Name of Journal will go here

An Official Publication of the Informing Science Institute
InformingScience.org

Journal URL will go here

Volume x, 20xx

REQUIREMENTS MONITORING AND DIAGNOSIS FOR IMPROVING ADAPTIVE E-LEARNING SYSTEMS DESIGN

[1,2] Lamiae DOUNAS* [1]CRI, University Paris 1 Panthéon-Sorbonne, Paris, France. lamiae.bourkiza@malix.univ-paris1.fr

[2]LIIAN, Faculty of Sciences Dhar el Mehraz USMBA, Fez , Morocco.

[1]Camille SALINESI* camille.salinesi@univ-paris1.fr

[2]Omar EL BEQQALI* Omar.beqqali@usmba.ac.ma

* Corresponding author

ABSTRACT

Aim/Purpose	In this paper, we highlight the need to monitor and diagnose adaptive e-learning systems requirements at runtime to develop a better understanding of their behavior during learning activities and improve their design. Our focus is to reveal which learning requirements the adaptive system is satisfying while still evolving and provide specific recommendations regarding what actions should be taken and which relevant features are needed to help meet the specified learning requirements.
Background	Adaptive e-learning systems research has long focused on user modeling and social learning to personalize each learner experience, while fewer instruments are reported to assess the quality of the solutions provided by such adaptive systems and to investigate their design problems. The design problems may emerge due to ever-evolving requirements being statically specified at design stages and to the changing environments that can be difficult to control and observe. The combination of some or all of these factors can lead to a definition of inconsistent or insufficient adaptation rules, which in turn may prevent these systems from providing appropriate resources to learners even if the needed ones have been accounted for within the knowledge space.
Methodology	An empirical study has been performed to check and validate the behavior of a real-world adaptive e-learning system under four stated requirements. The study used a novel monitoring and diagnosing tool that reads the collected data from the system and checks its behavior against constraints that are derived automatically from the requirements specification.
Contribution	The results provide statistical insights and highlight some issues related to requirements compliance at runtime; which helped us detect unforeseen instructional design issues.
Recommendations for Practitioners	The study suggests that diagnosing requirements compliance at runtime can be an essential means to increase the confidence about their adaptive e-learning systems

	capabilities at runtime.
Recommendation for Researchers	The study suggests that further research for developing specific indicators related to requirements compliance, is needed in the field of adaptive e-learning systems.
Future Research	Future work includes the study of possible improvement of our diagnostic tool using probabilistic reasoning.
Keywords	Runtime requirements; requirements compliance; adaptive e-learning system; learning analytics; goal modeling; adaptive e-learning system; evaluation.

INTRODUCTION

Research in adaptive e-learning systems (AESs) can be traced back to the early 1970s. At that time, the field of computer-based learning has investigated combining research in Artificial intelligent (AI) and education. As a result, intelligent tutoring systems (ITSs) were born. Their goals were mimicking human tutoring capabilities to guide learners during the problem-solving process by personalized feedback and suggestions. Later, and by capitalizing on the technological advancements, especially in the field of computers, many researchers with different backgrounds (pedagogy, psychology, sociology, etc.), have contributed to the development of a new generation of adaptive online educational systems like ELM-ART (Brusilovsky et al., 1996), InterBook(Brusilovsky et al., 1997), AHA(De Bra et al., 2002), SQL-Tutor(Mitrovic, 1998), iWeaver(Wolf, 2003), Knewton[1], and INSPIREus(Papanikolaou, 2015) that are not necessarily tied to one specific curriculum like ITS. At their core, these systems are intended to allow interaction and adapt course resources and other learning activities to offer a unique experience to each learner. To achieve this objective, an AES relies on an adaptation model that implements rules describing the adaptation strategies for each learner's situation. These rules are basically of two types: (i) content selection rules that select appropriate resources from the knowledge space, and (ii) navigation rules that sequence the selected resources based on each learner's characteristics. The adaptation strategies are generally implemented as rule-based systems like ELM-ART and SQL-Tutor, or recommendations based systems like in (Tsiriga & Virvou, 2002; Jurado et al., 2008; Wang & Yang & Wen, 2009; Baker et al., 2010), which take advantages of AI and machine learning techniques including tree learning methods, Bayesian network, probabilistic learning methods, and case-based learning approaches to analyze learners data, identify gaps in the knowledge and redirect each learner to new topics when appropriate.

Although extensive research on adaptive learning has significantly improved different aspects of access, most studies have focused on enhancing the adaptation based on user modeling and social learning at the expense of design issues that remain underexplored (Karampiperis & Sampson, 2005), (Graf et al., 2012) and (Sampson & Karampiperis, 2012). The design issues are emerging due to evolving requirements being statically specified at design time, and to the inherent uncertainty of AESs. This uncertainty may arise from different sources, such varying learners' backgrounds and needs, dynamic changing environments that are hard to control and observe, incomplete information at design time, limited sensors capabilities and unexpected system behavior. The combination of some or all of these factors can lead to a definition of inconsistent or insufficient adaptation rules, which in turn may prevent these systems from providing appropriate resources to learners even if the needed ones have been accounted for within the knowledge space.

In this paper, we highlight the need to monitor AESs requirements at runtime to improve their design and to develop a better understanding of their behavior during learning activities. For this purpose, we have conducted an empirical study on 6165 historical traces from INSPIREus (Papanikolaou, 2015), a real-world AES. The analysis was based mainly on a monitoring and diagnosing tool that we have previously proposed in (Dounas et al., 2015) that we call here "RMAS" (Runtime Monitoring for Adaptive Systems). RMAS reads the collected data and checks the system behavior against constraints that are derived automatically from the requirements specification. The results indicate how and whether learners' requirements are effectively met under uncertainty. We have focused principally on the uncertainty related

[1] https://www.knewton.com

to (i) changes in the requirements, which are due for example to an evolution of the monitored system, (ii) unforeseen configurations at the specification time, and (iii) changes in the operating environment.

The paper addresses the following research questions:

1. Based on all the configurations provided by the system during learning activities, what is the overall accuracy of an AES?
2. How well does an AES fulfill its requirements?
3. Why are some requirements not fully (or at least partially) satisfied by the AES?
4. What are the typical features used during a successful learning process?

The ultimate goal of our research is to be able to generate recommendations that have the potential to guide AESs concerning what actions can be taken and which relevant features are needed to meet the specified requirements.

The remainder of this paper is structured as follows. Section 2 presents the basic concepts that are necessary to understand our research problem. Section 3 describes our case study. Section 4 presents the study methods. Section 5 presents and discusses the empirical results. Section 6 gives a brief discussion of related work. Section 7 discusses some of the limitations associated with the decisions of the experimental design. Section 8 concludes by briefly presenting some of the study's implications for research and the practice.

BACKGROUND

In this section, we briefly outline the basic notions related to requirements monitoring mechanisms, which we believe are necessary to explain this study later in the paper.

GOAL-ORIENTED REQUIREMENTS ENGINEERING (GORE)

There are many approaches in the literature that support the requirements specification, among which the most commonly used is GORE. GORE attempts to specify requirements to be monitored using goal-oriented models. Dardenne et al., (1993) define a goal as a high–level objective to be achieved by a software system. Goals can be categorized into functional (hard) goals and non-functional (soft) goals. A hard goal depends on exact criteria to determine whether it has been satisfied or not, while a soft goal is related to quality objectives that guide the search of an acceptable level of soft goal satisfaction when the optimal level cannot be reached. This process often involves a trade-off with other soft goals. In this view, the requirements can be seen as a special kind of goal that restraint system behavior. "AES shall personalize the course presentation" is an example of hard AES goals, while "AES shall provide each learner with course materials that match his/her learning style" is an example of a requirement related to that goal.

The use of GORE to specify the requirements offers many benefits such as: (1) it describes the relationships between a system and its environment and not just what the system should do. (2) It provides traceability links from high-level goals to low-level operational requirements (Dardenne et al., 1993). And (3) it offers means to drive a runtime adaptation, by reasoning between alternative solutions and managing the trade-offs of non-functional requirements (NFRs) or soft-goals such as performance and availability for each context.

Various proposals have also been made in GORE. In KAOS (Van Lamsweerde et al., 1998), a Goal refinement methodology and a formal refinement have been defined to help the analyst produce a complete and consistent goal specification. Goals are specified using AND/OR refinements of goals into sub-goals. These goals are to be achieved through the cooperation of various agents that may include software components, external devices or humans. Each agent is assigned with a responsibility that restricts its behavior to ensure its end-goal. The responsibility assignment is a stopping criterion for the refinement process. To formally specify the goals, linear real-time temporal logic operators are proposed, and which correspond to four temporal goals patterns: Achieve, Maintain, Avoid and Cease. These patterns allow the analyst to specify properties involving real-time deadlines. Alternatively, NFR framework (Mylopoulos et al., 1992), TROPOS (Fuxman et al., 2004), i* (Yu, 1997), and REFAS (Munoz-Fernández et al., 2014) approaches have been proposed with the aim to concentrate the software

development process on non-functional requirements. The methodologies followed in these approaches consist of identifying non-functional requirements in term of soft goals using AND/OR refinement process; similar to what was proposed in KAOS. The soft-goals are then used as selection criteria for choosing the alternative process configuration that best meets the non-functional requirements of the system.

REQUIREMENTS MONITORING

Monitoring consists of gathering and analyzing information about a software system while it is running. Recently, there has been a growing interest in monitoring requirements at runtime (Robinson, 2010), which seeks to continuously check if a running system meets its requirements specifications. The monitoring process operates by interpreting low-level system events as contributors to eventual requirements satisfaction or violations. In order to achieve this, the requirements can be either hard coded within the given approach or formally specified and checked against the monitored data. The former is suitable for monitoring isolated and small numbers of requirements, which can be written directly by the developer. However, the latter becomes more suitable to avoid significant errors when the number of the requirements increases, especially, when it comes to managing conflicting requirements.

In the context of our on-going research activities on runtime adaptive systems requirements monitoring, we have proposed RMAS (Dounas et al., 2015), a runtime requirements monitoring tool for adaptive systems that continuously assess the extent to which a monitored system satisfies its requirements specification during operation.

RMAS supports requirements being specified as a goal-oriented model using REFAS language. In this notation, functional requirements are represented as variability goals model (see Appendix A), by AND/OR refinement of goals into high-level hard goals to capture their hierarchical relationship and constraints. The refinement stops when 'features' can operationalize leaf-goals. Features represent rules and services used by the system's adaptation strategies. In addition, REFAS represents assumptions about the contexts and their implications over the soft goals satisficement (i.e. sufficient degree of satisfaction) through soft dependencies and claims. A 'Claim' is a predicate that indicates assumptions made at design time about the expected soft-goals satisficement levels from each feature, while a 'Soft-dependency' is a predicate that indicates the required soft-goals satisficement levels for particular context values. The trade-off between claims and soft-dependencies leads to reason about alternative adaptation strategies, and the impact of each of them on soft goals satisficement.

RMAS defines transformation rules to translate the specification into a constraints logic program over finite domain CLP (FD) that carried out the runtime reasoning. The runtime reasoning interprets each selected adaptation strategy as a configuration of features C={F1, F2...Fn}. The extracted configuration is evaluated as valid when all the mandatory goals are satisfied, while it is assessed as optimal if it satisfies as many soft goals as possible.

The reasoning about requirements at runtime can then be assimilated into a constraint satisfaction problem solving, where a violation means that the configuration at hand does not satisfy the constraints in the CLP. While, an improvement of the configuration consists of generating solutions that optimize as many constraints as possible.

RMAS is a generic tool and can be applied to any kind of adaptive system including adaptive e-learning systems. Therefore, we used this tool to analyze AESs requirements in this study.

CASE STUDY DESCRIPTION

First of all, a review of the literature of adaptive e-learning systems was undertaken to select an AES for the experiment from scientific articles. Among the AESs found, INSPIREus system (Papanikolaou et al., 2003; Papanikolaou, 2015) has been prioritized as it offers an adaptation based on various criteria along with the fact that it is maintained online.

INSPIREus is an adaptive educational hypermedia environment that provides personalized content and adaptive navigation support for each learner. Besides, INSPIREus offers collaborative functionalities and a flexible authoring process that allows users to freely explore the course content and reflect their pedagogical perspective on content development.

The knowledge modules in INSPIREus are organized in three performance levels: (1) the 'Remember' level is related to the ability of learners to recall their knowledge, (2) the 'Use' level is related to the ability

of learners to apply theory through case study, and (3) the 'Find' level is related to the ability of learners to propose and solve original problems.

The adaptation criteria in INSPIREus include three knowledge levels 'KL' (low, average and high) and four learning styles 'LS' from Honey and Mumford (1992) model (reflector, activist, theorist, and pragmatist). All learners are provided with the same knowledge modules contents including theory, examples, exercises, and activities using computer simulation; however, the method and order of their presentation on each page is personalized for each learner (see Table1).

Table 1: the adaptive presentation strategies of the knowledge modules (method and order of appearance) in INSPIREus : Q stands for "Question", **T** for "Theory", **Ex** for "Example", **E** for "Exercise", and **A** for "Activity".

Performance levels / learning style	**Remember level**	**Use level**
Activist	(Q, E, T)	(A, Ex, T, E)
Reflector	(T, Ex, Q)	(Ex, T, E, A)
Theorist	(Q, T, Ex)	(T, Ex, E, A)
Pragmatist	(Ex, T, Q)	(E, Ex, T, A)

The main objective is to enhance learning by matching the dominant learning preferences of the learners with the appropriate sequencing of educational material. As depicted in Table 1, there are two main presentation strategies of the modules on an educational material page at remember level of performance: (i) Inquisitory presentation starts with a question aiming to attract learner's attention, then example or theory modules (for activist and theorist respectively) are provided to answer that question. (ii) Expository presentation starts with example or theory modules (for pragmatist and reflector respectively), then the same question appears as a self-assessment question to attract their reflection. On the other hand, four adaptation strategies of the modules are applied on an educational material page at the Use level of performance: (1) Activity-based presentation (A, Ex, T, E), introduces the module "activity" at the top of page, which is followed by the other modules, for an activist learner (who prefers to assimilate new information through activities). (2) Example-based presentation (Ex, T, E, A), introduces the module "example" at the top of page, which is followed by the other modules, for a reflector learner (who learns best by watching people and thinking about what is happening). Similarly, (3) Exercise-based presentation (E, Ex, T, A), starts with "exercises" module, for a pragmatist learner (who prefers to learn through practice). And (4) theory-based presentation (T, Ex, E, A), starts with "theory" module, for a theorist learner (he who learns through theory).

METHODOLOGY

Our research approach focuses on analyzing AES compliance with its requirements when applying its adaptation strategies. Specifically, we explore the data generated by RMAS tool through diagnosing the execution traces from INSPIREus to answer the research questions.

In the rest of this section, we first describe the monitored requirements and the collected data. Then, we detail methods of our experimental research.

SAMPLE OF REQUIREMENTS

The requirements sample used for our experiment consists of four requirements, which are stemming from several reported issues in AESs and amenable to requirements monitoring:

Req1: *The AES shall provide learners with educational materials (EMs) that match their learning styles.*
As INSPIREus allows learners to have instructional control over the system (including modifying their learning style), there is a need to control the EMs delivery. Thus, the system should be more flexible to support the change and generate educational materials that fit their effective learning styles. This fact can be detected using system's log files. For instance, if a learner changes his/her learning style to "reflector" and the history of learner's selections of educational materials indicates that this learner spends most of

his/her study time on activities, then the system should generate activity-based educational materials for him/her because their effective learning style is "activist" instead of "reflector".

Req2: *The AES shall balance between navigation freedom and guidance to improve the curriculum sequencing.*
INSPIREus should track changes made by each learner in the knowledge level (KL) and activation/deactivation of system's adaptive behavior. We suppose that if KL ∈ {low, avg} then the AES shall enrich the domain presented to him/her or encourage him/her to activate "system adaptive behavior" if s/he is lost during the learning. We identify a lost learner when s/he spends more than twice the allocated time for a given concept.

Req3: *The AES shall enhance the communication between learners.*
As INSPIREus supports collaborative learning, it should help learners develop participation skills including creating topics and interacting in other topics within the forum, and guide them to more compatible learning groups whenever a perturbation among the desired state of interaction or in case of failure.

Req4: *the AES shall manage the sharing of control between the system and learners.*
Usually, learner models are hidden from learners in AESs. Several studies (Devoper & Quintin, 1992; Specht, 1998) have investigated the impact of allowing learners to modify their learner models like changing their knowledge levels, their learning styles or activating/deactivating system's adaptive behavior. These studies reveal that because most learners are usually unsure of their needs, the decision to open the learner model leads to failure and wrong decisions by learners. However, restricting them would push learners to lose trust in such a system and consequently give up using it. Accordingly, the amount of freedom should be dependent on the knowledge state of learners and the time spent on educational materials. For instance, if a learner who spent a great amount of time in learning without any progress deactivates system's adaptive behavior, the system should regain the control to guide him/her by encouraging them to activate the adaptive option or intervene directly by restricting/recommending additional educational materials for them.

DATA SAMPLE

Data for this study were gathered from INSPIREus. 21 learners have enrolled into a course, running for three months starting March 2016, given at Department of Informatics and Telecommunications of the University of Athens. As described in Table 2, the data sample includes three data sets.

Table 2: Datasets description

Name	Description
Log file	Describes all sessions log details of learners' interaction with INSPIREus. Specifically, the log file takes the form of an excel file with nine columns including 'StudentID', 'sessionID', 'Visited module', 'Operations' (represent changes made by the learners or by the system).
Assessment data	Stores information about the final score of each studied concept. It takes the form of an excel file with 5 columns, StudentID, scenario title, concept title, and score which denotes the final score for each concept.
Materials description file	Describes the available resource materials for each concept of a scenario, their allocated study time and their type (activity, example, exercise, question or theory) and level (remember, use or find).

The detailed log files recorded by INSPIREus constitute valuable data about the behavior of the learners and the system as well as the interactions between the two. As for the assessment data set, it used to determine whether learners have understood the covered materials. Finally, the third data set describes all the resources available for the learners.

DATA ANALYSIS

The data processing for this study includes three phases (see Figure 1):

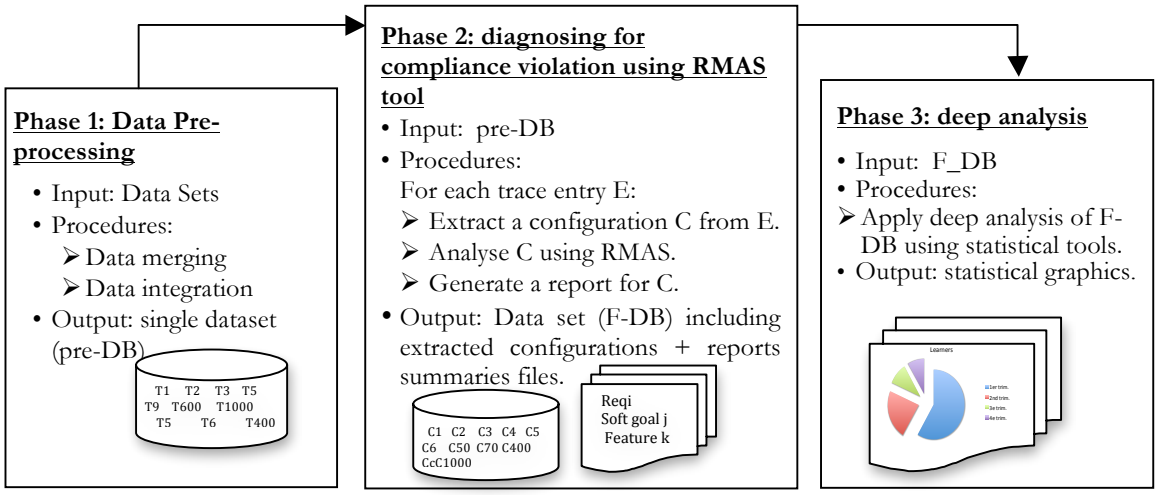

Figure 1: data processing phases

Phase 1

This phase aims to prepare the data before going into details. First, we enriched the data with additional information about the adaptation strategies of INSPIREus. These strategies are available in a detailed description of INSPIREus given in Papanikolaou et al. (2003). Then, Talend Open studio tool was used for cleaning, integrating, and merging the three datasets into a single dataset in excel format. StudentID, SessionID, and TraceID were used to identify each trace entry, which makes a total of 6165 entries in the data set.

After data pre-processing, 1 out of 21 learners were eliminated because s/he had not engaged in the learning (study time < 30min && no scores recorded). A preliminary analysis of the data identified 8 activists, 2 pragmatists, 4 reflectors and 6 theorists. Only 4 learners preferred to manually specify their learning styles in the learner model, while 16 learners took the questionnaire of Honey and Mumford when they logged in for the first time. The course proposes 3 concepts to study. The average study time was 440,53 min (SD=422,07) against allocated time (time proposed by a teacher) = 394 min. Generally, all learners have passed the final exam with an average of 11.67 out of 12 (SD=1,05).

Phase 2

In the second step, our goal was to assess how well INSPIREus is meeting the requirements at runtime, through monitoring and diagnosing the requirements at different levels of granularity, from high-level requirements (goals and soft goals) to low-level ones (components or features). To this end, we have developed a prototype of RMAS to analyze the trace entries in the pre-processed dataset against the requirements sample. The monitoring tool was fully developed in Java with the Eclipse IDE. It takes as inputs: (a) The execution traces from INSPIREus, and (b) The requirements specification of INSPIREus.

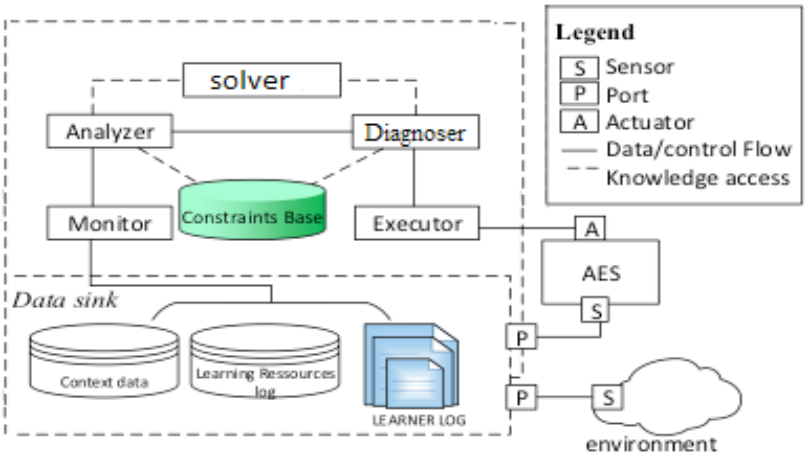

Figure 2: RMAS conceptual architecture for AESs

Figure 2 depicts the conceptual architecture of RMAS and the core components interactions between the managed AES and the RMAS during the learning process:

- Data sinks registers callbacks whenever measurements of certain nodes sensors/probes have been received and notifies the change to the monitor module.
- The monitor listens to data sinks; first, it preprocesses the collected data (e.g. normalization of data, extraction of configuration) and then notifies the change to the next component.
- The analyzer calls the solver to check the configuration under consideration, and notifies violation (satisfaction problem), if any to the diagnoser.
- Diagnoser identifies unsatisfied requirements and denial features and sends the result to next component.
- Executor could either act upon the monitored AES and execute remediation strategies through actuators (this option is not implemented yet) or notify warning to the managed systems in the form of a report describing the monitored requirements in term of functional requirements (features in the leaf goals) and non-functional requirements (soft-goals).

Figure 3 shows a prototype of the generated report by the executor component. Roughly speaking, for each monitored requirement (for this experiment, four requirements was monitored), the report describes whether each feature in the configuration under consideration is relevant or not (denial) and how it is satisficing the related soft goals to the requirement at hand. Thus, a soft goal is qualified as 'unsatisficeable' if the obtained satisfaction level is less than the required satisfaction level, while it is 'satisficed' in the opposite case. A feature is qualified as 'relevant' if it is not conflicting with other features in the same configuration and if it contributes to the soft goals satisficement; otherwise, it is qualified as 'irrelevant' (denial).

Requirement 1
Soft goal 1: [satisficed or not] satisfaction level
Feature i: [relevant or not]
[…]
Soft goal 2: [satisficed or not] satisfaction level
Feature j: [relevant or not]
[…]
Requirement 2
Soft goal 3: [satisficed or not] satisfaction level
Feature k: [relevant or not]
[…]
Requirement 3
Soft goal 4: [satisficed or not] satisfaction level
Feature m: [relevant or not]
[…]
Requirement 4
Soft goal 5: [satisficed or not] satisfaction level
Feature n: [relevant or not]
[…]

Figure 3: report summary prototype for each configuration of the case study

In this study, RMAS was run in an offline mode. However, an online monitoring mode is possible using a distance communication between the monitored system and RMAS. The latter will continuously observe the collected data received from the system, assess the requirements satisfaction whenever a given environmental condition occurs (defined by a soft-dependency), diagnose the source of violations, if any, and propose remediation actions to the system. These actions could be used on the fly to evolve the adaptive system via actuators. Furthermore, developers or analysts could exploit the diagnosis result to improve the adaptations strategies in a maintenance phase.

Phase 3

A more in-depth analysis is performed to explore RMAS output using IBM SPSS Statistics and Ms. Excel tools. This includes aggregating the data and doing some computations to answer the research questions.

RESULTS

The main results are summarized (question by question) as follows:

1. What is the overall accuracy of INSPIREus?

The precision and recall measures are generally applied in order to evaluate the effectiveness of system configurations, in terms of accuracy and completeness respectively.

The Precision in this study is the ratio of satisfactory configurations (evaluated as valid configurations by RMAS tool) to the total collected configurations:

$$Precision = \frac{retrieved\ satisfactory\ configurations}{retrieved\ configurations}$$

The recall is the number of satisfactory configurations divided by the total satisfactory configurations (including those that are not proposed to learners).

$$\text{Recall} = \frac{\text{retrieved satisfactory configurations}}{\text{total satisfactory configurations}}$$

As details of all satisfactory configurations were not available in the gathered data, the recall wasn't calculated for this study. Only the precision was calculated for each learner. Table 3 shows precision measurements for each learner. In many cases the precision was good (up to 68%) except for three learners where the precision was below 35%.

Table 3: INSPIREus precision for each learner.

StudentID	Total Configurations	Total Relevant Configurations	Precision (%)
0	181	109	60,22
1	643	603	**91,29**
2	956	589	61,61
3	266	250	**93,98**
4	350	216	61,71
5	191	63	34,55
6	435	132	33,1
7	165	98	59,39
8	286	89	30,77
9	161	110	68,32
10	351	164	88,03
11	286	233	81,47
12	105	58	55,24
13	451	443	**98,23**
14	393	353	76,34
15	221	135	61,09
16	90	79	87,78
17	193	137	70,47
18	208	208	**100**
19	233	110	47,21

By grouping the calculated system precision based on the learning style (see Table 4), the results show some variations regarding the system precision within the same learning group. For instance, a considerable variation in the system precision can be seen between activist learners (sample variance=301,83)

Table 4: descriptive statistics of INSPIREus precision within learning groups.

Groups	Sample size	Min	Max	Average	Variance
Activist learners	8	47,21	93,98	71,34	301,83
Pragmatist learners	2	55,24	70,47	62,85	115,97
Reflector learners	4	30,77	60,22	39,66	190,29
Theorist learners	6	61,61	100	84,28	208,13
Total	20	30,77	100	68,04	462,40

The ANOVA showed a significant difference in the mean precision between at least two of the learning groups (μreflector #μtheorist # μpragmatist #μactivist): F (3.238887)=6. 8679, p-value=003 < 0.05.

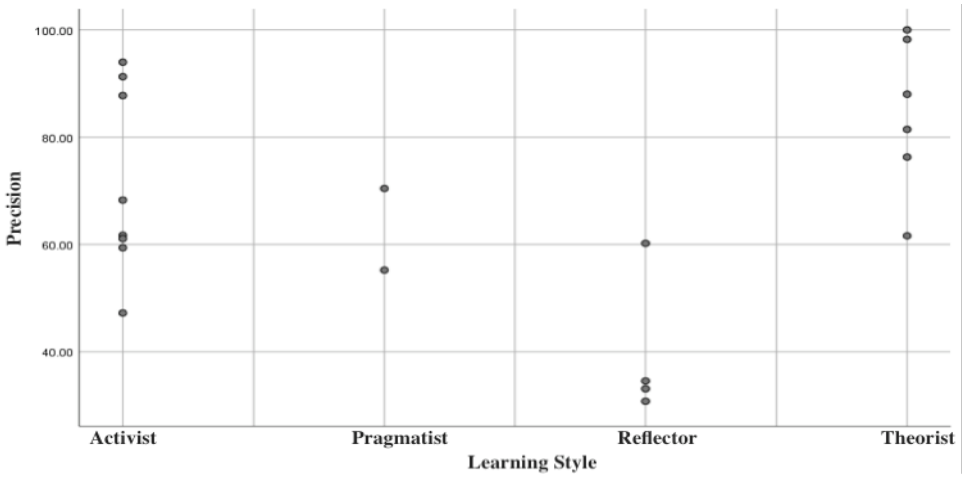

Figure 4: INSPIREus precision for each learner grouped by learning style

Specifically, as illustrated in the scatter points (Figure 4), the system was more effective for theorist learners and less effective for reflector ones.

2. How well does INSPIREus fulfill the learning requirements?

To have an answer to the above question, we have calculated the requirement fulfillment level ($Req_{fulfillment\_level}$) for each configuration by analyzing its related soft-goals satisficement levels as well as applying a prioritization strategy (see Table 5), which consists of giving weight (w_{SG}) to each soft goal according to its importance for the requirement at hand. Mathematically, this can be stated as:

$$Req_{fulfillment\_Level} = \sum_{i=1}^{k} w_{SGi} * \frac{\text{obtained satisfactionLevel}_{SGi}}{\text{required satisfactionLevel}_{SGi}}$$

Where k is the number of SGi related to the requirement under consideration, and $\sum w_{SGi} = 1$.

The $Req_{fulfillment_{Level}}$ ranges from 0 to 4, where 0 corresponds to fully denial and 4 corresponds to fully fulfilled.

To help understand the results, the min-max normalization is used to scale the $Req_{fulfillment_{Level}}$ values between 0 and 1, and the following four fulfillment classes were defined:

- Class A represents good level ∈ [0.7, 1]

- Class B represents suspected level ∈] 0.5, 0.7[
- Class C represents warning level ∈ [0.4, 0.5]
- Class D represents insufficient level ∈ [0, 0.4[

Table 5: Soft goal weights according to the prioritization strategy

Requirements	Req1		Req2	Req3	Req4
Soft goals to assess quality requirements	SG1	SG2	SG3	SG5	SG4
Soft goal definition	"Increase remember level content accuracy"	"Increase use level content accuracy".	"Improve curriculum sequencing"	"Enhance communication between learners".	"Optimize the sharing of control between learners and the system".
Weight	0.6	0.4	1	1	1

As shown in Figure 5, Req1 was fulfilled with fairly good levels (see color code) for all learners' configurations, except for learners '4' and '7' that were activists. Req2 was partially fulfilled, however as learners advanced in their learning, the system ended up achieving satisfactory levels for all learners, except for '5', '7' and '11'. Similarly, in Figure 6, Req3 and Req4 were partially fulfilled with a unique value (0.5), which indicates that the system did not adapt its strategies to support the 'communication' and the 'sharing control between learners and the system' respectively. Hence, there is a need for further improvement to support them.

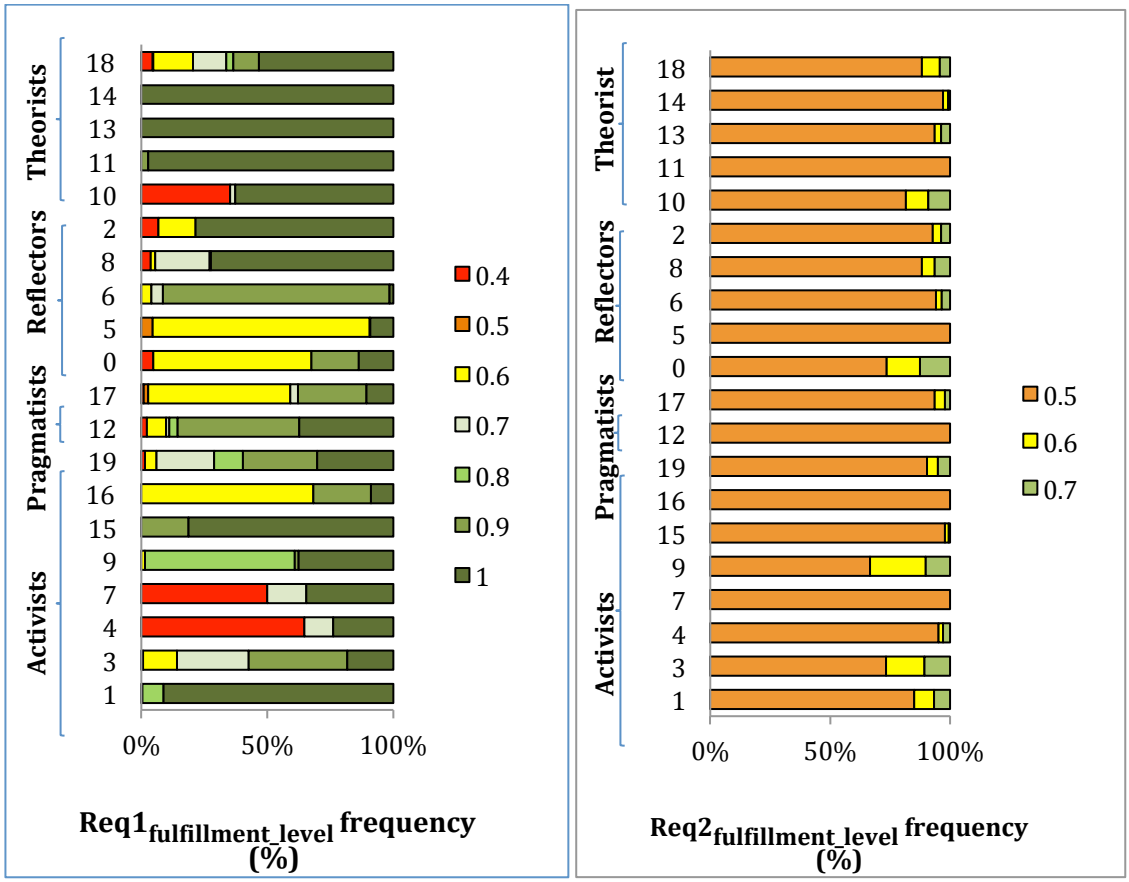

Figure 5: Frequency of Req1, Req2 fulfillment levels for each learner identified by learner ID and grouped by learning style. Color code: 'shade of green': Class A; 'yellow': Class B; 'orange': Class C; and 'red': Class D.

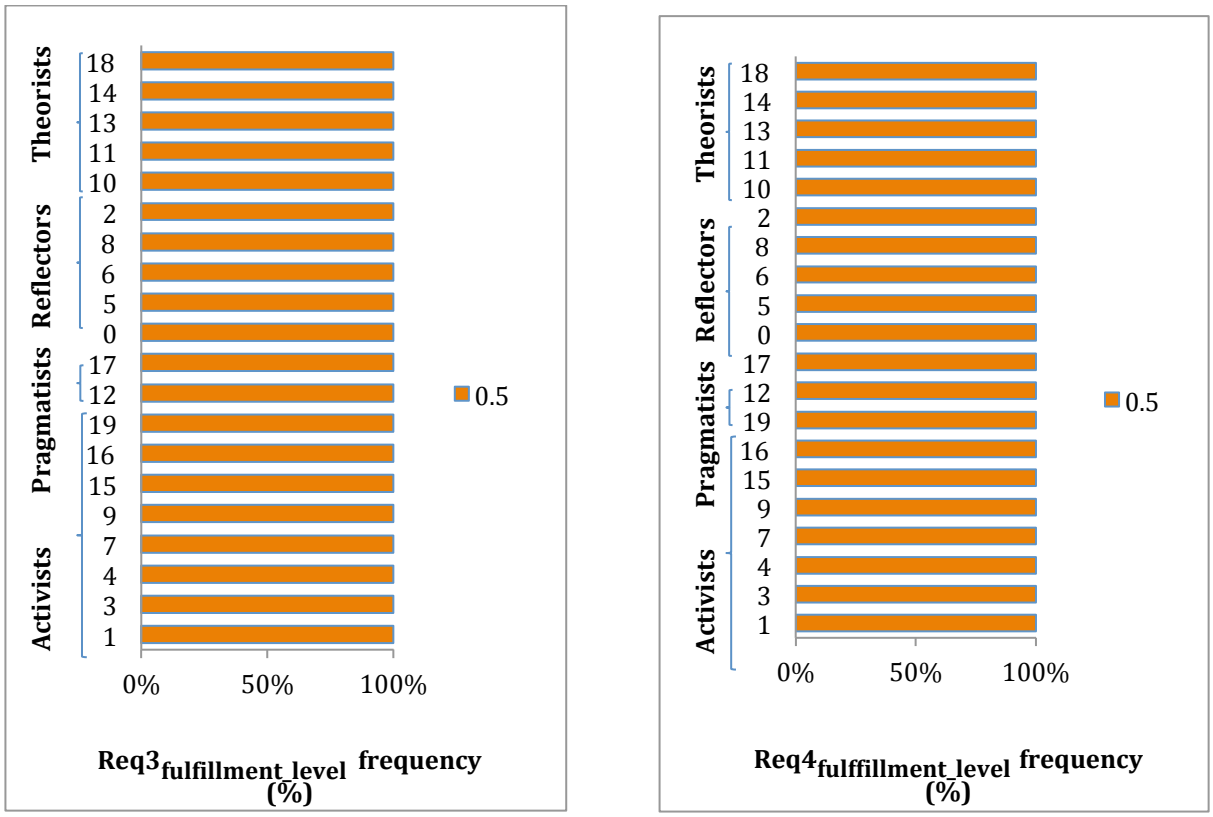

Figure 6: Frequency of Req3, and Req4 fulfillment levels for each learner identified by learner ID and grouped by learning style. Color code: 'shade of green': Class A; 'yellow': Class B; 'orange': Class C; and 'red': Class D.

Also, we have investigated the system behavior toward learning groups (based on learning style). As depicted in Figure 7, INSPIREus satisfied Req1 with very good levels for both pragmatist and theorist learners (Req1 was fulfilled at least with 0.7 in all the configurations), as well as activist learners. However, it partially satisfied Req1 for reflector learners. Moreover, it satisfied Req2 in a similar way for almost all learning groups, but less sufficiently for pragmatist learners. Req3 and Req4 were partially fulfilled with exactly '0.5' for all learners regardless of each learning group.

(Req1)

	A	B	C	D
Activists	80.75	11.33	0.08	7.85
Pragmatists	100	0	0	0
Reflectors	44.12	55.88	0	0
Theorists	100	0	0	0
Total	79.48	5.75	0.39	4.38

(Req3)

	A	B	C	D
Activists	0	0	100	0
Pragmatists	0	0	100	0
Reflectors	0	0	100	0
Theorists	0	0	100	0
Total	0	0	100	0

(Req2)

	A	B	C	D
Activists	5.42	7.36	87.22	0
Pragmatists	1.34	2.68	95.97	0
Reflectors	4.82	4.3	90.89	0
Theorists	3.63	3.75	92.62	0
Total	4.35	4.97	90.67	0

(Req4)

	A	B	C	D
Activists	0	0	100	0
Pragmatists	0	0	100	0
Reflectors	0	0	100	0
Theorists	0	0	100	0
Total	0	0	100	0

Figure 7: Frequency (%) of Req1, Req2, Req3 and Req4 compliance classes grouped by learning style

Furthermore, On the basis of the calculated requirements fulfillment levels, we have measured the quality of each configuration quality (Q_{config}) so as to have an overall idea about the system quality and how well it fulfills its requirements during operation. In so doing, we have applied the same prioritization reasoning for the requirements according to their importance for the system. Thus, we have given a weight (w_{Req}) to each requirement (see Table 6 bellow) based on the following assumptions:

- The requirements Req1 and Req2 are hard ones that *must be fulfilled* (i.e. must be fully fulfilled and without which the system cannot be adaptive).
- Req3 *should be fulfilled* (i.e. it is important for the system but can be partially fulfilled).
- Req4 *could be fulfilled* (i.e. it is not required but preferable to enhance system functionalities).

Mathematically, this can be stated as:

$$Q_{config} = \sum_{i=1}^{n} w_{Reqi} * Reqi_{fulfillement\_level} \qquad (*)$$

Where $\sum_{i=1}^{n} w_{Reqi} = 1$ and $Reqi_{fulfillement\_level} \in [0, 1]$.

The Q_{config} ranges from 0 to 1, where values close to 0 correspond to low quality.

Table 6: Requirements weight according to the prioritization strategy

	Req1	Req2	Req3	Req4
Weight	0.4	0.3	0.2	0.1

Finally, the requirements fulfillment measurements have helped us evaluate each of the configurations proposed to the learners. The quality configuration served as a first diagnostic marker that aids the discovery of satisfaction problems.

Figure 8 shows the frequency of each quality value retrieved by the equation (*) for each learner.

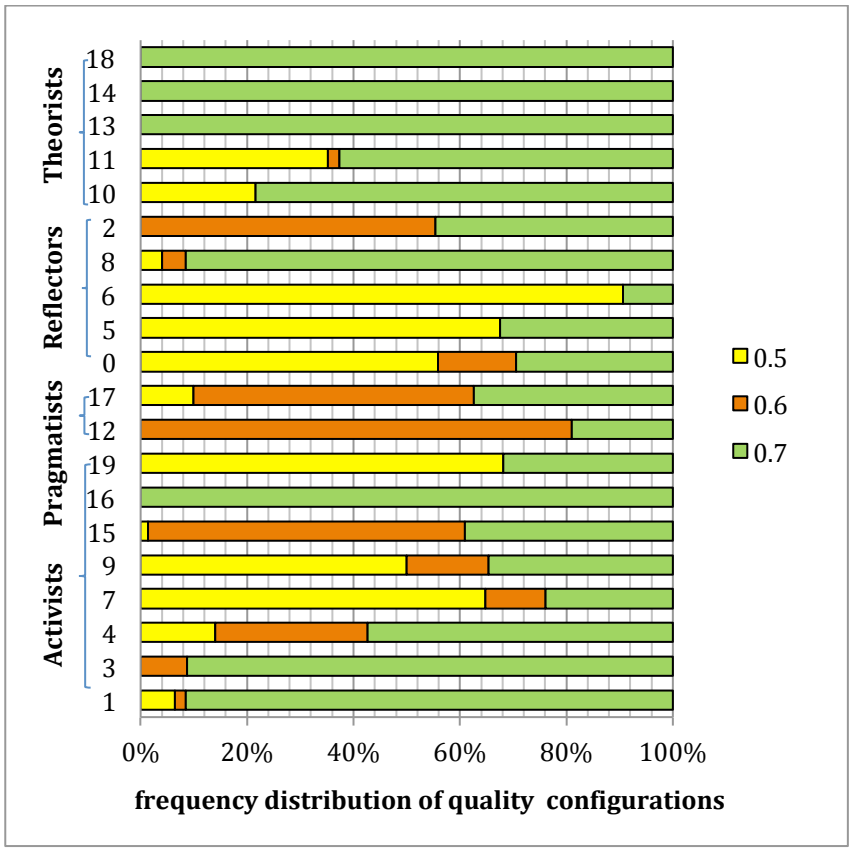

Figure 8: Frequency distribution of quality configurations for each learner. Color code: 'shade of green': Class A; 'yellow': Class B; 'orange': Class C; and 'red': Class D.

For instance, all the configurations proposed to the learners 16, 17, 18 and 19 were of good quality. While for the learner 6, the majority of the configurations (90,63%) were of low quality.

Table 7 depicts the frequency of each quality value grouped by the learning style. In general the system proposed satisfactory configurations (62.28% were fulfilled with at least 0.7 out of 1), but the results were less satisfactory for almost all reflector learners and some activist ones (especially learners '7' and '9' as shown in Figure 7 below).

Table 7: Frequency distribution of quality configurations values grouped by learning style.

	Configuration quality		
	0.5	**0.6**	**0.7**
Activists	19.26	14.16	66.59
Pragmatists	6.04	63.76	30.2
Reflectors	58.98	3.13	37.89
Theorists	5.39	21.68	72.93
Total	20.52	17.2	62.28

3. Why are some requirements not fully (or at least partially) satisfied by INSPIREus?

A requirement is more or less not satisfied in a given configuration if some features in the configuration contribute to poor satisficement of the soft goals related to this requirement. Therefore, in order to answer the 'why' question, we have identified these irrelevant features.

For the sake of clarity and brevity, we depict only the hard requirement Req1. Figure 9 shows the features that were proposed to each learner, while Figure 10 depicts the identified irrelevant features among those offered to each learner.

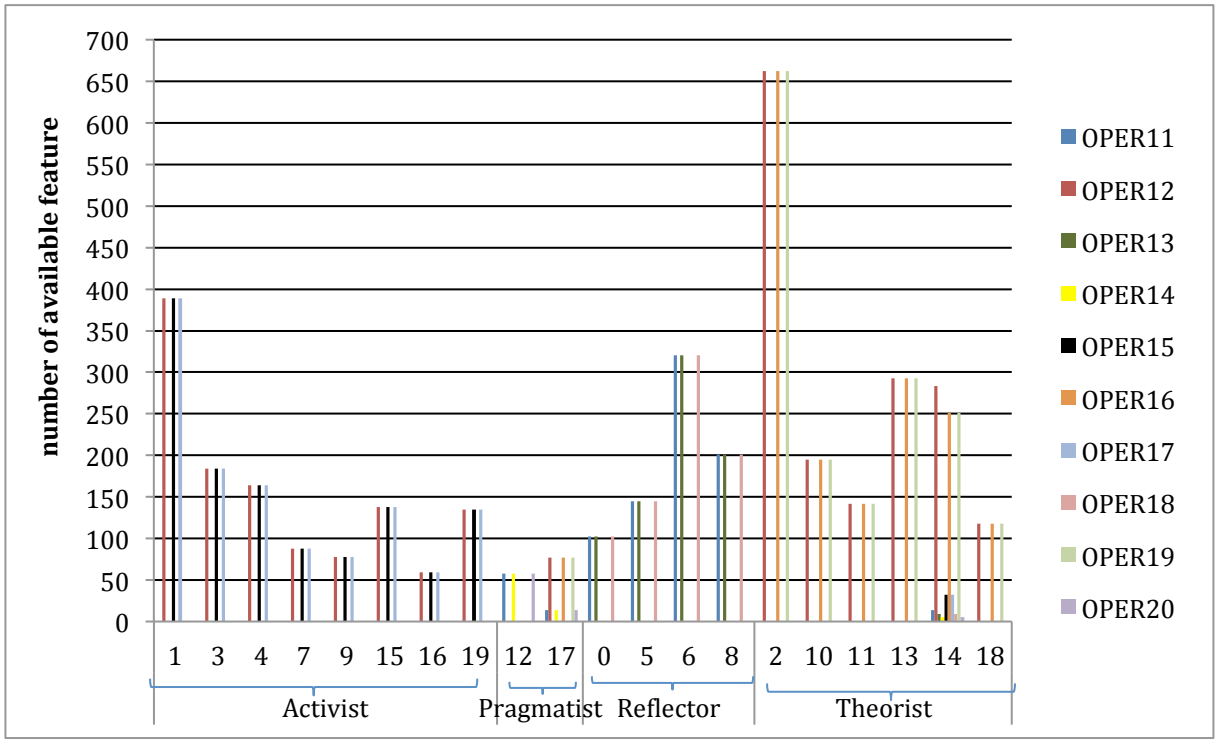

Figure 9: excerpt of the features proposed to each learner (frequency) while learning

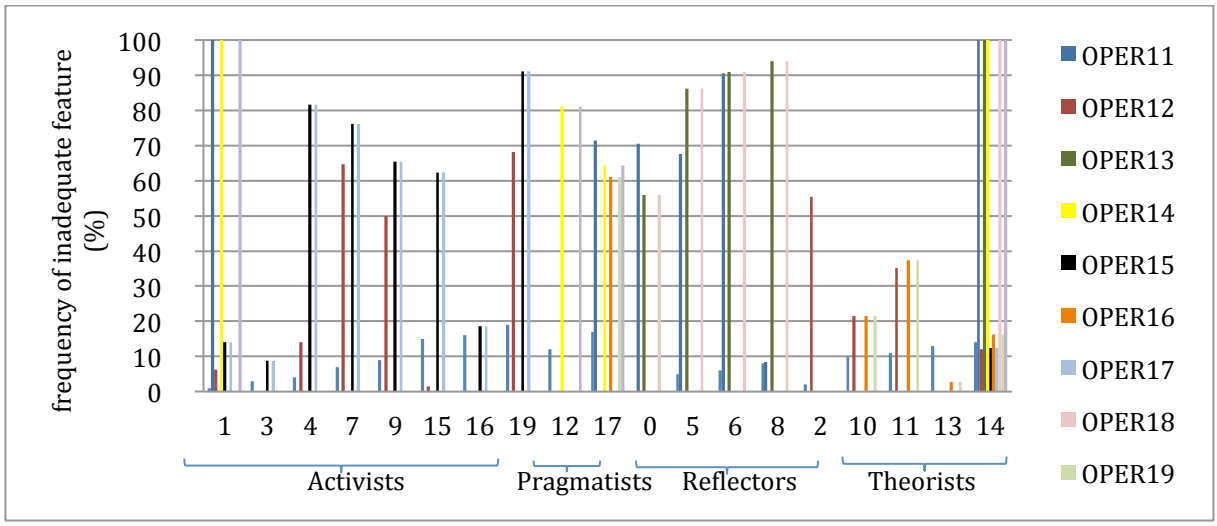

Figure 10: excerpt of frequency distribution of the features evaluated as irrelevant for each learner.

For instance, three features OPER13, OPER11 and OPER18 were evaluated as irrelevant (90% of the time) for learner 6, which refer to 'example-based presentation in the use level', 'expository presentation in remember level' and 'Turn to reflector' strategies respectively. We came back to the log files to verify and interpret the result. Indeed, the system considered learner 6 as reflector one, that is why it applied these strategies related to the learning style "reflector". While, in the history of her/his learning, we have noticed that this learner is theorist rather than reflector; s/he prefers to study using theory rather than examples. And as the system does not update the learning style, it maintains these strategies for learners even when they are not effective.

Similarly, for learner 2, 'OPER12' is the only feature evaluated as irrelevant (56% of the time) which refers to 'theory-based presentation in the use level' strategy. Based on the history of her/his learning, the system considered this learner as a theorist and applied 'theory-based' adaptation strategy for her/him. However, we have noticed that this learner preferred to study by examples rather than theory in the first half of the learning. This is why the theory-based was evaluated as irrelevant for her/him.

Following this reasoning, analysts/developers can improve the system's adaptation strategies by taking into account some unforeseen parameters at design time (here, the need to update the learning style and not rely on the learning style affected at the first time the learners log out the system).

4. What are the typical features used during a successful learning process?

Learning is qualified as successful when the learning requirements are better supported by the AES (precision >= 75%) and the learner performance is high (final score> = (¾ * 12) = 9).

We propose to identify learners that verify these criteria and group them by their learning styles. Then, analyze the resource materials for each group and extract the common resources evaluated as relevant more than 75% of the time during the learning process. The main goal is to optimize the adaptation strategies in a way that the system will recommend these extracted recourses for the learners group.

We were unable to answer this question because all learners in our sample passed the final exam with good grades (17 learners out of 20 had 12 out of 12 scores),

DISCUSSION

This paper has examined the learning associated with one type of adaptive e-learning systems, which is based on learning style dimension to personalize the learning process. The results reveal the following:

- High quality configurations were proposed to the learners (Qconfig was at least equal to 0.7). However, we found discrepancies in the effectiveness of AES strategies between learning style groups. That is, the adaptive system was more accurate for theorist learners and less accurate for reflector ones. Considerable differences within learning style groups were also detected.

- The monitored requirements have not been entirely fulfilled as expected. That is, the adaptive systems fulfilled Req1 and Req2 with fairly good levels for almost all learners, while it did not well support Req3 and Req4. On the one hand, the first two requirements focus is to offer more tailored learning for every learner, to enable them to evolve at their own pace and according to their needs. On the other hand, the latter two requirements aim to support learners when they are fully responsible for their learning, which induces a more effective pedagogy of mediation to be set up, with a view to help learners when they are confused or in case of bad decisions, and to improve communication between learners.
- The diagnosis of the requirements based on the studied configurations identifies discrepancies between the proposed configurations features and learner's interest. For example, we found that some learners prefer to study with a media format different than the one proposed by the system. A possible explanation is that learners are not merely reflector, theorist, activist or pragmatist but they are actually a combination of all. The adaptive system should be flexible enough to respond to the need of each learner and not treating them as a 'class' of learning style.

In the light of these results, we argue that to adopt better adaptation strategies an AES should continuously monitor its requirements (statically specified at design time as well as emergent ones) while evolving and link them to its strategies. The top-down diagnosis of requirements through a trade-off between the requirements and the adaptation strategies will lead to a good understanding of its adaptive behavior during learning activities and improve its design by taking into account some unforeseen parameters at design time.

RELATED WORK

This study is related to runtime monitoring, which seeks to automatically prove that system behavior complies with relevant constraints such as requirements, guidelines, and laws. There are several approaches in the literature of runtime monitoring, which differ regarding their capabilities and their related domains. These include mainly student profiling in distance education, debugging and requirements-based monitoring in software engineering.

Student profiling seeks to develop and maintain a record of what learners are doing and which strategies they are using to achieve their respective goals, after a learning episode. It has been a subject of significant research in self-regulated learning (SRL), which seeks "to understand how a particular learner learns and achieves the learning, despite apparent limitations in mental ability, social environment background or in quality schooling" (Zimmerman, 1989 p.4). The monitoring functionalities have been designed for both teachers and learners so they can understand and have information about learners' metacognitive skills, as well as for visualization concern through data mining (Graphvis, 2004; Mazza & Dimitrova, 2007; Romero-Zaldivar et al., 2012). However, in the interest of maintaining stability for AESs, analysts and designers need also to develop a good understanding of how these systems behave while operating and how the requirements are effectively fulfilled.

Monitoring provides a promising basis for debugging approaches, which seek to understand the internal activities of a software system. However, these approaches are related to low-level implementation such code and data structure, and the lack of depth in the information they provide such as how the communication of different components is achieved. Alternatively, requirements-based monitoring extends the idea of performance and events monitoring to present an abstracted view of the system's execution (Fickas & Feather, 1995; Robinson, 2003; Welsh & Sawyer, 2009; van Hoornet al., 2009; Bencomo, et al., 2012; Wang et al., 2009). It seeks to increase requirements awareness at runtime and improve self-adaptive systems (SASs). However, as pointed out by Vierhauser et al. (2016), this research field is quite fragmented. Thus important research has tended to focus on performance monitoring (Van Hoorn et al., 2012) while others are limited to particular domains such web service. As far as we know, there is no research that has applied requirements monitoring to adaptive e-learning systems. Accordingly, with the present study, we claim that requirements analysis at runtime is a key potential for evaluating AESs and improving their design.

THREATS TO VALIDITY

As for any experimental evaluation, there exist some threats that could affect the validity of our results. In fact, there are three minor limitations worth noting regarding our methods:

INTERNAL VALIDITY

The first limitation is related to the use of log files. Although they contain rich information about learners and system behavior, they often contain noisy information. Accordingly, we have performed an extensive pre-processing of the data sets. In this pre-processing step, we have eliminated missing values, translated the data sets from Greek to English, which was verified later by the administrator of INSPIREus, and unified information between the data sets to consolidate data. Also, to prevent extreme outliers, learners that did not fully engage in the learning were eliminated from this study (one learner was eliminated out of 21).

The second limitation is related to the requirements specification. In this study, the requirements were identified based on general issues in AESs and not extracted directly from the case study specification. This separation helps to evaluate the overall AESs against general requirements, indispensable to ensure an effective learning. Inevitably, some assumptions, which could be subjective, must be made to associate systems features to the monitored soft goals. Consequently, to ensure an accurate model, we validated the model by research teachers before its use.

EXTERNAL VALIDITY

Finally, the third limitation is related to the size of the data source. Even though the data was composed of 6165 configurations, the later is collected from small groups of learners (20) that had a good final score. These high grades may be due to the course content that was short and there was more time to pick up concepts. These could be a reason why no significant relationships were found between learners performance and the calculated system precisions, and which prevents us from being able to generalize the results to a larger population without further research.

CONCLUSION

Learning environments are constantly changing and so their requirements. AES verification and evaluation processes as part of software construction are no longer sufficient to guarantee that the adaptive system is built correctly and it meets its requirements.

The present research highlights the potential of monitoring the requirements of AESs at runtime to improve their design and enable their evolution. On the basis of an experimental study, we have identified some insights and issues related to requirements compliance at runtime that helped us to detect unforeseen instructional design issues. Thus, the monitored requirements were not totally fulfilled as expected, and discrepancies in the effectiveness of adaptation strategies were identified between learning style groups and also within each learning style group. Analysts and developers can exploit the issues in question to improve the adaptation strategies on the fly or further in a maintenance phase. This will help ensure dynamic adaptation strategies that evolve to meet learners' needs, with fewer failures and higher requirements satisfaction.

There are several implications of the findings for the research and the practice:
-For researchers, the study suggests that (1) further studies are needed to investigate AESs design issues that emerge at runtime, (2) further research on developing specific indicators related to requirements compliance are needed in the field of adaptive e-learning systems.
-For practitioners, the study suggests that diagnosing requirements compliance at runtime can be an essential means of receiving feedbacks at the requirements level, which can optimize their performance (because it opens up the possibility to act whenever a perturbation among the desired state of interaction is detected), and increase confidence about their capabilities at runtime by keeping up a dashboard that visualizes the received feedbacks.

ACKNOWLEDGEMENTS

We would like to thank Kyparissia Papanikolaou, Professor in Department of Education at School of Pedagogical & Technological Education (ASPETE), Athens, Greece, for providing us the data sets used in this research, and Ms. Dionisia Chinou for the valuable technical information about INSPIREus.

References

Baker, R. S., Goldstein, A. B., & Heffernan, N. T. (2010). Detecting the moment of learning. In Proceedings of the ACM international conference on interactive tabletops and surfaces (pp. 25–34). Saarbrücken, Germany.

Bencomo, N., Welsh, K., Sawyer, P., & Whittle, J. (2012, July). Self-explanation in adaptive systems. In Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on (pp. 157-166). IEEE.

Brusilovsky, P., Schwarz, E., & Weber, G. (1996, June). ELM-ART: An intelligent tutoring system on World Wide Web. In International conference on intelligent tutoring systems (pp. 261-269). Springer, Berlin, Heidelberg.

Brusilovsky, P., Eklund, J., & Schwarz, E. (1997). Adaptive navigation support in educational hypermedia on the World Wide Web. In Human-Computer Interaction INTERACT'97 (pp. 278-285). Springer, Boston, MA.

Dardenne, A., Van Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. Science of computer programming, 20(1-2), 3-50.

Dounas, L., Mazo, R., Salinesi, C., & El Beqqali, O. (2015, November). Continuous monitoring of adaptive e-learning systems requirements. In Computer Systems and Applications (AICCSA), 2015 IEEE/ACS 12th International Conference of (pp. 1-8). IEEE.

De Bra, P., Aerts, A., Smits, D., & Stash, N. (2002, october). AHA! Version 2.0, More Adaptation Flexibility for Authors. Proceedings of the AACE ELearn'2002 conference, pp. 240-246.

Devoper, C. & Quintin, J. J. (1992). Learner Control vs. Computer Control in a professional Training Context. In M. Giardina (Eds.), Interactive Multimedia learning Environments (Vol. 93, pp. 234-247). Berlin: Springer Verlag.

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.

Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., & Traverso, P. (2004). Specifying and analyzing early requirements in Tropos. Requirements Engineering, 9(2), 132-150.

Honey, P., & Mumford, A. (1992). The manual of learning styles.

Graf, S., Lin, F., & McGreal, R. (2012). Intelligent and Adaptive Learning Systems: Technology Enhanced Support.

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

Jurado, F., Santos, O. C., Redondo, M. A., Boticario, J. G., & Ortega, M. (2008). Providing dynamic instructional adaptation in programming learning. In Proceedings of the 3rd international workshop on hybrid artificial intelligence systems (pp. 329–336). Burgos, Spain.

Karampiperis, P., & Sampson, D. (2005). Adaptive learning resources sequencing in educational hypermedia systems. Journal of Educational Technology & Society, 8(4).

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.

Mitrovic, A. (1998). A knowledge-based teaching system for SQL. In Proceedings of ED-MEDIA (Vol. 98, pp. 1027-1032).

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.

Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and using nonfunctional requirements: A process-oriented approach. IEEE Transactions on software engineering, 18(6), 483-497.

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.
Papanikolaou, K. A. (2015). Constructing interpretative views of learners' interaction behavior in an open learner model. IEEE Transactions on Learning Technologies, 8(2), 201-214.
Robinson, W. N. (2003, September). Monitoring web service requirements. In Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International (pp. 65-74). IEEE.
Robinson, W. (2010). A roadmap for comprehensive requirement modeling. Computer, 43(5), 64-72.
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.
Sampson, D. G., & Karampiperis, P. (2012). Decision models in the design of adaptive educational hypermedia systems. In Intelligent and Adaptive Learning Systems: Technology Enhanced Support for Learners and Teachers (pp. 1-18). Igi Global.
Specht, M. (1998). Empirical Evaluation of Adaptive Annotation in Hypermedia.
Tsiriga, V., & Virvou, M. (2002). Initializing the student model using stereotypes and machine learning. In Proceedings of the 2002 IEEE international conference on system, man and cybernetics (pp. 404–409).
Van Hoorn, A., Rohr, M., Hasselbring, W., Waller, J., Ehlers, J., Frey, S., & Kieselhorst, D. (2009). Continuous monitoring of software services: Design and application of the Kieker framework.
Van Hoorn, A., Waller, J., & Hasselbring, W. (2012, April). Kieker: A framework for application performance monitoring and dynamic software analysis. In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (pp. 247-248). ACM.
Van Lamsweerde, A., Darimont, R., & Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. IEEE transactions on Software engineering, 24(11), 908-926.
Vierhauser, M., Rabiser, R., & Grünbacher, P. (2016). Requirements monitoring frameworks: a systematic review. Information and Software Technology, 80, 89-109.
Wang, X., Yang, Y., & Wen, X. (2009). Study on blended learning approach for English teaching. In Proceedings of the 2009 IEEE international conference on systems, man, and cybernetics (pp. 4641–4644).
Wang, Y., Mcilraith, S. A., Yu, Y., & Mylopoulos, J. (2009). Monitoring and diagnosing software requirements. Automated Software Engineering, 16(1), 3.
Welsh, K., & Sawyer, P. (2009, June). Requirements tracing to support change in dynamically adaptive systems. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 59-73). Springer, Berlin, Heidelberg.
Wolf, C. (2003, January). iWeaver: towards' learning style-based e-learning in computer science education. In Proceedings of the fifth Australasian conference on Computing education-Volume 20 (pp. 273-279). Australian Computer Society, Inc.
Yu, E. S. (1997, January). Towards modelling and reasoning support for early-phase requirements engineering. In Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on (pp. 226-235). IEEE.
Zimmerman, B. J. (1989). Models of self-regulated learning and academic achievement. In Self-regulated learning and academic achievement (pp. 1-25). Springer, New York, NY.

BIOGRAPHY

Lamiae Dounas received the bachelor degree in computer science and master degree in information system, networking and multimedia from the faculty of science, Sidi Mohammed Ben Abedellah University, Fez, Morocco in 2010 and 2012 respectively. Currently, she is a Ph.D candidate at the faculty of science, Sidi Mohammed Ben Abedellah University, Morocco and paris 1 Panthéon-Sorbonne University. Her main research interests include (self) adaptive systems, e-learning, requirements engineering and machine learning.

Camille Salinesi is a Professor at Paris 1 Pantheon Sorbonne university . He is the head of the Centre de Recherche en Informatique, which specializes in Information Systems Engineering. Salinesi received a PhD from Universite´

Paris 6, Pierre et Marie Curie. His main research interests include requirements engineering, strategic alignment and product lines.

Omar El Beqqali is a Professor at Sidi Mohammed Ben Abdellah University. He is holding a Master in Computer Sciences and a PhD respectively from INSA- Lyon and Claude Bernard University in France. He is leading the GRMS2I research group, and was invited professor at UCB-Lyon1 university, INSA-Lyon and UIC University. His main research interests include distributed databases, pervasive systems and mobile environments.

Appendix A: Variability goals model of INSPIREus using REFAS language