Engineering Adaptive Requirements

SEAMS- 2009
(May 18-19 2009)

By: Nauman A. Qureshi¹
Anna Perini²

Fondazione Bruno Kessler (FBK-IRST)¹,² / University of Trento, Italy¹
Premise

- **Self-adaptive systems** can **configure** and **reconfigure** themselves, **augment** their **functionality**, continually **optimize** themselves, **protect** themselves, and **recover** themselves, while keeping most of their **complexity hidden** from the user and administrator. “Dagstuhl RoadMap 08”

- Research focus on Design-time to Run-time adaptation
  - **RAINBOW** [Garlan 04 et al.]
  - **Self-Managed Systems: An Architecture Challenge** [J. Kramer 07 et al.]

- Less attention on Requirements engineering for self-adaptive software systems. “Dagstuhl RoadMap 08, Betty Cheng et al.”
Setting up the Stage

- Introduction
  - Example Scenario
  - Research Objectives
- Research Baseline
- Adaptive Requirements
  - Characterizing Adaptive Requirements
- Capturing Adaptive Requirements
  - Steps for Eliciting Adaptive Requirements
  - Specification of Adaptive Requirements
- Conclusion & Future Work
Introduction

- Example Scenario *(Travel Companion)*

How the requirements for self-adaptive system will look like?

**User Goal:** To enjoy Convenient & Comfortable Travel.
Example Scenario *(Travel Comp.)*:

- To help clarifying how the analyst can capture variability and flexibility that identify requirements for self-adaptive system

- **Planning**: Travel booking (involving the user)
- **Monitoring**: Booking, User Context, Assets, Events
- **Evaluating**: Decide about Change, Some Uncertain Event
- **Adapting**: Enact adaptation with the chosen behavior
Research Objectives

- **First** objective is to support the system analyst to elicit requirements for self-adaptive systems at requirements-time
  - What are the requirements for self-adaptive system?
  - How to support analyst to elicit and specify them?

- **Second** is to make software able to reason on requirements at run-time in order to enable a goal-oriented adaptation
  - How to make requirements as live artifact enabling the software to reason on for adaptation?

There is a need to make explicit the variability in the requirements for self-adaptive systems

We call them Adaptive Requirements!
Research Baseline

- Goal Oriented Requirements Engineering “GORE”
  - High Variability Design [Penserini 07 et al.]
  - Goal Oriented development of SAS [Morandini 08 et al.]
  - Automatic Software Configuration [Liaskos 05 et al.]

- Knowledge Representation “Ontologies”
  - Redefining the core RE ontology [Jureta 08 et al.]
  - GOORE: Goal-Oriented and Ontology Driven Requirements Elicitation Method [Shibaoka 06 et al.]

- Variability design approaches
  - Problem frame extension to (monitor and switching) reason on problem variants [Salifu 07 et al.]

- Natural Language to Specify requirements for adaptive systems
  - Using Modal Verbs [Whittle 08 et al.]
Adaptive Requirement Characterization

- **Adaptive Requirements:**
  - By adaptive requirements, we mean that a requirement encompasses the notion of variability and flexibility in it, while elaborating either a functional or quality aspects of the software system.

- **To make explicit the variability, we consider:**
  - Uncertainty in time
  - Variability in user Context (Profile, location and Resource)
  - Variants of behavior (Monitoring Parameters) based on domain assumptions
  - Alternatives “ OR “ predefined before execution / exploiting user’ assets

- **An Example:**
  - **[Req¹]**: A user friendly confirmation message after the booking should be communicated to the user’s email using a proper format.
  - Here we have 4 functional & 1 Non functional requirement:
    - **[FRs]**: Book a ticket, Send confirmation message, Message communication to email, set format representation of message
    - **[NFR]**: User friendly Message
Adaptive Requirement Characterization

- Adaptive Requirement Analysis:
  - [ReqAR¹]: A confirmation message for booking is generated as soon the booking is processed, and required to possibly communicate the message to the user eventually on his current device (e.g. PDA/Laptop) by seamlessly observing (monitor) the user’s context (Profile, Location, Device), run-time events and QoS attributes until the message is delivered in a correct format (by scaling it, size, etc) and with personalized representation (e.g. SMS, Email) to his current device i.e. PDA or a different way of notification is applied i.e. exploiting his personal assets (e.g. Contact List)

  [Followed approach from Whittle 08 et al.]

- At Run-time (e.g. cases):
  - Message was delivered; the user’s device battery went down (so the message could not be confirmed as delivered)
  - The software again employs other alternatives at run-time using user’s contact info asset, either sending an email to her secretary or any colleague; or notifying her family/friends
Dimensions of Adaptive Requirements

- **Why** the user wants this; **Why** in this way?
- **What** the system should do?
- Possibly **How** (well) should it do?
- **What** requirements may lead also to answer questions about **Where** and **When** aspects?

![Diagram showing dimensions of adaptive requirements with a cycle and arrows labeled WHY, WHAT, HOW, WHERE, WHEN, and Time. The diagram includes labels for Soft-goal, Hard-goal, Plans, Resource (User Asset), and Time.](image-url)
Capturing Adaptive Requirements

- **Analysis (Steps) Guidelines:**
  
  1. Analyze domain concepts and assumptions
     - **Outcome:** Ontology
  2. Analyze stakeholder’s intensions, dependency and variability in goals
     - **Outcome:** Goal model
  3. Link domain assumptions/concept properties to goals/plans
     - **Outcome:** Link properties
  4. Develop a specification which encompasses the notion of variability
     - **Outcome:** Adaptive Policy / Plan Sepcs. *(A live Artifact)*
Capturing Adaptive Requirements

- **Plan Specification (Example Template):**

  Plan Model(<SendMessage>) to accomplish Goal (BookingConfirmed)

  ```
  begin procedure Plan Model(<SendMessage>)
  do triggerGoals (UserContextIdentified,ItineraryMonitored)
  begin
    for Goal (UserContextIdentified)
    do executePlan <DetectDevice>; //@param: phone type, phone setting
    return; //@result: device
  end;

  for Goal (ItineraryMonitored)
  do executePlan <DetectChanges>; //@param: msg delivery err, conn err
  return; //@result: eventMessage
  end;

  decision = decision on AltPlans(device, eventMessage);
  case decision:
  - Select case:<SendSMS>; //if device = PDA, eventMessage = null
  - Select case:<SendEmail>; //if device = Laptop, eventMessage = null
  - default case: <SendFax>; //if device = null, eventMessage = null

  If not [decision]
  then lookupContact; //@param: cust name, contact info
  alt decision = decision on AltPlans(cust name, contact info);

  case alt decision:
  - Select case: <SendEmail>; //contact info
  - default case: <SendFax> //contact info
  end procedure;
  ```
Conclusions

- Proposed an integrated approach using *Ontologies* and *Goal-oriented modeling* to better support the analysis of variability, a key step towards understanding adaptive requirements for self-adaptive software systems.

- Defined & Characterized Adaptive Requirements

- Presented an Example to justify adaptive requirements along *Why, What, How, When* and *Where* dimensions.
Future Work

- Consolidate our analysis, especially with real scenarios, and **to define a step by step analysis process**.

- Investigating a suitable technique to specify Adaptive requirements to keep requirements “**ALIVE**”

- Demonstrating the role of these requirements artifacts (**goal models plus ontology**) to enable adaptation at runtime.
Thank you!!

We cannot undo things.... So we adapt....

Can we have softwares, which can...?