Jurnal Siliwangi Vol. 2. No. 1. Mei. 2016 **ISSN 2477-3891**
Seri Sains dan Teknologi

STRUKTUR *KNOWLEDGE BASE* SEBAGAI KOMPONEN PEMBENTUK PERANGKAT LUNAK *SELF-ADAPTIVE*

Aradea[1], Iping Supriatna[2], Kridanto Surendro[3]

[1]Department of Informatics Engineering Faculty of Engineering, Siliwangi University
[2]Bandung Institute of Technology
[3]Bandung Institute of Technology
email : aradea@unsil.ac.id[1], iping@informatika.org[2], endro@informatika.org[3]

Abstrak

Suatu sistem terlahir didasari atas target visi dari concern yang digelutinya masing-masing. Dalam perjalanannya, setiap sistem senantiasa bersentuhan dengan kompleksitas dan ketidakpastian didalam lingkungannya. Hal ini berhubungan dengan bagaimana sistem itu hidup, yaitu mampu tumbuh dan terus berkembang. Keberadaan perangkat lunak self-adaptive merupakan jawaban atas persoalan dan tantangan tersebut, suatu perangkat lunak sebagai penggerak utama sistem, dituntut memiliki kemampuan untuk dapat memahami dan bertindak atas apa yang terjadi didalam lingkungannya, termasuk penanganan pertumbuhan pengetahuannya. Dalam kebanyakan kasus, konstruksi struktur pengetahuan biasanya dirancang melalui suatu kekhusuan, sehingga lingkup yang dilayaninya menjadi terbatas. Makalah ini mengusulkan model representasi pengetahuan sebagai jaminan bagi pengelolaan sistem dan kebutuhan adaptasinya, model diformulasikan melalui rule-based systems dengan struktur knowledge base yang lebih generik dan luwes, sehingga mampu menangani pertumbuhan pengetahuan yang merepresentasikan ragam konteks dan perilaku sistem.

Kata kunci : *self-adaptive software, intelligent systems, knowledge base, rule-based systems.*

I. PENDAHULUAN

Perangkat lunak saat ini telah menjadi bagian penting dalam berbagai aktivitas kehidupan, dimana beragam unsur sistem didunia nyata dapat berinteraksi dengan perangkat lunak. Keterlibatan berbagai unsur dan aktivitasnya tersebut, memunculkan persoalan kompleksitas didalam pengembangan perangkat lunak, sebut saja misalnya lahirnya kebutuhan akan sistem terdistribusi, heterogenitas, desentralisasi, inter-dependen yang beroperasi pada lingkungan dinamis dan tidak dapat diprediksi.

Kondisi ini menuntut bahwa perangkat lunak harus memiliki kemampuan beradaptasi terhadap perubahan lingkungannya, karena kebutuhan sistem akan terus tumbuh dan berkembang. Pendekatan tradisional untuk mengkonstruksi perangkat lunak tidak dapat menjadi solusi dalam persoalan ini, dimana perilaku dan fitur sistem ditetapkan hanya untuk menangani fungsi operasional, tanpa mempertimbangkan kebutuhan pertumbuhan sistem dan kemampuan adaptasinya.

Pendekatan *self-adaptive* merupakan solusi dari persoalan ini, namun penetapan spesifikasi kebutuhanya membutuhkan kajian tersendiri, yang tidak sama dengan kebutuhan sistem tradisional. Pada makalah ini kami mengusulkan suatu pendekatan, dimana sistem memiliki *knowledge* sehingga mampu mengatasi persoalan pertumbuhan sistem, melalui penangkapan fakta-fakta pada dunia nyata. Strategi kunci yang dikembangkan adalah memformulasikan *rule-based systems* dengan struktur *knowledge base* yang dikonstruksi secara lebih generik, sehingga mampu merepresentasikan variabilitas dalam konteks dan perilaku sistem.

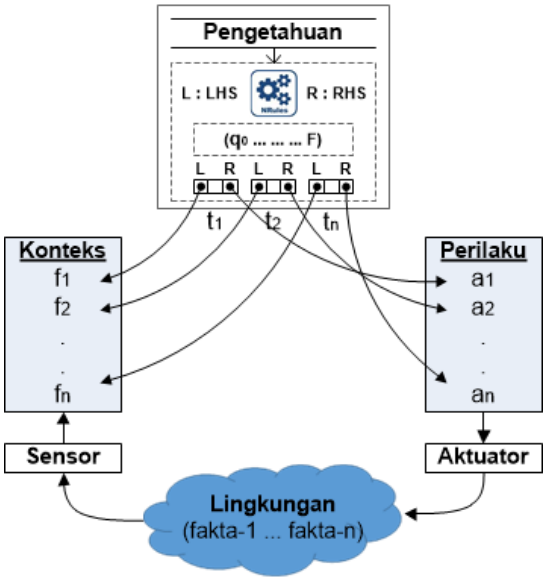
Motivasi dari pengembangan struktur *knowledge base* ini, didasari atas fakta bahwa pendekatan *rule-based* dalam kebanyakan penerpannya, hanya terfokus untuk pengaturan perilaku perangkat lunak melalui suatu struktur *knowledge base* yang dirancang secara khusus. Persoalan yang mungkin muncul dari kekhususan struktur tersebut adalah lingkup variabilitas yang dapat dilayani, sehingga timbul suatu pemikiran dengan memfomulasikan suatu model struktur yang lebih umum dan lebih luwes, sehingga penerapannya pada berbagai lingkup variabilitas dapat diperluas secara fleksibel.

II. METODOLOGI

Metodologi yang dikembangkan dilandasi dari kajian yang telah kami lakukan sebelumnya [1], yaitu karakteristik lingkungan sistem saat ini memiliki sifat yang sangat dinamis dan heterogen, sehingga pertumbuhan pengetahuan akan berkembang dan berubah sangat cepat untuk menyesuaikan diri terhadap lingkungan. Hal ini merupakan kondisi yang kritis, apabila tidak dapat

mengevaluasi dan merencanakan suatu strategi yang tepat bagi pengaturan suatu pengetahuan.

Terinspirasi dari teori *intelligent systems* [2] bahwa dalam menyelesaikan suatu persoalan, sistem yang cerdas harus terdiri dari *state* awal, koleksi operator, dan himpunan *state* target. Koleksi dari *state* awal dan himpunan *state* target disebut sebagai koleksi *state*, sementara koleksi lebih lanjut dari *state* dan operator inilah yang dikenal sebagai pengetahuan (*knowledge*). Berdasarkan pernyataan teori tersebut, kami mengembangkan usulan model seperti dapat dilihat pada Gambar 1.



Gambar 1. Model perangkat lunak *self-adaptive*

Model terdiri dari 5-*tuple* :

a. $\sum$: $f_1...f_n$ = informasi konteks merupakan himpunan masukan.
b. Q : $a_1...a_n$ = perilaku sistem terdiri dari himpunan *state* berhingga.
c. δ : $t_1...t_n$ = fungsi transisi atau operator yang bertindak sebagai fungsi pengubah *state*.
d. q_0 = data awal sebagai *state* awal.
e. F = informasi *state* target atau himpunan *state* akhir.

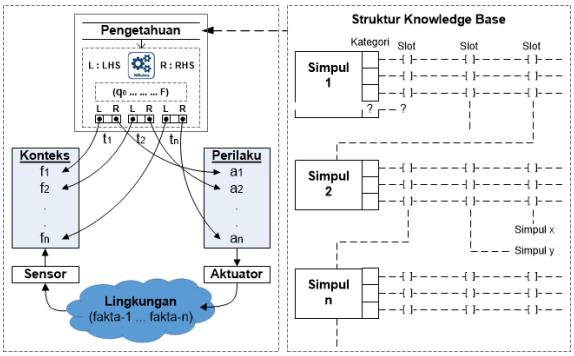
Sementara representasi *rule-based* yang menjadi acuan proses adaptasi, didefinisikan dengan struktur dasar *rule* sebagai berikut :

WHEN <*event*> *; satu atau lebih transisi state*
IF <*condition*> *; kondisi yang harus dipenuhi untuk memicu aksi*
THEN <*action*> *; satu atau lebih aksi ketika event berlangsung*
VALID-TIME <*time period*> *; periode waktu selama aksi kontrol berlaku, selama adaptasi perlu diterapkan*

Informasi koneks menjadi himpunan masukan yang akan dimonitor sebagai <*event*> dan ditangkap oleh *context processor*, yaitu fakta-fakta dari setiap informasi konteks. *Controller* mengevaluasi <*condition*>, mengacu pada <*event*> tertentu yang terjadi didalam himpunan *state*. Akhirnya akan memicu komponen *action performer* melakukan <*action*> adaptasi melalui fungsi transisi terhadap <*condition*> yang berlaku pada <*time period*> tertentu, sehingga dapat dicapai *state* target sebagai perilaku yang diharapkan.

Dimana $\sum$: informasi konteks berupa himpunan fakta ($f_1...f_n$) yang ditangkap dari *event* lingkungan dan dapat menentukan perubahan *state*. Q : perilaku aksi ($a_1...a_n$) berupa himpunan perilaku *state* sistem, dari mulai *state* awal (q_0) hingga *state* yang dapat menjadi target (F). δ : fungsi transisi ($t_1...t_n$) sebagai waktu *action* adaptasi berdasarkan evaluasi terhadap *condition*, untuk merubah *state* awal (q_0) menjadi *state* target (F). Kualitas dari mesin inferensi sangat bergantung pada pemilihan salah satu *state*, dimana *state* yang dipilih sebagai aksi adaptasi pada *class right hand side* (RHS), sangat ditentukan oleh ekspresi yang berada pada *class left hand side* (LHS), yaitu kesesuaian antara *rule* dan fakta.

Rincian penjelasan dari formulasi *rule-based* ini dapat dilihat pada makalah kami sebelumnya [3][4], sementara *requirements* yang melatar belakangi pembentukan *rule-based* tersebut dapat dilihat pada makalah kami [5][6]. Pada bahasan makalah ini, kajian difokuskan pada konstruksi struktur *knowledge base* yang dapat membentuk kemampuan adaptasi menjadi lebih fleksibel dan luwes. Ilustrasi posisi struktur *knowledge base* dalam model *self-adaptive* Gambar 1, seperti ditunjukan pada Gambar 2. Setiap perubahan dalam sistem dapat dinyatakan sebagai perubahan rangkaian *simpul* [7], setiap *state* dimodelkan kedalam simpul dan perubahan *state* digambarkan kedalam busur. Dengan demikian, struktur yang dipergunakan untuk menangani keterkaitan antara elemen *knowledge* adalah berupa suatu graf berarah.



Gambar 2. Struktur *knowledge base* dalam model *self-adaptive*

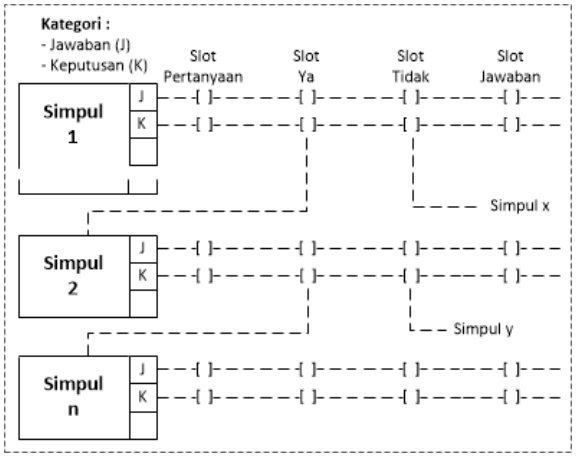
Pada rancangan struktur *knowledge base* yang dikembangkan, setiap simpul dalam graf dapat memiliki beberapa kategori *slot*, dan setiap kategori mempunyai sejumlah atribut yang dapat bertambah. Mekanisme yang dikembangkan, dipersiapkan untuk memenuhi lingkupi suatu daur hidup sistem, yaitu lahir (*create*), digunakan (*use*), diperbaharui (*update*), dan dihapus (*delete*). Sehingga modul-modul pengelolaan pengetahuan yang dikonstruksi terdiri dari, pembentukan *slot*, perluasan *slot*, penyesuaian *slot*, dan pemeriksaan *slot*. Mekanisme penelusuran dari suatu ruang pengetahuan, dilakukan melalui modul akses *slot*, kendali terhadap lintasan, dan telusur tunda. Sementara mekanisme yang dapat menangani masukan berupa informasi konteks dan perubahannya, serta keluaran dalam bentuk perilaku aksi-aksi adaptasi, diwujudkan dalam modul sensor dan aktuator sebagai modul *interface*.

III. HASIL DAN PEMBAHASAN

Berdasarkan rincian model yang telah dibahas pada bagian 2, bagian ini mendeskripsikan kajian dan hasil penerapannya. Fokus bahasan diarahkan pada konstruksi struktur *knowledge base* pada suatu contoh kasus, dan evaluasi perbandingannya dengan karya para peneliti terkait sebelumnya.

A. Studi Kasus

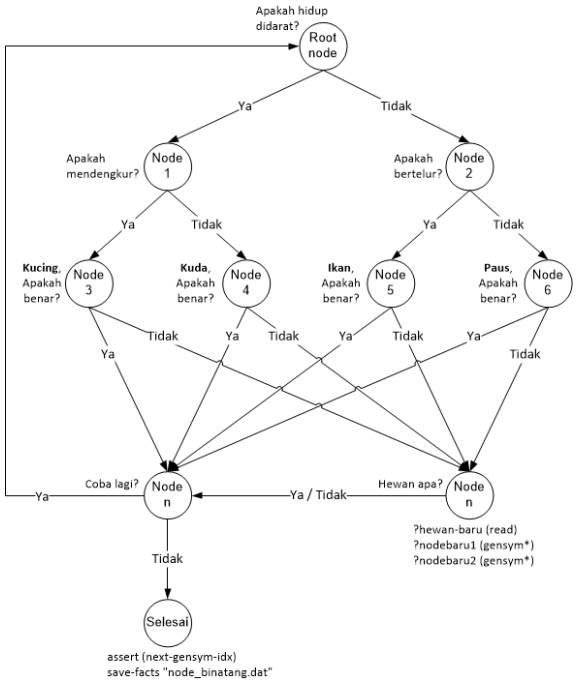
Studi kasus ini mendeskripsikan struktur dasar yang nantinya akan dikembangkan lebih lanjut, untuk menangani keterkaitan antara elemen pengetahuan dalam suatu graf berarah. Sebagai motivasi kasus, dikembangkan sebuah aplikasi sederhana untuk mendefinisikan tebak-tebakan taksonomi binatang. Mekanisme dasarnya adalah sistem menangkap fakta dari masukan *user* atas pertanyaan yang diajukan, kemudian menyimpannya sebagai pengetahuan. Sehingga pengetahuan sistem dapat bertambah, dan dapat menyimpulkan jawaban atas fakta-fakta yang diterimanya. Termasuk ketika jawaban sistem dinyatakan bernilai salah oleh *user*, maka sistem memberikan alternatif jawaban lain dari pengetahuan yang telah didapatkannya, dan jika *user* masih menyatakan salah, maka sistem akan bertanya atas jawaban yang benar dan menyimpannya sebagai pengetahuan baru.



Gambar 3. Struktur *knowledge base*

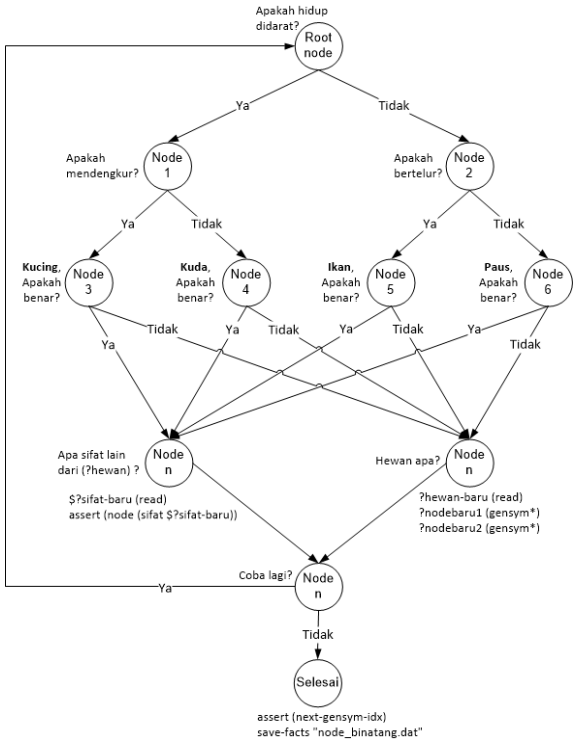
Dalam implementasinya setiap simpul (*node*) memiliki dua kategori *slot*, yaitu tipe jawaban atau tipe keputusan, dan setiap kategori mempunyai sejumlah atribut yang dapat bertambah, yaitu atribut pertanyaan, ya, tidak, dan jawaban. Atribut *slot* pertanyaan dan jawaban merupakan pengetahuan yang dapat bertambah dari fakta yang ditangkap, sementara atribut *slot* ya dan tidak adalah representasi dari perubahan *state*, yang akan menentukan keputusan relasi terhadap *node* lain berdasarkan fakta yang diterimanya. Ilustrasi dari struktur graf ini seperti ditunjukan pada Gambar 3.

Sementara alur kerja struktur graf Gambar 3 dapat dilihat pada Gambar 4. Pada saat sistem tidak dapat menebak jawaban dengan benar, maka sistem akan menanyakan jawabannya, kemudian sistem akan menangkap fakta tersebut sebagai pengetahuan baru. Proses ini dilakukan saat node 3, *node* 4, *node* 5, dan *node* 6 bernilai salah (tidak), dan setiap kali *user* memberikan masukan fakta baru tersebut, maka graf akan membentuk *node* baru secara otomatis, dan hal ini terus terjadi selama kondisi tersebut berlaku.



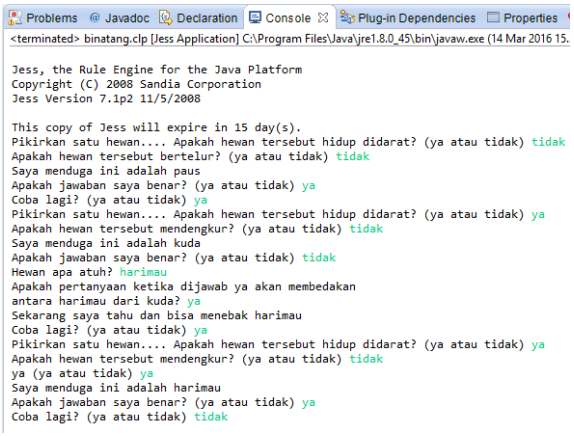
Gambar 4. Alur graf skenario ke-1

Sekenario berikutnya adalah ketika sistem dapat menebak jawaban dengan benar, maka sistem akan mengajukan pertanyaan lanjutan, dan saat *user* memberikan jawabannya, maka fakta tersebut akan disimpan sebagai pengetahuan baru lagi. Pada Gambar 5 proses ini dilakukan saat *node* 3, *node* 4, *node* 5, dan *node* 6 bernilai benar (ya), sehingga proses ini juga akan membentuk *node* baru secara otomatis selama kondisi tersebut berlaku.



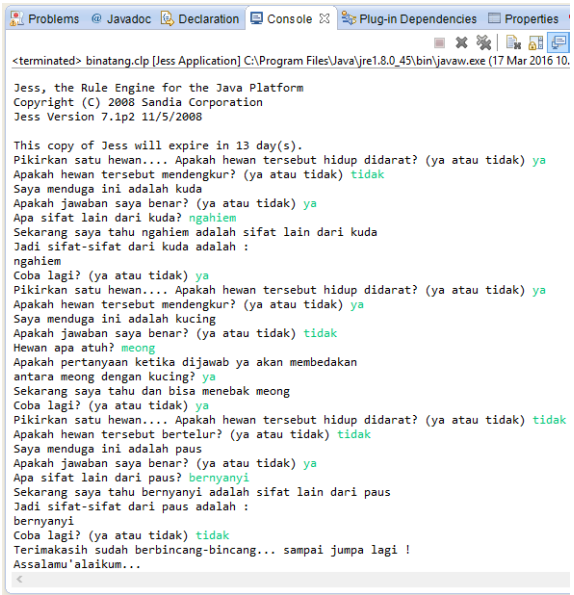
Gambar 5. Alur graf skenario ke-2

Prototipe perangkat lunak dikembangkan dengan menggunakan JESS 7.1p2 (*java expert system shell*), suatu perangkat lunak *rule engines* yang menginterpretasikan *rule* dan fakta yang diekspresikan dalam bahasa pemrograman. Sementara lingkungan pemrograman yang digunakan adalah Eclipse Standard/SDK, versi Kepler Service Release 2.



Gambar 6. Dialog layar skenario ke-1

Berdasarkan hasil implementasi yang telah dilakukan, contoh tampilan dialog layar dari prototipe perangkat lunak ini seperti dapat dilihat pada Gambar 6 dan 7. Sebagai contoh ilustrasi hasil, penambahan *node* secara otomatis yang merepresentasikan penambahan fakta baru alur graf skenario ke-1 Gambar 4, ditunjukan pada Tabel 1, dan hasil penambahan fakta baru alur graf skenario ke-2 Gambar 5, dapat dilihat pada Tabel 2.



Gambar 7. Dialog layar skenario ke-2

Tabel 1. Pertumbuhan pengetahuan ke-1

Fakta	node_binatang.dat
Fakta Awal (F.1)	V.1 : Fakta awal (node (nama root) (tipe keputusan) (pertanyaan Pikirkan satu hewan.... Apakah hewan tersebut hidup didarat?") (ya-node node1) (tidak-node node2) (jawaban nil)) (node (nama node1) (tipe keputusan) (pertanyaan Apakah hewan tersebut mendengkur?") (ya-node node3) (tidak-node node4) (jawaban nil)) (node (nama node2) (tipe keputusan) (pertanyaan Apakah hewan tersebut bertelur?") (ya-node node5) (tidak-node node6) (jawaban nil)) (node (nama node3) (tipe jawaban) (pertanyaan nil) (ya-node nil) tidak-node nil) (jawaban kucing)) (node (nama node4) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kuda)) (node (nama node5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban ikan)) (node (nama node6) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban paus)) (MAIN::next-gensym-idx "3")
Fakta Baru (F.2)	F.2 : Setelah diinput fakta baru - harimau (node (nama root) (tipe keputusan) (pertanyaan Pikirkan satu hewan.... Apakah hewan tersebut hidup didarat?") (ya-node node1) (tidak-node node2) (jawaban nil)) (node (nama node1) (tipe keputusan) (pertanyaan Apakah hewan tersebut mendengkur?") (ya-node node3) (tidak-node node4) (jawaban nil)) (node (nama node2) (tipe keputusan) (pertanyaan Apakah hewan tersebut bertelur?")(ya-node node5) (tidak-node node6) (jawaban nil)) (node (nama node3) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kucing)) (node (nama node4) (tipe keputusan) (pertanyaan "ya") (ya-node gen4) (tidak-node gen5) (jawaban kuda)) (node (nama node5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban ikan)) (node (nama node6) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban paus)) (node (nama gen4) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban harimau)) (node (nama gen5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kuda)) (MAIN::next-gensym-idx "6")

Tabel 2. Pertumbuhan pengetahuan ke-2

Fakta	node_binatang.dat
Fakta Awal (F.1)	F.1 : Fakta awal (node (nama root) (tipe keputusan) (pertanyaan Pikirkan satu hewan... Apakah hewan tersebut hidup didarat?") (ya-node node1) (tidak-node node2) (jawaban nil)) (node (nama node1) (tipe keputusan) (pertanyaan Apakah hewan tersebut mendengkur?") (ya-node node3) (tidak-node node4) (jawaban nil)) (node (nama node2) (tipe keputusan) (pertanyaan Apakah hewan tersebut bertelur?") (ya-node node5) (tidak-node node6) (jawaban nil)) (node (nama node3) (tipe jawaban) (pertanyaan nil) (ya-node nil) tidak-node nil) (jawaban kucing)) (node (nama node4) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kuda)) (node (nama node5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban ikan)) (node (nama node6) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban paus)) (MAIN::next-gensym-idx "3")
Fakta Baru (F.2)	F.2 : Setelah diinput sifat-sifat baru hewan (node (nama root) (tipe keputusan) (pertanyaan Pikirkan satu hewan... Apakah hewan tersebut hidup didarat?") (ya-node node1) (tidak-node node2) (jawaban nil) (sifat)) node (nama node1) (tipe keputusan) (pertanyaan Apakah hewan tersebut mendengkur?") (ya-node node3) (tidak-node node4) (jawaban nil) (sifat)) (node (nama node2) (tipe keputusan) (pertanyaan Apakah hewan tersebut bertelur?") (ya-node node5)(tidak-node node6)(jawaban nil) (sifat)) (node (nama node3) (tipe keputusan) (pertanyaan "ya") (ya-node gen4) (tidak-node gen5) (jawaban kucing) (sifat penurut)) (node (nama node4) (tipe keputusan) (pertanyaan "ya") (ya-node gen6) (tidak-node gen7) (jawaban kuda) (sifat betenaga kuat)) (node (nama node5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban ikan) (sifat bertelur)) (node (nama node6) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban paus) (sifat beranak)) (node (nama gen4) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kuda) (sifat ngahiem)) (node (nama gen5) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban meong) (sifat memanjat)) (node (nama gen6) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban kucing) (sifat beranak)) (node (nama gen7) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban paus) (sifat bernyanyi)) (node (nama gen8) (tipe jawaban) (pertanyaan nil) (ya-node nil) (tidak-node nil) (jawaban

```
kucing)
    (sifat menggosokan tubuh))
(node (nama gen9) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban meong)
    (sifat mengeong))
(node (nama gen10) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban kucing)
    (sifat mendesis))
(node (nama gen11) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban kuda)
    (sifat berlari))
(node (nama gen12) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban kuda)
    (sifat penurut))
(node (nama gen13) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban ikan)
    (sifat permukaan air))
(node (nama gen14) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban paus)
    (sifat bermigrasi))
(node (nama gen15) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban paus)
    (sifat beranak))
(node (nama gen16) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban
    harimau) (sifat ganas))
(node (nama gen17) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban kuda)
    (sifat beranak))
(node (nama gen18) (tipe jawaban) (pertanyaan nil)
    (ya-node nil) (tidak-node nil) (jawaban
    harimau) (sifat mengaum))
(MAIN::next-gensym-idx "19")
```

B. Pekerjaan Terkait

Para peneliti telah melakukan berbagai upaya rekayasa untuk kebutuhan pengembangan perangkat lunak *self-adaptive*, yaitu bagaimana mewujudkan suatu sistem yang adaptif, dari mulai aktivitas perencanaan dan pengembangan, sampai pada saat sistem tersebut sedang berjalan, dan area penelitian ini memiliki cakupan yang cukup luas [8]. Saat ini terdapat ragam bentuk perangkat lunak *self-adaptive*, misalnya model Tropos4AS [9], memberikan kemampuan variabilitas yang tinggi untuk kebutuhan informasi saat *run-time* melalui pendekatan *transition rule*. Model lainnya yaitu Adaptive STSs [10], mengusulkan model arsitektur dengan kemampuan *self-reconfiguration* bagi perilaku *multi-agent*, namun terkait preferensi pengguna dan kebutuhan evolusi belum tercakup. Model kami sebenarnya dapat menjadi pelengkap untuk kedua karya ini, karena menyediakan kemampuan untuk penanganan pertumbuhan pengetahuan melalui konsep *context-aware*.

Model CARE [11], mengeksploitasi *ontology* untuk mengantisipasi perubahan konteks dan sumber daya yang beragam, namun mekanisme penalaran dalam model ini sebenarnya dapat ditingkatkan, disini kami mengusulkan model *rule* dinamis sehingga penalaran saat *run-time* dapat dilakukan secara otomatis. Model ZANSHIN [12] dan Self-adaptive ASMs [13], sangat menekankan pada fungsi generik *feedback loop* untuk penalaran saat *run-time*, namun representasi entitas dari problem domain sebagai domain models, pada kedua model ini belum tercermin, sementara dalam model yang kami kembangkan, hal tersebut merupakan salah satu *concern* yang diusulkan. Model REFAS [14], merupakan model *requirements* sistem *self-adaptive* untuk mengatasi ketidakpastian, melalui penyediaan bahasa yang cukup memiliki kekuatan ekspresi, namun terkait representasi pengetahuan sebagai kebutuhan penalaran saat *run-time* seperti dalam model kami, belum tercakup.

IV. KESIMPULAN

Paparan dalam makalah ini merupakan bagian dari aktivitas penelitian yang sedang berjalan. Fokus bahasan menyoroti formulasi dari suatu struktur *knowledge base*, sebagai salah satu komponen pembentuk perangkat lunak *self-adaptive*. Deskripsi kerja prototipe perangkat lunak yang dikembangkan, mencerminkan kemudahan dalam menangani pengetahuan yang dapat terus tumbuh dan berkembang. Namun eksperimen awal ini membutuhkan eksperimen lanjutan secara lebih rinci dan menyeluruh. Selain kemudahan dalam proses penambahan pengetahuan, diperlukan kriteria lainnya yang merupakan bagian utama dalam suatu motor inferensi, misalnya menjaga konsistensi pengetahuan, serta efisiensi dari ruang dan mekanisme pencarian.

Hal ini akan berhubungan dengan strategi pengelolaan ruang pengetahuan dan pengendalian masukan pengetahuan, melalui pendefinisian modul-modul sistem secara lebih spesifik. Selain itu, penangkapan fakta dalam eksperimen ini masih mengandalkan masukan dari *user*, sehingga pengembangan fungsi suatu modul sensor menjadi salah satu agenda penelitian kedepan. Sisa dari pekerjaan tersebut merupakan aktivitas penelitian berikutnya yang akan dilakukan, untuk mewujudkan suatu model perangkat lunak *self-adaptive* secara utuh.

DAFTAR PUSTAKA

[1] Aradea. Supriana, I. Surendro, K. 2014. An overview of multi agent system approach in knowledge management model. *Proceeding of International Conference on Information Technology Systems and Innovation* (ICITSI). IEEE. ITB.

[2] Supriana, I. Agustin, R. 2012. The multi control strategy for intelligent system. *Proceeding of International Conference on Intelligence System and Informatics* (ISI). IEEE. Parahyangan University Bandung.

[3] Supriana, I. Aradea. 2015. Automatically relation modeling on spatial relationship as self-adaptation ability. *Proceeding of Internationl Conference on Advanced Informatics: Concept, Theory, Application* (ICAICTA). IEEE. Thailand. 2015.

[4] Supriana, I. Aradea. 2016. Model *self-adaptive* sebagai landasan sistem untuk menunjang penumbuhan komunitas. *Keynote Paper Prosiding Seminar Nasional Teknologi Informasi dan Komunikasi* (SENTIKA). Vol. 6.

[5] Aradea. Supriana, I. Surendro, K. 2015. Prinsip paradigma agen dalam menjamin keberlangsungan hidup sistem. *Prosiding Konferensi Nasional Sistem Informasi* (KNSI), ITB - Universitas Klabat Sulawesi.

[6] Aradea. Supriana, I. Surendro, K. 2015. Konsepsi data dan informasi sebagai penyedia layanan pengetahuan. *Prosiding Konferensi Nasional Sistem Informasi* (KNSI), ITB - Universitas Klabat Sulawesi.

[7] Supriana, I. Wahyudin, I. Mulyanto, A. 1989. Pengembangan motor inferensi untuk aplikasi sistem pakar dalam model diagnosa. *Technical Report*. Sekolah Teknik Elektro dan Informatika ITB.

[8] Aradea. Supriana, I. Surendro, K. 2015. Roadmap dan area penelitian self-adaptive systems. *Prosiding Seminar Nasional Teknik Informatika dan Sistem Informasi (SeTISI),* FTI Universitas Marantha Bandung, 9 April 2015.

[9] Morandini, M. Penserini, L. Perini, A. Marchetto. 2015. Engineering requirements for adaptive systems. *Journal of Requirements Engineering*. pp. 1-27. Springer.

[10] Dalpiaz, F. Giorgini, P. Mylopoulos, J. 2013. Adaptive socio-technical systems: a requirements-based approach. *Journal of Requirements Engineering.* 18(1):1–24. Springer.

[11] Qureshi, N. A. Jureta, I. J. Perini, A. 2012. Towards a requirements modeling language for self-adaptive systems. *Lecture Notes in Computer Science*. vol. 7195. pp. 263-279. Springer.

[12] Souza, V.E.S. Lapouchnian, A. Angelopoulos, K. Mylopoulos, J. 2013. Requirements-driven software evolution. *Journal of Computer Science - Research and Development*. 28(4), pp. 311-329, Springer.

[13] Arcaini, P. Riccobene, E. Scandurra, P. 2015. Modeling and analyzing MAPE-K feedback loops for self-adaptation. *International Symposium on SE for Adaptive and Self-Managing Systems*. IEEE.

[14] Fernandez, J. C. M. Tamura, G. Mazo, R. Salinesi, C. 2015. Towards a requirements specification multi-view framework for self-adaptive systems. *CLEI Electronic Journal*. Vol. 18, No. 2.