KOMPONEN PENILAIAN KUALITAS PERANGKAT LUNAK BERDASARKAN *SOFTWARE QUALITY MODELS*

Wayan Gede Suka Parwita, Luh Arida Ayu Rahning Putri
Fakultas MIPA, Universitas Gadjah Mada, Yogyakarta
E-mail : gede.suka@gmail.com

Fakultas MIPA, Universitas Gadjah Mada, Yogyakarta
E-mail : ariebat@yahoo.com

ABSTRAK

Tujuan dari pembuat perangkat lunak adalah untuk menciptakan perangkat lunak yang berkualitas. Tujuan tersebut dapat dicapai dengan melakukan penilaian terhadap kualitas perangkat lunak. Penilaian kualitas perangkat lunak melibatkan banyak komponen. Komponen-komponen yang dilibatkan dalam penilaian sangat bergantung pada model yang digunakan dalam melakukan penilaian. Software Quality Model merupakan model yang digunakan untuk menentukan komponen yang terlibat dalam penilaian. Paper ini menjabarkan berbagai jenis software quality model yang ada beserta komponen-komponen yang digunakan dalam melakukan penilaian dari masing-masing model.

Key word : Komponen Penilaian Software, Software Quality Model

1. PENDAHULUAN

Metode peningkatan kualitas perangkat lunak memiliki peran yang berharga dalam praktek rekayasa perangkat lunak [6]. Beberapa metode ini termasuk inspeksi kode, penelusuran desain, simulasi prototipe, dan pengukuran berbasis analisis [12].

Beberapa organisasi, seperti ISO dan IEEE, mencoba untuk mebuat standar kualitas software dengan mengkombinasikan model dan mengaitkan karakteristik dan sub-karakteristik *quality model*. Beberapa penelitian juga mengusulkan metrik perangkat lunak sebagai alat untuk mengukur *source code* program, arsitektur, dan kinerja dari perangkat lunak. Sampai saat ini proses penilaian kualitas perangkat lunak tetap menjadi masalah terbuka dengan banyak model karena belum jelas dan belum adanya kesepakatan hubungan diantara *software quality model* serta hubungan antara model dan metrik [4].

Tujuan dari *qulity model* adalah untuk memberikan definisi operasional untuk kualitas. Tidak ada kesepakatan mengenai apa yang merupakan kualitas dalam pengertian umum dalam rekayasa perangkat lunak [5]. Masalah yang umum dalam *software quality model* adalah mencari model optimal dan memadai untuk memenuhi tujuan peningkatan kualitas perangkat lunak.

2. STUDI PUSTAKA

Langkah awal untuk memahami kualitas perangkat lunak adalah dengan menjawab pertanyaan penting yang sering ditanyakan, yaitu: apa itu kualitas? Pemahaman terhadap konsep kualitas akan memudahkan dalam memahami perbedaan struktur dari kualitas yang tersedia di pasar. Ada dua kubu besar ketika membahas makna dan definisi kualitas perangkat lunak [9]:

1) Kesesuaian dengan spesifikasi: Kualitas yang didefinisikan sebagai materi produk dan layanan yang terukur dimana memenuhi karakteristik spesifikasi tetap yaitu, kesesuaian dengan spesifikasi yang sebelumnya didefinisikan.
2) Memenuhi kebutuhan pelanggan: Kualitas yang diidentifikasi independen dari setiap karakteristik terukur. Artinya, kualitas didefinisikan sebagai kemampuan produk atau jasa untuk memenuhi harapan pelanggan baik secara eksplisit atau tidak.

Dalam buku "Quality is free: the art of making quality certain", Philip B. Crosby menulis [7]: Kesalahan pertama adalah dengan mengasumsikan bahwa kualitas berarti kebaikan, atau kemewahan atau bersinar. Kata "kualitas" sering digunakan untuk menandakan nilai relatif dari sesuatu dalam frase seperti "kualitas baik", "kualitas buruk" dan "kualitas hidup" - yang berarti hal yang berbeda untuk setiap orang. Jika kita ingin mengelola kualitas, maka kualitas harus didefinisikan sebagai "kesesuaian dengan persyaratan", sehingga jika terdeteksi suatu ketidaksesuaian maka dapat dikatakan tidak adanya kualitas. Masalah kualitas menjadi masalah ketidaksesuaian dan pada akhirnya kualitas dapat didefinisikan.

Crosby dengan jelas mengatakan "kesesuaian dengan spesifikasi" adalah definisi dari kualitas. Crosby juga menekankan bahwa mendefinisikan kualitas secara jelas sangat penting untuk dapat mengukur dan mengelola konsep kualitas itu sendiri.

Crosby merangkum perspektifnya terhadap kualitas dalam empat belas langkah yang kemudian membentuk empat dasar "absolut" dari manajemen kualitas/mutu [7]:

1) Kualitas didefinisikan sebagai penyesuaian dengan persyaratan, bukan sebagai "kebaikan" atau "elegan"
2) Sistem yang menyebabkan kualitas adalah pencegahan, bukan penilaian. Artinya, sistem kualitas untuk pembuat perangkat lunak dalam memenuhi kebutuhan pelanggan adalah melakukannya dengan benar dari awal. Crosby adalah pendukung kuat dari pencegahan, bukan inspeksi. Dalam setiap organisasi berorientasi kualitas Crosby, setiap orang memiliki tanggung jawab untuk pekerjaannya sendiri. Tidak ada orang lain yang boleh menemukan kesalahan dari pekerjaan kita.
3) Standar kinerja harus “Zero Defects”, bukan " that's close enough ". Crosby telah menganjurkan gagasan bahwa tidak terjadinya kesalahan dapat dan harus menjadi sasaran yang akan dicapai.
4) Pengukuran kualitas adalah biaya kualitas. Biaya ketidaksempurnaan, jika dikoreksi, memiliki efek menguntungkan langsung pada kinerja “bottom-line” serta pada hubungan dengan pelanggan. Untuk mencapai tingkatan tersebut, investasi harus dilakukan dengan memeberikan pelatihan dan kegiatan pendukung lainnya untuk meminimalkan kesalahan dan memperoleh kembali biaya yang terbuang.

2.1. McCall Model

Model McCall mencoba untuk menjembatani kesenjangan antara pengguna dan pengembang dengan berfokus pada sejumlah faktor kualitas perangkat lunak yang mencerminkan pandangan pengguna dan prioritas pengembang [11]. Gagasan utama dalam model McCall adalah untuk menilai relativitas hubungan sosial antara faktor-faktor kualitas eksternal dan kriteria kualitas produk. Model ini dikembangkan oleh angkatan udara Amerika Serikat pada sistem keputusan elektronik (Electronic System Decision), pusat pengembangan Rome Air (Rome Air Development Center) dan General Electric (GE), dengan tujuan meningkatkan kualitas produk perangkat lunak [1].

Salah satu kontribusi besar dari model McCall adalah hubungan antara karakteristik kualitas dan metrik, walaupun terdapat kritik bahwa tidak semua metrik adalah obyektif. Salah satu aspek yang tidak dipertimbangkan langsung oleh model ini adalah fungsionalitas dari produk perangkat lunak [1].

2.2. Boehm Model

Model Boehm menambahkan beberapa karakteristik pada model McCall dengan penekanan pada pemeliharaan produk perangkat lunak. Pertimbangan tentang apa yang terlibat dalam evaluasi produk perangkat lunak sehubungan dengan kegunaan program juga termasuk dalam model ini. Model Boehm serupa dengan model McCall dalam merepresentasikan struktur hirarkis karakteristik, yang masing-masing memberikan kontribusi terhadap kualitas keseluruhan.

Gagasan dari model Boehm mencakup kebutuhan pengguna, seperti pada model McCall, namun model Boehm hanya memuat diagram tanpa adanya saran tentang pengukuran karakteristik kualitas [1].

2.3. FURPS/FURPS+ Model

Model FURPS diusulkan oleh Robert Grady dan Hewlett-Packard Co. Model ini menguraikan karakteristik dalam dua kategori yang berbeda dari persyaratan (*requirement*) [1], yaitu:

- Fuctional Requirement (F): Ditetapkan oleh input dan output yang diharapkan.
- Non-Fungsional Requirements (URPS): Kegunaan (*usability*), kehandalan (*reliability*), kinerja (*performance*), daya dukung (*supportability*).

Terdapat lima persyaratan yang tercakup dalam dua kategori karakteristik tersebut. *Function* meliputi himpunan fitur yang diharapkan serta kemampuan dan keamanan. *Usability* meliputi faktor manusianya, seperti estetika, konsistensi dalam user interface, bantuan yang sifatnya online dan “context-sensitive”, *wizards* dan agen, dokumentasi untuk pengguna, dan materi pelatihan. *Reability* mencakup frekuensi dan tingkat keparahan kegagalan (*failure*), pemulihan (*recovery*), akurasi, prediksi dan waktu rata-rata antar terjadinya kegagalan (*Mean Time Between Failure*). *Peformance* menekankan pada kondisi persyaratan fungsional seperti kecepatan, efisiensi, ketersediaan, akurasi, *throughput*, waktu respon, waktu pemulihan, dan pemanfaatan sumber daya. *Supportability* meliputi kemampuan untuk dapat diuji, dapat dikembangkan, kemampuan adaptasi, pemeliharaan, kompatibilitas, dapat dikonfigurasi, servis, kemampuan instalasi, localizability (internationalization?).

Model FURPS kemudian diperluas oleh Rational Software - sekarang IBM Rational Software - menjadi FURPS +. Perluasan (+) yang dilakukan meliputi persyaratan untuk batasan desain, persyaratan implementasi, persyaratan antarmuka dan persyaratan fisik.

Salah satu kelemahan dari model FURPS adalah model ini tidak mempertimbangkan portabilitas dari produk perangkat lunak [1].

2.4. Dromey Model

Model Dromey berusaha untuk meningkatkan pemahaman tentang hubungan antara atribut (karakteristik) dan sub-attribut (sub-karakteristik) dari kualitas. Lapisan-lapisan model ini didefinisikan sebagai atribut level atas dan atribut bawahan. Ide utama dalam pembuatan model baru ini adalah untuk mendapatkan model yang dapat bekerja dalam lingkup yang luas dalam sistem yang berbeda. Dromey mengakui bahwa evaluasi untuk setiap produk adalah berbeda sehingga diperlukan ide yang lebih dinamis untuk memodelkan proses evaluasi tersebut. Kekurangan dari model ini adalah tidak diberikan kriteria yang jelas untuk melakukan pengukuran kualitas perangkat lunak [1]. Ada tiga unsur utama pada model kualitas generik Dromey ini [2]:

1) Sifat produk yang mempengaruhi kualitas
2) Atribut kualitas level atas
3) Sarana yang menghubungkan sifat produk dengan atribut kualitas.

Quality model Dromey lebih lanjut dapat disusun dalam 5 langkah proses [2]:

1) Memilih satu set atribut kualitas level atas yang diperlukan untuk evaluasi.
2) Daftar komponen/modul dalam sistem.
3) Mengidentifikasi sifat pembawa kualitas untuk komponen/modul (kualitas komponen yang memiliki pengaruh paling besar pada sifat produk dari daftar di atas).
4) Tentukan berapa masing-masing sifat mempengaruhi atribut kualitas.
5) Mengevaluasi model dan mengidentifikasi kelemahan.

2.5. BBN Model

Bayesian Belief Network (BBN) adalah kategori khusus dari model grafis, dimana node mewakili variabel dan panah berarah mewakili hubungan antar node. Sebagai *quality model*, BNN dapat direpresentasikan dalam struktur hirarki. Akar dari pohon adalah node yang merepresentasikan kualitas dan terhubung ke node anak yang merepresentasikan karakteristik kualitas, kemudian setiap node karakteristik kualitas terhubung dengan sub-karakteristik kualitas yang sesuai.

Model ini dapat mewakili dan memanipulasi model yang kompleks yang mungkin tidak akan pernah diimplementasikan dengan menggunakan metode konvensional. Bagaimanapun, karena kurangnya kriteria, model ini tidak dapat digunakan untuk evaluasi terkoordinasi dari produk perangkat lunak [1].

2.6. Star Model

Kualitas perangkat lunak model Star adalah suatu model konseptual untuk menyajikan perspektif yang berbeda dari kualitas perangkat lunak. Meskipun model ini menggunakan sudut pandang yang berbeda dari kualitas, tetapi sama dengan model BBN, model ini memiliki kekurangan dari kurangnya kriteria [1].

2.7. Kazman Model

Pengelompokan pada model Kazman ini membagi karakteristik kualitatif ke dalam dua kelompok yang dapat diamati selama waktu bekerja dan selama adanya siklus keberadaan perangkat lunak. Karakteristik dari dua kelompok tersebut adalah sebagai berikut:

- Efisiensi, keamanan, ketersediaan dan fungsi.
- Modifiability, portabilitas, *usability*, *inheritability* dan *testability*.

Kelompok ini, pada kenyataannya, tidak disajikan dengan model kualitatif spesifik, tetapi telah memberikan cara evaluasi analisis arsitetur tradeoff untuk menyelidiki kualitas arsitektur perangkat lunak, dengan cara ini sistem harus menetapkan model kualitatif mengenai kebutuhan mereka. Pembangunan model kualitatif yang telah dibuat dari basis disebut "utilitas". Tingkat terakhir pohon ini didefinisikan oleh serangkaian skenario untuk menguji karakteristik kualitatif [8].

2.8. ISO/IEC 9126 Model

Banyaknya penelitian tentang model pengukuran kualitas memang sangat berguna, namun di sisi lain menimbulkan kebingungan karena aspek kualitas yang ditawarkan. Hal ini menyebabkan timbulnya kebutuhan akan suatu model standar.

Ini adalah alasan ISO / IEC JTC1 mulai mengembangkan konsensus yang diperlukan dan mendorong standarisasi seluruh dunia.

Pertimbangan pertama berasal dari tahun 1978, dan pada tahun 1985 perkembangan ISO / IEC 9126 dimulai. ISO 9126 adalah bagian dari ISO 9000 standar, yang merupakan standar paling penting untuk jaminan kualitas. Dalam model ini, totalitas atribut kualitas produk perangkat lunak diklasifikasikan dalam struktur pohon hirarkis karakteristik dan sub karakteristik. Level tertinggi dari struktur ini terdiri dari karakteristik kualitas dan tingkat terendah terdiri dari kriteria kualitas perangkat lunak.

Model ini menentukan enam karakteristik termasuk Functionality, Reliability, Usability, Efisiensi, Maintainability dan Portabilitas, yang dibagi lagi menjadi 21 sub karakteristik. Sub karakteristik diwujudkan eksternal ketika perangkat lunak digunakan sebagai bagian dari sistem komputer, dan merupakan hasil dari atribut perangkat lunak internal. Karakteristik didefinisikan berlaku untuk setiap jenis perangkat lunak, termasuk program komputer dan data yang terdapat dalam firmware dan memberikan terminologi yang konsisten untuk kualitas produk perangkat lunak. Model ini juga menyediakan kerangka kerja untuk membuat timbal balik antara kemampuan produk perangkat lunak .

2.9. IEEE Model

Lembaga IEEE, pada kenyataannya, telah memberikan skala dan standar untuk menyediakan model kualitatif dan belum memberikan model kualitatif yang jelas dan menekankan pada bagaimana merancang cara pengukuran faktor kualitatif. Konstruksi yang disarankan adalah semi-tree dan memiliki tiga tingkatan. tingkat terakhir adalah metrik perangkat lunak. dalam model ini diperbolehkan untuk mendefinisikan metrik untuk setiap faktor jika ada pengukuran langsung setelah tingkat pertama. Untuk peunjuk pertama, pohon dengan faktor dan sub faktor diberikan sebagai berikut [8]:

- *Efficiency* : ekonomi temporal dan sumber daya ekonomi.
- *Reability* : nondeficiency, toleransi kesalahan dan ketersediaan.
- *Function* : kelengkapan, keakuratan, keamanan, kompatibilitas dan interoperabilitas
- *Supportability*: testability, perluasan dan perbaikan.
- *Portability*: independensi dari perangkat keras, independensi dari perangkat lunak, instalability dan reusabilitas.
- *Usability* : ketelitian, mudah dipelajari, kegunaan, dapat diterangkan.

3. PERBANDINGAN *SOFTWARE QUALITY MODEL*

Adapun faktor, atribut dan karakteristik yang umumnya yg digunakan untuk perbandingan adalah sebagai berikut:

Correctness : Sejauh mana program memenuhi spesifikasi dan memenuhi tujuan tugas pengguna. Kebenaran adalah sejauh mana kerja produk dan output yang bebas dari cacat hasil kerja produk setelah dikirimkan. Kebenaran menjawab pertanyaan khas berikut: Apakah aplikasi dan data sudah lengkap, akurat dan konsisten? [3].

Efciency : Kemampuan produk perangkat lunak untuk memberikan kinerja yang sesuai dan relatif terhadap jumlah sumber daya yang digunakan pada saat keadaan tersebut. Efisiensi adalah sejauh mana sesuatu secara efektif menggunakan (yaitu, meminimalkan konsumsi atas) sumber dayanya. Sumber daya ini dapat mencakup semua jenis sumber daya seperti komputer (perangkat keras, perangkat lunak, dan jaringan), mesin, fasilitas, dan personil. Juga, jumlah sumber daya komputasi dan kode yang diperlukan oleh program untuk melakukan fungsi, kumpulan atribut yang digunakan pada hubungan antara tingkat kinerja perangkat lunak dan jumlah sumber daya yang digunakan saat itu. Efisiensi berkaitan dengan "membagi beban, deteksi kesalahan end-to-end: tes sederhana, cacat kinerja yang muncul di bawah beban berat, mengutamakan keselamatan, penskalaan, throughput, latency, ketersediaan. Apakah model memenuhi tujuannya tanpa pemborosan sumber daya?[3].

Flexibility : Upaya yang diperlukan untuk memodifikasi program operasional. Upaya untuk mengubah atau memodifikasi produk perangkat lunak untuk beradaptasi dengan lingkungan lain atau menjadi aplikasi lain yang berbeda dari rancangannya [3].

Functionality : Kemampuan produk perangkat lunak untuk menyediakan fungsi yang dinyatakan memenuhi dan mengandung yang dibutuhkan ketika perangkat lunak digunakan dalam kondisi tertentu. Fungsionalitas merupakan atribut yang menjaga keberadaan fungsi dan sifat spesifik mereka. Fungsi adalah sesuatu yang memenuhi atau mengandung kebutuhan. Fungsionalitas dinilai dengan mengevaluasi fitur dan kemampuan dari program, sifat umum dari fungsi yang dikirimkan dan keamanan sistem secara keseluruhan[3].

Integrity : Sejauh mana akses ke perangkat lunak atau data oleh orang yang tidak berhak dapat dikendalikan. Juga, atribut yang terkait dengan pengendalian produk perangkat lunak untuk akses ilegal untuk program dan data [3].

Interprobability : Kemampuan produk perangkat lunak untuk berinteraksi dengan satu atau lebih sistem tertentu. Juga, upaya yang diperlukan untuk memasangkan satu sistem dengan yang lain, atribut perangkat lunak yang bergantung pada kemampuannya untuk berinteraksi dengan sistem tertentu, sejauh mana sistem atau salah satu komponennya terhubung dengan benar dan beroperasi dengan sesuatu yang lain [3].

Maintainability : Kemampuan produk perangkat lunak untuk dimodifikasi. Modifikasi dapat mencakup koreksi, perbaikan atau adaptasi dari perangkat lunak untuk disesuaikan dengan lingkungan, dan dalam persyaratan dan spesifikasi fungsional. Juga, upaya yang diperlukan untuk menemukan dan memperbaiki kesalahan dalam program operasional. Pemeliharaan adalah saat dimana aplikasi atau komponen dapat dipertahankan antara rilis utama. Juga, atribut yang diperhatikan pada upaya yang diperlukan untuk membuat modifikasi tertentu, eberapa banyak perubahan atau memodifikasi komponen untuk memperbaiki kesalahan, untuk meningkatkan kinerja, atau untuk beradaptasi dengan lingkungan yang berubah [3].

Portability : Kemampuan produk perangkat lunak untuk dapat dipindahkan dari satu lingkungan ke lingkungan yang lain. Juga, upaya yang diperlukan untuk memindahkan program dari satu konfigurasi perangkat keras dan atau lingkungan sistem perangkat lunak ke sistem lain. Portabilitas adalah saat dimana aplikasi atau komponen dapat dipindahkan dari satu lingkungan yang lain.

Reability : Kemampuan produk perangkat lunak untuk mempertahankan tingkat kinerja tertentu ketika digunakan dalam kondisi tertentu. Keandalan adalah kemampuan yang diharapkan dari program untuk melakukan fungsinya yang membutuhkan ketelitian. Hal ini dievaluasi dengan mengukur frekuensi dan tingkat keparahan kegagalan, keakuratan hasil output, waktu yang berarti antara kegagalan (MTBF), kemampuan untuk pulih dari kegagalan dan prediktabilitas dari program ini karena program yang tidak dapat dipercaya sering gagal, atau menghasilkan data yang tidak benar. Juga, kehandalan merupakan atribut yang ditentukan pada kemampuan perangkat lunak untuk mempertahankan tingkat kinerja saat di bawah kondisi untuk jangka waktu tertentu. Keandalan adalah sejauh mana produk beroperasi tanpa kegagalan dalam kondisi tertentu selama periode waktu tertentu [3].

Reusability : Reusabilitas adalah saat dimana aplikasi atau komponen yang sudah ada dapat digunakan kembali. Ini adalah sejauh mana sebuah program dapat digunakan dalam aplikasi lain yang berkaitan dengan kemasan dan ruang lingkup fungsi yang program lakukan. Misalnya, reusabilitas yang mungkin ketika semua modul berisi dua atau lebih fungsi unik yang jika dipisahkan dari kode utama, dapat digunakan kembali oleh program lain. Selain itu, ia merupakan atribut yang berkaitan dengan beban transfer modul atau program untuk aplikasi lain [3].

Testability : Kemampuan produk perangkat lunak yang memungkinkan modifikasi perangkat lunak untuk divalidasi. Juga, upaya yang diperlukan untuk menguji program untuk memastikan ia melakukan fungsi yang diharapkan. Testability adalah saat dimana aplikasi atau komponen memfasilitasi penciptaan dan pelaksanaan keberhasilan tes (yaitu, tes yang akan menyebabkan kegagalan yang disebabkan semua cacat yang ada). Juga, atribut perangkat lunak yang berkaitan dengan upaya yang diperlukan untuk memvalidasi modifikasi perangkat lunak [3].

Understandability : Kemampuan produk perangkat lunak untuk memungkinkan pengguna untuk memahami apakah perangkat lunak tersebut cocok, dan bagaimana perangkat lunak itu dapat digunakan untuk tugas dan kondisi tertentu. Juga, atribut perangkat lunak yang diperlukan dalam upaya pengguna untuk mengenali konsep logis dan penerapannya [3].

Pembandingan *software quality model* dilakukan dengan memperhatikan faktor, atribut dan karakteristik di atas. Berikut merupakan tabel perbandingannya:

Tabel 1. Perbandingan *Software Quality Models*

Faktor/ Atribut/ Karakteristik	McCall	Boehm	FURPS / FURPS+	Dromey	BBN	Kazman	Star	ISO	IEEE
Correctness	✓						✓		
Efficiency	✓	✓	✓	✓	✓	✓	✓	✓	✓
Flexibility	✓					✓	✓		
Functionality			✓	✓	✓	✓	✓	✓	✓
Integrity	✓								
Interoperability	✓				✓		✓		
Maintainability	✓		✓	✓	✓	✓	✓	✓	✓
Portability	✓	✓		✓	✓		✓	✓	✓
Reliability	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reusability	✓			✓			✓		
Testability	✓	✓				✓	✓		
Understandability		✓	✓						

4. KESIMPULAN

Pengukuran kualitas perangkat lunak dapat dilakukan dengan menggunakan salah satu dari berbagai model yang ada. Pada penggunaan model Star dan BNN, kekurangan yang ada sama yaitu kurangnya kriteria yang digunakan. Akan tetapi model Star dapat mempertimbangkan kualitas dari sudut pandang yang berbeda, dan model BBN menitik beratkan pada faktor kualitas. Model ISO merupakan standar internasional yang ada saat ini, hanya saja model ini besifat umum. Akan tetapi model ini memiliki kriteria evaluasi dan memisahkan kualitas eksternal dan internal yang ada sehingga model ini cocok digunakan pada saat kebutuhan penilaian kualitas perangkat lunak tidak terlalu dalam. Model McCall dan Star memenuhi banyak komponen penilaian, akan tetapi ada penilaian yang tumpang tindih karena banyaknya komponen yang dinilai. Model ini sesuai digunakan jika permasalahan utama adalah penilaian secara menyeluruh dan mendalam. Model IEEE hampir sama dengan model ISO, hanya saja pada model IEEE menggunakan metrik perangkat lunak pada tingkat penilaian terakhir. Akibatnya waktu penilaian akan lebih lama dan juga kedalaman penilaian akan lebih mendetail. Salah satu keunggulan model Boehm adalah dilibatkannya faktor perangkat keras. Hanya saja pada model ini criteria yang digunakan masih sedikit. Model FURPS sendiri tidak memperdulikan portability, akan tetapi model ini memisahkan kebutuhan fungsional dan non fungsional. Pada model Kazman tingkat terakhir penilaian adalah menggunakan beberapa skenario untuk pengujian karakteristik. Penggunaan model tersebut disesuaikan dengan kebutuhan yang diperlukan seperti waktu, kedalaman pengukuran kualitas, kompleksitas dan juga fungsi dari perangka lunak tersebut. Kualitas perangkat lunak diukur menggunakan beberapa faktor, atribut dan juga karakteristik.

DAFTAR PUSTAKA

[1] B. Behkamal, M. Kahani, M.K. Akbari, “Customizing ISO 9126 quality model for evaluation of B2B applications”, Information and Software Technology, vol.51, pp. 599–609, 2009.
[2] D. Milicic, “Software Quality Models and Philosophies”, Software Quality Attributes and Trade-Offs, L. Lundberg, M. Mattsson, C. Woh lin Eds., Blekinge Institute of Technology, pp. 3-19, 2005.
[3] K. Khosravi, and Y.G. Guéhéneuc, “A Quality Model for Design Patterns”, Technical report 1249, University of Montreal, September 2004.
[4] K. Khosravi, and Y.G. Guéhéneuc, “On Issues with Software Quality Models”, ICFAI University Press, ch. 11, pp. 218--235, January 2008.
[5] M.A. Côté, W. Suryn, E. Georgiadou, “Software Quality Model Requirements for Software Quality Engineering“, Software Quality Management & INSPIRE Conference (BSI), 2006.
[6] N.F. Schneidewind, “Body of Knowledge for Software Quality Measurement,” Computer, vol. 35, no. 2, pp. 77-83, Feb. 2002.
[7] P. B. Crosby, “Quality is free : the art of making quality certain”, New York : McGraw-Hill, 1979.
[8] R. Khayami, A. Towhidi, and K. Ziarati, “The analytical Comparation of Qualitative Models of Software Systems”, World Applied Sciences Journal, Volume 6 (Supplement 1), 2009.
[9] R. W. Hoyer, and B. B. Y. Hoyer, "What is quality?", Quality Progress, no. 7, pp. 52-62, 2001.
[10] R.B. Grady, “Practical software metrics for project management and process improvement”, Prentice Hall, New Jersey, 1992.
[11] Rafa E. Al-Qutaish, PhD Journal of American Science 2010 “Quality Models in Software Engineering Literature: An Analytical and Comparative Study”, Journal of American Science; pp:166-175, 2010.
[12] Y. Liu, T.M. Khoshgoftaar, and N. Seliya, “Evolutionary Optimization of Software Quality Modeling with Multiple Repositories”, IEEE Transactions on Software Engineering, vol. 36, no. 6, November/December 2010 .