://www.innovativearchitects.com/KnowledgeCenter/basic-IT-systems/system-development-life-cycle.aspx
- https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/
- Chapter 8. Software Development Security
- Chapter 10. The Software Life Cycle

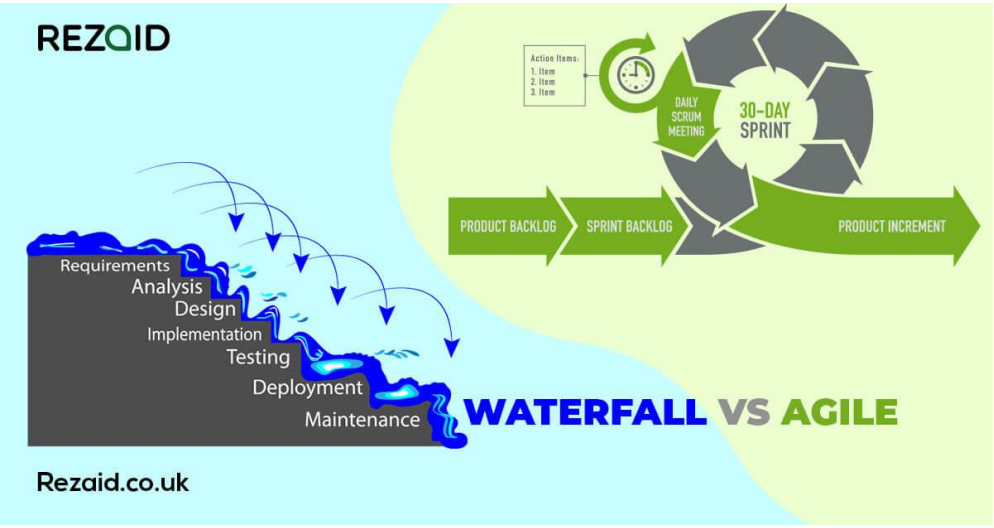
SOFTWARE DEVELOPMENT MODELS PART I

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS PART I

In every project, you need requirements, design, testing, deployment, and maintenance.

Exactly how you handle those tasks can vary depending on the scope of the project.



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS PART I

Agile methods allow a project's goals to change over time to track changing customer needs

Test-driven development forces programmers to write tests for their code.

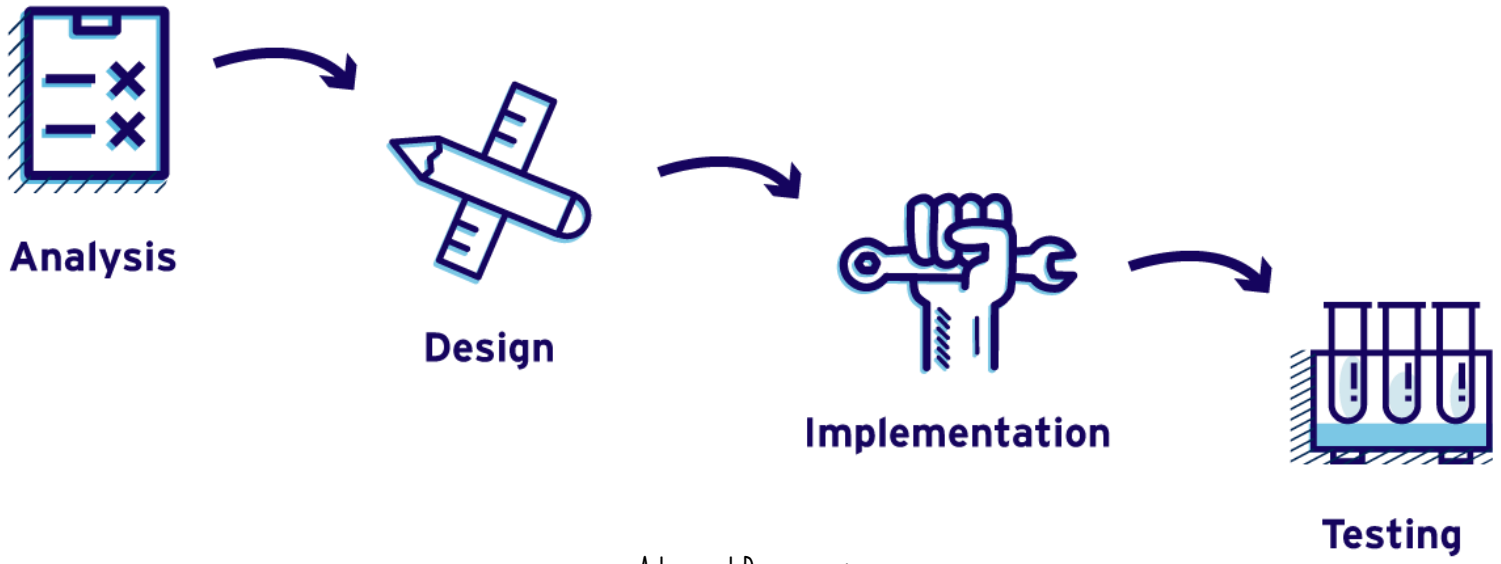
Each model has its own philosophy, set of rules, and lists of important principles.

.

SOFTWARE DEVELOPMENT MODELS PART I

Objetive: "what actually matters is whether you produce high-quality software reasonably close to on time and within your budget. A lot of development models use clever techniques."

.

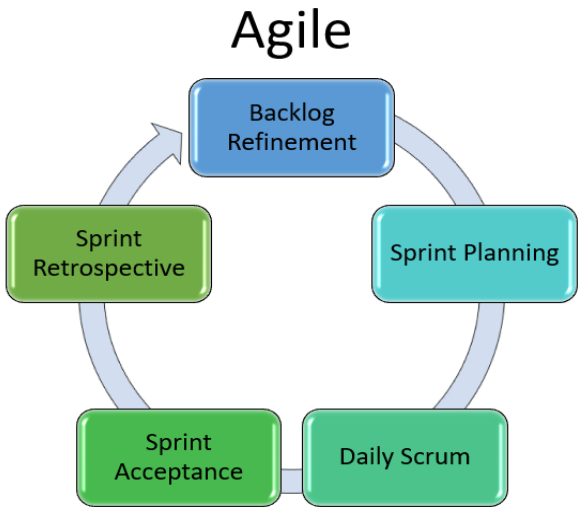


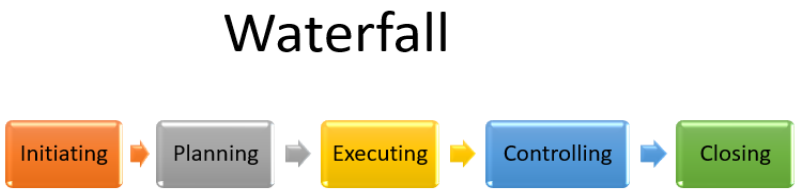
SOFTWARE DEVELOPMENT MODELS PART I

KIND OF DEVELOPMENT MODELS

1. Predictive.
2. Adaptive.

You use the requirements to design the system, and you use the design as a blueprint to write the code.





SOFTWARE DEVELOPMENT MODELS

Predictive Model

- You test the code, have customers sign off saying it really does what the specification says it should, and then pop the champagne.
- Experience
- Predict

SOFTWARE DEVELOPMENT MODELS

Adactive Model

- Adaptive development model enables you to change the project's goals if necessary, during development.
- An adaptive model lets you periodically reevaluate and decide whether you need to change direction.
- The adaptive model just gives you chances to fine-tune the project if necessary.

SOFTWARE DEVELOPMENT MODELS

Adactive Model analogy example

TV detective show

1. Your goal is to find the killer.
2. interview witnesses, check cell phone records
3. you don't know exactly where the case will lead.
4. You follow the first clue, and it leads to a second...
5. Each time you find a new clue, you update the direction of the investigation

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Predictive Model Success Indicators

- **User involvement**
- Clear vision: If the customers and developers have the same clear vision about the project's goals, development will stay on track.
- Limited size: A small size helps the customers and team members see the whole picture all at once. Requirements won't have time to change.

SOFTWARE DEVELOPMENT MODELS

Predictive Model Success Indicators

- Experienced team
- Realistic
- Established technology

SOFTWARE DEVELOPMENT MODELS

Advantages and Disadvantages to predictive models.

- Better documentation
- Stability
- Fix bugs early

- Inflexible
- Later initial release
- Big Design Up Front: You need to define everything up front. You can't start development until you know everything you're going to need to do.

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

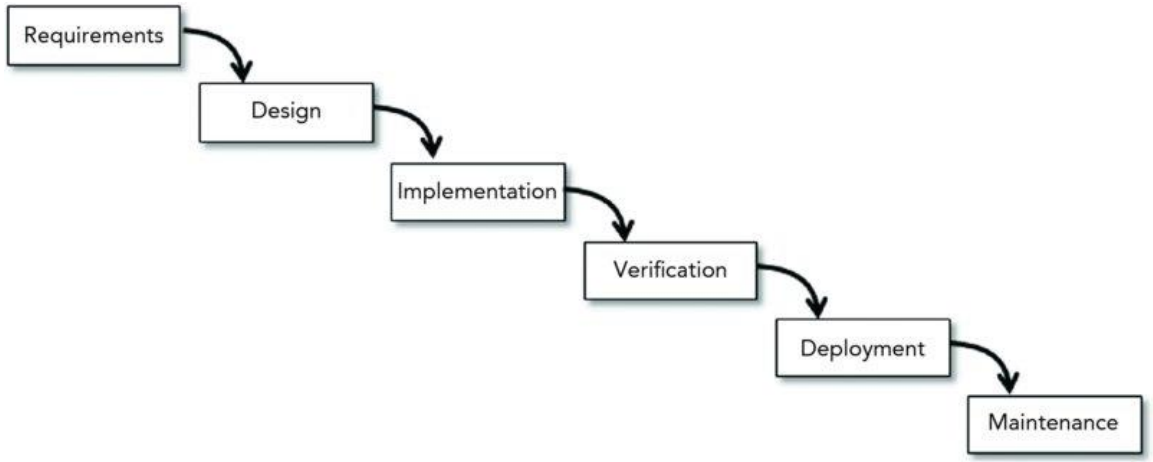
Consideration in order to use a predictive models.

- The most "pure" predictive models assume that each stage of development is finished completely and correctly before the next stage begins.
- One of the biggest ideas of BDUF projects is that the time spent on design up-front saves you time later in the project.

SOFTWARE DEVELOPMENT MODELS

WATERFALL MODEL.

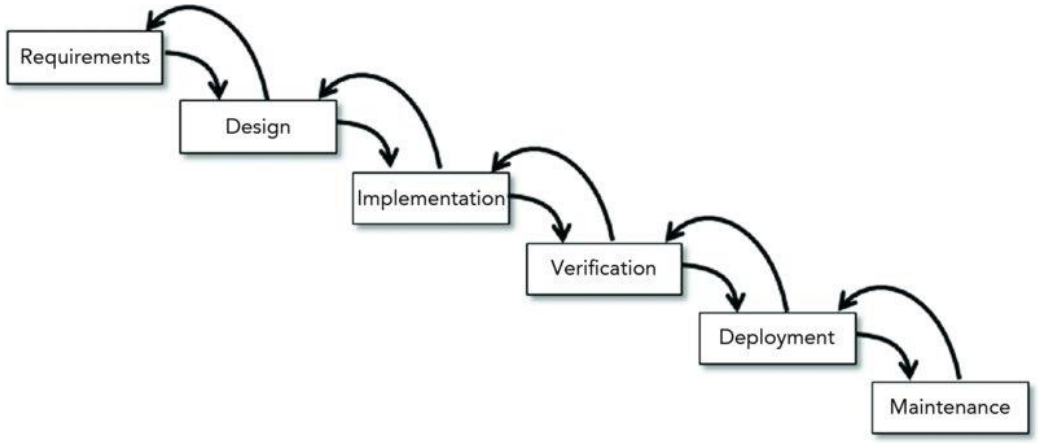
- Waterfall is the plain vanilla of the predictive model world. It assumes that you finish each step completely and thoroughly before you move on to the next step.



SOFTWARE DEVELOPMENT MODELS

WATERFALL WITH FEEDBACK.

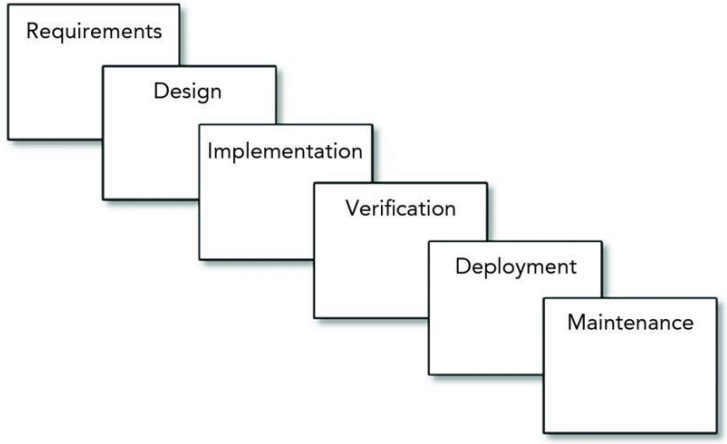
- The waterfall with feedback variation enables you to move backward from one step to the previous step..



SOFTWARE DEVELOPMENT MODELS

SASHIMI MODEL

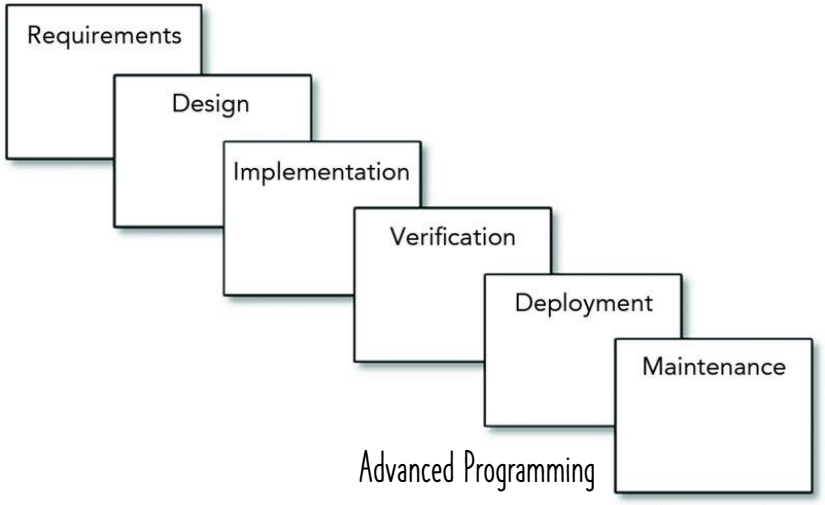
- Sashimi (also called sashimi waterfall and waterfall with overlapping phases) is similar to the waterfall model except the steps are allowed to overlap.



SOFTWARE DEVELOPMENT MODELS

SASHIMI MODEL

- In a project's first phase, some requirements will be defined while you're still working on others.
- You could have people working on requirements, design, implementation, and testing all at the same time.



SOFTWARE DEVELOPMENT MODELS

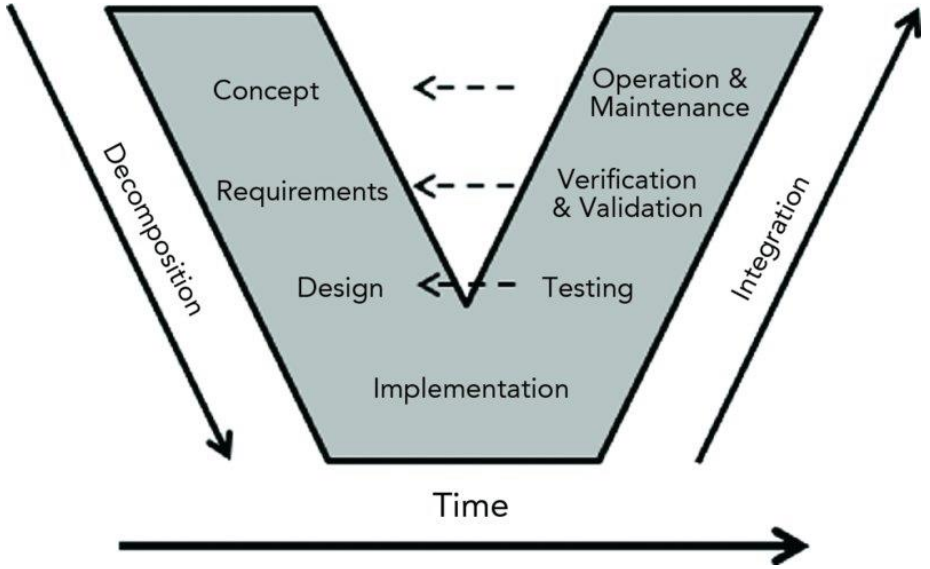
SASHIMI Advantages

- People with different skills can focus on their specialties without waiting for others.
- It lets you perform a spike or deep dive into a particular topic to learn more about it.
- You can refine the requirements.
- If you discover during design that the requirements are impossible or need alterations, you can make the necessary changes.

SOFTWARE DEVELOPMENT MODELS

V-MODEL

- V-model is basically a waterfall that's been bent into a V shape
- Each of the tasks on the left corresponds to a task on the right with a similar level of abstraction.



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Iterative models

- Predictive software development has some big advantages. It's predictable, encourages a lot of up-front design (hence the nickname big design up front or BDUF)

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

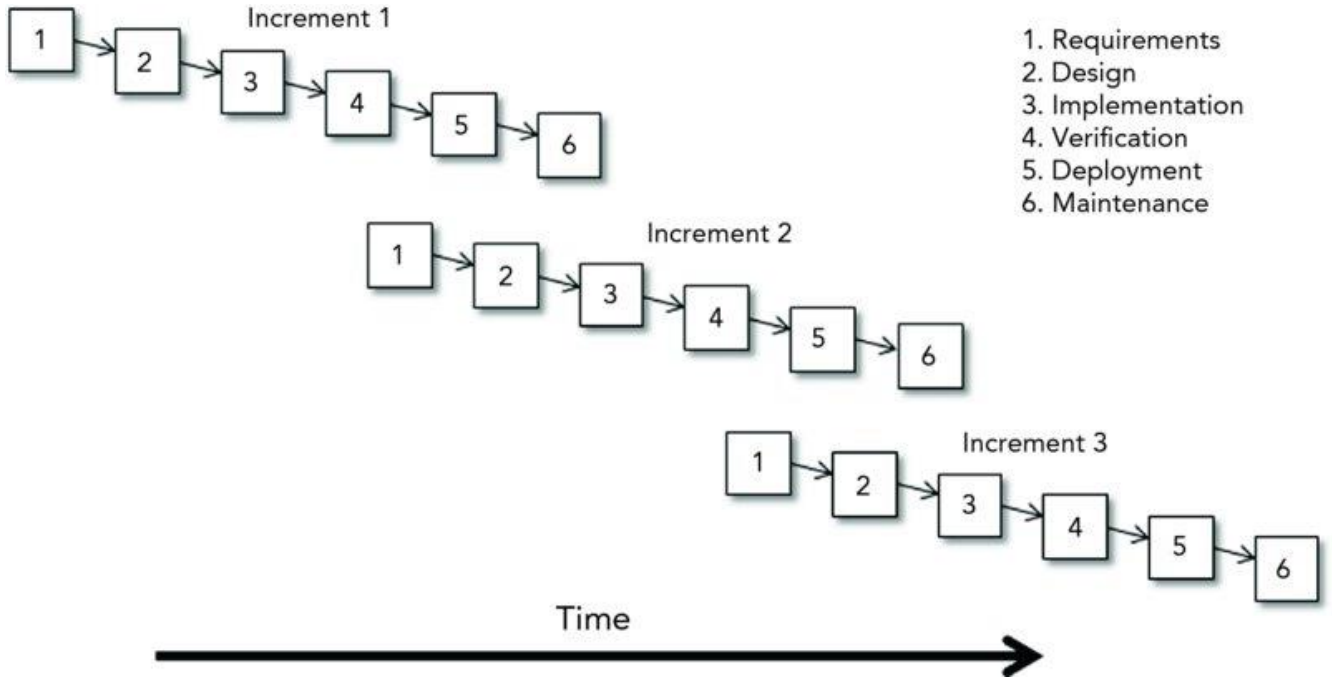
SOFTWARE DEVELOPMENT MODELS

Iterative models

- Iterative models address those problems by building the application incrementally. They start by building the smallest program that is reasonably useful. Then they use a series of increments to add more features to the program until it's finished (if it is ever finished).

SOFTWARE DEVELOPMENT MODELS

Iterative models



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Iterative models

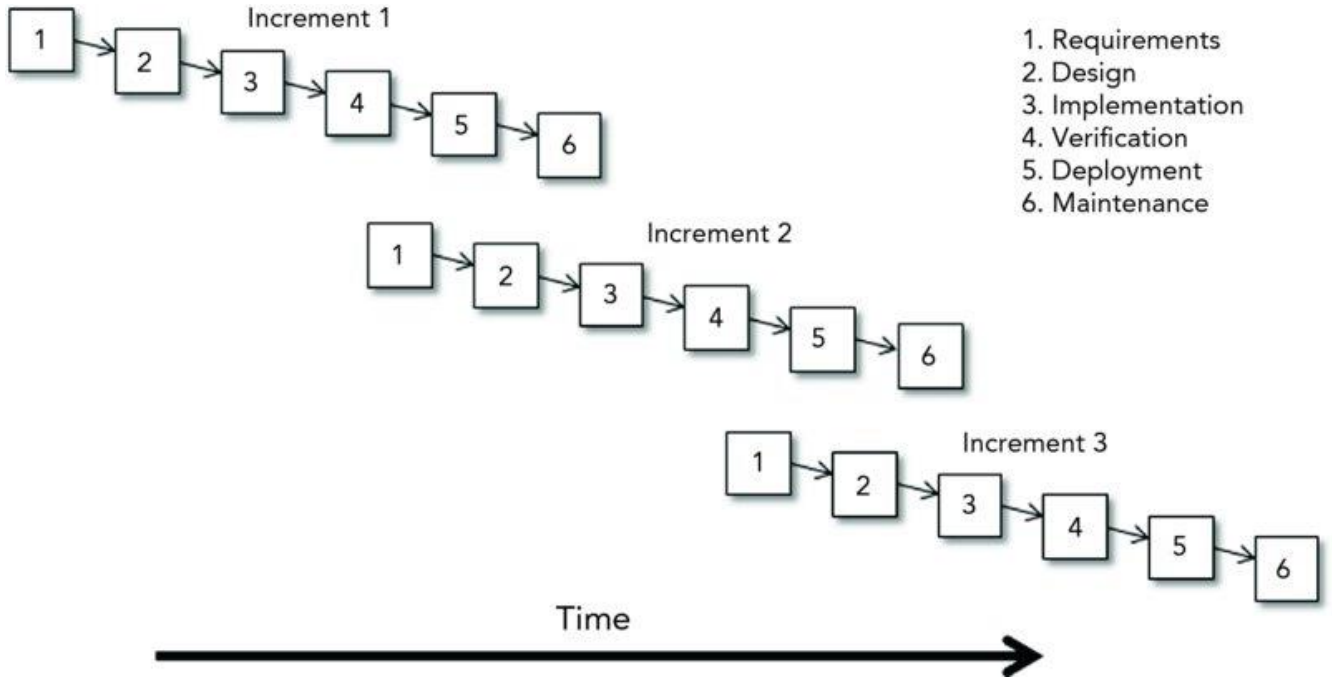
- Iterative models address those problems by building the application incrementally. They start by building the smallest program that is reasonably useful. Then they use a series of increments to add more features to the program until it's finished (if it is ever finished).

- Each increment has a relatively small duration (compared to a predictive project)

- Iterative models also handle fuzzy requirements reasonably well. If you're unsure of some of the application's requirements, you can start building the parts you do understand and figure out the rest later.

SOFTWARE DEVELOPMENT MODELS

Iterative models



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Iterative models (Prototypes)

- Throwaway prototype: You use the prototype to study some aspect of the system and then you throw it away and write code from scratch.
- Evolutionary prototype: the prototype demonstrates some of the application's features. As the project progresses, you refine those features and add new ones until the prototype morphs into the finished application.
- Incremental prototyping: You build a collection of prototypes of that separately demonstrate the finished application's features and joint it. (you need to use good programming techniques for all the prototypes.)

SOFTWARE DEVELOPMENT MODELS

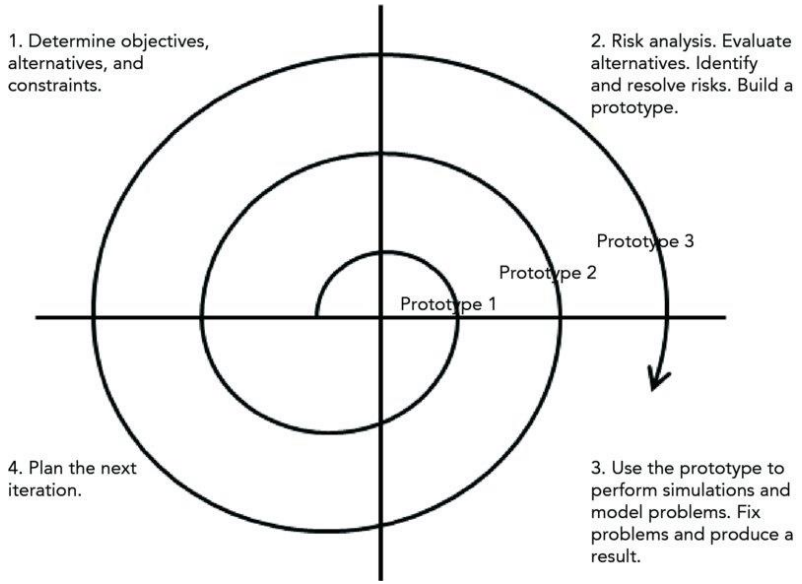
Iterative models (Prototypes)

- Improved requirements: allow customers to see the app and them feedback.
- Common vision: Custumers and Dev.
- Better design

SOFTWARE DEVELOPMENT MODELS

Iterative models (SPIRAL)

- If you don't understand all the requirements, then you might use an iterative approach for developing them.

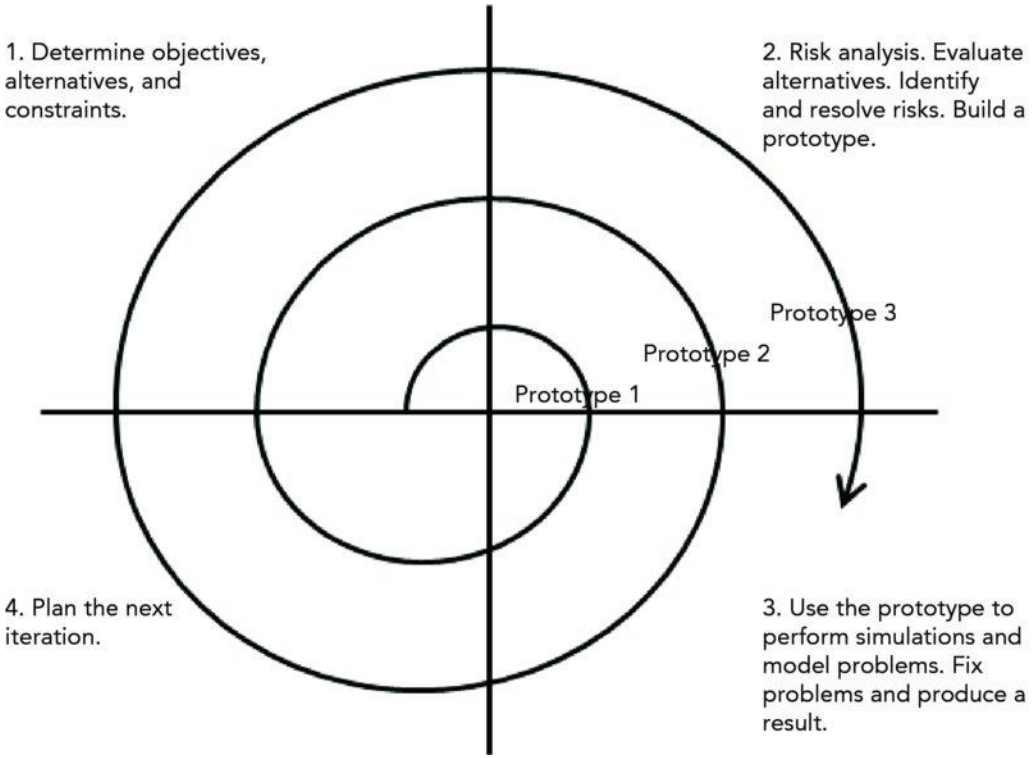


SOFTWARE DEVELOPMENT MODELS

Iterative models (SPIRAL)

Phases in Spiral Model

1-. Planning Phase
2-. Risk analysis phase.
3-. Engineering phase.
4-. Evaluation phase.



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile

Agile is a set of techniques followed by a team to administer a project or plan by dividing it into various stages with continuous collaboration with customers. There is constant monitoring at every phase of the software development of the project. The agile methodology advantages are that both the development plus testing actions are parallel and synchronized, unlike the conventional waterfall methodology.

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile Manifesto

4 Values

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile Manifesto

12 Principles

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers.
5. Projects are built around motivated individuals, who should be trusted.
6. Face-to-face conversation is the best form of communication (co-location)

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile Manifesto

12 Principles

7. Working software is the primary measure of progress.
8. Sustainable development, able to maintain a constant pace.
9. Continuous attention to technical excellence and good design
10. Simplicity–the art of maximizing the amount of work not done–is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Successful use of Scrum depends on people becoming more proficient in living five values:

- Commitment
- Focus
- Openness
- Respect
- Courage

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Successful use of Scrum depends on people becoming more proficient in living five values:

- Commitment
- Focus
- Openness
- Respect
- Courage

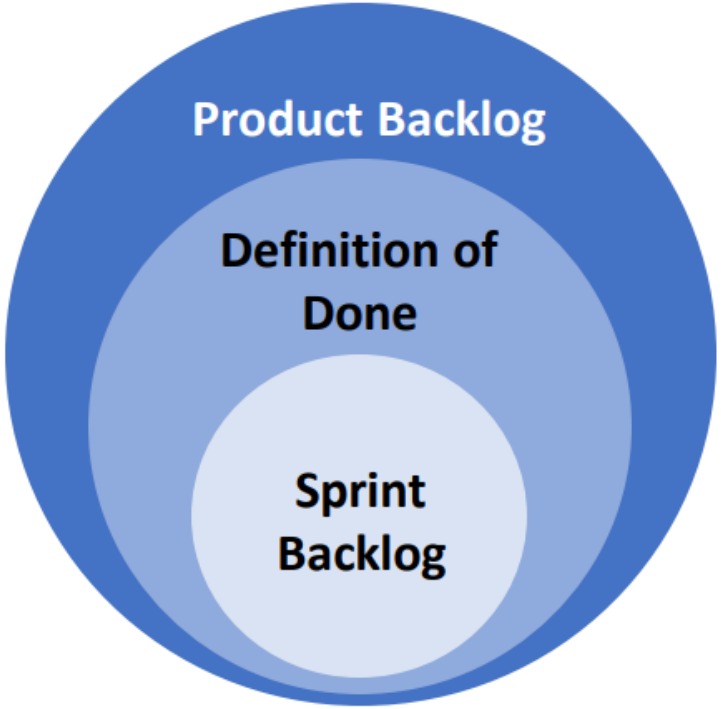
SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Roles	Artifacts	Time Boxing | Ceremonies
Product Owner	Product Backlog	Sprint Planning
Scrum Master		Daily Scrum
Team	Sprint Backlog	Sprint
		Sprint Review
		Sprint Retrospective

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum Artifacts

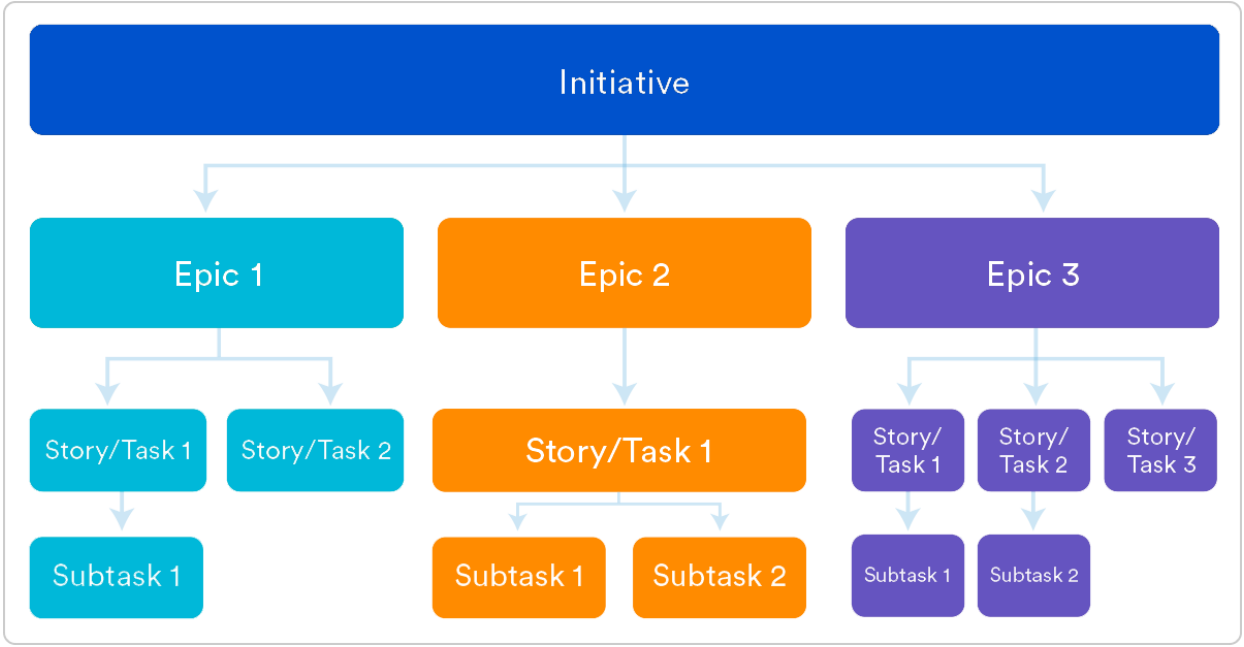


SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Variations:

- User Story
- Technical Story
- Story Points



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

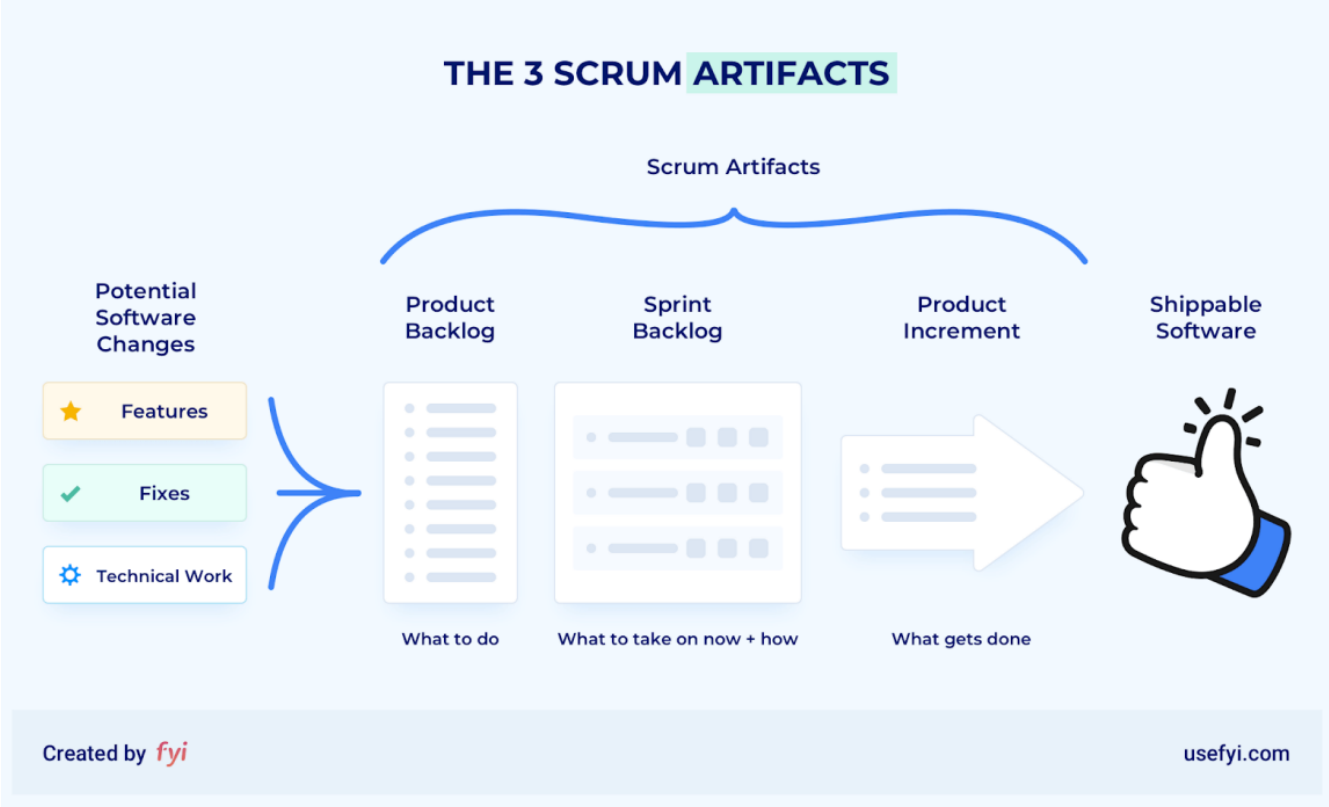
SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

Event	4 Week	3 Week	2 Week	1 Week
Sprint Planning	8 hr	6 hr	4 hr	2 hr
Daily Scrums	15 min daily	15 min daily	15 min daily	15 min daily
Sprint Review	4 hr	3 hr	2 hr	1 hr
Sprint Retrospective	3 hr	2.25 hr	1.5 hr	.75 hr

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum Artifacts



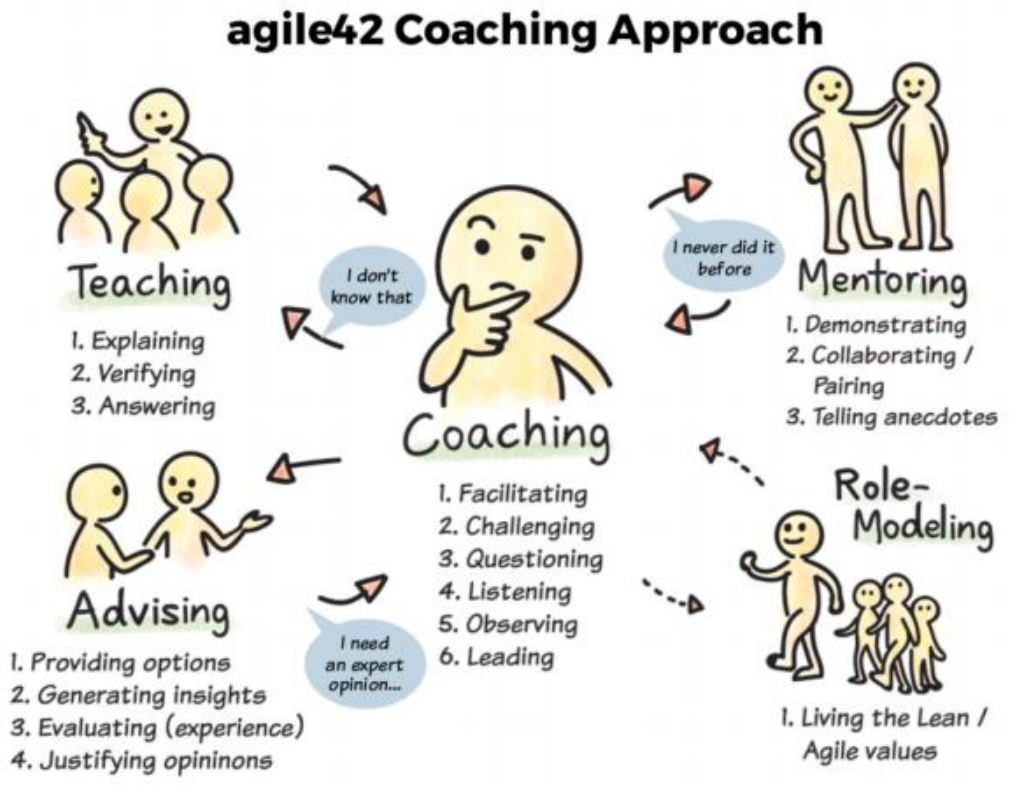
SOFTWARE DEVELOPMENT MODELS

Agile/Scrum

- Agile Project Management: The Daily Scrum meetings, all together give the Project Managertremendous awareness about the state of the project at all times.
- Scrum Master/Project Management: find that planning and tracking are easier and more concrete
- Design Thinking

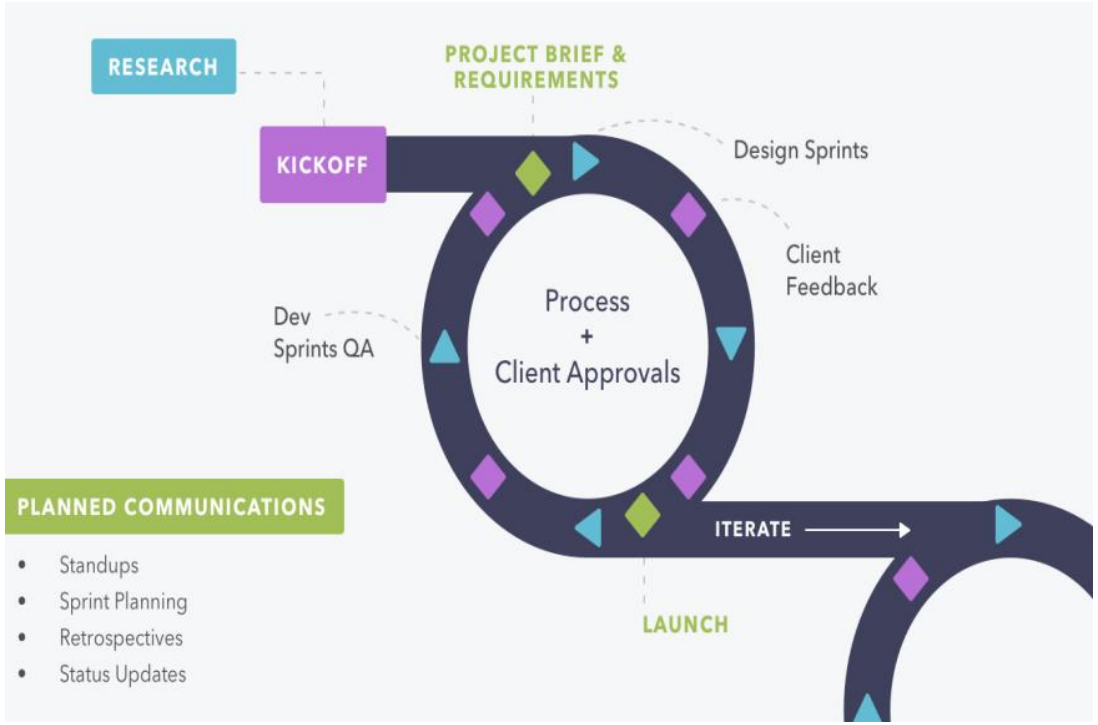
SOFTWARE DEVELOPMENT MODELS

Agile/Scrum



SOFTWARE DEVELOPMENT MODELS

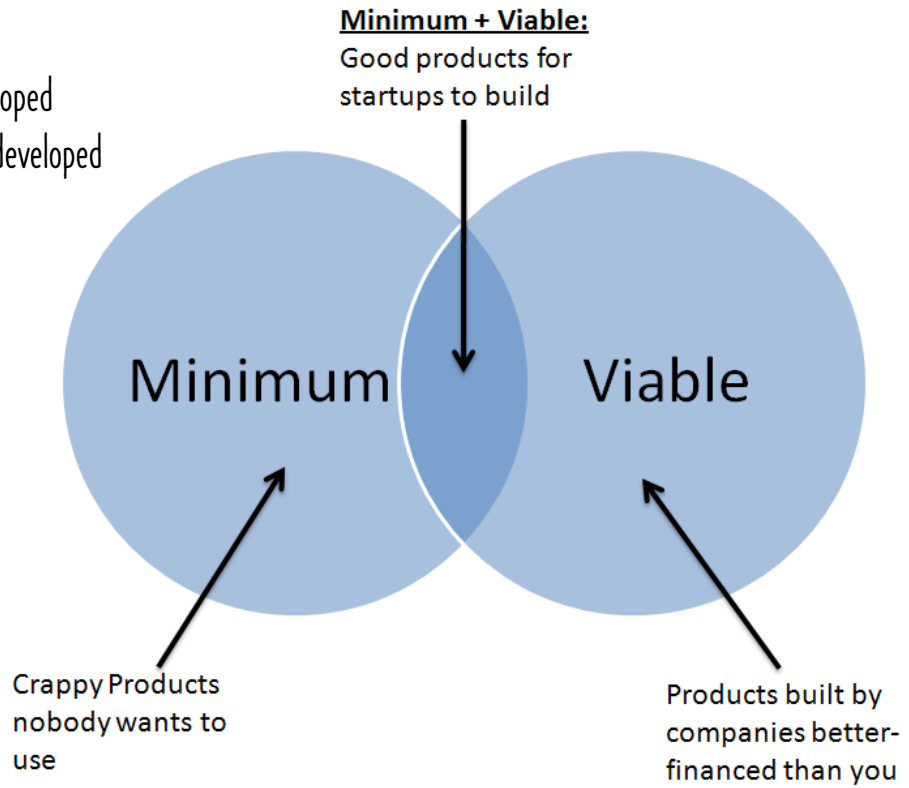
Agile/Scrum



SOFTWARE DEVELOPMENT MODELS

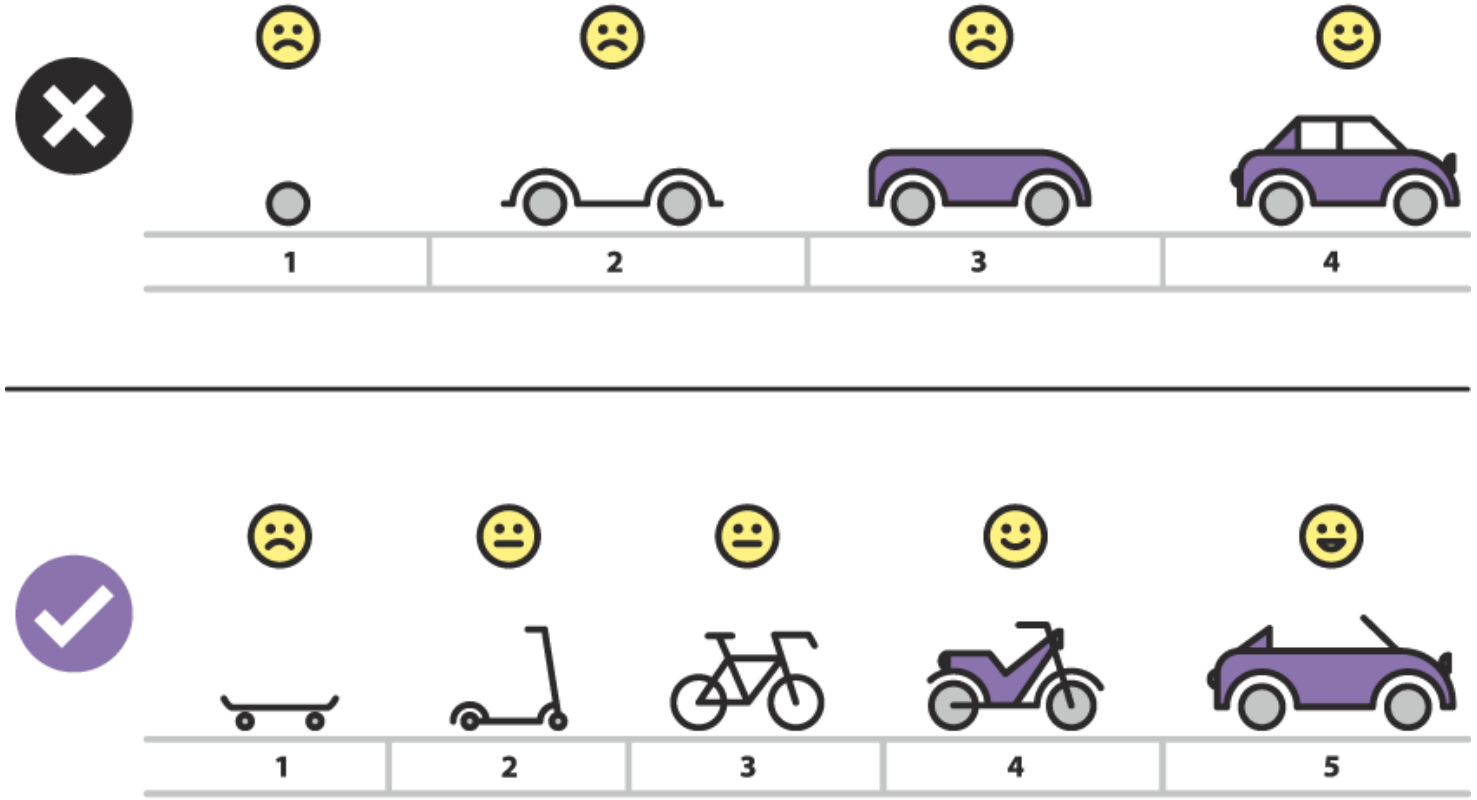
Minimum Viable Product

"A minimum viable product (MVP) is a development technique in which a new product or website is developed with sufficient features to satisfy early adopters. The final, complete set of features is only designed and developed after considering feedback from the product's initial users."



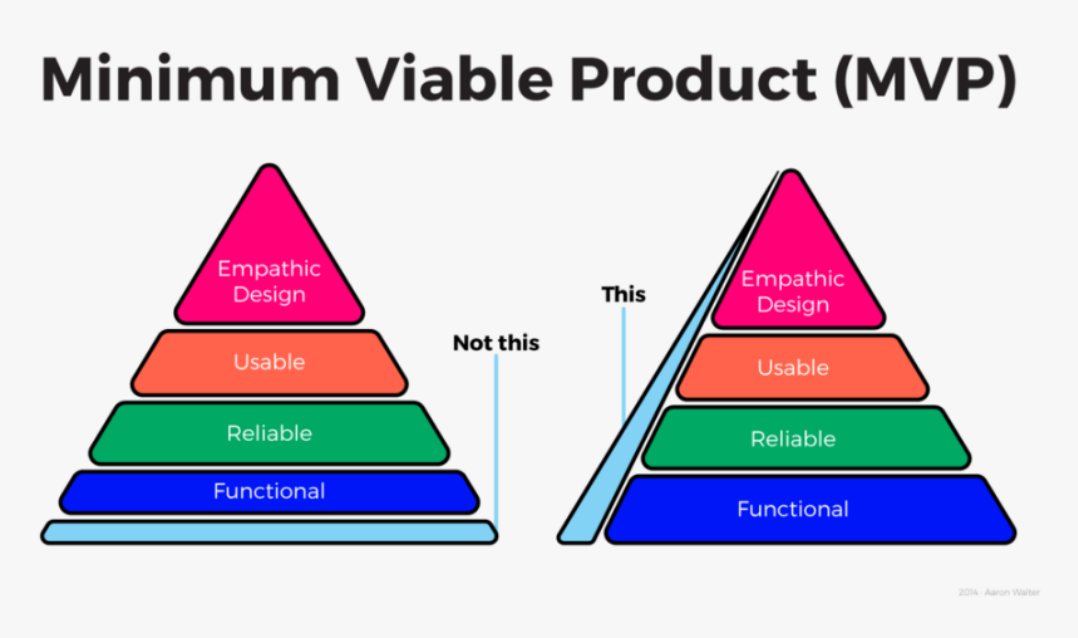
SOFTWARE DEVELOPMENT MODELS

Minimum Viable Product



SOFTWARE DEVELOPMENT MODELS

Minimum Viable Product



SOFTWARE DEVELOPMENT MODELS

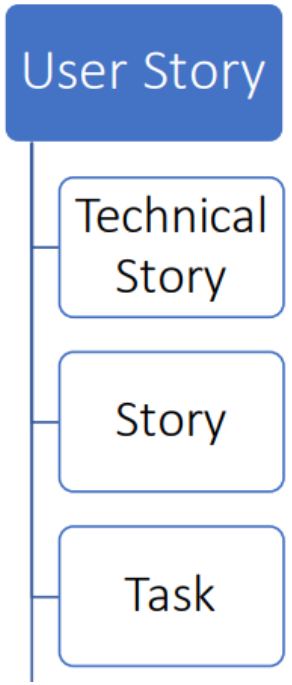
Minimum Viable Product

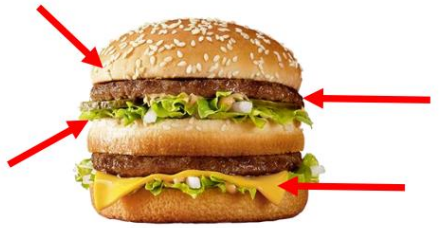
What is NOT an MVP in software development?

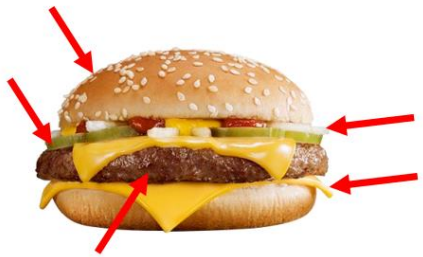
1. MVP is not a prototype
2. A prototype is a mock-up that conveys the idea of your product

SOFTWARE DEVELOPMENT MODELS

Agile/Scrum







UNIVERSIDAD POLITÉCNICA DE YUCATÁN

PROGRAMMING SESSION 1

Variables

- Variables are used to store data in JavaScript, such as numbers and text values. The name of a variable is called an identifier, and it must conform to certain naming rules. A variable can be declared using the var, let or const keyword.
- Global variables exist outside any function while local variables are defined inside a particular function.
- Variable hoisting allows accessing a variable before it is declared in code, but a hoisted variable returns the value of undefined.

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

PROGRAMMING SESSION 1

Variables

var

The var keyword can be used to declare a local or global variable. The variable can be initialized to a value.

let

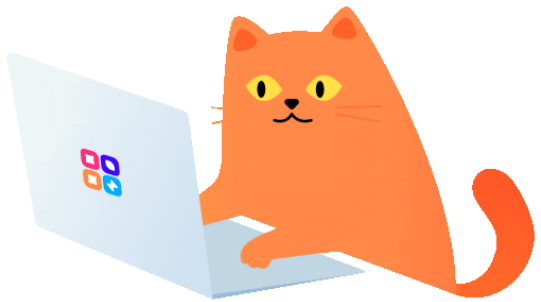
The let keyword can be used to declare a block-scoped, local variable. The variable can be initialized to a value.

const

The const keyword can be used to declare a block-scoped, read-only constant. It must be initialized to a value.

PROGRAMMING SESSION 1

Coding...



UNIVERSIDAD POLITÉCNICA DE YUCATÁN

PROGRAMMING SESSION 1

Data Types

- A variable can be declared and initialized using one of the eight data types.

Number: Number is a primitive data type allows storing an integer or floating point number.

String: String is a primitive data type that allows storing a sequence of characters that represent a text value.

BigInt: BigInt is a primitive data type allows storing an integer with arbitrary precision format.

Object: The Object data type can be used to store keyed collections of data items and complex entities.

Symbol: Symbol is a primitive data type that can be used to define unique and immutable values.

Undefined: The undefined keyword represents a primitive value that is not defined. It is auto-assigned to a variable that has been declared but not initialized.

Null: The null keyword represents any null value. It is a case-sensitive and primitive value.

Boolean: Boolean is a primitive data type that allows storing the value of true or false.

UNIVERSIDAD POLITÉCNICA DE YUCATÁN

PROGRAMMING SESSION 1

Typecasting

- variable of one data type can be explicitly converted to another data type which is called typecasting. The following methods can be used.

Number() When used with the *new* keyword, an **number object** is created. When used as a method, a **number primitive** is returned.	**String()** When used with the *new* keyword, a **string object** is created. When used as a method, a **string primitive** is returned.	**Boolean()** When used with the *new* keyword, a **boolean object** is created. When used a method, a **boolean primitive** is returned.
toString() Method used to convert a **number** to a **string**.	**parseInt()** Method used to parse a **string** argument to an **integer** or **NaN**.	**parseFloat()** Method used to parse an **argument** and return a floating point **number**

PROGRAMMING SESSION 1

Coding...