STUDI TINJAUAN PERBANDINGAN KIPI DAN CMMI SEBAGAI FRAMEWORK STANDAR KEMATANGAN PENGEMBANGAN INDUSTRI PERANGKAT LUNAK DI INDONESIA

Stanley Karouw
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Sam Ratulangi
Jl. Kampus UNSRAT Bahu, Manado, 95115
Telp : (0431) 852959, Fax : (0431) 823705
E-mail: Stanley.karouw@unsrat.ac.id

Abstrak

Model kematangan kemampuan atau *Capability Maturity Model* adalah salah satu pemicu pengembangan industri telematika, dalam hal standarisasi kemampuan pengembang perangkat lunak. KIPI merupakan suatu standar model kematangan kemampuan yang berlaku di Indonesia. KIPI diserap dari CMMI atau *Capability Maturity Model Integration* yang dikeluarkan oleh *Software Engineering Institute*, *SEI*, dengan penyesuaian tertentu, menurut karakteristik industri telematika dalam negeri. Studi tinjauan perbandingan KIPI dan CMMI sebagai framework standar kematangan pengembangan industri perangkat lunak ini, memaparkan beberapa perbandingan langsung antara CMMI dan KIPI. Perbandingan KIPI dan CMMI merupakan hal penting untuk memahami keterkaitan setiap area kunci proses dalam setiap tahap kematangan dari masing-masing standar ukuran tersebut. Dengan memahami masing-masing area kunci proses, maka implementasi KIPI akan merupakan salah satu faktor penentu tumbuhkembangnya pengembang perangkat lunak dalam negeri menuju terwujudnya visi bangun industri telematika 2020.
Kata Kunci: *Model Kematangan, Rekayasa Perangkat Lunak, Model Proses, Area Kunci Proses, CMMI, KIPI*

1. Pendahuluan

Ide dasar penulisan paper ini didasarkan pada asumsi bahwa *"the quality of the system is highly influenced by the process used to acquire, develop, and maintain the system to produce software"*.[1] Proses memegang peranan signikan dalam mengembangkan perangkat lunak. Proses bukan hanya sekedar suatu kegiatan yang dilakukan dalam pengembangan perangkat lunak, namun juga memiliki peran yang menentukan berhasil tidaknya suatu produk perangkat lunak yang berkualitas dihasilkan.

Berangkat dari pemahaman tersebut diatas, kita menyusun asumsi yang berikutnya, yaitu *"how we manage the software process is the key factor in increasing productivity and profitability"*[1]. Tentu saja, setelah menyadari peranan proses dalam pengembangan perangkat lunak, maka selanjutnya kita diperhadapkan pada ide bagaimana mengkondisikan atau mengatur proses itu sehingga bisa meningkatkan produktivitas dan keuntungan untuk semua yang berperan dalam pengembangan perangkat lunak.

Kedua ide dasar tersebut diatas, menyadarkan kita bahwa, dalam pengembangan perangkat lunak, dibutuhkan suatu proses yang terdefinisi, dilakukan berulang-ulang dan terkuantisasi. Dalam proses, karakteristik unik perangkat lunak sebagai suatu disiplin ilmu rekayasa dan sebagai karya seni diwujudnyatakan. Ini berarti, proses bisa diidentifikasi atau dikenali, setelah dikenali, proses dapat dikualifikasi atau dibedakan, dan selanjutnya proses dilakukan secara terus-menerus guna mendapatkan hasil yang optimal. Dalam implementasi proses, kemudian bisa didapatkan hal-hal yang harus dilakukan untuk memperbaiki proses atau umpan balik, sebagai hasil pembelajaran. Umpan balik ini kemudian digunakan dalam implementasi proses yang selanjutnya. Singkat kata, proses bisa dikembangkan (*improvement of process*) sehingga mencapai tingkat yang paling mendatangkan hasil yang optimal bagi pengembangan perangkat lunak. Inilah yang dimaksudkan dengan tingkat kematangan proses.

Paper ini akan memberikan suatu pengantar tinjauan perbandingan dari kerangka kerja kematangan proses pengembangan industry perangkat lunak, yakni KIPI dan CMMI.

2. Beberapa Pengertian Dasar

2.1 Perangkat Lunak

Dalam artian sempit, perangkat lunak

dianggap sebagai program. Yakni program yang dijalankan pada suatu pemroses. Pada hakikatnya, perangkat lunak, bukan hanya sekedar sebuah program dan/atau program yang dijalankan pada suatu alat pemroses. Perangkat lunak memiliki pengertian yang lebih luas, yakni menyangkut semua hal yang berkaitan dengan dokumentasi program dan konfigurasi data yang diperlukan untuk membuat program bisa dijalankan oleh pemroses.[2]

Perangkat lunak itu sendiri dapat dibedakan atas: 1) Perangkat lunak generik; yakni perangkat lunak yang dikembangkan sendiri oleh pengembang, menurut standar dan ukuran pengembang dan dimaksudkan untuk tujuan penggunaan tertentu oleh pengembang. Perangkat lunak generik ini, biasanya merupakan perangkat lunak *propietary*, yang dijual kepada siapa saja yang mampu membelinya. 2) Perangkat lunak yang disesuaikan; yakni perangkat lunak yang dikembangkan menurut kebutuhan pengguna itu sendiri. Perangkat lunak ini, dibangun menurut apa yang dibutuhkan oleh pengguna, dan digunakan untuk tujuan khusus, yang dimaksudkan oleh pengguna itu sendiri. Para pengembang perangkat lunak jenis ini mengembangkan perangkat lunak menurut kebutuhan spesifik yang ditentukan oleh *user*.[2]

2.2. Rekayasa Perangkat Lunak

Mengembangkan perangkat lunak, didasarkan pada prinsip-prinsip rekayasa[2][3]. Dalam hal ini rekayasa diartikan sebagai upaya untuk menghasilkan produk yang berkualitas, berdasarkan pendekatan sistematik dalam perancangannya dan pengoperasiannya. Produk yang dimaksud untuk dihasilkan adalah perangkat lunak itu sendiri. Pada kenyataannya, pengembangan perangkat lunak juga merupakan suatu hasil karya yang membutuhkan sentuhan imajinasi dan kecerdikan. Karena pada dasarnya perangkat lunak, bukan sekedar merupakan sebuah produk yang terkuantisasi, namun juga merupakan sebuah karya seni. Sebagai sebuah karya seni, pengerjaan perangkat lunak juga memerlukan ketrampilan tertentu, yang seringkali tidak bisa dikuantifikasi. Disinilah letak keunikan pengembangan perangkat lunak.

2.3. Tahap pengembangan Perangkat Lunak

Pengembangan perangkat lunak melibatkan manusia sebagai pelaku utama. Seperti yang dikemukakan diatas, pengembangan perangkat lunak sekarang ini menggunakan prinsip-prinsip rekayasa. Dengan pendekatan rekayasa, perangkat lunak dikembangkan menurut tahap-tahapan tertentu. Tahapan pengembangan perangkat lunak ini dapat dibedakan atas[4]:

1) Fase Pendefinisian; yang berfokus untuk menjawab pertanyaan untuk apa perangkat lunak ini dibangun. Dalam fase ini, kebutuhan (atau biasa disebut *requirements*) perangkat lunak didefinisikan. Secara praktis, informasi apa yang akan diolah oleh perangkat lunak, fungsi/kinerja yang perlu dilakukan, perilaku sistem yang diharapkan, antarmuka yang harus dibuat dan batasan-batasan rancangan perangkat lunak, ditentukan dalam fase ini. Sering fase ini disebut sebagai perencanaan atau spesifikasi.

2) Fase Pengembangan; yang berfokus pada pertanyaan bagaimana mengembangkan perangkat lunak yang dimaksud dalam fase pendefinisian. Keahlian teknis mulai diperlukan dalam fase ini, karena mulai bersentuhan dengan bagaimana data distrukturkan, fungsi-fungsi diimplementasikan, rincian prosedur yang berhubungan dengan perangkat lunak, penerjemahan rancangan ke bahasa pemrograman tertentu. Fase ini juga seringkali termasuk melakukan pengujian. Fase ini disebut analisis dan disain.

3) Fase Implementasi dan Pemeliharaan; berfokus pada digunakannya perangkat lunak dalam lingkungan yang dimaksudkan, perubahan-perubahan penyesuaian menurut kebutuhan yang terjadi, adaptasi yang dibutuhkan dan pemeliharaan yang berkaitan dengan perbaikan secara spesifik ataupun menyeluruh dari perangkat lunak.

2.4. Model Proses Pengembangan Perangkat Lunak

Yang dimaksud dengan model proses pengembangan perangkat lunak adalah deskripsi sederhana dari setiap proses yang dilakukan dalam aktivitas pengembangan. Model proses akan menjelaskan antara lain model alur kerja (yakni menyangkut masukan, keluaran dan keterkaitan yang ada), model aliran data atau aktivitas (yang menggambarkan alur data dan perubahan yang terjadi pada data) dan model peran/aksi (yang menjelaskan keterlibatan manusia selaku subyek pengembang perangkat lunak).[5]

2.5. Aktivitas Pengembangan Perangkat Lunak

Dalam pengembangan perangkat lunak, terdapat beberapa aktvitas yang penting; diantaranya adalah aktivitas teknis/dasar pengembangan, aktivitas penjaminan kualitas perangkat lunak, aktivitas manajemen konfigurasi, aktivitas manajemen proyek pengembangan perangkat lunak dan aktivitas peningkatan kematangan proses dan kemampuan organisasi. Keseluruhan aktivitas pengembangan ini memiliki peran dan fungsi tersendiri dalam mengembangkan perangkat lunak. [5]

2.6. Metode, Tools dan Proses dalam Pengembangan Perangkat Lunak

Guna mencapai kualitas produk yang diinginkan maka pengembangan perangkat lunak dilakukan dengan menggunakan metode, tools dan proses. Metode merupakan tata cara atau prosedur teknis untuk membangun perangkat lunak. Metode menyangkut seluruh aktivitas dalam rekayasa perangkat lunak, dimulai dari penentuan kebutuhan (untuk apa perangkat lunak akan dikembangkan?), analisis dan perancangan (bagaimana membangun perangkat lunak agar sesuai dengan yang dibutuhkan?), pengujian dan implementasi.[5]

Tools merupakan dukungan otomatis, semi otomatis dan non otomatis yang digunakan dalam aktivitas pengembangan perangkat lunak. Misalnya CASE atau Komputer Aided Software Engineering. CASE menggabungkan beberapa komponen inti yang berperan penting dalam kegiatan pengembangan misalnya perangkat lunak, perangkat keras dan basis data rekayasa perangkat lunak yang berisi informasi/dokumen analisis, rancangan, kode sumber dan pengujian).[5]

Proses merupakan lapisan dasar rekayasa perangkat lunak. Proses dapat dipahami sebagai sekumpulan tindakan atau aktivitas (yaitu metode) yang dilakukan pengembang (manusia) dengan menggunakan tools tertentu dalam mengembangkan perangkat lunak. Proses merupakan fungsi interaksi antara manusia, metode dan tools guna menghasilkan perangkat lunak yang berkualitas.[5]

2.7. Model Kematangan Kemampuan

Pengertian model kematangan kemampuan mengacu kepada suatu peta jalan atau kerangka kerja yang menjadi acuan, guna mencapai suatu tujuan, dalam pengembangan perangkat lunak. Dalam dalam melakukan kegiatan pengembangan, tingkat kemampuan kematangan sistem bisa diukur melalui model ini. Ukuran kematangan ini bukan hanya sekedar program proses, meskipun dari sudut pandang informal, bisa dianggap demikian. Ukuran kematangan juga dapat dianggap sebagai suatu ukuran pengembangan sistem.[6]

2.8. Area Kunci Proses

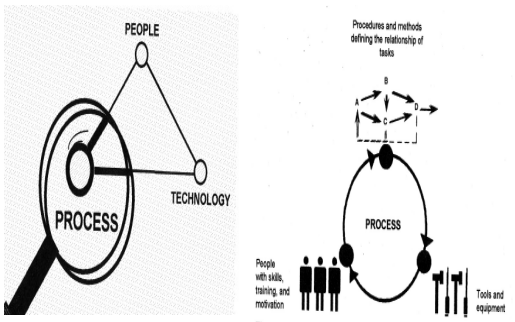
Area kunci proses merupakan sekumpulan acuan yang digunakan untuk mendefinisikan setiap proses yang harus dilakukan untuk mencapai suatu tingkat kematangan tertentu.[6]

3. Proses sebagai Faktor Krusial dalam Pengembangan Perangkat Lunak

Proses sebagai lapisan dasar pengembangan perangkat lunak mendefinisikan kerangka kerja atau serangkaian acuan aktivitas tertentu yang menjadi kunci pokok dalam melakukan kegiatan pengembangan. Sekumpulan bidang kunci pokok ini disebut Kunci Pokok Proses atau Key Process Area.

Proses rekayasan perangkat lunak, menunjukkan bahwa untuk menghasilkan produk perangkat lunak yang berkualitas, maka harus melewati serangkaian tahapan, aktivitas dalam beberapa bidang kunci dengan tujuan spesifik tertentu, yang bisa diprediksi. Keseluruhan urutan tahapan proses kunci ini dapat dianggap sebagai peta jalan (roadmap) yang dapat membantu manusia selaku pelaku pengembang untuk menghasilkan produk perangkat lunak yang berkualitas. Peta jalan inilah yang disebut sebagai "model proses pengembangan perangkat lunak".[6][7][8]

Proses bisa dipandang sebagai salah satu titik dalam kegiatan pengembangan perangkat lunak. Proses juga bisa dianggap sebagai perekat semua unsur-unsur lain dalam kegiatan pengembangan perangkat lunak lainnya. Gambar 1 (sisi kiri) memandang proses sebagai salah satu bagian atau unsur (*as part of*) dalam tahapan pengembangan perangkat lunak. Proses dipandang sebagai suatu aktivitas/kegiatan, dimana tujuannya atau hasilnya yang ditekankan.



Gambar 1 Pengertian Proses

Gambar 1 cenderung melihat proses sebagai suatu fungsi. Proses menyatukan manusia, sebagai subyek pengembang dan metode, tools dan teknik sebagai alat yang membantu mengoptimalkan kinerja. Dari sudut pandang ini, proses dapat dianggap sebagai suatu perekat (*enabler*; unsur yang memungkinkan), yang menyatukan keseluruhan kegiatan pengembangan. Fungsi dari proses itu sendiri yang ditekankan pada gambar ini.

Tidak menjadi persoalan, bagaimana kita memandang tempat proses itu sendiri dalam kegiatan pengembangan perangkat lunak. Yang pasti, proses merupakan suatu hal yang tidak bisa diabaikan. Karena dengan berkomitmen pada proses, maka pengembangan perangkat lunak dapat menghasilkan produk yang berkualitas.

4. CMMI sebagai standar ukuran kematangan

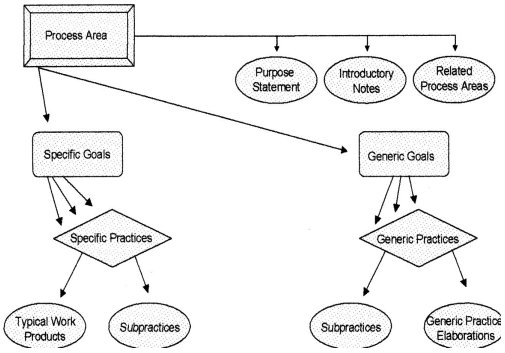
4.1.Sejarah CMM ke CMMI

CMMI, yang pada awalnya disebut CMM (Capability Maturity Model) sebagai ukuran standar kematangan pengembangan perangkat lunak memiliki sejarah panjang, sebelum diterima secara global. Diawali oleh Walter Shewhart di tahun 1930, yang memulai penelitian tentang perbaikan proses dengan metode kontrol kualitas statistik. Yang kemudian semakin diperluas oleh W. Edwards Deming, Philip Crosby dan Joseph Juran di era 80-an. Watts Humprey, Ron Radice dan lainnya semakin mengembangkan penelitian ini, melalui serangkaian implementasi di IBM dan SEI [6][8][9]. CMM kemudian mulai dikembangkan, hingga akhirnya diakui sebagai salah satu standar ukuran kematangan kapabilitas pengembang perangkat lunak. Apalagi sejak DOD (Departement of Defense) Pemerintah Amerika Serikat, mensyaratkan bahwa setiap pengembang perangkat lunak yang mendapatkan proyek dalam lingkungan DOD, harus memiliki tingkat kematangan CMM level 3, perkembangan CMM semakin mendunia.

Dalam perkembangan yang selanjutnya, selama kurun waktu 20 tahun lebih, dan semakin banyak perusahaan pengembang perangkat lunak yang menunjukkan hasil yang signifikan akibat penggunaan CMM, maka, semakin banyak pula perusahaan yang mencoba menerapkan skema CMM dalam mendukung proses bisnis perusahaan. Penggunaan CMM pun semakin meluas, bukan saja pada sebatas industri perangkat lunak, tapi semakin meluas pada industri lainnya. Oleh karena itu, SEI pun mulai mengembangkan suatu model standar ukuran kematangan yang baru, yang bisa diterapkan kepada seluruh industri, lahirlah yang dinamakan CMMI atau Capability Maturity Model Integration.[9]

4.2 Skema CMMI

CMMI[10][11] pada dasarnya merupakan sebuah konstelasi yang terdiri atas CMMI for Development (CMMI-DEV), CMMI for Acquisition (CMMI – ACQ) dan CMMI for Services (CMMI-SVC). Dalam perkembangan selanjutnya, ketiga konstelasi ini kemudian digabungkan menjadi CMMI saja, dengan 5 tahap kematangan dan mengadopsi 22 area kunci proses. 5 tahap kematangan CMMI adalah tahap 0 disebut Incomplete, tahap 1 disebut Performed, tahap 2 disebut Managed, tahap 3 disebut Defined, tahap 4 disebut Quantitatively Managed, tahap 5 disebut Optimizing.[9]
Gambar dibawah ini menunjukkan komponen area kunci proses CMMI.



Gambar 2 Area Kunci Proses CMMI

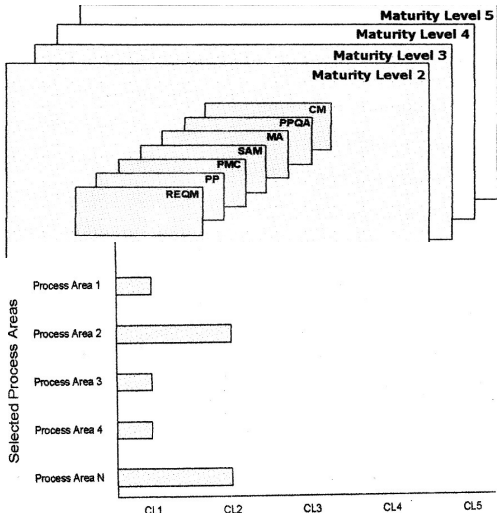
Area kunci proses dalam CMMI adalah Reqirements Management (REQM), Project Planning (PP), Project Monitoring and Control (PMC), Supplier Agreement Management (SAM), Process and Product Quality Assurance (PPQA), Configuration Management (CM), Measurement and Analysis (MA), Organizational Process Focus (OPF), Organizational Process Defintion (OPD), Organizational Training (OT), Integrated Project Management (IPM), Risk Management (RSKM), Product Integration (PI), Requirements Development (RD), Technical Solution (TS), Validation (VAL), Verification (VER), Decision Analysis dan Resolution (DAR), Quantitative Project Management (QPM), Organizational Process Performance (OPP), Causal Analysis and Resolution (CAR), Organizational Innovation and Deployment (OID). Masing-masing area kunci proses, memiliki tujuan yang harus dicapai.

Setiap Area Kunci Proses memiliki *purpose statement, introductory notes* dan r*elated process Areas*. *Purpose Statement* menjelaskan tujuan yang hendak dicapai dalam area kunci proses tersebut. *Introductory Notes* menjelaskan konsep umum yang melatarbelakangi area kunci proses yang dimaksud. Sedangkan untuk *Related Process Area* menjelaskan keterkaitan setiap area kunci proses yang ada.

Selanjutnya, untuk setiap area kunci proses, memiliki *specific goals* dan *generic goals*. Pengertian *specific goals* cenderung kepada tujuan khusus yang ingin dicapai dalam implementasi area kunci proses tersebut. Sedangkan untuk *generic goals* memiliki pengertian kepada tujuan

yang lebih umum. Dari *specific goals* diturunkan *specific practices*, yang akan menjadi panduan manual atau operasional, yang harus dilaksanakan untuk mencapai setiap tujuan spesifik/khusus yang terkait. Begitu pula dengan *generic practices*, merupakan penerapan operasional yang harus dilakukan dalam mencapai *generic goals*. Untuk setiap *generic goals* dan *specific goals* memiliki *generic practice elaborations, typical work products* dan *subpratices*. *Typical works products* merupakan daftar contoh keluaran atau output dari setiap *specific practices*. Sedangkan *generic practice elaborations* menjelaskan panduan bagaimana melaksanakan *generic practices* pada area kunci proses yang berkaitan. Untuk *subpractices* (pada *specific practices* dan *generic practices*) menjelaskan panduan untuk mengartikan dan melaksanakan *specific* dan *generic practices* terkait.

Satu hal yang harus dicermati, implementasi CMMI tru dilakukan dengan 2 cara. Yang pertama disebut *Continous Representation*; implementasi representasi kontinyu tru diartikan sebagai menerapkan CMMI menurut fungsi masing-masing area kunci proses. Pendekatan ini bersifat fleksibel, menyesuaikan menurut kebutuhan truct yang ingin dikembangkan terlebih dahulu. Pendekatan yang kedua disebut *Staged Representation*; merupakan implementasi representasi bertahap. Pendekatan bertahap ini, merupakan pendekatan yang tructural, didasarkan menurut setiap urutan tahap kematangan (dengan pasangan area kunci proses yang terkait). Gambar 3 menjelaskan tentang perbandingan *Continous Representation* (atas) dan *Staged Representation* (bawah) [9]



Gambar 3 Perbandingan *Continous Representation* dan *Staged Representation*

Hal yang dapat kita garis bawahi adalah CMMI bukan hanya sekedar acuan kerangka kerja atau roadmap untuk menajamkan peranan proses dalam pengembangan perangkat lunak. Utilitas dan peranan CMMI, jauh lebih besar dari sekedar program untuk memastikan dan menajamkan proses. Lebih dari itu, CMMI merupakan program untuk perbaikan yang terus-menerus (*continuous improvement programme*). Perbaikan yang terus-menerus merupakan tujuan yang ingin dicapai dalam impelementasi CMMI. Bukan hanya sekedar mendapatkan sebuah pengakuan, melainkan suatu pembentukan budaya baru dalam pengembangan perangkat lunak, yakni budaya perbaikan yang terus-menerus. Hal ini berarti, dalam mengimplementasi CMMI, diperlukan suatu perubahan pola pikir yang signifikan. Pola pikir yang diperbaharui dapat melahirkan suatu *awareness* baru, dan dengan adanya *awareness* tersebut, implementasi CMMI bisa berlangsung terus-menerus. Pada akhirnya, hasil *improvement* akan dikenali dan terukur, dan terutama memiliki pengaruh signifikan dalam menghasilkan produk yang berkualitas.

5. KIPI sebagai ukuran kematangan pengembang perangkat lunak di Indonesia

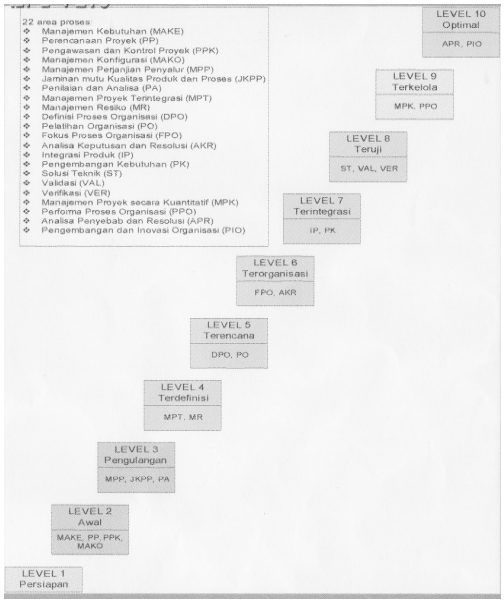
KIPI merupakan jawaban pemerintah untuk melindungi pertumbuhan industri pengembang perangkat lunak dalam negeri. Seperti kita ketahui bersama, pemerintah telah menetapkan pengembangan industri telematika sebagai salah satu pilar utama (diantara dua pilar lainnya) pada program bangun industri masa depan tahun 2020.[11][12] Pilar sektor industri telematika, dibagi dalam 6 sub bagian, yaitu industri software, industri komputer, industri peralatan komunikasi, industri animasi, industri multimedia dan industri peripheral. Pemerintah berkepentingan untuk mengembangkan industri telematika menjadi salah satu industri andalan di dalam negeri dan terdepan dalam lingkup internasional. Hal ini dikarenakan pemerintah menganggap bahwa dengan menumbuhkembangkan industri telematika, tujuan negara, untuk menciptakan kesejahteraan bagi seluruh masyarakat Indonesia dapat tercapai, sesuai yang diamanatkan dalam Undang-Undang Dasar ‘45. Penciptaan kondisi yang kondusif bagi tumbuh kembangnya industri telematika, ditempuh dengan cara membuat kebijakan publik, yang mana, dengan adanya kebijakan publik tersebut, suasana dan kondisi yang kondusif bisa memicu pertumbuhan industri telematik itu sendiri. Beberapa faktor krusial dalam kebijakan publik pengembangan industri telematika ini adalah iklim usaha, standarisasi, daya saing, kluster dan kompetensi. Dalam pemahaman standarisasi inilah sehingga

pemerintah memandang perlu untuk menetapkan suatu ukuran standar kematangan industri perangkat lunak dalam negeri, ukuran standar inilah yang dinamakan KIPI (Kematangan Industri Perangkat Lunak Indonesia).

Seperti kita ketahui bersama, dalam memandang bangun industri 2020, pilar industri telematika merupakan salah satu yang akan mendukung tumbuh kembang industri dalam negeri. Ini berarti, menuju 2020, pemerintah akan melakukan pembangunan baik menyangkut infrastruktur, pemanfaatan dan pendayagunaan, yang akan mendukung pencapaian visi 2020 tersebut. Hal ini berarti, proyek pengembangan yang berhubungan dengan industi telematika akan dilakukan secara kontinyu dan komprehensif oleh pemerintah. Atas dasar pemikiran inilah, pemerintah merasa perlu untuk melibatkan pengembang industri telematika dalam negeri. Karena tidak bisa dipungkiri, dalam era informasi sekarang ini, keinginan pengembang luar negeri untuk turut serta dalam setiap proyek-proyek telematika tidak bisa ditolak oleh pemerintah. Sebagai bagian dari masyarakat internasional yang turut serta dalam perdagangan global, pemerintah pun harus membuka diri, dalam keterlibatan pihak luar negeri. Namun disatu sisi, pemerintah pun merasa perlu untuk mengembangkan pengembang dalam negeri, karena pengembang dalam negeri inilah yang akan menjadi ujung tombak pendukung kesinambungan pembangunan industri telematika di masa depan. Ini berarti, pengembang dalam negeri harus diberi kesempatan yang seluas-luasnya untuk bertumbuh dan mengembangkan dirinya. Pengembangan ini membutuhkan proses. Yang namanya proses pengembangan, dibutuhkan lingkungan yang mendukung, dan juga membutuhkan waktu untuk belajar. Inilah yang harus diciptakan oleh pemerintah bagi pelaku pengembang industri telematika dalam negeri. Melahirkan KIPI sebagai standarisasi kematangan industri perangkat lunak dalam negeri merupakan salah satu kebijakan strategis pemerintah untuk melindungi tumbuh kembangnya pengembang industri perangkat lunak dalam negeri.

6. Perbandingan KIPI dan CMMI

Jika kita membandingkan KIPI dan CMMI, maka ada beberapa hal yang perlu kita cermati. KIPI memiliki 10 tahap kematangan sedangkan CMMI hanya memiliki 6 tahap kematangan. Tahap-tahap kematangan dalam KIPI (pada Gambar 4) adalah: Tahap 1 Persiapan, Tahap 2 Awal, Tahap 3 Pengulangan, Tahap 4 Terdefinisi, Tahap 5 Terencana, Tahap 6 Terorganisasi, Tahap 7 Terintegrasi, Tahap 8 Teruji, Tahap 9 Terkelola, Tahap 10 Optimal.



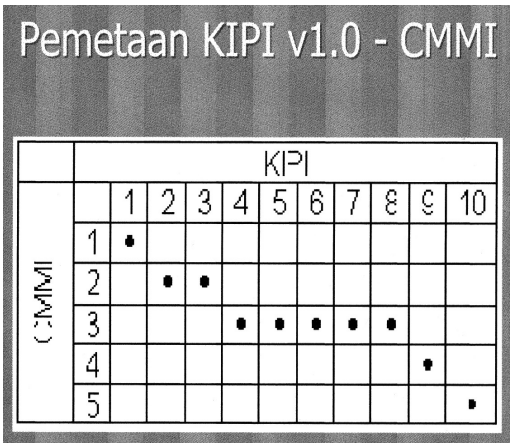
Gambar 4 KIPI versi 1.0

Tabel 1 berikut adalah penyebaran *key proses* dalam setiap Tahap Kematangan KIPI.

Tabel 1 Penyebaran *Key Process* dalam KIPI

Tahap Kematangan	Key Proses
Tahap 1 PERSIAPAN	---
Tahap 2 AWAL	Manajemen Kebutuhan Perencanaan Proyek Pengawasan dan Kontrol Proyek Manajemen Konfigurasi
Tahap 3 PENGULANGAN	Manajemen Perjanjian Penyalur Jaminan Mutu Kualitas Produk dan Proses Penilaian dan Analisa
Tahap 4 TERDEFINISI	Manajemen Proyek Terintegrasi Manajemen Resiko
Tahap 5 TERENCANA	Definis Proses Organisasi Pelatihan Organisasi
Tahap 6 TERORGANISASI	Fokus Proses Organisasi Analisa Keputusan dan Resolusi
Tahap 7 TERINTEGRASI	Integrasi Produk Pengembangan Kebutuhan
Tahap 8 TERUJI	Solusi Teknik Validasi Verifikasi
Tahap 9 TERKELOLA	Manajemen Proyek secara Kuantitatif Performa Proses Organisasi
Tahap 10 OPTIMAL	Analisa Penyebab dan Resolusi Pengembangan dan Investasi Organisasi

Sedangkan pada Gambar 5, kita dapat melihat matriks perbandingan setiap tahap KIPI dan CMMI.

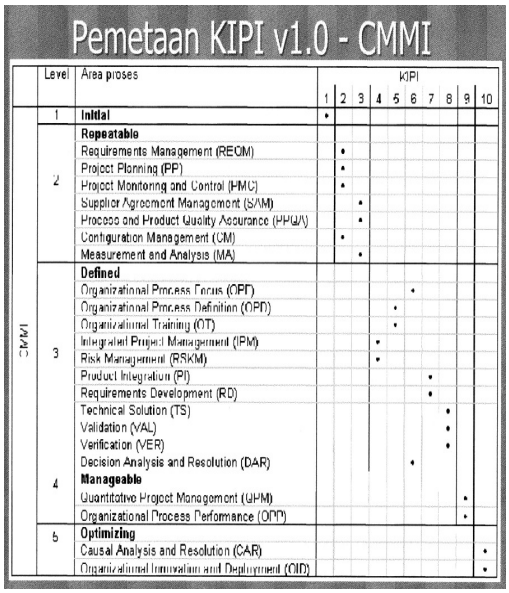


Gambar 5 Matriks Perbandingan KIPI dan CMMI

Gambar 5 menjelaskan bagaimana setiap *key proses area* yang ada dalam CMMI dipetakan dalam KIPI. Untuk tahap 1, KIPI dan CMMI memiliki 1 area kunci proses. Untuk tahap 2 CMMI, dipetakan menjadi KIPI tahap 2 dan 3. Untuk tahap 3 CMMI dipetakan menjadi KIPI tahap 4, 5, 6, 7, 8. Untuk CMMI tahap 4 dipetakan menjadi KIPI tahap 9 dan untuk CMMI tahap 5 dipetakan menjadi KIPI tahap 10. Ini menunjukkan bahwa, secara kuantitatif, KIPI sebenarnya memiliki standar ukuran kematangan yang sama dengan CMMI. Yang menjadi perbedaan hanyalah penyebaran level 3 pada CMMI menjadi 5 level pada KIPI (yakni tahap 4, 5, 6, 7, 8).

Mengapa KIPI memilih untuk menyebarkan 1 tahap CMMI (level 3) menjadi 5 tahap pada KIPI. Ini tentunya merupakan pertimbangan implementasi. Kita ketahui bersama, tahap 3 CMMI merupakan tahap yang memiliki area kunci proses yang cukup sulit untuk diimplementasikan, terlebih oleh para pengembang perangkat lunak dalam negeri. Strategi penyebaran tahap yang seperti ini, bisa menguntungkan para pengembang perangkat lunak dalam negeri, karena para pengembang akan memiliki cukup waktu untuk berkonsentrasi melaksanakan setiap area kunci proses dari setiap level. Dapat juga disebut bahwa, KIPI cenderung mendefinisikan CMMI level 3 menjadi CMMI level 3 awal (KIPI level 4 dan 5), CMMI level 3 menengah (KIPI level 6 dan 7), CMMI level 3 lanjut (KIPI level 8).

Gambar 6 menunjukkan bahwa keseluruhan area kunci proses yang dimiliki oleh CMMI, dipetakan semuanya tanpa terkecuali. Tidak ada satupun area kunci proses yang terlewati. KIPI mengadopsi CMMI secara langsung. Dari sudut pandang ini, dapat kita katakan, bahwa sebenarnya standar kematangan KIPI itu relatif tidak berbeda dengan CMMI. Model pemetaan yang seperti ini memberi asumsi, bahwa sekiranya setiap pengembang perangkat lunak, benar-benar mengimplementasikan semua area kunci proses dari KIPI, maka itu berarti pula, telah sesuai dengan CMMI. Kita tahu bersama, setiap *key proses* dalam CMMI, yang menjadi patokan ukuran kematangan, merupakan hasil penelitian dan pengembangan yang dilakukan lebih dari 1 dasawarsa (sejak masa CMM dimulai). Tentu saja, *key proses* yang telah diuji sekian lamanya, memiliki signifikansi pembuktian yang kuat dalam pengembangan perangkat lunak. Model pemetaan KIPI yang seperti ini, memberikan keuntungan tersendiri bagi para pengembang perangkat lunak dalam negeri.



Gambar 6 Pemetaan KIPI dengan CMMI untuk setiap Area Kunci Proses

7. Beberapa tantangan dan peluang

Ada beberapa tantangan[11][12] yang dihadapi KIPI sebagai suatu standar kematangan perangkat lunak. Sebagai suatu standar yang baru, dan yang dikeluarkan oleh pemerintah, KIPI harus telebih dahulu dikenali (*recognized*) oleh semua pihak yang berperan dalam pengembangan industri telematika dalam negeri. Bila tidak dikenali, maka KIPI akan sulit untuk mendapatkan pengakuan (*acknowledge*) sebagai suatu standar. Dan akhirnya, bila tidak diakui, maka KIPI tidak akan diimplementasikan (*implemented*) oleh para pengembang perangkat lunak dalam negeri.

Pemerintah, selaku pihak yang mensponsori lahirnya KIPI, harus bisa membuat suatu kebijakan strategis yang dapat membuat KIPI, sebagai suatu standar kematangan, dikenali oleh pengembang perangkat lunak. Langkah awal yang baik adalah dengan membangun kelembagaan KIPI. Dalam artian, KIPI harus dilembagakan pada dalam industri telematika. Kelembagaan ini bukan berarti membentuk suatu lembaga tersendiri. Kelembagaan bisa dilakukan dengan membuat aturan-aturan pelaksaanan sebgai payung hukum adanya KIPI. Selanjutnya membentuk komunitas yang bisa mengoptimalkan implementasi KIPI, seperti menunjuk badan penyelenggara pemberian sertifikasi KIPI.

Tantangan yang menghadang di depan juga, adalah perlunya menyiapkan sumber daya manusia, yang akan berperan sebagai assesor atau penilai. Kita tahu bersama, sebagai suatu standar, KIPI harus memiliki tim penilai, yang akan menentukan apakah sebuah pengembang perangkat lunak, telah melakukan setiap area kunci proses. Tim penilai juga harus bisa menentukan pada tingkat mana suatu pengembang perangkat lunak berada. Ketersediaan sumber daya manusia sangatlah signifikan dalam implementasi KIPI.

Untuk para pengembang perangkat lunak, menghadapi suatu aturan main yang baru dalam industri telematika Indonesia. Aturan ini, tentu saja, harus mampu dilihat sebagai suatu peluang, bukan hambatan. Kesiapan pengembang menjadi sesuatu yang krusial. Impelementasi akhir akan dilakukan oleh para pengembang, ini berarti, pada akhirnya, para pengembanglah yang harus mengenali, mengakui dan menerapkan KIPI dalam proses bisnis mereka sendiri. Peluang ini merupakan suatu kesempatan emas untuk mengembangkan diri.

8. Kesimpulan

1. Kelahiran KIPI sebagai suatu ukuran standar kematangan industri pengembang perangkat lunak di Indonesia merupakan suatu langkah maju dalam menumbuhkembangkan industri perangkat lunak dalam negeri. Hal ini sudah selaras dengan program pemerintah untuk membangun industri telematika tahun 2020.
2. Implementasi KIPI dalam waktu dekat, lebih ditekankan untuk melindungi pertumbuhan industri pengembang perangkat lunak Indonesia. Sehingga, usaha ini harus ditindaklanjuti dengan implementasi yang tepat sasaran. Dukungan semua pihak yang berkepentingan dalam pengembangan industri telematika dalam negeri mutlak diperlukan.
3. KIPI memiliki keunggulan yang unik. Karena, secara prinsip, KIPI mengadopsi semua area proses kunci yang dimiliki oleh CMMI – sebagai suatu ukuran standar kematangan yang berlaku internasional. Hal ini berarti, kematangan yang diharuskan dimiliki oleh pengembang perangkat lunak dalam negeri juga akan berlaku secara internasional apabila benar-benar diimplementasikan dengan baik.
4. Guna mengoptimalkan implementasi KIPI, pemerintah sebaiknya menerapkan aturan yang jelas dan tegas. Hal ini dimaksudkan agar supaya, KIPI bisa dikenali (*recognized*), diakui (*acknowledged*) dan diterapkan (*implemented*) oleh para pengembang perangkat lunak dalam negeri dengan baik.
5. Implementasi KIPI, sebagai suatu ukuran kematangan, harus dipandang sebagai kerangka kerja menuju proses peningkatan (*improvement*), bukan hanya sekedar kepatuhan kepada peraturan pemerintah. Hal ini tentu saja, harus menjadi perhatian para pengembang perangkat lunak dalam negeri sendiri.

Referensi

[1] Watt S. Humprey, *Managing the Software Process*, SEI Series in Software Engineering, Addison Wesley Longman, August 1990

[2] Ian Sommerville, *Software Engineering*, 8th ed, Pearson Education Limited, 2007

[3] Roger Pressman, *Software Engineering, A Practitioner's Approach,* 6th ed, McGrawHill International Edition, 2005

[4] Stephen Schach, *Object Oriented Software Engineering*, International Edition, 2008

[5] Bambang Harijanto, *Rekayasa Sistem Berorientasi Objek,* Penerbit Informatika Bandung, 2004

[6] [Kim Caputo, *CMM Implementation Guide, Choreographing Software Process Improvement*, Addison Wesley, 3rd Ed, June 1999.

[7] James Persse, *Project Management Succes with CMMI, Seven CMMI Process Areas*, Pearson Education, Inc. June 2007.

[8] *Capability Maturity Model Integration (CMMI®) version 1.2 Overview*. Diunduh dari www.sei.cmu.edu, Selasa, 18 Maret 2008, pukul 20.00 wib.

[9] Technical Report CMU/SEI-93-TR-024, ESC-TR-03-177, Februari 1993, *Capability Maturity Model for Software version 1.1.* Diunduh dari www.sei.cmu.edu, pada hari Selasa, 18 Maret 2008, pukul 20.00 wib.

[10] Technical Report CMU/SEI-2006-TR-008,

ESC-TR-2006-008, Agustus 2006. *CMMI for Development version 1.2, Improving processes for better products*, CMMI Product Team. Diunduh dari www.sei.cmu.edu, pada hari Selasa, 18 Maret 2008, pukul 20.00 wib.

[11] Eko K. Budiardjo, *Handouts Kuliah Proses dan Manajemen Rekayasan Perangkat Lunak*.

[12] Direktur Industri Telematika – Ditjen Industri Alat Transportasi dan Telematika *Handouts Kuliah Umum Manajemen dan Rekayasa Softwar*e, Jakarta, 13 Mei 2008.