The Software Development Life Cycle (SDLC) and Related Methodologies:

An Analysis of SDLC, Iterative and Prototyping Methods, and Modularization in SDLC Design

Dan Bailey

SDEV120 Computing Logic

Michael Heady

September 4, 2025

Abstract

This paper explains the Software Development Life Cycle (SDLC) as a process used to plan, build, and maintain software. It goes over the seven main steps of the SDLC, beginning with identifying the problem and ending with maintaining the finished product. It also looks at two other methods of building software, the iterative approach and prototyping, and compares how they are used. Modularization is also discussed, showing how breaking a program into smaller parts makes it easier to manage and expand. These practices highlight why structure and planning are so important in software development.

The Software Development Life Cycle and Methodologies: An Analysis of SDLC, Iterative and Prototyping Methods, and Modularization in SDLC Design

Software can be something simple like a calculator app or something much bigger, like a system for a hospital or a bank. In either case, programmers need a clear plan before they start coding. This plan is called the Software Development Life Cycle (SDLC). It gives structure to the development process so that programs are reliable, meet user needs, and can be updated later. This paper explains the steps of the SDLC, explores iterative and prototyping methods, and discusses modularization in program design.

The Software Development Life Cycle (SDLC)

The SDLC provides seven main stages (Farrell, 2023):

1. **Understanding the Problem –** Developers figure out exactly what the program should do by talking with users.
2. **Planning the Logic –** Flowcharts or pseudocode are created to outline the program's logic.
3. **Coding the Program –** Programmers write the code in a specific language.
4. **Translating the Program –** The code is compiled or interpreted into machine language.
5. **Testing the Program –** The program is tested to catch and fix errors.
6. **Putting the Program into Production –** The program is released for use.
7. **Maintaining the Program –** Updates and fixes are applied over time to make sure secure and useful for as long as the program is in production

These steps keep development organized and reduce mistakes.

Iterative and Prototyping Methods

The iterative method develops software in small cycles. A working version is released early, then new features are added in each cycle, with testing at every stage. This method is flexible

and works well when feedback is needed throughout the process.

Prototyping is about making a sample version of the software. It allows users to try out the design and give input before the real program is built. For example, a prototype of a food ordering app might test whether customers find the menu easy to use. Unlike the iterative method, prototypes are often temporary, but they save time and money by fixing design problems early (GeeksforGeeks, 2023).

Modularization in Program Design

Modularization means breaking a program into smaller sections that handle specific tasks. For instance, an event registration system might have separate modules for collecting input, checking data, processing payments, and sending confirmation emails. This makes it easier to fix problems and add new features without reworking the entire program. It also improves reusability since modules can sometimes be used in other projects.

Conclusion

The SDLC gives developers a structured plan to follow when creating software. Iterative and prototyping methods offer flexible ways to apply this process, and modularization helps keep programs organized and manageable. Together, these approaches help ensure that software projects are successful, reliable, and easier to maintain over time.

References

Farrell, J. (2023). *Programming logic and design* (10th ed.). Cengage Learning.

GeeksforGeeks. (2023, October 12). *Software engineering | Prototyping model*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-prototyping-model/