IRET: Requirements for Service Platforms

Luciano Baresi
Dip. Elettronica, Informazione e Bioingegneria
Politecnico di Milano
20133 Milano, Italy
Email: luciano.baresi@polimi.it

Gianluca Ripa
CEFRIEL S.Cons.R.L.
20133 Milano, Italy
Email: gianluca.ripa@cefriel.com

Liliana Pasquale
Lero, University of Limerick
Limerick, Ireland
Email: liliana.pasquale@lero.ie

***Abstract*—This paper describes IRENE (Indenica Requirements ElicitatioN mEthod), a methodology to elicit and model the requirements of service platforms, and IRET (IREne Tool), the Eclipse-based modeling framework we developed for IRENE.**

I. Introduction

Service platforms[1] are increasingly becoming the means many companies use to let customers exploit their products. Examples include platforms that provide services for enterprise resource planning, data storage, mobile communication, or travel planning. **Vertical** platforms are specialized to address the needs of particular domains, while **horizontal** platforms provide basic, general-purpose services, which can be both exploited as such, or act as building blocks of more specialized solutions. Given the availability of a multitude of service platforms, new solutions could be easily created by selecting —and maybe combining— the services provided by them. As the existing solutions cannot be replaced by a single holistic one, interoperability is required, and new platforms may become *virtual* platforms that provide a unified and homogenous view over the services provided by different (real) platforms. The engineering of these virtual service platforms is the goal of the INDENICA project.

This paper introduces the key elements of IRENE (Indenica Requirements ElicitatioN mEthod [1]), and the main characteristics of its supporting tool IRET (IREne Tool).

II. IRENE

IRENE is centered on an extended goal-based solution for requirements elicitation. It builds on FLAGS [2], a requirements modeling methodology specifically conceived for adaptive systems, which, in turn, extends the KAOS model [3].

An IRENE model, being based on KAOS, comprises four sub-models. The *goal model* defines the main objectives the application should meet through goals. The *operation model* associates the leaf goals with the operations necessary for their achievement. The *object model* represents the entities and the events on which goals and operations are specified. The *agent responsibility model* assigns goals to agents (e.g., software components, external devices), responsible for their achievement. According to KAOS, goals are formally expressed in terms of LTL (Linear Temporal Logic) expressions, and operations are formalized as pre- and post-conditions that predicate on modeled objects. IRENE allows for more freedom, and goals and operations can be informally expressed as textual annotations. For example, the user may decide to only provide informal descriptions of easy/well-known parts and concentrate on formalizing the newest, most complex, and less familiar ones.

FLAGS extends the KAOS model by distinguishing between *crisp* and *fuzzy* goals. The former goals can only be either fully true or false, while the latter goals can also be partially satisfied. One can quantify the degree —between 0 and 1— to which a goal is satisfied/violated. The user can also add *adaptation goals* to explicitly reason on the adaptation and evolution of modern software systems. Adaptations can be conceived as additional operations performed by the system, without modifying the requirements behind the system, and/or as changes in the requirements model, which means that there are new/different requirements that must be taken into account.

IRENE complements FLAGS by supporting *variability*. One can concentrate on: (a) some particular characteristics of a goal: for example, different levels of qualities (soft goals) or variations of intended functionality; (b) differences in the implementation of an operation; (c) alternative values for some parameters of an entity; (d) different responsibilities of an agent; (e) diverse behaviors of an agent; (f) special-purpose satisfaction of fuzzy goals; and (g) alternative specifications of the adaptation goals.

Note that variability could also be expressed by explicitly adding alternatives in the goal model, but this would increase the complexity of design models with many alternatives and dependences. Moreover, these models would be tangled and would hide the dependencies among the different alternatives. IRENE provides a simple, but effective, means to allow the user state the elements in the model in the way s/he prefers, thus making possible dependencies become explicit. This information is provided mainly as comments, but the user can also exploit the INDENICA Variability Modeling Language, which is not discussed here.

IRENE also supports the notion of *family* of related models, to let the user reason on interdependent applications, and provides means to integrate them into a single coherent model. IRENE supports both independent and dependent *views*. The

[1]Our idea of service platform is wider than the idea of platform-as-a-service. A service platform is a set of related services —along with some common guarantees— offered by the same provider.

former views do not share elements, and could be used to unify different models, that is, to merge the requirements that come from different applications that should be supported by the same platform. The latter views share elements, and could be used to structure a single model into meaningful and usable sub-models, where the root of a view is the leaf of another one.

III. IRET

IRET[2] is a graphical editor for specifying FLAGS/IRENE models implemented as an Eclipse plug-in. Figure 1 presents the main screen of the tool[3].

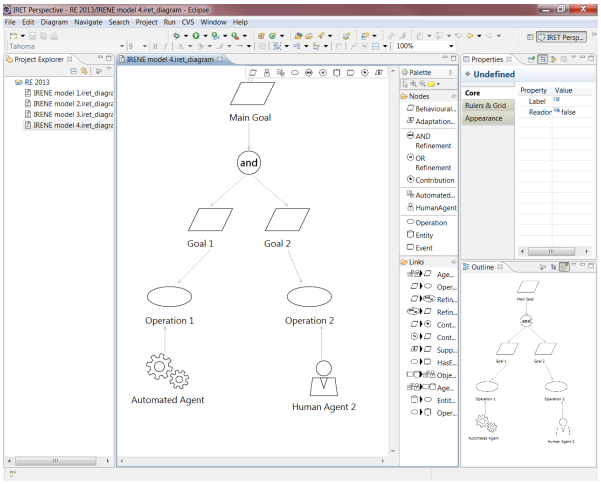

Fig. 1. IRET main screen.

IRET supports all the "conventional" editing activities on goal models through dedicated palettes and panels. It also provides peculiar features to specify complex models. Given the hierarchical nature of goal models, the user can easily identify suitable slices and deal with them independently. Usually, the high-level goals define the first view on the system; then the low-level goals can be developed in other, more specific views. The user can also follow different approaches to elicit the requirements into a complete and coherent specification. In some cases, the elicitation can easily be organized around a single model, or a hierarchy of views. In other cases, the requirements for a complete platform may come after modeling (some of) the applications it should support. This is why IRET offers means to integrate the different models.

IRET does not provide any fully-automated integration solution. The final decision about the actual similarity between two goals, entities, or operations is always up to the user and the tool is only supposed to help highlight the similarities. To keep things simple and usable, the similarity is only evaluated on the names of the different elements of the model and on the priorities of goals. We decided to reuse a well-known information-retrieval algorithm [4] to state the similarity between words. The user is in charge of setting the precision of the analysis, that is, the accepted *degree* of similarity. The extreme that any name is similar to any other would be useless, but also the strict equality between the two strings could be too tough. The integration of models always starts from a reference model, compares the elements in the to-be-merged model against those in the reference one, and merges similar elements. If entities and actors of the second model are not considered to be similar, they are simply added to the reference one. Goals are always added through predefined patterns [1] to keep the correct (hierarchical) organization of the reference specification.

[2]A short video illustrating the tool can be found at the following address: http://www.cefriel.com/projects/idenica.

[3]IRET can be installed through the Eclipse Update Site from the following link: http://indenica.dei.polimi.it/iret/update-site/.

IV. Related work

The configuration of adaptable/context-dependent systems is becoming a critical factor. For example, Ali et al. [5] propose the integration of the variability of requirements, expressed via goal models, and the variability of context. Variability is included in the requirements model and cannot be separated into different views.

Multiple viewpoints [6] are aimed to model and manage the inconsistencies of system specifications developed from multiple perspectives. Despite the similarity with our work, the objective of IRET is not to solve inconsistencies but only to offer a means to merge similar requirements. Sabetzadeh and Easterbrook [7] provide an algebraic framework to merge requirements viewpoints. This approach is mainly structural as it treats models as graphical artifacts while largely ignoring their semantics. Nejati et al. [8] focus on merging variant feature specifications described as Statechart models. The merging combines structural and semantic information in the models and ensures that their behavioral properties are preserved. IRET is limited to provide graphical, automated support to modeling and merging different requirements models and does not aim at the best algorithm for the actual merge. Instead, we plan to integrate the merge algorithms described above in the next releases of our tool.

References

[1] Indenica, "Requirements engineering framework, language and tools for service platforms," http://www.indenica.eu, Indenica project, Tech. Rep., October 2012.

[2] L. Baresi, L. Pasquale, and P. Spoletini, "Fuzzy Goals for Requirements-Driven Adaptation," in *Proc. of the 18th Int. Requirements Engineering Conf.*, 2010, pp. 125–134.

[3] A. van Lamsweerde, *Requirements Engineering, From System Goals to UML Models to Software Specifications*. John Wiley, 2009.

[4] N. Seco, T. Veale, and J. Hayes, "An Intrinsic Information Content Metric for Semantic Similarity in WordNet," in *Proc. European Conf. on Artificial Intelligence*, 2004, pp. 1089–1090.

[5] R. Ali, F. Dalpiaz, and P. Giorgini, "A Goal-Based Framework for Contextual Requirements Modeling and Analysis," *Requir. Eng.*, vol. 15, no. 4, 2010.

[6] A. Finkelstein, D. M. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh, "Inconsistency Handling in Multiperspective Specifications," *IEEE Trans. Software Eng.*, vol. 20, no. 8, 1994.

[7] M. Sabetzadeh and S. M. Easterbrook, "View Merging in the Presence of Incompleteness and Inconsistency," *Requir. Eng.*, vol. 11, no. 3, 2006.

[8] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave, "Matching and Merging of Variant Feature Specifications," *IEEE Trans. Software Eng.*, vol. 38, no. 6, 2012.