Perspektif Pengembangan Self-Adaptive Systems : Requirements Engineering

Article · January 2016

CITATIONS
0

READS
421

3 authors:

Aradea Dipalokareswara
Siliwangi University
45 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Iping Supriana
Bandung Institute of Technology
289 PUBLICATIONS **456** CITATIONS
SEE PROFILE

Kridanto Surendro
Bandung Institute of Technology
144 PUBLICATIONS **427** CITATIONS
SEE PROFILE

Perspektif Pengembangan Self-Adaptive Systems : Requirements Engineering

[1]Aradea, [2]Iping Supriana Suwardi, [3]Kridanto Surendro
[1]Teknik Informatika Universitas Siliwangi Tasikmalaya
[2 3]Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
aradea@unsil.ac.id; iping@informatika.org; endro@informatika.org;

Abstrak—Kebutuhan pengembangan self-adaptive systems saat ini banyak mendapatkan perhatian, baik dari para praktisi maupun akademisi. Hal ini disebabkan oleh tuntutan dari kompleksitas sistem yang terus berkembang dari waktu ke waktu. Self-adaptive systems merupakan bidang penelitian yang sangat luas, pemahaman yang mendalam terhadap bidang tersebut dapat menjadi modal awal yang strategis untuk mengetahui ketepatan dalam aktivitas pengembangannya. Makalah ini menguraikan perspektif untuk memulai aktivitas pengembangan self-adaptive systems. Diawali dengan pemahaman terkait area-area penelitian dan permasalahan yang menjadi tantangan sekaligus peluang. Kemudian membahas pendekatan-pendekatan yang dapat digunakan, termasuk klasifikasi metode dan alternatif pendekatan yang ada saat ini. Selanjutnya menguraikan mindmap dan perkembangan penelitian terkait. Pada akhir bahasan, kami memetakan kerangka pemikiran awal untuk kebutuhan penelitian pada area *requirement engineering* sebagai studi pendahuluan kami.

Kata Kunci—self-adaptive systems, self-adaptive roadmap; self-adaptive approach; requirements engineering

Abstract—Requirements of self-adaptive systems development has been gained more attention today, from both practitioners and academics. Caused by need of system complexity that continue to evolve over time. Self-adaptive system is wide research area, deep understanding to this field could become an initial capital to determine the accuracy of its development activities. This paper depict the perspective to start activity in self-adaptive systems development. Begin from understanding of related research areas and the problem that could become challenge or chance. Then this paper explain about approaches that can be used, including classification methods and alternative approach that exist at a current time. After that, this paper depict mind map and developments related research. At the end of the explanation, we are mapping initial thought construction for research need in requirement engineering area for self-adaptive systems development as our preliminary study.

Keywords—self-adaptive systems; self-adaptive roadmap; self-adaptive approach; requirements engineering;

I. PENDAHULUAN

Suatu kondisi yang alamiah, bahwa lingkungan suatu sistem akan berubah sesuai siklus hidupnya, oleh karena itu suatu perubahan harus dikendalikan secara menyeluruh. Pengendalian perubahan tidak cukup hanya memikirkan kebutuhan operasional, kebutuhan suatu evaluasi, terutama yang berhubungan dengan perencanaan untuk pengendalian pertumbuhan sistem [1], baik pada saat sistem tersebut sedang dirancang maupun pada saat sistem tersebut sedang berjalan, merupakan kondisi yang harus dipersiapkan dalam mewujudkan suatu kebutuhan *self-adaptive systems* (SAS). Oleh karena itu, penting menetapkan suatu perspektif yang dapat memandu dalam memahami suatu *domain* dan mengidentifikasi kemungkinan perubahan-perubahannya [2]. Faktor ketidakpastian lingkungan, keberagaman unsur yang terlibat, *ubiquitous* perangkat keras, otomatisasi proses, serta faktor-faktor lainnya yang berhubungan dengan globalisasi saat ini, menjadi penyebab sekaligus pendorong terciptanya suatu kebutuhan sistem yang memiliki karakteristik SAS.

Berdasarkan uraian tersebut SAS merupakan spektrum yang sangat luas, oleh karena itu diperlukan suatu pemahaman yang spesifik terhadap *domain* permasalahan untuk mendefinisikan pengembangan SAS yang relevan dengan suatu kebutuhan [3]. Berdasarkan [4], SAS merupakan kemampuan sistem untuk menyesuaikan perilakunya dalam menanggapi lingkungan, kata "*self*" menunjukkan bahwa sistem secara otonom memutuskan (dengan minimal atau tidak ada gangguan) bagaimana beradaptasi atau untuk mengatur dirinya sendiri sehingga dapat mengakomodasi perubahan dalam suatu konteks dan lingkungan, SAS dapat menjalankan fungsinya tanpa campur tangan manusia, berupa bimbingan dalam bentuk *higher-level objectives* (misalnya, melalui kebijakan) yang berguna dan dapat dicapai dalam berbagai sistem.

Selain itu definisi SAS lainnya [4], yaitu merupakan suatu sistem yang dapat mengkonfigurasi dan mengkonfigurasi ulang sendiri, meningkatkan fungsinya, secara terus-menerus mengoptimalkan diri, melindungi diri, dan memulihkan diri sendiri. SAS dapat berhubungan dengan sifat-sifat seperti *self-managed systems*, *autonomic systems*, *self-repair*, *self-configure*, *self-heal*, *self-protect, self-tune* serta *self-**. Dari luasnya spektrum SAS ini, kita dapat mengawali pemahaman melalui pengelompokan area dan permasalahan yang ada (bagian II), pendekatan-pendekatan yang digunakan saat ini (bagian III), serta penelitian terkait dan pemetaan dari *requirements* pengembangan SAS (bagian IV).

II. ROADMAP PENELITIAN SELF-ADAPTIVE SYSTEMS

TABEL I. ROADMAP SAS PERTAMA [4]

Tantangan dan Peluang Penelitian	
Area	***Deskripsi***
Dimensi - *Self-Adaptive Systems*	
Goal	Penyelesaian beberapa *goal* yang berpotensi bertentangan
Perubahan	Monitoring perubahan menghsailkan benefit bagi QoS
Mekanisme	Prediksi pengelolaan yang berhubungan dengan skalabilitas
Efek	Prediksi dampak perilaku sistem dan kehandalan sistem
Requirements - *Self-Adaptive Systems*	
Bahasa	Bahasa *requirement* yang dapat menangani ketidakpastian
Pemetaan Arsitektur	Bahasa *requirement* dapat menentukan suatu model, hingga spesifikasi arsitektur untuk adaptasi saat *runtime*
Run-time	Model sebagai dasar penentu perilaku otonom sistem
Refleksi	Secara dinamis mengamati struktur dan perilakunya sendiri
Pelacakan	Mengatasi ketidak lengkapan informasi lingkungan sistem
Rekayasa - *Self-Adaptive Systems*	
Pemodelan	Menentukan *control loop* dan mengekspos sifat adaptif
Arsitektur	Mengembangkan referensi arsitektur untuk sistem adaptif, interaksi, aliran data, toleransi, kompromi, stabilitas, dll
Disain	Menyusun katalog dan skema *control loop* serta karakteristik elemennya, pola klasifikasi *control loop*, dll
Middleware	Mengembangkan dukungan *middleware* untuk kerangka kerja dari fungsi sistem adaptif
V & V	Pengujian dan evaluasi *control loop*
Rekayasa Ulang	Eksplorasi dan ketepatan pemilihan teknik-teknik untuk pengembangan sistem yang ada
Manusia-Komputer	Mengembangkan pola interaksi antar elemen sistem dan penggunanya
Jaminan - *Self-Adaptive Systems*	
Perubahan Dinamis	Mengidentifikasi secara dinamis perubahan kebutuhan sistem
Model-driven	Model yang dapat melakukan verifikasi dan validasi saat *design-time* dan saat *run-time*
Jaminan *Run-time*	Solusi dan efisiensi algoritma untuk kebutuhan verifikasi saat sistem sedang berjalan
Sosial	Realisasi fungsi adaptif didalam *safty-critical systems*

Komunitas self-adaptive system (SAS) pada Internationales Begegnungsund Forschungszentrum fuer Informatik (IBFI) Schloss Dagstuhl, Germany, telah melakukan seminar dan menetapkan agenda penelitian SAS yang diterbitkan pada Tahun 2009 [4] dan 2013 [5]. Rangkuman dari hasil seminar Tahun 2009 dapat dilihat pada tabel I. Seminar pertama ini, mengklasifikasikan tantangan dan peluang penelitian SAS menjadi 4 area. Sementara hasil seminar yang kedua, mengklasifikasikannya menjadi 4 area juga, seperti dapat dilihat pada tabel II, terdiri dari ruang disain, proses, desentralisasi *control loops*, serta verifikasi dan validasi (V&V) *runtime*.

Seminar kedua ini posisinya bukan untuk menggantikan hasil seminar pertama, melainkan sebagai pelengkap dan melanjutkan dengan topik tambahan yang belum dibahas pada seminar pertama. Apabila hasil seminar kedua ini dipetakan terhadap hasil seminar pertama, maka area ruang disain ini terkait dengan area dimensi pemodelan, area desentralisasi *control loops* terkait dengan area rekayasa, dan area V&V terkait dengan area kriteria jaminan, sementara area proses dikatakan dalam seminar tersebut merupakan topik baru, namun jika ditelaah isinya sebenarnya masih ada keterkaitan dengan *requirements*.

TABEL II. ROADMAP SAS KEDUA [5]

Tantangan dan Peluang Penelitian	
Area	***Deskripsi***
Ruang Disain - *Self-Adaptive Systems*	
Observasi	Pengamatan dan pengukuran yang dapat menyimpulkan keadaan untuk keberhasilan atau kegagalan adaptasi
Representasi	Penyediaan informasi keadaan sistem saat berjalan dan menjadi pemandu bagi target adaptasi dan sifat masalah
Kontrol	Mekanisme *control loops* yang sesuai dengan struktur sistem dan kompleksitas *goal* dari adaptasi
Identifikasi	Mengenali konteks dan memberlakukan instanisasi untuk struktur, perilaku, keadaan, nilai parameter yang relevan
Aktifasi Adaptasi	Mekanisme pemicu/ pembangkit, dukungan dan penangnanan kegagalan dalam adaptasi secara otomatis
Proses - *Self-Adaptive Systems*	
Pemahaman Proses	Menentukan sebuah *library* berisi definisi elemen proses dan dapat digunakan kembali dalam situasi ketidakpastian
Pemodelan Proses	Bahasa pemodelan untuk menspesifikasikan proses yang dapat berkembang pada saat/ kebutuhan adaptasi *runtime*
Perancangan Proses	Dukungan untuk penalaran, analisis dan *tuning* dari spesifikasi proses sesuai dengan *goal* dan lingkungan
Desentralisasi Control Loops - *Self-Adaptive Systems*	
Penerapan Pola	Menetukan pola *control loops* untuk situasi tertentu, dengan pertimbangan kualitas *requirement* penghambat/ pendorong, serta kesesuaiannya dengan kebutuhan domain
Kelengkapan	Mendefinisikan set lengkap dari pola yang bisa diterapkan untuk *self-management*
Kualitas Analisis Layanan	Penyediaan teknik dan skema koordinasi untuk pendekatan desentralisasi, serta dapat menjamin *goal* seluruh sistem dan sifat-sifat adaptasi

Tantangan dan Peluang Penelitian	
Area	***Deskripsi***
Verifikasi dan Validasi (V&V) Run-time - *Self-Adaptive Systems*	
Evolusi Requirements	Melacak *requirement evolution* yaitu apa dan kapan serta teknik V&V untuk mengatur ketercapaian *requirement*
Kontrol Kompleksitas	Mengontrol kompleksitas sistem dan mengintegrasikan *design-time* dan *run-time* V&V
Pemantauan Dinamis	Instrumentasi pemantauan dinamis untuk *run-time* V&V dapat dilaksanakan sepanjang proses adaptasi

III. PENDEKATAN DALAM SELF-ADAPTIVE SYSTEMS

Terdapat berbagai pendekatan yang dapat digunakan dalam pengembangan SAS, disini kami mengelompokan pendekatan berdasarkan [6], yaitu pendekatan utama yang menyoroti peran kunci dari bidang *computer science* dan *cybernetics*, serta pendekatan umum dan khusus merupakan *tools* dan metode untuk mengatasi topik kunci dari pendekatan empat proses utama adaptasi sistem, yaitu *monitoring*, *analyzing*, *planning*, dan *execution* (MAPE). Pendekatan umum dapat membantu perancang dan pengembang dalam menciptakan SAS dari sudut pandang sistem secara keseluruhan, sedangkan pendekatan khusus dapat membantu dalam pembelajaran, perancangan, implementasi dan dukungan bagi salah satu proses tertentu pada SAS seperti dalam proses MAPE.

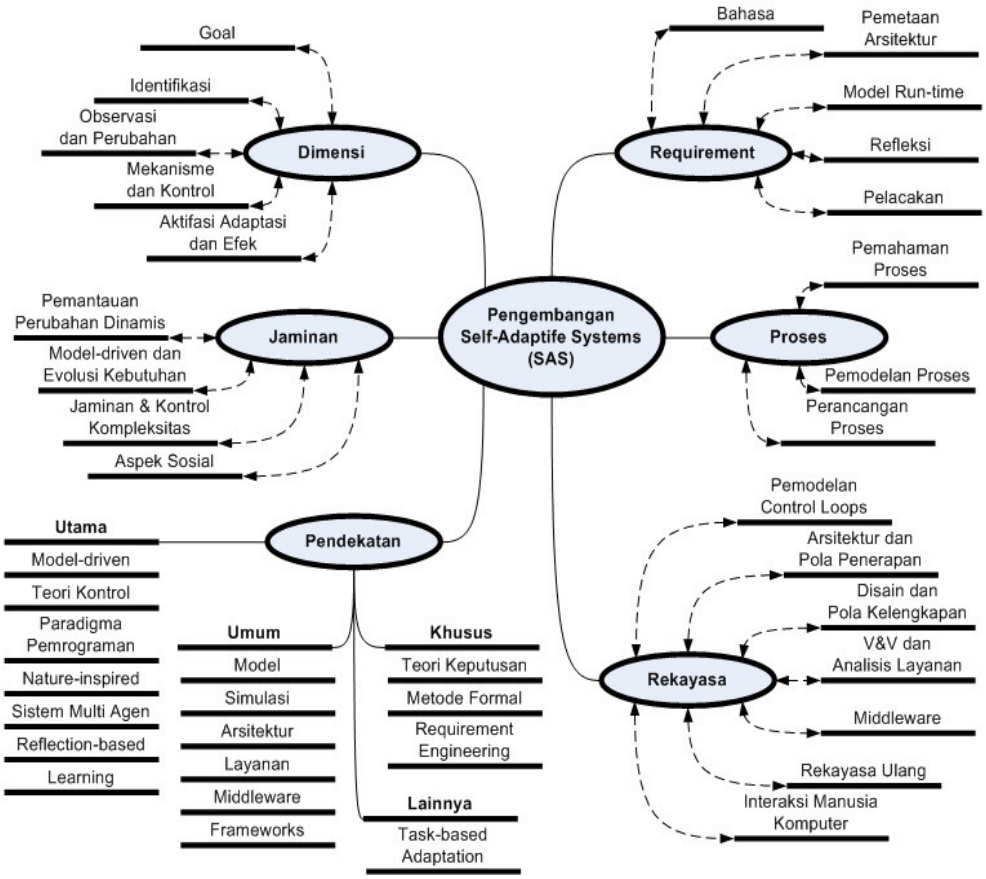
Tabel III menyajikan pengelompokan pendekatan tersebut, dan kami meringkas isi dari setiap pendekatan dengan mengacu dari [6][7][8][3], serta paper-paper terkait lainnya. Kolom pendekatan adalah nama dari pendekatan, kolom deskripsi merupakan penjelasan singkat dari setiap pendekatan, dan kolom aternatif adalah beberapa pendekatan yang telah ada.

TABEL III. PENDEKATAN SELF-ADAPTIVE SYSTEMS

Pendekatan Engineering		
Pendekatan	***Deskripsi***	***Alternatif***
Pendekatan Utama		
Model-driven	Model penalaran, *goal*, konteks, serta abstraksi untuk transformasi dari model konseptual ke kode otomatis	Model@run-time, SPL, MUSIC, Meta-models, Software generator, Model-based process
Teori Kontrol	Mekanisme *feedback* dan *feedforward loops* untuk pengendali perubahan	PID controller, MIAC, MIRAC, MAPE, MAPE-K, UML profile
Paradigma Pemrogram-an	Teknik pemrograman untuk kebutuhan adaptasi struktural dan parameter system dan pengaturan perilaku perangkat lunak	Component-based, Aspect-oriented, Generative, Adaptive, Context-oriented
Nature-inspired	Komputasi sistem yang terinspirasi dari sifat dan perilaku alam, terdiri dari empat metafora kunci inspirasi, yaitu biologi, fisika, kimia dan sosial.	Optimization, Autonomic computing, RAPPID, Data harvesting, Region detection, Immune system, Ecosystem framework, Social convention
Sistem Multi Agen	Adaptasi melalui sifat utama agen, yaitu otonom, reaktif, proaktif, serta kemampuan bersosial.	Agent modeling, Hybrid agent, Design pattern, MOCAS, Unity, Dynamical analysis, Decentralized
Reflection-based	Kemampuan memeriksa dan memodifikasi struktur dan perilaku sistem saat *runtime*, dalam berbagai tingkatan.	Introspection, Intercession, Reflection, Carisma, Reflective middleware, Forms, Req.-awareness
Learning	Optimasi struktur, parameter, algoritma untuk performasi	FUSION, Unity, Control-based, Evolutionary
Metode dan Tools Umum		
Model	Pemodelan sistem untuk mencapai sifat adaptasi dalam menangani perubahan	Model@run-time, Goal-based model, SIG, SPA, BDI, System of systems
Simulasi	Tknik-teknik serta prosedur-prosedur komputasi untuk aktivitas pemodelan dan simulasi adaptasi sistem	Game theory, Spin glasses, Time series, Fuzzy logic, Multiobjective optimization, Evolutionary dynamics
Berbasis Arsitektur	Pendekatan untuk mewakili struktur sistem, tanggung jawab, dan penalaran adaptasi pada tingkatan arsitektur	Rainbow framework, 3L approach, Architectural run-time configuration manager, Archstudio
Berorientasi Layanan	Logika adaptasi dan cara-cara pertukaran layanan melalui adaptasi struktural dan perubahan komposisi layanan	MUSIC, SASSY, MetaSelf, Aspect & service, Agent & SOA, MOSES, Component & services, RE & service
Middleware	Pendekatan untuk adaptasi dan fleksibilitas *middleware* sebagai artefak terintegrasi	SOA, DEVS, DEVSML, adaptive middleware, generic middleware
Frameworks	Kerangka kerja dan standar untuk suatu kelayakan teknis dari koleksi sistem adaptif	J2EE, .NET, ACE, OSGi, JADE, Jadex, Jason, Netlog, JACK, DEVS
Metode dan Tools Khusus		
Teori Keputusan	Mekanisme pengambilan keputusan untuk tindakan yang tepat pada perubahan	Goal, Game theory, ANFIS, Cognitive decision-making, Mathematical framework
Metode Formal	Menjamin dan memverifikasi kebenaran dari perilaku dan struktural adaptasi sistem	FORMS, Restore-invariant approach, Self-testing framework, Timed hazard
Requirement Engineering	Kabutuhan untuk keputusan adaptasi yang berhubungan dengan situasi ketidakpastian, serta perubahan pada saat *design-time* dan *run-time*	Requirement@run-time, LoREM, Zanshin, Self-tuning method, KAOS, i*, SIG, FLAGS, RELAX, CARE, Tropos4AS
Pendekatan Lain		
Task-based adaptation	Kebijakan adaptasi berbasis karakteristik *users' task*	Utility-based adaptation, Task-based self-adaptation

IV. REQUIREMENTS ENGINEERING UNTUK SAS

Berdasarkan pembahasan seminar pada tabel I dan II, maka tantangan dan peluang penelitian pengembangan SAS dapat dipetakan seperti dapat dilihat pada gambar 1. Terdiri dari lima area, yaitu dimensi (berisi deskripsi dimensi pemodelan dan ruang disain), *requirement*, proses, rekayasa (berisi deskripsi rekayasa dan *control loops*), dan jaminan (berisi kriteria jaminan dan V&V *runtime*). Dimana setiap deskripsi pada tabel I dan II dapat dijadikan rujukan untuk menentukan area penelitian, tentunya setelah disesuaikan dengan permasalahan penelitian yang akan diselesaikan.



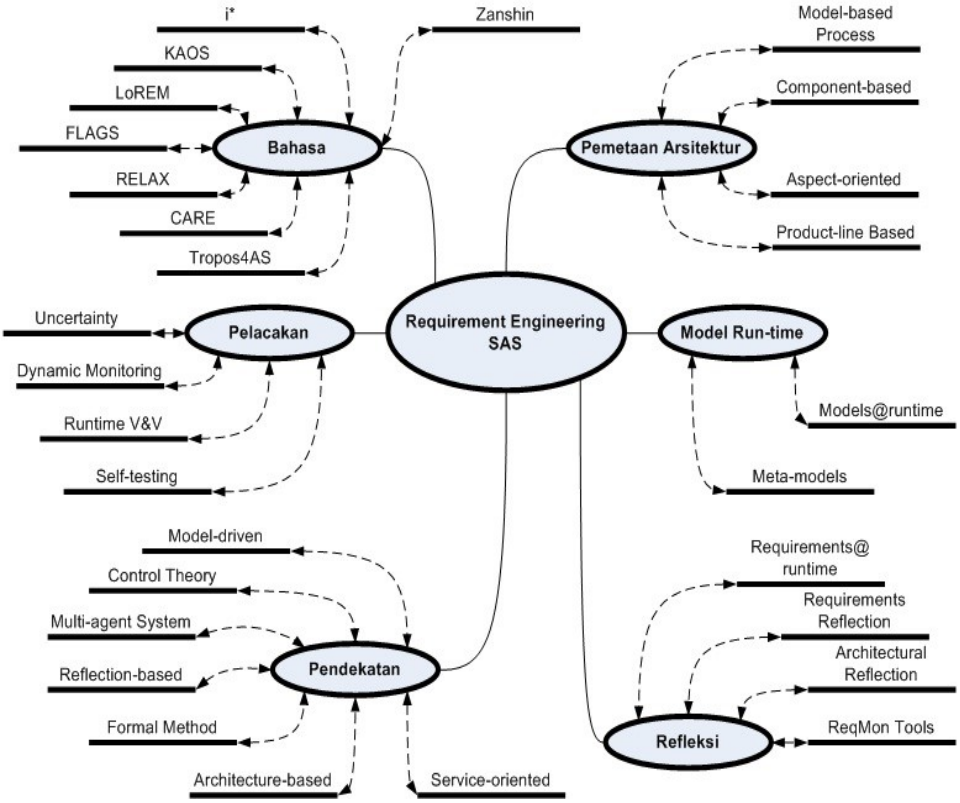
Gambar 1. Area dan pendekatan pengembangan *self-adaptive systems*.

A. *Area Requirement Engineering dan Penelitian Terkait*

Seperti telah diuraikan sebelumnya SAS merupakan spektrum yang sangat luas, pada bagian ini kami memetakan posisi area *requirement engineering* dalam SAS dengan merujuk dari pembahasan sebelumnya, sehingga gambaran umum dari area ini seperti dapat dilihat pada gambar 2, terdapat lima sub area, yaitu bahasa, pemetaan arsitektur, model run-time, refleksi, dan pelacakan, serta berbagai pendekatan yang dapat digunakan.

Bahasa *requirement* menjadi area tersendiri dalam penelitian SAS sebagai bagian dari *requirement engineering*. Telah banyak model bahasa *requirement* yang dikembangkan para peneliti, diantaranya adalah model i* yang saat ini banyak dijadikan sebagai rujukan bagi pengembangan model untuk *requirement*, termasuk kebutuhan SAS. i* menggunakan pendekatan *goal-based* diperkenalkan oleh Eric Yu [9], melalui perspektif intensional digunakan untuk pemahaman terhadap kebutuhan sistem, terutama bagaimana mengetahui adanya kebutuhan khusus dalam sebuah domain, dan mempersiapkan kebutuhan untuk menghadapi perubahan. i* diadopsi oleh model CARE [10] dan Tropos4AS [11], dengan menggunakan pendekatan *agent-based* untuk mengembangkan SAS. Model bahasa *requirement* lainnya dengan pendekatan *goal-based* adalah KAOS [12], yaitu menyediakan suatu mekanisme grafis untuk menyajikan adaptasi *semantics*, peneliti [13] mengidentifikasi *high-level objectives* untuk setiap adaptasi *semantics* dan direpresentasikan sesuai dengan entitas *goal* KAOS. Peneliti [14] menggunakan KAOS sebagai metode untuk menganalisis dan memodelkan kebutuhan dalam mengkonstruksi SAS. Selain itu, LoREM [15] melalui pendekatan *goal-based modeling* dapat digunakan untuk *requirement* SAS secara dinamis, dan FLAGS [16][17] dengan konsep *fuzzy goal* nya dapat diterapkan untuk pemodelan kebutuhan pada saat *runtime*. Berdasarkan catatan beberapa paper, pendekatan *goal-based* ini telah banyak mengalami keberhasilan sebagai dasar pembentuk perilaku otonom untuk spesifikasi kebutuhan SAS. Bahasa

requirement lainnya yang lebih terfokus pada kebutuhan penanganan ketidakpastian bersama-sama dengan CARE, LoREM dan FLAGS adalah RELAX [18][19], melalui notasinya menentukan kebutuhan SAS berdasarkan pendekatan *natural language prose* dan *fuzzy logic*. Pendekatan lainnya yang dapat digunakan dalam *requirement* SAS adalah teori kontrol, Souza [20][21][22] mengusulkan model *requirement* (Zanshin) dengan menetapkan dua kelas baru yaitu *awareness requirements* dan *evolution requirements*, yang masing-masing direpresentasikan sebagai indikator dan strategi untuk menangani adaptasi perubahan. Selain itu, pendekatan teori kontrol juga digunakan melalui penerapan model i*, UML profile [23], dan pendekatan *discrete time markov chains* [24][25].



Gambar 2. *Requirement engineering* dalam *self-adaptive systems*.

Sebagai upaya dalam memenuhi kebutuhan adaptasi sistem secara *run-time*, diperlukan pemetaan terhadap arsitektur tertentu sesuai dengan model bahasa *requirement* yang telah dikembangkan. Secara umum pendekatan *model-driven architecture* banyak diadopsi untuk kebutuhan tersebut, seperti yang telah dilakukan Renata [26] melalui pendekatan *agent-based* dengan mengadopsi model i* dan Tropos mengembangkan sebuah bahasa *requirement* yang kemudian dipetakan kedalam tiga tingkatan abstraksi *model-driven*. Zhang [27] mengusulkan *model-based process* dalam mengembangkan model adaptasi, pembuatan kode secara otomatis dari model, untuk membangun program yang adaptif serta model verifikasi dan validasi. Ghorbani [28], mengusulkan *model-driven* melalui simulasi dan pemodelan *agent-based*, dengan *procedural semantics* melakukan transformasi model kedalam *executable simulation*. Selain itu menurut [4], terdapat berbagai pilihan teknis yang dapat digunakan untuk merekonfigurasi pada setiap tingkatan arsitektur, diantaranya adalah pendekatan *component-based*, *aspect-oriented* dan *product-line based*, serta kombinasi diantara ketiganya. Misalnya [29], melalui pendekatan *component-based* mengusulkan model untuk mendukung pengembangan *model-driven* yang memiliki sifat *reflective*, serta model arsitektur yang dapat membangkitkan operasi sistem adaptif berbasis komponen [30], melalui *fractal initiative* [31], serta digunakan untuk kebutuhan fungsi *self-management* [32], dan *self-repair* [33]. Beberapa contoh pendekatan *aspect-oriented* dalam pemetaan arsitektur, misalnya pendekatan *model-driven* untuk mengelola variabilitas dinamis [34], kebutuhan arsitektur *self-organizing* [35], serta penggunaan model dan aspek untuk penanganan adaptasi secara dinamis [36]. Sementara pendekatan *product-line based*, misalnya digunakan untuk komputasi otonom dalam model *runtime* [37], serta pemodelan konteks dan adaptasi dinamis melalui *feature model* [38].

Faktor penting lainnya dalam *requirement engineering* untuk SAS adalah model *runtime*, area ini menyoroti bagaimana aktivitas *requirement* pada waktu disain dapat memenuhi model perilaku otonom untuk spesifikasi kebutuhan baru pada saat *runtime*, atau melakukan aktivitas *requirement* pada saat *runtime*. Cheng [39] menyajikan pandangan terkait kriteria jaminan SAS yang digolongkan berdasarkan kebutuhan fungsional dan non-fungsional, selain itu Cheng juga membahas beberapa pendekatan yang

dapat digunakan. Wang [40] mengusulkan pendekatan *goal tree* yang dapat digunakan pada saat sistem melakukan penyesuaian sumberdaya dan lingkungannya melalui algoritma seleksi. Beydoun [41] membuat mekanisme pengidentifikasian *ontology* yang tepat sesuai kebutuhan pengembangan sistem, melalui evaluasi kesamaan *semantics* antara *ontology* yang dibutuhkan dengan domain *ontology* dalam repositori. Sniezynski [42] menerapkan pendekatan *agent* untuk mengawasi dan menentukan strategi adaptasi arsitektur layanan pada suatu perubahan kondisi, melalui optimasi sistem dengan penggunaan *supervised learning*. Penelitian lainnya yang berhubungan dengan model *runtime* ini, diantaranya melalui evolusi model dengan parameter adaptasi [43], *context-aware systems* [44], serta beberapa pendekatan model@runtime [45][46]. Penanganan kebutuhan model *runtime* juga dapat didukung oleh penyediaan suatu *meta-models*, misalnya melalui suatu bahasa untuk merekayasa *feedback loops* sehingga dapat mengeksekusi *runtime megamodels* [52], dan independen *meta-models* yang berhubungan dengan independensi *meta-data* [53].

Requirements reflection merupakan kebutuhan yang harus dipenuhi dalam *requirement engineering* suatu SAS, yaitu kemampuan sistem untuk memeriksa dan memungkinkan melakukan modifikasi struktur (*structural reflection*) dan perilaku (*behavioral reflection*) pada saat *runtime*. Terdiri dari dua aktivitas yaitu *introspection* mengacu pada pengamatan perilaku sebagai alasan untuk adaptasi, dan *intersecession* sebagai reaksi dari hasil *instrospection* yaitu pengendali struktur, parameter dan konteks adaptasi [7]. Sawyer [49] mengusulkan *requirements-awareness* dalam melakukan introspeksi kebutuhan pada saat *runtime* untuk menangani aspek ketidakpastian. Bencomo [50] mengusulkan model *requirement* sebagai entitas *runtime* dan mengintegrasikan kemampuan *reflection* pada aktivitas *requirement engineering*. Selain itu *requirement reflection* dapat diwujudkan misalnya melalui pendekatan *active architecture* [51] dan pengembangan suatu *framework* untuk memonitor aktivitas *requirement* [52].

Dalam mengatasi ketidak lengkapan informasi mengenai lingkungan dan evolusi sistem saat melakukan aktivitas *requirement* suatu SAS, adalah dengan menyediakan suatu mekanisme pelacakan kebutuhan secara dinamis baik pada saat fase pengembangan maupun pada saat *runtime*. Misalnya suatu bahasa *requirement* harus dapat melacak kebutuhan terkait kebutuhan arsitektur, disain, dan seterusnya, serta harus dapat memastikan bahwa perilaku sistem sesuai dengan *requirement*. Dengan demikian dalam rangka menjamin suatu perilaku adaptif, penting untuk memantau kepatuhan dan pelacakan serta evolusi *requirement* selama *runtime*. Murray [53] berpendapat bahwa *feedback loop* adalah alat utama dalam menangani ketidakpastian, dengan mengukur pengoperasian suatu sistem, membandingkannya dengan referensi pada saat *runtime*, dan menyesuaikan dengan variabel kontrol yang tersedia. Penanganan ketidakpastian dalam *requirement engineering* untuk SAS harus dilakukan pada saat *runtime* [64], pendekatan yang dapat digunakan misalnya *state machines* untuk model adaptif sistem dengan transisi sebagai sistem rekonfigurasinya [55]. Pemantauan suatu konteks secara dinamis dapat dilakukan, sehingga memungkinkan dapat melakukan verifikasi dan validasi secara *runtime* melalui mekanisme pemantauan yang relevan, dengan melacak aspek untuk divalidasi, bahkan ketika memantau perubahan kebutuhan pada saat *runtime* [56]. Aktivitas verifikasi pada saat *runtime* dapat dilakukan misalnya dengan pendekatan *probabilistic runtime model* menggunakan *discrete time markov chains* [57]. Sementara King [58] mengusulkan *self-testing framework* untuk sistem komputasi otonom yang memvalidasi permintaan perubahan struktural dan perilaku saat *runtime* untuk menghindari kegagalan sistem, pendekatan ini menggunakan kontrol adaptasi eksternal.

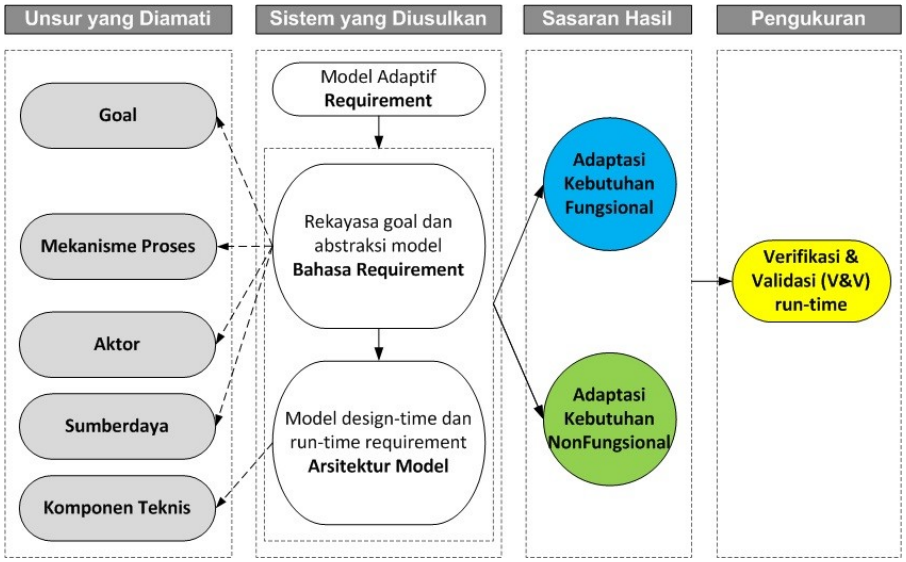
B. Kerangka Umum Requirement Engineering

Menurut [4], *requirement engineering* berkaitan dengan apa yang harus sistem lakukan dan di mana kendala harus diselesaikan. Oleh karena itu, *requirement engineering* harus mengatasi adaptasi apa yang mungkin, dan apa kendalanya, serta bagaimana adaptasi direalisasikan. Secara khusus, pertanyaan yang harus ditangani meliputi "aspek lingkungan apa yang relevan untuk adaptasi". Kebutuhan yang diizinkan untuk mengubah atau berevolusi pada saat sistem *runtime*, dan yang harus selalu dipertahankan. Singkatnya, *requirement engineering* harus berurusan dengan ketidakpastian, karena informasi tentang lingkungan di masa depan tidak lengkap, dan oleh karena itu kebutuhan untuk perilaku sistem perlu dirubah (pada saat *runtime*) dalam menanggapi perubahan lingkungan.

Dengan melihat pembahasan poin A pada bagian IV makalah ini, dapat disimpulkan bahwa area *requirement engineering* merupakan area penelitian yang cukup luas. Disini kami menetapkan kerangka pemikiran awal seperti ditunjukan pada gambar 3, berdasarkan asumsi kriteria yang ditetapkan, misalnya pendekatan bahasa *requirement* yang diadposi adalah *goal-based* (dari beberapa *paper survey* pendekatan ini cukup menunjukan tingkat keberhasilan). Dengan demikian, maka unsur lingkungan yang harus diamati dengan merujuk pada paper kami sebelumnya [2] terdiri dari *goal*, mekanisme proses, aktor, semberdaya dan komponen teknis. Kelima unsur ini direkayasa menjadi suatu abstraksi model yang dapat memenuhi kriteria SAS, yaitu bagaimana model *requirement* dapat adaptif terhadap kebutuhan pada saat *design-time* dan *run-time*. Selain itu, pemilihan pendekatan untuk kebutuhan pemetaan arsitektur yang sesuai juga harus dilakukan.

Dalam *requirement engineering* dikenal adanya kebutuhan fungsional dan non-fungsional, dalam mengembangkan SAS maka kedua kebutuhan ini harus disesuaikan dengan karakteristik dari SAS tersebut. Menurut [4], jaminan terhadap kedua

kebutuhan tersebut diperlukan sepanjang/ pada seluruh siklus hidup sistem, termasuk pada saat sistem *runtime*, yaitu harus dipenuhi sebelum, selama, dan setelah adaptasi. Dalam rangka memenuhi adaptasi bagi kebutuhan fungsional dan non-fungsional tersebut, pendekatan *goal based* sebagai salah satu teknik yang dapat digunakan sebagai jaminan dari sifat adaptif sistem pada saat *design-time* dan *run-time*, dapat diadopsi melalui berbagai mekanisme. Misalnya pada waktu pengembangan, dapat digunakan untuk menentukan ekspektasi *stakeholder*, dan kriteria keputusan untuk perilaku sistem yang sesuai dapat diturunkan dari model ini [59].



Gambar 3. Kerangka umum *requirement engineering* dalam SAS.

Selain itu, *high-level goals* dapat memberikan kriteria jaminan yang sesuai dalam *high dynamic systems,* misalnya representasi perilaku fungsional dalam *high-level goals* didekomposisi menjadi *sub-goals*, setiap alternatif dekomposisi dapat memenuhi syarat suatu kriteria kualitas, preferensi pengguna, dan konteks yang dapat berkontribusi positif atau negatif. Untuk memastikan perilaku yang diharapkan, sistem harus memilih jalur dekomposisi *goal* yang paling tepat [60]. Model *goal-based* dapat dialihkan dari *design-time* menjadi *run-time* untuk melacak perubahan kebutuhan SAS ketika *runtime*, misalnya melalui penginvestigasian siklus hidup *goal* saat *runtime* [61], atau sebuah *framework* untuk memonitor kebutuhan dengan pendekatan *multiple feedback loops* untuk memantau *awareness requirements* dan evolusi yang mengarah pada tujuan adaptasi saat *runtime* [62].

Selain beberapa alternatif rekayasa *goal* yang telah dibahas tersebut, terdapat beberapa rujukan yang dapat menjadi sumber inspirasi seperti telah dibahas pada poin A bagian IV makalah ini. Sebagai upaya kebutuhan pengembangan arsitektur sistem secara utuh, pendekatan *model-driven* dan beberapa teknik pemetaan arsitektur lainnya dapat dikolaborasikan dengan bahasa *requirements* untuk menciptakan sifat *self-adaptation*.

V. KESIMPULAN

SAS merupakan spektrum yang sangat luas, perspektif terhadap kebutuhan pengembangannya memerlukan pemahaman yang komprehensif dan mendalam. Sebagai langkah awal dalam upaya memulai aktivitas pengembangan, dapat dilakukan melalui pemahaman terhadap area-area yang menjadi bagian dari pembahasan SAS, termasuk tantangan dan peluang penelitian terkini. Berdasarkan permasalahan yang akan diselesaikan, maka memahami berbagai pendekatan dalam setiap area SAS dapat menjadi pengetahuan yang relevan bagi setiap alternatif solusi dari permasalahan tersebut.

Pada makalah ini disajikan beberapa informasi terkait area SAS yang merujuk dari seminar IBFI - Schloss Dagstuhl Germany, dan dapat dijadikan sumber pengetahuan untuk mengenali tantangan dan peluang terkini, selain itu berbagai alternatif pendekatan disajikan melalui pengelompokan dan contoh pendekatan yang ada saat ini, dan diakhir bahasan kami memetakan kebutuhan penelitian terkait area *requirement engineering*.

Kerangka yang kami susun merupakan konsep awal dan masih memerlukan kajian yang lebih mendalam, pada contoh bahasan dilontarkan kebutuhan suatu model yang dapat memiliki sifat SAS baik pada saat aktivitas pengembangan, maupun pada saat sistem sedang berjalan. Penting menjadi suatu catatan bahwa dalam perspektif SAS jaminan terhadap pemenuhan kebutuhan sistem harus dilakukan pada seluruh tahapan siklus hidup sistem, oleh karena itu diperlukan mekanisme yang dapat mengelola kebutuhan adaptasi, melalui pengaturan suatu *control objective*, proses adaptasi itu sendiri, dan sistem monitoring. Langkah kami berikutnya adalah menentukan spesifikasi dari bahasa *requirement* serta kebutuhan pemetaan terhadap arsitektur, dengan mengacu pada mekanisme tersebut.

REFERENSI

[1] Aradea, I. Supriana S., K. Surendro, "An Overview of Multi Agent System Approach In Knowledge Management Model", International Conference on Information Technology Systems and Innovation (ICITSI), School of Electrical Engineering and Informatics, ITB, 2014.

[2] Aradea, I. Supriana S., K. Surendro, "Prinsip Paradigma Agen Dalam Menjamin Keberlangsungan Hidup Sistem", Konferensi Nasional Sistem Informasi, STEI ITB - Universitas Klabat Sulawesi Utara, 2015.

[3] Aradea, I. Supriana S., K. Surendro, "Roadmap dan Area Penelitian Self-Adaptive Systems", Seminar Nasional Teknik Informatika dan Sistem Informasi, Universitas Maranatha Bandung, 2015.

[4] B.H.C. Cheng, et. al., "Software Engineering for Self-Adaptive Systems : A Research Roadmap", Self-Adaptive Systems, LNCS 5525, Springer-Verlag Berlin Heidelberg, 2009.

[5] R. de Lemos, et. al., "Software Engineering for Self-Adaptive Systems : A Second Research Roadmap", Self-Adaptive Systems, LNCS 7475, pp. 1–32, Springer-Verlag Berlin Heidelberg, 2013.

[6] F. D. Macías-Escrivá, R. Haber, R. del Toro, V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications", Expert Systems with Applications 40, 7267–7279, Elsevier Ltd. All rights reserved, 2013.

[7] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, C. Becker, "A survey on engineering approaches for self-adaptive systems", Pervasive and Mobile Computing, 1574-1192, Elsevier B.V. , 2014.

[8] D. Weyns, Sam Malekb, Sam Malekb, Bradley Schmerlc, "Introduction to the special issue on state of the art in engineering SAS", The Journal of Systems and Software 85 (2012) 2675– 2677, Elsevier Inc., 2012.

[9] E.S. Yu, "Towards modeling and reasoning support for early-phase requirements engineering", 3rd IEEE International Symposium on Requirements Engineering, Washington, DC, USA, p. 226, 1997.

[10] N.A. Qureshi, A. Perini, "Continuous adaptive requirements engineering: an architecture for self-adaptive service-based applications", in: Proc. RE@RunTime, IEEE, pp. 17–24., 2010

[11] M. Morandini, L. Penserini, A. Perini, "Modelling self-adaptivity: a goal-oriented approach", in: Proc. SASO, IEEE, pp. 469–470., 2008.

[12] Dardenne, A., van Lamsweerde, A., Fickas, S., "Goal directed requirements acquisition", In: Selected Papers of the Sixth International Workshop on Software Specification and Design, pp. 3–50, 1993.

[13] Brown, G., Cheng, B. H. C., Goldsby, H., & Zhang, J., "Goal-oriented specification of adaptation requirements engineering in adaptive systems", Proceedings of international workshop on self-adaptation and self-managing systems (pp. 23–29). Shanghai, China: ACM., 2006.

[14] Nakagawa, H., Ohsuga, A., & Honiden, S., "Constructing Self-Adaptive Systems Using a KAOS Model". Proceedings of second IEEE international conference on self-adaptive and self-organizing systems workshops (pp. 132–137): IEEE Computer Society, 2008.

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

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

[17] L. Pasquale, L. Baresi, B. Nuseibeh, Towards adaptive systems through requirements@runtime, in: Proc. MRT, CEUR-WS, pp. 13–24., 2011.

[18] J. Whittle, P. Sawyer, N. Bencomo, B.H.C. Cheng, J. Bruel, "RELAX: incorporating uncertainty into the specification of self-adaptive systems", in: Proc. RE, IEEE, pp. 79–88., 2009.

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

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

[21] V. E. Silva Souza, A. Lapouchnian, and J. Mylopoulos, "Requirements-Driven Qualitative Adaptation", R. Meersman et al. (Eds.): OTM 2012, Part I, LNCS 7565, pp. 342–361, Springer-Verlag Berlin, 2012.

[22] J. Pimentel, K. Angelopoulos, V. E. Silva Souza, J. Mylopoulos, and J. Castro, "From Requirements to Architectures for Better Adaptive Software Systems", Proc. i* Workshop, CEUR Vol-978., 2013.

[23] R. Hebig et al., "Making Control Loops Explicit when Architecting Self-Adaptive Systems". In Proc. of the 2nd International Workshop on Self-organizing Architectures, pages 21-28. ACM, 2010.

[24] A. Filieri et al., "Self-Adaptive Software Meets Control Theory: A Preliminary Approach Supporting Reliability Requirements". In Proc. of the 26th IEEE/ACM International Conference on Automated Software Engineering, pages 283-292. IEEE, 2011.

[25] A. Filieri et. al., "Reliability-driven dynamic binding via feedback control". In Private communication, 2012.

[26] G. Renata S., A. Perini, and V. Dignum, "Socially Grounded Analysis of KM Systems and Processes", Invited Chapter in Social Modeling for RE, Cooperative IS Series, Cambridge, MA: MIT Press, 2009.

[27] J. Zhang, B.H.C. Cheng, "Model-based development of dynamically adaptive software", in: Proc. ICSE, ACM, pp. 371–380., 2006.

[28] A. Ghorbani, G.P.J. Dijkema, P. Bots, H. Alderwereld, V. Dignum, "Model-driven agent-based simulation: Procedural semantics of a MAIA model", Simulation Modelling Practice and Theory 49, 27–40., 2014.

[29] N. Bencomo, P. Grace, C. Flores, D. Hughes, G. Blair, Genie: supporting the model driven development of reflective, component-based adaptive systems, in: Proc. ICSE, ACM, pp. 811–814., 2008.

[30] N. Bencomo, G. Blair, Using architecture models to support the generation and operation of component-based adaptive systems, in: Software Engineering for Self-Adaptive Systems, in: LNCS, vol. 5525, Springer, pp. 183–200., 2009.

[31] G. Blair, T. Coupaye, J.B. Stefani, Component-based architecture: the fractal initiative, Ann. Telecommun., 64 (1–2) 1–4., 2009.

[32] D. Sykes, W. Heaven, J. Magee, J. Kramer, From goals to components: a combined approach to self-management, in: Proc. SEAMS, ACM, pp. 1–8., 2008.

[33] S. Sicard, F. Boyer, N. De Palma, Using components for architecture-based management: the self-repair case, in: Proc. ICSE, ACM, pp. 101–110., 2008.

[34] B. Morin, O. Barais, G. Nain, J.-M. Jézéquel, Taming dynamically adaptive systems using models and aspects, in: Proc. ICSE, IEEE, pp. 122–132., 2009.

[35] B. Morin, F. Fleurey, N. Bencomo, J.-M. Jézéquel, A. Solberg, V. Dehlen, G. Blair, An aspect-oriented and model-driven approach for managing dynamic variability, in: Model Driven Engineering Languages and Systems, in: LNCS, vol. 5301, Springer, pp. 782–796., 2008.

[36] R. Haesevoets, E. Truyen, T. Holvoet, W. Joosen, Weaving the fabric of the control loop through aspects, in: Self-Organizing Architectures, in: LNCS, vol. 6090, Springer, pp. 38–65., 2009.

[37] C. Cetina, P. Giner, J. Fons, V. Pelechano, Autonomic computing through reuse of variability models at runtime: the case of smart homes, IEEE Comput. 42 (10) 37–43., 2009.

[38] M. Acher, P. Collet, F. Fleurey, P. Lahire, S. Moisan, J.-P. Rigault, Modeling context and dynamic adaptations with feature models, in: Proc. MRT, vol. 509, CEUR-WS.org, pp. 89–98., 2009.

[39] B. H. C. Cheng, K. I. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. A. M¨uller, P. Pelliccione, A. Perini, N. A. Qureshi, B. Rumpe, D. Schneider, F. Trollmann, and N. Villegas, "Using models at runtime to address assurance for SAS", LNCS 8378, pp. 101–136, Springer, 2014.

[40] T. Wang, B. Li, L. Zhao, and X. Zhang, "A goal-driven self-adaptive software system design framework based on agent", ICAPIE Organization Commite, Published by Elsevier B.V., 2012.

[41] G. Beydoun, G. Lowb, F. García-Sánchez, R. Valencia-García, R. Martínez-Béjar, " Identification of ontologies to support information systems development", International Journal of IS, 46, pp. 45–60, 2014.

[42] B. Sniezynski, "Agent-based adaptation system for service-oriented architectures using supervised learning", 14th International Conference on Computational Science, Volume 29, Pages 1057–1067, 2014.

[43] I. Epifani, C. Ghezzi, R. Mirandola, G. Tamburrelli, "Model evolution by run-time parameter adaptation", in: ICSE, IEEE, pp. 111–121., 2009.

[44] G.H. Alférez, V. Pelechano, "Dynamic evolution of context-aware systems with models at runtime", in: MDE Languages and Systems, in: LNCS, vol. 7590, Springer, pp. 70–86., 2012.

[45] B. Morin, O. Barais, J.-M. Jézéquel, F. Fleurey, A. Solberg, "Models@ Run.time to support dynamic adaptation", IEEE Comput. 42 (10) 44–51., 2009.

[46] G. Blair, N. Bencomo, R.B. France, "Models@ Run.time", IEEE Comput. 42 (10) 22–27., 2009.

[47] T. Vogel, H. Giese, "A language for feedback loops in self-adaptive systems: executable runtime megamodels", in: Proc. SEAMS, IEEE, pp. 129–138., 2012.

[48] S. McGinnes and E. Kapros, "Conceptual independence : A design principle for the construction of adaptive information systems", International Journal of Information Systems, 47, pp. 33–50, 2014.

[49] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, A. Finkelstein, "Requirements-aware systems: a research agenda for RE for self-adaptive systems", in: Proc. RE, IEEE, pp. 95–103., 2010.

[50] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, E. Letier, "Requirements reflection: requirements as runtime entities", in: Proc. ICSE—Vol. 2, ACM/IEEE, pp. 199–202., 2010

[51] R. Morrison, D. Balasubramaniam, F. Oquendo, B. Warboys, R.M. Greenwood, "An active architecture approach to dynamic systems co-evolution", in: Software Architecture, in: LNCS, vol. 4758, Springer, pp. 2–10., 2007.

[52] Robinson, W., "A requirements monitoring framework for enterprise systems". Requirements Engineering 11, 17–24., 2006.

[53] Murray, R.M., A˙ stro¨m, K.J., Boyd, S.P., Brockett, R.W., Stein, G., "Future Directions in Control in an Information Rich World", IEEE Control Systems 23, 20–33, 2003.

[54] A.J. Ramirez, A.C. Jensen, B.H.C. Cheng, "A taxonomy of uncertainty for dynamically adaptive systems", in: Proc. SEAMS, IEEE, pp. 99–108., 2012.

[55] Goldsby, H.J., Cheng, B.H.C., "Automatically Generating Behavioral Models of Adaptive Systems to Address Uncertainty". In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., V¨olter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 568–583. Springer, Heidelberg, 2008.

[56] Villegas, N.M., M¨uller, H.A., Tamura, G., "Optimizing Run-Time SOA Governance through Context-Driven SLAs and Dynamic Monitoring". In: IEEE International Workshop on the MESOCA, pp. 1–10., 2011.

[57] A. Filieri, G. Tamburrelli, "Probabilistic verification at runtime for self-adaptive systems", in: Assurances for Self-Adaptive Systems, in: LNCS, vol. 7740, Springer, pp. 30–59., 2013.

[58] King, T.M., Ramirez, A.E., Cruz, R., Clarke, P.J,:" An Integrated Self-Testing Framework for Autonomic Computing Systems". Journal of Computers 2(9), 37–49, 2007.

[59] Nguyen, C., Perini, A., Tonella, P., Miles, S., Harman, M., Luck, M., "Evolutionary testing of autonomous software agents". In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 521–528. IFAAMAS, 2009.

[60] Qureshi, N.A., Liaskos, S., Perini, A., "Reasoning about adaptive requirements for self-adaptive systems at runtime". In: Proceedings of the 2nd International Workshop on Requirements@run.time (RE@run.time 2011), pp. 16–22, 2011.

[61] Morandini, M., Penserini, L., Perini, A., "Operational semantics of goal models in adaptive agents". In: Proceedings of the 8th InternationalConference onAutonomous Agents and Multiagent Systems—Volume 1 (AAMAS 2009), pp. 129–136. IFAAMAS, 2009.

[62] Souza, V.E.S., Lapouchnian, A., Robinson, W.N., Mylopoulos, J., "Awareness requirements for adaptive systems". In: Proceedings of the 6th International Symposium on SEAMS, pp. 60–69, 2011.