ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

UNIVERSITY FOR INDUSTRY

Software Component Design Project

Build and fix Model

Group Members	IdNumber
Selihom Demeke	1159/13
Yihun Shikuri	1317/13
Yodahe Ketema	1328/13
Yordanos Seyoum	1371/13
Yordanos Yirgu	1374/13

1.Introduction

In the world of software development, many methods help teams create effective software solutions. One of these methods is the Build and Fix model, which is known for its simplicity and straightforward approach. This model is especially appealing for small projects and teams with limited resources.

The Build and Fix model works by quickly building a basic version of a software product. After this initial build, developers focus on finding and fixing any problems that come up. This process allows them to respond to user feedback and make changes right away. While this approach can lead to fast results, it also raises concerns about the long-term quality and stability of the software.

In the next sections, we will explore the Build and Fix model in more detail. We will look at its main features, the steps involved, and its advantages and disadvantages. By understanding this model better, we can see how it fits into the larger picture of software development methods and when it is best to use it.

2. Overview of the build and fix model

2.1 History of build and fix model

The **Build and Fix Model** has its roots in the early days of software development, emerging during a time when programming was still a nascent field. This model reflects the informal and unstructured practices prevalent in the 1960s and 1970s, where software was often developed with minimal planning and documentation.Initially, developers focused on creating functional software quickly, often without formal specifications or design processes. This approach was particularly suited for small projects with limited complexity, allowing teams to respond rapidly to user feedback. However, as software systems grew in complexity and the demand for higher quality increased, the limitations of the Build and Fix Model became apparent.Over the years, more structured methodologies such as the Waterfall Model and later Agile practices emerged to address the shortcomings of earlier models, emphasizing planning, documentation, and iterative development. Despite its simplicity, the Build and Fix Model remains relevant in certain contexts today, especially for small-scale projects or prototypes where speed is prioritized over comprehensive documentation.

2.2 Phases of the build and fix model

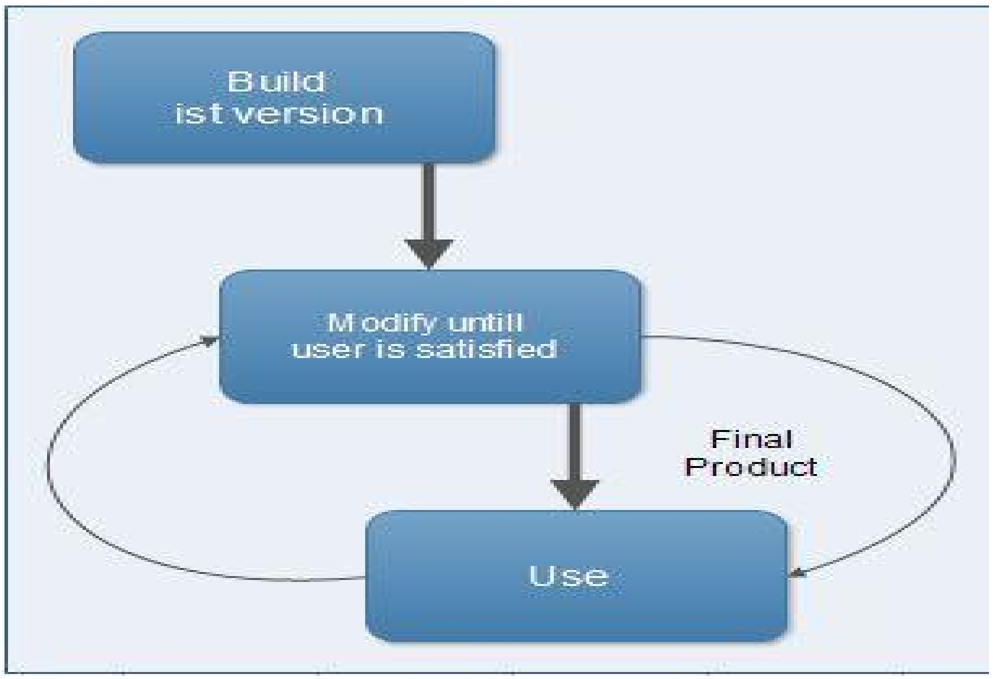
The **Build and Fix Model** is a simple software development approach that focuses on quickly creating a working version of a software product, often referred to as a prototype. The primary goal is to deliver a functional product to users as soon as possible.

1. Build
Initially, developers create a basic version of the software to get it into users' hands quickly. Once this version is released, user feedback becomes essential. Users evaluate the software, identifying issues, bugs, and areas that need improvement.

2. Fix
After collecting feedback, developers enter the "fix" phase. Here, they address the problems reported by users and make necessary changes. This may include fixing bugs or adding new features based on user requests.

3. Iterate
The process of building and fixing continues in cycles until the software meets an acceptable level of quality.

4. Release
After several iterations, consider the software complete or stable enough for broader use, while acknowledging potential quality trade-offs.



2.3 Characteristics of the build and fix model

The **Build and Fix Model** is characterized by several distinct features:

- **Rapid Prototyping**: The model emphasizes quickly creating a working prototype to demonstrate functionality and gather user feedback, allowing for immediate interaction with the software.

- **Minimal Planning**: There is little to no formal planning or documentation involved. Developers jump straight into coding without extensive requirements gathering or design.

- **Continuous Iteration**: The development process is iterative, with frequent updates based on user feedback. This allows the software to evolve as issues and new requirements are identified.

- **Minimal Testing**: Comprehensive testing is often deferred until later stages, which can lead to quality issues in the final product due to insufficient bug detection.

- **User Involvement**: Users are engaged early in the process, providing valuable insights that shape the ongoing development of the software.

- **Informal Nature**: The lack of structured processes can lead to chaotic development, making it challenging to manage larger projects effectively.

3. Advantages and Disadvantages

The Build and Fix Model offers several advantages that make it appealing for certain software development scenarios. One of its primary benefits is rapid development, allowing for quick creation and deployment of software, which is particularly advantageous for small-scale projects or prototypes where speed is essential. Additionally, the model's flexibility enables easy adaptation to changing requirements, allowing developers to respond swiftly to user feedback and needs. User involvement is another positive aspect, as users can interact with the software early in the development process, providing valuable insights that help shape the final product. Furthermore, for very small projects, the model can be cost-effective due to its minimal planning and documentation requirements.

However, the Build and Fix Model also has significant disadvantages. The lack of structure can lead to chaotic development, making it difficult to manage larger projects effectively. Quality issues often arise because minimal testing typically results in software with numerous bugs and reliability problems, which can compromise user satisfaction. Additionally, without proper requirements gathering, projects are prone to scope creep, leading to delays and increased costs as new features are continuously added. The absence of documentation and design specifications further complicates maintenance and updates over time. Ultimately, while the Build and Fix Model may be suitable for small projects, it is often ineffective for complex or larger applications where structured methodologies are more appropriate.

4. Applications

The Build and Fix Model is primarily applied in scenarios where rapid development and user feedback are essential. Its applications include:

Prototyping: The model is often used for creating prototypes or proof-of-concept software, allowing developers to quickly demonstrate functionality and gather user input for further refinement.

Small-Scale Projects: It is suitable for small software projects with limited complexity, where the speed of delivery is prioritized over extensive planning or documentation.

User-Centric Development: The model thrives in environments where user involvement is critical, enabling developers to make iterative improvements based on direct feedback.

Educational Purposes: It serves as a teaching tool in academic settings, helping students understand the basics of software development without the constraints of formal methodologies.

Startup Environments: Startups often utilize this model to rapidly develop and iterate on their products, allowing them to pivot quickly based on market feedback.

While the Build and Fix Model offers flexibility and speed, it is most effective in contexts where the scope is manageable, and the risks associated with quality and maintainability can be controlled.

5. Comparisons with other models

5.1 Build and fix model vs Waterfall model

As discussed build and fix model emphasizes rapid development by initially creating a basic version of the software, which is then refined based on user feedback.In contrast, the Waterfall Model follows a structured, linear process where each phase must be completed before moving on to the next. This model includes distinct stages such as requirements gathering, design, implementation, testing, and maintenance. While the Waterfall Model promotes thorough documentation and clear project milestones, it is inflexible and does not accommodate changes easily once a phase is completed. This rigidity can result in challenges when user requirements evolve during development.

Overall,while the Build and Fix Model offers speed and adaptability at the cost of quality and structure, the Waterfall Model provides a disciplined approach with clear phases but lacks flexibility for changes. The choice between these models largely depends on project size, complexity, and the importance of user feedback during development.

5.2 Build and fix model vs Agile model

The Build and Fix Model and the Agile Model are both iterative approaches to software development, but they differ significantly in their structure, processes, and focus.

The Agile Model emphasizes flexibility and continuous collaboration with stakeholders throughout the development process. Agile practices involve breaking projects into smaller increments or iterations, allowing teams to deliver functional components regularly and incorporate user feedback continuously. Agile teams are self-organized and often utilize structured methodologies that include systematic requirements gathering and testing after each iteration, ensuring higher quality control.

While the Build and Fix Model is suitable for small projects where speed is essential, it may struggle with larger or more complex applications due to its lack of structure. The Agile Model, however, is designed for projects with changing requirements and promotes ongoing customer involvement, making it more adaptable to evolving needs. Ultimately, the choice between these models depends on project size, complexity, and the importance of user feedback during development.