Towards Self-Explainable Adaptive Systems (SEAS): A Requirements Driven Approach

Nabia Khalid[a], Nauman A. Qureshi[b]

[a]Dept. of Computing, SEECS, National University of Sciences and Technology ,44000, Islamabad, Pakistan
[b]Dept. of Comp. Sci, CCSIT, King Faisal University, 31982, Al Ahsa, Saudi Arabia

Abstract

Self-Explainable Adaptive Systems (SEAS) focus on liaising their decisions to the end-users while providing solutions to reduce the complexity and costly maintenance by operating in an uncertain and dynamically changing environment. At the time of crises, when a system behaves differently from its usual behaviour leaving the end-user clueless about what went wrong; a SEAS imparts user explanations in natural language for better understandability of the system. This mitigates the risks generated from the system's negative behaviour. [**Research Problems**] Engineering SEAS is challenging. In this planned research, we aim to answer the following questions i.e. How to model the requirements and domain knowledge so that the system adapts to runtime changes according to the end-user's environment? How to generate user explanations in natural language form exploiting the model at runtime? [**Results**] To realize SEAS, we aim to explore the requirements driven approach such that the an adaptive service-based system (e.g. SBA) can deduce self-explanations based on the data generated from end-user's interactions with itself [**Contribution**] In this paper, we present our planned approach by investigating the relationship between Explainable Artificial intelligence (XAI) and runtime requirements engineering (RE) to realize SEAS which is capable to adapt and produce scenario-based positive and negative user explanations at runtime. We elaborate our proposal with a motivating example.

Keywords

Requirements Engineering, Self-Adaptive Systems, User Explanations, Ontology

1. Introduction

Goal Models have been used extensively in engineering adaptive systems as an effective requirements modeling technique [1, 2]. Goal models provide effective way to model functional (Hard goals) and non-functional requirements (Soft goals) at an early stage of RE process to support decision making over alternative design choices through goal reasoning.

Artificial Intelligence (AI) systems on the other hand are more centered on Machine Learning (ML) processes without catering too much of end-user needs. Recent surge in AI-based systems has promoted research in several dimensions. One of them is to equip the system to provide user explanations enabling the end-users to know about system decisions i.e. how responsible and

In: F.B. Aydemir, C. Gralha, S. Abualhaija, T. Breaux, M. Daneva, N. Ernst, A. Ferrari, X. Franch, S. Ghanavati, E. Groen, R. Guizzardi, J. Guo, A. Herrmann, J. Horkoff, P. Mennig, E. Paja, A. Perini, N. Seyff, A. Susi, A. Vogelsang (eds.): Joint Proceedings of REFSQ-2021 Workshops, OpenRE, Posters and Tools Track, and Doctoral Symposium, Essen, Germany, 12-04-2021

11msitnkhalid@seecs.edu.pk (N. Khalid); nqureshi@kfu.edu.sa (N. A. Qureshi)

transparent is their decision making process?. However, research tackling this idea is limited so far.

In our planned research, we aim to bridge the gap between RE and AI-based systems. Precisely, we envision to engineer Self-Explainable Adaptive Systems(SEAS) by investigating the idea that Self-Adaptive Systems (SAS), which continuously run to overcome the challenges of complexity in uncertain environments, can be well facilitated with run-time representation of requirements (i.e. goal-based ontology as live artifact) to support reasoning over requirements change and generating user explanations thereby enabling the end-user to take informed decisions or rely on SEAS decisions at run-time. The common ground between Explainable AI (XAI) [3] and Continuous Runtime RE [1] (CARE) is the end-user, where the former concept is user-centric as the system communicates its decisions in human readable format to the end-user.

Recent implementations of XAI systems uses black-box Deep Neural Network (DNN) models allowing users to infer why a decision is made. However, it has been argued that using deep neural network (DNN) based approach has its own limitation in case of runtime changes or model extrapolation [4].

Runtime RE involves Goal driven adaptation of the system. Modeling of system's behaviour, operating environment, resource availability and the end-user's context in the form of a goal model provides flexibility to reason about design alternatives [5]. Design models of Service-based Applications (SBAs) are defined as a set of loosely coupled variation points or services facilitating either web client or smart phone applications. To enable adaptation of SBAs at runtime, Continuous Runtime RE has been proposed as a candidate solution [1] (CARE). To enable continuous Runtime RE, requirements problem requires reformulating and resolving the end-user attitude (preferences, selection between requirements) or the operating environment (Domain assumption, Context, Quality Constraint) when changes occur [6](ReCORE), [7] (ARML).

In this work, we adopt the above approaches by introducing a requirements-driven approach to model end-user's operating environment and attitude in such a way that the system is equipped to provide user explanations upon a requirement failure. For instance, when a smart phone user is paying a bill via credit or debit card after placing the meal order and right after the payment made, the machine halts and there is no electronic receipt generated. Such a situation leaves the user in a state of confusion and it leads to some questions from end-user's perspective like: *Whether or not my order has been placed as the payment was already done?*, *Should I place a meal order again?*, *Should I wait for my meal to arrive as may be the order was placed?*. In order to cater the such confusions, the end-user should be supported with explanation in accordance with the runtime requirements model. We envision Self-Explainable Adaptive System (SEAS) to overcome such problems. In this paper, we present our planned approach with a motivating example.

The paper is organized as follows. Section 2. gives a brief background of the current approaches and deduced research objectives to the planned work, while Section 3 highlights the planned proposed approach towards the Explainable Requirements Driven Reasoning (ERDR) method and Section 4 discusses the aimed evaluation methodology of the proposed framework and conclusion.

2. Background Preliminaries

The concept of combining self adaptivity and self explainability is not new as there are recent approaches which argue that AI-based systems to be transparent in their decisions. [8], [2] emphasizes on a model based approach for introducing self-explanations in adaptive systems. The approach is based upon keeping the record of non-functional requirements (NFRs) with the functional requirements, hence producing self explanations. There are different kinds of systems upon which research has been conducted lately including Cyber-physical systems, Cloud-aware mobile systems, Agent-based systems etc. The basic infrastructure of self adaptation for the above mentioned systems is MAPE-K loops [9], where the technique allows the system to monitor the end-user and the system's environment and then analyze the faulty requirement, plan the alternate solution and then execute it by exploiting a knowledge base. In [10], a Monitor, Analyze, Build, Explain (MAB-EX) framework has been proposed to realize self-explainable systems from runtime requirements and explainability models i.e. modeled using Scenario Modeling Language (SML) to generate explanation in textual format at runtime. [11] emphasizes on the use of decision trees stored in the form of ontology for recording user explanations in order to reduce the cognitive load from the end-user and making explanations more simpler to grasp in natural language form. For this purpose, the domain knowledge of the system in question is mapped in the form of Requirements Model/ Goal Model in the Ontology where the schema is also internally reasoned over on machine learning level.

The above approaches are similar to our proposal. Differently, we argue to use ontology as a live requirements model which enables the system to generate scenario-based positive or negative explanations taking into account the user-system interaction at runtime. Although, our previous works [6](ReCORE) and [7](ARML) lays the foundation of revised CORE ontology for RE (Re-CORE) and Adaptive Requirements Modeling Language (ARML) respectively, where the former research describes how to classify and map the requirements in the form of classes and entities in ontology and the later describes a modeling language which help to reason and evolve the requirements in the ontology. In this paper, we aim to extend our previous works and provide a more practical framework for realizing SEAS.

2.1. Research Objectives

Taking into account the recent research, we are planning to incorporate the Requirements Model adapted and enhanced from the RECore Ontology [6] and [7] for designing SEAS. Following research objectives (ROs) guide our planned research.

– **RO1:-** *Performing CRUD operations over live requirements to achieve runtime self-adaptation in service-based smart phone applications:* To achieve this, we model functional requirements of the front-end SBA to enable self adaptation at runtime by actuating RE actions and rule based reasoning over the domain knowledge encoded inside the model resulting into the Requirements Model Ontology (RMO) schema to either **expand** or **contract** analogous to databases. To realize such runtime adaptation, a requirements protocol or a requirements modeling language is needed which defines relations between requirement concepts. Here we are planning to adapt and enhance Adaptive Requirements Modeling Lanuage (ARML) [7] in our proposed RMO.

– **RO2:-** *Generating user-explanations about the failed requirements by concatenating the envi-*

ronmental variables: This RO is setting off the proposed idea of chaining all the environmental variables (Domain assumption, end-user Context, Quality Constraint related to a particular requirement. The domain instantiated data variables related to the functional requirements contribute in building a scenario-based negative or positive explanations for the end-users. This is further explained with an illustrative example in section 3.

– **RO3:-** *Providing transparency in software behaviour by using Requirements Model Ontology as a knowledge base of SEAS:* Any AI-based solution involves nodes attached via weighted edges and the decision making in mapping the inputs to optimal outputs in layers lies in functional computation. Analogically, requirements modeled as ontology[1] in the form of Hard Goal nodes connected by some [7] relations. An ontology is semantically more enriched to accommodate domain concepts which are used to record end-user and system interactions and enhance back-end reasoning. Such data is needed to adapt and perform runtime RE. In comparison with other techniques, it is envisaged that it may lead to performance delays as more computations and reasoning is involved. However, the benefit of successful adaptation and generating end-user explanations are more rewarding comparing to performance glitches. Although, we are further investigating this in our research.

– **RO4:-** *Consistency of Requirements Model before and after the execution of each scenario during Runtime Adaptation:* The state of the Requirements Model should be consistent before the initial stage and after the final stage of the adaptation. Here the term consistency defines that the Requirements Model is in a valid state. The valid state of the Requirements Model at a certain point of time is achieved by the successful execution of the particular scenario in which requirements must be achieved and end-users are involved through explanations.

Based on the above mentioned ROs, we are planning to propose an Explainable Requirements Driven Reasoning (ERDR) Method for Adaptive SBAs which is described briefly in the section below.

3. Explainable Requirements Driven Reasoning (ERDR) Method

In this section, we briefly explain our proposed Explainable Requirements Driven Reasoning (ERDR) method in which the reasoning over requirements allow them to be evolved at runtime while explaining the reason of requirement failure to the end-user. The basic work flow of our proposed approach starts from the end-user's interaction with SBA for instance. The environmental data (*C-Q-K*) is generated as the result of each interaction where *C* is the Context of the end-user which is further divided in sub-entities i.e. location context,device context,user behavioural context; (*Q*) is the Quality Constraint which is further defined as conditional rules on domain specific data sets and *K* is the Domain Assumption which comprises of pre- and post- conditions specific to a requirement, which is captured by the monitoring module (**M**) of the ERDR method. Subsequently, the end-user's functional requirement is forwarded to the analysis (**A**) module based on the axioms/rules that defines the entities semantically. In case of a requirements failure, the rule based reasoner (**R**) suggests the best possible alternative action. Based on the concatenation of the recorded results (*C-Q-K*) from **R**, the user explanations (**Ex**) are generated in natural language format. Hence, this forms an end-user and SBA feedback

[1] https://protege.stanford.edu

interaction loop. All the conversation between the end-user and the SBA in the (*C-Q-K*) format is recorded at the backend and the requirements are continuously refined in the RMO. Central to our approach is the Requirements Model Ontology (RMO), which is briefly explained further along with it's structure and how the domain specific requirements are being reasoned over for evolution.

Requirements Model Ontology (RMO) conceptually comprises of two components i.e. the Abstract Goal Model (AGM) and the Domain Specific Requirements Model (DSRM). AGM is the structure of the Requirements Model which instantiates the domain model. The structural taxonomy of the AGM embodies itself into three main parts. First and foremost is the **GoalModelConcepts** adapted from [6], second one is the **Relations** adapted and enhanced from ARML [7] and the third one is the **REActions**, which enhances the previous two. These base concepts collectively define the semantics of the AGM. The concept **GoalModelConcepts** further divides in sub-concepts which defines the requirements database. Sub-concepts include Actor (*A*), Context (*C*), Quality Constraint (*Q*), Domain Assumption (*K*), Plan (*P*), Goal (*G*), Resource (*R*) and User Explanations (*UE*) where each concept is defined by a specific axiom or rule (combination of other concepts) in the RMO so that the domain specific requirements can be reasoned over them. All these concepts are joined to each other via **Relations** and therefore, evolve (CRUD operations as define in RO1 in section 2) at runtime using the **RE Actions**.

3.1. Scenario Illustration

To put into practice, the work sequence of our proposed method, let's look at a scenario shown in the Figure 1. Here the 'User Environment' box is highlighting a problem at the user's end from 'My Restaurant' SBA, which uses RMO as a live artifact.

Julia had a busy day at work. She gets out from her office at 12:45 hrs to pick up her friend and come to the restaurant where she booked a table a day before for lunch around 13:00 hrs. Due to heavy traffic and long distances she ***checks-in*** *'My Restaurant' around 13:30 hrs. She chooses to avail her booked reservation. She tries to check-in but her request gets failed leaving her in a confused state.* Here arises the cognitive load on Julia which drives the need for the following question:

Why I am unable to check-in?

In order to seek answer the question above, we show how ERDR method works at the backend of the SBA. From the above mention problem scenario, we can deduce that Julia's functional requirement is **check-in**. Figure 1 is showing that the requirement task **Check-in** has been initiated by Julia, so at first **M** fetches the Relation **associated actor** of the given task. After that the **associated actor** Julia's associated user explanation (UE) is going to be accessed with the Relation **associated UE**. As the Relations associated with all the three environment variables i.e. Domain Assumption, Context and Quality Constraint are chained as the sub properties of **associated UE**, so they can be directly called by the Actor. The environment variables are modeled with the help of axioms as explained above, so in order to successfully monitor the environment variables,**M** calls the **A** module to verify their axioms.

As shown in the figure 1, the Quality Constraint (qc1) has a condition or axiom to fulfill based on the actor's data i.e. the **difference** between the **reserved time** and the **current**

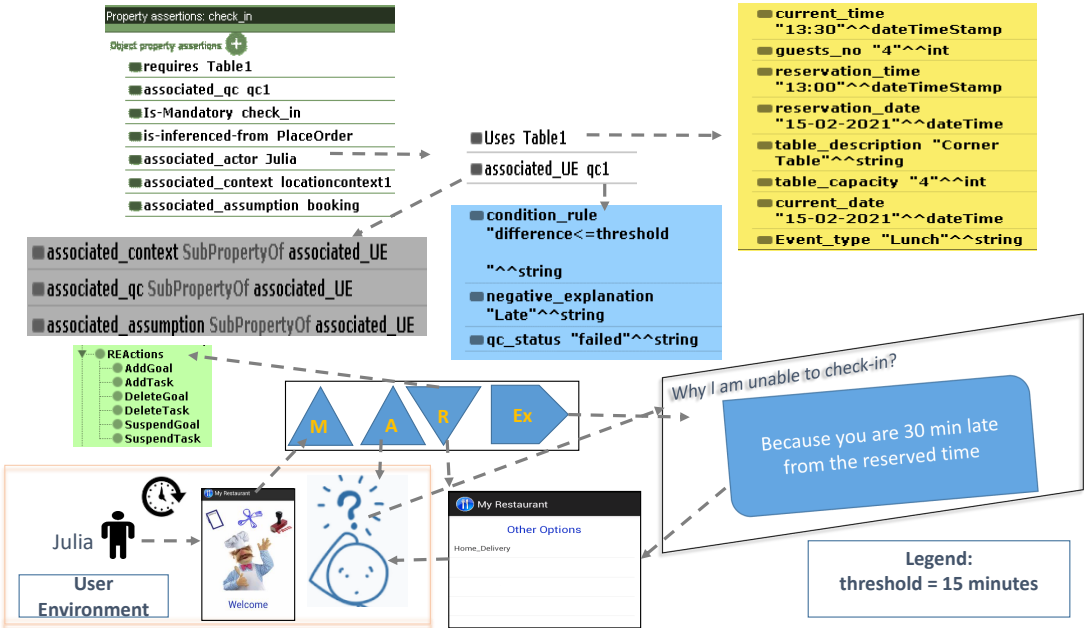

Figure 1: User Scenario for the functional requirement 'Check-in'

time of check-in should not exceed the **threshold** which is fifteen minutes. This method of axiom verification is actually a pass/fail test, if the runtime end-user monitored data exceeds the axiom then there is no un-anticipated change in the running environment of the particular requirement otherwise in case of the failure of any one of the three environment variables(qc1 status='failed'), the requirement in question fails. Based on the result, We plan to use negative explanation variable = 'Late' to generate user explanations (**Ex**) in natural language. So the answer to Julia's question is computed by combining all the environmental and data variables as shown in figure 1 and she is been redirected to an alternate optional requirement that is 'Home Delivery' by **R** based on the schema change operators i.e. RE Actions of the RMO.

4. Discussion and Conclusion

In this section, we briefly reflect on our proposed ERDR method, its evaluation methodology and conclude with future work directions.The nucleus of our ERDR method is RMO, which has been modeled and realized as an ontology. Model expansion and contraction is handled by our encoded axioms and rule based reasoners so we plan to adopt the ROMEO Methodology proposed in [12] to determine it's preciseness and optimality in our proposed approach. Summarizing the above proposed approach, our ERDR method requires further investigation for its correctness and applicability to realize SEAS. The simple illustration used to describe the proposed approach in section 3 has been chosen to prove the efficacy of the approach. However, we aim to scale and enhance the implementation of ERDR method in future so that it be used to address more realistic problems like smart vehicles or discovering evacuation paths in disaster management

systems where the RMO monitors and stores the complex environmental data sets (*C-Q-K*) of the end-user like location context, device context, pre and post conditions of the Domain Assumptions and Quality Constraints as a way to generating user-explanations when required while application adapts according to the RMO. This helps in reducing the user's cognitive load. Finally, we plan to evaluate our SEAS implementation with real users to evaluate it's efficacy.

References

[1] N. A. Qureshi, A. Perini, Continuous adaptive requirements engineering: An architecture for self-adaptive service-based applications, in: 2010 First International Workshop on Requirements@Run. Time, IEEE, 2010, pp. 17–24.

[2] K. Welsh, N. Bencomo, P. Sawyer, J. Whittle, Self-explanation in adaptive systems based on runtime goal-based models, in: Transactions on Computational Collective Intelligence XVI, Springer, 2014, pp. 122–145.

[3] C. J. Cai, J. Jongejan, J. Holbrook, The effects of example-based explanations in a machine learning interface, in: Proceedings of the 24th International Conference on Intelligent User Interfaces, 2019, pp. 258–262.

[4] D. C. Elton, Self-explaining ai as an alternative to interpretable ai, in: International Conference on Artificial General Intelligence, Springer, 2020, pp. 95–106.

[5] E. D. Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, K. Pohl, A journey to highly dynamic, self-adaptive service-based applications, Automated Software Engineering 15 (2008) 313–341.

[6] N. A. Qureshi, I. J.Jureta, A. Perini, Requirements engineering for self-adaptive systems: Core ontology and problem statement, in: International Conference on Advanced Information Systems Engineering, Springer, 2011, pp. 33–47.

[7] N. A. Qureshi, I. J.Jureta, A. Perini, Towards a requirements modeling language for self-adaptive systems, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2012, pp. 263–279.

[8] N. Bencomo, K. Welsh, P. Sawyer, J. Whittle, Self-explanation in adaptive systems, in: 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, IEEE, 2012, pp. 157–166.

[9] P. Arcaini, E. Riccobene, P. Scandurra, Modeling and analyzing mape-k feedback loops for self-adaptation, in: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE, 2015, pp. 13–23.

[10] M. Blumreiter, J. Greenyer, F. J. C. Gracia, V. Klös, M. Schwammberger, C. Sommer, A. Vogelsang, A. Wortmann, Towards self-explainable cyber-physical systems, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2019, pp. 543–548.

[11] R. Confalonieri, F. Martín, S. Agramunt, D. Malagarriga, D. Faggion, T. Weyde, T. R. Besold, An ontology-based approach to explaining artificial neural networks, ArXiv abs/1906.08362 (2019).

[12] J. Yu, J. A. Thom, A. Tam, Requirements-oriented methodology for evaluating ontologies, Information Systems 34 (2009) 766–791.