paper 6

by Paper Aradea

Variety of Approaches in Self-adaptation Requirements: A Case Study

Aradea[1](✉), Iping Supriana[2], Kridanto Surendro[2], and Irfan Darmawan[3]

[1] Department of Informatics Engineering, Faculty of Engineering, Siliwangi University, Tasikmalaya, Indonesia
aradea.informatika@gmail.com

[2] School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia

[3] Department of Information System, School of Industrial and System Engineering, Telkom University, Bandung, Indonesia

Abstract. Self-adaptation requirements are requirements engineering studies to develop self-adaptive systems. This approach provides a way how activity at design-time requirements to meet stakeholder needs and system-to-be. Currently, there is a variety of approaches were proposed to the researchers through the development of goal-oriented requirements engineering. The ideas expressed through the expansion of this model into a way that is quite promising, however the various approaches proposed, does not mean no shortage. This paper describes in detail the variety of approaches available today through the implementation of a case study, and analysis of the results, we found 5 main features that can be used as consideration in formulating self-adaptation requirements, namely goal concept, environment model, behavior analysis, run-time dependencies, and adaptation strategy. Besides that, we saw of future research chance through deep study at goal-based modeling and loop feedback with utilizing data mining technique.

Keywords: Self-adaptive systems · Adaptation requirements · Goal oriented requirements engineering

1 Introduction

Requirements engineering approach used for the development of self-adaptive systems (SAS), has the form of a different approach to the traditional requirements engineering, which only represents the understanding of the problem domain requirements at design-time. In SAS, attention to changes in requirements that may occur at run-time, a problem that must be anticipated and determined handling solutions. In general, the model of goal-oriented requirements engineering (GORE) widely adopted as the basic concept to develop an alternative solution to the issue. This model was expanded through the establishment of requirement specifications are prepared to requirements at design-time and run-time. Design-time specification is realized through various proposals to create monitoring mechanisms and adaptation, while the run-time

© Springer International Publishing AG 2017
T. Herawan et al. (eds.), *Recent Advances on Soft Computing and Data Mining*,
Advances in Intelligent Systems and Computing 549, DOI 10.1007/978-3-319-51281-5_26

specification is represented through the implementation of the mechanism of dynamic systems, for example through reconfiguration and evolution solutions. Survey and preliminary study of the research topic have been discussed in our previous paper [1], and we conclude importance of feature specification determining in more detail from both those design-time and run-time specification, to gain pattern more clearly in formulating self-adaptation requirements.

In the discussion of this paper, we develop a case study approach is applied to the four previous researchers then evaluated to identify opportunities perfected. The main objective of this paper is to focus on the discussion of the stages of modeling requirements, and the proposed extension of the findings of the case study discussion. Discussion begins with a case study (Sect. 2), four approaches adaptation requirements (Sect. 3), proposed the expansion and future work (Sect. 4), and conclusion (Sect. 5).

2 Case Study: Goal Model

The case studies developed as an illustration of the adaptation needs of the lecture at the college. Where there are two actors of the system (the classroom system and the majors system) representing the needs of lecturer, students, and majors. The system must be able to ensure that the activities of the lecture can be held as scheduled, the number of sessions to be in the range of 14 sessions – 16 sessions. If the range is not met, then the replacement schedule should be established, it relates to the willingness of lecturer time and space.

Figure 1 illustrates the modeling of the case through Tropos models [2], in which the actor classroom system has a major goal “lecture monitoring” which can be achieved through one of its sub-goal. Goal “organizing the lecture” can be achieved if the activity of lectures “start on scheduled”, “completed on schedule” as well as the lecturer and students “filling of attendance”, this may be a achieved through the plan “monitoring” and “check the numbers of lectures”, which can contribute to positive/full satisfaction (++) against soft goal “availability of classroom” and “lecture targets”. While the plan for achieving the goal “cancelation of lecture” has contributed partial negative (-) against both soft goal, considering the cancelation of the lecture could adversely affect the achievement of the lecture and meeting classroom availability, but will not influence negatively if the determination of the replacement schedule can be achieved. Meanwhile, if the goal of “organizing the lecture” is not achieved, then the scenario that was developed was a “determining a replacement schedule” by delegating goal “choose schedule” to the majors system actor. This goal can be achieved by doing “collect time-lecturer”, “find free classroom”, “choose the schedule”, and “confirmation message”.

3 Adaptation Requirements Modeling

In this part of the case study mapped the standpoint of adaptation requirements, based on the four works most closely with the research that is being done.

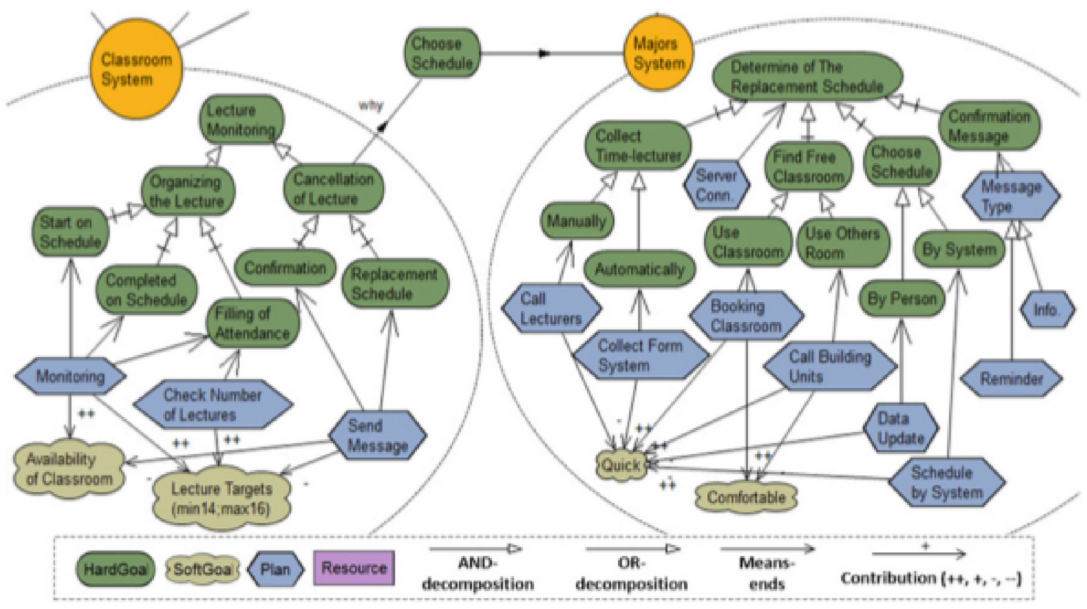

Fig. 1. Modeling of system monitoring and scheduling college tuition replacement.

3.1 Tropos for Adaptive Systems (TROPOS4AS)

Tropos4AS proposed by Morandini et al. [3, 4], the approach is to expand the model Tropos [2] by introducing (a) the model type of goal and conditions of satisfaction, (b) the model environment that will affect the satisfaction of a goal that must be monitored, (c) a model of failure through the selection of alternative behavior when the goal should be achieved not met. Figure 2 shows the modeling of lecture monitoring. Satisfaction goal adapted into three types of goal, namely (a) achievement (A), namely, when certain circumstances is reached or reached the state at a time, (b) maintain (M) is the need to maintain a specific state in a period of time or the state from time to time, (c) perform (P) is successfully implementing an activity or one of the model (plan or subgoal). Meanwhile, to represent the run-time dependencies between goals, realized through, (a) <<sequence>>, for example, the goal "began on schedule" is reached before activating the goal "is completed on schedule", (b) <<inhibits>>, for example goal "completed on schedule" cannot be active as long as the goal "fill out absentee news show" active.

Environmental models represented into a UML class, and can provide functionality to feel and act as an evaluation of a condition, for example, associated resources (database/schedule), or system device used (sensors in the classroom). Each goal and plan to connect into artifacts environment through conditions of state transition, for example (a) precondition, namely the goal or plan can only be activated if true, (b) Creation Condition, which was to determine the criteria to activate the goal or starting the process of satisfaction goals, please note that the sub-goal activated implicitly through decomposition, such as "creation-c: 09:00 AM", "creation-c: 11:00 AM", "creation-c: max 15 min before the end", (c) Achieve Condition, which determines when the goal is reached so that these goals will be dropped, like "achieve-c: course achieved", (d) Failure Condition, which shows the situation where it is impossible to achieve a goal, such as "failure-c: > 10 min of activation". In the

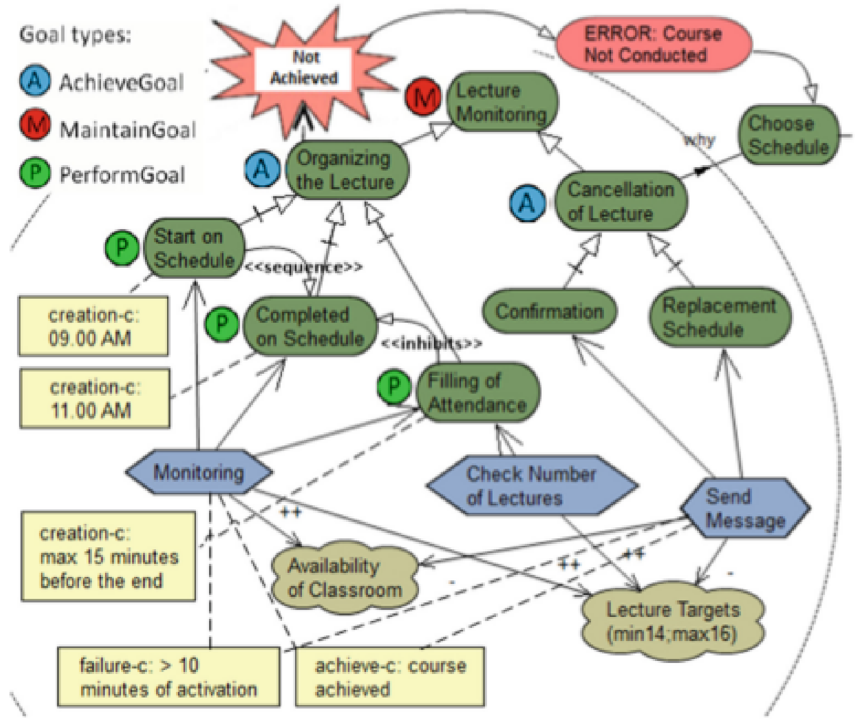

Fig. 2. Modeling of Tropos4AS in lecture monitoring.

scenario developed, if such failure occurs, the system will change the plan to give a warning to the lecturer, and if the goal remains “not achieved”, it will set a replacement schedule, by delegating goal “choose the schedule” to the major actor.

3.2 Adaptive STSs (Socio-Technical Systems)

This work was developed by Dalpiaz et al. [5], which defines the need for adaptation of a socio-technical system (STS). This model proposes architecture based approaches to requirements so that the STS becomes a self-reconfigured. Architecture that was developed to expand the model Tropos [2], the concept utilizes UML 2.0 component diagrams to show the architectural components and connections, which is determined by the cycle MDRC (monitoring, diagnosis, reconcile, compensation). Model goals and requirements of stakeholders to express dependencies actor and multi-agent plan to meet the requirements, while the BDI paradigm to guide diagnosis and selection plan for each actor, and a compensation mechanism exploited to handle failure. Interaction among actors is supported by elements of context sensors, agent, and context actuators.

The behavior of the model in Fig. 3. using the goal models, context models, plan specifications, and domain assumptions (core ontology), which together capture system requirements. Model expressed for use at run-time, through (a) Context, a partial state which is relevant to the status and intention actor, which connects the context of variation points, such as OR-decomposition to G1 including contextual decomposition link for G2, the achievement of these goals is necessary to achieve G1 only if context C1 implemented. While the G3 become a legitimate alternative, only if C2 implemented.

In addition C3, C4 and C5, a context that applies to AND-decomposition to G2. (b) Activation consists of triggering event rules and precondition. A goal that is activated when a trigger event occurs, and the precondition implemented, for example, activation rules for top-level goals include G3 class system is activated as send messages automatically, and G4 delegated to the actor of majors if C2 implemented. While the G5, G6, G7 activated as the send a warning message before the specified time or exceeds the time. (c) Declarative goals are goals that are met only if the achievement of the conditions is met, the satisfaction of these goals independent of satisfaction sub-goal or plan that is connected by means-end decomposition. Achievement of the above conditions is expressed as state context models, such conditions are likely to G4 achievement is to determine the replacement schedule. This goal declarative face of uncertainty, for example, the collection goal lecturer time and find an empty room that had an OR-decomposition. (d) Time limits define the maximum amount of time that an agent can achieve a goal (goal timeouts), for example, G3 must be reached within 60 min prior to the time schedule of lectures, while the G5, G6, G7 must be achieved no more than 10 min from the time schedule. (e) The plan is a collection of actions connected with the goals that were executed by agents, each action performed correctly if post condition achieved within a certain time limit, and at that time precondition done, if not, then the plan includes failed.

3.3 Adaptive Requirements Modeling Language (ARML)

ARML approach [6, 7] model each goal and plan to use a domain ontology, through inference rule are representing. The models were developed using the Techno language through the addition of a new concept (context and resources), with two rules of

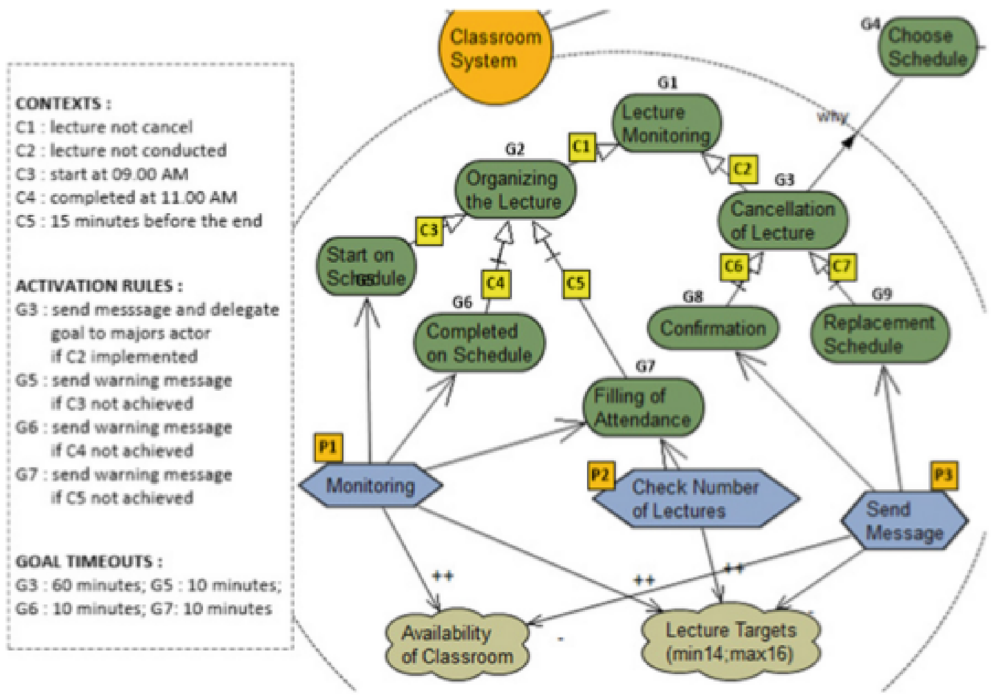
CONTEXTS :
C1 : lecture not cancel
C2 : lecture not conducted
C3 : start at 09.00 AM
C4 : completed at 11.00 AM
C5 : 15 minutes before the end
ACTIVATION RULES :
G3 : send messsage and delegate goal to majors actor if C2 implemented
G5 : send warning message if C3 not achieved
G6 : send warning message if C4 not achieved
G7 : send warning message if C5 not achieved
GOAL TIMEOUTS :
G3 : 60 minutes; G5 : 10 minutes;
G6 : 10 minutes; G7: 10 minutes
Classroom System
G4 Choose Schedule
G1 Lecture Monitoring
why
C1
C2
G2 Organizing the Lecture
G3 Cancellation of Lecture
C3
Start on Schedule
C4
C5
C6
C7
G6 Completed on Schedule
G8 Confirmation
G9 Replacement Schedule
G7 Filling of Attendance
P1 Monitoring
P2 Check Number of Lectures
P3 Send Message
++
++
++
Availability of Classroom
Lecture Targets (min14;max16)

Fig. 3. Modeling of Adaptive STSs in lecture monitoring

relations, namely relegation and influences. In addition, a requirement can be mandatory (M) or optional (O) through inference relations and conflict, to represent how the element of satisfaction can affect the satisfaction of others.

In Fig. 4, the goal "determine of the replacement schedule" is modeled as a mandatory nodes (M node, an unary relation), decomposed through inference relations to mandatory goal, namely the collect time lecturers, find free classroom, choose schedule (black I node, binary relation) as the fact that the goal of "determining of the replacement schedule" will be met through mutual satisfaction of the three goals. Quality constraints (done if the number of lecture < 14 sessions) is placed in inference relation. Influence relations between the three decomposition occurs goal, to describe the context of the prevailing conditions and the availability of resources that may affect the achievement of the "choose schedule" goal.

Requirements analysis for the "manually" goal, connected through inference node (I) which is decomposed into several sub-task as a candidate solution. Each candidate's solutions include mobile phone resource and the domain assumptions (lecturers have a mobile phone and a laptop), connected by symbols preferences are used to compare the requirements in the candidate solutions, for example, send SMS preferable to send the email. While other issues, for instance, related to the lecturer does not have access so did not get the message, then this may be identified through the relegation relation, this

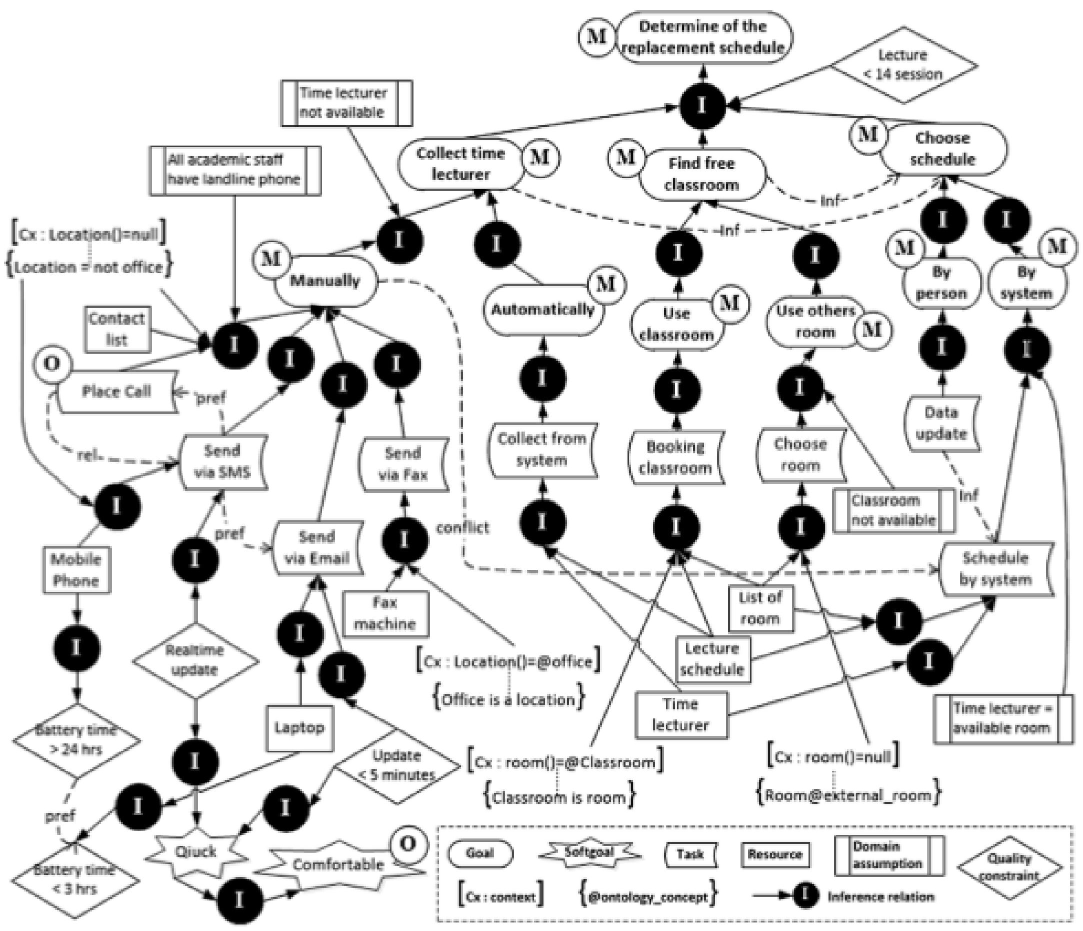

Fig. 4. Modeling of ARML in college scheduling of a replacement

condition makes it possible to take into account situations where a lecturer context changed, so did not get the message. In determining the choice of solution, this can be done through the plan “place call” with the domain assumptions e.g. academic staff have a land-line phone and a contact list resource. Here, the candidate solution is rated and evaluated by quality constraints.

3.4 ZANSHIN Framework

Zanshin framework [8, 9] consists of two approaches, namely awareness requirements (AwReqs) and evolution requirements (EvoReqs). AwReqs an indicator in the convergence of run-time requirements, or constraint for another state requirements (goal elements). While EvoReqs evolve the model automatically, in response to AwReqs failure to determine the change requirements when certain conditions apply. Modeling begins by identifying AwReqs, through the perspective that the system is able to adapt and the importance of determining the level of failure can be tolerated. In Fig. 5, there are seven AwReqs (AwReqs-1 to AwReqs-7). AwReqs also can represent trends success rate requirements, for example, AwReqs-7 empty classroom should not always be available, if at a certain time classroom situation really busy. So that the requirements (for domain assumptions) are not always treated as invariants that must always be accomplished (or should always true). It means that the system may fail to achieve one of the initial requirements (or assumptions become incompatible). Therefore, through a feedback loop provided a way to determine the level of each critical requirements, and monitor the system so aware of the failure. After AwReqs obtained, identification parameters (variation points (VP) and the control variable (CV)), which in the event of a change can have an impact on the indicator (AwReqs) relevant.

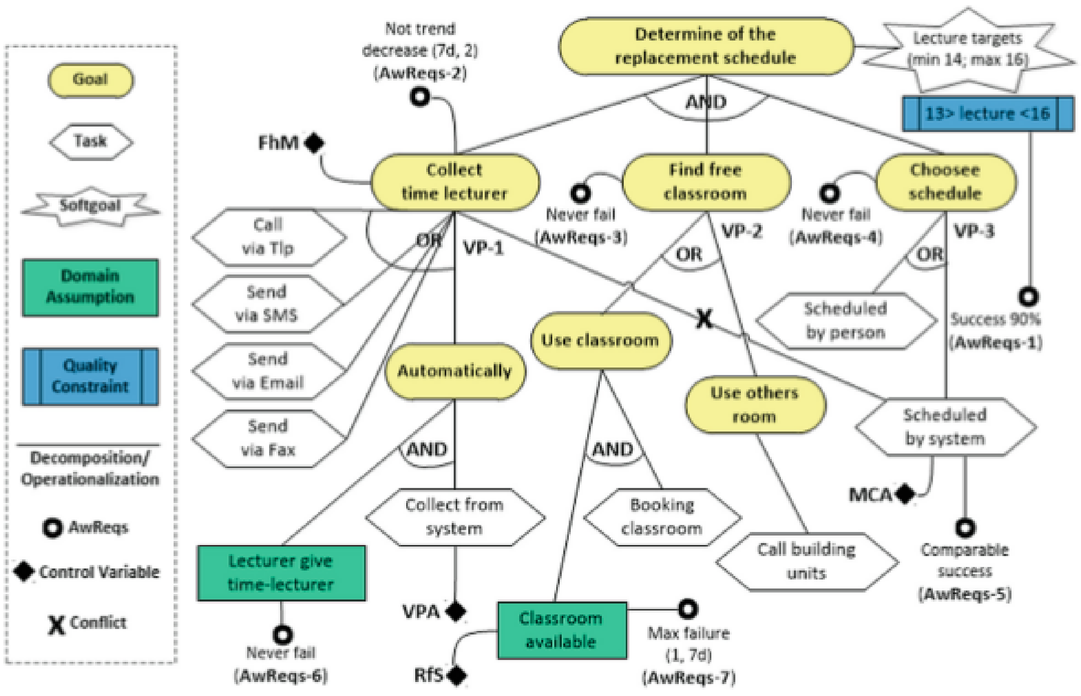

Fig. 5. Modeling of AwReqs. and EvoReqs. in college scheduling of a replacement.

In the case, for example if there is a negative trend in the success rate of collect time lecturer for 2 weeks (AwReqs-2: 7 days (d), 2), such as the agenda of other work, it can be tolerated at most occurs 3 times per semester, thus it (relaxing) reducing constraints to 3 weeks. So variable FHM (From how Many), determine how much time the willingness of lecturer to be collected, changes in the value of these variables can affect the goal itself and the satisfaction of other requirements became clear. Then if classrooms are not available (AwReqs-7: 1, 7d), the system cannot determine a new room, then it does is increase the RFS (Rooms for schedules). As another example, if we know the domain assumptions willingness time lecturers (AwReqs-6) is invalid, then replace (replace it) with other tasks that would verify the validity of the use of willingness time lecturers, it is associated variables VPA (View Private Appointments) that can be worth a yes or no. MCA (Maximum Conflicts allowed) variable associated with (AweReqs-5: Comparison of success rate), requires the system to choose the date on which the number of scheduling conflicts does not exceed the value specified in this variable.

After identifying the parameters that influence AwReqs, then modeling the impact of the differential relations, for example, Δ(AR8 /RFS) [0, maxRooms] > 0, represents the fact that by increasing the number of classrooms, then the domain assumptions classrooms are available to be fulfilled more often. Figures in brackets indicate the interval in which this relationship applies (maxRooms: the qualitative value that should, and can be replaced by a corresponding amount). Then after doing modeling impact activities for each pair of indicators and parameters individually, activities for the improvement of the impact analysis of the indicator to a specified combination, whether the cumulative and if possible, the effects are sorted from the largest to the smallest.

4 Discussion and Future Work

Based on the description of the four works, we identify the essential elements and main features in Table 1. Two aspects of the basic needs, namely the specification of design-time and run-time, consisting of, (a) goal concept: the requirements specification are represented as models goal with details of the concept, (b) environment model: concept developed to represent property system environment (context) associated with the model goals, (c) behavior analysis: the mechanism applied to determine the choice of system behavior to meet every goal, (d) run-time dependencies: dependency artifacts requirements at design time with the operation of the system during run-time, (e) adaptation strategies: mechanisms for determining the solution space as an alternative solution (reconfiguration), and modification goal to represent space new problems (evolution).

Tropos4AS able to analyze the needs of stakeholders and system-to-be through defining the concept of its goal, so as to represent the requirements and reasoning at run-time. This capability can actually be enhanced through the implementation of a domain ontology, which can help in detailing the behavior of the system, such as in the Adaptive STSs and ARML. Domain ontology management in goal models requires a specific strategy if the model decision in selecting candidates wants more solutions to fulfill the properties of self-adaptive systems. Zanshin offers these capabilities through

Table 1. Comparison of self-adaptation requirements specification.

Model	Design-time			Run-time	
	Goal concept	Environment model	Behavioral analysis	Run-time dependency	Adaptation strategy
Tropos 4AS (2011, 2015)	Goal, softgoal, plan, resources, goal type (achieve, maintain, perform)	UML Class, Condition relation (pre, context, creation, achieve, failure)	Variability design for failures model (goal satisfaction)	Transition rules, run-time goal state (<<sequence>> <<inhibits>>)	Inference rules: L/R [rule-name], elicitation and recovery model for failure
Adaptive STSs (2013)	Goal, softgoal, plan, resources, domain assumptions	Context model (context sensor, agent, context actuator), plan specification	Reconcile and compensation strategy, cost and contribution to soft-goals	Activations rules, context, time limits, declarative goals, plan	MDRC, DLV-complex reasoner, spesific thread for failure model
ARML (2011, 2012)	Goal, task, quality constraints, domain assumptions	Domain ontology, context concept, relation concept, resources	Linking domain ontology to goal model, and ECA rules modeling	Inference relation, conflict, preferences, relegation, influence rels	Run-time requirements artifact, SALmon, CARE App., ECA rules
Zanshin (2012, 2013)	Goal, task, quality constraints, domain assumptions	AwaReqs. in goal models, parameter: variation points, control variable	Diferential relations (Δ (AwReqs-"n"/ CV) > 0), ECA rules modeling	EvoReqs., controller and system target dependency, MAPE loop	Qualitative reasoning, and diferential relations, ECA rules

the centralization of feedback control loop, but related domain model that represents the problem domain as requirements, have not been included. While Tropos4AS have this capability, in addition to the ability of the high variability and the concept of independence, so that integration Tropos4AS and Zanshin to the attention of our work in the future.

Adaptive STSs also proposed a self-adaptation capabilities similar to Zanshin, based on the architecture approach of a model driven requirements, but through a compensation strategy for the behavior of multi-agent systems. While the new Tropos4AS developing behavior based on the single-agent system. Comparison of both the architectural concept is also our concern. When viewed from the run-time requirements specification, Tropos4AS and Adaptive STSs ability to realize the goal through agent executable models, while Zanshin and ARML using ECA rules as primitive operations on the model of the goal. Thus, we assume, design-based independent software agents can exploit human-oriented abstractions such as agent and goals so that construction of this language suitable for representing real-world requirements and reasoning at run-time. While the centralized feedback loop approach can actually be integrated to enhance the adaptability functions at run-time, through the development of data mining algorithm.

5 Conclusion

Self-adaptation requirements is a concept that can address the needs in translating real-world conditions, related to the diversity of the elements involved and the anticipation of amendments. The four works that are discussed in this paper enough to give an idea of the variety offered alternative solutions to achieve self-adaptive requirements capability. Based on these studies, we see two main strengths that are considered to be a great opportunity to leverage the power of self-adaptation requirements, namely (a) the goals oriented approach through Tropos/i * model, can capture and represent the variability in context and behavior of the system (domain models), as the concept of requirements, (b) models of dynamic systems through a feedback loop, can be developed to establish assurance criteria for management system and mechanisms of adaptation, as the concept of self-adaptation. Based on these assumptions, our next job is to formulate a formal framework that can bridge the integration of both approaches, and how the data mining technique could facing uncertain environment at run-time, through determining inference context algorithm based model of sensor and its data history.

References

1. Aradea, D., Supriana, I., Surendro, K.: Roadmap dan area penelitian self-adaptive systems. Prosiding Seminar Nasional Teknik Informatika dan Sistem Informasi (SeTISI), Universitas Maranatha Bandung (2015)
2. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: TROPOS: an agent-oriented software development methodology. J. Auton. Agent. Multi-agent Syst. **8**(3), 203–236 (2004)
3. Morandini, M.: Goal-oriented development of self-adaptive systems. Ph.D. thesis, University of Trento (2011)
4. Morandini, M., Penserini, L., Perini, A., Marchetto, A.: Engineering requirements for adaptive systems. J. Requirements Eng. **21**, 1–27 (2015). Springer
5. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Adaptive socio-technical systems: a requirements-based approach. J. Requirements Eng. **18**(1), 1–24 (2013)
6. Qureshi, N.A.: Requirements engineering for self-adaptive software: bridging the gap between design-time and run-time. Ph.D. thesis, University of Trento (2011)
7. Qureshi, N.A., Jureta, I.J., Perini, A.: Towards a requirements modeling language for self-adaptive systems. In: Regnell, B., Damian, D. (eds.) REFSQ 2012. LNCS, vol. 7195, pp. 263–279. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28714-5_24
8. Souza, V.E.S.: Requirements-based software system adaptation. Ph.D. thesis, University of Trento (2012)
9. Souza, V.E.S., Lapouchnian, A., Angelopoulos, K., Mylopoulos, J.: Requirements-driven software evolution. J. Comput. Sci. Res. Dev. **28**(4), 311–329 (2013). Springer

paper 6

ORIGINALITY REPORT

3%	2%	3%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

1%

★ mygoodplace.ciando.com

Internet Source

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		