Towards a Requirements Modeling Language for Self-Adaptive Systems

Nauman A. Qureshi[1], Ivan J. Jureta[2], and Anna Perini[1]

[1] Fondazione Bruno Kessler - CIT, Software Engineering Research Group
Via Sommarive, 18, 38050 Trento, Italy
{qureshi,perini}@fbk.eu
[2] FNRS & Louvain School of Management,
University of Namur, Belgium
ivan.jureta@fundp.ac.be

Abstract. [Context and motivation] Self-adaptive systems (SAS) monitor and adapt to changing end-user requirements, operating context conditions, and resource availability. Specifying requirements for such dynamic systems is not trivial. Most of the research on self-adaptive systems (SAS) focuses on finding solutions to the requirements that SAS is built for. However, elicitation and representation of requirements for SAS has received less attention at early stages of requirements engineering (RE). **[Question/problem]** How to represent requirements for SAS in a way which can be read by non-engineering stakeholders? **[Principal ideas/results]** A requirements modeling language with a diagrammatic syntax to be used to elicit and represent requirements for SAS and perform analysis based on our recently proposed core ontology to perform RE for SAS. **[Contribution]** A modeling language, called Adaptive RML, for the representation of early requirements for Self-adaptive systems (SAS). The language has graphical primitives in line with classical goal modeling languages and is formalized via a mapping to Techne. Early validation is performed by modeling the same case study in an established goal modeling language and in Adaptive RML. The results suggest that context and resource concepts, as well as relegation and influence relations should be part of graphical modeling languages used to make early requirements models for SAS and to perform analysis over them.

Keywords: Requirements Engineering, Requirements Modeling, Self-Adaptive Systems.

1 Introduction

A self-adaptive system (SAS) can change its behavior in response to anticipated and unanticipated variations in its operating context, its users' requirements, and the availability of its resources. Requirements engineering (RE) for SAS is receiving increasing attention in research and has been recognized as one of the key areas where progress is needed in order to enable the engineering of SAS [1].

Initial work on high-variability design in [2] models variability in user's goals and alternatives for goal achievement, which is reflected in the design and coding of Belief-Desire-Intention (BDI) agents. This work provided a basis to extend *Tropos* for adaptive

B. Regnell and D. Damian (Eds.): REFSQ 2012, LNCS 7195, pp. 263–279, 2012.

systems [3], where design abstraction like goal-conditions and environment modeling are added to *Tropos* goal models and correspondingly a mapping is provided to Jadex BDI architecture. This approach is confined to the design of adaptive BDI agents and requires fine grained knowledge about the domain to specify the alternative solutions and goal achievement conditions enabling the agent to switch its behavior in a given environment.

In [4], Whittle et al. proposed a language to represent uncertainty in requirements via fuzzy operators and using Fuzzy Branching Temporal logic as the underlying framework. In the context of KAOS [5], mitigation strategies are proposed to accommodate uncertainty and failures with obstacle analysis [6]. We proposed to engineer *adaptive requirements* using goal models and ontologies to make explicit the domain assumptions and requirements for feedback loop functions (i.e. monitoring, evaluation criteria, and adaptation alternative) [7]. Similar ideas were adopted by Baresi et al. [8] to extend KAOS goal models. The concept of *adaptive goals* has been introduced to specify adaptation strategies, while qualitative goals are relaxed by being replaced with fuzzy goals, the satisfaction of the former being binary, while the latter are associated to a continuous fuzzy membership function, the value of which is interpreted as the level of satisfaction of the fuzzy goal.

Ongoing research has also recognized the need to ensure that SAS have a runtime representation of their own requirements, i.e., that requirements should become artifact used, processed, and changed at runtime [7–10]. Considerable part of current research into the RE for SAS focuses on the specification of requirements for SAS, while there is comparatively less interest in what information should be part of *early* requirements models for SAS. In particular, how should early requirements models reflect (i) that there is uncertainty in the behavior and properties of the operating context and of the SAS, (ii) that the context of the SAS can vary and that this should influence the behavior of the SAS, and (iii) that resources of the SAS may vary, and that the SAS should adapt to those variables.

In our view, this requirements problem in case of SAS should be treated as a *dynamic RE problem*, where changes in requirements, contexts and resources lead to a new requirements problem – finding new candidate solutions to the changed requirements [11]. To fulfill this aim, we build on the core ontology for RE [12] and introduce two new concepts i.e. context and resource as well as two new relations, relegation and influence, formalized using Techne expressions [13] that are helpful in the early phases of RE to formulate the requirements problem for a SAS.

Based on this, we introduce here a new modeling language for early requirements for SAS, called *Adaptive RML*, to model the dynamic RE problem and perform analysis by finding candidate solutions. The aim of this paper is to introduce Adaptive RML, its concepts, relations, modeling guidelines and analysis features needed for early RE for SAS. This is a first attempt to provide a concrete RML for early RE that provides the necessary concepts and relations to model requirements for SAS and enables the analyst to perform analysis about the candidate solutions as function of contexts, where not only the conditions or resource demand changes but also the requirements problem changes. We motivate the need for Adaptive RML to model requirements for SAS using an example from a travel domain. As a preliminary validation, we model the example

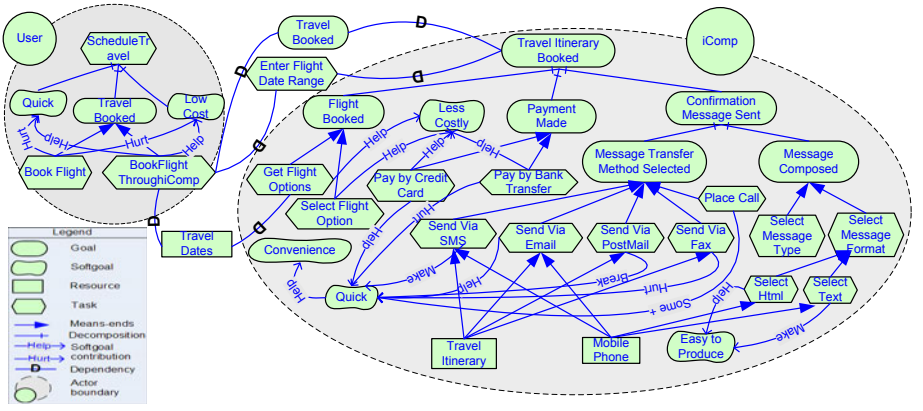

Fig. 1. Requirements Modeling using iStar Concepts and Relations only

first with *i** [14], identify the information needed for early RE of SAS and that cannot be modeled in *i**, then introduce Adaptive RML. In fact, we adopt *i** at this stage of the work due to its wide adoption in requirements modeling and to make Adaptive RML accessible to researchers using *i**.

The rest of the paper is organized as follows. Section 2 introduces the example modeled with *i**, and compares it with the same example in Adaptive RML. Section 3 presents Adaptive RML. Section 4 discusses Adaptive RML in light of related work. Section 5 summarizes conclusions and directions for future work.

2 Modeling the Requirements of *iComp*

In this section, we focus on a simple scenario for a travel booking application. We model it with the *i** requirements modeling language and identify key elements that are missing in this language for modeling requirements for SAS. We describe an excerpt of a scenario from a travel companion case study introduced in our prior work [7], in which self-adaptive properties of the system are illustrated.

Scenario: *The iComp application is a self-adaptive system that aids business travelers while on the move. It supports them in booking their travels, making payment and receiving timely updated information about their booking confirmation (e.g., confirmed, canceled, in progress). The booking confirmation messages must be sent to the user (customer) via Email or SMS instantly (in less than an hour or maximum less than 1 day) on their device (i.e. laptop or mobile) and depending on their context (e.g. home). Payment must be ensured before iComp sends the message (i.e. composing the message) by selecting a suitable message format (e.g., size, scaling, format) to adapt to the device from which they will be read. Finally, in case there are some problems (i.e. user is not accessible) and the message cannot be delivered to the user then iComp must send the message to an alternative recipient (e.g., the customer's secretary).*

2.1 Requirements Modeling with i*

Fig. 1 shows *i** strategic dependency and rationales models for the travel booking scenario. In the main scenario, the user and the system are represented by circles, whereby the content of the dashed ovals (strategic rationales) represents their goals, tasks, and resources. We can see in this model that what leads the user to chose the iComp for travel booking results from the analysis of the root task of `Schedule Travel`, which is decomposed into the goal of `Travel Booked` and the softgoals of `Low Cost` and `Quick`. These softgoals are negatively influenced (shown with contribution links) by the subtask of `BookFlight`. The task `Book Flight through iComp`, however, partially satisfices the `Low Cost` and `Quick` softgoals. This task in turn depends on the `iComp` Actor, since the associated goal `Travel Itinerary Booked` has been assigned to the system.

The strategic rationale model of iComp reveals a decomposition of goal `Travel Itinerary Booked` into three main goals `Flight Booked`, `Payment Made` - and `Confirmation Message Sent`. For example, we can now reason about `Confirmation Message Sent`, which is decomposed into two goals i.e. goals: `Message Transfer Method Selected` and `Message Composed`. Along their subsequent means of accomplishing tasks, and assess their contributions towards softgoals `Quick` and `Easy to produce`, which helps in ranking a particular solution. For example, tasks: `Send via SMS` and `Send via Email` with means to use resources: `Travel Itinerary` and `Mobile Phone` contributes fully and partially to the softgoal `Quick`, which in turn satisfices softgoal `Convenience`. The aim of this analysis is to identify one particular solution that satisfies the high level goals and optimally satisfies the softgoals.

Modeling iComp in *i** lead us to identify some limitations of the language when used for SAS. *i** does not provide concepts for the modeling of alternative solutions to the requirements problem, which are feasible in different contexts. For instance, in context (e.g. Home) the candidate solution should be *Send Message via Email* and in another context, e.g. Market, *Send message via SMS* should be more appropriate in case no 3G or no smartphone is available for the user, and so on. That is we were not able to model the fact that the context of the user may change as well as resources availability, and ultimately to capture monitoring conditions and evaluation criteria that should characterize the dynamically adaptive behavior of the system. Moreover, in *i**, it is not possible to model quality constraints, such as *send the message within one hour after the payment*, and domain assumptions that need to be made explicit during the analysis as they contribute to the definition of the requirement problem, such as *standard Credit Card Options must be Displayed*.

Efforts has been made to capture requirements for SAS by extending *i*/Tropos* [3, 15]. The main idea behind these extensions is to annotate goal models. For instance, in [3] goal achievements conditions and environment modeling (using UML class diagrams) is used to annotate the *i*/Tropos* goal model, and transform them for use with an implementation architecture (e.g. BDI). Similarly in [15] location abstraction is used to formalize context and annotating the variation point (e.g. AND/OR decomposition) within a goal model. This approach provides a systematic design-time approach to build context models based on locations concepts (e.g. using UML class diagrams). Common

to both approaches is the use of UML notation to formalize the concept of environment and context hierarchies. Both approaches are focused on finding a single best solution in case of adaptation. Moreover, both of the approaches are limited to show how the system can move across contexts (with changing domain assumptions, resource availability) by altering the requirements problem with respect to the variety of candidate solutions.

2.2 Requirements Modeling with Adaptive RML

Differently from the previously mentioned extensions of goal-oriented modeling languages for SAS, we rest on Techne [13], an abstract modeling language for early requirements, which adopts the core ontology for RE [12]. This core ontology extends the goal-oriented perspective allowing to model optional requirements, preferences, and to treat non-functional requirements in terms of approximations and quality constraints. The basic elements of Techne models are requirements, modeled as natural language propositions that are labeled as domain assumptions, goals, quality constraints, or tasks. A requirement can be mandatory or optional. Links between model elements are used to represent how the satisfaction of an element may impact the satisfaction of the other, through inference and conflict. Preferences are used to compare requirements in terms of desirability. Performing the analysis of a requirements problem specified in Techne, results in finding candidate solutions in terms of tasks and quality constraints that together satisfy all mandatory goals and cover, as much as possible, optional ones.

The proposed modeling language for SAS, called Adaptive RML, builds on Techne by adding two new concepts, namely, context and resource, and two relations, i.e. relegation and influence. Adaptive RML has its own visual notation. In the rest of this section we illustrate an Adaptive RML model of iComp with the aim to provide a preliminary qualitative evidence about its support in overcoming the limits mentioned above in modeling requirements for SAS. A detailed account of Adaptive RML will be given in the following sections.

Fig.2, shows a requirements model for iComp in Adaptive RML (in form of a Techne r-net). Its root level goal `Travel Itinerary Booked` is modeled as a mandatory node (*modeled as **M** node, a unary relationship*). It is decomposed via an a binary inference relation (*modeled as black **I** node with a arrow*) into the other mandatory goals: `Flight Booked`, `Payment Made` and `Confirmation Message Sent`, to represent the fact that it will be satisfied through the joint satisfaction of these three goals.

Let's focus on the goal: `Confirmation Message Sent` (i.e. the shaded part of the model), which is decomposed into two goals: `Message Transfer Method Selected` and `Message Composed` via inference relation. We can add here information that were missing in *i** model, i.e. the domain assumption `Booking Confirmation is sent after the payment is assured` (*modeled as rounded rectangle*) and the quality constraint `Message sent in < 1 hour after the payment` (*modeled as diamond shape*) connecting them through the same inference node.

An influence relation is added among the two decomposed goals: `Message Transfer Method Selected` and `Message Composed` (*modeled as dotted green line with arrowhead*) to account for the prevailing context conditions and resource

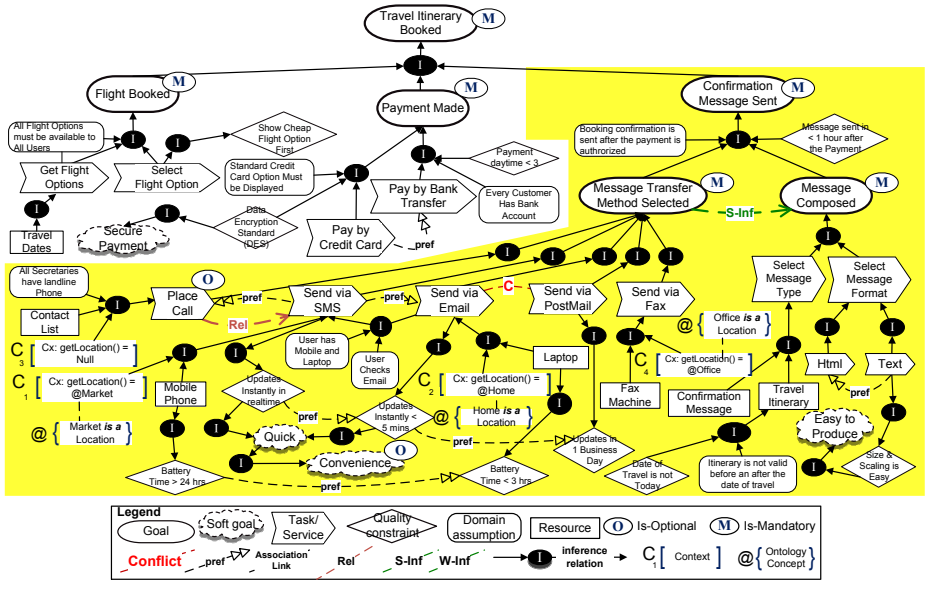

Fig. 2. Modeling using Adaptive RML Concepts and Relations

availability that influences the achievement of goal: `Message Composed`. For example, if the context conditions support to choose Email as a candidate transfer method, the ways to satisfy goal:`Message Composed` is by selection a correct format that is either text or html.

The analysis of the `Message Transfer Method Selected` proceeds by linking via inference nodes task-rooted subgraphs, which defines candidate solutions. Besides tasks e.g. `Send via SMS` (*modeled as hollow motion arrow*), each candidate solution includes domain assumptions e.g. `User has mobile and laptop`, context e.g. `Market`, `Home` (*labeled as* ***C*** *with its number, associated to @ symbol*[1]), and resources e.g. `Mobile Phone` (*modeled as a rectangle*). Preferences (*dotted line with doubled empty arrow heads*) are used to compare requirements in candidate solutions, and thereby compare candidate solutions; e.g., `Send via SMS` is preferred over `Send via Email`. Requirements can be in conflict (e.g. `Send via Email` is in conflict with `Send via PostMail`. *Conflict is shown as a dotted line with* ***C*** *in the middle with red color*). Here, conflict exist due to the difference in the quality constraints e.g. `email updates in < 5 mins, whereas post mail updates in 1 business day`. Notice that it was not possible to model these information with *i**.

Optional solutions, in case of problems (e.g. user is not accessible, as mentioned in the scenario) can be identified via a relegation relation (*drawn as dotted light red line with arrowhead* between two possible candidate solutions). For instance, `Place Call` relegates `Send via SMS`. This allows to take into account the situation in which a user's context changed resulting in being not accessible

[1] @ labels a concept defined in domain ontology e.g. travel.

(e.g. `C3 [Cx:getLocation()= Null]`), and to describe as preferred the solution to make the user able to access the resources `Confirmation Message` and `Ticket Itinerary`, via contacting her secretary. The `Place Call` task is inferred via a domain assumption (e.g. `All secretaries has landline phone`) and a resource (e.g. `contact list`) and the context (e.g. `C3 [Cx:getLocation()= Null]`).

Summary. Gain in expressiveness of Adaptive RML with respect to *i** models are summarized below:

- we can model information about context, resources and domain assumptions that need to be monitored by the SAS in order to enable adaptation;
- softgoals evaluation in *i** is subjective and provides no clear evidence to rank a solution. In Adaptive RML, candidate solutions can be ranked and evaluated via quality constraints over measurements that may be collected by the SAS;
- candidate solutions can be associated with contexts and requires resource.

3 Definition of Adaptive RML

3.1 Concepts and Relations

We define the concepts and relationships in order to formulate the requirements problem for SAS. Addition of these concepts and relations leads us to an ontology for requirements in SAS and the formulation of the *runtime requirements adaptation problem* as a dynamic problem of changing (e.g. switching, re-configuring, optimizing) the SAS from one requirements problem to another requirements problem, whereby the changing is due to change in requirements, context conditions, and/or resource availability [11]. We add two new concepts, Context and Resource as well as relations Relegation and Influence that enhance the tool set for the proposed Adaptive RML to model and analyze requirements for SAS.

Context: This concept allows modeling information that the stakeholders assume to hold when they communicate particular requirements. We say that every requirement depends on one or more contexts to express the fact that the requirement would not be retracted by the stakeholders in every one of these contexts. This information needs to be made explicit in the early requirements model for SAS. For instance, in our example we modeled "context" as information about location (e.g. Office or Market), which are defined as concepts in a specific domain ontology (e.g. travel), and we linked them to tasks via an inference relation. In Fig.2, context is shown as e.g. "C1 [Cx:getLocation()= @Market]" where "@Market" is an instance of a concept term (i.e. Location) defined in a domain ontology. Combining requirements and context reveals interesting cases, where we can see requirements maybe in conflict.

Resource: The concept of resource has been well supported in RE methods such as in goal-oriented approaches [2, 5, 14]. In our case, we define it as an entity that is referred to by the requirements, e.g., physical/tangible entities such as mobile phone, ticket itinerary; e.g., intangible entities, such as user assets (social relations or contacts). In order to introduce resources in the definition of the requirements adaptation problem for SAS, we need to elicit a resource availability function that tells us which resources

Visual Notation	Concepts & Relations
Goal	**Definition:** A **Goal** represents a desired state of affairs, the achievement of which can be measured and is definitively concluded. **Example:** "*Meeting to be Scheduled*"
Soft goal	**Definition:** A **Soft goal** represents a desired state of affairs, the achievement of which can only be estimated, not definitively concluded. **Example:** "*Convenience*", "*Easy*"
Task/ service	**Definition:** A **Task** corresponds to an activity, an action whose achievement leads to the definitive conclusion of its means. **Example:** "*Download music*", "*Show song listed as most viewed*"
Quality constraint	**Definition:** A **Quality constrain**t is desired value of non-binary measurable properties of the system-to-be that constrains a goal or a soft goal. **Example:** "*Music download speed must not be less than 128kbps/sec*"
Domain assumption	**Definition:** A **Domain assumption** is a condition within which the system-to-be will be performing tasks in order to achieve the goals, quality constraints, and satisfy as best as feasible the soft goals. **Example:** "*Subscribers can download the music from the online database*"
inference relation	**Definition:** An **<Inference> relation** stands between a requirement that is the immediate consequence of another set of requirements, the former is called the conclusion, the latter the premises. Alternatively, inference relation can be used to connect the refined requirement to the requirements that refine it. **Example:** "*Generate revenue from the audio player*" has <inference> relation with two requirements: "*Music is available to subscribers*", "*Display ads in the player*".
Conflict	**Definition:** A **<Conflict> relation** stands between all members (two or more) of a minimally inconsistent set of requirements. **Example:** "*Req1: Music is available to subscribers*" is in <Conflict> with "*Req2: Music is available to users*"
pref	**Definition:** A **<Preference>** is a binary relation that exists between two requirements and it defines the stakeholder evaluations of requirements that determine the desirability of a requirement. **Example:** "*The bitrate of music delivered via the online audio player should be at least 256kb/s*" is <Preferred> over "*the bitrate of music delivered via the online audio player should be at least 128kb/s*"
O Is-Optional	**Definition:** An **<is-Optional> relation** is unary that states the evaluation of stakeholder of requirement, which may be desirable. Functional requirements, which are "nice to have". **Example:** "*Color printing of a meeting schedule*" <is-Optional>.
M Is-Mandatory	**Definition:** An **<is-Mandatory> relation** is unary that states the evaluation of stakeholder of requirement, which must be satisfied. Functional requirements. **Example:** "*Each Participant must have meeting schedule available*" <is-Mandatory>.
Association Link	**Definition:** An **<Association> link** is used to define a link between two elements. **Example:** "*High level Context (e.g. Outdoor)*" is <associated> to "*an ontology concept (e.g. place)*".
Relegation Relation	**Definition:** A **<Relegation> relation** is n-array relation that stands between one or more requirements, to relax or to suspend conditions imposed over them. A mandatory requirement can have a <relegation> relation with an optional requirement. **Example:** "*download the music*" has <Relegation> relation with the "*stream the song online*".
Weak-Influence / Strong-Influence	**Definition:** An **<Influence> relation** is said to exist between a set of requirements, where satisfaction of one requirement warrants the satisfaction of the other. This determines the satisfaction of the requirements set. There are two types, weak-influence (where partial satisfaction is possible) and strong-influence (when there is no way to satisfy the requirement). **Example:** "*subscribe and pay*" have <Strong-Influence> over the "*download the music*". "*subscribe and make payment*" have <weak-Influence> over the "*listen music online*"
Resource	**Definition:** A **Resource** is an entity either tangible or intangible referred to by one or more instances of the information communicated during elicitation by the stakeholder. **Example:** Tangible Resource: "*Physical e.g. Mobile phone*" Intangible Resource: "*Data e.g. Agenda*"
Requires	**Definition:** A **<Requires>** relation is a binary relation that exists between a task and a resource. **Example:** "*Task: Download song*" <requires> "*Resource: internet connection*"
C_1 [Context]	**Definition:** A **Context** is defined as a set of information (condition) that is presupposed (or believed to be true) by the stakeholders to hold when they communicate a particular requirements. **Example:** "*System states (e.g. searching a song)*", "*User states (e.g. Listening to music)*", "*User Location (e.g. at home)*", "*Device Status (e.g. Battery is low)*"
@ { Ontology Concept }	**Definition:** An **Ontology Concept** defines an entity and its characteristics or essential features in a particular domain of discourse. **Example:** "*Frame rate in Music Ontology*"

Fig. 3. Visual guide for concepts and relations in Adaptive RML

are available and used in some way, in order to ensure that the relevant domain assumptions and context propositions hold, and that the tasks can be executed. Here again we may exploit ontology definitions of user-assets and asset modifiers that represents tasks effects on their resources, as proposed in [16]. In the modeled example shown in Fig.2, we introduced "Mobile Phone" and "Laptop" as resources available in different contexts.

Relegation Relation: The purpose of the Relegation relation (`Rel` for short) is twofold. First, it facilitates engineer at design-time to analyze requirements (including goals, quality constraints, preferences) and relegate their associated conditions (e.g. pre/post, achievement, trigger conditions) by anticipating runtime change scenarios. Secondly, it enables SAS at runtime to analyze requirements problem in case of changes that can occur dynamically e.g. change in user's context, violation of domain assumption, resource usage or change in user's need or preference, either through sensing the operational environment or explicitly given by the end-user.

A `Rel` is applied to manage unanticipated events, by flexibly relegating some of the requirements, with the aim to avoid failure in achieving the critical ones. In this case by applying `Rel`, either the solution that operationalizes a goal needs to be replaced, or an instance of the same goal with revised conditions is linked using `Rel` with the original goal e.g. in Fig.2, candidate solution "Send via SMS" is relegated by "Place Call", when context conditions changes. In this example, the instance of the original goal is not compromised rather relegation is considered by replacing the preferred solution with an optional solution.

Influence Relation: An influence relation (`Inf`) is introduced to analyze the impact of changes in model elements that define different, mutual dependent requirements. This means, if change in the operational environment or in end-user requirements happens at runtime it might cause a change in another requirement. This chain of dependency needs to be identified, since along them we may identify changes consequences such as violation of a goal or a invalid solutions. For example, in Fig.2, if no candidate solution is possible to achieve the goal "Message Transfer Method Selected" due to invalid context and domain conditions, then this goal will fail, which causes a violation in satisfying the corresponding goal i.e. "Message Composed". Similar dependencies can be collected and subsequent consequences are determined by analyzing the impact of changed solution.

3.2 Adaptive RML Visual Notations

The Adaptive RML language provides a graphical notation, which is in line with classical goal modeling languages and is formalized via a mapping to Techne. A detailed guide on visual elements is presented in the Table shown in Fig. 3: each row contains a graphical symbol and a short description of it's intended meaning. For the elements that map the Techne core ontology, the corresponding semantics is given in [13], while the formal semantic of the additional concepts is defined in [11].

Worth to be mentioned is that recent research evaluated weaknesses of widely used goal-oriented modeling notations with respect to principles for cognitively effective visual notations [17]. The proposed visual notation considers two among the principles

discussed in [17]. The first is visual expressiveness: notation must comprise of color, shape and brightness instead of shape only. Second is Semiotic clarity, which postulates that each graphical symbol must have a 1:1 correspondence with its semantic definition. Our proposed notation takes as much as possible these principles into account, but further effort is needed to fit with the proposed recommendation for improving usability and communicative effectiveness of visual notations in RE modeling.

3.3 Modeling in Adaptive RML

Modeling requirements in Adaptive RML enables the analyst to construct the requirements model by recording and structuring relevant information obtained through elicitation. As a result, the runtime requirements adaptation problem is formulated for the SAS-to-be. New pieces of information are gathered during modeling time to refine the problem iteratively. At analysis time, all candidate solutions to that problem are sought along with their differences to each other and are compared with respect to varying context situations and resource availability.

The modeling process develops by performing iterations of the following activities.

1- Modeling Mandatory and Optional Goals:

We start modeling goals, optative statements that defines the desired properties of the SAS-to-be, via inference relation (i.e. symbol (I)). We use (I) node to depict refinements (e.g. AND/OR decomposition, or means-end relation). Each (I) node connects the model element to be refined to simpler or more concrete elements that refine it. In this way it is concluded that if the requirements defined by the concrete elements are satisfied then the more abstract one will be achieved. Further, we add softgoals vague properties of SAS-to-be, which are approximated in terms of quality constraints that determines the criteria to measure them. Goals can be either mandatory or optional (i.e. (M)) or (O) respectively), we model this by adding these unary relation over goals.

2- Modeling Domain Assumptions:

While modeling goals we discover domain assumptions that are statements in the domain which are assumed to be always true. We add them via (I) node and add (if any) to each goals. Subsequently, during refinement, quality constraints can be inferred. We add criteria to measure the goal satisfaction via (I) node. During this, new pieces of information are discovered such as conflicts and preferences among the goals.

3- Modeling Conflicts and Preference Relations:

Conflicts and preferences are identified during refinement. We discover conflicts between inconsistent / contradictory requirements or tasks node between conflicting set of requirements / tasks. Further, we identify preferences taking into account stakeholder's evaluations about different requirements. We add preference relation between requirements where satisfying one is strictly more desirable than satisfying the other.

4- Modeling Mandatory or Optional Tasks:

Likewise, we model tasks as further refinement of goals. Task modeling can be seen as an analysis activity, where we add tasks via (I) node to operationalize goal. This means, if the tasks will be successfully completed, the goal will be achieved.

5- Modeling Context and Resources:

Once the requirements model is constructed, we further anticipate the various situations in which requirements or tasks can be either achieved or not. We add *context* node to each requirement/task. Context refers to any information, which is presupposed by the stakeholder and we make it explicit, e.g. a location etc.. A domain ontology compliments this context information by precisely defining the terms (instances of context). We link context with an ontology annotation (shown as @) via an association link.

While discovering tasks and context that can satisfy requirements, we may also identify **resources** that the tasks need to use. We add *resource* node via (I) node with each task. Note that resource concept is also available in other RML, however, we distinguish it as not only tangible e.g. mobile phone, Fax machine, but also intangible e.g. assets such as money, time, agenda. In our model, each resource may have domain assumptions or quality constraint attached to it via (I) node.

6- Modeling Influence and Relegation Relations:

Finally, identify during refinement requirements/tasks may have influence on the achievement of each other. *Influence relation* is added between a set of requirements/ task, where the achievement of the former becomes critical due to the achievement of others (strong influence i.e. s-inf). If achievement of the latter is not critical, it will be modeled as weak (w-inf). However, it becomes interesting in case of tasks, where execution of one tasks may have influence of other tasks.

Finally, we look for conflicting context conditions, resource availabilities, quality criteria which may helps to determine requirements/tasks whose achievement can be delayed or relaxed. We add *relegation relation* between requirements/task that are less critical to the requirements/task more critical/preferred to in corporate uncertainty about changes in context or resource availability.

3.4 Towards Detailed Specification Analysis

Analysis in Adaptive RML suggests which candidate solutions are relevant in the prevailing context conditions and resource availability. A requirements model defines the requirements problem for a SAS-to-be, along with candidate solutions. This model is used by the analyst to discover *adaptive requirements* by looking at differences between candidates solutions that are modeled.

Adaptive requirements are requirements that not only hold the definition of functional or non-functional requirements but encompass the notion of variability, by having monitoring specification, evaluation criteria and adaptation alternatives. To discover them detailed analysis is performed on the available information represented in the early requirements model. We analyze the candidate solutions that remain valid in a particular situation. We look at the context nodes and domain assumptions, we anticipate changes as we move to a different context and this leads to different resource availability requirements. Alternative solutions can be inferred during this process.

Adaptive requirements help specifying alternative ways to adapt to context and resource changes via a pattern, details of which are out of the scope of this paper. Consider, while monitoring runtime changes, SAS moves across different contexts by altering the requirements problem that leads to change in candidate solutions. At runtime, several solutions get activated based on context and based on resource availability. Mechanisms

for adaptation are triggered, therefore, reasoning over the adaptive requirement leads SAS moves (i.e. enact adaptation) to the candidate solution which is appropriate to the new current context.

For example, an adaptive requirement can be defined as **AR1:** *Message must be composed by selecting an appropriate format*. From this we determine that appropriate format i.e. HTML or Text, needs to be selected as modeled in Fig.2. But to select the candidate solution, we need to *monitor* the user's context (e.g. Office, Home) and resources (e.g. Mobile phone or Laptop) and domain assumptions with quality preferences. Along monitoring specification, we need also to specify *evaluation criteria* to check the difference between two tasks. Based on this criteria, among the possible candidate solutions that are adaptation actions *e.g. tasks and domain assumptions in a context*, a possible candidate solution will be selected. For instance, while monitoring the user context, resource, any change can lead to change the selected format, i.e. either html or text format.

So far, we argued on the need of a requirements modeling language (RML) for SAS that enable the analyst to capture and analyze requirements for SAS by incorporating the above core properties of SAS at early stages of RE. Below we present how the SAS at runtime tries to resolve a runtime requirements adaptation problem, by finding and comparing a candidate solution in response to changing context, resource variability using its own requirements model and detailed specification i.e. adaptive requirements.

3.5 Detailed Specification at Runtime

We recognize that in case of SAS, not all information can be collected, defined during requirements- or at design-time, but that this will depend at runtime when the system exploits its solutions implemented using different technologies (e.g. exploiting available services or agents). For example, any variation in the context and resource availability can be monitored or recorded by gathering the data through sensors, then matching patterns of data provides implications on the satisfaction of the goals. However, regardless of the technologies used, the SAS still needs to be designed to ensure the general conditions and relations that the requirements problem states: e.g., that the SAS needs an internal representation of information pertaining to contexts, domain assumptions, tasks, goals, and so on.

To give an intuition about how the adaptive requirements specification can support runtime adaptation, in Fig.4, an adaptation sequence is shown along the time dimension, where the SAS operates as per the candidate solution (S1) selected to satisfy the particular context and resource variation. At this time (t1) the SAS, while monitoring, evaluates the user's current situation and attempts to satisfy a given set of goals (e.g. *Confirmation Message Sent*, *Message Transfer Format Selected*) and quality constraints via its candidate solution. A candidate solution is composed of tasks, domain assumptions that hold valid for a context and available resources to achieve such tasks. E.g., candidate solution **S1**: *Context: (@Market), Resource: (Mobile Phone), Task: (Send via SMS), Domain Assumption: (All Users have Mobile Phone & Laptop)* was selected, but due to traveling, the context is not recognized anymore. Therefore the SAS has to reason about this change at time (t2) by looking at the difference in candidate solutions with respect to context conditions, resource availabilities and user preferences.

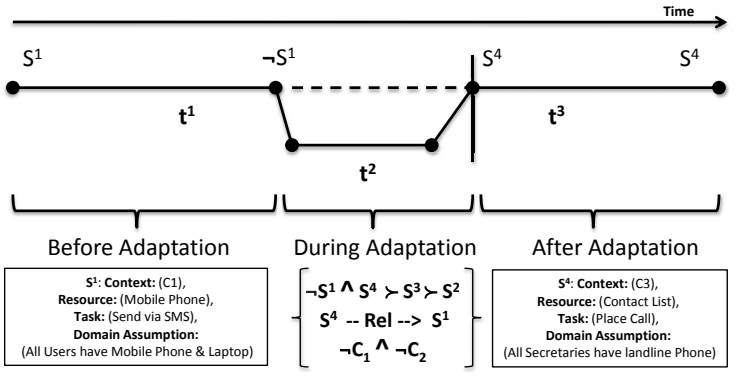

Fig. 4. Runtime Adaptation Sequence of SAS

SAS performs the reasoning based on the differences among the alternative candidate solutions, which states a comparison and ranking of the solutions based on criteria e.g. (S1) Send via SMS is not valid, (S2) Send via Email is not feasible as user's context is not recognized. Thus the change in requirements problem, changes the candidate solution in different contexts and with different resources. The adaptive requirements play critical role here, as they operationalize the mechanisms for adaptation i.e. monitor and evaluating the difference between candidate solutions and provides criteria to compare and rank them. To reason on adaptive requirements, automated reasoning techniques (e.g. AI Planning) can be employed. Discussion on such techniques is out of the scope of this paper.

Finally SAS selects a candidate solution e.g. "Place Call" by evaluating the relegation relation, specified earlier in the adaptive RML model and detailed in adaptive requirements e.g. S4 $\succ$ S3 $\succ$ S2. The new candidate solution **S4**: *Context: (Null), Resource: (Contact List), Task: (Place Call), Domain Assumption: (All Secretaries have landline Phone)*.

4 Discussion and Related Work

Advantages and open aspects of the proposed language are discussed with respect to state-of-art work and along well recognized issues in requirements modeling for SAS, which includes uncertainty about environment conditions and resource availability, context awareness and monitoring, requirements reflection and runtime reasoning. Adaptive RML provides visual notations to the concepts defined in the revised core ontology of RE for SAS. On the correctness of the concepts used to model requirements, we refer to the definitions in [11].

Systems that operate in an open environment, need to be able to manage uncertainty about environment conditions and resource availability. So for instance our system has to be designed in a way that it can communicate through SMS if the cell phone is on and connection is available, and if not, choose a different way to communicate. An attempt to address this problem at requirements time has been proposed within the RELAX

framework [4], through the use of a language that provides three types of operators to handle uncertainty: temporal (e.g. eventually, until, as early as), ordinal (e.g. as close, as many), and modal (i.e. shall, may / or). The RELAX language semantics is formalized in Fuzzy Branching Temporal Logic. In [6], a set of analysis methods are then provided to support goal modeling refinement towards detailed design, which exploit mitigation strategies based on obstacle analysis, and lead eventually to relax constraining conditions (i.e. our quality conditions). Analogously, the approaches proposed in [3] and [8] propose interesting methods to deal with uncertainty at detailed design.

In Adaptive RML, we provide, the Relegate relation, which is more general than the RELAX operators [4], since we do not commit to fuzzy logic: we only ask for a way to represent alternatives and to compare them. In this sense, RELAX can be seen as a particular way to implement the Relegate relation, and obtains a straightforward interpretation in the language we used here. There are other ways to handle uncertainty and relaxation of requirements, and our aim in this paper was to remain independent of particular approaches.

Concerning the knowledge about the ***resources***, which are needed to achieve specific behavior while the SAS is operating, this notion of resource has been implicit in the requirements modeling languages like KAOS and *i*/Tropos*. In case of requirements for SAS, we believe that it is necessary to model resources in a more explicit way, not only to express their variability, but also to include dynamic lifecycles that might describe their availability.

Along the dimension of ***context***, in RE, context has been defined as*An abstraction of location, an event, environment or as a set of conditions that may change overtime* in [15, 18, 19]. Another common and well accepted definition of context to date is by Dey in [20], i.e., *Context is any information that can be used to characterize the situation of an entity*.

Specifically, in RE for SAS, it has been argued that alternative behaviors must be supplied to the system, which can be switched to meet the changes in the environment by monitoring the context [18]. To capture the contextual variability, explicit knowledge about the domain is required. In [15], variation points are used to annotate the goal models, for representing pre-defined contexts and alternative behaviors to be exploited while reasoning over them. To use this approach, a requirements driven reconfiguration architecture is proposed in [21], which leverages the concept of context and monitor-diagnosis-compensate loop. Moreover, our Adaptive requirements, follow similar ideas, but go beyond the above mentioned approaches by making explicit domain assumptions and requirements for feedback loops [7]. However, the notion of context is trickier and brings newer requirements to be analyzed while specifying requirements for SAS. In Adaptive RML, we provided an explicit graphical notation, where context properties can be modeled exploiting specific domain ontology, which defines the domain concepts and their instances.

On the basis of recent works, we recognized issues in requirements modeling for SAS that provide premise to the proposal of Adaptive RML. For instance ***requirements monitoring*** [22–24], where the running systems must be monitored during its execution as per its own requirements model. Any runtime deviation or violation leads to needs for the system to reconcile its behavior to its requirements. In case of SAS this is

critical, as it operates in an open environment where changes can occur dynamically in the operating context, availability of resources and end-user needs can change over time. In Adaptive RML, we propose modeling concepts so as to model early requirements for SAS, which then guides the detailed specification, which will eventually include monitoring specification. However, implementation of monitoring and linking early models with runtime events is nontrivial.

Requirements reflection is another issue, where ideas from computational reflection has been borrowed to provide SAS the capability to be aware of its own requirements [10]. Similarly, online goal refinement [25] is of prime importance considering the underline architecture of the intended SAS. To support runtime reasoning of requirement by SAS itself, in [9, 26, 27], we proposed a Continuous Adaptive RE (Care) framework and architecture for continuous online refinement of requirements by the system itself. This work describes different types of runtime adaptation, which are realized by exploiting incremental reasoning over adaptive requirements represented as runtime artifact. The main aim of this framework is to provide continuous refinement of requirements and provide solutions (i.e. leveraging available services) by the system at runtime involving the end-user.

Adaptive RML models and their support in deriving detailed specification in terms of adaptive requirements, represents a relevant contribution towards realizing continuous adaptive requirements engineering.

5 Conclusion and Future Work

This paper introduced Adaptive RML, a visual language for the modeling of early requirements for SAS. In contrast to previous proposals [3, 6–8] that rest on well established goal-oriented modeling languages (i.e. *i**, Tropos, Kaos), Adaptive RML builds on the abstract requirements modeling language Techne [13], which provides a richer set of concepts, along the CORE ontology for RE defined in [12], and supports requirements analysis leading to sets of candidate solutions for the stated requirements problem. A few additional concepts and relationships are used in Adaptive RML (i.e. context, resource, relegation and influence) to model and represent the runtime requirements adaptation problem and perform analysis.

The motivations for Adaptive RML were first introduced by contrasting requirements modeling of an example of SAS, made with *i** and with Adaptive RML, providing also an early qualitative validation of its advantages. A detailed account of Adaptive RML was then given in terms of concepts, visual notation, modeling and analysis guidelines. Finally, novel features of Adaptive RML were discussed along the research challenges, which have been recently identified in RE for SAS [1, 10] and open points were highlighted.

As future work on Adaptive RML, we will focus on investigating easier-to-use visual syntax, tool support for modeling and automated reasoning methods for the analyst to find candidate solutions in the model. To further consolidate the approach, a systematic process to guide the detailed specification in terms of adaptive requirements should also be provided. A survey is also planned to acquire feedback on the effectiveness of the proposed visual modeling notions and their adequacy for early requirements modeling of SAS involving subjects.

References

1. Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Di Marzo Serugendo, G., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H.M., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H.A., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., Whittle, J.: Software Engineering for Self-Adaptive Systems: A Research Roadmap. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Self-Adaptive Systems. LNCS, vol. 5525, pp. 1–26. Springer, Heidelberg (2009)
2. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: High variability design for software agents: Extending Tropos. TAAS 2(4) (2007)
3. Morandini, M., Penserini, L., Perini, A.: Towards goal-oriented development of self-adaptive systems. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008), pp. 9–16 (2008)
4. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.-M.: RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In: 17th IEEE Int. Requirements Eng. Conf., Atlanta, pp. 79–88 (2009)
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. 20(1-2), 3–50 (1993)
6. Cheng, B.H.C., Sawyer, P., Bencomo, N., Whittle, J.: A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 468–483. Springer, Heidelberg (2009)
7. Qureshi, N.A., Perini, A.: Engineering adaptive requirements. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2009), pp. 126–131 (2009)
8. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: 18th IEEE Int. Requirements Eng. Conf., pp. 125–134 (2010)
9. Qureshi, N.A., Perini, A.: Requirements engineering for adaptive service based applications. In: 18th IEEE Int. Requirements Eng. Conf., pp. 108–111 (2010)
10. Sawyer, P., Bencomo, N., Whittle, J., Letier, E., Finkelstein, A.: Requirements-aware systems a research agenda for re for self-adaptive systems. In: 18th IEEE Int. Requirements Eng. Conf., pp. 95–103 (2010)
11. Qureshi, N.A., Jureta, I., Perini, A.: Requirements Engineering for Self-Adaptive Systems: Core Ontology and Problem Statement. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 33–47. Springer, Heidelberg (2011)
12. Jureta, I.J., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: 16th IEEE Int. Requirements Eng. Conf., pp. 71–80 (2008)
13. Jureta, I.J., Borgida, A., Ernst, N.A., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: 18th IEEE Int. Requirements Eng. Conf., pp. 115–124 (2010)
14. Yu, E.: Towards modeling and reasoning support for early requirements engineering. In: Proc. 3rd IEEE Int. Symp. on Requirements Eng., pp. 226–235 (1997)
15. Ali, R., Dalpiaz, F., Giorgini, P.: A Goal Modeling Framework for Self-contextualizable Software. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 326–338. Springer, Heidelberg (2009)
16. Marchetto, A., Nguyen, C.D., Di Francescomarino, C., Qureshi, N.A., Perini, A., Tonella, P.: A design methodology for real services. In: Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems, PESOS 2010, pp. 15–21. ACM (2010)

17. Moody, D.L., Heymans, P., Matulevicius, R.: Improving the effectiveness of visual representations in requirements engineering: An evaluation of i* visual syntax. In: 17th IEEE Int. Requirements Eng. Conf., pp. 171–180 (2009)
18. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In: 15th IEEE Int. Requirements Eng. Conf., pp. 211–220 (2007)
19. Finkelstein, A., Savigni, A.: A framework for requirements engineering for context-aware services. In: Proc. of 1st International Workshop From Software Requirements to Architectures (STRAW 2001), pp. 200–201 (2001)
20. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (2001)
21. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: An Architecture for Requirements-Driven Self-reconfiguration. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 246–260. Springer, Heidelberg (2009)
22. Fickas, S., Feather, M.S.: Requirements monitoring in dynamic environments. In: RE 1995: Proceedings of the Second IEEE Intl. Symp. on Reqs. Eng., p. 140. IEEE CS (1995)
23. Feather, M.S., Fickas, S., Lamsweerde, A.V., Ponsard, C.: Reconciling system requirements and runtime behavior. In: IWSSD 1998: Proceedings of the 9th International Workshop on Software Specification and Design, p. 50. IEEE CS (1998)
24. Robinson, W.: A Roadmap for Comprehensive Requirements Monitoring. Computer 43(5), 64–72 (2009)
25. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: Future of Software Engineering, FOSE 2007, pp. 259–268 (May 2007)
26. Qureshi, N.A., Perini, A., Ernst, N.A., Mylopoulos, J.: Towards a continuous requirements engineering framework for self-adaptive systems. In: RE 2010 Workshops, First International Workshop on Requirements@Run.Time (RE@RunTime), pp. 9–16 (2010)
27. Qureshi, N.A., Perini, A.: Continuous adaptive requirements engineering: An architecture for self-adaptive service-based applications. In: First IEEE International Workshop on Requirements@Run.Time (RE@RunTime), pp. 17–24 (2010)