**Rochak Kunwar**

**Product Characteristics Evaluation**

**Definition of Product characteristics:**

**D** -. This describes the overall complexity of the product. the degree to which a system or component’s design or implementation is difficult to understand and verify. It is not related to the size of the software. Higher the score lowers the complexity.

**DISP** - It is related to the interface. According to the reading we have 4 tiers of DISP categorization, they are simple, user-friendly, interactive and complex. DISP is also not related to the size of the software interface requirement. Higher the score more complex the interface.

**HOST:** It is related to the re-hosting of software.  general example: porting of the mobile number from one carrier to another. More precise example: when two company merge their software might be written in a different language, now they have to come in a common platform that a major change in host. Higher the number complex the rehosting.

**MEMC:** It is a memory constraint. Software using deep neural network requires big memory in CPU while software like video sharing platform would not require big memory from CPU. CPU memory is very expensive so, complex the memory management of the software is higher the score.

**RELY:**It measures the degree of reliability of a software. A software giving service to a government organization needs to very reliable than a non-profit organization. But reliability not similar to security it’s a different factor. Higher the score, higher the reliability of the system.

**RTIM:** real-time computing. It is mostly done at a low level and is very difficult to implement. IOT is the best example of RTIM. The higher the score; higher the amount of implementation of RTIM in the system.

**RVOL:** The cost factor that comes with the change in the requirement of stakeholder. It can be during the development process as well as after the development is over. Facebook itself has gone through lots of updates and changes since 2005. The more frequent and bigger the change is higher the score of RVOL.

**SERC:** It is the degree of security in software. It might look something similar to RELY but it is more about the security of the system. It is the degree that measures the security of software in order to get customers or fulfill the requirement. Higher the score, higher the security requirement for software.

Scenario 1: Mobile Game

D- 18 (I feel there is some math logic needs to be done for graphics but not like building a whole new system on top of the operating system)

DISP- 1.11 (The game needs to be interactive to know the input of the user. Especially with the app it needs to read some sensor in app)

HOST – 1.00 ( We can run the mobile app in our developing environment with common platform)

MEMC : 1.04 (Some threading and multiprocessing are necessary)

RELY: 1.31 ( Commercial software with essential documentation)

RTIM: 1.00 (The only real time I can see of is some sensor in phone)

RVOL: 1.00 (Changes will be small for every update in these types of software, usually the product is familiar on different changes, and the requirement do not change a lot)

SERC: 1.03 (I don’t see the need for very secured system other than making sure their data and money is safe)

**Scenario 2: E-commerce**

D- 21 (the only complex thing I see is database as an internal data storage)

DISP- 1.05 (It Is simple website so the more focused should be in making its user-friendly)

HOST – 1.63 (I feel when creating web application, we develop them locally while our database can be in some different server and can be hosted from different server)

MEMC : 1.00 (No multiprocessing required, an async task in single thread works. And there is enough memory for our database)

RELY: 1.16 (Commercial software)

RTIM: 1.00 (There is minimal real-time operation; mostly querying data)

RVOL: 1.00 (Requirement do not change a lot in these types of software, but some non-critical redirections can happen)

SERC: 1.05 (I don’t see the need for very secured system other than making sure their data and money is safe)

**Scenario 3: Life Support**

D- 4 (I feel like the software for life support needs to be real time and must read different signals coming from different devices in run time)

DISP- 1.11 (I would say it needs to be interactive since its output depends on the input from sensors and secondary interface unit [1].

HOST – 1.17 (Minor language and minor system change)

MEMC : 1.04 (Some run time memory efficiency required)

RELY: 1.77 ( Very high reliability required)

RTIM: 1.27 ( I feel everything is real-time operation)

RVOL: 1.00 (Requirement do not change a lot in these types of software, but some non-critical redirections can happen)

SERC: 1.08 (There is no money in this software so it will need less secured system)

**Score:** When we analyze the tables in section 10 we can see that all the characteristics difficulty is directly proportional to the score except the complexity. Complexity is inversely proportional to the score it gets. Also, the complexity score is way different than the rest of the characteristics score. So, we would group the characteristics into two groups: Complexity and rest. If we add up all the variables except the complexity value than we can get a linear equation that can somewhat define the characteristics of software in general.

Hypothesis: If the overall score is higher than it means the software is complex, big, safe, more time consuming to build, needs a different level of skills, etc.

Here is our linear model and score for each software:

**Mobile Game:** 1.11 + 1.0 + 1.04 + 1.31 + 1.0 + 1.0 + 1.03 = 7.49 D = 18

**E-Commerce:** 1.05 + 1.63 + 1.0 + 1.0 + 1.0 + 1.16 + 1.05 = 7.89 D = 21

**Life-Support:** 1.11 + 1.17 + 1.04 + 1.77 + 1.27 + 1.0 + 1.08 = 8.44 D = 4

We can see that the overall score is inversely proportional to the complexity. But complexity as one characteristic does not describe the needs of rest of the characteristics. From the above result, we can say that: when the overall score is higher than it means the software is complex.