Design and architectural support for self-adaption

Rafat Aghazadeh

Master of Computer Engineering, Pardis Shahid Beheshti University, Tehran, Iran
Aghazadeh.ra@gmail.com

Abstract: Software systems have been developed in various centres extensively. They play important role and basic in any organization. If these systems became out of service can lead to irrecoverable damage to the organization. In order to keep these systems running continuously. They need to repair, recover and control themselves without human intervention, until any abnormal condition happens they will be able to keep system available with making right decision. For these reasons to be felt requirement to self-adaptive capability in systems. Researchers proposed different designs and trends for achieve to better and more efficient to self-adaptive goals. We are going to study a number of them in this paper.
[Rafat Aghazadeh. **Design and architectural support for self-adoptation.** *Researcher* 2015;7(9):7-18]. (ISSN: 1553-9865). http://www.sciencepub.net/researcher. 2

Keywords: self-adaptation, software-architecture

1-Introduction

In the beginning of new century researcher in the IBM introduced a new idea for management of software. This system has ability to repair itself and adapt new application automatically. This operate based on Monitoring, Analysis, Planning, Execution, which is known as MPAE-K.

We briefly describe each of the components. (figure1).

Monitor: They collect information with various method in order to share them with human users.

Analysis: This component has ability to identify situations and relationships between them and also broadcast the future.

Planning: Provides procedures to achieve specified goals.

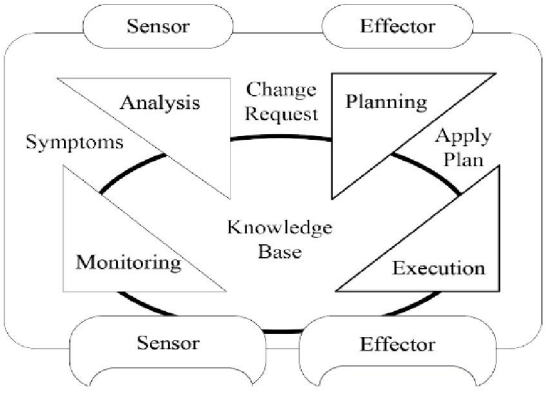

Figure 1. Autonomous control loop

Execute: Provide Procedures for implementing specific programs this part includes required files and standards for policies and also includes analysing, planning and the source of knowledge for execute.

Sensor and effector create access point to managed resources. Any change in structure should apply through sensor and effector. [1]

System perform adopt should integrate functional aspects and management logic. [2]

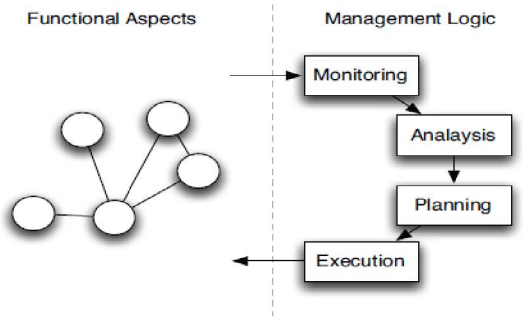

Figure 2. Overview of an adaptive system

Managment logic in terms of MAPE included Monitoring phase collecting run-time data, analysis phase evaluate system behaviour, Plannig phase identifying the corrective actions, Execution phase perform operations.

We can divide this self –adaptive system specification into three level.

1-General Level include Self-Adaptivness

2–Major Level includes Self-Configuring, Self-Healing, Self-Optimizing, Self-Protecting

3- Primitive Level includes Self-Awareness, Context-Awareness

2-factors

Number of factors to evaluate system is given below.

Self-configuration: When system needs changes, this change happens automatically and dynamic perform configuration.

Self-healing: In case of any bugs or problems happens in the systems, reacts to those problems those every process to solve problems.

Self- optimization: Regulating the use of resources to perform the desired operation in the best way possible.

Self-protection: Detects security attacks and perfom defensive measures as needed.

Self-managing: Operations management in a coordinate system.

Self-awareness: Understanding of the system behaviour.

Self-situation: Notification to external situations.

Self-monitoring: The ability to identify and observe the system.

Self-adjustment: Perform the necessary implementing which suits with conditions.

Self-organization: Capability to be accessible straight away.

Self-maintenance: The ability to maintain system automatically maintain.

Self-repairing: Having ability to repair fault and error and return system to normal condition.

Self-tunning: Having ability to self –supervising and self –adjusting mean while having ability to recognise problems and solve them.

Self-contained: This part uses meta model and meta computation to perform adjustments and adaptation.

Self-control: System controls itself and data entry continusly.

Self-evaluating: Are able to evaluate system and own situation in order to increase system performance.

Self-diagnosing: Ability to recognise problems and errors.

Other features include self-configuration, Self-anticipation (ability to predict), Self-critical (enable critical to their ability), Self-defining (ability to define attributes and ….), Self-governing (Ability to self-determination,…), Self-recovery, Self-reflecting, Self-defense.

Functionality, Integrity, Maintainability, Portability, Scaleability, Availability, Reliability, stability are quality factors.

3-existing models/framework

3-1-Ecological Architecture

In this system a software controls instead of human being [3]. (figure 3)

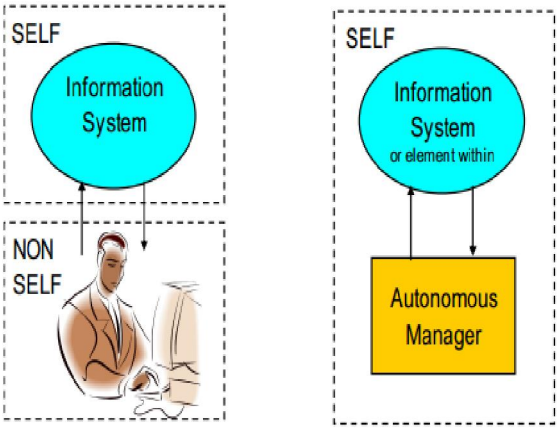

Figure 3. Humans as managers and Autonomous as manager

Autonomous managers perform self –management activities as multiple and distributed.

In a self-organization approach self-organizing components operate with together locally. components include functional component and management components.

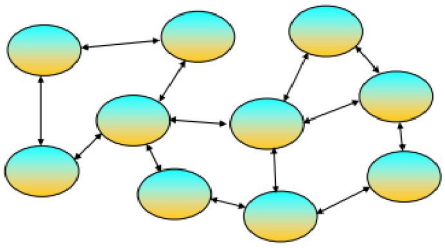

Figure 4. A model of Self-organization

In an ecological approach, self-organizing components operate with together and manager components.

Manager could control overall of system.

However a couple of component are placed into the system as a manager, this component interact with other component, therefore they act as a self-adaptive component.

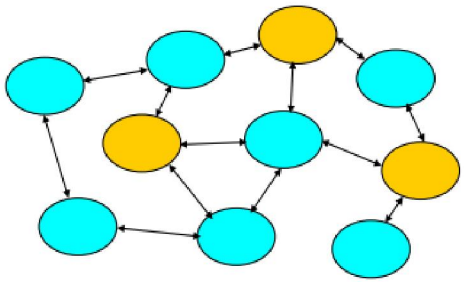

Figure 5. A model of Ecological Approaches.

3-2- on Decentralized Self-Adaption

Distributed system has a number of self-adaptive systems which placed in local systems. This article consider each camera as a self-adapting system which able to report traffic information automatically [4].

A local self-adaptive system include local managed system and self- adaptive units. A self-adaptive unit adapts the local managed system. For adapt uses as meta-level computations and meta-level models in distributed systems.

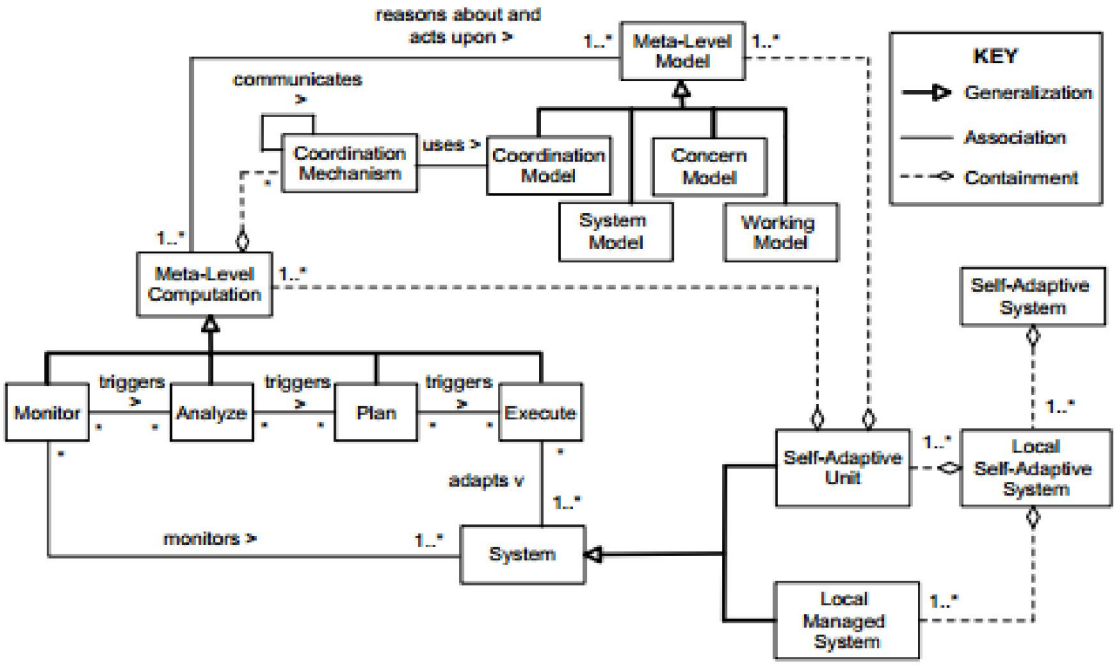

Figure 6. Reference model for decentralized self-adaptive systems

Meta-level computations returned control loop computations in self-adaptive systems. Include four steps monitor, analyze, plan, execute.

This article introduces a decentralized self – adaptive system as a reference. (figure 6)

Mete-level model includes system model consists system model (part of system that managed by the self-adaptive unit), concern model (objectives of a self-adaptive unit),working model(data structures and information shared),coordination model (coordinate data between meta-level computation of self-adaptive units).

3-3-Rainbow: Architecture-Based Self-Adaptation With Reusable Infrastructure

Rainbow is a framework that for support self-adaption uses two Items 1- SoftWare Architecture 2 – reusable infrastructure[5].

And too It use of external adaptation mechanism. As illustrated figure 7.

RainBow have a framework that is an abstract model to perform monitor, evaluate model and Adaptation when is need and there is problem in running system.

Software Architecture prepare a global perspective of the system and represent important Behaviors and properties of system different levels.

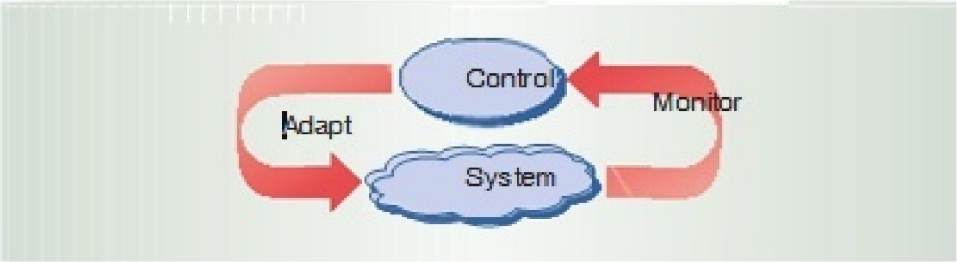

Figure 7. External control of self-adaptation

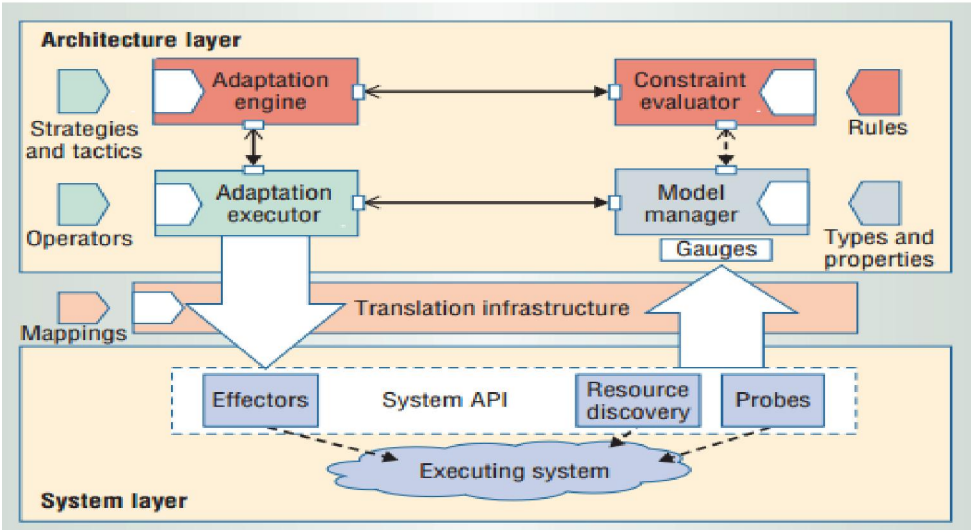

Figure 8. Rainbow framework

Figure 8 show rainbow framework.

Rainbow have reusable Units, different component reusable from a system to another systems framework divided two part 1-Adoptation Infastructure 2-system-specific adaptation knowledge, also adaptation Knowledge include architecture and translation layers.

Rainbow Included Layers system-Layer Infrastructure, architecture-Layer Infrastructure, Translation Infrastructure, system-specific adaption knowledge.

system-layer Infrastructure describe system access interface then make an infrastructure that begin implement it system modifications transfer by Effector mechanism to actual system.

architecture-Layer Infrastructure perform aggregate information and update properties in the architecture model. periodically model and events checked, finally perform necessary adaptation.

Translation Infrastructure map information of system to model and maintain various mappings. system-specific adaption knowledge includes parameters, component types, properties, behavioral contraints and adaptation strategies. these help to perform system require adaptation.

Architectural style in RainBow have four sets of entities 1-component and connector types (a vocabulary of elements) 2- constraints 3- properties (attributes of the component and connector types) 4- analyses (make an appropriate architectural style).

3-4- RA4SaS

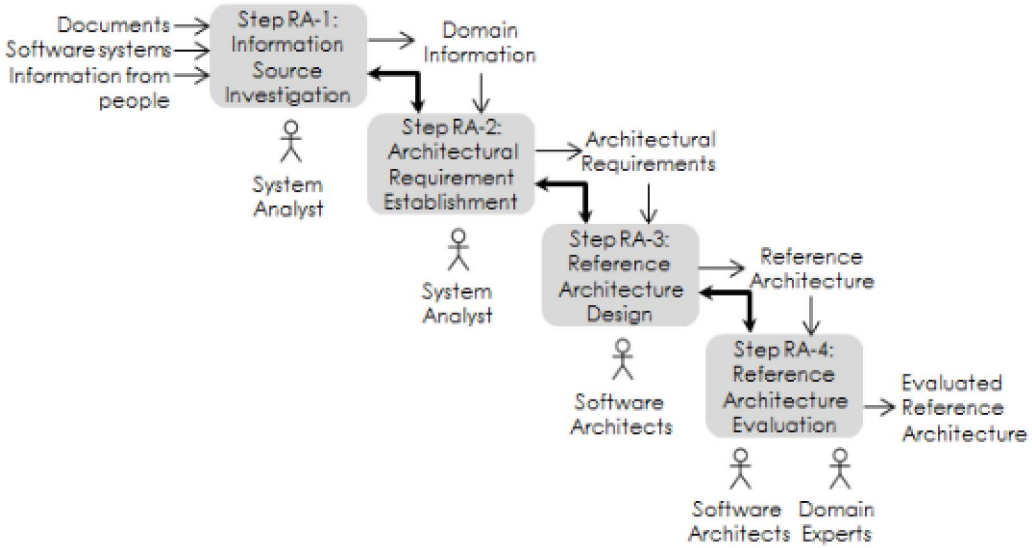

Figure 9. Outline Structure of ProSA-RA

RA4SaS support adaptation system at running. This architecture have 4 main purpose 1-for new system have design of concrete architecture 2-focus in extensions of systems 3- evaluation of systems that use of reference architecture 4-improvement in the standardization and interoperability. RA4SaS done Software Monitoring and Software Adaptation without shareholders invention. RA4SaS proposed an architectural model two level (meta and base) that two level have reflective characteristics and explore reflection for sas development [6].

Table 1. THE MODEL 5W1H AND ADAPTATION SCOPE

Questions	Scopes
What will be adapted?	Attributes, software entities, architectural components, etc.
Where will the adaptation occur?	Software system, software architecture, ar- chitectural components, etc
Who will perform the adaptation?	Autonomous systems, humans, both, etc
When will the adaptation be applied?	How often, anytime, constantly, etc
Why will it be adapted?	Errors in project, new requirements (users or technology), etc
How will it be adapted?	Is there any action plan for adaptation? Was the adaptation planned?, etc

Table 2. SELF-* PROPERTIES VS QUALITY FACTORS

Primitive term	Synonyms	Quality factor
Self-configuring	no	Functionality Integrity Maintainability Portability Scalability usability
Self-healing	Self-diagnosing self-reparing	Availability Integrity Maintainability Reliability survivability
self-optimizing	self-adjusting self-tunning	Efficiency Functionality Integrity Maintainability Performance Troughput
Self-protecting	Self-defense	Availability Confidentiality Functionality Integrity Reliability

A refrence architectures named Prosa-Ra (process based on software architecture–reference architecture) illustrated in figure 9.

Step1 named Information Source Investigation. Inputs includes Documents, Software Systems and Information from people.

In this level all information and resources collected as a specified questions in order to find out about architectural requirements.

Step 1 show questions for adaptation and scopes of action. (Table 1).

Table show terms and quality factors. (Table 2)

Only these four characteristics have been studied and other characteristics have not been considered.

Choice a refrence architecture for Self-Adaptive system that show dynamic behavior and refrence models in a set. It included bloch diagrams (BD: a diagram the represented relationship of the blochs), Formal Methods (FM: process that uses formal methods).

Formal Notation for Business Process (FN: show a business process in business process model), Informal Notation (IN: a type of special notation), Layer Diagrams (LD: represented organization physical artifacts) and UML diagrams (UD: a

graphical language) also specifies Knowledge/Element for design reference model and Refrence Architecture.

It included Action Plane (an actions sequence to execute a specific activity), Agents (types agent that show dynamic behavior), Autonomous SubSystems (implement architecture model of autonomic computing), computational Reflection (performed any activity by a system on itself), Nature-inspired service ecosystems (get inspiration from natural systems for modeling and deployment of services), Process flow (show step sequence an activity), Rulbase (a set of rules to represent a dynamic behavior of software), service composition (new new functionlities at runtime), Sub System in Layers (a set of subsystems that could performed one or more activity), Supervisor System (responsible for monitoring the operation of another system).

Figure 10 show relation between 6 catrgories and type of knowledge.

Table 3. PART OF THE RA4SAS REQUIREMENT)

ID	Requirement	Concept
RS-1	The reference architecture must enable the development of SaS that has a mechanism to define the adaptation level of software systems.	Ds
RS-2	The reference architecture must allow the development of SaS that are able to reflect on current execution state, identifying their structure and behaviors.	Ds
RS-3	The reference architecture must enable the development of SaS that are able to keep their representation after performed an adaptation activity.	RS
RS-4	The reference architecture must allow the development of SaS that are able to adapt by modifying their structure and behavior	DS
RI-1	The reference architecture must enable the development of SaS that are able to replace the software system (software entity) at runtime, which represents the capacity to perform dynamic compilation and dynamic loading of software entities without restarting the application or redeploying its components.	IS-CD
RI-2	The reference architecture must allow the development of SaS that are able to control the adaptation number so that do not quickly increase in size, making unfeasible for future adaptation.	IS-NA
RI-3	The reference architecture must enable the development of SaS that are able to establish an action plan for software adaptation, which represents a sequence of steps so that the system software is adapted at runtime.	IS-AP
RI-4	The reference architecture must allow the development of SaS that are able to identify and report (diagnosis) problems occurring when an adaptation activity is being performed	IS
RI-5	The reference architecture must enable the development of SaS that are able to fix problems (healing) at runtime when an adaptation activity is being performed.	IS
RI-6	The reference architecture must allow the development of SaS that has a mechanism able to preserve its execution state (current information) when a software entity is being adapted	IS-RE

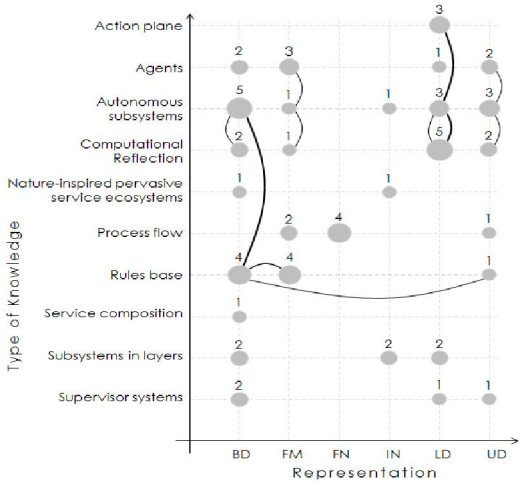

Figure 10. Representation & Type of knowledge

Step 2 established a set of architectural requirements for RA4SaS.

Included Architectural requirements of Sas domain and architectural requirements related to infrastructure of SaS. Part of the RA4SaS requirements show in Table 3.

First column show requirement identification. R-S is abbreviation requirement related to self-adaptive software domain. R-I is abbreviation requirement related to infrastructure of adaptation. second column is requirement description. Third column is concepts related requirements. DS is abbreviation development of SaS, RS is abbreviation representation of SaS, IS is abbreviation infrastructure of SaS, DC is abbreviation dynamic compilation, NA is abbreviation number of adaptations, AP is abbreviation action plan, RE is

abbreviation restoring execution.

Step 3 represent a Architectural design composed a core for adaption (a set of modules that is responsible for managing software entity adoption at runtime) and Development module (provides a set of guidelines for sas development. this guidelines consist requirements analysys, design, implementation and evolution), action plan module (aims at assisting in the adoption activity of software entities), adaptation rules module (Responsible for automatically extracting adaptation rule of software entity) and Infrastructure module (provide support for software entity adoption at runtime).

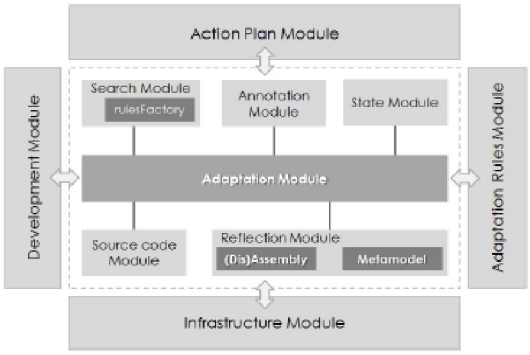

Figure 11. General representation of RA4SaS

Step 4 is a reference for evaluation architecture and improving the quality of RA4SaS.

3-5-- Self-adaptation of mobile system driven

DSPL (Dynamic Software Product Line) manage the runtime variability of applications[7].

For availability of resources could decide best functionality from architectural configuration and fast algorithms.

DSPL is a single system that adopt system behavior at runtime. Combine with automatic computing (AC) to development of adaptation in software systems.

Variation points modeled as part of the software architecture that may change at runtime.

The runtime environment monitored to change reflect in dynamic variation points.

System need analysed to defined new changes or configuration then generate them at runtime and modified architectural vartion points.

Now describe three cases SPL, CVL, GA

SPL is a software-intensive systems. DSPL use of SPL to perform adoptation and move system to a state safe and valid.

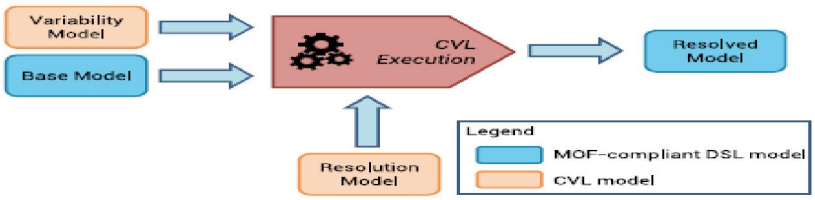

Figure 12. CVL approach

A language model choice to model the system variability at different levels is common variability language (CVL) because CVL is am OF-based variability language and standard (submitted to the OMG) State proposed by CVL show in figure 12.

CVL is a domain-independent language that use a MOF-based meta model to specified variable over any model.CVL Execution required variability model and base model.

Variability model includes 1-variation points (variable points and modify of base model).

2- variability specification tree (show tree structure of elements) OCL Constrains (describe constraint between elements of vspec tree) Genetic Algorithms: GAs are a search heuristic that could find solutions for optimizations problems and generates a application configuration at run-time.

This approach propose a middleware that consists Context Monitoring service (monitoring environment and collect information) And Dynamic Reconfiguration Service (analysis of information and execution of the reconfiguration plans).

These perform covering the steps of the MAPE-K loop. Main approach show in figure 13.

Knowledge represented dynamic variation points (points of the base model), vspec Tree (tree structure the elements), OCL Constraints, software architecture, resource &utility information, reconfiguration policy.

Monitor phase provides with CMS, DRS information for evolution resources. Analysis phase Received information and perform analyze for adaption process if change is suitable. Defined several reconfiguration policies as knowledge base.

Plan phase analyzer perform *adopt* for application. Then DAGAME algorithm (this algorithm find a nearly –optimal configuration) and genetic algorithm (find nearly –optimal solution for optimization problem) is executed. Uses VSPECS Tree, context information, else to Generated new plan.

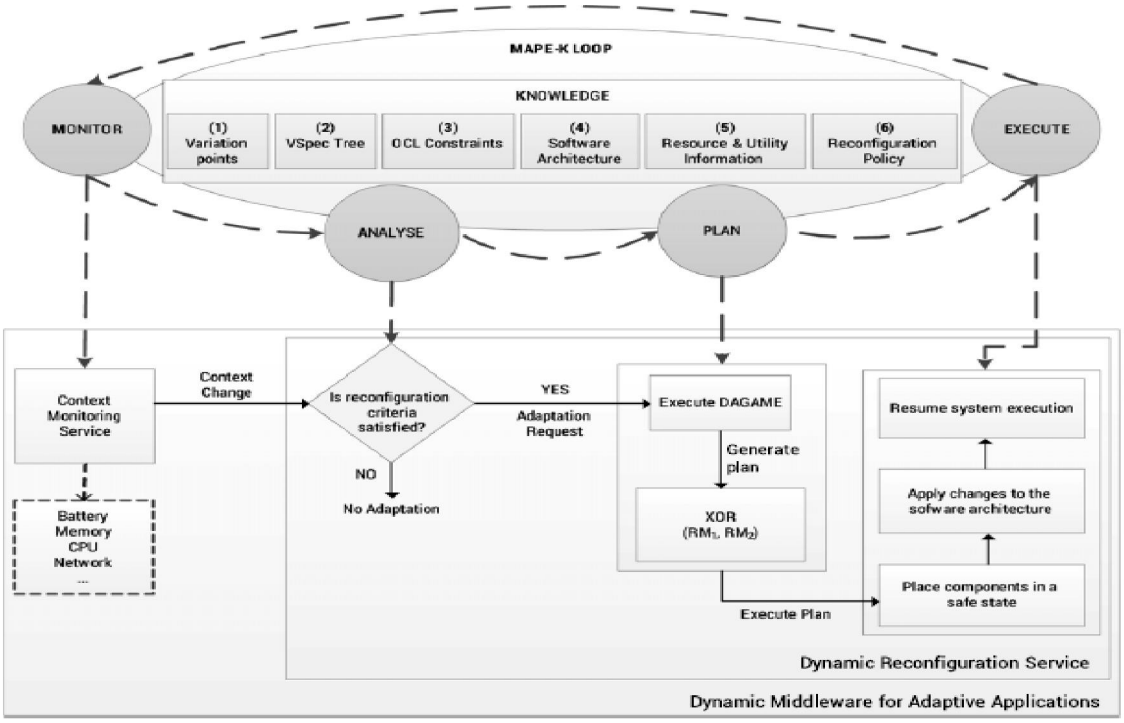

Figure 13. Approach Self-adaptation of mobile system driven

In execute phase reconfiguration plan is executed, add new components and connection and perform compare between configurations.

Finally perform an adopt the running architecture of the application.

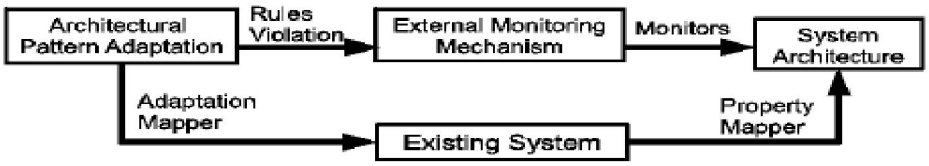

Figure 14. Adaptive Framework

3-6-self adaptation by integrating adaptive framework with architectural patterns

This approach integrate the adaptive framework with Architectural patterns for perform dynamic self-adaptation in software systems [8].

An adaptive framework include monitoring (External and Internal), decision making and reconfiguration. this framework uses external monitoring (rules described in mechanism) and control mechanism.

Properties a system at running write and updated in ADL (a language for describing software architecture) then monitored (uses rules in external monitoring mechanism) and analyzed and build a reconfiguration the architecture of the system with architectural pattern (say a fundamental structural organization schema for software system and are the highest level patterns).

Finally new configuration is returned to the running system.

This way is an approach for dynamic self-adaption at running system. Adaptive framework show in figure 14.

When a rule in system violated. system required to new decision and reconfiguration.

This reconfiguration build on the architectural patterns then reflect to the running system.

4-Discussion

The following tables shows all the important factors of each project considered and compared with each other.

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
Ecological Architecture	✓			✓										

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
on decentralized self-adaptation				✓	✓			✓		✓		✓		

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
RainBow Architecture	✓					✓	✓							

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
RA4SaS	✓					✓			✓	✓	✓		✓	✓

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
self-Adaptaion of mobile system driven	✓							✓					✓	

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
integrity adaptive framework with architectural patterns	✓	✓	✓	✓			✓						✓	

Factor plan	self-managing	Self- maintenance	self-control	self-organization	self-configuration	self-healing	Self-repairing	self-optimization	self-tunning	self-protection	Self-Awareness	self-contained	self-Monitoring	self-Suited
Ecological Architecture	✓			✓										
decentralized self-adaptation				✓	✓			✓		✓		✓		
Rainbow Architecture	✓					✓	✓							
RA4SaS	✓					✓			✓	✓	✓		✓	✓
self-Adaptation of mobile system driven	✓							✓					✓	
integration adaptive framework with architectural patterns	✓	✓	✓	✓			✓						✓	

5- Conclusion

Systems and application programs and services that requires running all the time.

These systems should be able to recognize changes and errors in the environment and then perform analysis, planning and monitoring to solve failure in systems. A huge benefit of self-adaptive systems is their ability to reduce dependency to human resource and recover different events quickly.

Because of the complexity of software systems task for managing them need to architecture with determined framework.

Each one of these plans, attempts to improve and provide the framework of self-adaptive software systems.

All these plans and proposal in the future should present optimal solutions for problems and fault to be able to cover quality-attributes and evaluation of each of these plans should based on mention attributes.

Self-adaptive system should designed in such away be able support properties such as validation, productivity, reliability, availability, maintainability, dependability, performability and also share information, reduce cost, agile, reusability, Rapid responsivness to environmental changes, amend fault, error and failure, recognize events and them reason, interact components and devices, fault-tolerance, solution deployment, utilization, resource, throught put, safety, security management, quality of service, reporting total, detail in all levels and distributed environments.

References

1. Lanyon-Hogg, Richard, Devaprasad K. Nadgir, and Amr F. Yassin. "A practical guide to the IBM autonomic computing toolkit." (2004).
2. Baresi, Luciano, and Sam Guinea. "Architectural Styles for Adaptive Systems: A Tutorial."*Self-Adaptive and Self-Organizing Systems (SASO)", 2012 IEEE Sixth International Conference on*. IEEE, 2012.

3. Zambonelli, Franco. "Self-management and the many facets of nonself." *IEEE Intelligent Systems* 21.2 (2006): 50-58.
4. Weyns, Danny, Sam Malek, and Jesper Andersson. "On decentralized self-adaptation: lessons from the trenches and challenges for the future." *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2010.
5. Garlan, David, et al. "Rainbow: Architecture-based self-adaptation with reusable infrastructure." *Computer* 37.10 (2004): 46-54.
6. Affonso, Frank José, and Elisa Yumi Nakagawa. "A Reference Architecture Based on Reflection for Self-Adaptive Software." *Software Components, Architectures and Reuse (SBCARS), 2013 VII Brazilian Symposium on*. IEEE, 2013.
7. Pascual, Gustavo G., Mónica Pinto, and Lidia Fuentes. "Self-adaptation of mobile systems driven by the Common Variability Language." *Future Generation Computer Systems* (2014).
8. Krishnamurthy, Vallidevi, and Chitra Babu. "Effective self adaptation by integrating adaptive framework with architectural patterns." *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*. ACM, 2010.
9. Oreizy, Peyman, et al. "An architecture-based approach to self-adaptive software." *IEEE Intelligent systems* 14.3 (1999): 54-62.
10. Wang, Qianxiang. "Towards a rule model for self-adaptive software." *ACM SIGSOFT Software Engineering Notes* 30.1 (2005): 8.
11. Oreizy, Peyman, Nenad Medvidovic, and Richard N. Taylor. "Runtime software adaptation: framework, approaches, and styles." *Companion of the 30th international conference on Software engineering*. ACM, 2008.
12. Liao, Xiaofeng, Shiyue Lai, and Qing Zhou. "A novel image encryption algorithm based on self-adaptive wave transmission." *Signal Processing* 90.9 (2010): 2714-2722.
13. Brest, Janez, et al. "Performance comparison of self-adaptive and adaptive differential evolution algorithms." *Soft Computing* 11.7 (2007): 617-629.
14. Perez-Palacin, Diego, and José Merseguer. "Performance sensitive self-adaptive service-oriented software using hidden markov models." *ACM SIGSOFT Software Engineering Notes*. Vol. 36. No. 5. ACM, 2011.
15. Perez-Palacin, Diego, et al. "Qos-based model driven assessment of adaptive reactive systems." *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*. IEEE, 2010.
16. Perez-Palacin, Diego, Raffaela Mirandola, and José Merseguer. "Software architecture adaptability metrics for QoS-based self-adaptation." *Proceedings of the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on Quality of software architectures--QoSA and architecting critical systems--ISARCS*. ACM, 2011.
17. Mejias, Boris, and Peter Van Roy. "From mini-clouds to Cloud Computing." *Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on*. IEEE, 2010.
18. Villegas, Norha M., et al. "A framework for evaluating quality-driven self-adaptive software systems." *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*. ACM, 2011.
19. Chaudhuri, Surajit, and Vivek Narasayya. "Self-tuning database systems: a decade of progress." *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007.
20. Stanfel, Zeljko, et al. "A self manageable rule driven enterprise application." *29th International Conference on Information Technology Interfaces (ITI 2007)*. 2007.
21. Miede, André, et al. "A Comparison of Self-Organization Mechanisms in Nature and Information Technology." (2009).
22. Di Marzo Serugendo, Giovanna, John Fitzgerald, and Alexander Romanovsky. "MetaSelf: an architecture and a development method for dependable self-* systems." *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010.
23. Prokopenko, Mikhail. "Design vs. Self-organization." *Advances in applied self-organizing systems*. Springer London, 2008. 3-17.
24. Weyns, Danny, and Michael Georgeff. "Self-adaptation using multiagent systems." *Software, IEEE* 27.1 (2010): 86-91.
25. Rodríguez-Fernández, Carlos, and Jorge Jesús Gómez-Sanz. "Self-management capability requirements with SelfMML & INGENIAS to attain self-organising behaviours." *Proceedings of the second international workshop on Self-organizing architectures*. ACM, 2010.
26. Jamont, Jean-Paul, Clément Raievsky, and Michel Occello. "Handling Safety-Related Non-Functional Requirements in Embedded Multi-Agent System Design." *Advances in Practical*

Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. Springer International Publishing, 2014. 159-170.
27. Taranu, Stefan, and Jens Tiemann. "On assessing self-adaptive systems." *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010.
28. Rattani, Ajita, Gian Luca Marcialis, and Fabio Roli. "Self adaptive systems: An experimental analysis of the performance over time." *Computational Intelligence in Biometrics and Identity Management (CIBIM), 2011 IEEE Workshop on*. IEEE, 2011.
29. Ferrante, Alberto, et al. "Self-adaptive Security at Application Level: a Proposal." *ReCoSoC*. 2007.
30. Ferrante, Alberto, et al. "Self-adaptive Security at Application Level: a Proposal." *ReCoSoC*. 2007.
31. Salehie, Mazeiar, and Ladan Tahvildari. "Self-adaptive software: Landscape and research challenges." *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 4.2 (2009): 14.
32. Ramirez, Andres J., and Betty HC Cheng. "Design patterns for developing dynamically adaptive systems." *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2010.
33. Al-Shishtawy, Ahmad, et al. "A design methodology for self-management in distributed environments." *Computational Science and Engineering, 2009. CSE'09. International Conference on*. Vol. 1. IEEE, 2009.
34. Villegas, Norha M., et al. "A framework for evaluating quality-driven self-adaptive software systems." *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*. ACM, 2011.
35. Balasubramanian, Sowmya, et al. "Characterizing problems for realizing policies in self-adaptive and self-managing systems." *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2011.
36. Weyns, Danny, Sam Malek, and Jesper Andersson. "FORMS: a formal reference model for self-adaptation." *Proceedings of the 7th international conference on Autonomic computing*. ACM, 2010.
37. Cheng, Betty HC, et al. "Software engineering for self-adaptive systems: A research roadmap." *Software engineering for self-adaptive systems*. Springer Berlin Heidelberg, 2009. 1-26.
38. Weyns, Danny, and Jesper Andersson. "On the challenges of self-adaptation in systems of systems." *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems*. ACM, 2013.
39. Cheng, Shang-Wen, and David Garlan. "Stitch: A language for architecture-based self-adaptation." *Journal of Systems and Software* 85.12 (2012): 2860-2875.
40. Said, Mouna Ben, et al. "Design patterns for self-adaptive RTE systems specification."
41. Jha, Shantenu, Manish Parashar, and Omer Rana. "Self-adaptive architectures for autonomic computational science." *Self-Organizing Architectures*. Springer Berlin Heidelberg, 2010. 177-197.
42. Weyns, Danny, et al. "SOAR: Self-Organizing Architectures." *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009)*. 2009.
43. Weyns, Danny, Sam Malek, and Jesper Andersson. "FORMS: Unifying reference model for formal specification of distributed self-adaptive systems." *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7.1 (2012): 8.

8/31/2015