Navigational Web-interfaces from Formal Tropos Specification

Komminist Weldemariam

Center for Information Technology, FBK-Irst,
via Sommarive 18, Trento 38100, Italy
Email: sisai@fbk.eu

Abstract. This paper presents a method of building executable and interactive application interface prototypes from requirements. The specification of the requirements uses i* and Formal Tropos languages.

1 Introduction

A methodology for automatically generating prototypes from requirements that are specified using Formal Tropos (FT) [1] language is presented. FT follows the agent-oriented requirements modeling concepts. It allows to specify the structural and behavioral aspects of a system in the form of actors, goals, tasks, resources, softgoals and dependencies. The specification is then iteratively verified and refined using NuSMV [2] tool, for the consistency and completeness checking. The verified requirements are then passed to our tool for generating the actual prototypes, which can easily be embedded in web application.

2 Prototype Generation

We summarize our prototype generation framework in three steps (detail can be found in [3]). First we model and validate the business logic. During which we identify, model, and specify requirements. More specifically, we represent the actors of the system, their goals, and dependencies using i* [4] or Tropos [5]. We derive FT specification for the corresponding visual models. The FT specification is then mapped into an intermediate language using T-Tool [6] for formal analysis. T-Tool calls the NuSMV [2] engine for formal validation. Secondly, we derive relational database structures from FT specification. During which we capture structural and behavioral properties of the system from validated FT specification and map them into relational database structures with stored procedures and functions. Finally we integrate the relational structures and application service interfaces. During which we generate skeletons of application service interfaces from verified requirements so that the stakeholders can directly interact with the prototype.

Our prototype generator schema produces an architecture that comprises of application navigational interfaces, stored programs and functions, and application database models. The model part of the architecture resembles the model

view control (MVC) architecture. It contains the structural aspect of the prototype in the form of relational structure. The stored procedures and functions contain the behavioral aspects of the prototype in the form of behavioral constraints managed through Java Server Pages (JSP) and relational procedures. The application interface is organized as a set of web-based interfaces with navigational features that capture user and system interactions. The user interacts with the application interfaces. The interactions are captured and stored into the database model. The stored procedure observes the runtime state of the database model and controls the view accordingly.

3 Capturing Application Data Models

Information related to the structural and behavioral aspects of the system are derived from the verified FT specification. We organized such information in the form of relational models.

Related to the structural aspect we extracted *actor list*, *goal list*, *goal hierarchy*, and *dependancy list*. The *actor list* is a one to one mapping of actors to their respective root goals. An actor is associated with exactly one root goal, which is formalized in outer layer of the FT specification through a goal or task specification. The *goal list* is a representation of a goal decomposition for every actor. The decomposition is done from the point of view of the actor who committed for its fulfillment. The *goal hierarchy* represents the hierarchy of a goal in the form of a collection of $< supergoal, subgoal >$ tuples. Finally we capture the dependencies between two actors in the *dependancy list*, where the depender's goal depends on dependee actor's local activities.

Behavioral aspects of the objects in the model and dependencies among them are essential for rapid application service prototyping. They are annotated in the inner layer of FT specification to capture the circumstances on the goal creation and fulfillment as constraints. Further, they allow to capture pre- and post- conditions on tasks and sub-tasks creation and fulfillment. We captured two relational tables that model both constraints, namely the *goal creation* and *goal fulfillment* relational structures. The *goal creation* captures the creation constraints for each goal. It is generated from the inner layer of FT specification through an analysis of goal creation properties. Whereas, the *goal fulfillment* captures the fulfillment constraints for each goal. Like the goal creation relation, the goal fulfillment is generated with the same analysis strategy. However, unlike that it uses fulfillment constraints.

4 Integrating Relational Models and Interfaces

The structures mentioned in the previous section are then further analyzed in the second pass to generate service interfaces for application prototyping. The goals which are to be achieved are made unavailable from the service interface once they are fulfilled. Application service interfaces may be easily generated from relational model through the following strategy.

- *Top Level Interface.* An interface is displayed for every root goal for every actor. They are organized by actors role. If the top level interface is navigated, the root goal corresponding to the interface is said to be created in the system. The pre-condition for navigation must attain true value for created flag of the goal which is being navigated.
- *Creation of subgoal interfaces.* A subgoal interface is displayed as soon as creational constraints of the subgoal are satisfied and the super goal is being analyzed. Subgoal instance is created once the super goal interface is navigated.
- *Goal status the display.* Upon navigating through a specific goal subgoal are created, and based on it's fulfillment conditions, its completion status is also displayed.

5 Conclusion

A strategy for modeling and generation of application service interfaces from agent based specifications is discussed. We address interactions between product users and service interfaces, and interdependencies among services. Application interfaces are generated by applying *actor as roles* interpretation. Actor's role equips the actor to interact with the system through an interface corresponding to the responsibilities of the role. In order to find responsibilities, a view of *goals as responsibilities* is taken. A service interface per actor role is generated. A service interface is a collection of goals, or responsibilities of the given actor role. Goal dependencies result in service dependencies. The runtime interface interactions with the relational database model is supported by the MySQL stored procedures and functions.

References

1. M. Pistore A. Fuxman, R. Kazhamiakin and M. Roveri. Formal Tropos: Language and Semantics. Technical Report 4, University of Trento, November 2003.
2. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark, July 2002. Springer.
3. Komminist Weldemariam. An Agent Oriented Approach for Rapid Application Prototyping. Master's thesis, Indian Institute of Technology, Bombay, Powai, Mumbai-400076, India, Auguest 2006.
4. Eric Siu-Kwong Yu. Modelling Strategic Relationships for Process Reengineering. In *PhD Thesis, Dept. of Computer Science, University of Toronto*, Toronto, Canada, 1995.
5. Fausto Giunchiglia, John Mylopoulos, and Anna Perini. The Tropos Software Development Methodology: processes, models and diagrams. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 35–36, New York, NY, USA, 2002. ACM.
6. M. Pistore R. Kazhamiakin and M. Roveri. T-Tool Tutorial. Technical report, University of Trento, 2003.