U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Software Processes

V-Model

Adaptive Software Development

Software Engineering 2018

Bruno Carvalho	Diogo Yaguas	Tiago Castro
up201606517	up201606165	up201606186

October 25, 2018

Contents

1 V-Model **2**

1.1 The Process of V-Model . . . 2
1.1.1 Design Phase . . . 2
1.1.2 Testing Phases . . . 3
1.2 Advantages of V-Model . . . 3
1.3 Disadvantages of V-Model . . . 4

2 Adaptive Software Development **5**

2.1 The Process of Adaptive Software Development . . . 5
2.1.1 Speculation . . . 5
2.1.2 Collaboration . . . 6
2.1.3 Learning . . . 6
2.2 Artifacts . . . 7
2.3 Pros and Cons . . . 7

V-Model

The V-Model is a unique, linear and plan-driven development methodology, defined by the late **Paul Rook** in the 1980s. It was accepted in Europe and UK as a progressive alternative to the Waterfall model.

The V-Model focuses on fairly typical waterfall-esque method that follows strict, step-by-step stages. While initial stages are broad design stages, progress proceeds down through more and more granular stages, leading into implementation and coding, and finally back through all testing stages prior to completion of the project.

The Process of V-Model

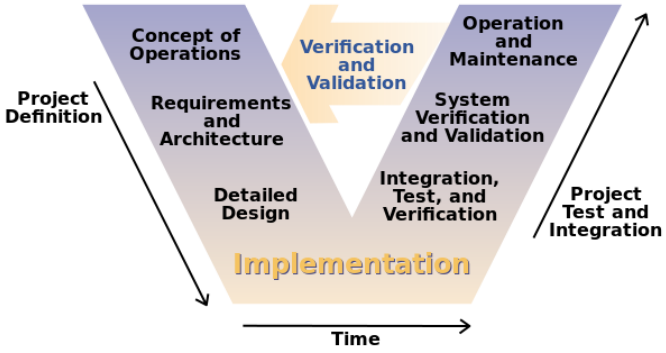


Figure 1: The V-Model stages

Much like the traditional waterfall model, the V-Model specifies a series of *linear* and well-defined stages. For this reason, V-Model is not considered an agile development method. Due to the sheer volume of stages and their integration into the overall process, understanding the model in detail can be challenging for everyone on the team, let alone the client or users.

The V-shape of the V-Model represents the various stages that will be passed through during the software development life cycle. Beginning at the top-left stage and working, over time, toward the top-right tip, the stages represent a linear progression of development, which laid out is rather similar to the waterfall model.

So V-Model contains Verification phases on the left and Validation phases on the right of the development phase — hence the name.

Design Phase

Requirements Gathering and Analysis Comprises detailed communication with the customer to understand the requirements of the project and their expectations.[8]

System Design Here the system engineers analyse and understand the business of the proposed system by studying a requirements document.[7] If requirements are not feasible, the user is informed and they are adjusted. This document contains general system organization, protocols, etc.

In this phase system testing is designed.

Architectural Design The system design is broken down further into modules taking up different functionality and interactions. Here some data structures and interfaces are specified. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.

In this phase integration testing is designed.

Module or Low-Level Design Here the detailed design of each module is completed. Further data structures, low-level interfaces, database tables and dependency issues are designed and specified.

In this phase unit testing is designed.

Testing Phases

Unit Testing Unit Test Plans are developed during the module design phase. These Unit Test Plans are executed to eliminate simple bugs at code or unit level.

Integration testing After completion of unit testing come integration tests, which verify the communication of modules among themselves.

System Testing Tests the application as whole, including functionality, interdependency and communication.

User Acceptance Testing (UAT) UAT is performed in a user environment that resembles the production environment. It verifies that the delivered system meets the user's requirements and is ready for use in real time.

Advantages of V-Model

Suited for Restricted Projects Due to the stringent nature of the V-Model and its linear design, implementation, and testing phases, it's perhaps no wonder that V-Model has been heavily adopted by the medical device industry in recent years. In situations where the project length and scope are well-defined and limited, the technology is stable, and the documentation and design specifications are clear and constant, then this will be a great method.

Ideal for Time Management Along the same vein, V-Model is also well-suited for projects that must maintain a strict deadline and meet key milestone dates throughout the process. With fairly clear and well-understood stages that the whole team can easily comprehend and prepare for, it is relatively simple to create a timeline for the entire development process, while generating milestones for each stage along the way. Of course, V-Model in no way ensures milestones will always be met, but the strict nature of the model itself enforces the need to keep to a fairly right schedule.

Disadvantages of V-Model

Lacks Adaptability Similar to the issues facing the traditional waterfall model on which the V-Model is based, the most problematic aspect is the inability to adapt to any necessary changes during the development process. For example, an overlooked issue within some fundamental system design, that is then only discovered during the implementation phase, can present a severe setback in terms of lost man-hours as well as increased costs.

Timeline Restrictions While not an inherent problem with the V-Model itself, the focus on testing at the end of the life cycle means that it's all too easy to be pigeonholed at the end of the project into performing tests in a rushed manner to meet a particular deadline or milestone.

Ill-Suited for Lengthy Life Cycles Like the waterfall model, the V-Model is completely linear, and thus projects cannot be easily altered once the development train has left the station. V-Model is therefore poorly suited to handle long-term projects that may require many versions or constant updates/patches.

Encourages 'Design-by-Committee' Development While V-Model is certainly not the only development model to fall under this criticism, it cannot be denied that the strict and methodical nature of the V-Model and its various linear stages tend to emphasize a development cycle befitting managers and users, rather than developers and designers. It can be all too easy for project managers or others to overlook the vast complexities of software development in favor of trying to meet deadlines or to simply feel overly confident in the process or current progress, based solely on what stage in the life cycle is actively being developed.

Adaptive Software Development

Adaptive software development (ASD) is an iterative, circular process of software development created by **Jim Highsmith** and **Sam Bayer**. It is a methodology for building complex software applications and systems published in 2000.

ASD is characterized by a rapid creation and evolution of the software application with an emphasis on collaboration. It is a successor to rapid application development and a predecessor of modern agile development techniques.[1,6]

ASD derives its conceptual perspective from the Complex Adaptive Systems (CAS) theory — from which it inherits its Emergence and Complexity — and its practical perspective on the Rapid Application Development techniques and deterministic methodologies.

The ASD cycle has six basic characteristics:[3]

1. Mission focused
2. Feature based
3. Iterative
4. Time-boxed
5. Risk driven
6. Change tolerant

The Process of Adaptive Software Development

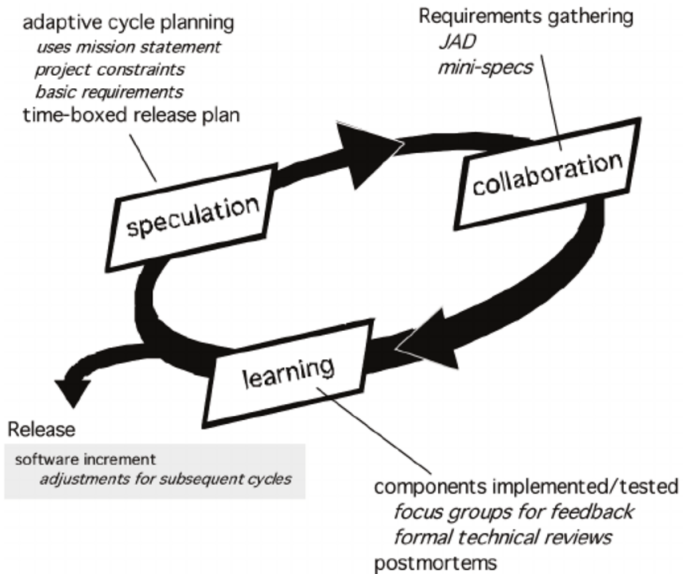


Figure 2: The Adaptive cycle

Adaptive replaces the traditional linear waterfall model with a repeating series of *speculate*, *collaborate* and *learn* cycles. It is characterized by constant change, re-evaluation, peering into an uncertain future and intense collaboration among developers, testers and customers in the various phases.

1. Speculation

This is the setup phase. During it, developers attempt to understand the exact nature of the product's requirements — what features need to be added, what problems need to be fixed, and what changes need to be made.

To facilitate this, a previous cycle feeds information to this phase, namely bugs, user reports, new requirements found during developments and finished or unfinished features.

Let's see how *speculation* comprises *project initiation* and *cycle planning*.

Project Initiation The project's constraints and *mission statement* are set by the customer. The development team determines the project's organization, gathers more requirements, and estimates size and scope.

Cycle Planning The mission statement's aim is to expose the customer's desired result for the project. It's likely that the customer has no knowledge of the field, so this can be done adequately in layman's terms. Using the mission statement, the team forces it's focus on a single area — such as features, schedule, defects, or resources — in which it must excel in the development phase. This is inherently risk-driven: attempting to excel in all dimensions usually results in a product being mediocre in all of them and excellent in none (or delayed releases), so compromises must be made.

Finally, the team fixes a set of features to implement in the next iteration and also *fixes* a certain amount of time for it.

2. Collaboration

This is the development phase. Here, effective collaboration within the team is key. Complex applications are not built — they evolve. In this high information flow environment, one person or small group can't possibly *know it all*, so collaboration skills are paramount.

Collaboration and communication with the client continues, but only to handle minor issues/requirements adjustments and preference resolutions.

3. Learning

This is the quality control phase. Here is where many integration and system tests are performed. Knowledge is gathered about the project's state to prepare for the next speculation phase.

Focus groups, technical reviews and project artifacts are characteristic of this phase. Giving a status report to the customer and receiving feedback is the common goal. In a focus group, the application or system being developed is presented to the customer, who then gives feedback, records change requests, and helps prepare a mission statement.

Summing up, there are four aspects to report on at the end of each iteration:

- Result quality from the customer's perspective.
- Result quality from a technical perspective.
- The functioning of the delivery team and the practices team members are utilizing.
- The project's status.

Artifacts

Project Vision A document which serves to establish a focus for the project,[2] and identifies the foundation on which to build the team's commitment. It should help developers explain the project's purpose within a few minutes. The document may include: Project Background; Project Scope; Sponsors; Customers; Business Functional Objectives; Project Risks; Staffing Requirements; Constraints; Assumption...

Project Data Sheet A document which results from a project initiation. Includes: client, project objective statement, features, performance or quality attributes, architecture, issues and risks, milestones, and the team's core members.

Product Mission Profile A short document used to record focus for the next development iteration, using for instance profile matrices (Figure 3). These may be done per iteration or per feature.[2] In the example, the *Excel* column identifies Scope as the most important dimension for the next iteration.

Product Quality Dimensions	Priority Level		
	Excel	Improve	Accept
Scope (Features)	◆		
Schedule		◆	
Defects			◆
Resources			◆

Figure 3: Mission Profile's Matrix

Project Specification Outline It serves several purposes. First, it provides the stakeholders and team members with a reasonable understanding of the boundaries and scope of the development effort. Second, it is a baseline for size estimation. Finally, the PSO facilitates adaptive cycle planning and is accomplished by assigning product features to specific cycles. The specification outline's primary objective is to define the features or functionality of the product.[2]

Pros and Cons

Pros

- Allows for transparency because developers explain the state of the project to the client before, during and after each stage;
- Generates high quality software, breaking down the project into components, concentrating on high quality development, testing of one component at a time, aids in producing a high quality end product in the end run.

Cons

- Requires a lot of the client's insight, thus requiring a lot of time from them;
- Due to the lack of predictability, it is hard to develop the business case for the project and makes it even more difficult to negotiate for fixed prices and cost of the projects;
- Testing at every stage helps deliver a quality product but increases the cost of the project in the long run and brings a lot of projects to failure.

References

[1] adaptivesoftwaredevelopment.wikidot.com. *Adaptive Software Development*. URL. Aug. 2011. (Visited on 10/25/2018).

[2] James A. Highsmith. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Addison-Wesley Professional, July 2013.

[3] Jim Highsmith. *Agile Software Development Ecosystems*. Addison-Wesley Professional, Apr. 2002, pp. 173–179.

[4] Andrew Powell-Morse. *V-Model: What is It And How Do You Use It?* URL. Dec. 2016. (Visited on 10/24/2018).

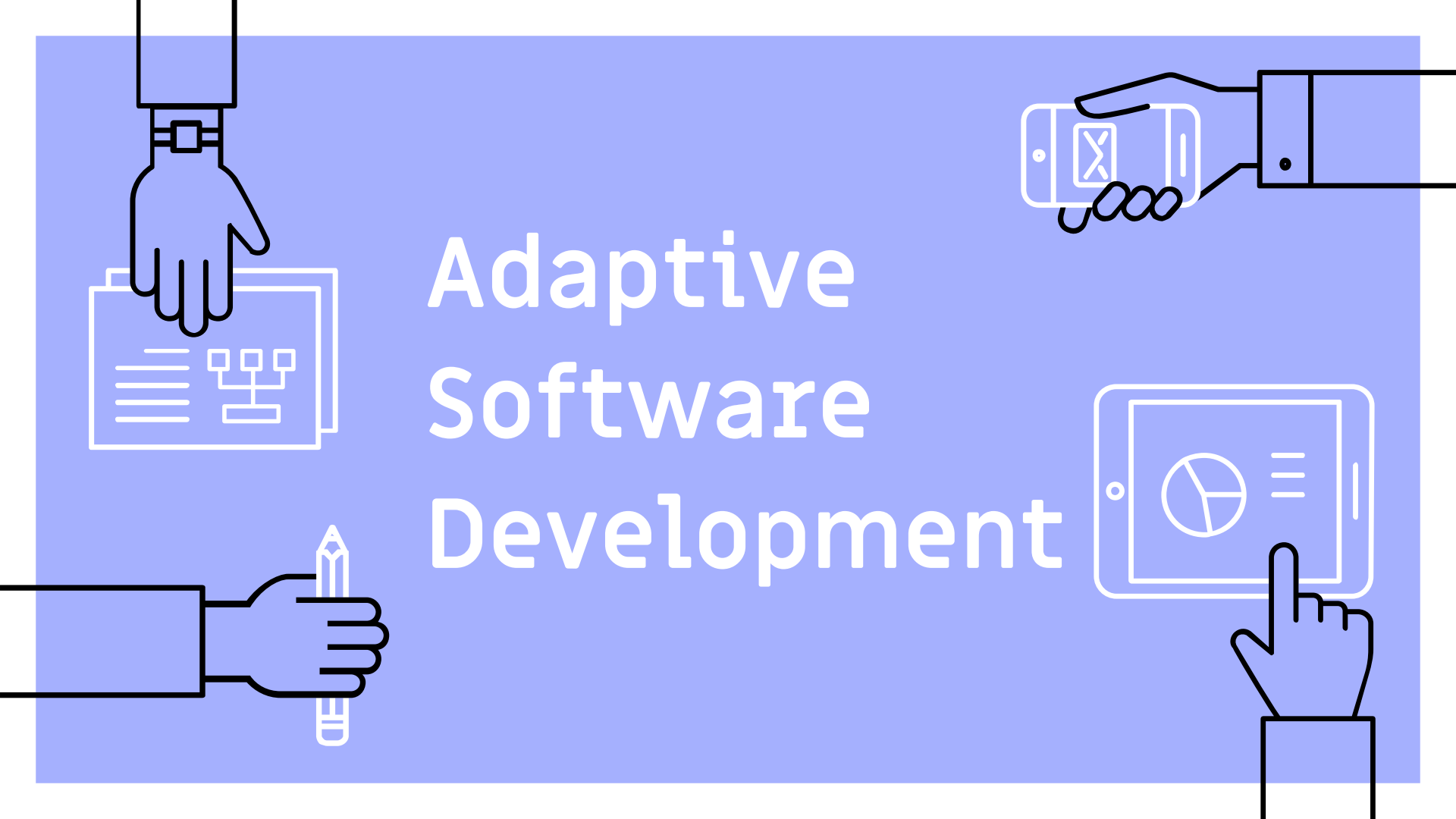
[5] Margaret Rouse. *What is V-Model (Vee-Model)?* URL. June 2016. (Visited on 10/24/2018).

[6] Wikipedia. *Adaptive Software Development*. URL. (Visited on 10/23/2018).

[7] Wikipedia. *V-Model (Software Development)*. URL. (Visited on 10/24/2018).

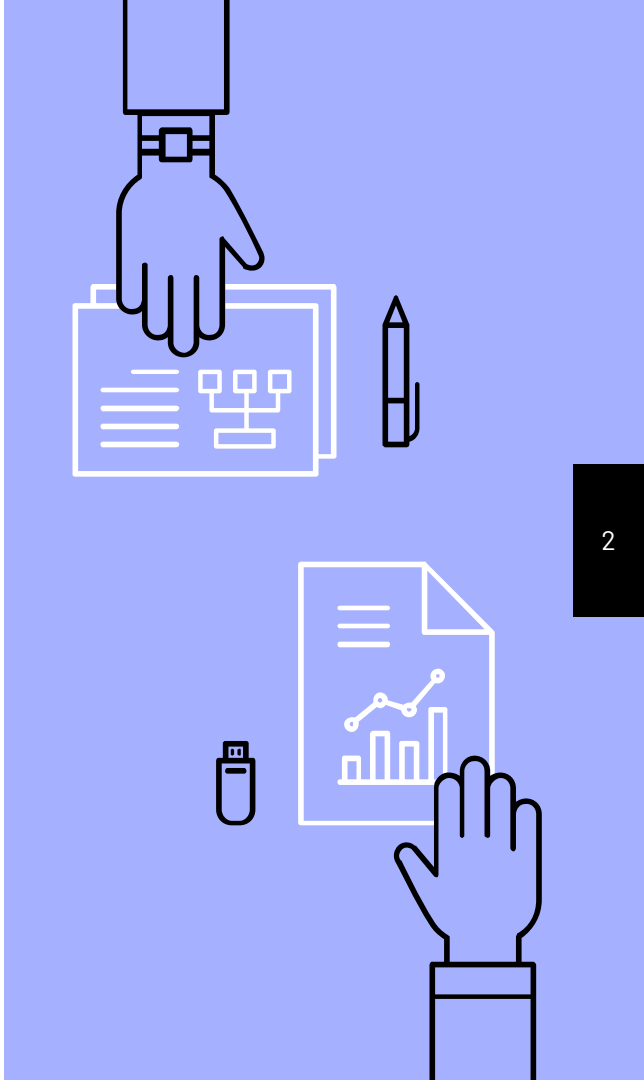
[8] www.geeksforgeeks.org. *Software Engineering | SDLC V-Model*. URL. (Visited on 10/24/2018).

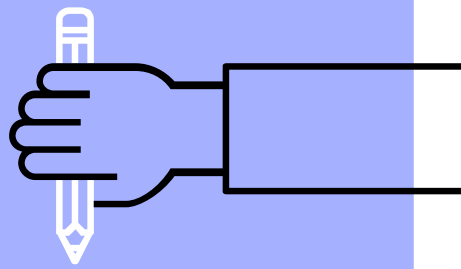
Adaptive Software Development



ASD HISTORY

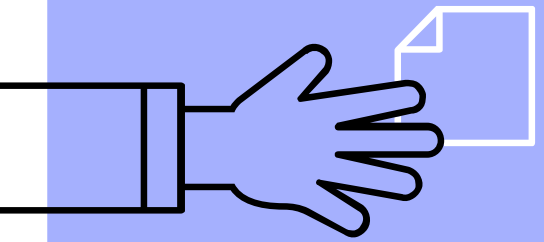
- Iterative software development process
- Proposed by Jim Highsmith and Sam Bayer
- Published in 2000
- Focused on rapid creation and evolution of software systems
- It is one of the agile development techniques





1.

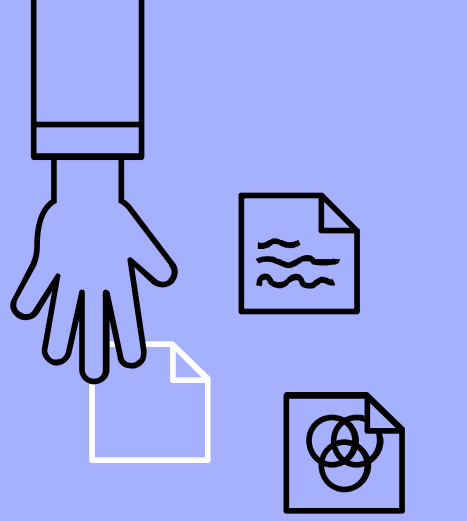
Principles, Values & Practices

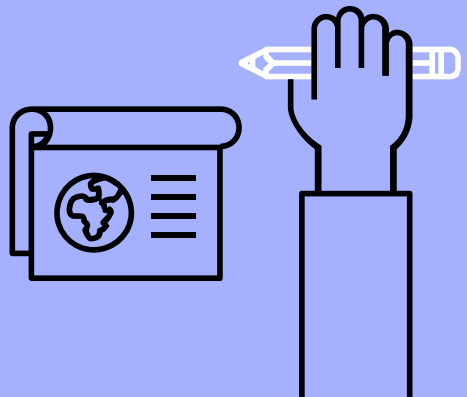


Replaces the traditional waterfall cycle which is feature-box oriented

Is time-box oriented

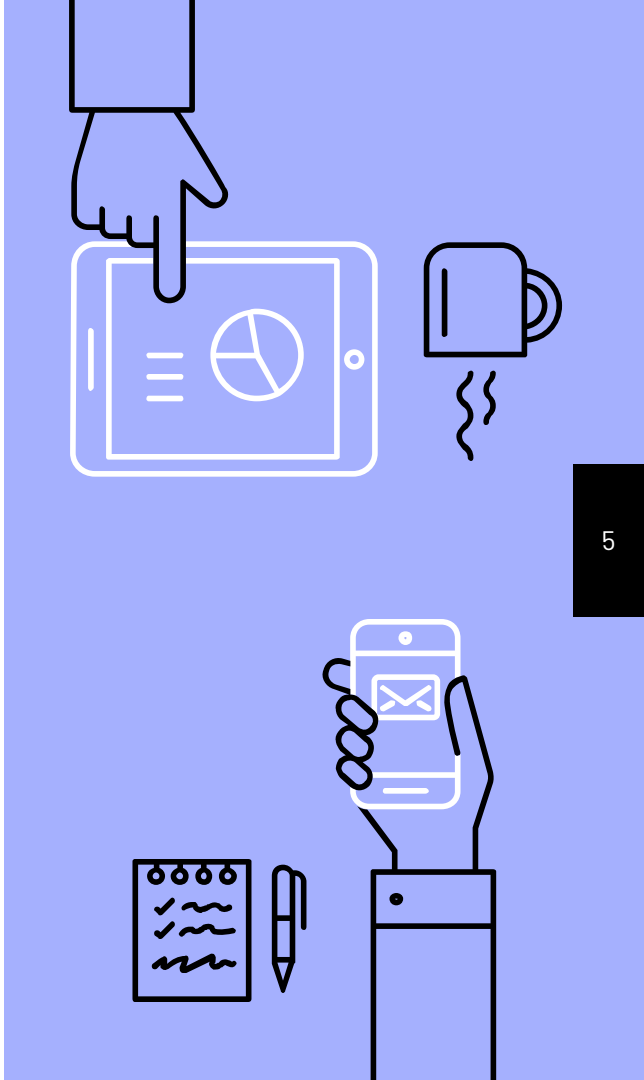
Characterized by constant change, re-evaluation and collaboration between parties

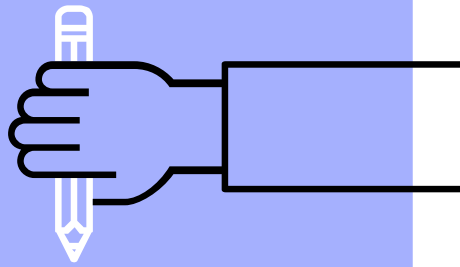




ASD life cycle has six basic characteristics:

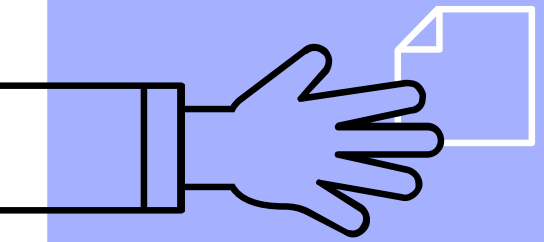
1. Mission focused
2. Feature based
3. Iterative
4. Time-boxed
5. Risk driven
6. Change tolerant



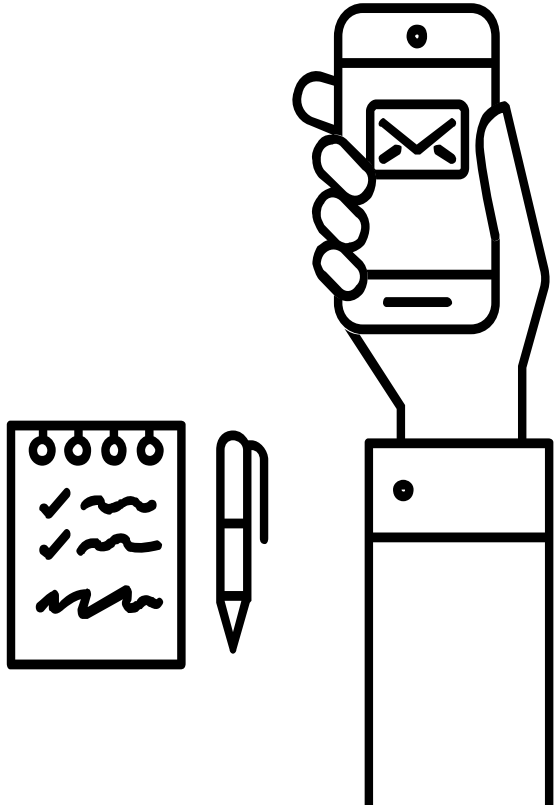


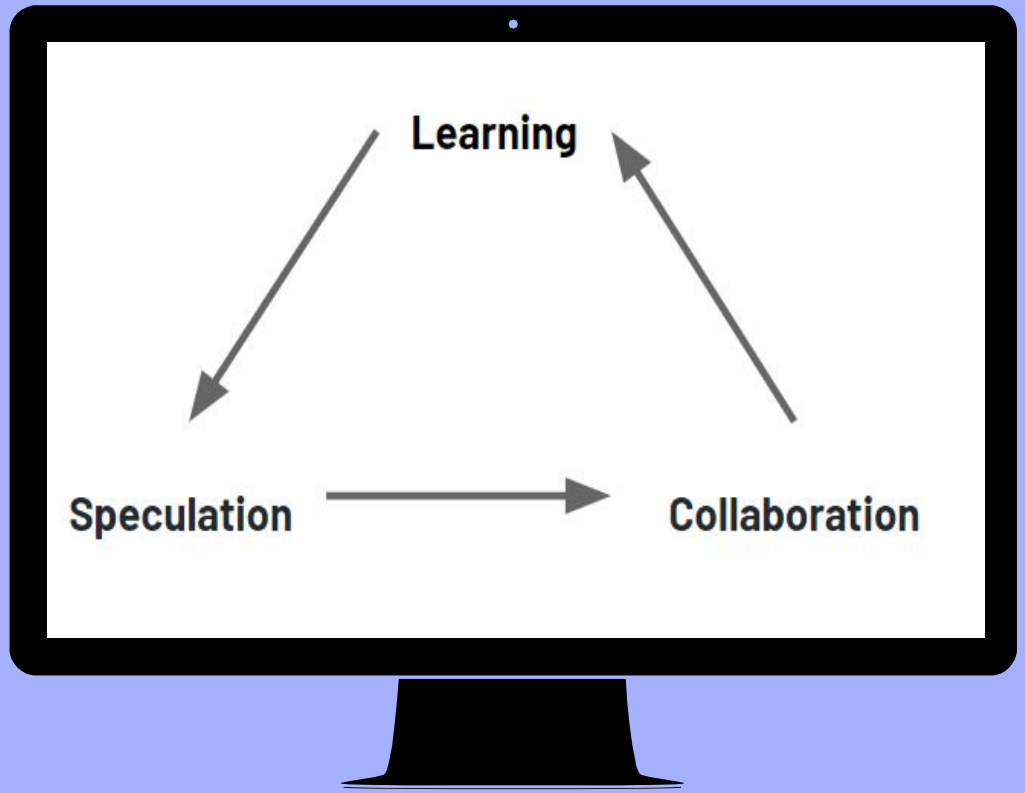
2.

Roles, Artefacts, Activities, Techniques & Tools



You do not navigate a company to a predefined destination. You take steps, one at a time, into an unknowable future. ... In the final analysis, it is the walking that beats the path. It is not the path that makes the walk.




Learning
Speculation
Collaboration

SPECULATION

Relies on bug and user reports to guide the project

The Speculation step is divided in project initiation phase and adaptive cycle planning

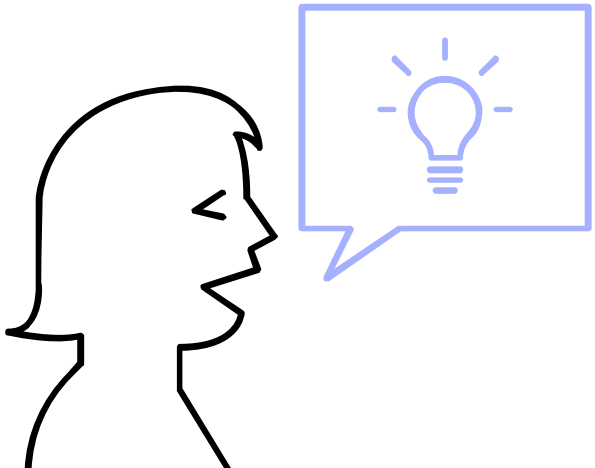
In the project initiation phase, the team and clients gather and determine the project's:

- Constraints
- Mission
- Organization
- System requirements
- Estimated size and scope

In the adaptive cycle planning phase, the client decides on which features take priority and the team assesses their risks and dependencies. Then a time box is set for the development of the next features, and is fixed regardless of whether the features are finished or not

COLLABORATION

Effective collaboration between teams is key and the client must give feedback

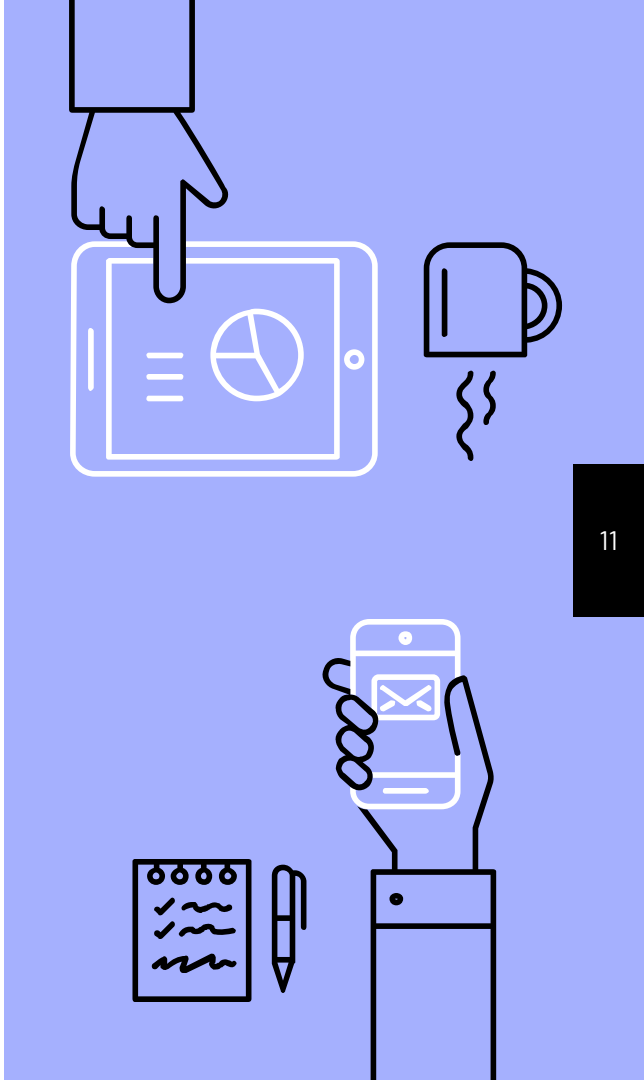


The efforts for balancing the work based on predictable parts of the environment, planning and guiding them, and adapting to the changes caused by various factors, such as technology, requirements, stakeholders and software vendors are the main focus.

In this high-information-flow environment, in which one person or small group can't possibly "know it all," collaboration skills (the ability to work jointly to produce results, share knowledge, or make decisions) are paramount.

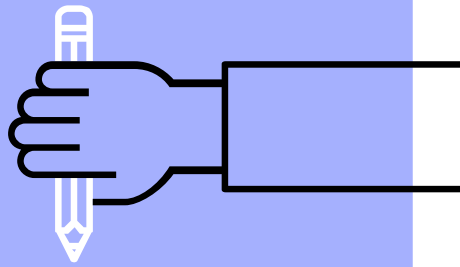
LEARNING

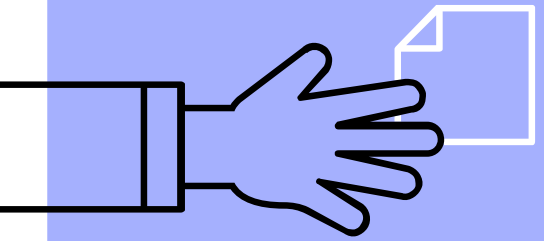
- The learning cycles are based on the short iterations with design, build and testing
- Knowledge is gathered by making mistakes based on false assumptions and correcting those mistakes
- Focus groups, Technical reviews and Project *Postmortems* are conducted in this phase.
- Getting feedback from the customers is the first priority in this phase of Adaptive projects
- Derived from the concept of marketing focus groups, customer focus group sessions are designed to explore a working model of the application and record customer change requests.



3.

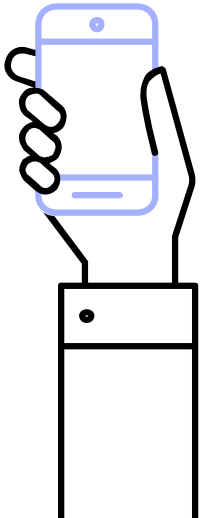
Artefacts

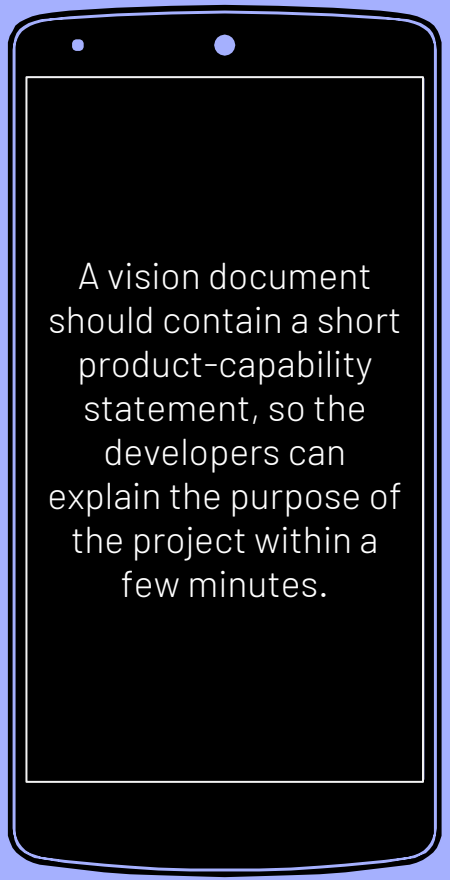




Project Vision (Charter)

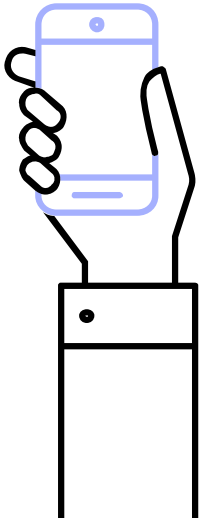
The project vision is recorded in a document, or artifact, which establishes a focus for the project and identifies the foundation on which to build the team's commitment.





Project Data Sheet (PDS)

The PDS is the minimum deliverable from any project initiation activity. Whatever the detailed contents of a project vision paper, a one page PDS should also be developed.

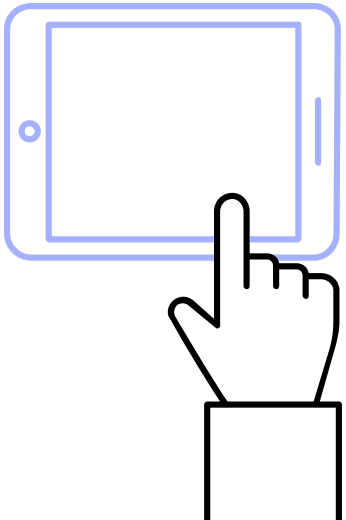


The PDS includes the following details:

-Clients / customers;
-Project objective statement;
-Features;
-Client benefits;
-Performance / quality attributes;
-Architecture;
-Issues/risks;
-Major project milestones;
-Core team members;

Product Mission Profile

It's an important tool for documenting focus, a contract between the development group and the executive sponsor or primary customer.

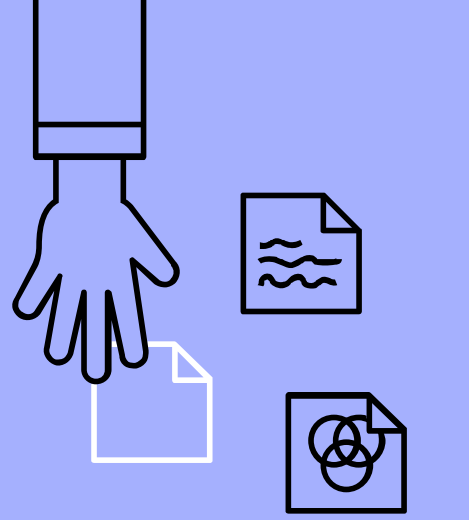


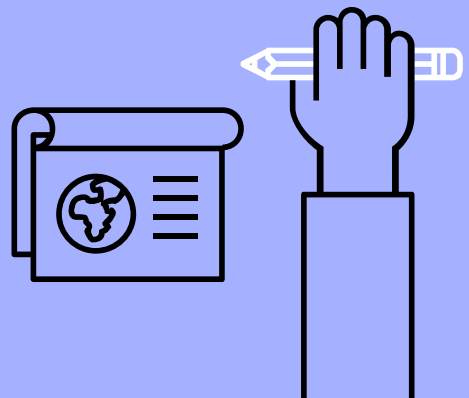
Mission Profile Matrix

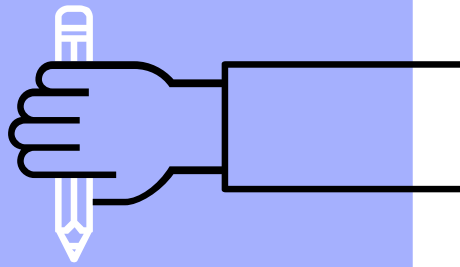
Product Quality Dimensions	Priority Level		
	Excel	Improve	Accept
Scope (Features)	♦		
Schedule		♦	
Defects			♦
Resources			♦

Project Specification Outline (PSO)

- It provides the stakeholders and core project team members with a reasonable understanding of the boundaries and scope of the development effort
- Is the baseline for size estimation.
- Facilitates adaptive cycle planning and is accomplished by assigning product features to specific cycles.
- The specification outline's primary objective is to define the features or functionality of the product

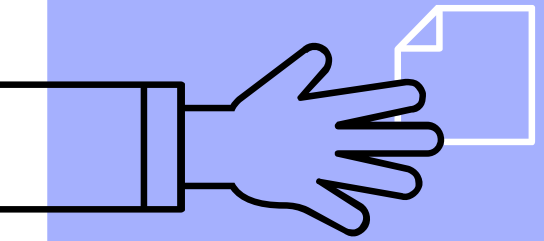






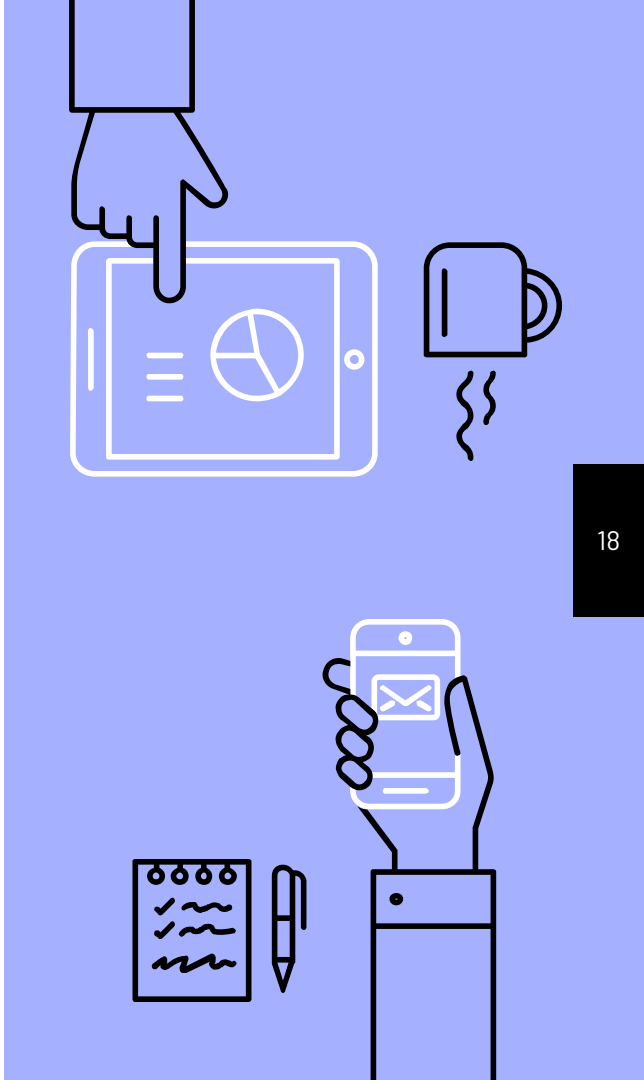
4.

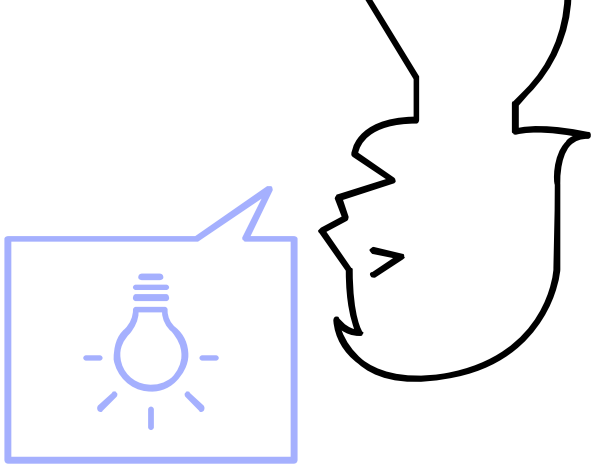
Pros and Cons

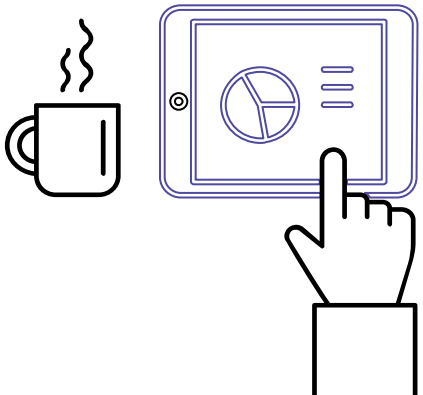


- Allows for transparency because developers explain the state of the project to the client before, during and after each stage.

- Generates high quality software, breaking down the project into components, concentrating on high quality development, testing of one component at a time, aids in producing a high quality end product.







- Requires a lot of the client's insight, thus requiring a lot of time from them
- Due to the lack of predictability, it is hard to develop the business case for the project and makes it even more difficult to negotiate for fixed prices and cost of the projects
- Testing at every stage helps deliver quality products but increases the cost of the project in the long run and brings a lot of projects to failure.

THANKS!

Any questions?

Bruno Carvalho

- E-Mail: up201606517@fe.up.pt

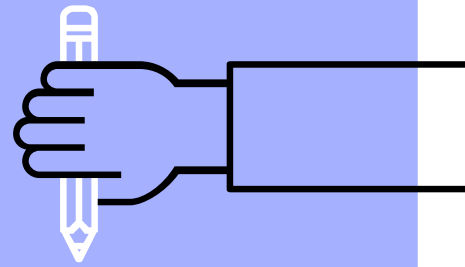
Diogo Yaguas

- E-Mail: up201606165@fe.up.pt

Tiago Castro

- E-Mail: up201606186@fe.up.pt.

References

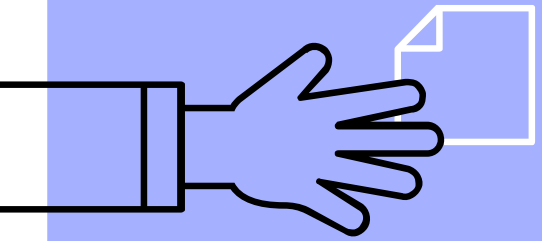


Adaptive Software Development. James A. Highsmith III.
http://homepages.herts.ac.uk/~comqjs1/Highsmith.pdf

http://www.exa.unicen.edu.ar/catedras/agilem/cap23asd.pdf

What is Agile Development. SourceSeek. July. 2013.
https://www.youtube.com/watch?v=-zDct5d2smY

https://www.youtube.com/watch?v=N4NQ6rZiQ5Q

Adaptive Software Development. 2011

http://adaptivesoftwaredevelopment.wikidot.com/