Timothy Jelinek

5/5/2024

Part 1

The main strategic options for legacy system evolution are rewriting the whole system, continuing work in the same style, creating a new application over the old one, and partial refactoring. You would rewrite the entire system if it doesn’t work at all or if it does work, but it is so bad that the users abandon it. You would rewrite part of the system to allow the users to continue using the rest of the system while you update the parts of the system that don’t work. The advantages of rewriting the entire system include finding out what mistakes were made in the code and find ways to avoid them happening again, you have the ability to enjoy working on your own code, and you can reuse any good practicies that may be in place with the legacy system. The advantages of working in the same style of the legacy system include the ability for the system to continue working while you add new features and the customers can see improvements to the system right away at the beginning of the process of updating it. The advantages of creating a completely new application over the old one are allowing the legacy system to continue working while the new features are added, you’re able to take advantage of the positive aspects of the legacy system while also making new functionality, and the customer can immediately see the results of their investment as you write the new application over the old one. Finally, the advantages of partial refactoring include the users can still use the system while dealing with occasional updates and it is possible to plan updates and new features gradually instead of all at once.

Part 2

The main three types of software maintenance include corrective maintenance, perfective maintenance, and adaptive maintenance. It is sometimes difficult to distinguish between the three types of software maintenance because they all have to do with modifying the system. Corrective maintenance is focused on repairing defects that result in operation errors as well as other issues. This maintenance happens when customers experience software errors and defects that are found in released software during testing cycles that have led to operational failures. Perfective maintenance is the practice of adding new requirements to the software as it is becoming the perfect product so that it retains its value to the stakeholders. This maintenance happens when customers or stakeholders want to add new requirements to the software to make sure it is successful and works correctly while filling demands over a long course of time. Finally, adaptive maintenance is focused on modyifing an existing software after it has changed external interfaces, environments, platforms, operating systems, or other elements. This maintenance happens when external interfact definiations change, such as communications, protocalls, or external data file structures, if there are new government of business rrules, and if the hardware or software have new or updated versions. It is important to make sure your systems stay up to date and current with everchanging rules and regulations and to make your stakeholders happy. After researching these three main types, I can see how it is difficult to distingish between them as they all include modifications, which may be for similar reasons like making sure the user is content with how the software performs and that it does perform.

Sources:

Cprime Inc. (2023, February 20). *Modernization Strategies for legacy Systems: Choosing the right one for your project*. Cprime. https://www.cprime.com/resources/blog/modernization-strategies-for-legacy-systems-choosing-the-right-one-for-your-project/

Westfall, L. (2024, January 8). *Types of maintenance*. https://www.linkedin.com/pulse/types-maintenance-linda-westfall-rtoyc/