Prototyping and Quality Assurance

The Software Quality Assurance

Software quality is the degree to which a software product meets the gathered requirements. **Software quality assurance** is a set of activities that define and assess the adequacy of software process to provide evidence which establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended processes. This is based on the planning and implementation of a series of activities for ensuring quality in the software development process.

Many organizations implement software quality assurance to ensure that their software system product is of high quality. Here are some key reasons that drive software quality assurance:

- **Reputation** – Software developers and their organizations rely on this. Software bugs can have immediate impacts on clients or customers.
- **Limiting Technical Debt** – Poor quality software tends to be expensive to develop and maintain, which can negatively affect organizations or end up maintaining the software in the longer term. These costs are often referred to as "technical debt."
- **Software Certification** – The development and use of the software might require some form of certification, which can often require evidence of the application of various quality control and assessment measures.
- **Legality** – There may be overriding legal obligations that apply to organizations that use the software. As such, every practicable measure must be taken to demonstrate that the software system does not pose a risk to its users.
- **Ethical Codes of Practice** – In cases where a software system is not covered by software certification and legislation, and where its failure is not necessarily business or safety-critical, there can remain moral obligation to the users. This implies that software engineers should do whatever is possible to maximize the quality of their software and to prevent it from containing potentially harmful bugs.

Software Quality Assurance Through Prototyping

Software prototyping refers to building software application prototypes which displays the functionality of the product under development, but may not actually hold the exact logic of the original software. A prototype moves the quality assurance earlier and makes it easier. Prototyping software is used to achieve the following:

- Allow users to evaluate the software system design and how the software actually works.
- Clarify the functionalities of the software to both users and developers.
- Reduce cost and time relative to building software product and iterating the process.
- Reduce the need to recreate the software system due to some gaps on the implementation.

Models of Prototyping and Tools

Prototyping Paradigms

- **Throwaway Prototyping** – This is a relatively fast method of prototyping that focuses on employing prototypes to generate insights about the software design idea. The created prototypes on this paradigm are discarded after testing and after obtaining feedback. This is used in early design stages where the gathered requirements are still unclear.

- **Evolutionary Prototyping** – The prototypes on this paradigm are reused after the testing in a way that they are altered according to the test results and then reused in a new test cycle. This evolves to become the end product itself. This is usually used in late development stages where sufficient insides about principle design decisions exist and the prototype addresses technical and long-term use issues.

Prototype Fidelity and Prototyping Tools

Prototypes can be created in different fidelities. The fidelity of the prototype pertains to how it conveys the look and feel of the final product. It can range from low-fidelity to high-fidelity, which are explained further below.

- **Low-Fidelity Prototyping** – Its role is to check and test the visual appearance and user flows of a software. A throwaway prototype can be created in low-fidelity.

  These are some techniques to create low-fidelity prototypes:

  - **Paper-Based Prototyping** – This prototyping technique is used to create a prototype based on hand drawings that represent user interfaces of the software. This allows developers to explore ideas and refine designs quickly.

  - **Wireframing** – A wireframe is a visual representation of a product page that developers can use to arrange pages of user interfaces. Wireframes are used to create interactive prototypes by linking different wireframes together.

    Prototyping Tools for Wireframing

    There are various available tools made specifically for prototyping which are designed to help developers create wireframe prototypes for mobile applications, websites, and computer software. Some of the low-fidelity user interface (UI) wireframing tools that reproduce the experience of sketching an interface mockup using a computer are Balsamiq Wireframes, Axure, MockFlow, and Adobe XD.

- **High-Fidelity Prototyping** – High-fidelity prototypes are created when developers have a solid understanding of what they are going to build. They need to test it with the actual users. For example, evolutionary prototypes are created as high-fidelity prototypes. These prototypes are usually created in the form of programming codes and Integrated Development Environment (IDE) software.

REFERENCES:

Babich, N. (2017, November 29). *Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each* [Web log post]. Retrieved from https://theblog.adobe.com/prototyping-difference-low-fidelity-high-fidelity-prototypes-use/

Bahr, B. (2017). *Prototyping of user interfaces for mobile application*. Cham, Switzerland: Springer International Publishing AG.

Galin, D. (2018). *Software quality. Concepts and practice.* Retrieved from https://books.google.com.ph/books?id=5aNMDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Walkinshaw, N. (2017). *Software quality assurance. Consistency in the face of complexity and change.* Cham, Switzerland: Springer International Publishing AG.