**UML Design Modeling**

Rodrigo Cruz

10/01/2023

CST 499 Capstone for Computer Software and Technology

Instructor Charmelia Butler

Design metrics in software engineering can be described as metrics used to evaluate and describe software in different ways. Metrics are objective and are tracked regularly in order to determine any software needs or updates as these also measure the quality of the design. There are a variety of design metrics that can be used to measure software build quality. These metrics can be used to evaluate the high-quality design in different ways such as software testing, source code analysis, and analyzing the structural design and complexity.

As previously mentioned there are different types of metrics used to evaluate different parts of a software. Traditional design metrics include Halstead design metric which is one of the earliest software metrics used to analyze program source code, McCabe’s Cyclomatic Complexity which measures the structural design complexity within the program, and Henry-Kafura Information Flow which measures intermodular flow. New design metrics include Glass and Card design metrics which suggest three design complexities that have to be measured which include structural complexity which depends on fan-out models, Data complexity which is the complexity within the interface of internal module, and system complexity which is a combination of structural and data complexity.

Many of these metrics can relate to one another in many ways as these measure similar things in different ways as they all serve the same purpose. Cohesion identifies a degree at which elements in one unit are related to one another. Each metric being measured can be dependent on another metric as it can change the result, for example a program source code depending on how it is coded can determine or change the complexity of the program both of these being metrics that are usually analyzed. As previously mentioned system complexity is dependent on structural and data complexity.

Concepts of completeness and consistency refer to concepts that are essential for software requirements and engineering and are related to the quality and reliability of the software. Completeness refers to the completeness of the software in terms of it having all the required functions in order to operate the way it was designed to operate this is important as it avoids missed requirements and reduces errors. Consistency on the other hand refers to how consistent the software design is in terms of all functions working smoothly without conflicts. This also minimizes errors and facilitates the design.

To conclude software design metrics are important tools that are used to determine the quality of the software and should be used continuously throughout the life of the software.

References:

Zhao, M., Wohlin, C., Ohlsson, N., & Xie, M. (1998). A comparison between software design and code metrics for the prediction of software fault content. *Information & Software Technology*, *40*(14), 801–809. https://doi.org/10.1016/s0950-5849(98)00098-6

Aggarwal, K. (2007). *Software Design Metrics for Object-Oriented Software*. <https://www.semanticscholar.org/paper/Software-Design-Metrics-for-Object-Oriented-Aggarwal-Singh/1f5f0e82072bc5461eeef0cdba72e7f32b519345>

**UML Diagrams**

A diagram of a company

Description automatically generated

A diagram of a company

Description automatically generated

A diagram of a sequence diagram

Description automatically generated

A diagram of a class

Description automatically generated