Self-adaptive Service Based Applications: Challenges in Requirements Engineering

Anna Perini
Center for Information Technology CIT-Irst, FBK
Trento, Italy

Keynote at 16/05/12
Google's Driverless Car & Car2X

- Radar, GPS, a rotating roof sensor
- Video camera
- P2P cars comm.
- Google map
- Traffic management comm.
- Odometer
- Passenger services
- Traffic info
- Green Driving

“Integrating services into car environment” W. Wahlster 2012

http://money.cnn.com/2012/05/10/autos/google-driverless-cars-safety/?source=cnn_bin
Self-Adaptive Service Based App.

- Google's Driverless Car; Car2X
- Travel Domain (e.g. Travel Companion)

Examples of Self-Adaptive Service-Based Applications (SBA) which rest on multiple services, made available over the Internet by different service providers, which are dynamically discovered, selected and composed at run-time
General Features

• By nature SBA are:
  – Distributed systems, they run over the Internet
  – Not under direct control of their developers, services can be used, but they are owned by third parties
  – Live in an open world, external events can create unforeseen situations
  – Dynamic, since services used in a composition can be discovered and selected dynamically at run-time

• With self-adaptivity mechanisms we make them:
  – Context-aware, that is able to assess: availability of needed operational resource (e.g. bandwidth), availability of component services … and the state of real objects they manipulate
  – Able to prevent failure and to enact reconciliation or compensation mechanisms if service qualities are degrading
  – Accessible seamlessly through a variety of devices (e.g. smart phones, laptop), by heterogeneous end-users, with different skills and changing needs and preferences on the go
Self-Adaptivity

• “Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible … This implies that the software has multiple ways of accomplishing its purpose, and has enough knowledge of its construction to make effective changes at runtime.”

Definition reported as DARPA Broad Agency Announcement on Self Adaptive Software (BAA-98-12) [Laddaga 1997], cited in following publications by Robertson & Laddaga

• Realized as Monitor-Adapt internal control loop in SBA
• And more generally in the MAPE-K architecture of autonomic elements proposed in autonomic computing [Kephart ‘03]
Engineering self-adaptive SBA

- Middleware for enabling context aware, self-adaptive SBA (e.g. MUSIC, DiVA European projects)
- A considerable amount of work on technology, methods and standards for engineering self-adaptive SBA has been developed (e.g. ref. S-Cube network results: s-cube-network.eu):
  - Service infrastructure (SOC and grid computing)
  - Service composition and coordination
    - Context-aware automated retrieval and composition of services based on AI techniques (e.g. Al planning [Bertoli El al. 2010])
    - Verification and cross-cutting monitoring & adaptation (e.g. [Kazhamiakin et al. 2010])
  - Business Process Management
  - Methods for Analysis, Design, Development, Quality Assurance of services, Testing …
    - E.g. SOA paradigm; life cycles models, specification languages

That is the machinery for engineering self-adaptive SBA seem well defined and consolidating …
Outline

• Service Based Applications
  – Features
  – Self-adaptivity

• Requirements Engineering
  – Research challenges
  – Review of some recent work

• Conclusion & Open issues
• RE is conventionally defined as a systematic process aiming to
  – understand the problem domain (e.g. stakeholder goals and strategic dependencies; domain assumptions)
  – identify the purpose for the system-to-be
  – build a specification for the system-to-be that is able to satisfy the goals of its users
• It is usually performed at the outset of the whole development process, at Design-Time
• Underlying assumption: “closed world”, i.e. static models are acceptable approximations
• How should we interpret these statements in the case of distributed, open and highly-dynamic systems, as self adaptive SBA?
• Is it needed an RE at Run-Time? … and how should it be conceived?
  – Questions analyzed in the last 4-5 years
  [Cheng et al, Daghstul 2008]; [Di Nitto et al. 2008]; [Sawyer Et al. RE’10]
RE for Self-adaptivity

• Among main challenges:
  – Run-time representation of requirements
    • Imply: “alive requirements” and sort of Run-Time RE (≠ Design-Time RE?)
  – Uncertainty and unpredictable changes
    • i.e. limits of the knowledge available at Design-Time vs. limits of expressiveness of the languages used at DT
  – End-user centrality
    • From user types to personalization at individual level

• Recent work address them … I’ll try to pick up essential features …
RE for Self-adaptivity

- Recent work address self-adaptivity, among them:
  - Tropos4AS [Morandini Et al. 2008-2011]
  - Adaptive Req. (ARML+CARE) [Qureshi Et al. 2009-2012]
  - FLAGS [Pasquale Et al. 2010-2011]
  - Awareness Req. (Zanshin) [Souza Et al. 2010-2012]

- Build on seminal work, including among the others requirements monitoring and context aware systems
  - E.g. from Fickas and Feather 2005 and following work, e.g [Robinson’06], on RE for DAS [Berry’05 ]; on context-aware systems (e.g.[Salifu’07])
• Capture **objectives** of stakeholders as **goals** of the system
• Decompose goals into smaller, more concrete ones (AND/OR decomposition)
• Goals are *operationalized* into a set of specifications (*tasks*) for the system-to-be, which satisfy the requirements (goals), given the domain assumption
• Identify **alternative** ways to achieve goals (variability design)
  – “Goal refinement generates a space of possible specifications and the requirements problem amounts to finding those that satisfy R” [Mylopoulos’12]
• Several GO approaches (e.g. i*, KAOS, …), with additional entities, e.g. Actor; dependencies, etc.
• **Techne** [Jureta Et al. RE10] adds Preferences & priorities; Quality constraints
  – a solution consists of a specification that satisfies all mandatory goals and a maximal consistent subset of preferred ones ... an **optimization** problem
A View on RE for Self-adaptivity

- Requirements Engineering perspectives:
  a) Design-Time
     • An ontology for requirements to represent elicited information; modeling primitives
     • Analysis (e.g. variability, consistency, goal satisfaction)
  b) Run-Time
     • Monitoring requirements
     • Reconfiguring for requirements satisfaction
     • Requirements Change management
  c) Foundations
     • Notion of requirements compliance
     • Requirements problem formulation / underlying framework
Outline

• Service Based Applications
  – Features
  – Self-adaptivity

• Requirements Engineering
  – Research challenges
  – Review of some recent work
    • Tropos4AS
    • Adaptive Req. (ARML+CARE)
    • FLAGS
    • Awareness Req. (Zanshin)

• Conclusion & Open issues
Tropos4AS [Morandini: SEAMS08, ASE08, AAMAS09]

- **Requirements** specifications represented as goal models (Tropos foundation, \{G, S, P, R\});
  - explicit modeling of goal types (achieve, maintain, perform), environment and failure;
  - Relations between environment and conditions for goal creation, achievement, failure
- Variability design supports analysis for adaptation to prevent failure (i.e. expliciting alternative behaviours for goal satisfaction)

---

- **JADEX implementation** of the goal model with all the alternatives
- **Operational semantics** defined by transition rules

---

**Goal states: intuition**
- “Suspend”
- “Active, deliberate”: find applicable subgoals
- “Active, undefined”: subgoal achievement taking place
- “Active, succ./failed”: “provisional” success/failure state.
**ARML+CARE** [Qureshi: SEAMS08, CAISE10, RE10, REFSQ12]

- **Requirements** specifications represented as **goal models** (CORE ontology revisited, Techne foundation, \( \{G, T, Q, D, \text{Context}, R\}; \text{relegate, influence rels}\));
- **Adaptive reqs**: “A confirmation message for booking is generated as soon as the booking has been processed and sent to the user on her current device (e.g. PDA) by seamlessly observing: the user’s context (Profile, Location, Device) and QoS attributes until the message is delivered in a correct format to her current device i.e. PDA, or an alternative notification is performed.”
FLAGS [Pasquale: SEAMS’10, RE’10]

- **Requirements** specifications represented as goal models (KAOS foundation, with LTL formalization);
- **Adaptive Goal**, which specifies adaptation countermeasure that are associated with events that trigger its execution at runtime;
- **Fuzzy Goal**, which coexist with clearly defined goals
- **Goal satisfaction** fuzzyfying temporal operators, i.e. “always”, become “almost always”

---

**Run-time**

Supervision manager that map goals to BPEL processes

The washing machine turns off suddenly

Condition: **wm.state=off**

Trigger: **G1.4.1 violated**

**Objective**: turn on the machine

**Countermeasure**: define a set of actions to achieve the objective

---

**Design-time**

- Achieve [WashingComplete]
- AND
- G1.4
  - G1.4.1 Achieve [PerformWashing]
  - G1.4.2 Achieve [EmptyWashingMachine]

**Op5** [Wash]

**Op6** [Empty WM]

AG 1.4

C4

Add goal (G1.5) Add operation (Turn On WM)

1. Live Goal model, connected to environment monitoring probes

2. Process level
- **Requirements** specifications represented as goal models (Techne foundation, \{G, T, Q, D\});
- **Awareness reqs**: “Goal 'Find a suitable room' should never fail / should have 90% success” (AR4);
- **Parameters** for reconfiguration:
  - OR-refinements / variation points (VP2);
  - Control variables (RfM = Rooms for Meetings);
  - Differential relations: Δ(AR4 / RfM) > 0.

1. Users should always find a suitable room (Awareness)  
   ![Diagram](image)

2. No rooms available! (Aw. Req. failure)
3. Reconfigure: arrange a new meeting room? Look for a room elsewhere (partners, hotels)?
RE for Self-adaptivity – Conclusion

- All considered work address Design-Time RE proposing suitable concepts and analysis technique to design for Run-Time monitoring and adaptation
- DT uncertainty is managed combining variability design (variability in context and system behavior) with monitoring specs.; by relaxing goal satisfaction criteria (e.g. using fuzzy logics, qualitative reasoning, relegate operator)
- RT representation of requirements is provided either as Java ECA rules (e.g. CARE, Zanshin) or as JADEX executable goal-model (Tropos4AS)
- End-user centrality is only partially addressed through modelling user context, but individual end-user involvement is still poorly addressed (CARE)
Outline

• Service Based Applications
  – Features
  – Self-adaptivity

• Requirements Engineering
  – Research challenges
  – Review of some recent work
    • Tropos4AS
    • Adaptive Req. (ARML+CARE)
    • FLAGS
    • Awareness Req. (Zanshin)

• Conclusion & Open issues
Adaptation and Evolution are two simultaneous and connected processes (double lifecycle); [S-Cube Book at s-cube-network.eu]
Self-Adaptation or Self-Evolution?

• **Self-adaptation**
  – Design-Time: Analysing problem variants and expliciting corresponding alternative solutions elements, with evaluation criteria for their selection (Variability design+ relaxed compliance analysis)
  – Run-Time: representation of the whole requirements solution space; monitoring and adapt according to revised notion of requirement compliance (e.g. fuzzy criteria; qualitative reasoning)
  – This seems fine for anticipated changes and in case not-exhaustive scenarios for future changes can be identified at DT (i.e. level 3-uncertainty)

• **Self-Evolution**
  – So far offline evolution, human in the loop
  – If new requirements emerge, the problem space change so both problem and solution spaces co-evolve (addressed as “Fluidity of design” in [Jarke Et al. ‘11], addressed as Dynamic RE Problem in [Qureshi Et al. Caise’11], but needs end-user involvement

Anna Perini, CIT-Irst FBK perini@fbk.eu
Dynamic Requirements Problem

Requirements Problem dynamic formulation [Qureshi Et al. Caise’11], based on a revision of CORE ontology for RE [Jureta Et al. Re’08]

- **K**: Domain Assumption
  - e.g. Seats Available
- **G**: Goals, Soft Goals
  - e.g. Travel for Business, Convenience
- **Q**: Quality Constraint
  - e.g. Booking done in < 5 Screens
- **A**: Attitudes: Optional and Preferred Requirements
  - e.g. Book an Aisle Seat and Flight must not transit through USA (due to Visa Requirements)
- **T**: Tasks (specification to meet Goals under K and Q)
  - e.g. Book Flight, Schedule a Meeting

**Depend on**: Context (C)

- e.g. Location) Airport & Scheduled flight is cancelled today

**Refer to**: Resource (R)

- e.g. Mobile, laptop

i.e. a Candidate Solution to the requirements problem in C exists if:

\[ K_C, T_C, R_C | \sim G_C, Q_C, A_C \]

Available resources R are needed to realize CS in C

CS are compared using preferences and is-optional relations
Dynamic Requirements Problem

• Possible lines for further investigation:
  – Suitability of the framework to investigate (co-) evolution ?
  – End-user (explicit/implicit) on-line feedback to support both dynamic adaptation and evolution ?
  – Pro-active adaptation ?
Acknowledgements

• A special thank to Cu D. Nguyen (CIT, FBK Trento), Vitor Souza (DISI-UNITN), Antonio Bucchiarone and Annapaola Marconi (SOA-CIT,FBK) for enlightening discussions and to the colleagues of the SE group for useful suggestions
THANK YOU

Questions?
References 1/3


[Berry et al. REFSQ 2005] D. M. Berry, B. H. Cheng, and J. Zhang. The four levels of requirements engineering for and in dynamic adaptive systems. In REFSQ05, pages 95–100, 2005


References 2/3

[Wahlster ‘12] “Towards Software-Defined Networked Cars as Social Agents”. The 7th International Conference on Agents and Artificial Intelligence, ICAART 2012 Keynote Lecture


TROPOS4AS [Morandini: SEAMS08, ASE08, AAMAS09]]


ARML+CARE [Qureshi: SEAMS08, CAISE10, RE10, REFSQ12]


FLAGS [Pasquale: SEAMS’10, RE’10]


AWARENESS Req. [Souza: SEAMS 2011, ER 2011]


OTHER PROJECTS/NETWORKS