# **APPENDIX C**

# **USER’S EVALUATION FORM**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ EVALUATION USING McCall’s SOFTWARE EVALUATION CRITERIA FOR SOFTWARE QUALITY.

Instruction: Please put check mark in the box that closely represents your choice. Please use the rating provided below.

|  |  |
| --- | --- |
| **Numerical Rating** | **Descriptive Interpretation** |
| 5 | Very Good |
| 4 | Good |
| 3 | Average |
| 2 | Fair |
| 1 | Poor |

**Criteria***.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **5** | **4** | **3** | **2** | **1** |
| **Auditability**. The ease with which conformance to standards can be checked. |  |  |  |  |  |
| **Accuracy**. The precision of computation and control |  |  |  |  |  |
| **Completeness**. The degree to which full implementation of the required function has been achieved. |  |  |  |  |  |
| **Communication Commonality**. The degree to which interfaces and protocols are understood. |  |  |  |  |  |
| **Conciseness**. The compactness of the program in terms of lines and code. |  |  |  |  |  |
| **Consistency**. The use of uniform design and documentation techniques throughout the software development project. |  |  |  |  |  |
| **Observability**. The process of streaming the software components can be easily identified and understand. |  |  |  |  |  |
| **Security**. The availability of mechanisms that control or protect programs and data. |  |  |  |  |  |
| **Self-Documentation**. The degree to which the source code provides meaningful documentation. |  |  |  |  |  |
| **Simplicity**. The degree to which the program/software can be understood without difficulty. |  |  |  |  |  |
| **Software System Independency**. The degree to which the program/software is independent of non-standard programming language features, operating system characteristics, and other environmental concern. |  |  |  |  |  |
| **Traceability**. The ability to trace the design representation or actual program component back to requirement. |  |  |  |  |  |
| **Training**. The degree to which the software assists in enabling new users to apply the system. |  |  |  |  |  |
| **Controllability**. The system/software can be easily controlled and manipulated in terms of execution, program structure, and design. |  |  |  |  |  |
| **Data Commonality**. The use of standard data structures and types throughout the program/software. |  |  |  |  |  |
| **Decomposability**. The software is built from series of modules and can be tested independently. |  |  |  |  |  |
| **Error Tolerance**. The damage that occurs when a program/software encounters an error. |  |  |  |  |  |
| **Execution Efficiency**. The run-time performance of a program/software. |  |  |  |  |  |
| **Expendability**. The degree to which architectural, data or procedural design can be extended. |  |  |  |  |  |
| **Generality**. The breadth of potential application of program components. |  |  |  |  |  |
| **Hardware Independence**. The degree to which the software is decoupled from the hardware on which it operated. |  |  |  |  |  |
| **Instrumentation**. The degree to which the program/software monitors its own operation and identify errors that occur. |  |  |  |  |  |
| **Modularity**. The functional independence of program/software components. |  |  |  |  |  |