33rd ACM/SIGAPP Symposium On Applied Computing

Improving Self-adaptive Systems Conceptual Modeling

João Pablo S. da Silva, Miguel Ecar, Marcelo S. Pimenta, Fabio Natanael Kepler, Gilleanes T. A. Guedes, and Carlos Michel Betemps

Presenter: Ana Carolina Tomé Klock

Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil

April 12, 2018

Outline

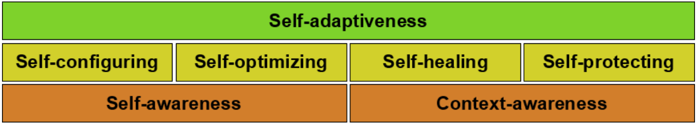
Outline

1. Work Contextualization
2. Modeling Approach
3. Approach Evaluation
4. Final Considerations

Conceptual Background

Self-adaptive Systems (SaSs) are able to adapt or (re)organize their behavior at runtime in response to contextual changes [1, 2].

- They operate under uncertainty conditions [3].
- They have intrinsic properties [4].



Conceptual modeling is the act of creating models that describe problem structures independently of the solution strategy [5].

- Abstract representations of a situation under investigation [6].
- Aid to understand the situation in which a problem occurs [7].
  - Useful for requirements analysis.

Motivation and Objective

- SaSs conceptual modeling deal with requirements uncertainty, contextual changes, and behavior adaptations.
- Model quality is related to its capability in providing the same understanding for stakeholders [8].
- Conceptual models are built by humans, therefore, their quality heavily depends on the humans expertise.

This is not a good software engineering practice!

This work proposes a modeling approach that provides higher-level abstractions for building SaSs conceptual models (metamodel) and procedures for capturing the concepts from SaSs requirements (process).

.inf INSTITUTO DE INFORMÁTICA UFRGS PPGC

Related Works

- There are three main concerns in the founded papers:

  adaptation establishes means to model different adaptation aspects [9, 10, 11, 12, 13, 14];

  context defines metamodels to support context requirements modeling [15, 16, 17];

  uncertainty deals with the uncertainty inherent to SaSs [18, 19].
- These papers support conceptual modeling by providing metamodels related to SaSs.
- A metamodel itself does not specify how to extract abstractions from requirements.

> Besides proposing a metamodel, this approach defines a process to guide the SaSs conceptual modeling.

Outline

Requirements Specification

- The RELAX [20] language was chosen to specify the requirements.
  - Structured natural language sentences.
  - Enriched with operators and uncertainty factors.

Requirement Example:

The SmartCar SHALL plan AS FEW AS POSSIBLE refuelings AFTER trip starts.

ENV: SmartCar; Roadmap.

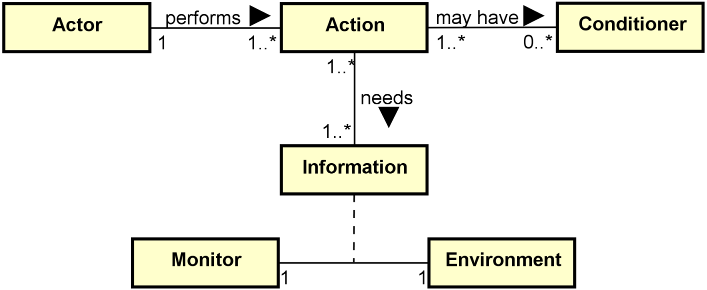
MON: Service Sensor; Consumption Sensor.

REL: Service Sensor monitors Roadmap to provide information about fuel stations.
Consumption Sensor monitors SmartCar to provide information about fuel autonomy.

DEP: Does not apply.

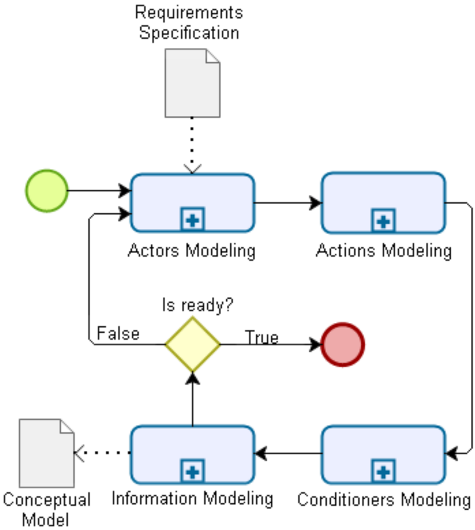
Conceptual Model Metamodel

The metamodel defines higher-level abstractions for creating conceptual models from requirements written in the RELAX language.



Conceptual Modeling Process

The process defines a way to instantiate the metamodel for creating conceptual models from requirements written in the RELAX language.

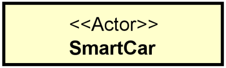


Actors Modeling

Procedures:

1. Read the text written before modal operators.
2. Identify the actors that perform actions.
3. Create a class for each identified actor.
4. Assign to each class the ≪`Actor`≫ stereotype.

Example:

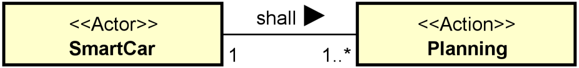


Actions Modeling

Procedures:

1. Read the text written between modal operators and temporal or ordinal operators.
2. Identify the actions related to the requirements behavior.
3. Create a class for each identified action.
4. Assign the ≪Action≫ stereotype to action classes.
5. Create an association between the actors and their actions.

Example:

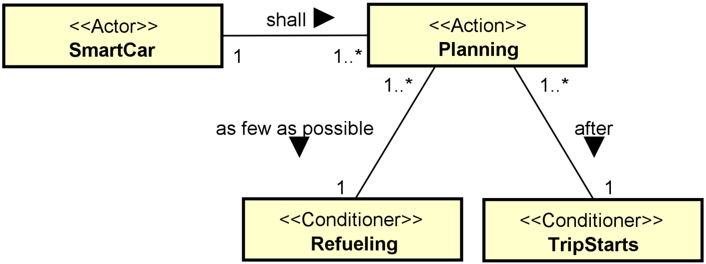


Conditioners Modeling

Procedures:

1. Read the text written after temporal or ordinal operators.
2. Identify the events, counters, states, or timers that condition the actions.
3. Create a class for each identified conditioner.
4. Assign the ≪Conditioner≫ stereotype to conditioners classes.
5. Create an association between the actions and their conditioners.

Example:



Information Modeling I

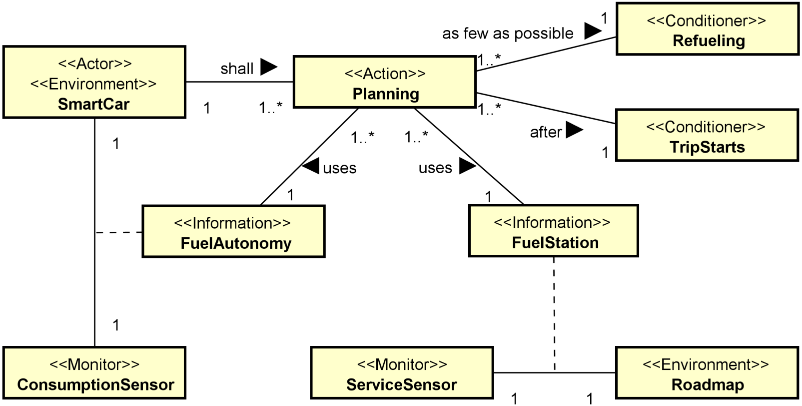
Procedures:

1. Read the text written in uncertainty factors.
2. Identify the environments specified in ENV.
3. Create a class for each identified environment.
4. Assign the ≪`Environment`≫ stereotype to environment classes.
5. Identify the monitors specified in MON.
6. Create a class for each identified monitor.
7. Assign the ≪`Monitor`≫ stereotype to monitor classes.
8. Create an associative class for each relation between MOM and ENV specified in REL.
9. Assign the ≪`Information`≫ stereotype to information associative classes.

Information Modeling II

10. Create an association between the actions and their support information.

Example:g



Outline

Experiment Planning I

- Proposed approach effectiveness evaluation.
- Experiment with software engineering students.
- Conceptual model building from a SaSs requirement.
- Information retrieval metrics to measure effectiveness [21].
- F-scores analysis of two groups:
  - experimental group (proposed approach);
  - control group (ad hoc approach).

Hypothesis:

- $H_0 : \mu_{\textit{F-Score}_{\textit{Experimental}}} = \mu_{\textit{F-Score}_{\textit{Control}}}$
- $H_1 : \mu_{\textit{F-Score}_{\textit{Experimental}}} \neq \mu_{\textit{F-Score}_{\textit{Control}}}$

Variables:

Independent the proposed and the ad hoc modeling approaches.

Dependent the effectiveness.

Experiment Planning II

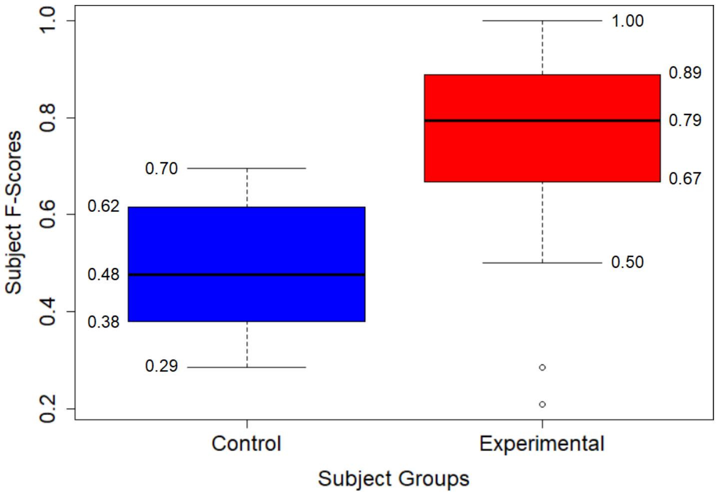
Subjects:

- Software engineering undergraduate students with similar skills.
- Students that had attended software analysis and design classes.
- 36 subjects group by according their maturity in the course.
- Randomly and equally allocated into the groups.

Roadmap:

1. Read about conceptual modeling with UML.
2. Read about requirements specification with RELAX.
3. Read about conceptual modeling approaches.
4. Model of a requirement written in RELAX.
5. Answer the evaluation questionnaire.

Results and Analysis I



Results and Analysis II

- The medians show that the experimental group had a better performance than the control group.
- The experimental group lower quartile is greater than the control group upper quartile.
- Regarding groups spreads, there is no significant difference between them.
- The experimental group has two outliers.

Normality Test (Shapiro-Wilk):

- $W_{\textit{F-Score}} = 0.968 > W_{(0.05,36)} = 0.935$
- $\textit{P-value}_{\textit{F-Score}} = 0.377 > \alpha = 0.05$
- The sample comes from a normal population.

Hypothesis Test (T-test):

- The two-tailed *P-value* is 0.0003 ($\alpha = 0.05$).
- The difference is considered statistically significant.

Outline

Conclusions

- The experiment results:
  - allow to accept the alternative hypothesis.
  - show that the proposed approach is effective.
- These conclusions are restricted by the scope of this experiment.
- The main contributions are:
  - the metamodel that provides higher-level abstractions;
  - the process that defines how to instantiate the metamodel.
- The main limitation is related to generality:
  - the approach currently requires RELAX as the language for writing requirements;
  - the experiment was limited to a single scenario and a sample from a restricted population.
- In future works, the authors intend to address these limitations.

References I

[1] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw.
Engineering Self-Adaptive Systems through Feedback Loops.
In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 48–70. Springer Berlin Heidelberg, Berlin, DE, 2009.

[2] Betty H. C. Cheng, Rogério De Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Di Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, Holger M Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, Hausi a Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle.
Software Engineering for Self-Adaptive Systems: A Research Roadmap.
In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, Berlin, DE, 2009.

[3] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker.
A survey on engineering approaches for self-adaptive systems.
Pervasive and Mobile Computing, 17(Part B):184–206, 2015.

[4] Mazeiar Salehie and Ladan Tahvildari.
Self-adaptive software: Landscape and research challenges.
ACM Transactions on Autonomous and Adaptive Systems, 4(2):1–42, 2009.

[5] Roland Kaschek.
On the evolution of conceptual modeling.
In Lois Delcambre, Roland H. Kaschek, and Heinrich C. Mayr, editors, *The Evolution of Conceptual Modeling*, volume 08181, pages 1–12, Dagstuhl, Germany, 2008. Schloss Dagstuhl.

[6] Bernhard Thalheim.
The Theory of Conceptual Models, the Theory of Conceptual Modeling and Foudation of Conceptual Modeling.
In David W. Embley and Bernhard Thalheim, editors, *Handbook of Conceptual Modeling: Theory, Practice and Research Challenges*, chapter 17, pages 543–577. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

References II

[7] IEEE Computer Society.
Guide to the Software Engineering Body of Knowledge.
IEEE Press, Piscataway, USA, 3 edition, 2014.

[8] Elizabeth Hull, Ken Jackson, and Jeremy Dick.
Requirements Engineering.
Springer, London, UK, 3 edition, 2011.

[9] Markus Luckey, Benjamin Nagel, Christian Gerth, and Gregor Engels.
Adapt Cases: Extending Use Cases for Adaptive Systems.
In *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 30–39, Waikiki, Honolulu, HI, US, 2011. ACM.

[10] Vítor E. Silva Souza, Alexei Lapouchnian, William N. Robinson, and John Mylopoulos.
Awareness Requirements for Adaptive Systems.
In *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 60–69, New York, 2011. ACM.

[11] Raian Ali, Carlos Solis, Inah Omoronyia, Mazeiar Salehie, and Bashar Nuseibeh.
Social Adaptation: When Software Gives Users a Voice.
In *7th International Conference Evaluation of Novel Approaches to Software Engineering*, page 10, Berlin, Heidelberg, 2012. Springer-Verlag Berlin Heidelberg.

[12] R. Gowri, S. Kanmani, and D. Punitha.
Tropos Based Adaptation Framework for Self Adaptive System.
Journal of Theoretical and Applied Information Technology, 63(3):790–799, 2014.

[13] Hyo-Cheol Lee and Seok-Won Lee.
Towards Knowledge-intensive Software Engineering Framework for Self-Adaptive Software.
In *27th International Conference on Software Engineering and Knowledge Engineering*, page 6, Pittsburgh, 2015. SEKE.

References III

[14] Mirko Morandini, Loris Penserini, Anna Perini, Alessandro Marchetto, Loris Penserini, and Alessandro Marchetto.
Engineering requirements for adaptive systems.
Requirements Engineering, 22(1):77–103, 2015.

[15] Paola Inverardi and Marco Mori.
Requirements models at run-time to support consistent system evolutions.
In *2011 2nd International Workshop on Requirements@Run.Time*, pages 1–8, Trento, 2011. IEEE.

[16] Wei Liu and Zaiwen Feng.
Context-based Requirement Modeling for Self-adaptive Service Software.
Journal of Computational Information Systems, 8(24):10131–10140, 2012.

[17] Tetsuo Tamai and Supasit Monpratarnchai.
A Context-Role Based Modeling Framework for Engineering Adaptive Software Systems.
In *2014 21st Asia-Pacific Software Engineering Conference*, pages 103–110, Jeju, 2014. IEEE.

[18] Manzoor Ahmad, Nicolas Belloir, and Jean-Michel Bruel.
Modeling and Verification of Functional and Non-Functional Requirements of Ambient Self-Adaptive Systems.
Journal of Systems and Software, 107:50–70, 2015.

[19] Deshuai Han, Qiliang Yang, Jianchun Xing, Juelong Li, and Hongda Wang.
FAME: A UML-based framework for modeling fuzzy self-adaptive software.
Information and Software Technology, 76:118–134, 2016.

[20] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H. C. Cheng, and Jean-Michel Bruel.
RELAX: a language to address uncertainty in self-adaptive systems requirement.
Requirements Engineering, 15(2):177–196, 2010.

[21] Andrea De Lucia, Carmine Gravino, Rocco Oliveto, and Genoveffa Tortora.
An experimental comparison of ER and UML class diagrams for data modelling.
Empirical Software Engineering, 15(5):455–492, 2010.

Thank you!

Questions?

E-mail:jpsilva@inf.ufrgs.br