p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

MODEL BERORIENTASI AGEN UNTUK ANALISIS DAMPAK PERUBAHAN PADA APLIKASI WEB DINAMIS

Khegi Walesa[1*], M. Sukrisno Mardiyanto[2], Wikan Danar Sunindyo[3]
[*123]Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung
Jl. Ganesha No.10, Lb. Siliwangi, Coblong, Kota Bandung
E-mail: khegiw@gmail.com

ABSTRAK

Pada siklus hidup pengembangan perangkat lunak, *Software Mintenance* menggunakan lebih dari 60% biaya dan waktu pengembangan. Hal ini berperan penting dalam menjaga keberlangsungan hidup perangkat lunak. Aplikasi Web Dinamis memiliki sifat heterogen. Menggunakan bermacams metodologi pengembangan, teknologi yang bervariasi, multi bahasa pemrograman, serta menggunakan komponen pihak ketiga yang selalu berkembang dan berubah. Perubahan yang terjadi sangat mempengaruhi bagaimana sistem tersebut dapat beradaptasi. Analisis Dampak Perubahan menawarkan suatu pendekatan untuk mendeteksi pengaruh yang diakibatkan oleh suatu perubahan. Terdapat berbagai teknik dan pendekatan yang telah ditawawrkan, namun belum ada standar yang nenetapkan bagaimana cara analisis dampak dilakukan, khususnya pada kasus aplikasi web.

ABSTRACT

In the life cycle of software development, software Mintenance use more than 60% the cost and development time. It plays an important role in maintaining the survival of the software. Dynamic Web applications have a heterogeneous nature. Using various development methodologies, technologies are varied, multi-language programming, as well as the use of third party components is always evolving and changing. The changes were affecting how the system can adapt. Analysis of Impact of Change offers an approach to detect the effect caused by a change. There are a variety of techniques and approaches that have ditawawrkan, but there is no standard nenetapkan how do impact analysis, particularly in the case of a web application.

Pengembangan perangkat lunak berorientasi agen menawarkan suatu pendekatan, sebagai upaya menghindari berakhirnya hidup suatu sistem secara lebih cepat. Terdapat beragam metodologi yang dapat digunakan dalam pengembangan perangkat lunak berorientasi agen, beserta kelebihan dan kekurangannya. Ketepatan menspesifikasikan karakteristik kebutuhan pengguna dapat dijadikan target dalam menciptakan suatu konstruksi pendekatan yang sesuai dalam mengelola perubahan, termasuk dalam menjamin keberlangsungan hidupnya.

Dalam penelitian ini, dilakukan kajian dan analisis terhadap sejumlah pendekatan berbasis agen, untuk mengembangkan model yang dibutuhkan. Permasalahan kasus sistem yang dibahas disini adalah kebutuhan analisis dampak perubahan yang sesuai diterapkan pada kasus aplikasi web dinamis. Pada akhir pembahasan kami mengusulkan suatu model sebagai temuan awal, khususnya yang dapat diterapkan saat perawatan perangkat lunak.

Keywords—*software maintenance*, agen oriented software engineering, change impact analysis.

I. PENDAHULUAN

Perawatan perangkat lunak (Software Mintenance) merupakan fase sulit pada siklus hidup pengembangan perangkat lunak. Banyak organisasi pengembangan perangkat lunak menghabiskan 60% biaya dan tenaga untuk melakukan proses perawatan. Dengan tumbuh berkembangnya kebutuhan presentase tersebut masih terus meningkat Pada saat perangkat lunak semakin berkembang dan menggunakan banyak komponen yang saling berinteraksi satu sama lain. Hal ini akan mempersulit perawatan, sebuah perubahan kecil dapat berakibat kegagalan pada sistem perangkat lunak.

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

Ditambah jika perangkat lunak yang dirawat adalah sistem warisan, pengembang sebelumnya sudah tidak ada dan tidak didukung oleh dokumentasi yang lengkap serta mudah dipahami pihak yang melakukan perawatan. Pada saat perangkat lunak berevolusi, pekerjaan perawatan pun menjadi kompleks dan mahal. Menurut standar IEEE 1219, Software maintenance adalah memodifikasi perangkat lunak untuk memperbaiki kesalahan, untuk meningkatkan kinerja atau agar produk dapat beradaptasi dengan perubahan lingkungan. Perawatan perangkat lunak sangat diperlukan untuk menjamin sistem dapat terus memenuhi kebutuhan pengguna.

Penting bagi pihak pengembang membuat keputusan yang efektif dalam mengelola perubahan dalam pekerjaan perawatan perangkat lunak. Salah satu tipe informasi yang membantu pengambilan keputusan perubahan adalah prediksi seberapa besar dampak yang diakibatkan oleh suatu perubahan pada perangkat lunak. Prediksi ini dapat dilakukan dengan melakukan analisis dampak perubahan (*Change Impact Analysis).* [1],[2], [3].

Telah banyak pendekatan analisis dampak perubahan dikembangkan oleh peneliti, namun masih terdapat beberapa kekurangan. Sebagai contoh teknik yang dikembangkan adalah teknik static analysis[4]–[7] dan teknik dynamic analysis [5], [8]–[11]. Implementasi dari teknik-teknik tersebut tergantung pada kasus-kasus tertentu. Belum adanya standar atau ketentuan yang menetapkan proses analisis dampak perubahan pada waktu perawatan perangkat lunak mengakibatkan pihak pengembang kesulitan dalam memilih teknik yang sesuai dengan kebutuhan perubahan pada saat pengerjaan perawatan.

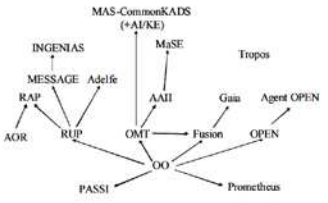
Paradigma agen dalam pengembangan perangkat lunak, merupakan suatu pendekatan dalam upaya pengembangan dan pembangunan perangkat lunak dengan menggunakan abstraksi berorientasi agen secara alamiah, untuk memodelkan sebuah sistem kompleks seperti dunia nyata. Suatu sistem kompleks dapat dipandang terdiri dari subsistem-subsistem atau agen, interaksi agen, atau berprilaku seperti agen pada suatu organisasi [12]. Terdapat penelitian yang mengusulkan paradigma agen untuk pengelolaan perubahan, agen dekonstruksi mewakili manusia dalam domain model dan abstraksi organisasi, kontribusi yang diusulkan terkait pada aktivitas analisis pada awal pengembangan perangkat lunak, ini memungkinkan pemahaman terhadap kebutuhan perubahan proses bisnis dan menganalisis kebutuhan integrasi komponen lama dengan komponen baru [13]. Namun pendekatan ini tidak mendukung pada aktivitas desain sistem secara terperinci dan proses pengembangan perangkat lunak selanjutnya.

Penelitian ini bertujuan mengembangkan sebuah pendekatan berorientasi agen untuk analisis dampak perubahan yang dapat mencakup seluruh aspek yang terkait pada perawatan perangkat lunak. Penelitian ini diharapkan dapat berkontribusi pada analisis dampak perubahan baru yang lebih sistematis dan penerapan metode berorientasi agen dalam pengelolaan perubahan yang kompleks.

II. LANDASAN TEORI

A. *Rekayasa Perangkat Lunak Berorientasi Agen (AOSE)*

Metodologi AOSE menjanjikan kemampuan untuk membangun sebuah sistem secara sangat fleksibel [14], [15], kompleksitas dan perilaku sistem dibangun dengan kombinasi canggih dari komponen-komponen cerdas. Sebuah agen atau disebut juga sistem agen merupakan sebuah perangkat lunak autonomous yang hidup, aktif dan mampu beradaptasi secara mandiri serta proaktif terhadap lingkungan yang diciptakan untuknya [12]. Agen adalah sesuatu yang dapat mengamati lingkungan melalui sensor dan melakukan tindakan berdasarkan kondisi lingkungan melalui kemampuan yang dimiliki (actuator) [16]. Sistem agen memiliki ciri-ciri seperti kemandirian, penalaran dan pembelajaran, kemampuan mobilitas dan menetap, delegasi, reaktivitas, proaktif dan berorientasi tujuan, serta mampu melakukan koordinasi dan komunikasi. Secara teknis [17], agen dibangun melalui konsep memiliki status mental kepercayaan/pengetahuan, keinginan dan intensi. (Belief-Desire-Intention, BDI).

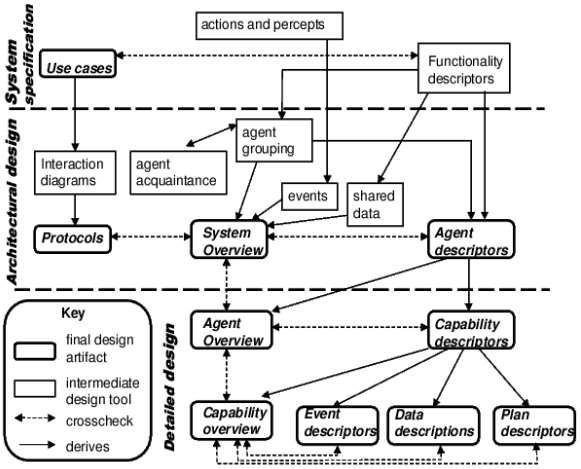


Gambar II-1 *Metodologi AOSE* [18]

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

B. Metodologi AOSE

Beberapa peneliti telah mengembangkan metodologi AOSE. salah satu metodologi dikembangkan berbasiskan ide dari *artificial Intelligence* (AI), yang lain merupakan pengembangan langsung dari metodologi *object oriented* (OO). Seperti yang digambarkan pada Gambar II.1, hampir semua metodologi berbasiskan metodologi OO. Hanya metodologi *Tropos* yang merupakan basis *AI* dan juga metodologi *Prometheus* adalah metodologi yang paling lengkap [18].



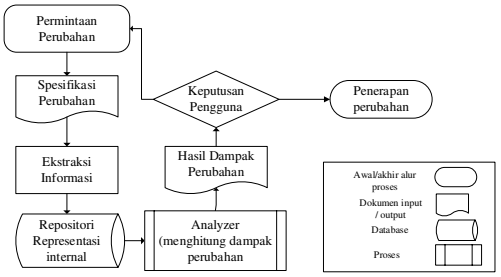
Gambar II-2 Tahapan Metodologi Prometheus [19]

struktur metodologi *prometheus* terdiri dari 3 tahap; (1) spesifikasi sistem, (2) desain arsitektur, (3) desain terperinci.

Tropos menawarkan analisis kebutuhan yang menyeluruh pada awal pengembangan perangkat lunak, hal ini menjadi salah satu kelebihan yang dimiliki metode ini. Tahapan pengembangan pada metodologi ini (1) *early requirement analysis, (2) Late requierment analysis, (3) Architectural desing, (4) Detailed Design, (5) implemetation.*

C. Analisis Dampak Perubahan (CIA).

CIA didefinisikan sebagai identifikasi konsekuensi potensial dari perubahan, atau memperkirakan apa yang perlu diubah untuk mencapai perubahan yang diinginkan. CIA adalah aktivitas mengelola perubahan [20], *Ripple effect* atau efek beriak yang sebabkan oleh perubahan kode sumber didefinisikan sebagai konsekuensi efek yang ditimbulkan dan pengaruhnya terhadap bagian lain dari sistem. CIA dapat dibagi menjadi 2 kategori umum yaitu *Dependency analysis* dan *Traceability analysis*, pada *dependency analysis* rincian hubungan antar entitas (variable atau fungi) disaring dari kode sumber. Sedangkan *traceablity analysis* analisis hubungan antar entitas (spesifikasi kebutuhan, rancangan arsitektur, rancangan terperinci, kode dan dokumentasi) telah diidentifikasi selama pengembangan dilaksanakan[21], [22].



Gambar II-3 proses CIA [23]

Pada gambar diatas proses CIA dipecah menjadi beberapa tahapan sebagai berikut: (1) menerjemahkan usulan perubahan menjadi spesifikasi perubahan, (2) menyaring informasi perubahan dari spesifikasi perubahan dan disimpan ke dalam repositori representasi internal. (3) menghitung dampak perubahan dari usulan perubahan yang diajukan. (4) mengembangkan estimasi dampak perubahan, berdasarkan ukuran dan kompleksitas perangkat lunak.(5) menghitung ukuran dampak dan keuntungan usulan perubahan, (6) pihak yang melakukan perawatan memberikan saran terhadap implikasi yang ditimbulkan dari permintaan perubahan kepada pengguna.

III. ANALISIS

A. Perbandingan Metodologi AOSE

Beragam pendapat mengenai definisi dari istilah agen, namun semua definisi sepakat bahwa agen pada dasarnya adalah sebuah komponen perangkat lunak khusus yang otonom, dapat menyediakan antarmuka yang *interoperable*, berperilaku seperti agen manusia, dan bekerja untuk beberapa klien untuk mencapai agendanya.

beberapa penelitian yang mengusulkan framework dalam mengevaluasi metodologi AOSE. Strun (2003) mengusulkan evaluasi berbasis fitur. Perbandingan dilakukan berdasarkan kumpulan set fitur yang ditentukan kemudian dinilai oleh koresponden. Teknik ini telah banyak diterapkan dalam berbagai hal.

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

Strun (2003) mengusulkan 4 kriteria utama yang akan dievaluasi pada metode AOSE

1. Konsep dan properti

Pada kriteria ini berkaitan dengan pertanyaan apakah metodologi telah sesuai dengan konsep dasar agen, yaitu; *Autonomy*, *Reactiveness, Proactiveness*, *Sociallity, Agent Belief, Desire*, *Intention, Message*, *Norm, Organization*, *Protocol,* dan *Role.*

2. Notasi, teknik dan pemodelan

Pada kriteria ini, atribut yang dinilai adalah *Accessibility, Analyzeability, Complexity Management*, *Executability Modularity*, *Preciseness*.

3. Proses

Kriteria ini berkaitan dengan aspek proses pengembangan yang terdapat pada metodologi. Atribut yang dievaluasi adalah:

Konteks Pengembangan, Cakupan siklus hidup, Pragmatics, *Resource*

4. Keahlian

Isu terkait memeriksa latar belakang ilmu yang dibutuhkan agar metodologi sesuai dengan keahlian yang dimiliki pengguna Kesesuaian bahasa pemrograman (arsitektur dan paradigma). Apakah metodologi ditargetkan sebagai implementasi bahasa pemrograman tertentu?. Apakah basis konsep yang digunakan metodologi adalah berdasarkan arsitektur spesifik atau paradigma tertentu?

5. Penerapan domain

Isu ini memeriksa kesesuaian penerapan metodologi pada domain tertentu seperti RealTime System dan sistem informasi.

6. Scalability

Isu yang terkait yaitu kesesuaian ukuran aplikasi yang akan dikembangkan sesuai dengan metodologi yang akan digunakan.

Terdapat penelitian yang menerapkan kriteria ini dalam mengevaluasi AOSE [24]–[26]. *Tropos* disebutkan memiliki kelebihan dengan adanya fitur *early requirement analysis*, sedangkan *Prometheus* pada saat penelitian dilakukan dianggap belum memenuhi properti *autonomy*. Pada kriteria *pragmatis*, *Prometheus* dinilai lebih praktis digunakan dari pada Tropos.

B. Klasifikasi Analisis Dampak Perubahan

Lebih dari 180 literatur yang mengusulkan teknik analisis dampak perubahan pada perangkat lunak[27]. Teknik CIA diklasifikasikan menurut kriteria yang ditentukan berdasarkan kebutuhan-kebutuhan sebagai berikut;

1. REQ1 : menyediakan informasi input (Change set) yang dibutuhkan (tipe artifak, tipe perubahan.
2. REQ2 : menyediakan informasi output (Impact Set) yang dihasilkan (tipe dampak).
3. REQ3 : menyediakan informasi algoritma atau teknologi yang digunakan.
4. REQ4 : menyediakan informasi dukungan pemodelan atau bahasa pemrograman.
5. REQ5 : menyediakan informasi ketersediaan tools.
6. REQ6 : menyediakan informasi skalabilitas teknik.
7. REQ7 : menyediakan informasi kualitas hasil yang dicapai.
8. REQ8 : menyediakan informasi interaksi manual dan upaya yang kerjakan.

Berdasarkan kebutuhan yang telah ditentukan, klasifikasi yang dihasilkan adalah

1. Scope of Analysis : menentukan tipe artifak perangkat lunak yang dianalisis, seperti kode, model arsitektur, atau kombinasi artifak (REQ1).
2. Granularity of Analysis : menentukan artifak yang digunakan sebagai input, dukungan perubahan, dan tipe dampak yang terdeteksi (REQ2).
3. Utilized Technique : menentukan algoritma yang digunakan (REQ3).
4. Supported Language : menentukan bahasa pemrograman atau bahasa pemodelan yang digunakan (REQ4).
5. Tool Support : menentukan ketersediaan tools (REQ5).

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

6. Scalability : menentukan kompleksitas teknik (REQ6).
7. Experimental Results : menentukan laporan tentang ukuran yang dihasilkan dari studi kasus (REQ7).
8. Style of Analysis : menerangkan bagaimana cara analisis dampak akan dilaksanakan (REQ8).

Tabel III-1 Tabel pemetaan klasifikasi dan kategori analisis dampak perubahan [28]

C. *Rancangan Implementasi*

Berdasarkan gambar II.3, diagram alur proses analisis dampak perubahan, sudah dapat diidentifikasi kebutuhan analisis dampak perubahan secara global. Pada tahap implementasi, metode AOSE yang akan digunakan untuk pengembangan model adalah *Prometheus.* Karena berdasarkan dari hasil analisis, *Tropos* dikembangkan dengan tujuan menganalisis seluruh kebutuhan secara detail. Pada tahap ini analisis kebutuhan sudah dapat dilakukan dengan menggunakan gambar alur yang telah ada. Dari gambar tersebut juga dapat diidentifikasi *Goals* yang akan dicapai dan menjadi tugas agen-agen nantinya.

Pada konteks CIA, agen yang akan dikembangkan diharapkan mampu menghasilkan data *change impact set* (CIS), data tersebut akan digunakan dalam melakukan *Dynamic Execution Trace Analysis*, *Call Graph Analysis dan Program Slicing*. Untuk sampel uji, aplikasi web dinamis yang digunakan adalah aplikasi yang sedang dalam proses *reengineering*. Tidak banyak perubahan yang dilakukan pada proses dan aturan bisnis. Dan untuk spesifikasi kebutuhan aplikasi sebelumnya tidak tersedia.

IV. IMPLEMENTASI DAN PENGUJIAN

Proses implementasi model berorientasi agen untuk analisis dampak perubahan pada aplikasi web dinamis yang dijadikan sebagai studi kasus, dengan menerapkan metode *prometheus* untuk pemodelan analisis dampak perubahan.

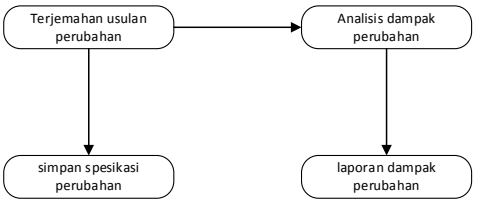
A. *System Spesification*

Pada fase pertama pembangunan model berorientasi agen, fokus utama pada fase ini meliputi :

1. Identifikasi goal dari analisis dampak perubahan, telah diterangkan tujuan utama dari pengembangan metode analisis dampak perubahan. Sesuai dengan alur proses analisis dampak perubahan yang telah dijelaskan dapat diidentifikasi *goal* sistem. *Goal* utama adalah

a. menerjemahkan usulan perubahan menjadi spesifikasi perubahan.
b. spesifikasi perubahan disimpan ke dalam repositori representasi internal.
c. Melakukan analisis dampak perubahan dari usulan perubahan yang diajukan. Proses ini dapat dilakukan bersamaan dengan usulan perubahan lainnya.
d. Menyediakan informasi dampak dari usulan perubahan pengguna,

Hasil dari 4 daftar *goal* utama yang ditetapkan, kemudian digunakan sebagai acuan dasar untuk menyaring *goal* yang lebih rinci dan kemudian dikelompokkan sesuai dengan fungsionalitasnya.



Gambar IV-1 Diagram Goal analisis dampak perubahan

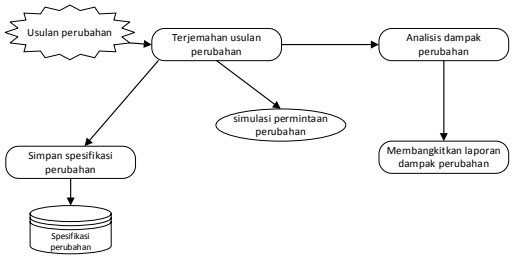
2. *Goal Refinement*

Setelah penjabaran goal utama dilakukan, diperlukan penyempurnaan dari goals tersebut. Penyempurnaan dilakukan dengan mempertanyakan "bagaimana cara goal tersebut dapat tercapai?", hal ini dilakukan agar goal yang telah dijabarkan menjadi lebih spesifik. Dan menghasilkan kelompok sub-goal berada dibawah goal utama. Pengelompokan ini kemudian menghasilkan penjelasan fungsionalitas sistem yang akan dibangun.

Sesuai dengan alur proses analisis dampak perubahan, goals utama yang terdapat analisis dampak perubahan telah cukup mendetail. Oleh

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

sebab itu tahap goals refinement tidak dilakukan. Berikut adalah gambar pengelompokan goals analisis dampak perangkat lunak



Gambar IV-2 Diagram Goal Refinement

B. Rancangan arsitektural

Pada metode *prometheus* terdapat tiga aspek kunci yang perlu diperhatikan dalam merancang arsitektural sistem yang akan dibangun :

1. Memutuskan tipe agen yang digunakan pada system.
2. Menjabarkan interaksi antar agen menggunakan diagram interaksi dan protokol interaksi.
3. Merancang struktur sistem secara keseluruhan.

Rancangan arsitektural akan menjelaskan di bagian mana pada sistem agen akan menjalankan tugasnya dan bagaimana cara agen saling berinteraksi dalam memenuhi fungsionalitas system.

1. Agen Translator

Memiliki 2 *goal*, menerjemahkan usulan perubahan pengguna dalam bentuk *user story*, dan memberikan *feedback* hasil analisis dalam bahasa alami pula. Artifak yang dihasilkan oleh agen dapat digunakan sebagai dokumentasi pengujian. Menggunakan bahasa *Gherkin* [29], dan *codeception* [30] sebagai agen.

2. Agen profiler

Mempunyai kemampuan melakukan analis dampak secara dinamis pada saat *runtime*. Agen ini mampu menyediakan informasi bagian-bagian kode program yang termasuk dalam suatu proses eksekusi. Agen ini bersifat otonomi tidak tergantung pada pada agen lain. Namun agen ini mampu menyediakan informasi seperti *code coverage,* informasi *traceability* pada level kode program, dan bahkan dapat menemukan *dead code* atau potongan method atau fungsi pada kode program yang ada tapi tidak pernah digunakan. Hasil dari Informasi ini dapat digunakan pada praktek *refactoring*. *Tools* yang diguakan sebagai agen adalah *Xdebug.*

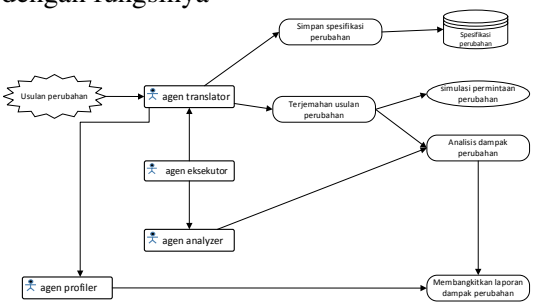
3. Agen analyzer

Agen ini mempunyai tugas melakukan analisis secara statis pada aplikasi web. Agen ini mampu menghasilkan grafik keterkaitan pada level abstrak perangkat lunak, seperti kelas, *interface,* fungsi. Agen ini dapat diberi batasan analisis statis pada level pake. Grafik yang dihasilkan oleh agen ini adalah visualisasi *call graph, inheritance graph* dan *usage* graph. Pada penelitian ini yang bertindak sebagai agen *analyzer* adalah paket *PhpDependecyAnalysis*

4. *Agen Eksekutor*

Merupakan agen yang mempunyai tugas melakukan eksekusi atau bertindak sebagai automator *task* pada agen yang telah dijabarkan. Perintah eksekusi aksi agen yang lain dimulai dari instruksi-instruksi yang dimiliki oleh agen ini. Pada penelitian ini yang menjadi agen eksekutor adalah *Gulp*[31].

Dari daftar agen yang diidentifikasi di atas, berikut adalah pemetaan posisi agen sesuai dengan fungsinya



Gambar IV-3 Pemetaan agen pada model.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian analisis dampak perubahan pada aplikasi web dinamis, maka dapat diambil kesimpulan sebagai berikut;

1. Analisis dampak perubahan yang dilakukan pada aplikasi web dinamis mampu menerjemahkan bahasa alami

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

permintaan perubahan dari pengguna, dan memberikan output hasil analisis dengan bahasa yang mudah dimengerti.

2. Beberapa tipe agen yang digunakan selama proses analisis telah mampu memenuhi kriteria otomasi dan reactivity agen. Seperti agen translator, yang mampu membangkitkan laporan analisis ke dalam format file html.
3. Hasil artifak analisis dampak perubahan dapat digunakan ulang pada proses pengujian perangkat lunak, yang dalam konteks penelitian ini artifak dapat digunakan pada proses acceptance testing, hal ini secara tidak langsung menambah kualitas reusability pada produk perangkat lunak.
4. Implementasi model berorientasi agen pada penelitian ini sangat bergantung pada tools yang sesuai dengan kriteria agen, dan tools yang digunakan spesifik untuk bahasa pemrograman PHP.
5. Model analisis dampak perubahan dapat diimplementasikan pada proyek perangkat lunak berbasis web berskala menengah ke bawah.

B. Saran

Berdasarkan hasil penelitian dan pembahasan serta kesimpulan yang telah dikemukakan, berikut ini adalah beberapa saran yang dapat menjadi masukan untuk pengembangan selanjutnya:

1. Pengembangan agen untuk analisis dampak perubahan pada level requirement dan design, khususnya traceability dan depedency antara kode program dengan artifak requirement.
2. Pengembangan visualisasi laporan hasil dampak perubahan yang lebih detail dan luas.
3. Pengembangan agen-agen pada model untuk bahasa pemrograman lain seperti JAVA, C#, dan Ruby.
4. Pengembangan analisis dampak perubahan pada level unit dan fungsi yang pada penelitian ini belum mampu di implementasikan.

REFERENSI

A. E. Hassan dan R. C. Holt, “Predicting change propagation in software systems,” dalam 20th IEEE International Conference on Software Maintenance, 2004. Proceedings., 2004, pp. 284–293.

R. Yongchang, X. Tao, L. Zhongjing, dan C. Xiaoji, “Software Maintenance Process Model and Contrastive Analysis,” 2011 Int. Conf. Inf. Manag. Innov. Manag. Ind. Eng., vol. 3, pp. 169–172, 2011.

B. G. Ryder dan F. Tip, “Change impact analysis for object-oriented programs,” dalam Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering - PASTE ’01, 2001, pp. 46–53.

L. Badri, M. Badri, dan D. St-Yves, “Supporting predictive change impact analysis: a control call graph based technique,” 12th Asia-Pacific Softw. Eng. Conf., p. 9 pp., 2005.

T. Feng dan J. Maletic, “Applying dynamic change impact analysis in component-based architecture design,” Seventh ACIS Int. Conf. Softw. Eng. Artif. Intell. Networking, Parallel/Distributed Comput. SNPD 2006., 2006.

J. Hassine, J. Rilling, J. Hewitt, dan R. Dssouli, “Change impact analysis for requirement evolution using use case maps,” Int. Work. Princ. Softw. Evol., vol. 2005, pp. 81–90, 2005.

J. Law dan G. Rothermel, “Whole program path-based dynamic impact analysis,” dalam 25th International Conference on Software Engineering 2003 Proceedings, 2003, vol. 6, pp. 308–318.

T. Apiwattanapong, A. Orso, dan M. J. Harrold, “Efficient and precise dynamic impact analysis using execute-after sequences,” dalam Proceedings of the 27th international conference on Software engineering - ICSE ’05, 2005, p. 432.

T. Goradia, “Dynamic impact analysis: A cost-effective technique to enforce error-propagation,” dalam ACM SIGSOFT Software Engineering Notes, 1993, vol. 18, no. 3, pp. 171–181.

Gupta, “An Efficient Dynamic Impact

p-ISSN : 2407 – 1846
e-ISSN : 2460 – 8416
Website : jurnal.umj.ac.id/index.php/semnastek

Analysis using Definition and Usage Information.," Int. J. Digit. Content Technol. its Appl., 2009.

A. Mesbah, A. van Deursen, dan S. Lenselink, "Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes," ACM Trans. Web, vol. 6, no. 1, pp. 1–30, 2012.

S. Franklin dan A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," Intell. agents III agent Theor. Archit. Lang., pp. 21–35, 1997.

M. V Dignum, A model for organizational interaction: based on agents, founded in logic, no. August. 2004.

G. Caire, W. Coulier, F. Garijo, J. Gomez, J. Pavon, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, dan P. Massonet, "Agent Oriented Analysis Using Message / UML," Proceeding AOSE '01 Revis. Pap. Invit. Contrib. from Second Int. Work. Agent-Oriented Softw. Eng. II, pp. 119–135, 2002.

Y. Shoham, "Agent-oriented programming," vol. 60, pp. 51–92, 1993.

M. Wooldridge dan N. R. Jennings, "Intelligent agents: theory and practice," Knowl. Eng. Rev., vol. 10, no. 02, p. 115, 2009.

M. P. George, "BDI Agents: From Theory to Practice," 1995.

P. Giorgini dan B. Henderson-Sellers, "Agent-Oriented Methodologies: an introduction," Agent-Oriented Methodol., pp. 1–19, 2005.

L. Padgham dan M. Winikoff, "Developing Intelligent Agent Systems A practical guide."

C. W. Angelis, "An Empirical Study on Change Impact."

P. Jönsson, Impact Analysis Organisational Views and Support Techniques. 2005.

C. Ackermann dan M. Lindvall, "Understanding change requests to predict software impact," Proc. 30th Annu. IEEE/NASA Softw. Eng. Work. SEW-30, vol. 30, pp. 66–75, 2006.

R. Moreton, "A process model for software maintenance," J Inf Technol, vol. 5, no. 2, pp. 100–104, Jun. 1990.

K. H. Dam, "Evaluating and comparing agent-oriented software engineering methodologies," 2003.

Q.-N. Tran, G. Low, dan M.-A. Williams, "A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies," Found. Intell. Syst., vol. 2871, pp. 613–617, 2003.

A. Kumar, "Comparative Analysis & Evaluation of Agent Oriented Methodologies Using Enhanced Feature Analysis Framework," vol. 2, no. 1, 2012.

S. Lehnert, "Multiperspective Change Impact Analysis to Support Software Maintenance and Reengineering," pp. 1–221, 2015.

S. Lehnert, "A Review of Software Change Impact Analysis," Technology, p. 39, 2011.

"gherkin," 2016. [Online]. Available: https://cucumber.io/docs/reference. [Di akses: 20-Aug-2016].

"Codeception," 2016. [Online]. Available: http://codeception.com/. [Di akses: 20-Aug-2016].

"gulp.js - the streaming build system," 2016. [Online]. Available: http://gulpjs.com/. [Di akses: 21-Aug-2016].