Runtime Requirements Monitoring Framework for Adaptive e-Learning Systems

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

Runtime Requirements Monitoring Framework for Adaptive e-Learning Systems

[1, 2]Lamiae DOUNAS, [1]Raul MAZO, [1,3]Juan C. MUÑOZ-FERNÁNDEZ, [1]Camille SALINESI, [2]Omar EL BEQQALI

[1]CRI, University Paris 1 Panthéon-Sorbonne, Paris, France
[2] LIAAN, Faculty of Sciences Dhar el Mehraz USMBA, Fez , Morocco
[3]I2T/DRISO, Facultad de Ingeniería, Universidad Icesi, Cali, Colombia
lamiae.bourkiza@malix.univ-paris1.fr;{raul.mazo, camille.salinesi}@univ-paris1.fr;
jcmunoz@icesi.edu.co; omar.beqqali@usmba.ac.ma

Abstract: As academic learners and companies are turning to e-learning courses to achieve their personal and professional goals, it becomes more and more important to handle service quality in this sector. Despite scientific research conducted to personalize the learning process and meet learner's requirements under adaptive e-learning systems, however, the specification and management of quality attribute is particularly challenging due to problems arising from environmental variability. In our view, a detailed and high-level specification of requirements supported through the whole system lifecycle is needed for a comprehensive management of adaptive e-learning systems, especially in continuously changing environmental conditions. In this paper, we propose a runtime requirements monitoring to check the conformity of adaptive e-learning systems to their requirements and ensure that the activities offered by these learning environments can achieve the desired learning outcomes. As a result, when deviations (i.e., not satisfied requirements) occur, they are identified and then notified during system operation. With our approach, the requirements are supported during the whole system lifecycle. First, we specify system's requirements in the form of a dynamic software product line. This specification applies a novel requirements engineering language that combines goal-driven requirements with features and claims and avoid the enumeration of all desired adaptation strategies (i.e. when an adaptation should be applied) at the design time. Second, the specification is automatically transformed into a constraint satisfaction problem that reduces the requirements monitoring into a constraint program at runtime.

Keywords: Requirements monitoring, adaptive e-learning systems, constraint programming, dynamic software product lines, goal modelling.

1 INTRODUCTION

E-learning, also known as online learning or technology-enhanced learning is expected to make a radical difference to education, specifically, the quality and effectiveness of learning experience [1]. In recent years, e-learning systems have gained in popularity among academic and companies; the global e-learning market is expected to reach USD 107 billion by 2015 as reported by the Global Industry Analysts, Inc. Therefore, it becomes more and more important to handle service quality of this sector.

Among the existing technologies enhanced learning, adaptive e-learning systems have arisen as an alternative to traditional "one size fits all" learning approach that provides the same content for all learners regardless of their cultural diversity or how they learn[2][3]. On the one hand, some studies have attempted to provide personalized contents and navigation support for each learner, based on several characteristics of learners including knowledge level, goals, learning style or personal preferences [4-12]. On the other hand, others studies have stressed technical aspects including learning standards like AICC, IMS, SCORM, and LOM, to give flexibility to learning systems, in term of contents as well as in structure.

As we move into a new era of dynamic adaptation, more attention is needed to support the requirements of adaptive e-learning systems and allow their evolution at runtime. So far, the main used approach to model adaptation is the design time enumeration of all desired adaptation strategies, on every context. But, this is clearly a fastidious and cumbersome task due to the number of decisions needed, considering continuously changing environmental conditions (e.g., learner engagement, learner location, device status, and network connectivity). Moreover, some requirements or features of the environment may be unknown at design time which may limit the system self-awareness [13] and reduce guarantee that adaptation strategies will remain

adequate during the system operation. Finally, the lack of rigorous evaluation of adaptive e-learning systems is another important issue. This because, it is hard to specify control conditions once the system is build and attributes cause ([14] [15] [16]).

In order to overcome the issues mentioned above, we propose a runtime requirements monitoring framework to guide the evolution of adaptive e-learning systems. This framework continuously checks the conformity of an adaptive e-learning system to its requirements and informs the system whenever it detects a deviation. The requirements specification is inspired from Sawyer et al.'s approach [17] that consists in specifying and transforming an extended goal-driven requirements model into a constraint program. Accordingly, we use its evolution, called REFAS [18] (Requirements for (Self-) Adaptive Systems), a language to specify the requirements in the form of dynamic software product line specification combined with claims. Afterward, the specification is automatically transformed into a constraint satisfaction problem that reduces the requirements monitoring into a constraint program (CP).

The rest of the paper is organized as follows. Section 2 presents background information about materials used throughout the paper. Section 3 presents our approach. First, we introduce an adaptive e-learning system that will be used as a running example, and then we describe the proposed requirements monitoring architecture. Section 4 describes the application of the approach to the running example and how the monitoring requirements detect deviations and plan optimal configurations using a constraint program. Section 5 discusses related work. And finally Section 6 concludes and presents future work.

2 BACKGROUND

The main objective of our requirements monitoring framework is to check the satisfaction of learning requirements at runtime. This implies to be able to specify the requirements of the adaptive e-learning systems and be aware of changes in their environments at the same time. Therefore, we are particularly interested in establishing relationships among three key elements:

- Learning materials
- Learning requirements
- Assumptions about the environment

A simple way to do would be the design time enumeration of all possible instances of these elements and the relationship between them in a declarative statement. However, this statement can miss some unknown requirements of the operational environment, and it would also produce a tangled design. In contrast, we propose a much convenient way that uses a detailed and high-level requirements specification and supports the monitoring in the whole system lifecycle.

2. 1 REFAS language

REFAS language is a novel requirements engineering language that seeks to manage uncertainty and to be sufficiently expressive for self-adaptive systems. It allows the representation of all relevant concepts in different views [18]. From these views, we are interested in the following four views that conform with our requirements model:

- **Variability view**: represents system requirements (functional requirements) as goals. The top-level goal should state the overall objective of the system. This goal is decomposed then in several sub-goals (AND/OR-refinement) and the process of refinement ends when the leaf goals can be operationalized (i.e. solved by an operation or components). A goal may have several operationalization goals, which represents system's architectural variability. The decision about the satisfiability of selected configuration is reported at runtime to ensure acceptable behaviour under the current context.
- **Soft goals view**: represents system's non-functional requirements as soft goals.
- **Context view**: defines the context variables that impact the learning process and relation between these variables.
- **Soft goals satisficing view**: represents assumptions about the context and their implication over the soft goals satisfaction. The view specifies soft-dependencies to express required levels of soft goal satisfaction for condition based on context values, and claims to indicate satisfaction level of soft goals expected from a combination of operationalization goals.
  Note that, the levels of satisfaction can be expressed from a set of values {0, 1, 2, 3, 4} ranging from complete denial (0) to complete satisfaction (4).

2. 2 VariaMos Tool

VariaMos (Variability Models) is a Java-based tool for specification, automatic verification, analysis, configuration, integration and simulation of variability models [19].It offers an automatic transformation of requirements specifications of self-adaptive systems into a high-level constraint program [20], and then compiled into the language of the solver at hand to execute the resulting model. In this paper, it is used to implement REFAS requirements model, realize simulations about how the goals can be reached, and the soft goals can be satisfied, and transform the model into constraint program. More details about the tool can be found online[1].

3 MODELING APPROACH

3.1 Running example

To support discussion about the need for continuous requirements monitoring at runtime for adaptive e-learning systems, we apply our approach to an adaptive e-learning system proposed by Franzoni et al. [21]. The system aims to adapt the presentation of course materials (text, graphics, video, audio, forum, etc.) to each learner based on his learning style and encourage collaborative learning.

The learning style is defined using the Felder & Silverman model [22] that classifies a learner under four dimensions: sensing/intuitive, visual/verbal, active/reflective and sequential/global. Then, the learning style and teaching strategies are matched based on the following taxonomy:

- *Sensitive* learners prefer facts, data and experimentation (laboratory exercises, problem-solving), they are patient with details but don't like complication. Whereas *intuitive learners* prefer theories and principles, they get bored with details and accept complication.
- *Visual* learners remember the things they see. The information gathering must use visual representations (images, graphics, and text). Whereas *verbal learners* remember the things they have heard. The learning content must have a lot of oral representation (video, audio).
- *Active learners* tend to apprehend and assimilate new information when they practice and work with others (discussion, chat, and forum). Whereas *reflective learners* think about information quietly, before go ahead and stop periodically to review what have been learned. The learning content must be related to with experience (question and answer methods, case study).
- *Sequential* learners learn through small orderly steps (chapters, daily homework). Whereas, *global learners* learn through big steps and prefer to see everything as a whole (Project, case study).

To extend our running example, we assume the system stimulates collaborative learning by managing two groups namely, an *online* group and a *physical self-help* group. Learners living in the same geographical space can use a *physical self-help* group. So they can have face-to-face meetings as well as online meetings with this group.

Now, let us describe a scenario we expect the need for requirements monitoring at runtime:

According to Felder & Silverman, the system knows what appropriate presentation of courses materials for a learner. However, there are some ignored driving characteristics to be considered when proposing course materials like network connection speed, bitrate and learner's preferences. For example, for a video lecture, the system need to consider bandwidth measurements and deliver the video file in different quality or when the bandwidth is too low, ask learners if they prefer textual presentation. In this scenario, the impact of the event of bandwidth was not anticipated at design time. Moreover, the system needs to work with learners, at runtime, to find relevant presentation of course materials in each context and avoid high waiting time.

3.2 Runtime requirements monitoring framework for adaptive e-learning systems

The requirements monitoring framework is performed in two stages:

- *At design time*, as shown in Figure1, the administrator of the system elucidates the business goals and the requirements (non-functional requirements) to be monitored. The designer builds then the requirements model using REFAS language and specifies soft-dependencies and claims. Finally, the model is transformed

[1]http://variamos.com/

into a constraint satisfaction problem where the goals, soft-goals, operationalization goals and context variables are mapped as variables and the claims and soft-dependencies as constraints.

At this stage, our approach aims to facilitate the requirements specification and verify the requirements model using simulations under a graphical tool, namely VariaMos.

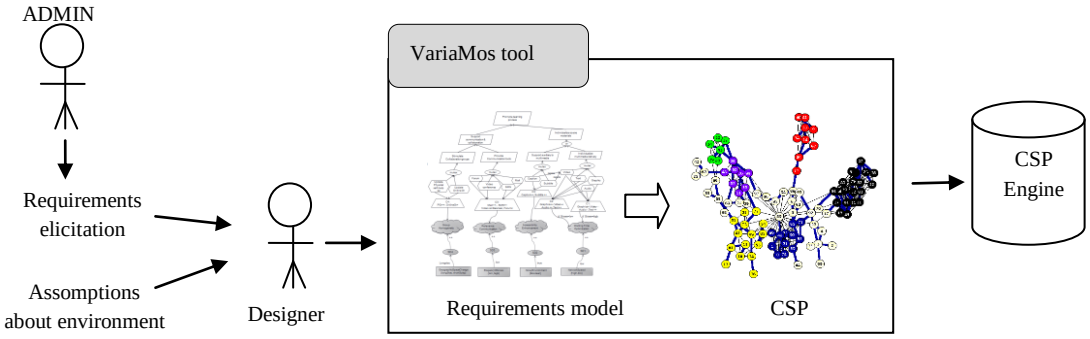

Figure 1: Requirements monitoring at design time

- *At runtime:* the requirements monitoring is reduced to a constraint program that verify the satisfaction of requirements and find optimal configurations that satisfy all constraints using the generated constraint satisfaction problem (CSP) from the design time stage.

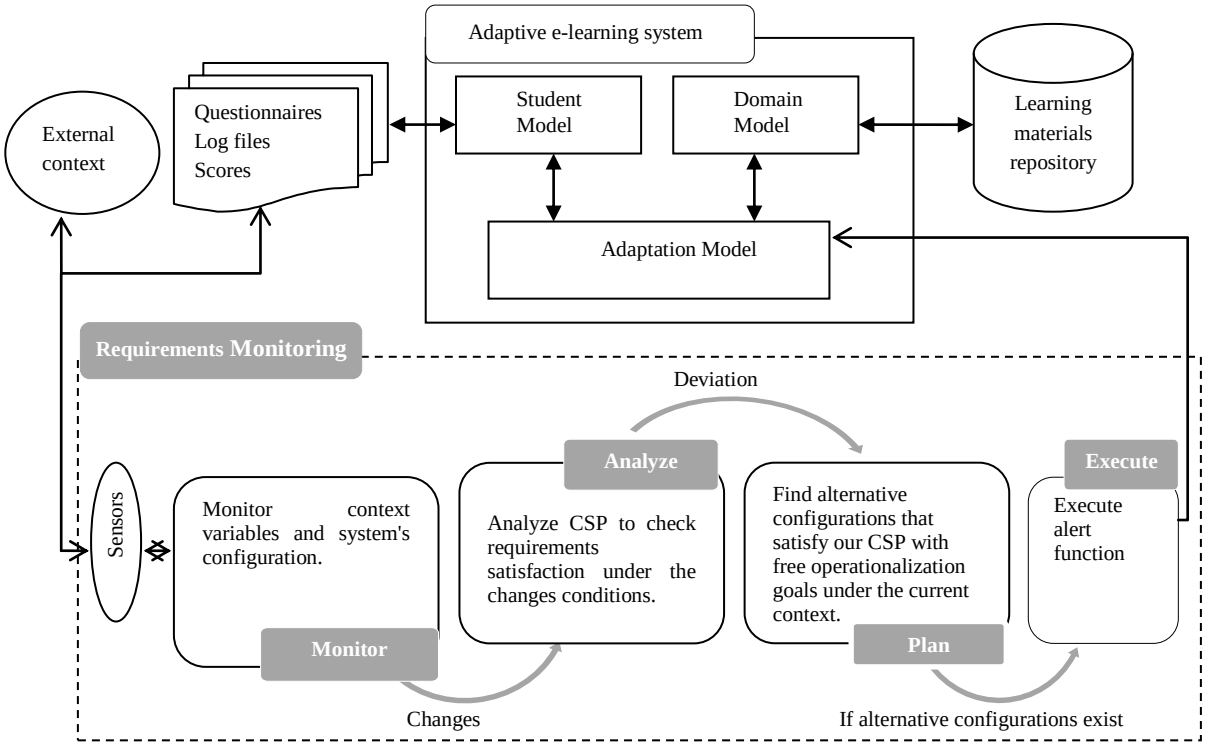

Figure 2: Runtime Requirements monitoring framework of adaptive e-learning systems

As illustrated in Figure 2, our requirements monitoring is implemented as an autonomic computing MAPE (monitor-analyze-plan-execute) loop architecture [23] under adaptive e-learning system:

In the upper part of Figure 2 shown, a typical architecture of adaptive e-learning that is organized in the form of three basic models: the *student model* represents information about learners. This information can be gathered from questionnaires or dynamically from log files which give a complete description of the current

state of each learner. The *domain model* structures knowledge about the domain to be learned, by describing the relationship between different concepts of the learning domain. And *adaptation model* implements the specification of adaptation rules to fit the learning materials to learners.

In the lower part of Figure 2 shown the MAPE loop control that continuously checks the satisfaction of learning requirements through four stages:

The first stage of the MAPE loop involves monitoring both:

1) System's context (or the external environment) by means of sensors. For example, the monitoring can use GPS to track learners and detect change in their geographical location.

2) Configuration (i.e. the whole operationalization goals used by the system) using log files from running instances of the system.

The current state information about managed elements is then stored in a data file and updated progressively to reason about. In this respect, when the monitoring identifies changes, the analysis stage is intended to verify the impact of the changes on learning requirements satisfaction by executing our constraint program using the values in the data file from the first stage. And if e-learning system is not providing the required satisfaction level, it triggers the next stage.

The planning stage is then employed to find optimal configurations of operationalization goals according to the values of the current context from the data file.

The executing stage alerts the system about deviations and presents a list of the available optimal configurations.

Note that, our monitoring framework may identify new requirements and verify some variables values from the system. For example, the monitoring may ask the system available representations of a typical course to enable/disable some multimedia options (operationalization goals).

4 APPLICATION AND SIMULATIONS

4.1 Applying our approach to the running example

4.1.1 Requirements analysis

In the running example, the aim is to promote the learning process by adapting the representation of learning materials to each learner based in learning style and supporting learner collaboration. Besides these functional requirements, the administrator of the system provides additional non-functional requirements. The focus herein is to list the overall requirements and describe the relation between these requirements and assumptions about the operational environment:

R0 : *Learners should have learning materials that fit their learning styles.*

In our view, monitoring this 'hard' requirement is not important and would be redundant, since our monitoring would likely not be able any more effective that the system itself. The remaining 'soft' requirements are then supported by the requirements model.

R1 : *Learners should not have to wait unduly long for loading learning materials.*

The monitoring framework shall consider network connectivity to identify relevant digital contents representations (video, graphics and audio, or text). For example, a learner can connect over a low bandwidth modem line, if the system presents a heavy video lecture, the learner may wait unduly long for the transmission of data. In this case, the monitoring should identify alternatives learning material based in their sizes to optimize the waiting time.

R2 : *The system should support collaborative learning and ensure a good match between collaborative groups.*

The monitoring framework shall track change in the geographical space and maintain the homogeneity of collaborative groups. And if changes occur, inform the system, so it can update their groups.

R3 : *The system should provide learners an accessible and available communication tools.*

The monitoring framework shall monitor learners' access to the e-learning platform and detect relevant communication tools for every learner depending of its frequency of access. For example, the system needs to use SMS or Email for learners with low frequency to keep learners aware of what happening in the platform, which might motivate them to connect and follow courses.

R4 : *The system needs to support the accessibility to learning materials in different contexts.*

The monitoring framework shall detect noisy environments and identify relevant auxiliary to multimedia like caption and subtitle.

4.1.2 Requirements model

The requirements model of the running example is supported by the four aforementioned views of REFAS language:

- Considering *the variability view* (see Figure 3), the root goal is *PromoteLearningProcess* which is AND-refined into two sub-goals *IndividualizeCourseMaterial* and *ProvideCommunicationService*. The former is AND-refined into two leaf goals *OfferCommunicationTools* and *UpdateCollaborationGroup*. And the latter is OR-refined into two leaf goals *IndividualizeMultimediaDelivery* and *SupportAuxiliaryMultimedia*. Each leaf goal has several operationalizations, for instance, *OfferCommunicationTools* goal can be operationalized using *Forum*, *Chat*, *InternalMail*, *SMS*, or *ExternalMail.* In addition, the use of some operationalization can be conditioned by constraints like *require* or *exclude*. This kind of constraints can considerably reduce the configuration space at runtime. In our example, *UseSubtitle*/*UseCaption* and *Video* are related by required relations, which means if subtitle or caption is activated, a video lecture must be selected.
- Considering *the soft goals view* (see Figure 4), the view supports five soft goals: *WaitingTimeOptimization*, *GroupHomogeneity*, *RelevantCommunication*, *InteractiveCommunication* and *AccessibilityEnhancement.* These soft goals represent the aforementioned 'soft' requirements {R1, R2, R3, R4}.
- Considering *the context view* (see Figure 5), the context variables that may affect the system include the *NetworkSpeed*, *GeographicalChangeSpace {true, false}*, *IsLongStay* {true, false}, *VideoBitRate* {high, low}, *AudioBitRate* {high, low}, *FrequencyOfUse* {high, low} to the system and *NoisyEnvironment* {true, false}.

Note, that to avoid overloading the system, the monitoring supports *GeographicChangeSpace* only for long stay (i.e., when duration of change exceeds 30 days). Thus, when a change in the geographical space of learner is detected, the monitor informs the system, this latter, ask the learner to verify if he will stay more than 30 days and send feedback to the monitoring.

- *Considering the soft goals satisficing view* (see Figure 6), the view supports 13 claims and 4 soft-dependencies. For instance, *C8* indicates that using *Text* course fully satisfies *WaitingTimeOptimization,* and *SD4* indicates that when *NetworkSpeed* is *low*, the soft-goal *WaitingTimeOptimization* should be fully satisfied (level 4).

Note that, the last view acts as a filter and permits the exclusion of non-relevant configurations by using the claims and soft-dependencies to reason about soft-goals. The combination of the five SD defines 16 scenarios of soft goal satisfaction. Within each scenario, the monitoring considers the variation of course materials and learner preferences. Without the proper constraints on the *soft goal satisficing view*, the configuration space grows exponentially. In our running example, each scenario has around 40 possible configurations.

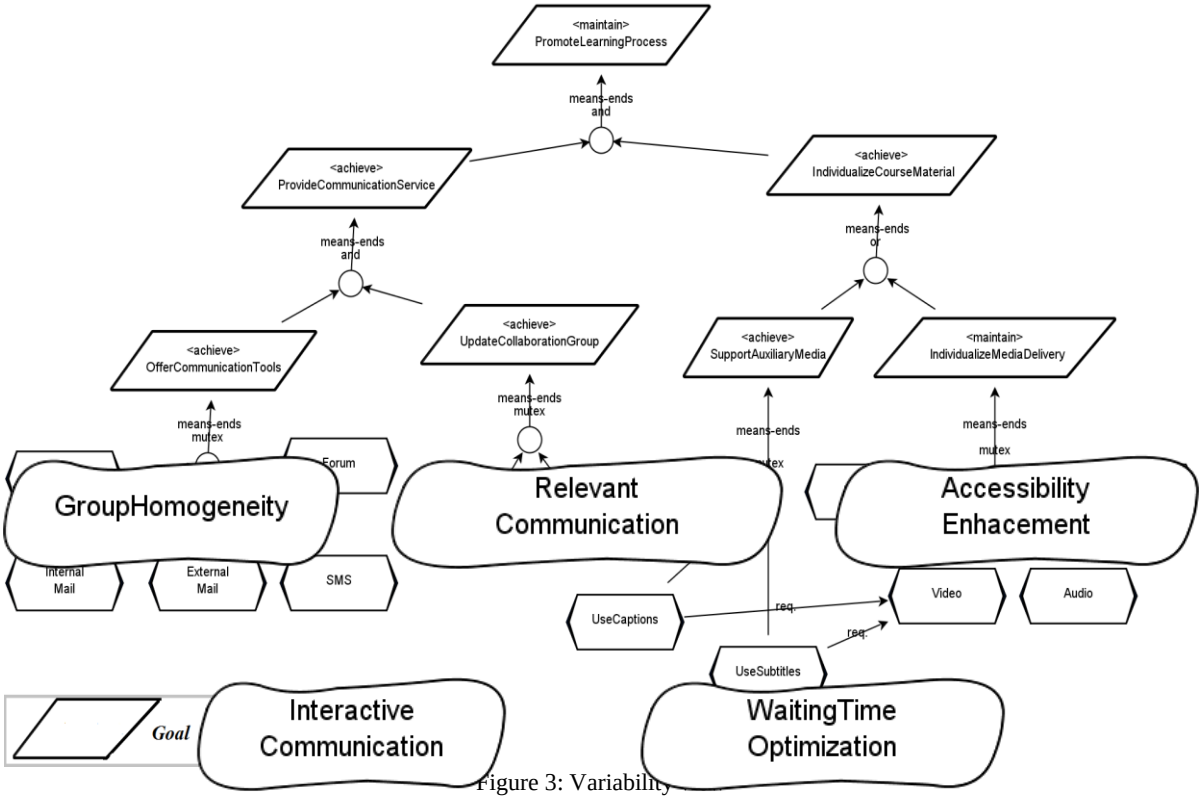

Figure 3: Variability

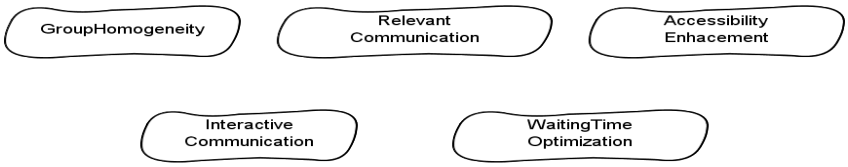

Figure 5: Soft goals view

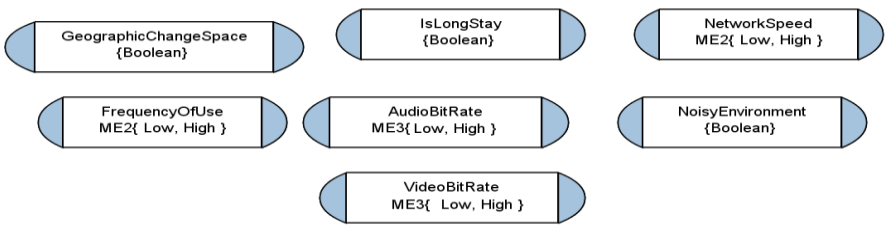

Figure 6: Context view

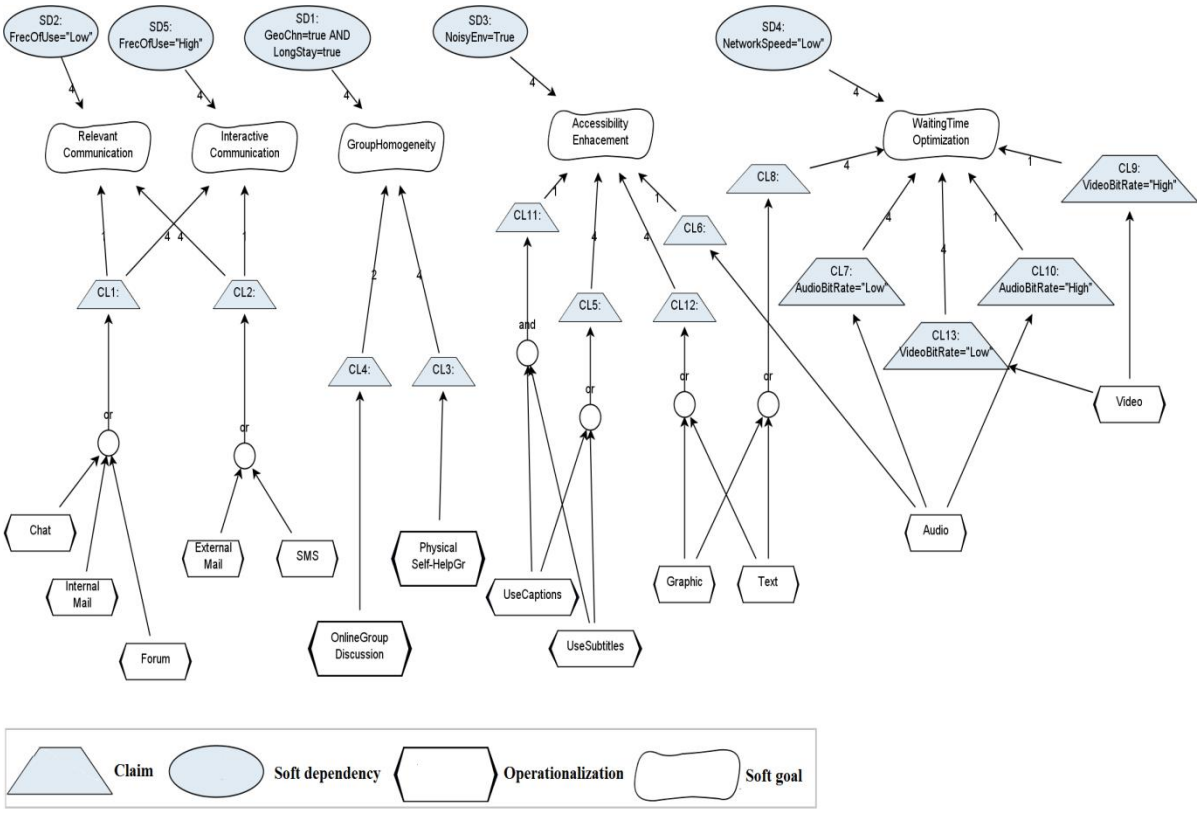

Figure 7: Soft goals satisficing view

4.2 Simulations

The primary objective of our requirements monitoring is the detection of deviations. In the following, we present two examples to illustrate how the requirements monitoring infrastructure dynamically detects a satisfaction problem using the generated constraint satisfaction problem and plans optimal configurations.

Verbal Learner Example

Let us suppose that we have a verbal learner, i.e. she understands information more effectively by listening to sounds and spoken words, according to Felder & Silverman [22]. Consequently, the system prefers to deliver video lectures to this learner. Nevertheless, in this example video lectures are only available in high quality, i.e. they require high bitrate. This example also has text lectures available.

The *soft goals satisficing view* defines two claims; these claims affect the expected level of the soft goal *WaitingTimeOptimization* for the *Video* operationalization, as illustrated in Figure 6. The claims *CL9* and *CL13* require *Video* refined from the goal *IndividualizeMultimediaDelivery*. The *soft goals satisficing view* also expresses, with soft dependency SD4, the required level of the soft goal *WaitingTimeOptimization*. The claims and SD required in this example can be expressed in a simplified way as:

$$CL8\_selected \Leftrightarrow Text\_selected \vee Graphics\_Selected$$

$$CL8\_selected \Rightarrow WaitingTimeOptimization\_ExpectedLevel = 4$$

$$CL9\_selected \Leftrightarrow Video\_selected \wedge VideoBitRate = low$$

$$CL9\_selected \Rightarrow WaitingTimeOptimization\_ExpectedLevel = 1$$

$$CL13\_selected \Leftrightarrow Video\_selected \wedge VideoBitRate = high$$

$$CL13\_selected \Rightarrow WaitingTimeOptimization\_ExpectedLevel = 4$$

$$SD4\_selected \Leftrightarrow NetworkSpeed=low$$

$$SD4\_selected \Rightarrow WaitingTimeOptimization\_RequieredLevel = 4$$

Claim *CL8* indicates the highly positive influence on the satisfaction of *WaitingTimeOptimization* (expected level = 4) when the system delivers text or graphics. Claim *CL9* indicates the highly positive influence on over the satisfaction of *WaitingTimeOptimization* (expected level = 4) when the system delivers video, and the *VideoBitRate* is low. Claim *CL13* indicates the expected denial in the satisfaction of *WaitingTimeOptimization* (expected level = 1) when the system delivers video, and the *VideoBitRate* is high.

Soft dependency *SD1* specifies that if the *NetworkSpeed* connection is low then the waiting time optimization should be fully satisfied.

In this example we consider two context scenarios, the first *NetworkSpeed* variable is *high* and second *NetworkSpeed* variable is *low*. In both scenarios, we guaranty the *VideoBitRate* in *high*. In the former, no constraints exist over the soft goal *WaitingTimeOptimization*; i.e., the unique soft dependency, *SD1*, for this soft goal is not selected. Figure 8, on the left, presents this configuration in VariaMos.

On the contrary in the latter, the system needs to satisfice the soft goal *WaitingTimeOptimization*, regarding the network speed is low. The initial configuration uses text as *CL8* also has a highly positive influence on *WaitingTimeOptimization*. Figure 8, on the right, presents this configuration in VariaMos. Nevertheless, the monitoring constantly takes the bandwidth measurement. If it sees that the bandwidth is getting better, it notifies the system to change to the *Video* operationalization as this is the preferred option for the learner´s learning style.

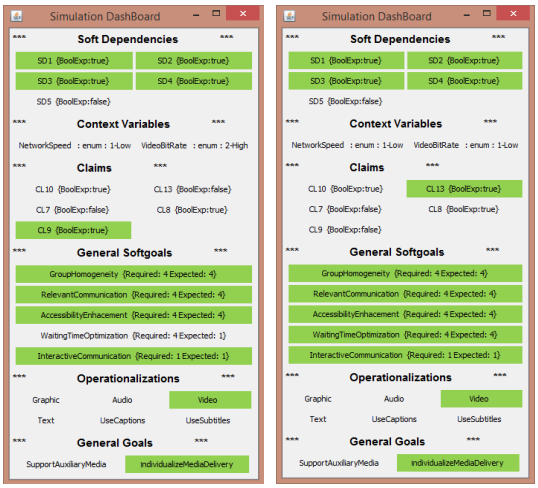

Figure 8: Configuration scenarios for Verbal Learner using VariaMos tool

Frequency of Use Example

Let us suppose we have a learner that is not a frequent visitor of the system. In this case, the system should maintain relevant communications using external systems, such as *ExternalEmail* or *SMS*. Nevertheless, if the user becomes a frequent visitor, the system should encourage interactive communications.

The claims and SD required in this example can be expressed in a simplified way as:

$$Cl1\_selected \Leftrightarrow SMS\_selected \vee ExternalMail\_selected$$

$$CL1\_selected \Rightarrow RelevantCommunications\_ExpectedLevel = 4$$

$$CL1\_selected \Rightarrow InteractiveCommunications\_ExpectedLevel = 1$$

$$Cl2\_selected \Leftrightarrow Chat\_selected \vee InternalMail\_selected \vee Forum\_selected$$

$$CL2\_selected \Rightarrow RelevantCommunications\_ExpectedLevel = 1$$

$$CL2\_selected \Rightarrow InteractiveCommunications\_ExpectedLevel = 4$$

$$SD2\_selected \Leftrightarrow (FrequencyOfUse = Low)$$

$$SD2\_selected \Rightarrow RelevantCommunicacions\_RequieredLevel = 4$$

$$SD5\_selected \Leftrightarrow (FrequencyOfUse = High)$$

$$SD5\_selected \Rightarrow InteractiveCommunicacions\_RequieredLevel = 4$$

In this example we consider two context scenarios, the first *FrequencyOfUse* variable is *low* and the second *FrequencyOfUse* variable is *high*. In the former, the system needs to satisfice the soft goal *RelevantCommunicacion* when the frequency of use is low. Figure 9, on the left, presents this configuration in VariaMos. In the latter, the system needs to satisfice the soft goal *InteractiveCommunicacion*, regarding the frequency of use is high. The initial configuration uses *ExternalMail* or *SMS* as *CL2* also has a highly positive influence on *RelevantCommunicacion*. Nevertheless, the monitoring constantly takes monitor the access logs to identify frequent users. Figure 9 presents the configuration in VariaMos on the right. If it sees that the user becomes a frequent user, it notifies the system to change to the *Chat*, *InternalMail* or *Forum* operationalization as this satisfice the soft goal *InteractiveCommunication*.

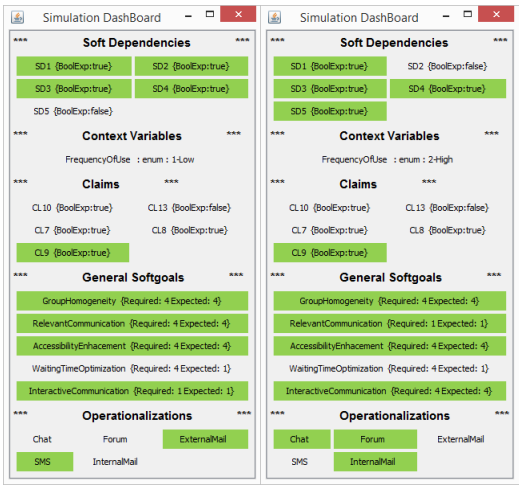

Figure 9: Configuration scenarios for Frequency of Use using VariaMos tool

Configurations of a scenario

From the generated optimal configurations, the e-learning system can decide the adaptation strategy to follow when the requirements monitoring detects a satisfaction problem. For Instance, we chose a scenario with the following context variables: *GeographicChangeSpace* is *true*, *IsLongStay* is *true*, *FrequencyOfUse* is *low*, *NoisyEnvironment* is *false* and *NetworkSpeed* is *low*. For this scenario, we obtained 40 relevant configurations. They are relevant because they ignore the selection between operationalization goals without influence in the adaptation, for example, the selection of *Chat*, *Forum* or *InternalMail* have the same influence.

From the scenario, we are interested in the visual learner that prefers video for learning materials. Thus, Table 1 presents the aggregated configurations for video, from 40 configurations, only 8 remain relevant. To simplify the table, we do not display not selectable claims. The columns of Table 1 are then organized as follow: Column 1 presents the *VideoBitRate* variable with Low and High values. Columns 2-9 represent the selection (Y) or rejection (N) of one or more operationalizations, in the last two cases the selection is irrelevant (-). Columns 10-15 represent the activation (Y) or deactivation (N) of claims. Columns 16-20 represent the soft dependency activation (Y) or deactivation (N), columns 21-25 represent the soft goals required satisfaction levels and columns 26-30 represent satisfaction (Y) or denial (N) of the soft goals.

Finally, in the table, the first configuration is the only alternative that satisfice all the soft goals. As the scenario defines low *NetworkSpeed*, the alternative requires low *VideoBitRate* to satisfice the soft goal *WaitingTimeOptimization*. Other alternatives do not satisfy all soft goals but is also possible to select them.

Table 1: Configurations for verbal learner style in a particular scenario

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
VideoBitRate	Chat OR Forum Or InternalMail	SMS OR ExternalMail	Audio XOR Graphics XOR Text	Video	Online Discussion Group	Physical Self-Help Gr	Use Captions	Use Subtitles	CL1	CL2	CL3	CL4	CL9	CL13	SD1	SD2	SD3	SD4	SD5	GroupHomogeneity - Required Level	RelevantCommunication - Required Level	AccesibilityEnhacement - Required Level	WaitingTimeOptimization - Required Level	InteractiveCommunication - Required Level	GroupHomogeneity - Satisfice	RelevantCommunication - Satisfice	AccesibilityEnhacement - satisfice	WaitingTimeOptimization - Satisfice	InteractiveCommunication - satisfice
L	**N**	**Y**	**N**	**Y**	**N**	**Y**	**-**	**-**	**N**	**Y**	**Y**	**N**	**N**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**4**	**4**	**4**	**4**	**1**	**Y**	**Y**	**Y**	**Y**	**Y**
L	N	Y	N	Y	Y	N	-	-	N	Y	N	Y	N	Y	Y	Y	Y	Y	N	4	4	4	4	1	N	Y	Y	Y	Y
L	Y	N	N	Y	N	Y	-	-	Y	N	Y	N	N	Y	Y	Y	Y	Y	N	4	4	4	4	1	Y	N	Y	Y	Y
L	Y	N	N	Y	Y	N	-	-	Y	N	N	Y	N	Y	Y	Y	Y	Y	N	4	4	4	4	1	N	N	Y	Y	Y
H	N	Y	N	Y	N	Y	-	-	N	Y	Y	N	Y	Y	Y	Y	Y	Y	N	4	4	4	4	1	Y	Y	Y	N	Y
H	N	Y	N	Y	Y	N	-	-	N	Y	N	Y	Y	Y	Y	Y	Y	Y	N	4	4	4	4	1	N	Y	Y	N	Y
H	Y	N	N	Y	N	Y	-	-	Y	N	Y	N	Y	Y	Y	Y	Y	Y	N	4	4	4	4	1	Y	N	Y	N	Y
H	Y	N	N	Y	Y	N	-	-	Y	N	N	Y	Y	Y	Y	Y	Y	Y	N	4	4	4	4	1	N	N	Y	N	Y

5 RELATED WORK

This research work is at the intersection of Dynamic Adaptive Systems (DASs) and adaptive e-learning systems.

While the literature widely addresses DAS development, we are interested by runtime monitoring techniques to detect deviations at runtime. The concept of requirements monitoring originates with Fickas and Feather [24], which means gather information that can determine whether and to what degree a running system is meeting its requirements. Later in other paper [25], they proposed a runtime event-monitoring system that combines KAOS goal driven specification and FLEA (Formal Language for Expressing Temporal Combinations of Events). FLEA translates KAOS assertions in temporal combinations of events. Whenever an event occurs, it is automatically stored in a database and its defined action is executed. Subsequently, Robinson [26] extended Feather's model to address the problem of web services monitoring. The KAOS model was used to define requirements model at design time while a database was used to support the runtime analysis. Goldsby *et al.* [27] used i* goal model for requirements specification, but they assume that they have no guarantee that such DAS could perform requirements engineering at runtime. Finally, despite the wide use of goal models like KAOS and i* to represent requirements specification, however, these models only help list different strategies that can be performed and they do not support system evolution. Consequently, the process of requirements monitoring using these approaches still depends on a static monitoring strategy that enumerate all possible alternative behaviour. To address this gap, Baresi *et al.* [28] have modified KAOS goal specification and extended it by adaptive goals. The idea consists of using a membership function to assess satisfaction requirements of goals at runtime and when some requirements reach a boundary trigger suitable adaptation actions (adding/removing goals). The system is then guided by a fuzzy logic operators, which were already used in RELAX [29] a declarative approach for specifying requirements for DAS. In contrast to use declarative approaches accounting for more flexibility, Sawyer *et al.* [17] proposed an approach to represent

and transform an extended goal driven requirements model into a constraint program to manage the configuration of Self-adaptive systems. In the same line with this approach we propose to use REFAS language for requirements specification and automatically transform the model into a constraint satisfaction problem. Our approach has the advantage that is guided by the graphical tool VariaMos, which allows the designer a performance modelling and verification of the specification through simulations as presented in this paper.

In addition to goal-driven requirements, formalisms like probabilistic and real-time logics have been widely used for requirements specification whose formal verification can then be carried out using probabilistic model checker [30]. Nevertheless, the formalisation and inference using such techniques are always complex and resources consuming [31]. In line with these approaches, Calinescu *et al. [30]*, proposed a Dynamic QoS management and optimization in service-based systems. They adopt Markov model to determine quantitatively the reliability and performance quality metrics of service-based systems, which are then formally and automatically analyzed to identify and enforce optimal system configurations. *Ghezzi et al.* [32] presented SPY@RUNTIME approach that focus on verification of system behaviour at runtime. This latter is represented by finite state automata. The tool can detect change and notify system designer by continuously checking the executed program against a specification. In contrast to runtime verification that reacts just after anomalies are detected, our purpose is to predict the deviations.

On the other hand, significant research effort has been devoted to support the development of adaptive e-learning systems including techniques for learning monitoring. To the best of our knowledge, this latter consist of just recording student interaction and analyzing log data to offer a visualization that help instructors become aware of what is happening in distance learning classes ([33][34][35]), but it remains to see if the instructors are willing to invest significant time and effort required for continuous monitoring. In contrast, to passive monitoring, our approach leads to detect deviations from desired learning requirements at runtime and notify the system with a list of alternative configurations. So it can adjust its strategies.

6 CONCLUSION

In this paper, an automated approach for requirements monitoring was introduced to explicitly support the lack of dynamic requirements monitoring in adaptive e-learning systems, which allow these systems to respond flexibly to dynamic learners needs and be aware of their environments. Our approach defines and implements the autonomic computing MAPE loop architecture. The loop control continuously checks the requirements satisfaction under changing context using a CSP solver. The strengths of our approach are that the requirements model is specified by means of a language that combines goal-driven requirements with features and claims, and it is fully automated. Thus, the requirements model is automatically transformed into a CSP, and the verification of the requirements model is carried out through simulations using VariaMos tool. With this automation, we have two advantages: first, this avoids loss of information and misinterpretation of the specification during the transformation. Secondly, it explores the power to reason about goals to represent complex constraints and expressiveness of constraint programming.

As the next step, we first aim to improve the generated constraint program and implement our monitoring framework and then study its impact on the learning process and learners' satisfaction using real world data.

While the focus of this paper was add "self-adaptability" to adaptive e-learning systems by allowing the evolution of these systems at runtime using constraint programming, future work will explore the proposed approach in the verification and the evaluation of adaptive e-learning systems.

REFERENCES

[1] Mulwa, C., Lawless, S., Sharp, M., Arnedillo-Sanchez, I., & Wade, V. (2010, October). Adaptive educational hypermedia systems in technology enhanced learning: a literature review. In Proceedings of the 2010 ACM conference on Information technology education (pp. 73-84). ACM.

[2] Brusilovsky, P., & Peylo, C. (2003). Adaptive and intelligent web-based educational systems. International Journal of Artificial Intelligence in Education, 13(2), 159-172.

[3] Brown, E., Cristea, A., Stewart, C., & Brailsford, T. (2005). Patterns in authoring of adaptive educational hypermedia: a taxonomy of learning styles. Educational Technology & Society, 8(3), 77–90.

[4] 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.

[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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
[12] 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.
[13] Dardenne, A., Van Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. Science of computer programming, 20(1), 3-50.
[14] Brusilovsky, P., Farzan, R., & Ahn, J. W. (2006). Layered evaluation of adaptive search. In Workshop on Evaluating Exploratory Search Systems at SIGIR06 (pp. 11-13).
[15] Gena, C., & Weibelzahl, S. (2007). Usability engineering for the adaptive web. In The adaptive web (pp. 720-762). Springer-Verlag.
[16] Paramythis, A., Weibelzahl, S., & Masthoff, J. (2010). Layered evaluation of interactive adaptive systems: framework and formative methods. User Modeling and User-Adapted Interaction, 20(5), 383-453.
[17] 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.
[18]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.
[19] Mazo, R., Salinesi, C., & Diaz, D. (2012, June). VariaMos: a Tool for Product Line Driven Systems Engineering with a Constraint Based Approach. In CAiSE Forum (pp. 147-154).
[20] Mazo, R., Salinesi, C., & Diaz, D. (2011). Abstract Constraints: A General Framework for Solver-Independent Reasoning on Product Line Models. INSIGHT-Journal of International Council on Systems Engineering (INCOSE), 14(4), 22.
[21] 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.
[22] Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. Engineering Education, 75(7), 674-681.
[23] 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.
[24] Fickas, S., & Feather, M. S. (1995, March). Requirements monitoring in dynamic environments. In Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on (pp. 140-147). IEEE.
[25] Feather, M. S., Fickas, S., Van Lamsweerde, A., & Ponsard, C. (1998, April). Reconciling system requirements and runtime behavior. In Proceedings of the 9th international workshop on Software specification and design (p. 50). IEEE Computer Society.

[26] Robinson, W. N. (2003, September). Monitoring web service requirements. In Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International (pp. 65-74). IEEE.
[27] Goldsby, H. J., Sawyer, P., Bencomo, N., Cheng, B. H., & Hughes, D. (2008, March). 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.
[28] Baresi, L., & Pasquale, L. (2010, July). Adaptive goals for self-adaptive service compositions. In Web Services (ICWS), 2010 IEEE international conference on (pp. 353-360). IEEE.
[29] Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H., & Bruel, J. M. (2009, August). Relax: Incorporating uncertainty into the specification of self-adaptive systems. In Requirements Engineering Conference, 2009. RE'09. 17th IEEE International (pp. 79-88). IEEE.
[30] 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.
[31] Lalanda, P., McCann, J. A., & Diaconescu, A. (2013). Autonomic Computing. Springer London Limited.
[32] Ghezzi, C., Mocci, A., & Sangiorgio, M. (2012, June). Runtime monitoring of component changes with spy@ runtime. In Software Engineering (ICSE), 2012 34th International Conference on (pp. 1403-1406). IEEE.
[33] Graphvis (2004) visualization tool monitoring group communications in order to help instructors detect collaboration problems.
[34] 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.
[35] 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.