Refactoring and Uniqueization of Legacy Code

Hong Kong, Hong Kong S.A.R. Jan 10, 2023 (Issuewire.com**)** - *Refactoring legacy code helps us to ensure that a codebase remains maintainable and effective over time.*

What is Legacy code?

Legacy code refers to code that has been inherited from previous developers or software systems and is no longer actively maintained or supported. This type of code can be difficult to work with because it may not have documentation, may use outdated programming languages or frameworks, and may not adhere to current coding standards.

Legacy code can be a problem for organizations because it can be difficult and time-consuming to maintain and modify, and it may not be compatible with newer systems or technologies. It can also be a risk as it may contain security vulnerabilities or bugs that have not been addressed.

To manage legacy code, organizations may need to invest in refactoring or rewriting the code to make it up-to-date and easier to maintain. They may also need to implement processes to ensure that new code is properly documented and follows current best practices so that it does not become legacy code in the future.

Rewrite and refactoring legacy code

Rewriting legacy code involves creating a new version of the code from scratch while maintaining the same functionality as the original code. This can be a time-consuming and expensive process, but it can be necessary if the legacy code is no longer maintainable or if it is not compatible with new technologies or systems.

Refactoring legacy code, on the other hand, involves modifying the existing code to improve its structure and make it easier to understand and maintain, without changing its functionality. This can involve reorganizing the code, improving variable names and comments, and following current coding standards. Refactoring can be a more efficient and cost-effective way to improve legacy code, but it may still require a significant investment of time and resources.

When deciding whether to rewrite or refactor legacy code, it is important to consider the complexity of the code, the resources available, and the long-term needs of the organization. It may be necessary to rewrite code if it is too complex to refactor or if it is not feasible to maintain the legacy code.

Example

Original code:

def calculate_total(items):

total = 0

for item in items:

total += item.price

return total

Refactored code:

def calculate_total(items):

return sum(item.price for item in items)

In this example, the original code calculates the total price of a list of items by iterating through the list and adding up the price of each item. The refactored code uses a built-in function (sum()) to achieve the same result in a more concise and readable way.

Refactoring can involve a wide range of techniques, such as renaming variables and functions to be more descriptive, reorganizing code to improve its structure and readability, and removing unnecessary or redundant code. The goal of refactoring is to make the code easier to understand and maintain, without changing its functionality.

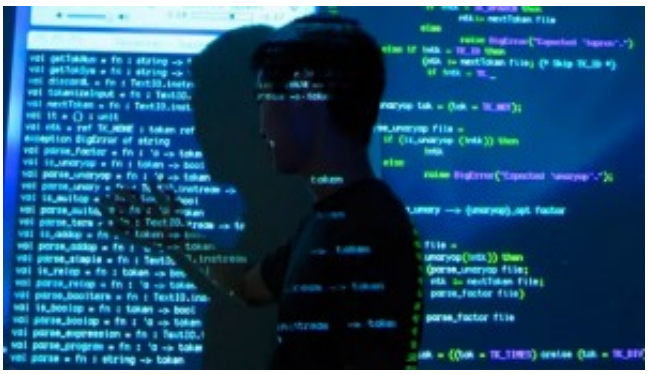
Improvement and uniqueness of legacy code

Refactoring can often be a more efficient and cost-effective way to improve legacy code, particularly if the code is still in use and needs to be maintained. Many developers use legacy as a base, reference

code to develop their new applications.

In this case, it may be necessary to uniquely code the application being developed. Mobile applications that are not unique may not pass the AppStore and Google Play check and get banned, and web applications created on template platforms may have problems with indexing in search engines.

For effective code refactoring and rewriting, you will benefit from the AppRefactoring tool, which automatically compares your code with legacy code and will help you write unique code for your application.



Media Contact

AppRefactoring

mkrtichyan.e@itcomp.org

Source : AppRefactoring

See on IssueWire