Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 12, Decembe 2024

Drafting Risks Mitigation Strategies with Defensive Programming

Mr.Rahul Chandrayan
(Research Scholar)

Abstract
Defensive programming is an programming art by which one can reduce the occurrence of errors in an application so as to prevent the application from crashing and security challenges. It core aspects of defensive programming is availability, safety and security. We may think of defensive programming brings over head and time consuming for its implementations. However in situation where there is an failure are occurs in programming due to non implementation of error handlers in such scenario defensive programming has been proved as a boon by preventing the application from crashing on other hand once can if we successfully implement the defensive programming we can easier to trace the errors and save time and efforts. The application of defensive programming techniques includes input validation, boundary checks, data validation, type checking, and error handling to guard against unexpected inputs, invalid states, and runtime errors etc.
We do not think of any thumb rule practices for implementing the defensive programming universally to any of the application however it depends upon lot of parameters how we have to implement it so that we can get cent present working application helps for tracking and tracing bugs / errors and capabilities to accommodate changes.

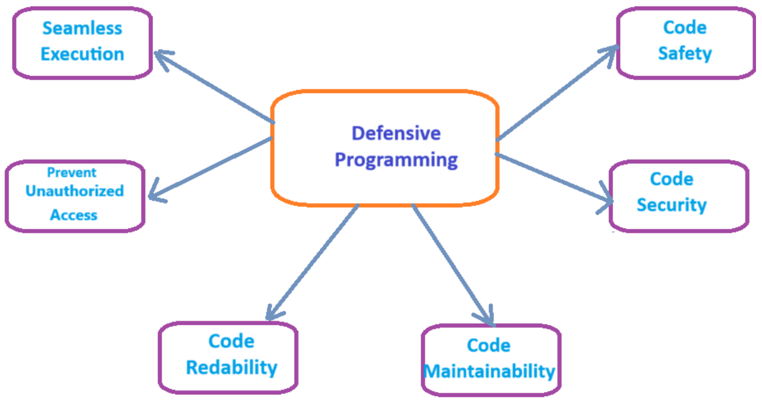
Keywords
Fail Fast, Coding Decisions, Detection Errors, Seamless execution

INTRODUCTION

A programming can be think of as an complex process since it has to lot of technologically implementation challenges includes business logic and algorithms logic etc. Moreover to be a successful application one has to ensure that the application should run without any failure or crashing for this we need to take lot of programming efforts such as error handling, validation, run time errors etc. which is a part of defensive programming. Thus, with defensive programming one can ensure clean code with improved readability, user input validation so as to avoid garbage in database, exception handling so that seamless running of application and testing of code which benefits correctly implementation of business logic or algorithms also it ensures functionality are correctly addressed.
Defensive programming we have a clear coding styles so that it can be easily readable, and maintainable. Also we have several testing approach to ensure error free application includes unit testing, manual and automated testing, code reviews, application of design patters helps to ease in complex programming efforts easily implementable. With successful testing and implementation we can prove the solution is more efficient, modular and increases its reusability. To have more code readability we can add the comments when and where required similarly we can have documentation such as SRS, FRS, Test Case etc. by which we can get idea about the architecture, business logic, functionality and testing conditions etc. Today we have several automated tools which can ease our task to achieve the defensive programming goals includes code formatters, linters, static analyzers, testing frameworks, code coverage tools, code review platforms, UML diagrams, design pattern libraries, design pattern generators, documentation generators, documentation platforms or documentation templates can be used to automate these processes.

Need of Defensive Programming

Good program design should reduce the need for defensive programming**.** Adopting defensive programming we can prevent the application from crashing, errors , bugs etc. Moreover defensive programming can be useful at place where random or end movement change in requirement are instantiated or expected hence to accommodate such changes and tackle challenge of modification at end movement . For Example modification of the structure of database entities for modifying the business logic and perform modified operations that is more difficult and error prone to do with the modification of queries to database.



Defensive Programming Benefits

Defensive programming implementation ensures cyber security by preventing user input attacks. These attacks take advantage of I/O vulnerabilities in web sites user can do malicious practices such as inserting – characters, strings, integers, scripts – by which the servers is more provoke to security risk.
We can find the use of defensive programming not only in Programming but also we can have it in algorithm, databases scripts, computer network etc.

Offensive Programming
Offensive programming is a branch of defensive programming that expressly departs from defensive principles when dealing with errors resulting from software bugs. Offensive programming work on the principal of Fail Fast which means to catch the bugs and crash early.
Offensive programming is same as that of normal programming however what is important here is once you identify the bugs immediately crash the program. This makes it different from defensive programming where recovery from the code is important , where as here we don't recover from bugs we simple crash the program / application.

Defensive Programming in OOD
In Object oriented design we get several benefits including code readability, quality and improving the reliability, usability, maintainability and at most enhance the user experience, further it benefited with ease in testing and debugging.
It also add values such that providing clear and helpful feedback , provide the status of application and handle errors in real time, and guide users to take the appropriate actions as per the error code etc.

Defensive Programming in Control Engineering
In Industrial Control System (ICS) we have PLCs, DCS, PACs programmable devices where we can implement defensive programming which provides not only resilient code that can prevent exploitations but also provide safety to industrial automation processes prevents damages to industrial assets and to human life.

Challenges in API Integrations
When we go for the API third party integration we have to integrating with third party code over a network where it becomes critical to deal hence we have to go with defensive programming approaches for providing the security. While programming defensive approach we have to handle timeouts, validate data protection, handle cashing and get the optimized performance.

PRINCIPLES OF DEFENSIVE PROGRAMMING

The principles on which the complete defensive programming based are:

Zero Users trust: While implementation the defensive programming, the programmer has to consider that the user can not always be true user or he/she may proceed in right desired steps or input the data as per he/she ask to do so, sometime the user tires to play with application and check the impact hence here the role of programmer to control the user to do so which is only possible if we think user is not sincere and try to act malicious with the application.

Authorized User Access:
The users are given permissions to access the user by creating their respective user for login and password so that only authorized user can access the application also the application has fixed timeouts if not used. Further the implementation to access with Https is more preferable rather than simple http access to application

Assigning Right Roles and Privileges: User has been control by providing them the right roles and access permission or privileges so that they can have limited control over the application and hence application get protected and used as per the requirement for which is made of.

Strategies of Defensive Programming
Defensive programming is an tact to handle application risks which could occurs at run time.
Thus defensive programming we try to mitigate the risks. Following are few question a programmer think of prior to begin the programming

- ✓ What and Where could it goes wrong (Program)?
- ✓ What is the impact if it goes wrong?
- ✓ How to adopt and implement the defensive strategies?
- ✓ What will be the cost of implementation or overhead for defensive programming?

Following are few listed Strategies that can be implemented in defensive programming:
1. Validation Issues: Validate inputs to methods and constructors to ensure they meet expected criteria.
2. Encapsulation: Encapsulate internal state and behavior of objects to control access to data and prevent unintended modifications.
3. Error Handling: Implement comprehensive error handling mechanisms to detect and handle unexpected conditions gracefully.
4. Preconditions and Post conditions: Implement the conditions as get the correct data from the users.
5. Fail-Fast Mechanisms: This help us to understand the errors and limitation and code accordingly to overcome from them.
6. Testing: Implement thorough unit tests to verify the correctness of individual components

Examples of Defensive Programming
In almost all the programming languages we have scope to implement the defensive programming languages for user input validation, exception handling, logging and tracking.

Input Validation:
Examples such as to prevent invalid, malicious, or unexpected inputs that can cause errors or security issues. In Java we have @NotNull and @Valid annotations to validate inputs to methods or classes. Similarly in Python we have the assert statement and raise keyword to check inputs to functions or modules.

Exception Handling: expels such as use of try, catch and finally block which helps to resume application without failures due to invalid data provided by user or data generated by application itself.
In C# or VB.Net we have try-catch-finally blocks. Similarly in Ruby we have begin-rescue-ensure blocks.

Logging : it helps to record and track the anomalies or errors occurs at runtime.
In JavaScript has the console.log() method and debugger statement. Similarly in PHP we have the error_log() function and var_dump() function.

Preventing Unauthorized Access:
User can be prevented from getting unauthorized access from accessing of OS, Other software's and Hardware this can be possible by defining the Roles, Securing tunnel communication/providing salt/Key authentications, allow for very specific serialization patterns of communications, block unwanted Ports and secure access to define ports etc.

Challenges of Defensive Programming
a) More Coding:
When we step towards defensive programming it simple means we need to write more code then actual desired logic and algorithm of the programming this is so to get benefits of defensive programming. On the contrary with increasing the code size, handling the code becomes challenging also it become and overhead and performance also get deteriorated of the application

b) Conflicting expectation:
To protect the application from runtime errors and error due to user inputs we sometime over use defensive programming which can raise conflicts and exceptions.

STEP UP TO WARDS DEFENSIVE PROGRAMMING

To implement the defensive programming every efforts that a programmer has to take so as to seamless running of the application as mentioned below:

a) Input Data:
It is one of the important parameter to be consider in defensive programming implementation, the input data at run time has expected to be provided by the user as per understanding the respective fields whereas due to lack of knowledge or maturity of user most of time the data input given by user is not expected hence special provision are to be added by the programmer such as validation, text messages so that user get clear understanding of what has been the clear input expected and all this a programmer can do by applying the defense mechanism practices.

b) Error Detection and Handling:
In an application there can be multiple errors such as runtime error, random error, logical error etc
Hence the programmer has to think of in details the mechanism to find the errors in the application by creating test cases and handle such error or remove the error from the application at earlier stage to avoid the cascade effect of errors.

c) Documentation and Code Comments
In those applications where more than one programmer is working, code comments plays an very important role such practices clarifies what the logic has been developed earlier programmer so that they can test or increase the code reusability.
Moreover while stating any project the organization should confirm the availability of technical, business logic, function and test documentation etc for smooth development. This document also named as SRS, FRS, Unit Case Document.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 12, Decembe 2024

Testing and Debugging in Defensive Programming
Testing an application using defensive programming is slightly different from testing a traditional one. However the testing approach remain the same i.e. we want to test whether the application performs safety operations when an unexpected inputs, invalid states, and runtime errors are provoked in the application, further the application perform seamlessly without being crash or goes into endless loop.
Whereas debugging in defensive programming is hard, the application behavior with invalid inputs, runtime errors, error logging, seamless working etc. has been checked / watch and if found so as to fix such errors in real time.

CONCLUSION

This paper describes the importance of defensive programming as a mechanism acting as a base for optimized application development, in this approach a programmer take efforts so as to ensure the smooth running of an application. An approach of adopting the defensive programming enables a programmer to think 360 degree before actual starting the coding. Overall the value what an application get by adopting the defensive programming approach is highly beneficial ensuring input validation, boundary checks, data validation, type checking, and error handling to guard against unexpected inputs, invalid states, and runtime errors for smooth running of the application with high performance.

REFERENCES

[1] Divya Rishi Sahu, Deepak Singh Tomar , Defensive Programming to Reduce PHP Vulnerabilities, International Journal of Advances in Computer Networks and Its Security– IJCNS Volume 4: Issue 2
[2] Mukesh Kumar Gupta, M. C. Govil, and Girdhari Singh. 2014. Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey. In International Conference on Recent Advances and Innovations in Engineering (ICRAIE). IEEE, 1–5. DOI: http://dx.doi.org/10.1109/ICRAIE.2014.6909173
[3] Jun Zhu, Jing Xie, Heather Richter Lipford , Bill and Chu. 2014. Supporting secure programming web applications through interactive static analysis. Journal of Advanced Research 50, 1 (July 2014), 449– 462. DOI: http://dx.doi.org/10.1016/j.jare.2013.11.006
[4] Hunt, A. and Thomas, D. (1999). The Pragmatic Programmer: From Journeyman to Master. Reading, MA: Addison-Wesley
[5] Carpenter, A.L. (2011). Job Security: Using the SAS® Macro Language to Full Advantage. Retrieved from http://www.lexjansen.com/pharmasug/2005/technicaltechniques/tt05.pdf.