A Quality Computation Model for Software Architecture

Gurbinder Singh Brar
Department of Information Technology,
Adesh Institute of Engineering and Technology,
Faridkot

Rupinder Pal Singh
Department of Information Technology,
Adesh Institute of Engineering and Technology,
Faridkot

ABSTRACT
Although general quality models are available, it is possible to construct a custom quality model. The set of metrics obtained from such quality model can then be used to evaluate candidates, whether designs, architectures or systems using a quality computation model. This paper adapts a quality computation model for this purpose, and discusses an example to demonstrate the same.

Keywords
Quality models, quality characteristics, quality computation models.

1. INTRODUCTION
ISO/IEC 9126-1: 2001 quality model defines "quality" as "a set of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [1]. A quality attribute is a property of a system by which some stakeholders will judge its quality. Quality attribute requirements, such as those for performance, security, modifiability, reliability, and usability, have a significant influence on the software architecture of a system. A system's functional requirements play an important role in the definition of the initial architecture. On the other hand, the quality requirements have to be "balanced" during the subsequent design process. Quality characteristics of software architecture need to be measured so that software designers are able to evaluate candidate architectures, and evaluate an existing or legacy architecture for upgrade to achieve some performance goal. More specifically,

a) Evaluate the candidate architecture in a stand-alone fashion to check if it meets the functional and non-functional requirements expected of it;

b) Evaluate the candidate in comparison to the software architecture of an existing, i.e. already deployed, systems;

c) Evaluate an architecture style in comparison to other competing, candidate architectures for the system under development;

d) Evaluate various possible designs for a system based on a selected style e.g. SOA.

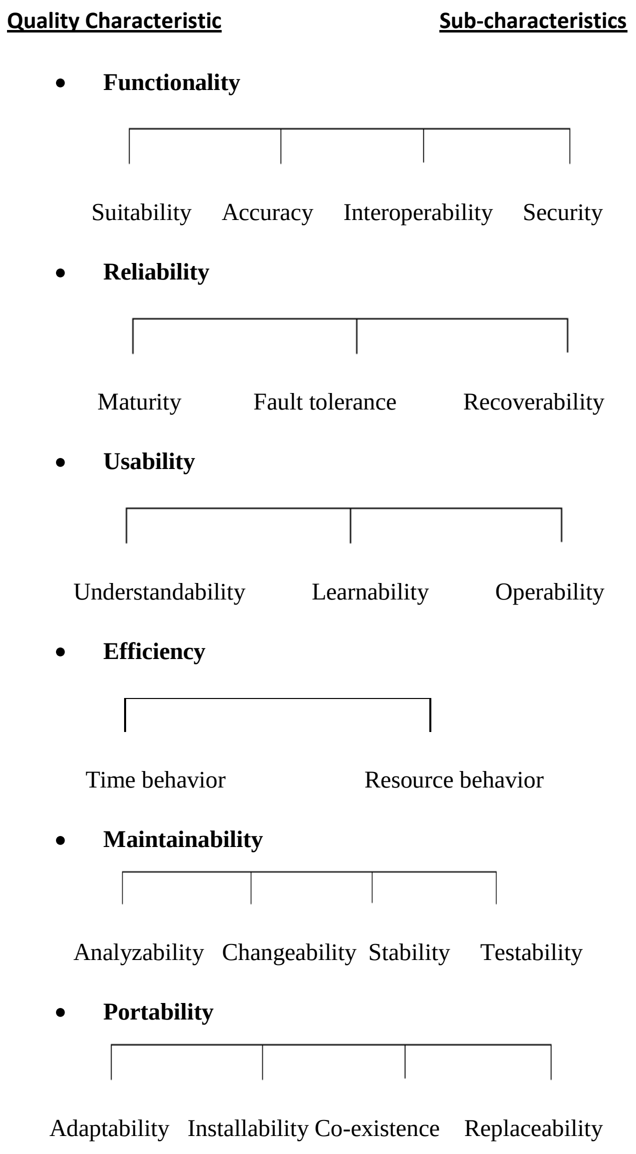

Figure 1. ISO 9126-1 quality model

The (1) decomposition of quality into characteristics, and the (2) definition of metrics for characteristics can be considered as the two basic elements of any quality model. These two steps are sufficient if the model is used for characterizing quality [3].

Although general quality models, e.g. ISO 9126-1 (see Figure 1), are available, it is possible to construct a custom quality model. For example, [2] describe how ISO/ IEC quality model can be customized to specific software domain using a six-step methodology. Fig. 2 illustrates their six-step methodology. The set of metrics so obtained can then be used to evaluate candidate designs using a quality computation model. We present below such quality computation model, with a restriction that all metrics employed should have numeric values.

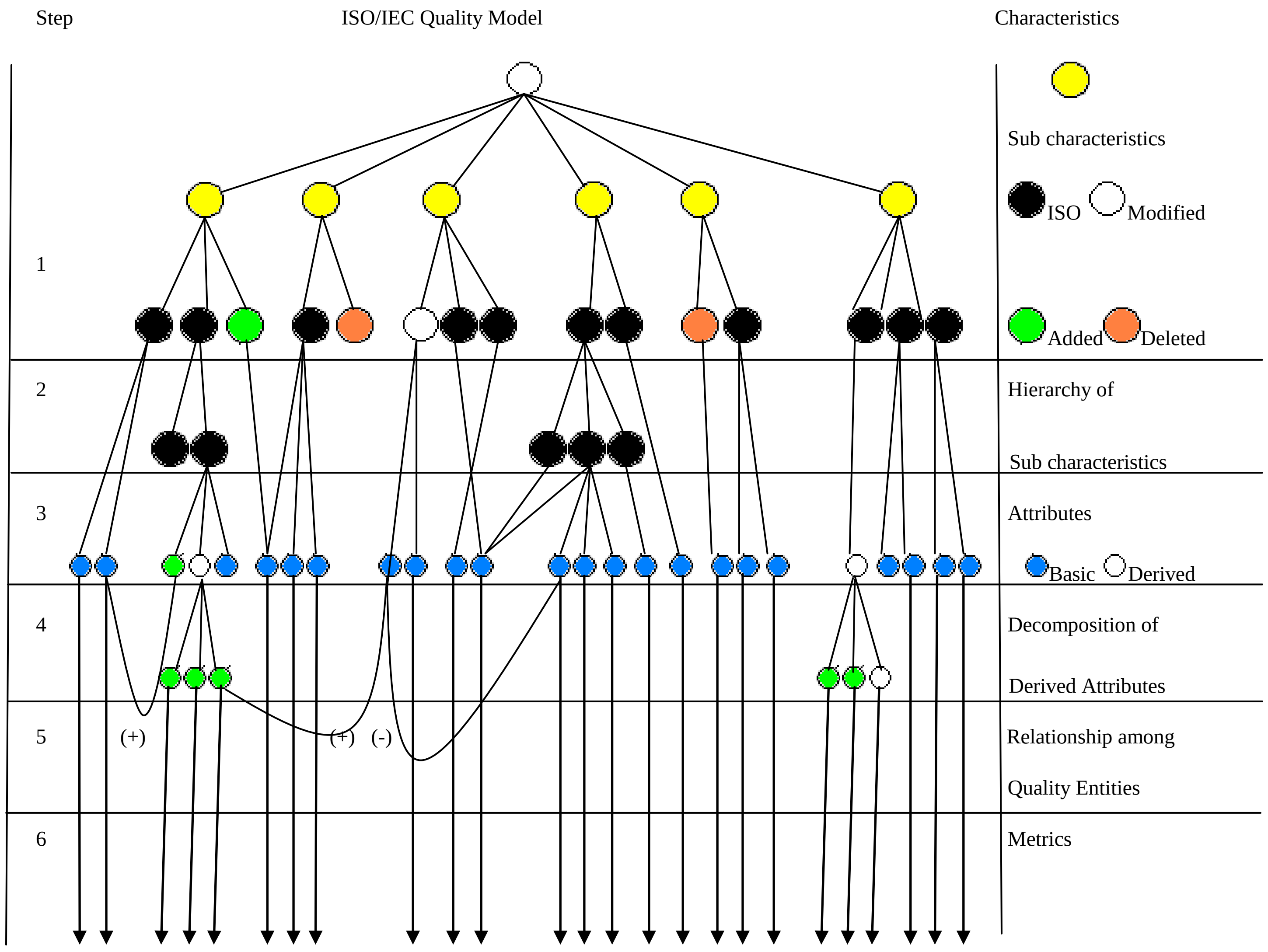

Fig. 2. The Six-step methodology to create a custom quality model

2. LNL QUALITY COMPUTATION MODEL

Y. Liu, A.H.H. Ngu and L.Zeng have proposed in [4] a QoS computation model for dynamic web service selection. Specifically, they argue that QoS is a broad concept that can encompass a number of context-dependent, domain-specific non-functional properties. Therefore, they argue that a quality computation model should be extensible, that is, it should include both generic and context-dependent, domain-specific criteria. That is, any new domain-specific criteria can be added and used to evaluate the QoS of competing designs. Further, they argue, and as expectedly, that different stakeholders may have different preferences or requirements on QoS. A QoS model should provide means for users to accurately express their preferences without complex coding of user profiles.

2.1 The Extensible Quality Computation Model

Assuming there is a set of *n* candidate software designs, *A1*, *A2*, *A3*…… And suppose we select *m* quality characteristics to evaluate these designs, we can obtain the following matrix.

$$Q=\begin{bmatrix} q_{1,1} & q_{1,2} \cdots\cdots\cdots & q_{1,m} \\ q_{2,1} & q_{2,2} \cdots\cdots\cdots & q_{2,m} \\ & & \vdots \\ q_{n,1} & q_{n,2} \cdots\cdots\cdots & q_{n,m} \end{bmatrix}$$

In order to rank the *n* designs, the matrix *Q* needs to be normalized. The purposes of normalization are: 1) to allow for a uniform measurement of design qualities independent of units.2) to provide a uniform index to represent design qualities.3) to allow setting a threshold regarding the qualities.

The number of normalizations performed depends on how the quality criteria are grouped. The second normalization is used to provide uniform representation of a group of quality criteria (e.g. group maintainability can be of three criteria, complexity, cohesion and coupling.) and set threshold to a group of quality criteria.

2.2 First Normalization

Before normalizing matrix *Q*, we need to define two arrays. The first array is $N = \{n_1, n_2,.. n_j \cdots n_m\}$ with $1 \leq j \leq m$. The value of n_j can be 0 or 1. n_j=1 is for the case where the increase of $q_{i,j}$ benefits the stakeholders while n_j=0 is for the case where the decrease of $q_{i,j}$ benefits the stakeholders. The second array is $C= \{c_1, c_2,...c_j....c_m\}$. Here c_j is a constant which sets the maximum normalized value. Each element in matrix *Q* will be normalized using the following equation. Here $(1/n) \sum q_{i,j}$ is the average value of an attribute q_j over *n* designs. The summation $\sum$ is over *i* = 1 *to n.*

$$V_{i,j}=\begin{cases} q_{i,j}/(1/n)\sum q_{i,j} & \text{if } ((1/n)\sum q_{i,j} \neq 0 \text{ and } \\ & q_{i,j}/(1/n)\sum q_{i,j} < c_j \text{ and } n_j=1 \\ c_j & \text{if } (1/n)\sum q_{i,j} = 0 \text{ and } \\ & n_j=1 \text{ or } q_{i,j}/(1/n)\sum q_{i,j} \geq c_j \\ (1/n)\sum q_{i,j} \;/\; q_{i,j} & \text{if } q_{i,j} \neq 0 \text{ and } \\ & (1/n)\sum q_{i,j} / q_{i,j} < c_j \text{ and } n_j=0 \\ c_j & \text{if } q_{i,j}=0 \text{ and } n_j=0 \\ & \text{or } (1/n)\sum q_{i,j} / q_{i,j} \geq c_j \end{cases}$$

Applying this equation to *Q*, we get matrix *Q'* as follows:

$$Q'=\begin{bmatrix} v_{1,1} & v_{1,2} \cdots\cdots\cdots & v_{1,m} \\ v_{2,1} & v_{2,2} \cdots\cdots\cdots & v_{2,m} \\ & & \vdots \\ v_{n,1} & v_{n,2} \cdots\cdots\cdots & v_{n,m} \end{bmatrix}$$

2.3 Second Normalization

In this quality model, quality attributes can also be represented as a group and manipulated as group. Each group can contain multiple criteria. Matrix *D* is used to define the relationship between quality criteria and quality groups. Columns represent *l* quality groups. Rows represent total of *m* quality criteria. If a quality criteria *i* is present in *j*th quality group, $d_{i,j}$ =1 else it is set to 0.

$$D=\begin{bmatrix} d_{1,1} & d_{1,2} \cdots\cdots\cdots & d_{1,l} \\ d_{2,1} & d_{2,2} \cdots\cdots\cdots & d_{2,l} \\ & & \vdots \\ d_{m,1} & d_{m,2} \cdots\cdots\cdots & d_{m,l} \end{bmatrix}$$

Applying *D* to *Q'*, we have:

$$G = Q' * D = \begin{bmatrix} g_{1,1} & g_{1,2} \cdots\cdots\cdots & g_{1,l} \\ g_{2,1} & g_{2,2} \cdots\cdots\cdots & g_{2,l} \\ & & \vdots \\ g_{n,1} & g_{n,2} \cdots\cdots\cdots & g_{n,l} \end{bmatrix}$$

Now, to normalize matrix *G*, two arrays are needed. In the first array $F= \{f_1, f_2,..f_n\}$, f_j is a weight for group. This is used to express user's preferences over *jth* group. In the second array $T= \{t_1, t_2,..t_n\}$, t_j is a constant which sets the maximum normalized value for the group *j*. Each element in *G* will be normalized using the following equation. Here $(1/n) \sum g_{i,j}$ is the average value of group g_j over *n* designs. The summation $\sum$ is *over i*=1 *to n.*

$$h_{i,j}=\begin{cases} g_{i,j}/(1/n)\sum g_{i,j} & \text{if } (1/n)\sum g_{i,j} \neq 0 \text{ and } \\ & g_{i,j}/(1/n)\sum g_{i,j} < t_j \\ t_j & \text{if } (1/n)\sum g_{i,j} = 0 \\ & \text{or } g_{i,j}/(1/n)\sum g_{i,j} \geq t_j \end{cases}$$

Applying the above equation to *G*, we get *G'* as follows:

$$G' = \begin{bmatrix} h_{1,1} & h_{1,2} \dots\dots\dots & h_{1,l} \\ h_{2,1} & h_{2,2} \dots\dots\dots & h_{2,l} \\ & & \vdots \\ h_{n,1} & h_{n,2} \dots\dots\dots & h_{n,l} \end{bmatrix}$$

Finally, we compute quality for all designs A_1, A_2.....as

$$Q(A) = \quad G' * F$$

2.4 An Example

Narasimhan, Parthasarathy and Das demonstrate in [5] the evaluation of candidate component-based softwares by comparing values for a set of metrics. Their comparison is reproduced in Fig. 3. Using one-to-one comparison, they argue that the softwares jGrasp and Junit need to be redesigned. We illustrate below that a similar conclusion is reached using the quality computation model discussed in the Section 2.3.

Table 1. Metric values for candidate component-based softwares.

Software / Metrics	Junit	Element	Mouse Gestures	Idap	JCIFS	jGrasp
CPD	13.375	19	4.5	6.25	0.5	70.7
CID	61	6	11	114	70	204
CIID(Ce)	315	6	10	89	51	159
COID(Ca)	91	1	1	25	19	45
CAID(CID/8)	7.625	6	5.5	7.125	8.75	11.34
CRIT Inheritance	93	19	8	91	4	1142
CRIT size	0	0	0	0	0	1
AC(=CID)	61	6	11	114	70	204
NOC	16	4	0	22	18	1
LCOM	0.91	0.855	0.778	0.627	0.753	0
DIT	6	4	6	8	7	1
WMC	822	407	46	763	539	37

$$Q = \begin{pmatrix} 70.7 & 204 & 159 & 45 & 11.34 & 1142 & 1 & 201 & 1 & 0 & 1 & 37 \\ 0.5 & 70 & 51 & 19 & 8.75 & 4 & 0 & 70 & 18 & 0.753 & 7 & 539 \\ 6.25 & 114 & 89 & 25 & 7.125 & 91 & 0 & 114 & 22 & 0.627 & 8 & 763 \\ 4.5 & 11 & 10 & 1 & 5.5 & 8 & 0 & 11 & 0 & 0.778 & 6 & 46 \\ 19 & 6 & 6 & 1 & 6 & 19 & 0 & 6 & 4 & 0.855 & 4 & 407 \\ 13.375 & 61 & 315 & 91 & 7.625 & 93 & 0 & 61 & 16 & 0.91 & 6 & 822 \end{pmatrix}$$

$$n = 6 \quad m = 12$$

$$N = \{n_1, n_2, n_3, \ldots\ldots, n_{12}\} = \{0,0,0,0,0,1,0,0,1,1,1,0\}$$

$$c = \{100, 250, 350, 100, 15, 1500, 1, 250, 25, 2, 10, 100\}$$

$$V = \begin{pmatrix} 0.269 & 0.38 & 0.66 & 0.674 & 0.681 & 5.05 & 0.17 & 0.38 & 0.09 & 0 & 0.19 & 11.77 \\ 38.1 & 1.11 & 2.06 & 1.6 & 0.882 & 0.02 & 1 & 1.11 & 1.77 & 1.16 & 1.3 & 0.80 \\ 3.05 & 0.68 & 1.18 & 1.21 & 1.083 & 0.4 & 1 & 0.68 & 2.16 & 0.96 & 1.5 & 0.57 \\ 4.23 & 7.06 & 10.5 & 30.33 & 1.404 & 0.03 & 1 & 7.06 & 0 & 1.19 & 1.12 & 9.47 \\ 1 & 12.94 & 17.5 & 30.33 & 1.287 & 0.08 & 1 & 12.94 & 2.54 & 1.31 & 0.75 & 1.07 \\ 1.42 & 1.27 & 0.33 & 0.33 & 1.012 & 0.41 & 1 & 1.27 & 1.57 & 1.4 & 1.12 & 0.53 \end{pmatrix} = \begin{pmatrix} 20.323 \\ 50.925 \\ 14.484 \\ 73.414 \\ 80.617 \\ 11.68 \end{pmatrix}$$

The normalized overall scores for jGrasp, JCIFS, Idap, Mouse Gestures, Element and Junit softwares are 20.232, 50.925, 14.484, 73.414, 80.617 and 11.68 respectively. The scores of jGrasp and Junit are significantly lower than those of all other software except in the case of Junit whose score is little higher than that of ldap. The inference is that jGrasp and Junit need quality improvements. This inference almost coincides with the one drawn by Narasimhan, Parthasarathy and Das in their paper [5].

3. CONCLUSION

A set of guidelines to develop a custom quality model for a specific domain has been discussed. The metrics obtained as end-result of applying such a quality model to competing candidates, whether designs, architectures or systems can then be converted to scores for every candidate using a quality computation model. A quality computation model has been adapted. The same has been applied to a previously published case study, and the results obtained thereof are encouraging.

4. REFERENCES

[1] Chastek, G., and Ferguson, R., "Toward Measures for Software Architecture," March 2006, Technical Note CMU/SEI-2006-TN-013, http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tn013.pdf**.**

[2] Franch, X., and Carvalo., J.P., "Using Quality Models in Software Package Selection," www.cs.helsinki.fi/u/przybils/causes/cbd06/papers/01159027.pdf

[3] Klas, M., and Munch , J., "Balancing Upfront Definition and Customization of Quality Models," www4.informatik.tu-muenchen.de/~Wagnerst/sqml/08/.../6-Klaesm.pdf

[4] Liu, Y., Ngu A.H.H. and Zeng, L. "QoS Computation and Policing in Dynamic Web Service Selection," WWW 2004, may 17-22, New York, USA. ACM 1-58113-912-8/04/0005 http://citeseer.ist.psu.edu/711840.html

[5] Narasimhan, V.L., Parthasarathy, P.T. and Das, M., "Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE)" iisit.org/Vol6/IISITv6p731-740Narasimhan652.pdf