ISBN: 979-756-061-6

OVERVIEW METODOLOGI REKAYASA PERANGKAT LUNAK BERORIENTASI AGEN

Azhari dan Sri Hartati
Fakultas MIPA, Universitas Gadjah Mada
Gedung SIC lt2/3, Sekip Utara, FMIPA UGM, Yogyakarta
E-mail: arisn@ugm.ac.id

Abstraks

Sebuah perangkat lunak aplikasi pada dasarnya adalah sebuah program komputer yang merupakan solusi terhadap sebuah problem dunia nyata dan dibangun dengan sebuah metodologi. Metodologi pengembangan rekayasa perangkat lunak yang telah ada, dirasakan masih kurang memadai, kurang efektif terutama untuk untuk mendukung pembangunan sistem aplikasi-aplikasi berbasis kecerdasan buatan. Kompleksistas dari perangkat lunak aplikasi akan semakin meningkat karena tersusun, dan terkait dengan banyak hal seperti kemampuan setiap bagian atau komponen dari perangkat lunak, kemampuan interaksi baik secara internal sistem ataupun dengan lingkungannya.

Dengan semakin pesatnya perkembangan teknologi berbasis agen, para pengembang perangkat lunak, telah mengusulkan pendekatan baru dalam upaya melakukan pembangunan sebuah perangkat lunak aplikasi, yaitu metodologi berbasis agen. Pada makalah ini, telah dilakukan pengkajian dan analisis secara sistematis dari paradigma metodologi pengembangan perangkat lunak berbasis sistem agen, seperti MaSE, Prometheus, PASSI. Secara menyeluruh metodologi tersebut tidak hanya menjanjikan banyak kemudahan, kecepatan dalam membangun sebuah aplikasi cerdas. Namun juga sangat memudahkan para pengembang sistem dalam memetakan, menspesifikasikan kompleksitas problem dunia nyata untuk menjadi konstruksi nyata sebuah sistem perangkat lunak aplikasi yang berkualitas.

Kata kunci: Agent Software, Agent Oriented Software Engineering, Metodology, Software Engineering.

1. Pendahuluan

Sebuah metodologi rekayasa perangkat lunak pada dasarnya harus menyediakan metode, pedoman, deskripsi, dan *tool* untuk setiap tahapan dalam siklus hidup dari sistem, sebagai jaminan untuk pembuatan dan perawatan dari produk rekayasa yang cocok dengan kegunaannya.

Setiap motodologi pembangunan perangkat yang diusulkan harus mendefinisikan seluruh urutan aktivitas dan tahapan yang harus dikerjakan, dan hubungan antara aktivitas-aktivitas tersebut. Setiap aktivitas dapat menghasilkan satu atau lebih keluaran seperti spesifikasi, dokumen, model, ataupun kode yang mendeskrisikan produk sistem. Pada umummya luaran tesebut dituliskan dengan notasi-notas tertentu seperti DFD, ERD, diagram class, psedo algoritma, dan sebagainya. Setiap notasi-notasi merupakan ciri atau pedoman khusus dari setiap pendekatan metodologi rekayasa perangkat lunak sistem.

Pada masa lalu, setiap pendekatan metodologi pembangunan perangkat lunak difokuskan kepada hasil yang lebih berorientasi kepada sistem komputasi secara terpusat, terstruktur, dan model pemrograman monolitik. Seperti pendekatan waterfall, prototipe atau experimental.

Pada masa sekarang, kecenderungan setiap produk perangkat lunak yang dihasilkan mengarah kepada sistem berorientasi komputasi terdistribusi, heterogen, *scalable*, dan model pemrograman terbuka dan terdistribusi, serta perangkat lunak berbasis agen. Seperti metodologi berbasis obyek, *thread*, ataupun agen

Padgham (9,10), rekayasa perangkat lunak berorientasi agen (*Agent Oriented Software Enginering*, AOSE), merupakan paradigma baru bagi para peneliti bidang rekayasa perangkat lunak, untuk melakukan, menghasilkan analisis dan sintesis setiap produk perangkat lunak sistem.

Metodologi AOSE merupakan suatu pendekatan dalam upaya mengembangkan dan membangun perangkat lunak dengan menggunakan abstraksi berorientasi-agen secara alamiah dengan memodelkan sebuah sistem kompleks seperti dunia nyata. Suatu sistem kompleks dapat dipandang terdiri dari subsistem-susbsistem atau agen, interaksi agen-agen, atau berprilaku seperti agen-agen didalam suatu organisasi (7, 11).

Seperti yang dituliskan oleh Caire, dkk (2), dan Shoham (11), metodologi AOSE menjanjikan kemampuan untuk membangun sebuah sistem secara sangat fleksibel. Dimana kompleksitas dan prilaku persoalan dibangun degan pengkombinasian sangat canggih, modular dari komponen-komponen *intelligent*.

Sifat-sifat cerdas dari sebuah sistem (perangkat lunak agen, atau cukup '*agent*' saja) dapat berupa (12,15) memiliki kemampuan belajar, merencanakan, berkomunikasi dan bernegosiasi Berikut contoh perangkat lunak berbasis agen (13):

- Animasi paperclip pada Microsoft Office.
- Computer virus (agen perusak).

ISBN: 979-756-061-6

- Pemain-pemain buatan atau aktor di dalam komputer game.
- Agen perdagangan dan negosiasi (seperti agen pelelangan Ebay.com)
- Laba-laba Web (pengumpul data, mesin pencari, seperti Google)

Pada makalah ini, dikaji dan dibahas empat metodologi pendekatan pengembangan perangkat lunak berbasis agen. Tujuan utama dari penelitian ini, mengkaji dan menganalisis karakteristik, ciri utama pada setiap tahapan analisis, desain, pembangunan, serta teknik, *tool*, notasi, keuntungan dari setiap metodologi.

2. Sistem Agen dan Multiagen

2.1 Definisi

Secara prinsip sebuah agen, atau disebut juga sebuah perangkat lunak agen, atau juga sebuah agen cerdas adalah sebuah perangkat lunak *outonomous* yang hidup, aktif, dan mampu beradaptasi secara mandiri, proaktif terhadap setiap kondisi lingkungan yang diciptakan untuknya (6,15).

Para pekerja dan peneliti dalam bidang teknologi agen telah mengusulkan berbagai definisi mengenai sebuah sistem agen. Pengertian dan definisi agen semakin berkembang pesat sesuai dengan kemampuan yang diciptakan untuk agen atau perkembangan kemapuan yang dihasilkan oleh agen tersebut sendiri. Berikut beberapa definisi agen dan sistem multiagen (3, 6, 12, 14-17):

"An agent is a computer system that situated in some enviroment, and that is capable of autonomous action in this enviroment in order to meet its design objective"

"Intelligent agents are software entities that carry out some set on behalf of user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of user's goals or desires"

"Software agents are programs that engage in dialogs and negotiate and coordinate transfer of information"

"An Agent is a software, a hardware or (more usually) sofware-based computer system that enjoys the following properties:

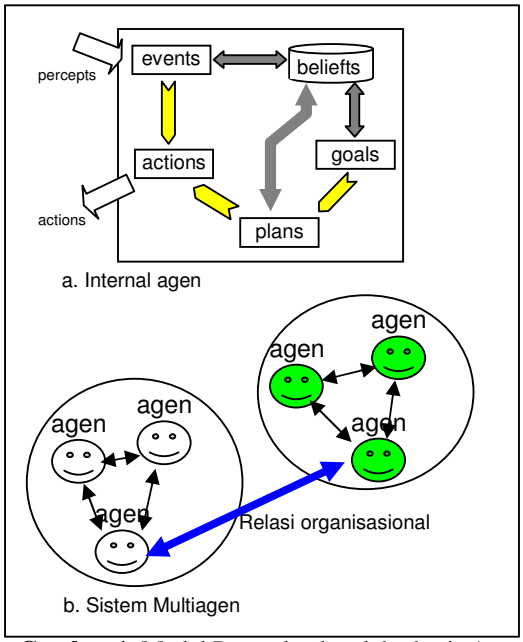
- *autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
- *social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*
- *reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
- *pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavour by taking the initiative."*

Pengertian agen dapat juga dibedakan menurut istilah notasi lemah dan kuat (15). Dalam istilah notasi lemah, agen-agen adalah memiliki kemampuan:

- *autonomy*, kemampuan beroperasi tanpa interaksi langsung manusia
- *social ability*, kemampuan untuk berinterkasi terhadap agen-agen lainya
- *reactivity*, kemampuan menangkap persepsi dan beraksi terhadap lingkungan
- *proactive*, kemampuan untuk melakukan inisiatif

Sedangkan istilah notasi lemah, agen-agen adalah memiliki:

- *mobility*, kemampuan berpindah didalam sebuah jaringan elekronik
- *veracity*, kejujuran
- *benevolence*, kemampuan bekerja apa adanya tanpa ada kompliks tujuan
- *rational*, kemampuan bertidak secara optimal untuk mencapai tujuannya



Gambar 1. Model Perangkat lunak berbasis Agen.

Karakteristik Agen

Pada gambar 1, diperlihat skema sifat-sifat, karakteristik model sebuah agen dan sistem multiagen secara organisasi, gambar dimodifikasi dari (10, 7).

ISBN: 979-756-061-6

- Sebuah agen didasarkan pada sekumpulan pengetahuan terpercaya (*beliefs*, *siqnificant knowledge*). Dengan pengetahuan, agen mempunyai kemampuan beragumentasi (*actions*). Pengetahuan agen direpresentasikan secara lebih umum dari pada basis-aturan (*rules*), dalam bentuk atribut-atribut, status, dan kepemilikan.
- Sebuah agen memiliki sifat-sifat autonomous dalam pengertian bahwa agen bekerja atau melayani diri sendiri dan memiliki kepentingan tanpa interpensi langsung secara ekternal. Menurut hasil penalarannya agen dapat memutuskan sendiri dan tindakan kapan akan dilaksanakan (*plans, and actions*) . Jika agen menyimpulkan sesuatu, dimungkinkan untuk memberitahukan kepada agen-agen lain. Prilaku autonomous ini diturunkan dari mesin aturan yang secara proaktif terus mengevalasi aturan.
- Sebuah agen memiliki tujuan jelas (*well-define goal*) yang disajikan oleh pengetahuan dan prilaku yang dimilikinya (yaitu kemampuan memberikan alasan berdasarkan pengetahuan). Sebuah agen mencoba memberikan alasan sederhana (*actions*), memberikan pengetahuannya, untuk memenuhi tujuannya.

Wooldrige (14) dan DeLoach, et.al (5), sistem Multiagen (*Multiagen System*, MAS) dapat dipandang sebagai sebuah grup dari entitas-entitas cerdas (atau agen-agen). Sistem multi agen mampu melakukan interaksi antara satu agen dengan agen lainnya untuk mencapai tujuan-tujuannya secara bersama-sama. Umumnya sebuah agen memiliki sekumpulan tujuan-tujuan, kemampuan-kemampuan khusus untuk membentuk aktivitas-aktivitas, dan pengetahuan (atau kepercayaan) mengenai lingkungan sekitarnya. Untuk mencapai tujuan-tujuannya, sebuah agen memerlukan pendapat dan pandangan mengenai lingkungannya (termasuk prilaku dari agen lainnya), untuk membangkitkan rencana-rencana dan melaksanakan rencana tersebut.

2.2 Prinsip dasar Metodologi Berorientasi Agen

Adapun tujuan utama dari AOSE adalah untuk menciptakan dan menyediakan metodologi dan perangkat lunak bantu yang mampu secara murah untuk mengembangkan dan merawat perangkat lunak berbasis agen. Perangkat lunak agen cerdas (*intelligent agent*) yang dihasilkan sebaiknya harus fleksibel, mudah digunakan, *scalable* dan berkualitas tinggi (1, 7, 8, 11, 13,16). Istilah-istilah (13), seperti *Software Engineering with Agents*, *Agent-Based Software Enginnering*, *Multi-agent System Enginnering* (MaSE) pada prinsipnya mempunyai tujuan dan maksud yang sama secara semantik.

a. Tahap Analisis Agen

Pada tahap awal analisis berbasis agen, para pengembang melakukan analisis kebutuhan masalah yang secara prinsip difokuskan pada persoalan-persoalan logis. Misalnya apakah yang harus dilakukan oleh perangkat lunak, sifat-sifat dan prilaku-prilaku perangkat lunak. Termasuk menentukan sasaran utama dari sistem, peranan dan aturan-aturan untuk penyelesaian masalah, ketersediaan sumberdaya-sumberdaya, kebutuhan interaksi eksternal, variasi-variasi prilaku yang dapat dicapai oleh sistem.

Bergenti dan Paola (1), tujuan utama tahap analisis berbasis agen adalah untuk mengetahui aktor utama yang akan beriteraksi dengan sistem, bagaimana cara beriteraksinya, apakah yang harus dikerjakan oleh sistem. Kemudian merumuskan bagaimana peranan, tanggung-jawab, kemampuan dari agen-agen, serta cara berinteraksi dengan agen lainnya.

Aktivitas tahap analisis agen:

1. Agen-agen ekternal. Agen ekternal diidentifikasi melalui tujuan utamanya yang penting.
2. Diagram interaksi/percakapan. Diagram ini dibuat membentuk aliran komunikasi dengan memperhatikan dunia nyata sebagai gabungan segala sesuatu kecerdasan dan memodelkan interaksi diantaranya untuk membentuk percakapan. Pada sistem berorientasikan agen diagram interaksi menyediakan sebuah cara alamiah untuk menggambarkan pengedalian dan pewaktuan informasi.
3. Sistem dan pesan eksternal. Pesan-pesan dimana sistem harus mampu memberikan tanggapan harus diidentifikasi.
4. Diagram domain pesan. Sebuah diagram domain pesan harus dibuat sebagai cakupan dan validasi komunikasi.

ISBN: 979-756-061-6

b. Tahap Perancangan Agen

Pada tahap desain, merupakan proses untuk indenfikasi dari analisis menjadi sebuah model phisik yang independen dan layak, agar dapat diimplementasikan. Mentrasformasikan sasaran utama dan aturan-aturan ke dalam jenis-jenis agen secara nyata. Seperti pembentukan aturan-aturan agen, perumusan jumlah dan jenis agen. Pembentukan protokol interaksi khusus, dan fungsi dari setiap jenis agen untuk melakukan prilaku-prilaku lanjutan dengan pendefinisian ontologi untuk setiap agen.

Juga dari tulisannya Bergenti dan Paola (1), tujuan dari percanangan berbasis agen adalah merekayasa interaksi komponen-komponen utama di dalam sistem, menentukan kemampuan, dan tanggung jawab dari setiap komponen di dalam sistem. Membentuk bagaimana cara komponen-komponen berinteraksi agar dapat diimplementasikan, dan medeskripsi-kan struktur arsitektur secara menyeruh dari sistem

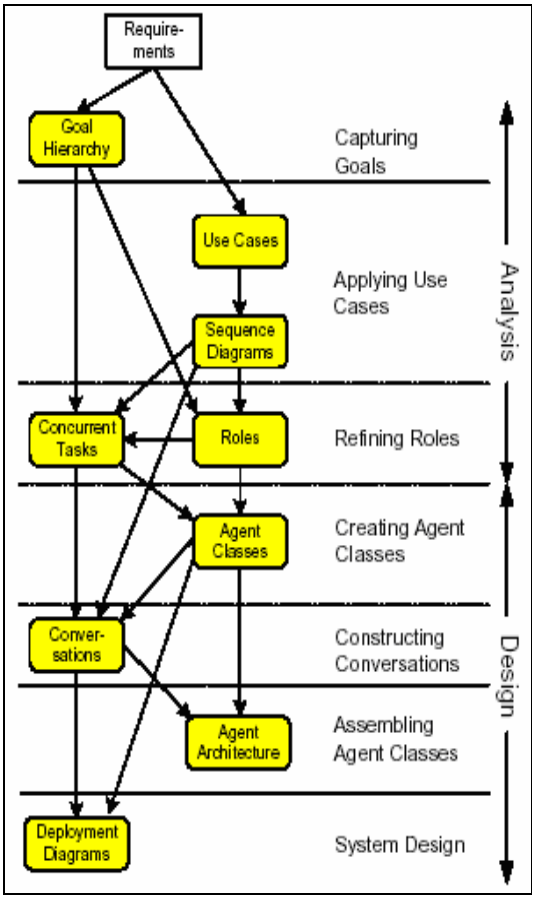
Aktivitas tahap desain agen:

1. Diagram agen internal. Tujuan agen dan servis yang dihasilkan digambarkan dalam sebuah diagram agen internal.
2. Diagram relasi class. Diagram agen diterjemahkan ke dalam diagram relasi class dengan memperhatikan keterkaitan antar agen.
3. Diagram interaksi pisik. Diagram interaksi pisik dibuat untuk setiap proses utama yang didasarkan pada diagram interaksi logis.

Banyak pendekatan telah diusulkan oleh para ahli untuk membangun perangkat lunak berbasis agen. Secara umum, meskipun cara penanganan berbeda namun metodologi-metodologi tersebut saling melengkapi. Berturut-turut metodologi AOSE diantaranya adalah Metodologi Gaia (17) merancang secara organisasional, TROPOS (18) mendesain dan membangun kebutuhan dengan konsep goal, Metodologi Prometheus (9,10) memfokuskan pembanguan sistem secara arsitektur internal agen; Metodologi MaSE (5) pembangunan merupakan konsep interaksi multiagen, MESSAGE (2) kombinasi dengan pendekatan terdahulu; Metolologi PASSI (4) mengintegrasi-kan model OO dan AI.

3. Metodologi MaSE

Metodologi rekayasa sistem multiagen (*Multiagent System Engineering*, MaSE), dimulai dengan langkah spesifikasi sistem, dan selanjutnya memproduksi sekumpulan dokumen rancangan formal dalam bentuk-bentuk grafis. Perhatian utama dari metodologi MaSE adalah untuk membantu para pengembang dalam merancang dan membangun perangkat lunak berbasis sistem multiagen, melalui siklus hidup perangkat lunak dari proses spesifikasi hingga implementasi sistem multiagen. Sebuah sistem yang dirancang dengan metodologi MaSE dapat diimplementasikan dalam berbagai cara dari rancangan yang sama. Setiap rancangan obyek dapat dialcak kembali dari/ke aktivitas sebelumnya dan



Gambar 2. Tahapan Metodologi MaSE

dari/ke aktivitas berikutnya pada setiap tahapan.

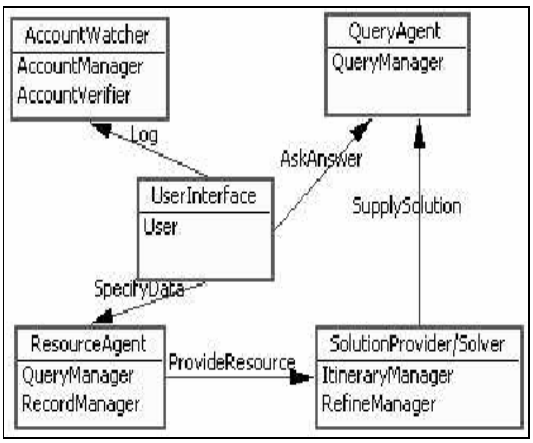
Secara singkat tahapan dan model dari metodologi MaSE diperlihatkan pada gambar 2. Tahapan proses-proses dari metodologi MaSE terdiri dari tujuh langkah aktivitas, yang dikelompokkan dalam dua phase dan memproduksi sepuluh model dokumen formal. Phase Analisis terdiri dari langkah-langkah seperti 1). *Capturing Goal*, 2) *Applying Use Cases*, 3) *Refining Roles*. Phase Desain terdiri dari langkah-langkah 4) *Create Agent Classes*, 5) *Contruction Conversations*, 6) *Assembling Agent Classes*, dan 7) *System Design*.

a. Phase Analisis MaSE

1. *Capturing Goals*, langkah yang membantu para analis untuk mengidentifikasi tujuan-tujuan, struktur dan menyajikannya sebagai hirarki/berjenjang tujuan sistem.
2. *Applying Use Cases*, meliputi penyerapan scenario utama dari konteks awal sistem. *Use Cases* juga digunakan untuk membangun diagram terurut (mirip seperti pada UML).

ISBN: 979-756-061-6

3. *Refining Roles*, merupakan langkah terakhir dari tahapan analisis, yaitu untuk pembentukan model peran dan model tugas konkuren. Model peran menggambarkan peranan dari agen dalam sistem. Di samping itu, peran-peran yang



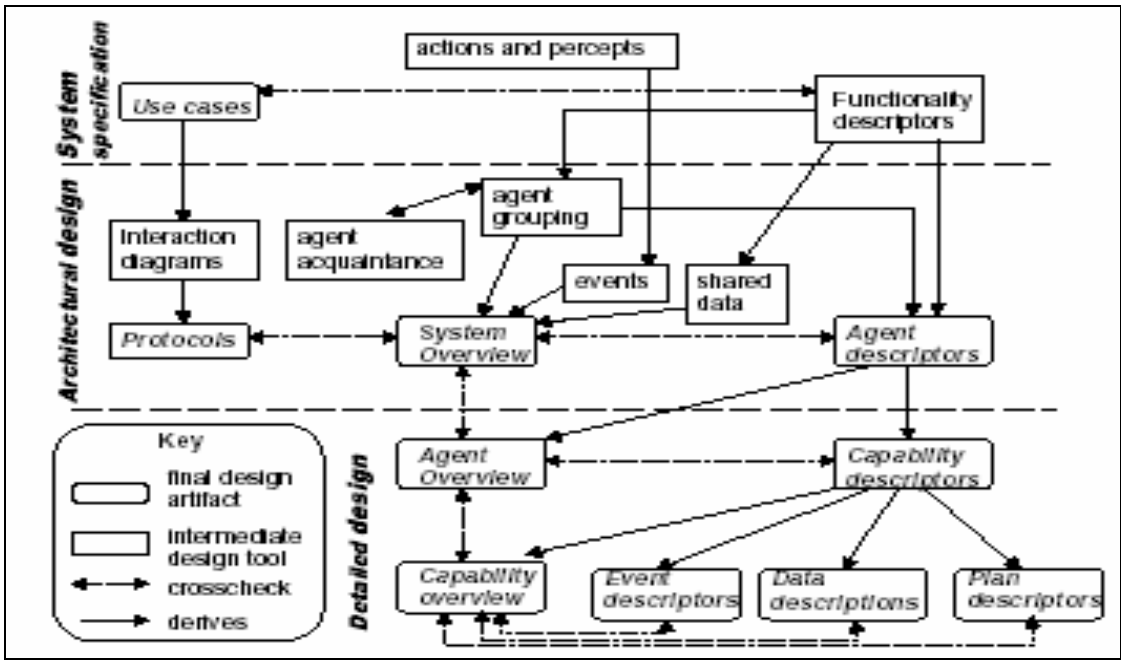
Gambar 3. Contoh diagram class agen.

dibentuk juga harus menunjukan bagaimana peranan yang dimainkannya. Hubungan (*link*) antar agen-agen diperlihatkan melalui notasi percakap-an, misalnya seperti yang ditunjukkan pada gambar 3.

b. *Construction Conversation*, merupakan diagram detail mekanisme konversasi dari agen-agen yang digambarkan dalam diagram komunikasi classes, yang berbentuk mesin status berhingga (*finite state machine*)
c. *Assembling Agent Classes*, pada langkah ini didefinisikan aristektur agen dan komponen-komponen penyusunnya.
d. *System Design*, meliputi pembuatan digram *deployment*, yang menspesifikan penempatan agen-agen dalam sebuah sistem.

4. Metolodogi Prometheus

Menurut Padgham dan Michael (9,10), Metodologi prometheus merupakan metode yang mudah dan ditujukan untuk para bukan ahli (*non expert*), dan telah sukses di digunakan oleh para mahasiswa tingkat sarjana. Metodologi prometheus terdiri dari phase-phase (gambar 4): *System Specification*, *Architecture Design*, dan *Detailed System*.



Gambar 4. Tahapan Motodologi Prometheus.

tujuan-tujuan sistem akan dicapai dan jalur komunikasi diantara peran-peran tersebut. Model tugas yang konkuren digambarkan dalam bentuk diagram status mesin.

b. Phase Desain MaSE

a. *Creating Agent Classes*, hasil dari langkah adalah sebuah diagram class agen yang mengambarkan secara menyeluruh sistem multiagen. Diagram class agen memper-lihatkan agen-agen dan

a. Phase Spesifikasi Sistem

Aktivitas utama pada phase ini terdiri dari:

a. Penentuan lingkungan sistem, dinyatakan dalam istilah persepsi (informasi yang masuk dari lingkungan) dan aksi (dalam arti agen bereaksi terhadap lingkungannya). Termasuk pada langkah ini juga pendefinisian data eksternal.
b. Pendefinisian fungsionalitas sistem, dilakukan dengan penetapan tujuan, dan fungsi yang akan dicapai terhadap tujuan tersebut dengan

ISBN: 979-756-061-6

menggunakan scenario *use case* dan dilakukan secara berulang. Sebuah scenario pada prinsipnya meliputi urutan langkah-langkah penentuan presepsi yang masuk, pengiriman pesan, aktivitas-aktivitas dan respon.

b. Phase Desain Arsitektur

Pada phase ini aktivitas utama terdiri dari:

a. Pendefinisian jenis-jenis agen, diturunkan dari kelompok fungsionalitas. Pemilihan jenis agen dari kelompok fungsionalitas perlu memperhatikan kopling dan kohesi yang diidentifikasi dari diagram kopling data dan diagram *acquinted* agen. Setiap jenis agen digambarkan dengan sebuah deskripsi agen, yang memperlihatkan siklus hidup agen, fungsionalitasnya, data yang digunakan dan dihasilkan, tujuannya, dan kejadian-kejadian untuk penangapan eksternal, tindakan yang dilakukan terhadap agen lainnya.
b. Perancangan aristektur sistem secara menyeluruh, diperloleh dari diagram overview sistem, merupakan ciri dan bagian terpenting pada metolologi prometheus. Diagram overview sistem menyediakan bagi para perancang dan pengembang gambaran umum bagaimana sistem akan berfungsi. Memperlihatkan jenis-jenis agen, komu-nikasi antara agen, dan data. Diagram ini juga memperlihatkan, batasan dan ling-kungan sistem (seperti tindakan, persepsi, dan data eksternal).
c. Pendefinisian interaksi antara agen-agen, perancangan protokol interaksi yang merupakan bentuk gambaran prilaku dinamis dari sistem melalui pendefinisian urutan pesan berurut antara agen. Protokol interaksi dibuat dari diagram interaksi.

c. Phase Rancangan Detail

Secara internal bagaimana setiap agen memenuhi tugas-tugasnya didalam sistem secara menyeluruh dirancang secara detail pada phase ini. Aktivitas utama dari phase ini terdiri dari:

a. Pendefinisian kemampuan, internal struktur dari kemampuan setiap agen, dilakukan melalui deskripsi kemampuan yang memuat informasi seperti kejadian yang dibangkitkan dan diterima. Deskripsi kamampuan juga detail interaksi dengan kemampuan lainnya, dan referensi untuk membaca dan menuliskan data. Kemudian deskripsi kemampuan agen dipetakan sebagai rencana, kejadian dan data.
b. *Internal event*, deskripsi detail pemacu tujual, seperti keinginan, kepercayaan, intensi (*Belief, Desire, Intention*, BDI).
c. *Plan*, kemampuan-kemampuan dari agen dideskripsikan dalam bentuk agenda rencana-rencana agen, deskripsi rencana individual, termasuk *user define plan*.
d. Detail data struktur. Merancang detial struktur data untuk akses phisik sistem.

5. Metolodogi PASSI

Cossentino dan Colin (4) mengusulkan, sebuah pendekatan proses pembangunan sistem agen untuk robotik yang mereka sebut metodologi PASSI (*Process for Agent Societies Specification and Implemen-tation*). Metodologi PASSI mengambarkan proses pembangunan sistem agen terdiri dari lima prinsip model yaitu: a) *System Requirements*, b) *Agent Society*, c) *Agent Implementation*, d) *Code Model* and e) *Deployment Model*, yang dilakukan dalam beberapa tahapan. Pada gambar 5, diperlihatkan tahapan dari metodologi PASSI.

a. Model Kebutuhan Sistem

Merupakan sebuah model kebutuhan sistem dalam bentuk agen dan kegunaannya. Pada pemodelan kebutuhan ini terdiri dari empat tahap:

Deskripsi Domain. Pendeskripsian fungsional sistem dengan menggunakan diagram-diagram use-case.

Identifikasi Agen. Pendefinisian atribut-atribut dari agen, yang disajikan dalam bentuk notasi-notasi stereotype UML.

Identifikasi peran/aturan. Merumuskan tanggungjawab setiap agen melalui skenario aturan/peran khusus dalam bentuk diagram-diagram terurut.

Spesifikasi tugas. Menspesifikasikan kemampuan dari setiap agen dalm bentuk diagram-diagram aktivitas.

b. Model Perkumpulan Agen

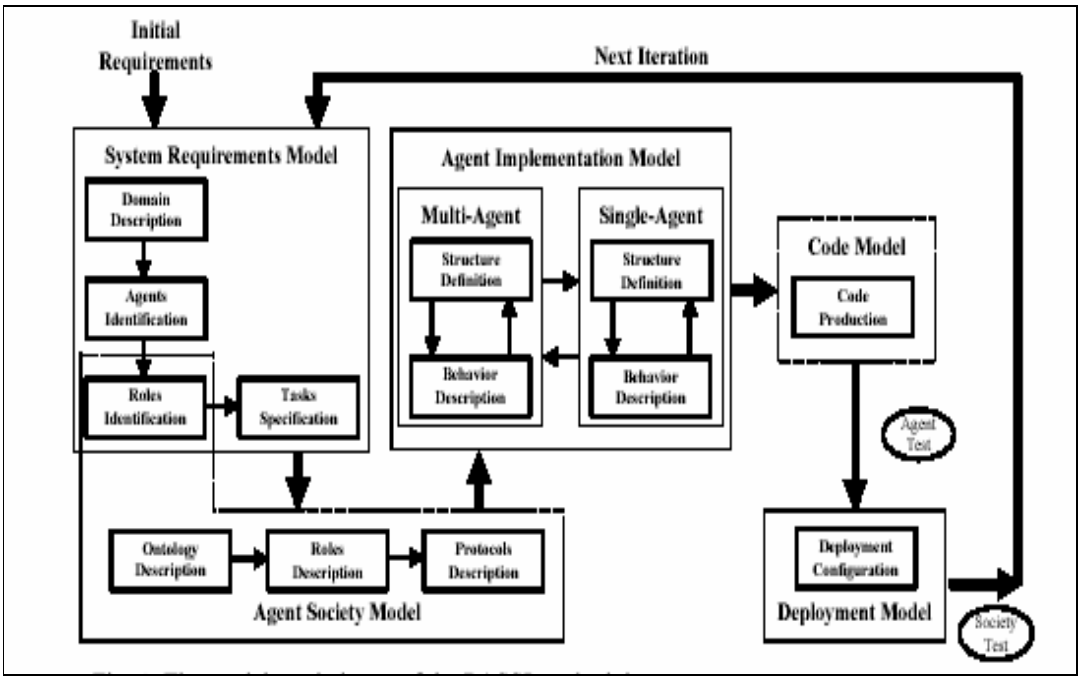
Merupakan model interaksi-interaksi sosial dan ketergantungan antara agen-agen untuk menanggapi solusi eksternal. Terdapat tiga aktivitas utama pemodelan pada tahapan ini.

a. Indetifikasi Peran Agen.
b. Deskripsi Ontologi. Pemanfaatan diagram class dan batasan untuk mendeskripsikan pengetahuan asal dari setiap agen dan cara interaksi pragmatisnya.
c. Deskripsi Peran. Diagram class digunakan untuk memperlihatkan peran yang dimainkan oleh agen-agen, termasuk tugas-tugas, kemampuan berkomunikasi, dan ketergan-tungan antar agen.
d. Deskripsi Protokol. Penggunaan diagram-diagram terurut untuk menentukan gramatika dari setiap protokol komunikasi pragmatis agen.

c. Model Implementasi Agen

Secara klasik adalah model arsitektur solusi dalam bentuk class dan metode seperti pada pendekatan berorientasi obyek. Hanya pada model ini memiliki level abstraksi sosial (*multiagent*) dan level agen tunggal.

 ISBN: 979-756-061-6



Gambar 5. Model dan tahapan dari Metodologi PASSI.

a. Definisi Struktur Agen. Diagram class menggambarkan struktur solusi dari class agen.
b. Deskripsi Prilaku Agen. Diagram aktivitas atau diagram status memperlihatkan prilaku agen secara individual.

d. Model Kode Program

Untuk menghasilkan sebuah model solusi pada level kode dilakukan aktivitas berikut:

a. Generalisasi kode dari model menggunakan salah satu program fungsionalitas PASS. Tool ini, memungkinkan membangkitkan kerangka kode, dan bagian-bagian yang dapat digunakan kembali (*reusable*) berdasarkan sebuah library yang disesuaikan dengan deskripsi rancangan.
b. Penyusunan manual lengkap dari kode program.

e. Model *Deployment*

Konfigurasi *Deployment*, meliputi deskripsi konfigurasi dari *deployment* sistem digambarkan sebagai alokasi dari agen terhadap unit prosesing dan batasan-batasan migrasi dan mobilitas.

6. Pengujian

Pengujian dilakukan dalam bentuk pengujian agen tunggal untuk memverifikasi prilaku setiap agen berdasarkan solusi kebutuhan awal oleh agen-agen secara khusus. Pada saat bersamaan dilakukan pengujian menyeluruh untuk memvalidasikan seluruh hasil dari interkasi tersebut dan integrasinya terhadap agen-agen yang berbeda.

6. Kesimpulan

Paradigma rekayasa perangkat lunak berorientasi agen merupakan proses pemodelan sistem kompleks dan terdistribusi yang efektif, seperti pengunaan dekomposisi, kesesuaian abstraksi, dan terfokus pada struktur secara organisasional. Ketiga metodologi yang direview menunjukkan bahwa setiap metodologi memiliki karakteristik khusus dalam cara memandang persoalan dunia nyata untuk diimplementasikan menajdi sebuah sistem perangkat lunak berbasis agen.

Metodologi prometheus relatif lebih memudahkan para pengembang, dikarenakan agen-agen yang dibangun telihat lebih detail secara internal sistem. Sedangkan metodologi PASSI sangat mendukung pembangunan siste agenm berbasis *real time*, seperti aplikasi robotika. Namun secaca menyeluruh, semua metodologi akan menjadi semakin mudah diterapkan dengan menggunakan atau didukung sebuah tool pembangun sistem agen (Agent atau Multiagent Generator Tool).

Daftar Pustaka

[1] Bergenti, F. dan Paola T., *Agent-Oriented Software Engineering*, 2002
[2] Caire, G., at.al., Agent Oriented Analysis using MESSAGE/UML. 2nd International Wokshop On Agent Oriented Software Engineering, LNCS No. 2222, Springer-Verlag, pp 119-135, 2001.
[3] Coen, M., *The SodaBot Agent*, 1988 http://ai.mit.edu/people/sodabot/slideshow/ total/P001.html [akses terakhir, Maret 2004].

ISBN: 979-756-061-6

[4] Cossentino, M. dan Colin Potts, A CASE Tool Supported Methodology for the Design of Multi-agent System, *Proc. SERP'02*, Las Vegas (NV), USA, June 2002.

[5] DeLoach, et.al, Multiagent Systems Engineering, *The International Journal of Software Engineering and Knowledge Engineering*, Vol. 11 (3), pp. 231-258., 2001

[6] Franklin, S., dan Art G., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agent, in A Knowledge Level Software in *Engineering Methodology for Agent Oriented Programming,* Poulo Bresciani (editor), dkk., Montr´eal, Quebec, Canada., 2001

[7] Jennings, N.R., An Agent-Based Approach for Building Complex Software Systems, *Communication of the ACM*, 44(4), pp35-41, 2001.

[8] Jennings, N.R., On Agent-Based Software Engineering, *Artificial Intelligence* 117, pp227-296, 2000.

[9] Padgham, L. dan Michael W, Prometheus: A Methodology for Developing Intelligent Agents, *Third International Workshop on Agent-Oriented Software engineering*. 2002

[10] Padgham, L. dan Michael W., *Developing Intelligent Agent Systems: A Practical Guide*, John Willey & Sons Ltd, The Atrium, Chicheter, West Susex, England, 2004

[11] Shoham, Y., An Overview of Agent-Oriented Programming, in J. M. Bradshaw, editor, *Software Agents*, pages 271–290. AAAI Press /The MIT Press, 1997.

[12] The IBM agents, 1981, IBM's *Intelligent Agent Strategy*, White paper, 1981, http://www.activist.gpl.ibm.com:81/WhitePaper /ptc2.htm [akses terakhir, Maret 2004].

[13] Tveit. A., A Survey of Agent-Oriented Software Engineering, *First NTNU CSGSC*, 2001, www.elcomag.com/amund/ [akses terakhir, Maret 2004]

[14] Wooldrige, M., *An Introduction to Multiagent Systems*, John Willey & Sons Ltd, The Atrium, Chicheter, West Susex, England, 2002.

[15] Wooldrige, M. dan Jennings NR., Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review 10(2)*, 115-152, 1999.

[16] Wooldridge, M.J. dan N. R. Jennings, Software Engineering with Agents: Pitfalls and Pratfalls, IEEE Internet Computing, Vol.3(3), 1999.

[17] Wooldridge, M, Jennings N.R dan Kinny, D., The Gaia Methodology for Agent-Oriented Analysis and Design, *Journal of Autonomous Agents and Multi-agent Systems*, 3(3), 2000.

[18] Khanh Hoa Dam dan Michael Winikoff, Comparing Agent Oriented Methodologies, ACM 089791886, 1997.