Karakteristik Implementasi Penjaminan Mutu pada Proyek Pengembangan Perangkat Lunak berbasis *Open Source* dan *Proprietary*[*]

Harry B. Santoso[1], **Dewi Mairiza**[2], **dan R.M. Samik Ibrahim**[3]

Fakultas Ilmu Komputer Universitas Indonesia
Kampus Baru Universitas Indonesia – Depok, Jawa Barat 16424
e-mail: {harrybs[1], mairiza[2]}[at] cs [dot] ui [dot] ac [dot] id dan rmsi.vlsm.org [at] gmail [dot] com[3]

[*] Paper ini disampaikan dalam Seminar Nasional Ilmu Komputer dan Teknologi Informasi (*National Conference on Computer Science and Information Technology*) VII, Depok, Indonesia, 2007, pp. 260-265 [prosiding]. Silakan Anda memanfaatkan paper ini untuk keperluan pendidikan dan penelitian lebih lanjut. Mohon dapat menggunakan metode penyitiran yang sudah umum digunakan.

ABSTRAK

Secara umum, mekanisme pengembangan perangkat lunak dapat dibedakan menjadi 2 (dua), yaitu pengembangan perangkat lunak berbasis *open source* dan yang berbasis *proprietary*. Dua hal ini menurut asumsi penulis akan menyebabkan adanya perbedaan dalam konteks penerapan metode penjaminan mutu perangkat lunak. *Open Source Software* memiliki karakteristik tersedianya jaringan atau basis SDM yang luas. Sedangkan di sisi yang lain, *Proprietary Software* yang merupakan metode pengembangan perangkat lunak konvensional memiliki dukungan biaya serta Sumber Daya Manusia (SDM) dari internal perusahaan. Pada paper ini akan dibahas karakteristik 2 (dua) mekanisme pengembangan perangkat lunak serta pengaruhnya terhadap pendekatan yang harus digunakan untuk menjalankan metode Penjaminan Mutu Perangkat Lunak.

Kata kunci: Penjaminan Mutu Perangkat Lunak (PMPL), Open Source Software (OSS), Proprietary Software (PS).

1. PENDAHULUAN

Pada era dewasa ini, Teknologi Informasi dan Komunikasi (TIK) telah merambah dan berkontribusi positif dalam berbagai sektor; mulai dari sektor pemerintahan dengan e-Government, di sektor perekonomian dengan e-Business, di sektor industri dengan e-Procurement, di dunia pendidikan dengan e-Learning, serta di berbagai bidang lainnya. Akumulasi modal yang sangat besar dalam sektor TIK membuat banyak pihak tertarik untuk terjun di dalamnya dengan melakukan penelitian atau menjadikan sarana bisnis. Secara umum TIK dapat dibagi menjadi 3 (tiga) bidang utama, yaitu perangkat keras (*hardware*), perangkat lunak (*software*), hingga layanan (*services*). Di antara ketiga bidang tersebut, perangkat lunak menarik untuk dibahas terkait dengan cepatnya evolusi atau perubahan yang terjadi serta karakteristik manajemen sumber daya yang bergerak dalam pengembangan perangkat lunak. Perangkat lunak sendiri dapat dibagi menjadi 4 segmen utama, yaitu (1) *Software Product*, yaitu produk perangkat lunak yang kadang dijual secara paket tersendiri (*sold shrink-wrapped*), disewakan, dipaket dengan perangkat keras atau dengan layanan konsultan, (2) *Software Services*, dimana layanan ini termasuk konsultasi dan disain sistem, integrasi sistem, pemrograman berdasarkan kontrak dan pemeliharaan sistem, (3) *In-house Software* yang dikembangkan oleh departemen Teknologi Informasi (TI) perusahaan untuk menurunkan biaya operasional bisnis, juga sebagai upaya strategis dalam menawarkan jangkauan yang lebih luas atas produk dan layanan yang lebih responsif kepada pelanggan, dan (4) *Embedded Software*, yaitu bagian dari produk atau layanan, namun tidak dijual secara terpisah sebagai perangkat lunak [1]. Sedangkan segmen yang paling terlihat (besar) adalah bisnis *software products*, yaitu sebesar 70 miliar USD untuk seluruh dunia di tahun 1993 [2]. Adapun pada tahun 2005, pasar dunia untuk bidang TI, baik dari sisi produk maupun layanan diperkirakan mencapai 2.1 triliun USD. Produk perangkat lunak sendiri diperkirakan mencapai 200 milyar USD untuk pasar global di tahun 2005 [3].

Dari sisi pengembangan perangkat lunak sendiri, terdapat 2 (dua) pendekatan yang digunakan, yaitu *Open Source Software* (OSS) dan *Proprietary Software* (PS). Perbedaan pendekatan ini lebih ditinjau dari segi lisensi, dimana pada OSS, *source code* terbuka untuk publik, dimana publik berhak untuk menggunakan, memodifikasi, serta mendistribusikan di bawah lisensi *open source* tertentu. Sebaliknya untuk PS, ada batasan atas hak

seseorang untuk menyalin atau menggunakan suatu perangkat lunak, sesuai dengan lisensi yang dikeluarkan oleh vendor. Konsekuensi logis dari perbedaan pendekatan ini adalah perbedaan metode pengembangan yang ada hubungannya dengan manajemen sumber daya pengembangan perangkat lunak.

Aktivitas PMPL menekankan pada aspek proses pengembangan perangkat lunak. Dengan proses pengembangan yang terukur dan terprogram sejak awal, dapat dijamin bahwa produk yang dihasilkan akan memenuhi kriteria kualitas. Dalam paper ini akan dibahas pemetaan komponen pengembangan perangkat lunak dengan komponen penjaminan mutu, sehingga tergambar variabel berpengaruh apa saja yang perlu dipertimbangkan dalam melakukan proses penjaminan mutu pada kedua jenis perangkat lunak, baik OSS maupun PS.

OSS dalam konteks kajian pembahasan Penjaminan Mutu Perangkat Lunak (PMPL) ini adalah OSS yang memang dikembangkan oleh komunitas *open source* (pendekatan '*bazaar*'), bukan proyek pengembangan perangkat lunak berbasis OSS yang dikembangkan dengan pendekatan pengelolaan yang profesional (*Professional Open Source* atau OSS v.2). Sehingga dengan pembatasan ruang lingkup OSS yang menjadi topik kajian akan tergambar perbedaan implementasi PMPL pada OSS dengan PS yang dikenal memiliki pendekatan '*katedral*'.

Sebelum membahas tentang metode PMPL pada kedua jenis pengembangan perangkat lunak tersebut, di bawah ini akan dibahas pengertian dan beberapa aspek dari masing-masing jenis pendekatan.

2. PENGEMBANGAN PERANGKAT LUNAK BERBASIS *OPEN SOURCE*

Ide dasar dari OSS sebenarnya sederhana, yaitu "Ketika para *programmer* dapat membaca, mendistribusikan, dan memodifikasi *source code* dari perangkat lunak, maka perangkat lunak akan ber-'evolusi'. Orang-orang memperbaikinya, beradaptasi dengannya, dan memperbaiki atau membuang berbagai kesalahan (*bugs*). Dan bila hal ini diimplementasikan dalam *slow pace* pengembangan perangkat lunak konvensional, akan terlihat mengagumkan [4]." Mengenai pengertian OSS, penulis sengaja mengutip dari Open Source Initiaves (OSI) [4], yang menyebutkan poin-poin terkait dengan OSS, diantaranya: (1) *Free Redistribution*, (2) *Source Code*, (3) *Derived Works*, (4) *Integrity of the Author's Source Code*, (5) *No Discrimination Against Persons or Groups*, (6) *No Discrimination Against Fields of Endeavor*, juga poin-poin lainnya seperti (7) *Distribution of License*; (8) *License Must Not be Specific to a Product*; (9) *License Must Not Restrict Other Software*; dan (10) *License Must Be Technology-Neutral*.

Adapun penerimaan yang luas dari masyarakat terhadap OSS terangkum dalam studi yang disebut GRAM (*Generally Recognized as Mature*) OSS/FS programs [5]. Dimana kriteria yang digunakan adalah penggunaannya yang signifikan (dalam *market area*), pengembangan atau dukungan yang signifikan, fungsionalitas yang matang, keamanan, kualitas, dan biaya. Di samping itu, studi terhadap OSS dalam perkembangan terakhir menunjukkan adanya pergeseran (transformasi) dari OSS menuju bentuk yang *commercially viable* (OSS 2.0) [6]. Studi tersebut mengidentifikasi bagaimana OSS 2.0 dapat mengakomodasi transformasi yang terjadi dengan jalan mendapatkan keseimbangan antara nilai keuntungan komersial (PS) dan nilai-nilai komunitas open source (OSS).

Pada siklus pengembangan OSS, terdapat beberapa tahapan yang perlu dilalui [7] dalam [6], yaitu:

Code: penulisan code dan dikirim pada komunitas OSS untuk direview.

Review: merupakan salah satu kekuatan OSS, yaitu bersifat independen serta ada mekanisme peer review.

Pre-commit test: jaminan bahawa semua kontribusi yang diberikan untuk membangun OSS telah melalui proses testing secara hati-hati sebelum mendapatkan status *committed*.

Development release: kontribusi berupa kode dapat dimasukkan ke dalam rilis pengembangan dalam waktu yang relatif singkat sejak kode tersebut dikirimkan – implementasi yang singkat ini menjadi motivasi tersendiri bagi developers.

Parallel debugging: besarnya jumlah *debuggers* yang berpotensi dengan platform yang berbeda dan konfigurasi sistem menjamin *bugs* dapat ditemukan dan diperbaiki secara cepat.

Production release: dalam kondisi yang relatif stabil, versi produksi dari sistem dirilis setelah melalui proses *debugging.*

3. PENGEMBANGAN PERANGKAT LUNAK BERBASIS *PROPRIETARY*

PS berbeda dengan OSS. Setidaknya hal tersebut dapat terukur dari metode penggunaan sumber daya yang digunakan, jangka waktu pengembangan, serta biaya pengembangan perangkat lunak yang tersedia. PS memiliki metodologi pengembangan yang biasanya telah ditetapkan oleh institusi atau vendor yang mengembangkan. Sebagai contoh Jatis Solutions telah memiliki metodologi pengembangan *in-house software*, Parallel Track [8]. SDM yang dimiliki pun dari awal telah berada dalam 'payung' perusahaan untuk mengembangkan perangkat lunak sesuai dengan tanggung jawabnya. Pos-pos SDM untuk *project manager*, *business analyst*, *system analyst*, *system architect*, s*ystem developer*, serta *programmer* juga sudah dipersiapkan pada masa-masa awal pengembangan perangkat lunak. Pada *software house*, jangka waktu pengembangan serta

biaya disesuaikan dengan kebutuhan klien. Atau bila perangkat lunak yang dikembangkan merupakan inisiatif vendor, tentu sudah memiliki target pencapaian yang jelas serta target biaya yang sudah ditetapkan sesuai dengan manajemen proyek yang ada.

Berdasarkan pengertian dalam [9], PS merupakan perangkat lunak yang memiliki batasan-batasan (*restrictions*) baik dalam menggunakan atau menggandakannya, yang biasanya hal tersebut ditentukan oleh pemilik perangkat lunak (*proprietor*). Upaya perlindungan untuk menggunakan, menggandakan, atau memodifikasi memang dapat diterima baik secara legal maupun teknis. *Technical means* termasuk di dalamnya hanya merilis kode binary yang hanya mampu dibaca mesin (*machine-readable binaries*), dan menyembunyikan source code yang dapat dibaca manusia (*the human-readable source code*). *Legal means* termasuk di dalamnya lisensi perangkat lunak, *copyright* dan hak paten. PS dapat dijual untuk mendapatkan uang sebagai perangkat lunak komersial atau dapat juga tersedia dengan tanpa harga (*zero-price*) dengan sebutan *freeware*.

Terminologi "*proprietary software*" digunakan oleh Free Software Foundation (FSF) untuk mendeskripsikan perangkat lunak yang bukan *free software* atau *semi-free software*. Secara teknis, maknanya adalah perangkat lunak yang mempunyai pemilik yang memiliki kontrol atas perangkat lunak tersebut. Terminologi ini dapat digunakan untuk semua perangkat lunak yang memang tidak berada di *public domain*, termasuk *free software*. Berdasarkan FSF, bagaimanapun fokus pada pemilik perangkat lunak merupakan hal yang teramat penting dalam PS, sebaliknya dalam *free software*, kebebasan bagi pengguna komputer merupakan yang menjadi fokus utama. Pemberian sifat "*proprietary*" juga menghindari bias frasa "*commercial software*", karena *free software* juga dapat dijual dan digunakan untuk tujuan-tujuan komersial. Open source Initiative (OSI) lebih menyukai menyebut PS sebagai "*closed source software*". Contoh dari PS antara lain Microsoft Windows, RealPlayer, Adobe Photoshop, Mac OS, WinZip dan beberapa varian/jenis UNIX.

4. KARAKTERISTIK IMPLEMENTASI PENJAMINAN MUTU DALAM PENGEMBANGAN PERANGKAT LUNAK

Sebelum pembahasan mengenai PMPL secara khusus terhadap kedua karakteristik pengembangan perangkat lunak (OSS dan PS), akan dibahas pengertian kualitas (*quality*) dan kualitas perangkat lunak (*software quality*). Hal ini disebabkan sering rancunya penggunaan terminologi kualitas, kualitas perangkat lunak, serta penjaminan kualitas perangkat lunak itu sendiri.

Dalam hal kualitas, The Institute of Electrical and Electronics Engineers' (IEEE) Standard Glossary of Software Engineering Terminology mendefinisikan kualitas sebagai:

> "*the degree to which a system, component, or process meets (1) specified requirements, and (2) customer or user needs or expectations* [10]."

Dimana kualitas didefiniskan sebagai tingkat atau level bagaimana sebuah sistem, komponen, atau proses memenuhi *requirements* yang diminta pengguna, dan kebutuhan atau ekspektasi pengguna.

Sedangkan The International Standards Organization (ISO) mendefinisikan kualitas sebagai:

> "*the totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs* [11]."

ISO menyoroti pada fitur-fitur dan karakteristik dari produk atau layanan dalam kemampuannya memenuhi kebutuhan yang ditentukan.

Adapun definisi kualitas perangkat lunak dalam the Handbook of Software Quality Assurance dalam berbagai makna yang mengerucut pada definisi:

> "*Software quality is the fitness for use of the software product* [12]."

Berikut ini adalah sebagian dari dimensi/komponen kualitas perangkat lunak yang termaktub dalam [13]: (1) *Acceptability*: apakah orang yang menggunakan sistem menemukan kepuasan dari sistem dan apakah sistem memenuhi kebutuhan akan informasi yang diperlukan? (2) *Availability*: apakah sistem dapat diakses; pada saat dan dimana ia diperlukan. (3) *Cohesiveness*: apakah terjadi interaksi antar komponen atau subsistem. (4) *Compatibility*: apakah sistem sesuai (kompatibel) dengan sistem lain. (5) *Documentation*: apakah terdapat dokumentasi yang baik untuk membantu komunikasi antara operator, pengguna, pengembang dan manager. (6) *Ease of learning*: apakah rentang waktu *learning curve* untuk pengguna-pengguna baru cukup pendek dan intuitif. (7) *Economy*: apakah sistem cost-effective dan baik dalam hal sumber daya dan batasa-batasan. (8) *Effectiveness*: apakah sistem menampilkan performa dan berjalan dalam kinerja terbaik untuk memenuhi semua tujuan-tujuan organisasi dan bisnis. (9) *Efficiency*: apakah sistem mendayagunakan resources untuk advantage terbaiknya. (10) *Fast development rate*: apakah waktu yang dibutuhkan untuk mengembangkan proyek cepat, dan relative sesuai dengan scope (size) dan kompleksitasnya. (11) *Flexibility*: apakah sistem mudah untuk dimodifikasi dan apakah cukup mudah untuk menambah atau menghapus komponen. (12) *Functionality*: apakah sistem sudah sesuai dengan requirements. (13) *Implementability*: apakah perubahan dari sistem lama dan sistem baru cukup feasible, dari segi teknis, social, ekonomi, dan organisasi. (14) *Low coupling*: apakah interaksi antar subsistem dapat dimodifikasi tanpa mempengaruhi sistem. (15) *Maintainalibity*: apakah diperlukan banyak effort untuk menjaga sistem agar tetap berjalan secara memuaskan dan terus dapat mengakomodasi perubahan requirement

sepanjang ‘masa hidupnya’. (16) *Portability*: apakah SI dapat berjalan pada equipment lain atau pada tempat lain. (17) *Reliability*: apakah error rate dapat diminimalisasi dan outputnya konsisten serta benar.

Sedangkan PMPL atau Software Quality Assurance (SQA) sendiri memiliki pengertian:

“*..a planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes, and procedures. SQA includes the process of assuring that standards and procedures are established and are followed throughout the software acquisition life cycle. Compliance with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, and audits. Software development and control processes should include quality assurance approval points, where an SQA evaluation of the product may be done in relation to the applicable standards.* [14]”

Dari pengertian di atas terdapat beberapa poin yang dapat kita perhatikan, (1) PMPL merupakan pendekatan yang terencana dan sistematis -- dalam arti tidak bersifat ad hoc semata -, untuk evaluasi terhadap kualitas dan keterikatan terhadap standar-standar, proses-proses, dan prosedur-prosedur. (2) PMPL terdiri dari proses penjaminan bahwa standar dan prosedur ada dalam pelaksanaan proyek pengembangan perangkat lunak.

4.1 KARAKTERISTIK PENJAMINAN MUTU DALAM *OPEN SOURCE SOFTWARE*

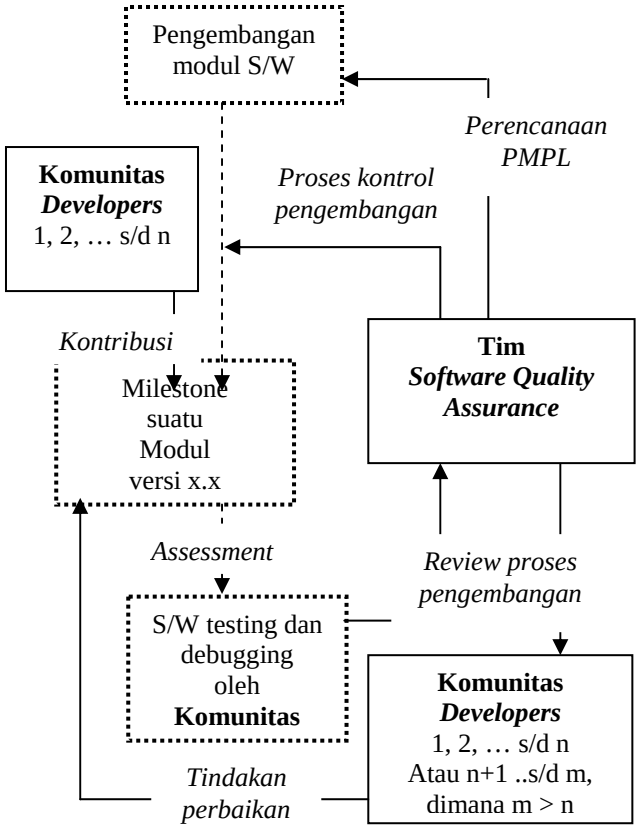
Pengembangan OSS identik dengan proses *versioning* yang cepat, dikembangkan berbagai pihak yang berbeda di seluruh dunia, penggunaan log yang ketat, serta tim pengembang yang solid. Mekanisme pemeliharaan (*maintainance*) perangkat lunak berbasis *open source* dari berbagai belahan dunia dapat dijumpai pada SourceForge [15]. SourceForge.net merupakan situs web pengembangan OSS terbesar di dunia, dimana dalamnya terdapat lebih dari 100,000 proyek dan lebih dari 1,000,000 pengguna yang teregister dengan sumber daya (*resource*) terpusat untuk mengatur proyek, *issues*, komunikasi, dan kode. SourceForge memiliki *repository* terbesar untuk kode dan aplikasi berbasis *open source* yang tersedia di Internet. SourceForge menyediakan hosting gratis bagi proyek-proyek pengembangan OSS. Esensi dari model pengembangan *open source* adalah *rapid creation of solutions* dalam lingkungan yang terbuka dan bersifat kolaboratif. Mekanisme kolaboratif dalam komunitas *open source*, yaitu tim pengembang dan pengguna menjanjikan standar kualitas yang tinggi, dan membantu penjaminan ‘keberlangsungan hidup’ jangka panjang (*long-term viability*) baik dalam hal data dan aplikasi.

Sacha Labourey selaku JBoss General Manager EMEA memiliki pandangan mengenai layanan *open source* (Big Market for Open Source Services) yaitu sebagai berikut [16]:

Pertama, ada anggapan bahwa proses penjaminan kualitas perangkat lunak berbasis *open source* tidak terdefiniskan dengan baik. Proses penjaminan kualitas dalam *open source* berbeda dengan apa yang dilakukan pada pengembangan PS, namun tidak seburuk yang dibayangkan. Pada OSS, kita memiliki versi rilis yang begitu banyak dan banyak masukan berharga yang diperoleh dari pengguna. Bagaimanapun hal ini tidak cukup. Apa yang kita lakukan adalah mengkombinasikan keuntungan atau kelebihan dari *open source* dengan proses pengembangan perangkat lunak yang dilakukan secara profesional, yang kita sebut dengan *Professional Open Source*. Aktivitas ini meliputi *testing*, *training*, dan tahap-tahap pengembangan lainnya yang kita ketahui dari disain PS. Terdapat 4 (empat) cara bagaimana kita mendapatkan proyek; Pertama, kita mengambil proyek yang sudah atau sedang dikembangkan (*existing project*). Kita melakukannya dengan cara mempekerjakan tim pengembang (*developers*), sehingga kita dapat mengontrol kualitas dari proyek. Cara kedua adalah bagaimana memulai proyek dengan cara *scratch in-house*. Cara ketiga adalah terlibat dalam *existing project*. Cara keempat adalah mendapatkan PS dan menjadikannya berbasis OSS.

Kedua, cara mengevaluasi atau melakukan *assessment* kualitas yang pada dasarnya merupakan kontribusi dari komunitas *open source*. Sistem *assessment* kualitas dapat didasarkan pada langkah-langkah melakukan *assessment* terhadap para kontributor untuk menjamin kualitas dari kontribusi yang diberikan. Untuk tiap-tiap proyek yang ditangani, perlu ada seorang pemimpin proyek, yang berwenang untuk menerima atau menolak kontribusi.

Pada Gambar 1 di bawah ini akan ditampilkan contoh mekanisme PMPL dalam pengembangan sebuah modul sebuah OSS.



Gambar 1. Contoh mekanisme PMPL dalam OSS

4.2 KARAKTERISTIK PENJAMINAN MUTU DALAM *PROPRIETARY SOFTWARE*

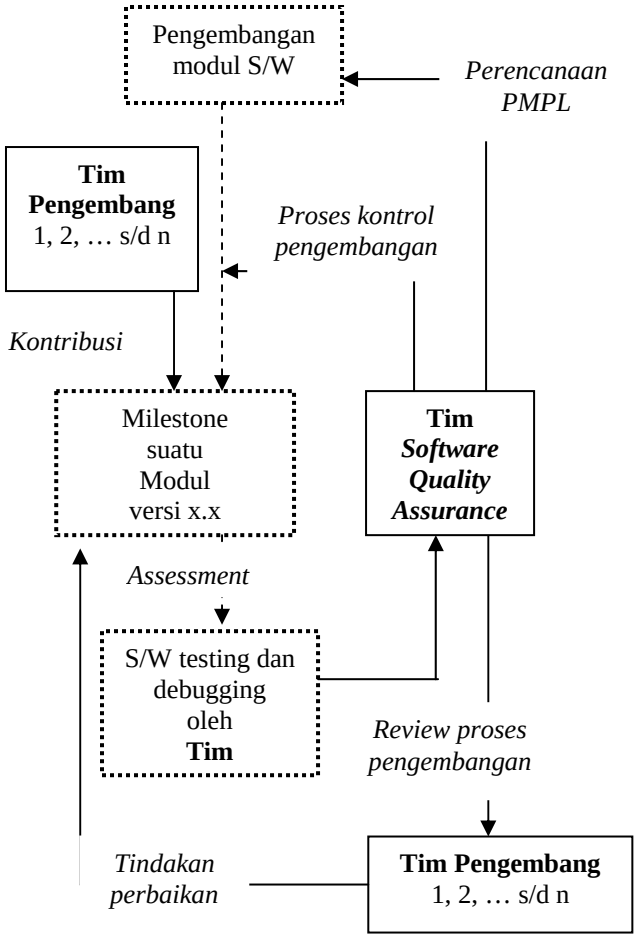
Organisasi yang menyerahkan pengembangan perangkat lunak yang mereka butuhkan kepada vendor eksternal biasanya menggunakan *acceptance test* untuk melakukan validasi terhadap perangkat lunak tersebut. Dalam *process metrics* dan informasi detail untuk melakukan *assessment* terhadap kualitas vendor-vendor perangkat lunak secara umum tidak ditemukan dalam *contracting organizations.* Oleh karenanya, indikator-indikator dan *metrics* yang berguna yang berhubungan dengan *acceptance testing* sangat penting bagi proses *assessment* terhadap perangkat lunak. Beberapa *metrics* akan berbeda dengan *metrics* berbasis kalendar karena *acceptance testing* pada umumnya pendek, dan dimungkinkan juga adanya *multiple code drops,* oleh karena itu, dilakukan *multiple mini acceptance tests* pada proses validasi [17].

Linda H. Rosenberg dalam [18] menyebutkan bahwa terdapat 12 *lessons learned* dalam PMPL, diantaranya adalah (1) Para manajer proyek dan tim pengembang perlu memahami PMPL dan bagaimana proyek yang ditangani mendapatkan manfaat darinya, (2) Implementasi PMPL merupakan *balancing activity* yang harus disesuaikan dengan proyek yang sesuai, (3) SQA harus mengevaluasi proses sebagaimana mengevaluasi produk, (4) Harus terdapat *Software Assurance Plan*, (5) PMPL harus men-*span* keseluruhan *Software Development Life Cycle*, (6) *Requirements*, merupakan awal dari proyek-proyek yang sukses, (7) PMPL tidak sama dengan testing, (8) Metrics adalah kebutuhan (*necessity*), (9) Keamanan dan *reliability* merupakan aspek kritis dari PMPL, (10) Verifikasi independen dan validasi merupakan alat yang penting dalam PMPL, (11) Perangkat keras tidak sama dengan perangkat lunak, dan (12) *Risk management* bukan sebuah pilihan. Yang menarik untuk disimak adalah *lesson learned* ketiga, yang menyebutkan bahwa PMPL harus melakukan evaluasi terhadap proses sebagaimana yang dilakukan untuk melakukan evaluasi terhadap produk. Kunci dari keberhasilan pelaksanaan PMPL terletak pada jaminan dan pemantauan terhadap terlaksananya proses pengembangan yang sesuai dengan metodologi yang digunakan organisasi.

Dalam pengembangan PS, manajer proyek dan tim pengembang memegang peranan penting dalam kesuksesan pengembangan perangkat lunak. Variabel-variabel yang sangat berpengaruh terhadap karakteristik implementasi PMPL dalam PS adalah:

1. Tempo rilis versi perangkat lunak sesuai dengan manajemen proyek yang telah ditetapkan organisasi/perusahaan.
2. Jumlah tim pengembang (*developers*) yang berkontribusi sudah *fixed*.
3. Jumlah tim penguji (*testers* dan *debuggers*) yang berkontribusi sudah *fixed*.

Pada Gambar 2 di bawah ini akan ditampilkan contoh mekanisme PMPL dalam pengembangan sebuah modul sebuah PS.



Gambar 2. Contoh mekanisme PMPL dalam PS

5. PENUTUP DAN KESIMPULAN

Dengan mengamati berbagai karakteristik dari OSS dan PS terlihat perbedaan mekanisme pengembangan kedua jenis produk perangkat lunak tersebut. Hal itu mengakibatkan adanya perbedaaan dalam melakukan implementasi PMPL. OSS dengan karakteristik pengembangan yang melibatkan banyak pihak (berbasis komunitas) dengan kuantitas serta kualitas SDM yang bervariasi serta perubahan versi yang sangat cepat untuk mengimbangi *request* dari pengguna juga inovasi dari *voluntary developers* dari berbagai penjuru dunia membutuhkan mekanisme PMPL yang responsif terhadap perubahan yang cepat. Pihak yang melakukan aktivitas PMPL dapat dilakukan oleh organisasi eksternal. Sedangkan di sisi lain PS yang sangat berorientasi pada pemenuhan *requirements* klien memiliki pendekatan PMPL yang berbeda dengan apa yang dilakukan pada OSS. PS memiliki komposisi SDM yang relatif stabil dan jauh dari perubahan komposisi baik kuantitas maupun kualitas. Selain itu PS memiliki target penyelesaian produk dengan berbagai batasan yang ada seperti biaya dan waktu. Dengan pertimbangan tersebut PMPL untuk PS sangat dimungkinkan untuk dipengaruhi budaya dan etos kerja dalam organisasi yang bersangkutan.

REFERENSI

[1] Avron Barr dan Shirley Tessler, “An Overview of the Software Industry Study”, Stanford Computer Industry Project, 1995, diakses tanggal 2 Januari 2006.

[2] For 1993, International Data Corp. (IDC) via U.S. Industrial Outlook, 1994 dalam [1].

[3] Plunkett Research, Ltd. -Industry Statistics, “Trends and In-depth Analysis of Top Companies, InfoTech, Computers & Software Trends > InfoTech Industry Overview”, http://www.plunkettresearch.com/Industries/InfoTechComputersSoftware/InfoTechComputersSoftwareTrends/tabid/173/Default.aspx, tanggal akses 29 Mei 2006.

[4] Open Source Initiative (OSI), “The Open Source Definition”, http://www.opensource.org/docs/definition.php, diakses 24 November 2006.

[5] David A. Wheeler, “Generally Recognized as Mature (GRAM) OSS/FS programs”, http://www.dwheeler.com/gram.html, diakses 24 November 2006.

[6] Brian Fitzgerald, ”The Transformation of Open Source Software”, MIS Quarterly Vol. 30, pp. 587-598, 2006

[7] J. Feller dan B. Fitzgerald, ”Understanding Open Source Software Development”, Addison-Wesley, London, 2002.

[8] Jatis Solution, “Services“, http://www.jatis.com/content/services.php, diakses 14 Oktober 2006.

[9] Proprietary Software“, ttp://en.wikipedia.org/wiki/Proprietary_software.

[10] IEEE Std 610.12-1990, “Glossary of Software Engineering Terminology”, Institute of Electrical and Electronics Engineers, Inc., 1990 in [16].

[11] ISO 9003-3-1991, “Quality Management and Quality Assurance Standards, Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software”, International Standards Organization, 1991 dalam [16].

[12] Schulmeyer, G. Gordon, dan James I. McManus, “Handbook of Software Quality Assurance 3rd ed.”, Prentice Hall PRT, 1998 dalam [16].

[13] Avison and Fitzgerald, “Information Sytems Development: Methodologies, Techniques and Tools second edition”, McGrawHil International (UK) Limited.

[14] The NASA Data Item Descriptions, “Software Quality Assurance Activities Product evaluation and process SOFTWARE QUALITY ASSURANCE”, http://jira.atlassian.com/secure/attachment/17146/sqa+activities.txt

[15] SourceForge, http://www.sourceforge.net

[16] Sacha Labourey (JBoss General Manager EMEA), “Big market for Open Source services”, http://www.eurescom.de/message/messageMar2006/Interview_with_Sacha_Labourey.asp,

tanggal akses 6 Juni 2006.
[17] Stephen H. Kan, “Metrics and Models in Software Quality Engineering”, Pearson Education, Inc., 2003.
[18] Linda H. Rosenberg (Chief Sci. for Software Assurance Goddard Space Flight Center, National Aeronautics and Space Administration - NASA), “Lessons Learned in Software Quality Assurance”, Volume 6 Number 2 - Software Engineering Education. http://www.softwaretechnews.com/stn6-2/lessons.html, tanggal akses 29 Mei 2006.