**Description of the Minimum Viable Product (MVP) Life Cycle**

Created on November 9, 2023

**Sponsor:** The Fellows Consulting Group (FCG)

**Sponsor Mentor:** Tom Hill

Done as a part of the SE4485 Software Engineering Project course at The University of Texas at Dallas.

**Authors:** Group 7 -

Jacob Kotrla

Neil Barot

Laura Genung

Duke Cha

Steven Vargas

**Abstract**

This document describes the MVP life cycle. It does into a detailed explanation about what the life cycle is, what its purpose is, and what each of its phases are. The MVP life cycle is a software development life cycle (SDLC) that combines the MVP development process, the waterfall SDLC, and open source software (OSS) into one process. Using the MVP life cycle, companies or developers can use a more formal software development process when they develop using OSS and MVP practices.

**Table of Contents**

[**List of Figures 3**](#_rakdjvlpflzs)

[**List of Tables 3**](#_z6qyjbzczplf)

[**1. Introduction 4**](#_gzowxj1qq5ei)

[1.1 The MVP Life Cycle 4](#_fb1v1qghv8np)

[1.1.1 More Information on the Waterfall Life Cycle 4](#_v12gh99lk336)

[1.1.2 More Information on the MVP Development Process 5](#_bnk20b5kkc9f)

[1.1.3 More Information on Open Source Software (OSS) 5](#_mlzuhpfdz4w0)

[1.2 The Purpose of the MVP Life Cycle 5](#_cz1irs4uarc9)

[1.3 Overview of the MVP Life Cycle and its Phases 5](#_tdjxijidyun3)

[**2. Life Cycle Phase 1: Requirements Gathering 6**](#_xj6g32vsk7f0)

[2.1 Initiation 6](#_lgwn134it1rh)

[2.2 Gather information 6](#_1rmeooyoa1x)

[2.3 Analysis 7](#_3q0t9fsh13y5)

[2.4 Requirements Specification 7](#_uf0l1ioe2ntm)

[2.5 Implementing MVP Releases 7](#_tbbj3txwpeba)

[**3. Life Cycle Phase 2: Open Source Software Selection 7**](#_dchwgesvlgic)

[3.1 Why Use OSS 7](#_2hsaahfhbmje)

[3.2 Some of the Things to Consider When Selecting an OSS 8](#_l4a2xiy387pj)

[**4. Life Cycle Phase 3: Design 8**](#_po7iu7sppcw2)

[**5. Life Cycle Phases 4-9: MVP Development Loop 9**](#_v8liw77hc45p)

[5.1 Life Cycle Phase 4: Implementation 9](#_itiyxjjeco56)

[5.2 Life Cycle Phase 5: Testing 9](#_tb6qk61zla41)

[5.3 Life Cycle Phase 6: Deploy 9](#_jbm0umfnuidw)

[5.4 Life Cycle Phase 7: Feedback 9](#_7bm6un1c8f54)

[5.5 Life Cycle Phase 8: Update Requirements 10](#_rj90oqbrb3nq)

[5.6 Life Cycle Phase 9: Update Design 10](#_s77u3wpj3sp5)

[**6. Life Cycle Phase 10: Maintenance 10**](#_bw6d88ax1529)

[**7. References 10**](#_9z8ftw9vid4q)

[**Appendix A: Content Management 11**](#_af6vltt2ypat)

[**Appendix B: Additional Resources for Using OSS in Enterprise Software Development 12**](#_10ur8c1cpn3)

## **List of Figures**

Fig. 1. Waterfall Life Cycle Diagram ……………………………………………………………….…4

Fig. 2. MVP Life Cycle Diagram ………………………………………………………………….......6

## **List of Tables**

Table I. Configuration Management ………………………………………………………………..…11

## **1. Introduction**

The purpose of this document is to explain the minimum viable product life cycle (MVP life cycle) in more detail and to guide the reader in implementing the life cycle. This document will walk the reader through each phase of the life cycle giving a high level overview of what should occur in each phase. Please note that the low-level details of how each organization and project should enact the life cycle depends on the situation and environment that the life cycle is being applied to.

### **1.1 The MVP Life Cycle**

The MVP life cycle is a software development life cycle (SDLC) that combines the waterfall SDLC, the MVP development process, and open source software (OSS) into one unified process.

#### **1.1.1 More Information on the Waterfall Life Cycle**

The waterfall life cycle is the most basic SDLC. It is a linear and sequential approach to software development [1]. It typically has around seven main phases. Those phases are requirements gathering, analysis, design, implementation, testing, deployment, and maintenance [1]. These phases are to happen one after another with very little overlap between a phase ending and another one starting.

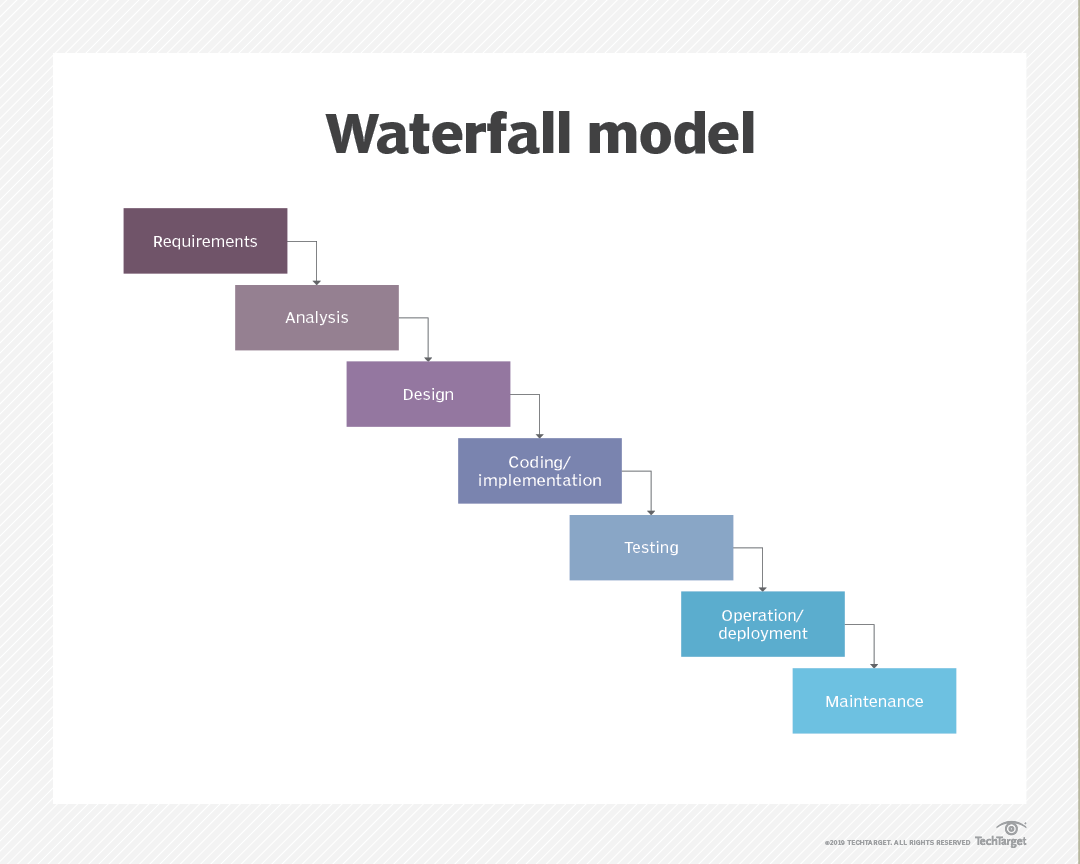


Fig. 1. Waterfall Life Cycle Diagram. [1]

#### **1.1.2 More Information on the MVP Development Process**

The MVP development process is an iterative development technique. To start this process, a barebones product is first built with the minimum amount of features needed to satisfy its early users, the minimal viable product [2]. This product is then released and feedback is gathered from users [2]. The product then goes back into development with the feedback being taken into consideration and new features added. That product then gets released and more feedback is gathered. This cycle of releasing versions of the product and gathering feedback is then continued until the final version of the product is released.

#### **1.1.3 More Information on Open Source Software (OSS)**

Open source software (OSS) is when software is developed and maintained through open collaboration with a public community [3]. Typically anyone with the right expertise can contribute to developing the software. Additionally, the software is typically free and available for anyone to use, analyze, change, and redistribute in any way [3].

This OSS is the opposite of proprietary or closed source software [3]. This software is sold to end users by the inventor of the software or the current copyright holder [3]. This type of software cannot be edited, enhanced, or redistributed without explicit permission from the copyright holder [3].

### **1.2 The Purpose of the MVP Life Cycle**

Many companies and software engineers use Lean Development Practices without a formal or standard process [4]. These Lean Development Practices can be incorporated into a combination of the MVP development process and the waterfall life cycle [4]. Many software development companies tend to use this combinated process and create some form of informal, non-standardized process [4]. Additionally, a lot of these companies already used this informal process with the act of analyzing and editing OSS to fit their enterprise needs. This entire process should be documented and made into a standard to ensure it is being properly done and that it can be reused from project to project. Thus, the MVP life cycle was created to formally document and standardize this process.

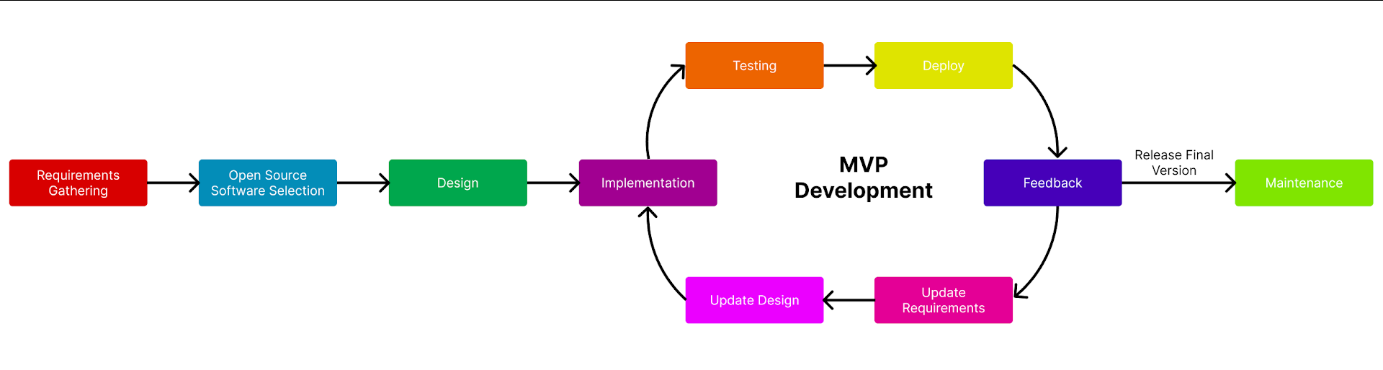
### **1.3 Overview of the MVP Life Cycle and its Phases**

The MVP life cycle has 10 phases. Some phases have a couple of subparts to them that will be further explained in the document.

The phases are:

1. Requirements Gathering
2. Open Source Software Selection
3. Design
4. Implementation
5. Testing
6. Deploy
7. Feedback
8. Update Requirements
9. Update Design
10. Maintenance

Phases 1-3 are to be done linearly, while phases 4-9 are a part of the MVP development loop. This MVP development loop goes through the same steps that the normal MVP development process does. This loop is entered in through the implementation phase (phase 4) and it is exited through the feedback phase (phase 7) once your product is in its final version. Once the final version of the product has been developed and released, the life cycle ends with the maintenance phase. Please see the figure below for a visual representation of the life cycle.

Fig. 2. MVP Life Cycle Diagram.

## **2. Life Cycle Phase 1: Requirements Gathering**

The first phase of the MVP life cycle is the requirements gathering phase. This phase combines the requirements gathering and analysis phases of the waterfall life cycle into one whole phase. The subparts of this phase are listed below and it is highly recommended that these subparts are done in the order listed.

### **2.1 Initiation**

First, identify the problem statement on why you need a new system or project. Then, using the problem statement, define the initial project’s scope, objectives, and constraints.

### **2.2 Gather information**

After you have a general starting point, gather information about the project domain. Engage with stakeholders, including end-users, customers, and other relevant parties. Use various techniques such as interviews, surveys, workshops, and observation to gather information. From these stakeholder interactions, identify and document user requirements, system requirements, and constraints [5].

### **2.3 Analysis**

Analyze all the information you have gathered and created thus far. After extensively interacting with all of your stakeholders, it is important to reevaluate the project’s problem statement, scope, objectives, and constraints to make sure it aligns with the information gathered from your stakeholders. Some slight changes might need to be made now that you and your team are more knowledgeable about the project domain.

Next, create models and diagrams to represent requirements. Make sure to establish the relationships and dependencies between different requirements.

### **2.4 Requirements Specification**

Now that you have done extensive research on the background of your project, it is time to create an official requirements specification document. Once you have a full requirements specification, you should prioritize requirements based on their importance and value to the business. This prioritization will allow you to plan out what initial features will be released with each MVP development cycle. Using prioritization, you should have identified the core features for the first MVP release, as this will be the set of requirements that you ranked with the highest priority.

### **2.5 Implementing MVP Releases**

Using the MVP development cycle, you should implement features in an iterative manner, incorporating feedback and making improvements. With each subsequent release, features and nonfunctional requirements should be added based on the prioritized requirements and any feedback gained.

## **3. Life Cycle Phase 2: Open Source Software Selection**

Once you have made the proper requirements specification and know what your software product must do, it is time to enter into phase two of the MVP life cycle. This phase is the OSS selection. In this phase you will be selecting an OSS that you can use as a tool or as the backbone to build your product off of. This will require you to research, analyze, and compare different OSSs to see which one would be the most beneficial to modify into your product.

### **3.1 Why Use OSS**

There are several benefits to using OSS in the development of your software product. The biggest advantage to using OSS is less development time. As long as you have a competent team that understands what they are doing, using an OSS can significantly save development time. OSS has already written code that you can use, most have documentation you can use to understand and build on top of the code, and most OSS have an active community that you use for help.

Another big advantage is that using OSS can help guide a team in building their own software, especially if this is their first time building this type of software. Sometimes your team needs to build something that they have never built before. Using OSS, a team can have a starting point and can gain a better understanding of that software type much faster than if they were figuring it out on their own.

It should also be noted that using OSS to make your product doesn’t mean that your final software product needs to be open source as well. As long as you abide by the rules of the OSS’s license and your product is significantly different enough from the OSS, you can sell your product as proprietary software and charge money for it.

### **3.2 Some of the Things to Consider When Selecting an OSS**

There are many things you need to consider when selecting the OSS to use in your project, both on a project and organizational level. A team must consider how an OSS will affect their development of the product, how well the OSS fits into their company's ideals and goals, and if the OSS fits with their company’s structure and procedures. If this is the first time a company is using OSS in their development, new organizational procedures might have to be discussed and put into place before the team can proceed with using the OSS.

When directly analyzing the OSS, some additional things to consider include:

* The number of features in the OSS that match your product’s desired features.
* The skill that is required to use the OSS and whether it is in-house.
* The hardware compatibility.
* The operating system compatibility.
* When the OSS was last updated.
* If there is an active community for the OSS and how large is it.
* How much documentation exists for this OSS.

## **4. Life Cycle Phase 3: Design**

Phase 3 of the MVP life cycle is the design phase. This phase is almost identical to the design phase found in the waterfall life cycle. Just like with the waterfall life cycle, in this phase, the project team makes design specification documents to plan and outline the technical design requirements of the project [1]. The program’s architecture, services, and software components are decided and detailed in this phase [1]. However, unlike with the waterfall life cycle, since the programs built with the MVP life cycle are using OSS as their base, the pre-existing design and limitations of the OSS must be taken into account when designing your program. It is best to thoroughly review the documentation and code of the OSS you are using before you start designing your program. Then make sure to always keep the design and structure of the OSS in mind when you are creating your own program’s design.

## **5. Life Cycle Phases 4-9: MVP Development Loop**

With this life cycle, the MVP development loop is an ongoing process where each iteration results in a new version of the MVP being deployed. The loop is initiated in the implementation phase and exits through the feedback phase after the successful release of the software’s final version.

### **5.1 Life Cycle Phase 4: Implementation**

The implementation phase marks the initiation of the MVP development loop. In the implementation phase, the team creates the product. Developers collaborate to write and integrate code according to the project's requirements defined in the requirements gathering phase. The goal is to create a functional product in a short amount of time to test if the idea is viable [2].

A first basic product is produced in the first iteration through the loop. Then each time you go through the loop and circle back to this phase, you develop more and more of the product. This eventually results in the final version of the product being built. It is important to note that during this phase careful consideration is given to the integration of the OSS.

### **5.2 Life Cycle Phase 5: Testing**

Given that the testing phase is a part of the development loop, requirements testing is important for the early validation of core features, which define the product's minimum viable functionality. Additionally, it's crucial to test the compatibility of the OSS with the new implementations to ensure they work well together, confirming a smooth interaction. Performing regression testing during each MVP cycle is equally important, ensuring that new features added in each iteration don't cause unexpected problems or disrupt existing features [5]. Each testing phase builds upon the previous one, validating new features while confirming the stability of existing functionalities.

### **5.3 Life Cycle Phase 6: Deploy**

The deployment phase marks the transition from development and testing to the actual use of the system by end-users. The goal of the deployment process is a smooth launch and seamless integration. For projects involving multiple releases, each release requires a distinct iteration of the deployment phase. This phase involves activities such as notifying users, conducting training sessions, installing hardware, and integrating the system. The aim is to ensure the system operates in alignment with the established user requirements.

### **5.4 Life Cycle Phase 7: Feedback**

In the MVP feedback phase, the main aim is to learn from users by releasing the initial version to a targeted audience and gaining valuable insights on the product. Various feedback methods, like surveys, interviews, user testing, analytics, reviews, or comments, are employed based on specific goals [6]. After collecting feedback, the crucial step is analyzing it for patterns, identifying strengths, weaknesses, and customer needs. When making changes to the MVP, prioritizing the feedback based on relevance, urgency, and impact is important [6]. This feedback loop continues until product goals are achieved and the final version of the software is released, marking the exit from the loop.

### **5.5 Life Cycle Phase 8: Update Requirements**

After gathering and analyzing feedback, the process of updating requirements is necessary. This involves implementing user feedback to improve the product and its functionality. The MVP approach involves reviewing existing requirements based on user feedback. Adjustments are then made to better match the product with user preferences and address concerns. This phase ensures an iterative improvement process through the development cycle.

### **5.6 Life Cycle Phase 9: Update Design**

Alongside updating requirements, updating design involves refining the product's design elements based on user feedback. Updating design involves careful consideration of the pre-existing design and limitations of the chosen OSS. Using feedback, adjustments are made to the program's architecture, services, and software components to enhance user experience, usability, and overall functionality. The process also involves aligning the design with the emerging requirements for the next MVP cycle. Iterative design updates ensure that the product meets user expectations and remains adaptable to evolving needs.

## **6. Life Cycle Phase 10: Maintenance**

The last phase of the MVP life cycle is the maintenance phase. The software should remain in this phase for the entire time that your organization supports the software. At this point in the life cycle, the final product has been released after going through the MVP development loop multiple times, implementing all of the program's features, and getting good enough feedback to consider the product to be fully launched. Now it is time to make updates, improvements, bug fixes, and other quality of life changes to the software. These maintenance changes to the software might be done through releasing patch updates or new versions of the software [1].

## **7. References**

[1] B. Lutkevich and S. Lewis. “Waterfall Model.” TechTarget. <https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model> (accessed Nov. 12, 2023).

[2] Powercode. ”What is MVP in Software Development?” LinkedIn. <https://www.linkedin.com/pulse/what-mvp-software-development-powercodecouk/> (accessed Nov. 12, 2023).

[3] ”What is Open Source Software?” IBM. <https://www.ibm.com/topics/open-source> (accessed Nov. 12, 2023).

[4] T. Hill, “Develop a MVP Process: Research and develop a New Minimum Viable Product Software Development Process,” presented at the SE 4485.001 First Class Project Picking, Richardson, TX, USA, August 25, 2023.

[5] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes," in ISO/IEC/IEEE 12207:2017(E) First edition 2017-11, pp.1-157, 15 Nov. 2017, doi: 10.1109/IEEESTD.2017.8100771.

[6] Lean Startup, “What are some best practices or tools for creating and managing feedback loops for your MVP?” LinkedIn. <https://www.linkedin.com/advice/1/what-some-best-practices-tools-creating-managing-feedback#define-your-feedback-goals> (accessed Nov. 13, 2023).

## **Appendix A: Content Management**

To make sure this document stays up-to-date, it is important to revise it when new information can be added. Additionally, it is important to keep track of the revising history and any changes made between versions of this document. The below table is used to track this document’s revision history and its current version.

**TABLE I**

**CONFIGURATION MANAGEMENT**

| **Version before check in** | **Version after check in** | **Date** | **Reason for change** | **Changes Made By** | **Other information** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## **Appendix B: Additional Resources for Using OSS in Enterprise Software Development**

An OSS strategy is needed for the successful use of OSS across the entire enterprise organization.

The follow resources can help with developing an OSS strategy and implementing OSS into enterprise software development:

* A Guide to Enterprise Open Source by The Linux Foundation
  + The guide can be found at the link: <https://www.linuxfoundation.org/research/guide-to-enterprise-open-source>
* 3 Ways Every Company Can Get Started With An Open-Source Software Strategy by Ashley Wolf from GitHub
  + Article can be found at the link: <https://github.blog/2022-08-26-3-ways-every-company-can-get-started-with-an-open-source-software-strategy/>