Meluruskan Salah Kaprah Rekayasa Perangkat Lunak

Romi Satria Wahono
romi@romisatriawahono.net
http://romisatriawahono.net

Rekayasa Perangkat Lunak yang merupakan terjemahan dari terminologi *Software Engineering*, sedikit mengalami pergeseran makna di realita dunia industri, bisnis, pendidikan maupun kurikulum Teknologi Informasi (TI) di tanah air. Di industri, tester dan programmer sering salah kaprah menyandang gelar *Software Engineer*. SMK di Indonesia juga latah dengan membuka jurusan Rekayasa Perangkat Lunak, meskipun secara kurikulum hanya mengajari bahasa C atau Pascal (mungkin lebih pas disebut jurusan pemrograman komputer) ;) Tulisan ini berusaha menyegarkan kembali pemahaman kita tentang apa itu Rekayasa Perangkat Lunak (*Software Engineering*) berdasarkan kesepakatan, acuan, dan standard yang ada di dunia internasional.

Sejarah munculnya Rekayasa Perangkat Lunak sebenarnya dilatarbelakangi oleh adanya krisis perangkat lunak (*software crisis*) di era tahun 1960-an. Krisis perangkat lunak merupakan akibat langsung dari lahirnya komputer generasi ke 3 yang canggih, ditandai dengan penggunaan *Integrated Circuit* (IC) untuk komputer. Performansi hardware yang meningkat, membuat adanya kebutuhan untuk memproduksi perangkat lunak yang lebih baik. Akibatnya perangkat lunak yang dihasilkan menjadi menjadi beberapa kali lebih besar dan kompleks. Pendekatan informal yang digunakan pada waktu itu dalam pengembangan perangkat lunak, menjadi tidak cukup efektif (secara cost, waktu dan kualitas). Biaya hardware mulai jatuh dan biaya perangkat lunak menjadi naik cepat. Karena itulah muncul pemikiran untuk menggunakan pendekatan *engineering* yang lebih pasti, efektif, standard dan terukur dalam pengembangan perangkat lunak.

Dari berbagai literatur, kita dapat menyimpulkan bahwa Rekayasa Perangkat Lunak adalah:

> *Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal requirement capturing (analisa kebutuhan pengguna), specification (menentukan spesifikasi dari kebutuhan pengguna), desain, coding, testing sampai pemeliharaan sistem setelah digunakan.*

Kalimat ”seluruh aspek produksi perangkat lunak” membawa implikasi bahwa bahwa Rekayasa Perangkat Lunak tidak hanya berhubungan dengan masalah teknis pengembangan perangkat lunak tetapi juga kegiatan strategis seperti manajemen proyek perangkat lunak, penentuan metode dan proses pengembangan, serta aspek teoritis, yang kesemuanya untuk mendukung terjadinya produksi perangkat lunak.

Kemudian tidak boleh dilupakan bahwa secara definisi perangkat lunak tidak hanya untuk program komputer, tetapi juga termasuk dokumentasi dan konfigurasi data yang berhubungan yang diperlukan untuk membuat program beroperasi dengan benar. Dengan definisi ini otomatis keluaran (*output*) produksi perangkat lunak disamping program komputer juga dokumentasi lengkap berhubungan dengannya. Ini yang kadang kurang dipahami oleh pengembang, sehingga menganggap cukup memberikan program yang jalan (*running program*) ke pengguna (*customer*).

Rekayasa Perangkat Lunak bukan merupakan cabang ilmu *Computer Science* yang mempelajari tentang *technical coding*. Ini yang sering salah kaprah dipahami, sehingga pelajar, mahasiswa atau bahkan calon dosen ;) shock ketika dihadapkan dengan buku-buku textbook Rekayasa Perangkat Lunak yang selalu tebal dengan penjelasan sangat luas tentang bagaimana perangkat lunak diproduksi, dari aspek requirement capturing, desain, arsitektur, testing, kualitas software, sampai people/cost management. Dan ini adalah suatu kesepakatan yang sudah diterima umum tentang Rekayasa Perangkat Lunak, sejak jaman Roger S Pressman menulis buku ”*Software Engineering: A Practitioner's Approach*”, sampai Ian Sommerville yang kemudian datang dengan buku ”*Software Engineering*” yang sudah sampai edisi ke 7, maupun pendatang baru semacam Hans Van Vliet, Shari Lawrence Pfleeger maupun James F Peters.

Terus bagaimana kalau kita ingin memperdalam masalah *technical coding* dan *programming*? Ada dua cabang ilmu lain yang membahas lebih dalam masalah ini, yaitu: Algoritma dan Struktur Data, dan Bahasa Pemrograman.

Kok bisa begitu, dasarnya darimana? Jadi pada hakekatnya, sebagai sebuah disiplin ilmu, *Computer Science* itu juga memiliki definisi, ruang lingkup, klasifikasi dan kategorisasinya. Klasifikasi yang paling terkenal dikeluarkan *Task Force* yang dibentuk oleh IEEE (*Institute of Electrical and Electronics Engineers*) dan ACM (*Association for Computing Machinary* (http://acm.org)) yang dipimpin oleh Peter J Denning, yang kemudian terkenal dengan sebutan Matriks Denning. Sangat jelas bahwa Matriks Denning memisahkan antara cabang ilmu *Software Engineering* dengan Algoritma dan Struktur Data, serta Bahasa Pemrograman. Itulah di paragraf awal saya sebut bahwa lebih tepat SMK, akademi atau universitas menggunakan nama jurusan (atau mata kuliah): Pemrograman Komputer, Algoritma dan Struktur Data, atau Bahasa Pemrograman, kalau memang materinya hanya mempelajari masalah bahasa pemrograman secara teknis.

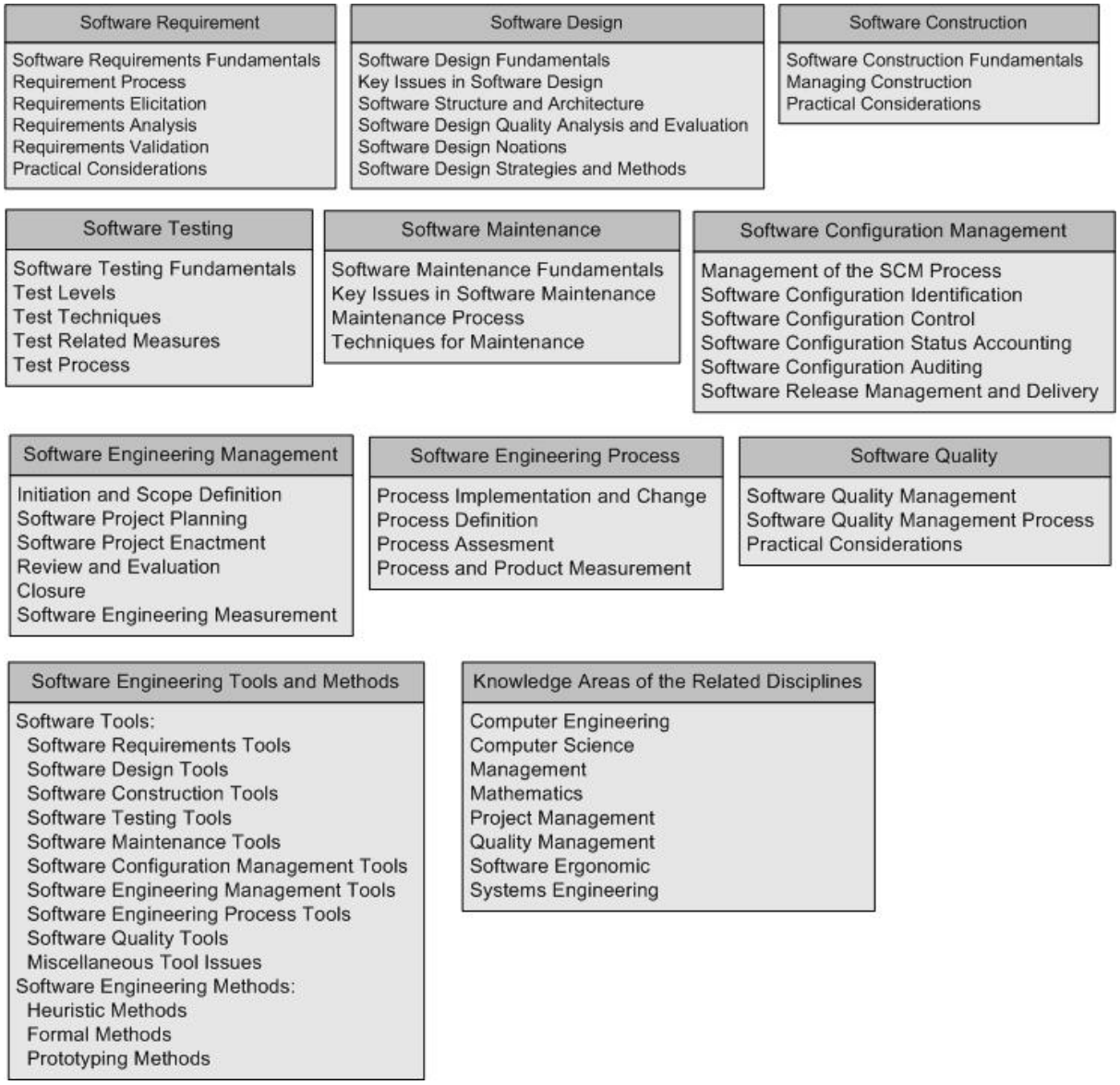
	Teori	Abstraksi	Desain
Algoritma dan Struktur Data	Teori Komputabilitas	Algoritma Paralel dan Terdistribusi	Program Aplikasi
	Teori Komputasi Kompleks		
	Komputasi Paralel	Algoritma Efisien dan Optimal	
	Teori Graf		
	Kriptografi		

Bahasa Pemrograman	Bahasa Formal dan Automata	BNF	Bahasa Pemrograman
	Turing Machines	Metode Parsing, Compiling, Interpretation	
	Formal Semantics		Translator, Kompiler, Interpreter
Arsitektur	Aljabar Boolean	Arsitektur Nueman	Produk Hardware (PC, Superkomputer, Mesin Von Neumann)
	Teori Coding	Hardware Reliability	
	Teori Switching	Finite State Machine	Sistem CAD dan Simulasi Logika
	Teori Finite State Machine	Model Sirkuit, Data Path, Struktur Kontrol	
Sistem Operasi dan Jaringan	Teori Concurrency	Manajemen Memori, Job Scheduling	Produk OS (UNIX, Windows, Mach, dsb)
	Teori Scheduling	Model Komputer Terdistribusi	File dan File Sistem
	Teori Manajemen Memori	Networking (Protokol, Naming, dsb)	Pustaka untuk Utilities (Editor, Formatter, Linker, dsb)
Software Engineering	Teori Reliability	Metode Spesifikasi	Bahasa Spesifikasi
	Program Verification and Proof	Metode Otomatisasi Pengembangan Program	Metodologi Pengembangan Software
	Temporal Logic	Tool Pengembangan Software	Tool untuk Pengembangan Software
Database dan Sistim Retrieval Informasi	Relational Aljabar dan Kalkulus	Data Model	Teknik Pendesainan Database (Relational, Hierarchical, Network, dsb)
	Teori Dependency		
	Teori Concurrency	Skima Database	Teknik Pendesainan Database Sistem (Ingres, Dbase, Oracle, dsb)
	Performance Analysis		
	Sorting dan Searching	Representasi File untu Retrieval	Hypertext System
	Statistical Inference		
Artificial Intelligence dan Robotik	Teori Logika	Knowledge Representation	Logic Programming (Prolog)
	Semantik dan Sintatik Model untuk Natural Language	Metode Pencarian Heuristic	Neural Network
	Conceptual Dependency	Model Reasoning dan Learning	Sistem Pakar
	Kinematics and Dynamics of Robot Motion	Model Memori Manusia, Autonomous Learning	Teknik Pendesaian Software untuk Logic Programming
Grafik	Teori Grafik dan Warna	Algoritma Komputer Grafik	Pustaka untuk Grafik
	Geometri Dimensi Dua atau Lebih	Model untuk Virtual Reality	Grafik Standar
	Teori Chaos	Metode Komputer Grafik	Image Enhacement System
Human Computer Interaction	Risk Analysis	Pattern Recognition	Flight Simulation
	Cognitive Psychology	Sistem CAD	Usability Engineering
Ilmu Komputasi	Number Theory	Discrete Approximations, Fast Fourier Transform and Poisson Solvers	Pustaka dan Paket untuk Tool Penelitian (Chem, Macsyma, Mathematica, Maple, Reduce, dsb)
	Binary Representation	Backward Error Propagation	
	Teori Quantum	Finite Element Models,	
Organizational Informatics	Organizational Science	Model dan Simlasi berhubungan dengan organizational informatics	Management Information Systems
	Decision Sciences		Decision Support Systems
	Organizational Dynamics		
Bioinformatik	Teori Komputasi	Model Komputasi DNA Kimia	Organic Memory Devices
	Ilmu Biologi	Protipe Retina dari Silikon	Proyek Database Genom Manusia

Matriks Denning versi 1999

Nah terus pertanyaan kembali muncul, jadi sebenarnya apa yang menjadi ruang lingkup ilmu *Software Engineering* itu apa?
Pertanyaan ini merupakan pertanyaan banyak orang, semakin banyak peneliti dan praktisi menulis maka semakin bervariasi pemahaman yang muncul, semakin banyak buku yang terbit semakin membingungkan pelajar dan mahasiswa dalam memahami secara komprehensif apa itu Rekayasa Perangkat Lunak.

Kegelisahan ini dijawab tuntas oleh IEEE Computer Society (http://computer.org) dengan membentuk tim di tahun 1998 dimana tim tersebut mulai menyusun pemahaman standard (*body of knowledge*) tentang bidang ilmu *Software Engineering*, yang kemudian terkenal dengan sebutan SWEBOK (*Software Engineering Body of Knowledge*). Sudah ada dua versi SWEBOK ini, yaitu yang diterbitkan tahun 1999 dan terakhir tahun 2004.



SWEBOK versi 2004

Tiada gading yang tak retak kata orang bijak, project IEEE Computer Society tentang SWEBOK ini sebenarnya juga banyak dikritik oleh pakar yang lain. Paling tidak dua tokoh besar dunia *Software Engineering* yaitu *Cem Kaner* and *Grady Booch* tidak terlalu setuju dengan materi yang ada di dalam SWEBOK, bahkan menyebutnya sebagai sebuah guide yang misguided ;) Terlepas dari hal itu, boleh dikatakan SWEBOK cukup bisa diterima banyak pihak.

Selain SWEBOK, sebenarnya ada project lain yang mirip dalam usaha menyusun pemahaman standard dalam bidang *Software Engineering*, yaitu CCSE (*Computing Curriculum Software Engineering*). Project ini juga disponsori oleh IEEE Computer Society dan ACM , hanya orientasinya sedikit berbeda, yaitu untuk membentuk kurikulum standard berhubungan dengan bidang ilmu *Software Engineering*. Hal ini berbeda dengan orientasi SWEBOK yang lebih umum melingkupi dunia akademisi dan praktisi.

Referensi

[1] Guide to the Software Engineering Body of Knowledge 2004 Version (SWEBOK), *A Project of the IEEE Computer Society Professional Practices Committee*, http://www.swebok.org, 2004.
[2] IEEE Standard Glossary of Software Engineering Technology, IEEE Std 610.12-1990, *Institute of Electrical and Electronics Engineers*, New York, 1990.
[3] Hans Van Vliet, “Software Engineering - Principles and Practice”, *John Wiley & Sons*, 2000.
[4] Peter J Denning, “Computer Science: the Discipline”, In *Encyclopedia of Computer Science* (A. Ralston and D. Hemmendinger, Eds), 1999.
[5] James F. Peters and Witold Pedrycz, “Software Engineering: An Engineering Approach”, *John Wiley & Sons*, 2000.
[6] Roger S. Pressman, “Software Engineering: A Practitioner’s Approach Fifth Edition”, *McGraw-Hill*, 2004.
[7] Ian Sommerville, “Software Engineering 7th Edition”, *Addison-Wesley*, 2004.

Biografi Penulis

Romi Satria Wahono. Menamatkan SMU di SMU Taruna Nusantara, Magelang pada tahun 1993. Menyelesaikan program S1 dan S2 di *Department of Computer Sciences, Saitama University*, Jepang tahun 1999 dan 2001. Saat ini program program S3 pada jurusan yang sama. Peneliti di Lembaga Ilmu Pengetahuan Indonesia (LIPI). Kompetensi inti pada bidang *Software Engineering*, *eLearning System*, dan *Knowledge Management*. Aktif sebagai penulis, dimana ratusan tulisan berupa scientific paper, artikel, dan tutorial telah diterbitkan dalam berbagai proceedings conference, jurnal ilmiah, majalah, koran dan portal, bertaraf nasional maupun internasional. Mendapatkan penghargaan dari PBB pada pertemuan puncak WSIS (*World Summit on Information Society*) tahun 2003 di jenewa, sebagai pendiri dari IlmuKomputer.Com. CEO dari PT Brainmatics Cipta Informatika, perusahaan IT yang bergerak di bidang training dan consulting.

Informasi lebih lanjut tentang penulis bisa didapat melalui:
Email: *romi@romisatriawahono.net*
URL: *http://romisatriawahono.net*
YM: *romi_sw*