ISSN : 2407 - 3911

# PEMODELAN REQUIREMENTS DALAM MENGKONSTRUKSI PERANGKAT LUNAK SELF-ADAPTIVE

**Aradea**
Teknik Informatika Fakultas Teknik
Universitas Siliwangi Tasikmalaya
Jl. Siliwangi no. 24 Tasikmalaya
aradea@unsil.ac.id

**Iping Supriana, Kridanto Surendro**
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha no. 10 Bandung
iping@informatika.org, endro@informatika.org

## Abstrak

Mengkonstruksi perangkat lunak *self-adaptive* sangat berbeda dengan mengkonstruksi perangkat lunak non s*elf-adaptive*, hal ini menuntut banyak cara yang harus ditempuh untuk mencapai tujuan tersebut. Salah satunya adalah pada tahapan pemodelan *requirements*. Pendekatan yang digunakan saat melakukan pemodelan *requirements* untuk perangkat lunak *self-adaptive*, tidak cukup hanya menangkap kebutuhan sesuai dengan kondisi *systems-as-is*. Namun kebutuhan *systems-to-be* yang berhubungan dengan spesifikasi perilaku, dan kemampuannya untuk menangani perubahan ketika sistem sedang berjalan, merupakan faktor penting yang harus terpenuhi. Makalah ini membahas pemodelan *requirements* untuk mengembangkan *self-adaptive systems*, dengan mengintegrasikan pendekatan *goal oriented requirements engineering* dan *feedback loop*. Diawali dengan latar belakang, kemudian menguraikan penelitian terkait, dilanjutkan dengan konsep yang diusulkan beserta contoh penerapannya, dan diakhir bahasan kami menguraikan pekerjaan untuk masa depan serta kesimpulan.

Kata kunci:

*Requirements modeling, goal oriented requirements engineering, self-adaptive systems, feedback loop*

## *Abstract*

*Construction of self-adaptive software is very different with the construction of non-self-adaptive software, its require many ways that must be through to gain these goals. one of them is on the requirement of modelling phase. The approach that used, when conduct modelling requirement is not enough to catch the needs appropriate with as-is system condition, but the requirement of to-be systems that connected with behaviour specification and its ability to handle transformation when system running is an important factor that must be fulfilled. this paper describes requirement modelling to develop self-adaptive systems, with goal oriented engineering integration approach and loop feedback. Started with the background, then untangle related research, continued with proposed concept and its implementation example, and in the last description, we untangle conclusion and our future works*.

*Keywords*:

*Requirements modeling, goal oriented requirements engineering, self-adaptive systems, feedback loop*

## I. PENDAHULUAN

Suatu kondisi yang alamiah, bahwa lingkungan sistem akan berubah sesuai siklus hidupnya. Saat ini, terdapat beragam faktor yang dapat menjadi penyebab perubahan sistem, yang berhubungan dengan kompleksitas dan ketidakpastian lingkungan sistem, diantaranya yaitu disebabkan oleh keberagaman unsur yang terlibat, pertumbuhan organisasi, perkembangan dan *ubiquitous* perangkat keras, serta faktor-faktor lainnya yang berhubungan dengan globalisasi saat ini. Faktor-faktor ini menjadi tantangan sekaligus persoalan dalam aktivitas pengembangan sistem. Dalam laporan Standish Group International (2013), menunjukan bahwa tantangan pengembangan proyek sistem perangkat lunak terus mengalami peningkatan dari tahun 2010 sampai dengan tahun 2012, yaitu mencapai angka 42% sampai 43%, walaupun tingkat keberhasilan (dari 37% menjadi 39%) dan kegagalan (dari 21% menjadi 18%) relatif membaik. Berdasarkan data tersebut, maka presentasi tantangan 43% menyisakan pertanyaan besar dan memerlukan jawaban sebagai bentuk solusi penyelesaiannya yang berhubungan dengan keterlambatan, melebihi anggaran, serta

kurangnya fitur dan fungsi yang diperlukan. Dalam laporan tersebut dikatakan, bahwa pekerjaan requirements dalam menangkap, memilih, dan mengimplementasikan suatu pengembangan custom applications adalah aktivitas yang paling sulit.

Selain itu, kompleksitas sistem terkait infrastruktur, pengembangan dan manajemen operasi terus mengalami peningatan, sehingga biaya perawatan sistem yang melingkupi biaya tenaga kerja dan administrasi terus bekembang dari 60% menjadi 80% dari total biaya kepemilikan sistem (Sherry dkk., 2012). Serta kompleksitas dari sistem perangkat lunak sangat sulit untuk dikonfigurasi dan dikelola, biaya terkait pemecahan persoalan kesalahan konfigurasi sistem merupakan laporan akuntansi yang substansial mencapai sekitar 17% dari total biaya kepemilikan (Attariyan dkk., 2010). Dengan demikian, persoalan perawatan sistem terkait konfigurasi dan rekonfigurasi ulang sistem telah menjadi tantangan tersendiri, yang membutuhkan penyelesaian. Mengutip dari contoh lainnya (Chelf dkk., 2008), studi yang telah dilakukan National Institute of Standards and Technology (NIST) menunjukan bahwa sekitar US$ 22.2 bilion dapat diselamatkan oleh peningkatan verifikasi dan validasi yang memungkinkan dilakukan lebih awal, serta melakukan identifikasi dan penghapusan cacat suatu sistem perangkat lunak.

Ilustrasi dari tantangan-tantangan tersebut menuntut bahwa sistem harus memiliki kemampuan beroperasi pada suatu lingkungan yang dinamis, jika sistem tersebut ingin bertahan hidup dan tidak membutuhkan banyak biaya yang harus dikeluarkan. Dengan demikian, pengembangan sistem dapat diawali melalui penetapan suatu mekanisme *requirements* yang dapat menangkap dan memformulasikan kebutuhan sistem pada saat sistem tersebut dirancang (*design-time*), namun dapat mengakomodasi kebutuhan pada saat sistem tersebut berjalan (*run-time*). Oleh karena itu, suatu perubahan sistem harus dikendalikan secara menyeluruh, pengendalian perubahan tidak cukup hanya memikirkan kebutuhan operasional saja, kebutuhan suatu evaluasi terutama yang berhubungan dengan perencanaan untuk pengendalian pertumbuhan sistem (Aradea dkk. 2014), merupakan kondisi yang harus dipersiapkan dalam mewujudkan suatu sistem yang dapat memiliki kemampuan beradaptasi secara mandiri, atau dikenal dengan istilah *self-adaptive systems* (SAS). Dengan demikian, penting menetapkan suatu perspektif yang dapat memandu dalam memahami suatu domain dan mengidentifikasi kemungkinan perubahannya (Aradea dkk. 2015).

## II. PENELITIAN TERKAIT

Salah satu tantangan yang memerlukan jawaban dalam suatu penelitian SAS, adalah terkait mekanisme *requirements* yang dilakukan pada saat *design-time*, namun dapat merencanakan dan memfungsikan sistem untuk melakukan adaptasi dalam menangani perubahan-perubahan ketika sistem tersebut sedang berjalan atau *run-time*. Berdasarkan hasil kajian terhadap beberapa *survey paper* dan literatur terkait, penelitian pada area *requirements engineering* telah banyak mengalami keberhasilan dan telah banyak diakui peran utamanya, terutama melalui pendekatan berbasis *goal* kedalam proses *requirements* tersebut. Atas fakta itulah, pada umumnya penelitian yang dilakukan didasarkan pada *state-of-the-art* dalam *goal-oriented requiements engineering* (GORE). Pertimbangan model goal yang dapat merepresentasikan *system-to-be* seperti model i* (Yu, E. dkk, 1995) dan KAOS (Lamsweerde dkk., 1991; Dardenne dkk., 1993), ketika digunakan sebagai konsep pemodelan primitif untuk merepresentasikan apa yang harus dilakukan sistem dan bagaimana sistem harus berprilaku, dihadapkan pada tantangan tersendiri ketika sistem harus beradaptasi pada saat *run-time*.

Para peneliti saat ini mencoba menjawab tantangan tersebut dengan memperluas *state-of-the-art* GORE, dengan memasukan unsur-unsur tambahan yang dapat memenuhi kebutuhan tersebut. Diantaranya pendekatan *goal* ini diterapkan untuk menangani ketidakpastian sistem dengan pendekatan *model-driven* LoREM (Goldsby dkk, 2008), atau melalui prinsip-prinsip konsep *fuzzy logic* seperti yang digunakan dalam model FLAGS (Baresi dkk., 2010), ADS-i* (Serrano dkk. 2011), atau pendekatan lain yang berbeda dengan GORE adalah penggunaan bahasa alami yang dikombinasikan dengan *fuzzy logic* RELAX (Whittle dkk., 2010). Selain itu, dalam GORE ini pendekatan berbasis i*/ agen telah banyak diadopsi oleh para peneliti, dengan mengeksploitasi kelebihannya untuk menangkap variablilitas konteks dan perilaku sistem, mengembangakan perilaku *self-organizing/ self-configure*, serta pemetaannya terhadap setiap tingkatan arsitektur sistem, seperti yang diterapkan dalam model Tropos4AS (Morandini dkk., 2011), CARE (Qureshi dkk., 2012), SOTA (Dhaminda dkk., 2014), GASD (Wang dkk., 2012), termasuk LoREM, FLAGS, dan ADS-i*. Sementara

peneliti lainnya, mengadopsi pendekatan berbasis *goal* ini melalui penerapan prinsip-prinsip teori kontrol, seperti yang dilakukan pada model GOCC (Nakagawa dkk., 2011), DYNAMICO (Norha, Gabriel dkk., 2013), dan ZANSHIN (Souza, Angelopoulos, dkk., 2012), dengan penekanan pada fungsi generik *feedback loops* dalam penalaran *requirements* saat *run-time*.

Berbagai variasi pendekatan para peneliti yang dikombinasikan dengan pendekatan *goal* tersebut, selain memiliki kelebihan masing-masing tentu memiliki kekurangan, misalnya pendekatan goal yang dikombinasikan dengan *fuzzy logic* dan atau bahasa alami memerlukan konstruksi pendekatan yang cukup kompleks yang sangat bergantung pada pendekatan tersebut, dan memerlukan spesifikasi requirements yang sangat lengkap. Selain itu, pendekatan *goal* dengan prinsip kemampuan agen dan atau *model-driven* pada dasarnya kurang dapat secara langsung melakukan pemetaan konsepnya untuk kebutuhan *run-time*. Sementara penerapan teori kontrol dalam pendekatan *goal* sangat memusatkan pada kendali *feedback loop*, dimana untuk memenuhi kebutuhan variabilitas dan sentralisasi suatu sistem independen yang dapat berkolaborasi untuk adaptasi, atau memenuhi partisipasi aktor manusia dan organisasi dalam mencapai kepuasan requirements, membutuhkan rancangan arsitektur khusus, misalnya beberapa jenis sensor tambahan, dan lain-lain, sehingga memerlukan penelitian lebih lanjut untuk menganalisis hubungan dari kebutuhan ini.

Berdasarkan uraian tersebut, persoalan yang tersisa dari para peneliti sebelumnya dan memerlukan perbaikan sebagai bentuk upaya peningkatan dari yang sudah ada, adalah memformulasikan mekanisme proses generik adaptasi dengan memperhatikan variablitas sistem dan sentralisasi pengguna atau faktor manusia sebagai konteks masukan yang dapat terlibat untuk menentukan perilaku adaptasi sistem. Karena salah satu penyebab kegagalan proyek pengembangan sistem adalah kegagalan memenuhi kebutuhan pengguna, kegagalan memenuhi kebutuhan pengguna ini meliputi kegagalan korespondensi yang disebabkan rancangan sistem tidak mencapai tujuan yang diharapkan pengguna, kegagalan interaksi yang tidak mudah digunakan, dan kegagalan ekspektasi (Xiang, 2012). Faktor penyebab kegagalan tersebut, dapat dikategorikan sebagai kegagalan requirements yang dilakukan saat *design-time*, sehingga tidak dapat memenuhi kebutuhan adaptasi pada saat *run-time*.

Dengan demikian timbul suatu pemikiran yang terinspirasi dari penelitian sebelumnya, yaitu memanfaatkan kemampuan agen dalam membentuk suatu pengetahuan untuk merepresentasikan variablilitas (konteks dan perilaku sistem), dan memadukannya dengan prinsip-prinsip teori kontrol untuk menyediakan suatu kerangka formal yang dapat mengimplementasikan fungsi generik *feedback loop* berdasarkan model *requirements* untuk kebutuhan adaptasi saat *run-time*.

## III. KONSEPSI ADAPTATION REQUIREMENT

Tujuan dari pembahasan makalah ini adalah menghasilkan model untuk pengembangan SAS berdasarkan *requirements* yang mendefinisikan pengetahuan saat *design-time* dan dapat memandu kemampuan adaptasi saat *run-time*, dengan mengakomodasi keberagaman unsur sistem dan jaminan keberlangsungannya. Dalam rangka memenuhi tujuan tersebut, untuk menangkap keberagaman unsur sistem yang terlibat diadopsi pendekatan *goal-oriented requirements engineering* (GORE) yang dikonfigurasikan sebagai model berbasis agen. Sementara untuk mendefinisikan kriteria jaminan bahwa *requirements* saat *design-time* dapat memenuhi kebutuhan adaptasi saat *run-time*, teori kontrol melalui *feedback loop* dijadikan sebagai prinsip-prinsip untuk mendefinisikan *control objective*, mekanisme adaptasi, serta kebutuhan monitoringnya.

### III.1 Model Goal – TROPOS

Pemodelan goal yang diadopsi dalam makalah ini, menggunakan model Tropos (Bresciani, 2004). Pendekatan ini dipilih karena memiliki mekanisme *requirements* yang lebih lengkap dari pada pendekatan yang lainnya, selain itu pendekatan ini memiliki rancangan pemetaan model *goal* secara otomatis terhadap arsitektur sistem. Pendekatan ini mengadopsi bahasa pemodelan dan teknik analisis model i*, yang dikembangkan menjadi metodologi pengembangan perangkat lunak berorientasi agen.

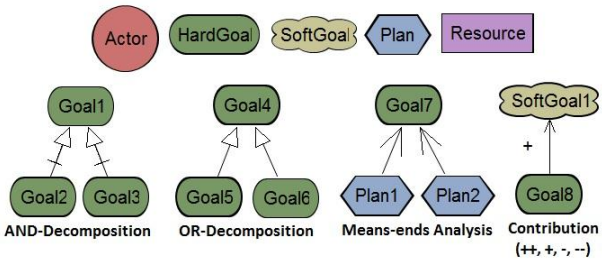
Berdasarkan hasil kajian para peneliti, misalnya (Perini dkk, 2004) dan (Giorini, 2010), bila dilakukan perbandingan dengan model sejenis lainnya, model Tropos ini memiliki cakupan yang lebih luas dalam suatu fase pengembangan sistem, terutama pada fase requirements. Hal senada juga diungkapkan Renata (2006) dalam disertasi doktor dan publikasinya. Merujuk dari referensi tersebut, kami memetakan perbandingan bahasa pemodelan dalam setiap fase

pengembangan sistem, seperti dapat dilihat pada Tabel 1.

**Tabel 1. Perbandingan Bahasa Pemodelan**

| Aktivitas Pengembangan Sistem | | | | |
|---|---|---|---|---|
| ***Model*** | ***Early Requirement*** | ***Late Requirement*** | ***Architecture Design*** | ***Detaild Design*** |
| KAOS | - | √ | √ | - |
| i* | √ | √ | - | - |
| Gaia | - | √ | √ | √ |
| MASE | - | √ | √ | √ |
| Roadmap | - | √ | √ | √ |
| OperA | - | √ | √ | - |
| Tropos | √ | √ | √ | √ |
| Prometheus | - | √ | √ | √ |
| AUML | - | - | - | √ |
| Message/ UML | - | √ | √ | √ |
| AORML | - | √ | √ | √ |
| MAS-Common KADS | - | √ | √ | √ |

Dalam Tabel 1 terlihat bahwa cakupan model Tropos melingkupi seluruh tahapan pengembangan sistem. Demikian juga dalam fase *requirements*, memiliki aktivitas yang lebih rinci yaitu *early requirements* dan *late rqruirements*, sama seperti model i*. Konsep notasi dan relasi dalam bahasa pemodelan Tropos direpresentasikan secara grafis seperti ditunjukan pada Gambar 1.

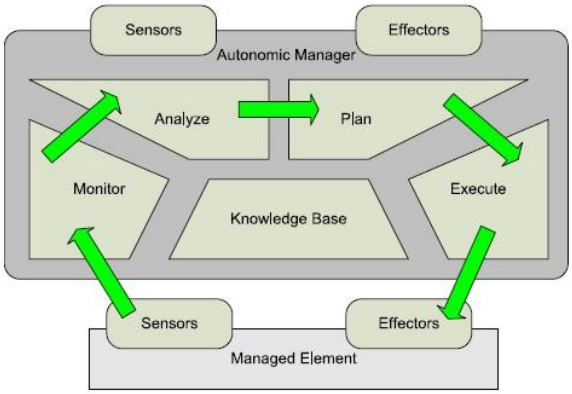



**Gambar 1. Notasi grafis dan relasi model Tropos.**

Tahapan *requirements* dalam Tropos terdiri dari (a) *early requirements*, pandangan untuk mengidentifikasi kebutuhan *stakeholder* yang terlibat dalam *domain*, *goal* mereka dan bagaimana hubungannya satu sama lain, serta apakah terdapat *strategic dependency* antar aktor untuk pencapaian *goal*, (b) *late requirements*, menangkap perubahan *domain* yang disebabkan oleh kebutuhan *system to-be* dan sifat sebenarnya dari sistem, dimulai dengan delegasi *goal* yaitu memperkenalkan aktor dan *goal* baru kedalam model *domain* yang merepresentasikan *system-to-be*.

### III.2 Feedback Loop – MAPE-K

Mekanisme utama dalam adaptasi sistem adalah *monitors*, *analyzers*, *planners*, *executors*, dan *knowledge* (MAPE-K) (Kephart, 2003)(IBM), seperti ditunjukan pada Gambar 2. MAPE-K ini merupakan abstraksi *feedback loop*, dimana perilaku dinamis dari pengelolaan sistem dikendalikan menggunakan *autonomic manager*. Setiap fase dalam MAPE-K dapat dideskripsikan sebagai berikut (a) *Monitor*, mengumpulkan dan pre-process informasi konteks yang relevan dari entitas didalam lingkungan eksekusi yang dapat mempengaruhi sifat yang diinginkan dari sistem target, (b) *Analyze*, mendukung pengambilan keputusan tentang perlunya *self-adaptation*, berdasarkan metode ECA yaitu kondisi *as-is* atau kegagalan *requirements* (*event*), perubahan kondisi (*condition*), tindakan adaptasi (*action*), (c) *Plan*, menghasilkan tindakan yang sesuai untuk mempengaruhi sistem target, sesuai dengan dukungan mekanisme adaptasi dan hasil *analyzer* tersebut, (d) *Execute*, mengimplementasikan tindakan dengan tujuan untuk mengadaptasi sistem target, (e) *Knowledge*, memungkinkan untuk berbagi data, presistensi data, pengambilan keputusan, dan komunikasi antar komponen *feedback loop*, serta, menyusun *multiple feedback loop*.




**Gambar 2. Proses MAPE-K (Kephart; IBM)**

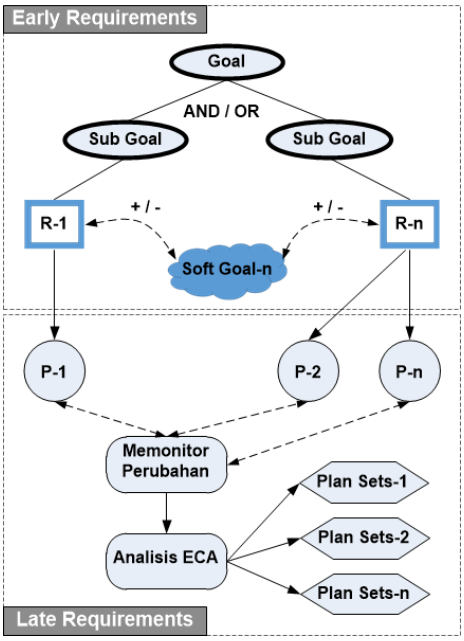
### III.3 Usulan Konsep Self-Adaptation Requirements

Kedua tahapan *requirements* Tropos dapat menangkap serta merepresntasikan variabilitas dalam

konteks dan perilaku lingkungan sistem, namun lebih terfokus pada niat *stakeholder* dan delegasi *goal*, kebutuhan penanganan perubahan ditangkap berdasarkan representasi *system-to-be* melalui penerapan tipe *goal* untuk model penanganan kegagalan, serta pertimbangan kontribusi positif dan negatif terhadap *softgoal*. Sementara kebutuhan monitoring variabel dari dekomposisi *goal* atas perubahan saat *run-time*, dan manajemen perubahan sebagai mekanisme adaptasi belum terakomodasi pada tahapan *requirements* ini. Disini kami mencoba melengkapi kedua fase *requirements* tersebut, untuk meningkatkan kemampuan proses adaptasi melalui *feedback loop* secara otomatis. Menurut (Zhuo-qun, 2012) dalam pemodelan i*, variabel dapat berasal dari *actor's inner task set*, *task* merupakan entitas yang dapat dideteksi dan merupakan sumber dari jenis parameter, jika satu *task* memiliki ketidakpastian, maka semua *task* yang memiliki hubungan dependensi dengan *task* tersebut perlu dimonitor, dan ketika menentukan apa yang dimonitor, nilai-nilai dari parameter dapat digunakan untuk merepresentasikannya.

Berdasarkan pendapat tersebut, jika teori ini diterapkan pada model Tropos maka *task* ini setara dengan *plan*, dimana setiap *plan* dapat memiliki hubungan dependensi dengan *goal*, *resource*, ataupun notasi lainnya. Sementara untuk monitoring terhadap variabel dan kebutuhan penanganan adaptasinya, sistem dinamis melalui *feedback loop* dapat dikembangkan sebagai kriteria jaminan dari mekanisme proses yang akan dibentuk. Ilustrasi konsep pemodelan *requirements* yang diusulkan ini seperti dapat dilihat pada Gambar 3.

*Early requirements* selain difungsikan mengidentifikasi kebutuhan *stakeholder* dalam model domain dan *strategic dependency* nya, fase ini juga difungsikan untuk menangkap kebutuhan monitoring dari setiap entitas yang memungkinkan terjadinya perubahan, beserta entitas terkait lainnya yang memiliki hubungan dependensi. Sehingga pemodelan *goal* pada tahapan ini diharapkan dapat merepresentasikan kebutuhan awal untuk menentukan mekanisme adaptasi. Diawali dengan melakukan dekomposisi AND/ OR terhadap *goal* menjadi *sub goal*, mengidentifikasi *requirements* (R-1, R-2, R-n) dari *goal/ sub goal* yang memungkinkan atau dapat melakukan perubahan yang berpengaruh pada parameter *goal*, serta memiliki kontribusi positif atau negatif (+ / -) terhadap satu atau lebih *soft goal* sebagai representasi kebutuhan *non-fungsional*, diakhiri dengan mendefinisikan variabel dan parameter dari setiap *goal/ sub goal* tersebut (P1, P-2, P-n).




**Gambar 3. Kerangka dasar adaptation requirements**

*Late requirements*, pada fase ini delegasi *goal* selain menangkap perubahan domain yang merepresentasikan *system-to-be*, diarahkan juga untuk menentukan variabilitas dan menganalisis kebutuhan adaptasi. Dimulai dengan menata kembali dependesi *goal*, kemudian memonitor perubahan, yaitu mengidentifikasi perubahan yang berpengaruh terhadap parameter setiap *goal/ sub goal* (P-1, P-2, P-n), selanjutnya melakukan analisis dengan metode *event-condition-action* (ECA), yaitu menganalisisi parameter dan perilaku perubahan, serta menentukan variasi perubahan berdasarkan penetapan aturan dan *control objective*, diakhiri dengan menetapkan *plan* (Plan Sets-1, Plan Sets-2, Plan Sets-n) sebagai mekanisme adaptasi.

### III.4 Ilustrasi Kasus

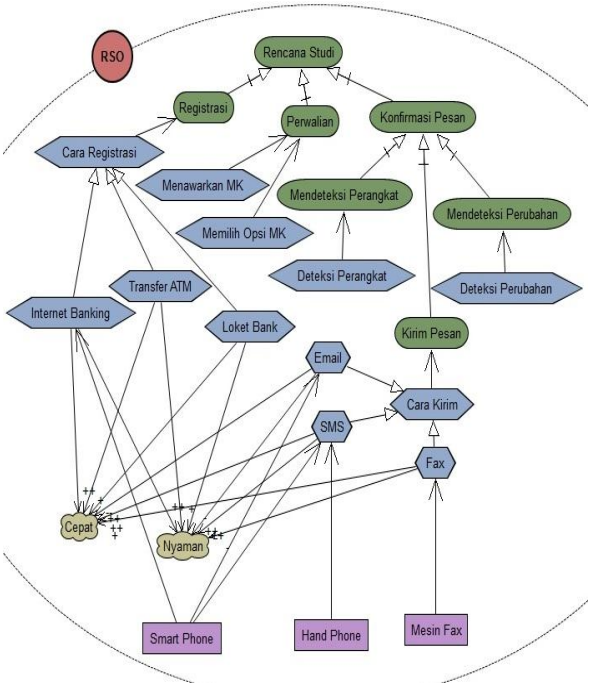
Berikut ini merupakan salah satu contoh ilustrasi kasus sederhana sistem di perguruan tinggi, berupa aplikasi rencana studi *on-line* untuk mahasiswa. Kasus merujuk dari studi kasus yang dibahas dalam penelitian Qureshi dkk., (2012), yang telah dimodifikasi dan disesuaikan dengan kebutuhan sistem di perguruan tinggi. Sistem ini dimodelkan dengan menggunakan bahasa pemodelan Tropos,

dimana kebutuhan dari sifat *self-adaptation* akan dianalisis. Tropos difungsikan untuk melakukan pemodelan *requirements*, serta mendefinisikan persoalan adaptasi. Sementara penyelesaian dari kebutuhan adaptasi tersebut, kita gunakan pendekatan *feedback loop* yang diintegrasikan kedalam model Tropos, melalui MAPE-K dengan menerapkan ECA *rules*.

Aplikasi rencana studi *on-line* menyediakan kebutuhan mahasiswa untuk melakukan registrasi dan perwalian pada setiap awal semester baru. Aplikasi ini mendukung mahasiswa dalam melakukan registrasi pembayaran, pengambilan mata kuliah dan penerimaan informasi sebagai konfirmasi pesan keberhasilan proses rencana studi, sehingga mahasiswa tersebut memiliki status aktif untuk semester baru tersebut. Pesan informasi harus dikirim kepada mahasiswa melalui beberapa alternatif pilihan media, misalnya melalui sms, email, yang merupakan cara yang paling disukai, serta fax, dan harus diterima misalnya dalam waktu ≤ 1 jam atau maksimum ≤ 24 jam.

Selain itu, proses pengiriman pesan ini juga mempertimbangkan kebergantungan pada pemilihan format dari pesan yang sesuai dengan perangkat yang akan digunakan, misalnya melalui format html atau teks. Dalam aplikasi ini juga harus dipastikan, bahwa pengambilan mata kuliah dapat dilakukan setelah mahasiswa melakukan pembayaran, dan pengiriman pesan dapat dilakukan setelah rencana studinya disetujui oleh dosen wali. Persoalan yang dapat muncul disini adalah ketika mahasiswa tidak dapat memperoleh pesan tersebut, dan sistem harus mengirimkan pesan misalnya kepada bagian tata usaha akademik, sebagai alternatif penerima pesan. Rincian dari pemodelan sistem ini seperti disajikan pada Gambar 4.

Pada ilustrasi model yang disajikan pada Gambar 4, *goal* "rencana studi *on-line*" didekomposisi menjdai tiga *sub goal*, yaitu "registrasi", "perwalian", dan "konfirmasi pesan". Sekarang kita coba fokus pada pembahasan *plan* "cara kirim" pesan yang merupakan *plan* untuk mencapai *goal* "kirim pesan" rencana studi, *plan* ini diwujudkan dengan *OR-Decomposition* beberapa *plan*, yang dapat berkontribusi positif atau negatif terhadap *soft goal* "cepat" dan "nyaman", dengan menggunakan beberapa sumber daya misalnya "*hand phone*", "*smart phone*", "mesin *fax*". Tujuan dari *plan* ini adalah mewujudkan perilaku alternatif untuk mengatasi variabilitas dalam konteks operasional yang dapat memenuhi *high-level goal* dan secara optimal memenuhi *soft goal*. Misalnya pemilihan *plan* kirim "*Email",* harus dilakukan ketika mahasiswa menggunakan *smart phone* atau *laptop*, karena memberikan kontribusi positif yang penuh (++) terhadap *soft goal* "*nyaman*", dibandingkan dengan kirim "*Fax"* yang berkontribusi negatif (-).




**Gambar 4. Pemodelan sistem rencana studi**

Berdasarkan pemodelan tersebut, kita dapat menangkap persoalan adaptasi yang harus diselesaikan. Dalam hal ini, *goal* "konfirmasi pesan" rencana studi merupakan proses yang harus dimonitor, karena memiliki variabilitas dalam pencapaiannya melalui *sub goal* "kirim pesan", yaitu *OR-decomposition* dari *plan* "cara kirim". Oleh karena itu berdasarkan tahapan *requirements* pada Gambar 3, dekomposisi dari *goal* "konfirmasi pesan" dapat kita tetapkan dua *sub goal* tambahan sebagai kebutuhan adaptasi, yaitu goal "mendeteksi perangkat" dan "mendeteksi perubahan". Kedua *sub goal* tersebut merupakan *AND-Decomposition*, serta *plan* nya masing-masing, artinya variabel dalam setiap *plan* yang ada memiliki hubungan dependensi satu sama lain dan harus dimonitor, hal ini dapat direpresentasikan dengan menggunakan nilai parameternya masing-masing.

Dengan demikian dapat kita tetapkan bahwa *plan* "cara kirim" akan dipengaruhi oleh *plan* "deteksi perangkat" yang berhubungan dengan *resource* jenis *device* (dengan asumsi *device* yang diidentifikasi adalah *hand phone* dan *smart phone*), dan *plan* "deteksi perubahan" yang berhubungan dengan kejadian yang tidak terduga atau kesalahan misalnya kesalahan pengiriman pesan (*msg_error*) atau hilang koneksi (*conn_error*). Jadi kita dapat memperoleh variabel yang direpresentasikan sebagai "*phone_type*" dan "*event_error*", sehingga *plan* dapat disajikan sebagai:

*Plan (phone_type, event_error)*

Sementara untuk kebutuhan proses analisis, kita harus menetapkan nilai dari setiap parameter, berdasarkan informasi sebelumnya, maka dapat ditetapkan *rules* sebagai berikut :

**Rule 1 :**

*if phone_type = hand_phone or smart_phone and event_error = null,*
*then plan = send sms.*

**Rule 2 :**

*if phone_type = smart_phone and event_error = null,*
*then plan = send email.*

**Rule 3 :**

*if phone_type = null and event_error = null,*
*then plan = send fax.*

**Rule 4 :**

*if not [criteria],*
*then alternative plan = send fax or email to lookup contact.*

**Rule 5 :**

*if event_error = not null,*
*then plan = send notify user.*

Berdasarkan aturan tersebut, maka dapat kita petakan kedalam tabel ECA (pada Tabel 2). "*Event"* mengacu keadaan sistem rencana studi saat ini, "*condition"* mengacu ketika terjadi perubahan kondisi sistem, "*action"* dalam kondisi tertentu, apa yang bisa dilakukan untuk menyesuaikan diri terhadap perubahan lingkungan, sehingga diperoleh tiga aksi sebagai solusi alternatif bagi kebutuhan adaptasi. Dalam menyusun suatu perencanaan secara menyeluruh, penetapan parameter lainnya yang dapat mendukung, serta formulasi mekanisme penalaran dapat dikembangkan lebih lanjut.

**Tabel 2. ECA Pengiriman Pesan**

| Pengiriman Pesan | | |
|---|---|---|
| ***Event*** | ***Condition*** | ***Action*** |
| $P_1$ *(phone_type = hand_phone or smart_phone)*<br>$P_2$ *(phone_type = smart_phone)*<br>$P_3$ *(phone_type = null)* | *event_error = null* | $P_1$ *= send sms*<br>$P_2$ *= send email*<br>$P_3$ *= send fax* |
| $P_1$ *(phone_type = hand_phone or smart_phone)*<br>$P_2$ *(phone_type = smart_phone)*<br>$P_3$ *(phone_type = null)* | *not [criteria] (msg_error)* | $P_{Alt}$ *= send email or fax to lookup contact* |
| $P_1$ *(phone_type = hand_phone or smart_phone)*<br>$P_2$ *(phone_type = smart_phone)*<br>$P_3$ *(phone_type = null)* | *event_error = not null* | $P_{err}$ *= send notify user* |

```
<beliefs>
    <!-- The belief contains the tropos hierarchy as facts. -->
    <beliefset name="goals" class="TGoal">
        <fact>Components.createGoal("Mendeteksi_Perangkat","ME")</fact>
        <fact>Components.createGoal("Perwalian","ME")</fact>
        <fact>Components.createGoal("Kirim_Pesan","ME")</fact>
        <fact>Components.createGoal("Registrasi","ME")</fact>
        <fact>Components.createGoal("Konfirmasi_Pesan","AND")</fact>
        <fact>Components.createGoal("Rencana_Studi","AND")</fact>
        <fact>Components.createGoal("Mendeteksi_Perubahan","ME")</fact>
    </beliefset>
    <beliefset name="softgoals" class="TSoftgoal">
    </beliefset>
    <beliefset name="decomp" class="TLink">
        <fact>new TLink("Konfirmasi_Pesan","Mendeteksi_Perangkat",1)</fact>
        <fact>new TLink("Konfirmasi_Pesan","Mendeteksi_Perubahan",1)</fact>
        <fact>new TLink("Konfirmasi_Pesan","Kirim_Pesan",1)</fact>
        <fact>new TLink("Rencana_Studi","Registrasi",1)</fact>
        <fact>new TLink("Rencana_Studi","Perwalian",1)</fact>
        <fact>new TLink("Rencana_Studi","Konfirmasi_Pesan",1)</fact>
    </beliefset>
    <beliefset name="meansend" class="TLink">
        <fact>new TLink("Mendeteksi_Perangkat","Deteksi_Perangkat")</fact>
        <fact>new TLink("Registrasi","Cara_Registrasi")</fact>
        <fact>new TLink("Perwalian","Memilih_Opsi_MK")</fact>
        <fact>new TLink("Kirim_Pesan","Cara_Kirim")</fact>
        <fact>new TLink("Perwalian","Menawarkan_MK")</fact>
        <fact>new TLink("Mendeteksi_Perubahan","Deteksi_Perubahan")</fact>
    </beliefset>
    <beliefset name="contributions" class="TContrib">
    </beliefset>
    <beliefset name="dependencies" class="TDependency">
```

**Gambar 5. Kode definisi goal dalam belief base**

Dalam mengembangkan ilustrasi kasus ini kami menggunakan alat bantu TAOM4E, *tools* ini memberikan beberapa fasilitas yang telah disediakan dalam model editornya, misalnya t2x (tropos *to* jadex) *code generation tool*, yaitu memetakan model *goal* yang diimplementasikan kedalam *Jadex BDI agent platform*, secara eksplisit dapat memelihara model *goal* saat *run-time* dan menyediakan *middleware* yang tepat untuk menavigasi model serta bertindak sesuai dengan kebutuhan. Kode agen dapat dihasilkan dari antarmuka grafis, dan implementasi prototipe dieksekusi langsung dari *Eclipse user interface*, selain itu tersedia juga kebutuhan untuk melakukan pemodelan lingkungan dan kondisi. Beberapa tampilan yang dihasilkan dari hasil

pemetaan model *goal* terhadap kode dapat dilihat pada Gambar 5 dan 6.

```
<!-- The meta-level goals for choosing between plans (and goals). -->
<metagoal name="meta_Mendeteksi_Perangkat">
    <parameterset name="applicables" class="ICandidateInfo"/>
    <parameterset name="result" class="ICandidateInfo" direction="out"/>
    <trigger>
        <goal ref="Mendeteksi_Perangkat"/>
    </trigger>
</metagoal>
<metagoal name="meta_Perwalian">
    <parameterset name="applicables" class="ICandidateInfo"/>
    <parameterset name="result" class="ICandidateInfo" direction="out"/>
    <trigger>
        <goal ref="Perwalian"/>
    </trigger>
</metagoal>
<metagoal name="meta_Kirim_Pesan">
    <parameterset name="applicables" class="ICandidateInfo"/>
    <parameterset name="result" class="ICandidateInfo" direction="out"/>
    <trigger>
        <goal ref="Kirim_Pesan"/>
    </trigger>
</metagoal>
<metagoal name="meta_Registrasi">
    <parameterset name="applicables" class="ICandidateInfo"/>
    <parameterset name="result" class="ICandidateInfo" direction="out"/>
    <trigger>
        <goal ref="Registrasi"/>
    </trigger>
</metagoal>
<metagoal name="meta_Mendeteksi_Perubahan">
    <parameterset name="applicables" class="ICandidateInfo"/>
    <parameterset name="result" class="ICandidateInfo" direction="out"/>
```

**Gambar 6. Kode meta-level goal**

## IV. DISKUSI DAN PEKERJAAN MASA DEPAN

Dalam pendekatan Tropos, aktivitas pemodelan dapat memberikan kemampuan variabilitas yang tinggi untuk *requirements* adaptasi. Salah satunya kebutuhan informasi yang mendukung penentuan *plan* untuk memenuhi *goal*. Misalnya dalam contoh ilustrasi kasus, untuk *goal* "konfirmasi pesan" rencana studi, informasi dapat dikumpulkan saat *run-time*, yaitu melalui *sub goal* "deteksi perubahan" untuk memonitor kejadian yang tidak terduga, seperti kesalahan pengiriman pesan atau hilang koneksi, dan memberikan kriteria kepada sistem untuk merubah perilakunya berdasarkan alternatif tertentu, atau misalnya melalui *sub goal* "deteksi perangkat" yang digunakan oleh pengguna. Tropos menangani kondisi ini melalui penerapan tipe *goal*, lingkungan, dan kegagalan *goal*.

*Requirements* ini sebenarnya dapat lebih ditingkatkan jika didasari dengan asumsi domain (misalnya, pengambilan mata kuliah dapat dilakukan setelah pembayaran/ pesan dikirimkan setelah disetujui dosen wali) dan fakta yang dibuat secara eksplisit, hal ini akan memberikan cara yang lebih baik dalam menganalisi variabilitas. Seperti yang telah dilakukan oleh Qureshi dkk. [2012], yaitu mengintegrasikan pemodelan *goal* dengan *ontology*, dimana setiap *plan* untuk mencapai suatu *goal* dapat dilengkapi dengan representasi pengetahuan (*ontology*). Dalam pendekatan ini, adaptasi dilakukan berdasarkan pada *log* yang di-*maintenance* dalam komponen monitor, namun pendekatan ini belum menangani rekonfigurasi untuk perubahan yang dinamis. Selain itu, evaluasi *soft goal* yang dilakukan dalam Tropos terlalu subjektif dan tidak memberikan bukti yang jelas hanya didasari pada pilihan kontribusi *full satisfaction* (++), *partial satisfaction* (+), *full negative* (--), *partial negative* (-), dalam hal ini *quality constraints* tertentu, misal spesifikasi batasan waktu (w) pengiriman pesan, $w \leq 1$ jam atau $w \leq 24$) dapat memberikan batasan sebagai bukti yang lebih pasti.

Sementara itu terkait dengan fokus pekerjaan kami kedepan, adalah bagaimana melakukan perkawinan antara konsep *requirements* berbasis *goal* dengan konsep *self-adaptation* berbasis *feedback loop* atau *system dynamic*, secara lebih rinci dan menyeluruh, karena terdapat beberapa unsur penting lainnya yang harus diteliti, diantaranya (a) mendefinisikan *problem requirements* yang dapat dijadikan rujukan untuk mendefinisikan *operational semantic*, (b). memperluas dan melengkapi bahasa pemodelan *requirements* yang ada, misalnya model baru untuk model *goal*, lingkungan, kegagalan, *domain assumptions*, dll., (c) mengembangkan mekanisme adaptasi dalam dua kategori, yaitu rekonfigurasi dan evolusi perangkat lunak. Sehingga diharapkan dapat memperkaya *state-of-the-art* yang ada, dengan target memiliki nilai tambah dan kebaruan.

## V. KESIMPULAN

Pendekatan *requirements* untuk pengembangan SAS sangat berbeda dengan *requirements* untuk non-SAS. Dimana *requirements* tidak hanya cukup dipenuhi saat *design-time* saja, namun *requirements* perlu dipenuhi juga pada saat *run-time*. Aktivitas dalam melakukan pemodelan *requirements* dapat diawali dengan memahami dimensi perubahan dan dimensi dari pendekatan yang dipilih, dalam makalah ini kami menetapkan pendekatan yang diadopsi adalah pendekatan *goal-based*. Para peneliti saat ini telah mencoba menjawab tantangan pada pendekatan *goal* tersebut dengan memasukan unsur tambahan yang dapat memenuhi kebutuhan saat *run-time*. Misalnya melalui pendekatan *model-driven*, prinsip *fuzzy logic*, bahasa alami, pendekatan *agent-based*, maupun prinsip *feedback loop*.

Berdasarkan hasil kajian para peneliti tersebut, kami menemukan peluang penelitian yang diharapkan dapat memberikan kontribusi lebih dari hasil penelitian yang telah ada, yaitu

mengkombinasikan pendekatan berbasis i*/ Tropos sebagai konsep *requirements* dan *feedback loop* sebagai konsep *self-adaptation*. Integrasi dari kedua pendekatan tersebut dapat saling melengkapi kekurangan dan kelebihannya masing-masing. Oleh karena itu, kami mengusulkan konsep dasar sebagai kerangka awal dan studi pendahuluan, seperti yang telah dibahas pada bagian III. Saat ini kami sedang melakukan evaluasi dan merumuskan primitif sistem secara lebih rinci, termasuk melakukan kajian terhadap mekanisme adaptasi yang akan dikembangkan, yaitu menetapkan cara pengambilan keputusan dalam menentukan solusi alternatif. Kami melihat beberapa peluang melalui berbagai pendekatan, seperti *multi-agent, qualitative reasoning, machine learing, risk mitigation*, dan lain-lain, termasuk keterlibatan pengguna.

## REFERENSI


The Standish Group (2013): Chaos Manifesto 2013, Think Big, Act Small, The Standish Group International, Copyright © 2013.

Sherry, J., Hasan, S., Scott, C., Krishanmurthy, A., Ratnasamay, S., dan Sekar, V. (2012): Making middleboxes someone else's problem, Proceedings of the ACM SIGCOMM 2012, Conference on Application, Technology, Architecture, and Protocols for computer communication, page 13, New York, USA, ACM Press.

Attariyan, M., dan Flinn, J. (2010): Automating configuration troubleshooting with dynamic information flow analysis, Proceedings of the 9th conference on operating systems design and implementation, USENIX Association.

Chelf, B., dan Chou, A., (2008): The business case for static source code analysis. Technical Report, Coverity Inc.

Aradea, Supriana, I., dan Surendro, K., (2014): An overview of multi agent system approach in knowledge management model. International Conference on Information Technology Systems and Innovation, School of Electrical Engineering and Informatics, Institut Teknologi Bandung.

Aradea, Supriana I., dan Surendro K. (2015): Roadmap dan area penelitian self-adaptive systems, Seminar Nasional Teknik Informatika dan Sistem Informasi (SeTISI), Universitas Maranatha Bandung.

Yu, E.S., (1995): Modelling strategic relationships for process engineering, Ph.D. Thesis, University of Trento.

van Lamsweerde, A., Dardenne, A., Delcourt, B., dan Dubisy, F., (1991): The KAOS Project: Knowledge Acquisition in Automated Specification of Software, In Proc. of the AAAI Spring Symposium Series, Stanford University, pages 59-62, AAAI.

Dardenne, A., van Lamsweerde, A., dan Fickas, S., (1993): Goal directed requirements acquisition. In: Selected Papers of the Sixth International Workshop on Software Specification and Design (IWSSD), pp. 3–50.

Goldsby, H., Sawyer, P., Bencomo, N., Cheng, B.H.C., dan Hughes, D., (2008): Goal-based modeling of dynamically adaptive system requirements. in: Proc. ECBS, IEEE. pp. 36–45.

Baresi, L., Pasquale, L., dan Spoletini, P., (2010): Fuzzy goals for requirements-driven adaptation. in: Proc. RE. IEEE, pp. 125–134.

Serrano, M., dan Sampaio, J. C., (2011): Development of agent-driven systems: from i* architectural models to intentional agents' code, CEUR Proceedings of the 5th International i* Workshop (iStar 2011).

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., dan Bruel, J.M., (2010): RELAX: a language to address uncertainty in self-adaptive systems requirement, Requir. Eng. 15 (2), pp. 177–196.

Morandini, M., (2011): Goal-oriented development of self-adaptive systems, Ph.D. Thesis, University of Trento.

Qureshi, N.A., Jureta, I., dan Perini A., (2012): Towards a requirements modeling language for self-adaptive systems, Lecture Notes in Computer Science, in Springer, REFSQ, vol. 7195, pp. 263-279.

Dhaminda B. Abeywickrama, Hoch N., dan Zambonelli, F., (2014): An integrated eclipse plug-in for engineering and

implementing self-adaptive systems, 23rd International WETICE Conference, 978-1-4799-4249-7/14, IEEE.

Wang, T., Li, B., Zhao, L., dan Zhang, X., (2012): A goal-driven self-adaptive software system design framework based on agent. ICAPIE Organization Commite, Published by Elsevier B.V.

Nakagawa, H., Ohsuga, A., dan Honiden, S., (2011): GOCC: A configuration compiler for self-adaptive systems using goal-oriented requirements description, Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pages 40-49, May 21–28, Waikiki, Honolulu, HI, ACM New York, NY, USA.

Norha, M. Villegas, (2013): Context management and self-adaptivity for situation-aware smart software systems, Ph.D. Thesis, University of Victoria.

Silva Souza V.E., (2012): Requirements-based software system adaptation. Ph.D. Thesis, University of Trento.

Xiang, Z., (2012): Problems in information system development, Lahti University of Applied Sciences.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). TROPOS: An agent-oriented software development methodology. Journal of Autonomous Agents and Multi-Agent Systems, 8(3), 203–236..

Perini, A., Giunchigila, F., Mylopoulos, J. (2004). Tropos : An agent-oriented software development methodology, Autonomous Agent and Multi-Agent Systems.

Giorgini, P. (2010). Agent-oriented software engineering, Lecture Notes in Tropos Basics, University of Trento.

Renata G., S., (2006). Agent-oriented constructivist knowledge management", Ph.D. Thesis, University of Twente.

Kephart, J.O., Chess, D.M. (2003). The vision of autonomic computing", IEEE Computer, 36(1), pp. 41–50, 2003.

IBM Corporation: An architectural blueprint for autonomic computing. White Paper, 4th edn., IBM Corporation.

Zhuo-Qun, Y., Zhi, J. (2012). Requirements modeling and system reconfiguration for self-adaptation of internetware", Proceedings of the Fourth Asia-Pacific Symposium on Internetware, ACM New York, NY, USA.