**Week 2 Discussion Forum 2 – Design Metrics**

Vicki Kelm

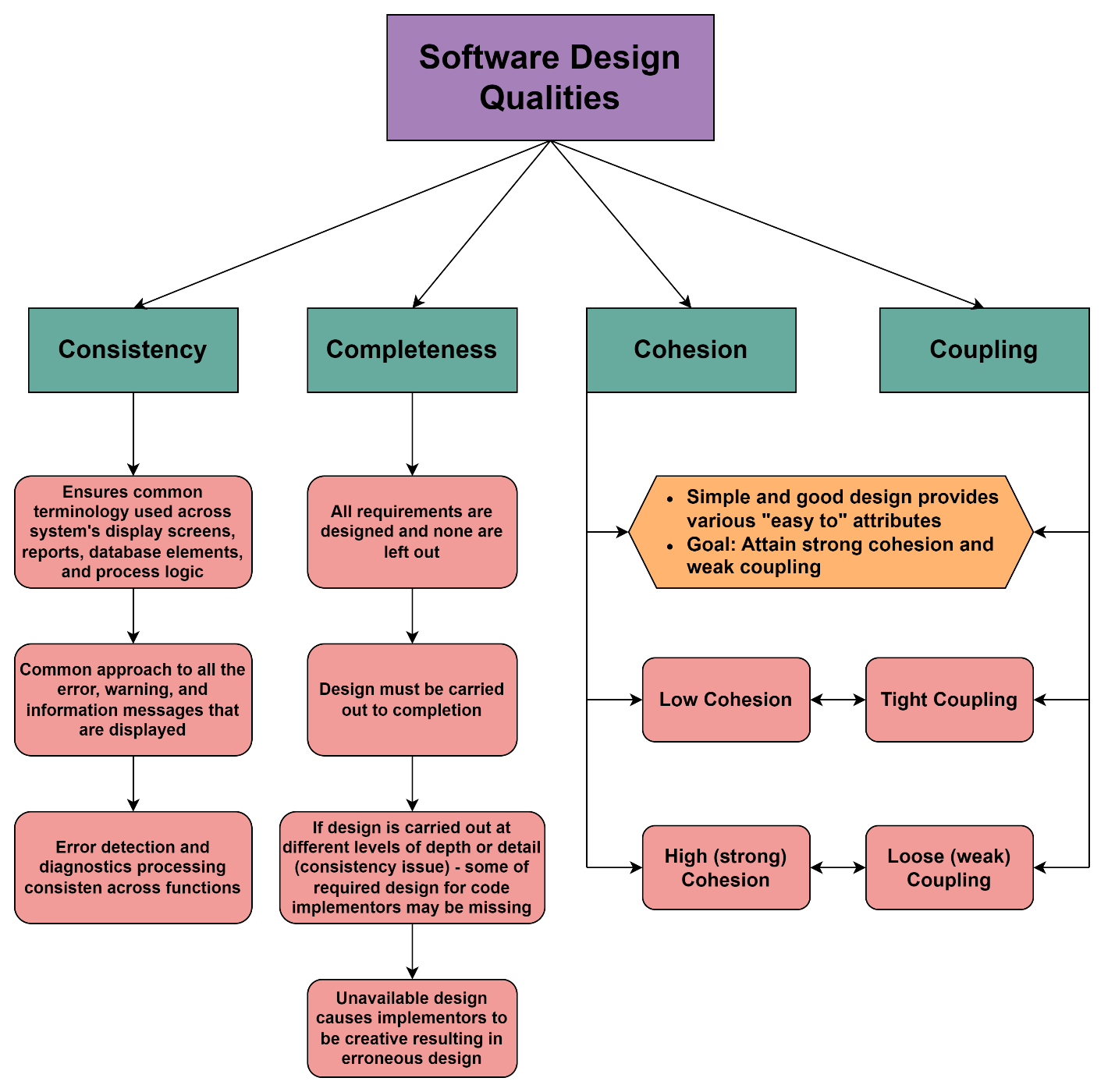
The University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Dr. Amjad Alkilani

December 28, 2023

**Concept Map**



**Week 2 Discussion Forum 2 – Design Metrics**

“A software metric is a measure of software characteristics that are quantifiable or countable” (Stackify, 2017). This paper will describe new and traditional metrics, explain how these metrics can be used to evaluate the quality of the design, discuss how the metrics relate to one another, and evaluate the concepts of completeness and consistency related to software requirements.

**Describe new and traditional design metrics**

Earlier metrics took different approaches to analyze source code for complexity as a measure of quality. The Halstead Complexity Metric measures the number of operators and operands of the source code and the sum of the operators and operands. McCabe's Cyclomatic Complexity uses control flow diagrams to determine the number of edges, nodes, and connected components of the code for measuring its complexity, and thus its quality. Henry-Kafura Information Flow analyzes the complexity of the source code through the input or output data, or the fan-in and fan-out, of the modules as a measure of its quality.

The newer design metrics revolved around the concept of cohesion and coupling. The concepts do not vary much from the complexity measurements of the earlier metrics. Similar to Halstead and McCabe, cohesion addresses intramodular characteristics, while coupling addresses intermodular characteristics just as Henry-Kafura and Card and Glass did (Tsui, Karam, & Bernal, 2018).

**Explain how these metrics can be used to evaluate the quality of the design**

Both Halstead and McCabe offer intramodular structural metrics. The Halstead metric was mainly developed to analyze program source code. It unfortunately has been criticized for contributing limited value in the analysis of program complexity or any design characteristics. McCabe observed that program quality was directly related to the complexity of control flow in the detail of the design. The Henry-Kafura metric is also a structural metric, and provides measure of an intermodular metric. Card and Glass also utilize a fan-in/fan-out approach to measuring complexity that also considers the data being passed in both the program-level and inter-program-level (Tsui, Karam, & Bernal, 2018).

**Discuss how the metrics relate to one another**

All of these metrics relate to one another in that they are measures of software quality. They are the desired results that are only achieved with a good design. They all have a common thread in that they are “easy to” properties in the notion of simplicity. Here is where a large and complex problem is simplified by separating it into smaller pieces to be solved in increments. When thinking about cohesion in that it addresses the intramodule characteristics similar to the Halstead and McCabe metrics, and coupling addresses intermodule characteristics similar to the Henry-Kafura fan-in and fan-out information flow measurements and the system complexity measure of Card and Glass (Tsui, Karam, & Bernal, 2018).

**Evaluate the concepts of completeness and consistency related to software requirements**

Completeness and consistency are both important characteristics that carry over from the requirements. “Completeness is the presence of all event handlers or actions for condition guards of all events. Consistency is the relations among actions that are not conflict/contradict to those of condition guards and their allied events” (Yu, Su, Luo & Su, 2008). Consistency ensures that common terminology is utilized for a system’s display screens, reporting, database elements, and also process logic. The development of requirements in their entirety ensures completeness in that no requirements are left out. If there is inconsistency into how the design is carried out, then there will be missing depth or detail and design requirements will not be present. Both of these characteristics are important and must be focused on during design reviews or inspections.

**Conclusion**

“Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and other functions of management” (javaTpoint, n.d.). This paper has described new and traditional metrics, explained how these metrics can be used to evaluate the quality of the design, discussed how the metrics relate to one another, and evaluated the concepts of completeness and consistency related to software requirements.

**References**

Alexandra. (2017, September 16). *What Are Software Metrics and How Can You Track Them*? <https://stackify.com/track-software-metrics/>

javaTpoint. (n.d.). *Software Metrics*. <https://www.javatpoint.com/software-engineering-software-metrics>

Tsui, F., Karam, O., & Bernal, B. (2018). [Essentials of software engineering](https://uagc.instructure.com/courses/126521/modules/items/6439323)(4th ed.). Jones & Bartlett Learning.