Continuous monitoring of adaptive e-learning systems requirements

Lamiae Dounas, Raul Mazo, Camille Salinesi, Omar El Beqqali

Dounas Lamiae, Mazo Raúl, Salinesi Camille, El-Beqqali Omar. Continuous Monitoring of Adaptive e-learning Systems Requirements. 12th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), Marrakech-Morocco, November 17-20, 2015.

Continuous Monitoring of Adaptive e-learning Systems Requirements

Lamiae DOUNAS[1,2], Raul MAZO[1], Camille SALINESI[1], Omar EL BEQQALI[2]
[1]CRI, University Paris 1 Panthéon-Sorbonne, Paris, France
[2] LIAAN, Faculty of Sciences Dhar el Mehraz USMBA, Fez , Morocco
lamiae.dounas@usmba.ac.ma, {raul.mazo, camille.salinesi} @univ-paris1.fr, omar.elbeqqali@usmba.ac.ma

***Abstract*—E-learning is a promising research area, as they are expected to increase enrollment and improve the quality of education. Adaptive e-learning systems, traditionally focused on content personalization, are in need to cope with continuous changing requirements and changing environment. Indeed, the specification and the management quality attributes of such systems, supported throughout the whole lifecycle are still missing. In this paper, we present an approach for continuous requirements monitoring that uses constraint programming to evaluate the satisfaction of the requirements in adaptive e-learning systems and identify solutions (valid configurations) when deviations occur at runtime. To this end, the requirements are conceptualized as a dynamic software product line. A new requirements engineering tool that supports (dynamic) software product line engineering is used for the specification of the adaptive e-learning systems and its continuous monitoring.**

Keywords— adaptive e-learning systems; requirements engineering; requirements monitoring; dynamic software product lines; constraint programming; goal-driven requirements.

I. INTRODUCTION

As advances in digital technologies increase, e-learning domain moves from being a purely "one-fits-all" online content delivery to an adaptive learning that fits a wide range of students and leads companies to bolster the productivity and the skills of their employees through more flexible and adaptive approaches.

The adaptation takes place at different areas of adaptive e-learning systems (AESs), it includes: (1) the presentation by adapting the content or the used media to each learner, and (2) the navigation by providing learners optimal paths for a flexible access to information and orientation supports. In that regard, the adaptation decision making involves the analysis of gathered information from learner interaction like learner's background, cognitive style, personal preferences, and knowledge level. Several techniques like artificial intelligence or semantic web are then used to interpret this information and propose educational resources that comply with learners needs [1-9].

However, while most of the research in the field focuses on the adaptation techniques to improve the learning experience and offer a personalized learning, less attention has been given to the specification of AESs. Indeed, with the complexity of the learning environment in which a multitude of connection and interactions come into play, the design of such systems is still a challenging task. Hence, it is important to support the requirements of such systems and allow their evolution at runtime, especially in dynamic changing environmental conditions (e.g., changing learner's need and learner device infrastructure). In our previous work [10], we stressed the need for a runtime requirements monitoring to detect deviations from the requirements at runtime. Therefore, here we present a requirements monitoring framework built as an autonomic MAPE (Monitor-Analyze-Plan-Execute) loop [11]. Our contribution focuses on the planning phase which is based on a constraint program (CP) to select optimal configurations (herein, refers to the set of educational resources proposed to a learner) whenever deviations occur at runtime. A constraint program is a set of constraints that define relations between declarative variables. In the context of adaptive systems these constraint programs are executed by tools, called solvers, to find solutions that satisfy the maximum number of constraints at the same time.

The idea of our modelling approach is to specify the requirements as a dynamic software product line (DSPL) using REFAS (Requirements for (Self-) Adaptive Systems) language [12] and the VariaMos tool [23] that supports some DSPL languages. Afterwards, the CP corresponding to our DSPL is automatically generated [28] from the specification.

The remainder of this paper is organized as follows. In Section 2 we reviewed some of work related to adaptive e-learning systems and requirements monitoring of self-adaptive systems. Section 3 introduces the running example that is used throughout this paper to illustrate our proposal. Section 4 presents our approach. Section 5 shows simulation of how the requirements monitoring detects deviations and plans optimal configurations using the CP. Finally, Section 6 concludes the paper.

II. RELATED WORK

This section presents related work on adaptive e-learning systems and requirements monitoring.

A. Adaptive e-learning systems

Several research has been developed to support adaptive e-learning systems including the learning monitoring. As far as we know, this monitoring aims to provide teachers with a data visualization tools through which they can track and record learners preferences during the interaction with the system, so that they can have a reliable view to help them understand learners behavior, and subsequently to enhance the quality of the proposed courses [13],[14],[15]. However, this passive

monitoring is related to the ability of teachers to spend more time and effort in order to interpret these data. With our approach, the monitoring aims to detect deviations from the desired requirements and propose a list of alternative configurations to improve the learning process at runtime.

B. Requirements monitoring

Interesting work has been performed in modeling and monitoring requirements for self-adaptive systems. Fickas et al. [16] and Feather et al. [17] proposed a runtime monitoring system that applies a formal language FLEA (Formal Language for Expressing Assumptions) that allows translating KAOS (Knowledge Acquisition and Automated Specification) assertions into temporal combinations of events. Consequently, if an event occurs, the system applies its remedial actions stored in a triggered database. Robinson [18] extended Feather's model and proposed a requirements monitoring for adaptive service-based applications, named ReqMon. In his approach, the requirements model is conceptualized using KAOS language and a time model is used to detect obstacles from which the web service monitors are derived automatically and applied to the running system. Goldsby *et al.* [19] proposed LOREM, an approach to specify the requirements of a dynamic adaptive system (DAS) as i* goal models; however, LOREM does not support the specification of requirements at runtime. In short, despite, the requirements monitoring using goal driven specification like KAOS and i* are widely adopted, they are still based on a static strategy that enumerates all alternative behavior at design time. To address this gap, Baresi & Pasquale [20] introduced the concept of "adaptive goals" to represent "fuzzy requirements" which can be satisfied at different degrees using a fuzzy logic operators. In their approach, a membership function is maintained to assess the satisfaction of goals at runtime and whenever a boundary level is reached, it activates adaptation actions (adding/removing goals) to update the KAOS goal model. Sawyer *et al.* [21] proposed an extend KAOS goal model to construct a requirements model and enable a transformation of the resulting requirements model into a constraint program. This latter is intended to carry out the re-configurations of self-adaptive systems. Along the same lines, we use REFAS language to specify the requirements. The specification is then automatically transformed into a constraint program. Our approach supports the design time modeling and verification of models through simulations (see Section V) using a graphical tool, namely VariaMos[1] [22].

Finally, other formalisms such probabilistic and real-time temporal logics are also applied for the requirements specification. The significant benefits of using such techniques include the ability to define rigorous requirements specifications and to analyze them using mathematically-based tools like probabilistic model checker [23]. However, the formalization and inference are generally complicated and resource-consuming [24]. Moreover, our main objective is to predict the deviations and not just to react when anomalies are detected.

[1] http://www.variamos.com/

III. RUNNING EXAMPLE

In this section, we describe the running example used to illustrate our approach. Further, we introduce REFAS language and explain in more detail how it supports the requirements modeling of dynamic product lines.

A. Description of the system

As an illustrative example, we use an adaptive e-learning system (AES) from a case study in [10]. The AES proposed by Franzoni et al. [25] offers a personalized teaching environment based on an adaptive taxonomy that combines Felder & Silverman's learning style (FSLS) [26] with suitable teaching strategies and electronic media. The learners are also supported by collaborative learning.

According to FSLM (see Table 1), learners are classified under four dimensions that describe the way in which they (1) receive (visual/verbal), (2) perceive (sensitive/intuitive), (3) understand (global/sequential) and (4) process (reflexive/active) information.

Table 1: FSLS dimensions

Dimension	Learning style
Perception	**Sensitive :** -prefer fact, experiments**,** sounds, physical sensations
	Intuitive : -prefer abstract material **-** learn through hunches and theories
Entry Channel	**Visual** : prefer images, diagrams and graphics
	Verbal : prefer spoken words and sounds
Understanding	**Global** :learn through leaps and an integral approach
	Sequential: -learn through small and continuous steps
Processing	**Reflective**: - think about information quietly and stop periodically to review. - prefer to work alone
	Active: - assimilate new information through physical activities - enjoy discussions (chat, forum ...)

In our running example, the system encourages collaborative learning by supporting two collaborative groups: "*online group*" and "*physical self-help group*". As a result, learners who share the same geographical space are able to have face-to-face communications in addition to online meetings.

B. Overall requirements

This subsection presents the requirements of the adaptive e-learning system we use as running example (functional and non-functional requirements), the relationships among them and the assumptions about the environment.

R0: Learners should have educational resources that match their learning styles.

R0 is a hard requirement that should be ensured by the system.

R1: Learners should not face a long waiting time for downloading educational resources.

The monitoring shall identify relevant educational resources in term of media (text, graphics, mage and video). This requirement can be monitored by keeping track of network performance over time. For instance, for an educational video resource, the monitoring may take bandwidth measurements and continuously suggest a video in different quality proportional with the measurements, or in case of inappropriate bandwidth inform the system to ask learners if they prefer other optimal media format even if they do not match their learning style.

R2: The system should encourage collaborative learning and maintain homogeneous discussion groups.

The monitoring shall keep track of learners' location and update collaborative groups accordingly, to enable learners to meet their peers who attend the same course if they are nearby. In this way, e-learning can be extended to face-to-face meetings, which help learners to be actively engaged.

R3: The system should ensure an available and effective communication tools.

The monitoring shall detect effective communication tools for every learner with regard to the frequency of accessing the learning system. As consequence, if the frequency is low, the system needs to communicate with the learner using Email or SMS to keep him aware of current events and available educational resources. This would increase the learner's motivation and the probability that the learner will not drop the course.

R4: Educational resources should be accessible to all potential learners.

The monitoring shall identify noisy environment and then provide learners text captioning or transcription for audio and video media.

C. REFAS language and requirements model

It is increasingly common for adaptive systems to support user requirements, system resources and the operating environment. In this context, the requirements models are used as specifications for the developers and the designers of the system. The system must deal with many variability decisions, which make the design a challenging task. In our modelling approach, we propose to use REFAS, a requirements engineering language that supports variability modelling concepts and relations to design accurate requirements models. Several types of views are proposed to specify different requirements concerns [11]. In particular, we use the following views to represent our requirements model.

(i) Variability view

This view represents functional requirements as goals and relations between them. A goal refinement decomposes each goal in several sub-goals (AND refinement) or alternative combinations of sub-goals (OR refinement). The refinement process of goals terminates when they can be "operationalized"; i.e., solved by components or operations. Each leaf-goal may have different operationalizations that represent system's variability. Moreover, relations like "requires" or "excludes" can be used to constrain the selection of some operationalizations and, subsequently, reduce the configuration space.

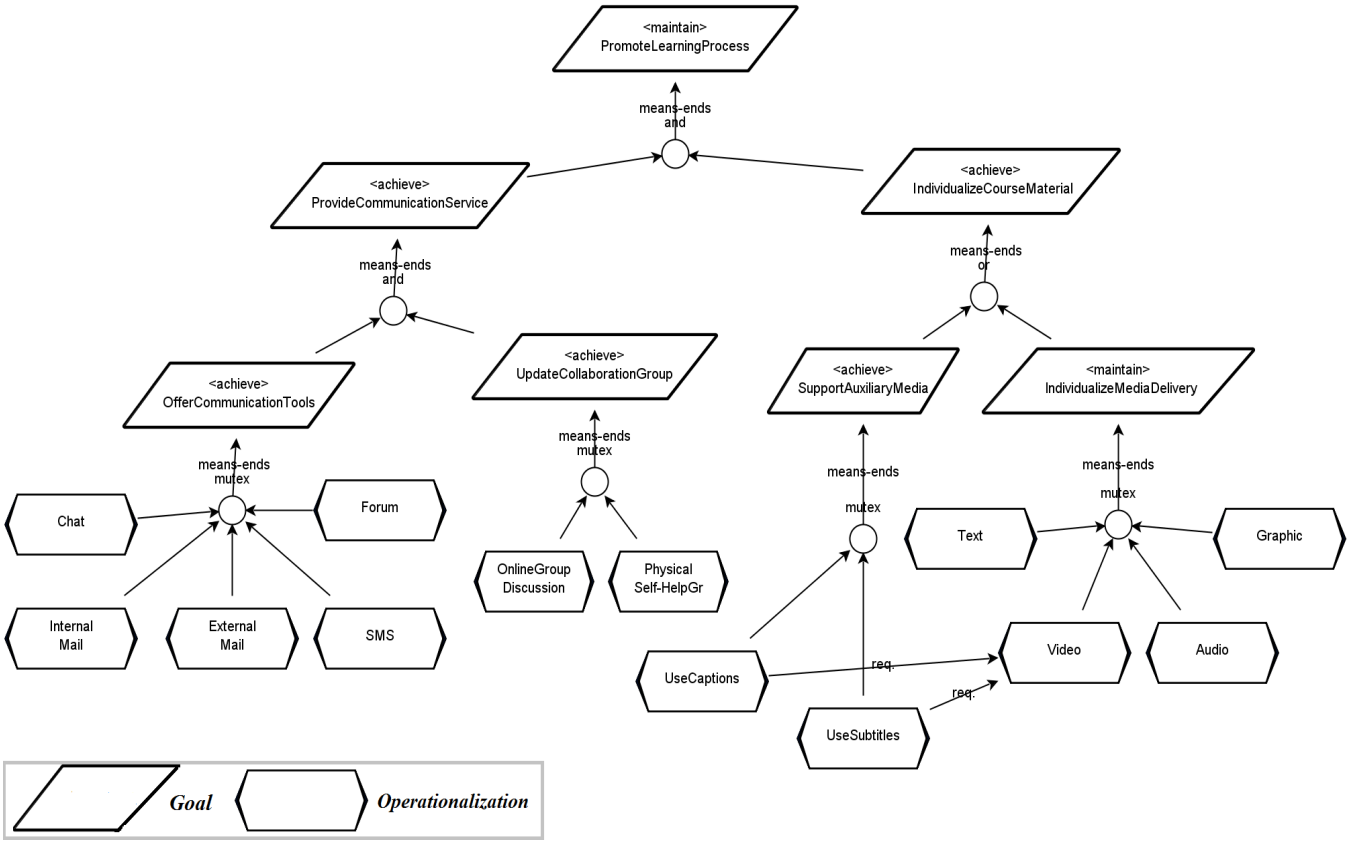

Figure 1: Functional variability view (from [10])

Figure 1 depicts the functional variability of our running example. The top goal "*Promote Learning Process"* depends on the two sub-goals "*Individualize Course Material"* and "*Provide Communication Service*" to be satisfied. Additionally, the goal "*Provide Communication Service*" depends on the two leaf-goals "*Offer Communication Tools*" and "*Update Collaboration Group*" to be satisfied, and so on.

Finally, each leaf-goal has at least one operationalization. For example, "*Individualize Media Delivery*" goal can be operationalized by relevant media like text, video, graphics and images.

(ii) Soft goals view

It represents the non-functional requirements as soft goals and relations between them.

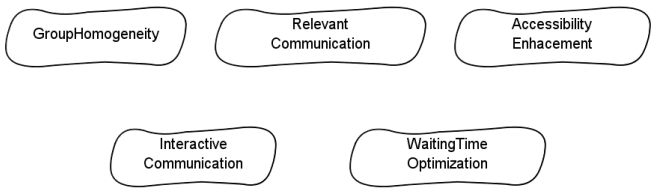

Figure 2: Soft goals view (from [10])

Figure 2 shows the soft-goals view for the running example. This model is composed of five soft goals related to the aforementioned "soft-requirements" {R1, R2, R3, and R4}:

1. *Waiting time optimization*
2. *Relevant communication*
3. *Group homogeneity*
4. *Interactive communication*
5. *Accessibility enhancement.*

(iii) Soft goals satisficing view

This view defines the impact of the operationalizations and the context variables on satisficing the soft-goals. To this end, it specifies (1) claims to define expected satisficing levels from the operationalizations and (2) soft-dependencies to define required satisficing levels for each context variable values. The satisficing level is defined using 5-point scale ranging from (0) completely denied to (4) completely satisficed.

Figure 3 depicts 4 soft-dependencies and 13 claims. For instance, *C7* indicates that using *Audio* with a low *Audio Bitrate* fully satisfices *Waiting Time Optimization,* while the SD4 indicates that the soft-goal *Waiting Time Optimization* should be fully satisficed when *Network Speed* is low.

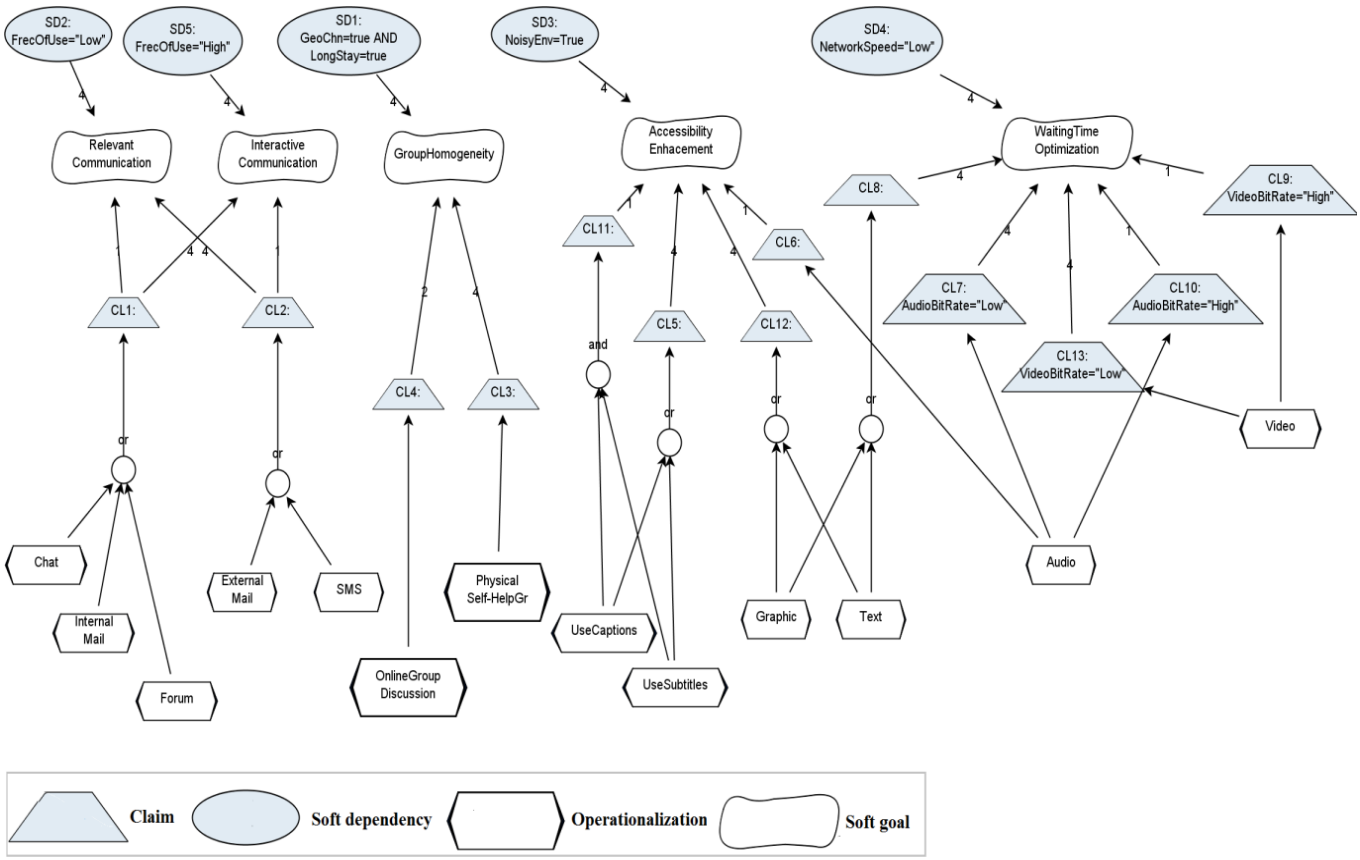

Figure 3: Soft goals satisficing view (from [10])

(iv) Context view

It supports the definition of context variables and relations between them.

Figure 4 depicts the following context variables for the running example:

- *Video bitrate* {high, low}
- *Network speed* {high, low}
- *Audio bitrate* {high, low}
- *Geographical change space* {true, false}
- *Frequency of use* {high, low}
- *Noisy environment {true, false}*

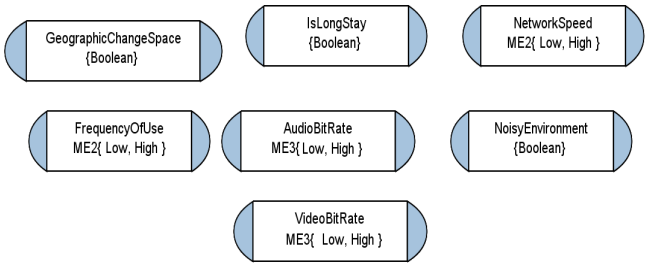

Figure 4 : Context view (from [10])

Notice that to avoid overloading the operating system, the running example allows *Geographic change* only for long stays; i.e., when the stay last more than 30 days. In this regard, when the monitoring system detects a change in learner geographical space, it alerts the system to verify and send feedback about the nature of the stay (long or short). We will return to the running example and give more details after describing our proposal.

IV. PROPOSED APPROACH

In this paper, we propose a monitoring process that should be performed in two stages: at design time and at runtime. The idea within the design time monitoring is to simplify the requirements specification and carry out the verification of the specification using graphical simulations. The verified specification is used as a good starting point to automatically build a CP. At runtime, the requirements monitoring rest on the generated CP to verify the satisfaction of the requirements and generate optimal configurations that satisfy most of the constraints regarding the learning requirements and the operating environment.

A. At design time

The administrator elucidates business goals, the requirements to be monitored and assumptions about the environment. And then the designer builds the requirements model as a DSPL using REFAS language and verifies the requirements model through VariaMos. This latter is used to automatically transform the verified requirements model into a CP, where context variables, soft-goals, goals, and operationalization goals are mapped as variables, and soft-dependencies and claims as constraints.

B. *At runtime*

The requirements monitoring framework is built as MAPE loop, under adaptive e-learning systems, which continuously checks the learning requirements satisfaction at runtime using the generated CP.

Figure 5 depicts the overall framework of our approach. The right upper part shown a general architecture of AES which is composed of three models: (1) the learner model represents the current state of a learner that includes relevant information such preferences, goal, learning style, progress and behavior. This information is derived from questionnaires, logging data and interaction sequences and it allows the system to perform the desired adaptation. (2) The domain model represents relations between concepts in the target subject area and general information such media format options of educational resources. And (3) the adaptation model provides different adaptations strategies to personalize the learning process by matching the educational resources to each learners.

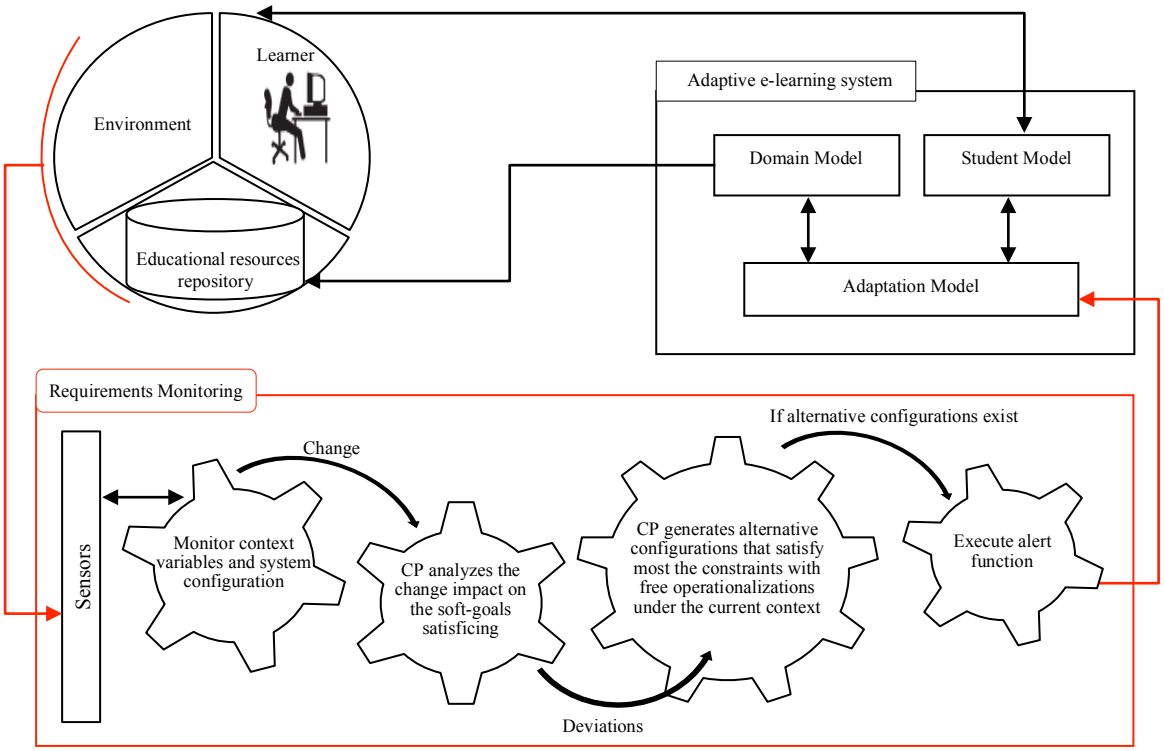

Figure 5 : Monitoring requirements framework under adaptive e-learning systems

As depicted in the Figure 5 (lower part), the MAPE loop involves feedback processes with four stages:

The first stage collects relevant data about the context (environment, learners and educational resources) from environmental sensors (external sensors) or other sources such logging data (internal sensors). In this stage, the requirements monitoring may verify certain variables values by working closely with the system and learners. For example, the requirements monitoring module may ask the system for available courses to enable or disable a specific media format option.

The collected information about the managed elements are stored in a parameter file and periodically updated whenever a change is detected.

Next, the analysis phase checks the impact of the change on the requirements satisfaction. The requirements evaluation is performed by the constraint programming solver which uses the available values of the variables from the parameter file, and if deviations are detected (i.e., the system is not providing the desired satisficing level of the soft-goals) triggers the planning phase.

Next, the planning stage is in change of executing the constraint program to find optimal configurations that maximizes the satisfaction of the soft goals.

Finally, the executing stage is charged to send feedbacks to inform the system about deviations and a list of available optimal configurations.

V. MORE ON THE RUNNING EXAMPLE—SIMULATION

Scenario: Let us suppose we deal with a verbal learner. According to FSLS, this learner apprehends knowledge by listening to spoken words. Consequently, the system should deliver educational video resources to this learner.

Design time simulation

By executing a simulation in VariaMos, the designer can verify the requirements model and generate solutions for a specific combination of context variables and operationalization goals. For instance, by considering an e-learning system with *Video, External Mail, Physical-Self-*

helpGr and *Use Subtitles*, and this context: *Network Speed* is 'low', *Frequency Of Use* is 'low', *Geographic Change Space* is 'true', *Video Bitrate is* 'high', *Is Long Stay* is 'true' and *Noisy Environment* is 'false', the designer can visualize the claims, soft dependencies and soft goals that are satisfied (or not) by VariaMos. This scenario is presented in Figure 6.

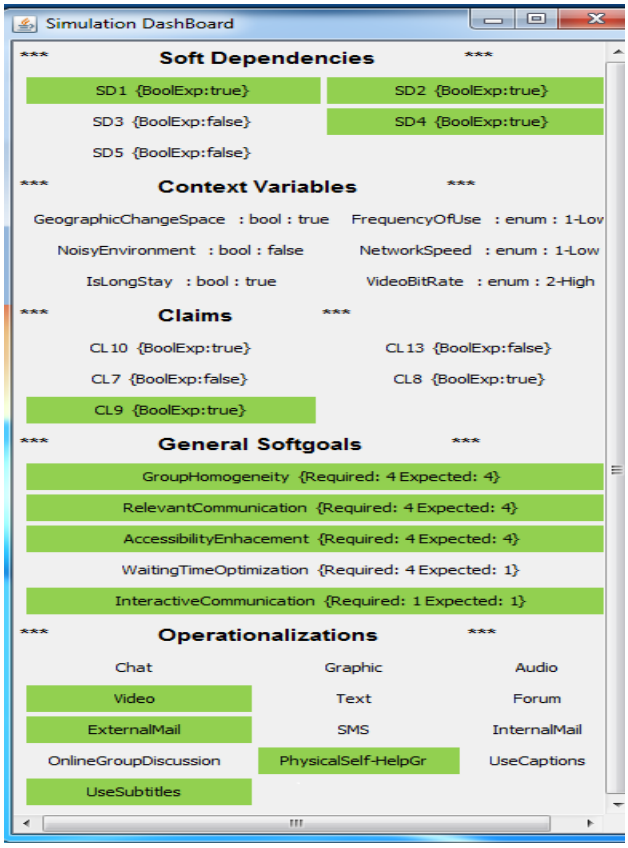

Figure 6 : Simulation of a configuration through VariaMos

Runtime reasoning

Consider the scenario described above. While the system is in operation, the requirements monitoring module selects the claim CL9 (see Figure 7) that indicates the highly positive influence of Video in satisficing the *Waiting Time Optimization* soft goal (i.e., expected level = 4). Whereas the Soft dependency SD1 (see Figure 7) indicates that the *Waiting Time Optimization* soft goal should be completely satisficed if the *Network Speed* connection is low. Therefore, the requirements monitoring detects that the soft goal *Waiting Time Optimization* is not satisficed.

As a next step, the requirements monitoring module looks for optimal configurations, then it generates 40 relevant configurations that satisfice as many soft goals as possible, with respect to the context at hand. For instance, as the scenario specifies a low *Network Speed*, an alternative may indicate the need to deliver a video with low *Video Bitrate* to satisfice the *Waiting Time Optimization* soft goal.

SD4 ⇔ (NetworkSpeed = low)

SD4 ⇒ (WaitingTimeOptimization_RequieredLevel = 4)
CL9 ⇔ ((Video is selected) ∧ (VideoBitRate = high))
CL9 ⇒ (WaitingTimeOptimization_ExpectedLevel = 1)

SD2 ⇔ (FrequencyOfUse=low)
SD2 ⇒ (RelevantCommunication = 4)
Cl2 ⇔ ((ExternalMail is selected) ∧ (SMS is selected))
CL2 ⇒ (RelevantCommunications_ExpectedLevel = 4)
CL2 ⇒ (InteractiveCommunications_ExpectedLevel = 1)

SD1 ⇔ ((GeographicChangeSpace = true) ∧ (IsLongStay = true))
SD1 ⇒ (GroupHomogeneity_RequieredLevel = 4)
CL3 ⇔ (PhysicalSelf-helpGr is selected)
CL3 ⇒ (GroupHomogeneity_ExpectedLevel = 2)

Figure 7: Extract of the constraint program in a simplified language

VI. CONCLUSION

In this paper, we have shown how to monitor the requirements for an adaptive e-learning systems (AESs) in order to improve the requirements satisfaction of these kind of systems (i.e., systems that are highly personalized under changing environmental conditions at runtime). The monitoring module continuously checks the conformity of AES to its requirements and generates optimal configurations that contribute most positively to the non-functional requirements (soft-goals). The proposed framework is implemented as a MAPE loop, our contribution focuses in the planning phase which is based on a generated constraint program (CP) from the requirements model.

Our approach allows designer to specify the requirements model with a high level specification language, to verify the resulting model at design time through simulations using VariaMos tool, and finally to transform the verified model into a CP. This automation presents two advantages: (1) it avoids a misinterpretation of the specification and a loss of information during the transformation, and (2) it explores the power of constraints programming reasoning and the expressiveness of goal modelling to represent complex systems.

Due to the fact that the tool uses a constraint programming representation of the dynamic product line, our approach is confronted to the limitations of any constraint program and the limitations of the solvers in which these programs are executed. Thus, we plan to conduct an empirical study to measure (i) the performance and scalability of this approach, (ii) the impact of the proposed framework on the learning process, and (iii) satisfaction of learners, using a real case.

REFERENCES

[1] Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for Web-based instruction. International Journal of Artificial Intelligence in Education (IJAIED), 12, 351-384.

[2] Mitrovic, A., Mayo, M., Suraweera, P. & Martin, B. (2001). Constraint-based tutors: A success story. In L. Monostori, J. Vancza & M. Ali (Eds.), Proceedings of the 14th International conference on industrial and engineering applications of artificial intelligence and expert systems IEA/AIE-2001,June 2001 (pp. 931–940). Budapest.

[3] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., ... & Stash, N. (2003, August). AHA! The adaptive hypermedia architecture. In Proceedings of the fourteenth ACM conference on Hypertext and hypermedia (pp. 81-84). ACM.

[4] Papanikolaou, K. A., Grigoriadou, M., Kornilakis, H., & Magoulas, G. D. (2003). Personalizing the Interaction in a Web-based Educational Hypermedia System: the case of INSPIRE. User modeling and user-adapted interaction, 13(3), 213-267.

[5] Tseng, J. C. R., Chu, H.-C., Hwang, G.-J., & Tsai, C.-C. (2008a). Development of an adaptive learning system with two sources of personalization information.Computers and Education, 51, 776–786.

[6] Hwang, G. J., Kuo, F. R., Yin, P. Y. & Chuang, K. H. (2010). A heuristic algorithm for planning personalized learning paths for context-aware ubiquitous learning. Computers & Education, 54, 404–415.

[7] Wang, S. L., & Wu, C. Y. (2011). Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. Expert Systems with Applications, 38(9), 10831-10838.

[8] Mampadi, F., Chen, S. Y., Ghinea, G., & Chen, M. P. (2011). Design of adaptive hypermedia learning systems: A cognitive style approach. Computers & Education, 56(4), 1003-1011

[9] Chen, W., Niu, Z., Zhao, X., & Li, Y. (2014). A hybrid recommendation algorithm adapted in e-learning environments. World Wide Web, 17(2), 271-284.

[10] Dounas L., Mazo R., Munoz Fernandez J., Salinesic.,El Beqqali O., (2015). Runtime requirements monitoring framework for adaptive e-learning systems. 26th edition of the International Conference on Software & Systems Engineering and their Applications (ICSSEA),,Paris, France.

[11] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., ... & Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In Software engineering for self-adaptive systems (pp. 48-70). Springer Berlin Heidelberg.

[12] Munoz-Fernández, J. C., Tamura, G., & Salinesi, C. (2014, September). Towards a requirements specification multi-view framework for self-adaptive systems. In Computing Conference (CLEI), 2014 XL Latin American (pp. 1-12). IEEE.

[13] Graphvis (2004) visualization tool monitoring group communications in order to help instructors detect collaboration problems.

[14] Mazza, R., & Dimitrova, V. (2007). CourseVis: A graphical student monitoring tool for supporting instructors in web-based distance courses. International Journal of Human-Computer Studies, 65(2), 125-139.

[15] Romero-Zaldivar, V. A., Pardo, A., Burgos, D., & Kloos, C. D. (2012). Monitoring student progress using virtual appliances: A case study. Computers & Education, 58(4), 1058-1067.

[16] Fickas, S., & Feather, M. S. (1995). Requirements monitoring in dynamic environments. In Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on (pp. 140-147). IEEE.

[17] Feather, M. S., Fickas, S., Van Lamsweerde, A., & Ponsard, C. (1998). Reconciling system requirements and runtime behavior. In Proceedings of the 9th international workshop on Software specification and design (p. 50). IEEE Computer Society.

[18] Robinson, W. N. (2003). Monitoring web service requirements. In Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International (pp. 65-74). IEEE.

[19] Goldsby, H. J., Sawyer, P., Bencomo, N., Cheng, B. H., & Hughes, D. (2008). Goal-based modeling of dynamically adaptive system requirements. In Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the (pp. 36-45). IEEE.

[20] Baresi, L., & Pasquale, L. (2010). Adaptive goals for self-adaptive service compositions. In IEEE international conference on Web Services (ICWS), (pp. 353-360). IEEE.

[21] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H., & Bruel, J. M. (2009). Relax: Incorporating uncertainty into the specification of self-adaptive systems. In Requirements Engineering Conference, 2009. RE'09. 17th IEEE International (pp. 79-88). IEEE.

[22] Sawyer, P., Mazo, R., Diaz, D., Salinesi, C., & Hughes, D. (2012). Using constraint programming to manage configurations in self-adaptive systems. Computer, (10), 56-63.

[23] Mazo R., Muñoz-Fernández J., Rincón L., Salinesi C., Tamura G. (2015). VariaMos: an extensible tool for engineering (dynamic) product lines. 19th International Software Product Line Conference (SPLC), Nashville-USA.

[24] Calinescu, R., Grunske, L., Kwiatkowska, M., Mirandola, R., & Tamburrelli, G. (2011). Dynamic QoS management and optimization in service-based systems. Software Engineering, IEEE Transactions on, 37(3), 387-409.

[25] Lalanda, P., McCann, J. A., & Diaconescu, A. (2013). Autonomic Computing. Springer London Limited.

[26] Franzoni, A. L., Assar, S., Defude, B., & Rojas, J. (2008, July). Student learning styles adaptation method based on teaching strategies and electronic media. In Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on (pp. 778-782). IEEE.

[27] Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. Engineering Education, 75(7), 674-681.

[28] Mazo R., Salinesi C., Diaz D., Djebbi O., Lora-Michiels A. (2012) Constraints: the Heart of Domain and Application Engineering in the Product Lines Engineering Strategy. International Journal of Information System Modeling and Design IJISMD. pp. 33-68. ISSN 1947-8186, eISSN 1947-819.Vol. 3, No. 2.