i* Diagnoses: A Quality Process for Building i* Models

Antonio de Padua A. Oliveira [1, 2], Julio Cesar S. P. Leite [2],
Luiz Marcio Cysneiros[3], Carlos Jose P. Lucena[2]

[1] Universidade do Estado do Rio de Janeiro – UERJ
Rua São Francisco Xavier, 524 - 6 andar - Maracanã - Rio de Janeiro, Brazil

[2] Pontificia Universidade Catolica do Rio de Janeiro – PUC-Rio
Departamento de Informatica, Rua Marques de Sao Vicente 225 – Rio de Janeiro, Brazil

[3]York University – School of Information Technology
4700 Keele St. – Toronto, Canada

{padua, julio, lucena}@inf.puc-rio.br - cysneiro@yorku.ca

Abstract. Modeling with i* is not a trivial task. Our work describes i* Diagnoses Framework, a quality oriented process to analyze i* models. Our process is similar to some of the reading techniques of inspection methods and bears some similarity with the inquiry based requirement analysis approach. Our process focuses on defect prevention considering both the efficiency and effectiveness of Multi-Agent System development.

Keywords: early requirements, MAS, software development.

1 Introduction

There seems to be a consensus that dealing with intentionality at early stages of software projects is a reasonable idea. i* Framework [9] models have been receiving greater attention from several researchers [1], [2] as an infrastructure to deal with intentionality. Although i* has been cited and used in different research projects, most of their users agree that i* models are complex artifacts [7]. Although comprised of few elements, the semantics involved in using them can make i* models prone to errors [7].

The majority of the work has been focused on i* modeling and how to use this information on later stages of software production. Our goal is to focus on analyzing i* models proposing a quality assurance process to produce better i* models. Process quality focuses on defect prevention rather than looking for defects on test phase. We propose an analysis technique to enhance the quality of i* models.

We illustrate our proposal using "The Expert Committee System" (EC System) exemplar [3], a system to support the organization of a conference program.

2 The i* Canonical Structures

Figure 1 (right) shows the basic structure of an SRconstruct, which is formed by a goal (the goal's name is the SRconstruct's name) (as being the end) and at least by one task (as being the means to achieve the end). Therefore, all components (and subcomponents) needed by tasks (subtasks, resources, softgoals, and goals) should appear in the structure. Despite the fact that the goal is only one part of the

SRconstruct, we identify each SRconstruct by the name of the goal that it fulfills. That is because there is only one goal (as being the END) in each SRconstruct.

Figure 1 (left) shows that one actor (CHAIR) and another actor (REVIEWER) can have multiple dependencies in each SDsituation Situations of dependency that occur in the organizational environment and the central idea of SDsituations is: "each dependency link (goal, softgoal, task or resource) that involves actors is not isolated"; it is part of one well defined situation of collaboration called one "strategic dependency situation" or one SDsituation [6].

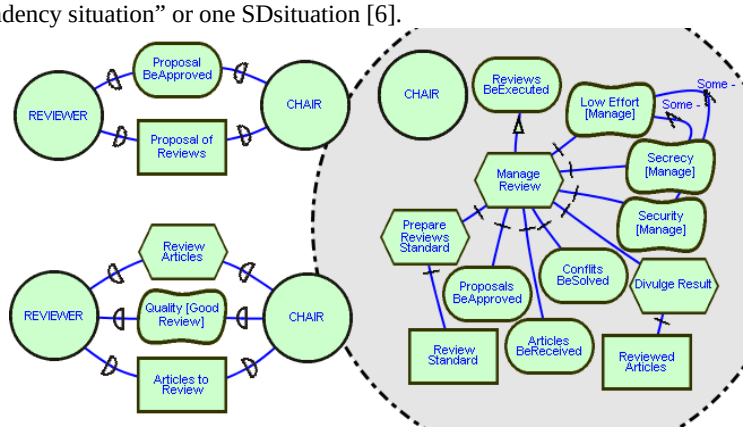

Figure 1 – Examples: left, two SDsituations, and right, one SRconstruct.

3 i* Diagnoses Strategy

The i* Diagnoses examine each canonical structure (SDsituations and SRconstructs) of a given model in order to bring questions that challenge the model consistency and completeness. The main idea is to focus on parts of an i* model and from these parts conduct an inquiry into the given construct.

Our process comprises 3 main sub-processes: IDENTIFY CONSTRUCTS, APPLY (INQUIRY) FRAMEWORK, and INTEGRATE QUESTIONS. The strategy is applied both to the Strategic Dependency Diagrams and to the Strategic Rationale Diagrams of i*.

The activity IDENTIFY CONSTRUCTS consists of breaking down the i* diagrams into constructs (SDsituations and SRconstructs). The activity APPLY (INQUIRY) FRAMEWORK consists of applying the inquiry framework to each construct. The activity INTEGRATE QUESTIONS consists of merging the questions to analyze them. The aim of each diagnose framework is to turn coupled SDsituations and SRconstructs inside out looking for problems, faults, deficiencies and potential improvements. "Diagnoses are important to deeply understand the problem before looking for the solution."

4 SDsituation & SRconstruct Diagnoses

Given the basic structures of SDsituations and SRconstructs, general questions are proposed to each element. In real cases these "hot-spots" or "place-holders" would be replaced in Templates 3.1 and 3.2 by the actual names used in the model.

Template 3.1 - **SDSITUATION**

SDSITUATION: "**SDsituation's name**"

I. INTER QUESTIONS
   1. *Who* else could collaborate with "***depender***" to have "***SDsituation goal's name***"? *How much can he collaborate?*
   2. *Why* does "***dependee***" collaborate with "***depender***" to have "***SDsituation goal's name***"?
   3. *What* SDsituations come before "***SDsituation's name***"?
   4. *What kind* of problems with previous SDsituations can be identified to have "***SDsituation goal's name***"?
   5. *What if* "***dependee***" cannot collaborate on the "***SDsituation's name***"?

II. INTRA QUESTIONS
   6. *What* are the problems inside "***SDsituation's name***"? What kinds of problems (accuracy, deficiencies, ambiguities, or omissions) are identified as having "***SDsituation goal's name***"?
   7. *What* details are needed by "***depender***"?
      a) Case: resource dependency - *What* are "***resource's name***" problems of availability? (Time, accuracy) *When? How? How much?*
      b) Case: goal dependency – *What* are "***goal's name***" problems to be achieved by "***dependee***"? (Time, ability) *When? How? How much?*
      c) Case: softgoal dependency – *What* are "***softgoal's name***" problems to be satisficed by "dependee"? (Capability) Is there "***softgoal's name***" at the end of "***SDsituation's name***"? *Why*? *Who* is demanding the softgoal?
      d) Case: task dependency - Has "***dependee***" received the directions of *how to* perform "***task's name***"? Can the "***dependee***" still perform it? (Time, ability)
   8. *What* dependency has the main duty of having "***SDsituation goal's name***"? Why?

Template 3.2 - **SRCONSTRUCT**

SRCONSTRUCT: "SRconstruct goal's name"

I. INTER QUESTIONS
   1. *Who* else has the "***endGoal's name***" achieved?
   2. *What* are the alternatives that the "***endGoal's name***" has achieved? *Why*?
   3. *What* are the elements of dependency of dependees?
   4. *What* kinds of problems (accuracy, deficiencies, ambiguities, or omissions) can be foreseen? How much? *What if* resources are unavailable? *Who* is to blame? *How to* avoid such problems?
   5. *What if* "***endGoal´s name***" is shared with another actor?
   6. What other construct depends on this goal? Why? How much?

II. INTRA QUESTIONS (for each meanTask)
   7. What are the problems with the task "***meanTask's name***"? *Why*?
   8. For the task "***meanTask's name***" *what* are the components needed to achieve "***endGoal's name***"?
      a) Case: resource – *What* are "***resource's name***" problems of availability? (Time, accuracy). *When? How?*
      b) Case: subGoal – *What* are "***subGoal's name***" problems to be achieved by "dependee/actor"? (Time, ability). *When? How?*
      c) Case: softgoal – *What* are "***softgoal's name***" problems to be satisficed by "***dependee/actor***"? (Capability) Is there "***softgoal's name***" at the end of

"***SRconstruct goal's name***"? *Why*? *What* are the contribution links to and from this ***"sofgoal´s name"***?

d) Case: subTask - Can "***dependee/actor***" perform "***task's name***"? (Time, ability)

9. Is there any softgoal details omitted, not fully operational or without operationalization? What kind? Why? How? How much?
10. Is there any resource missing? What kind? What if a resource is not available?

5 Conclusion

The first benefit of using i* canonic structures (SDsituations and SRconstructs) is managing complexity. Using SDsituations and SRconstructs, i* models can be divided into small pieces avoiding common misuses that appear in i* models [4] and also improving the stakeholders' understanding.

Our strategy provides a verification based analysis for i* models so as to assure better quality models overall. The verification analysis is performed on composing the constructs with well known general questions, the 5w2h framework [5] and with the ideas of Potts, Takahashi and Anton [8].

According to Moody [10], although software quality proposals have been concentrated at the end of the process, empirical works demonstrate that the majority of defects occur during the requirements phase.

We plan to continue the work in this direction as we will frame our diagnoses approach as a reading strategy for the inspection of i* models. By performing more analysis using the proposed i* diagnoses, we hope to improve the quality of the questions as they are today. We also foresee a possible automation, by generating the set of questions, given a set of i* models. Moreover, we plan to evaluate how this work may scale up to larger models.

References

1. Castro, J.; Kolp, M.; Mylopoulos, J. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project." In: The 13th international conference on advanced information systems engineering, Oxford: Elsevier Science Ltd, v.27, n.6. p. 365-389 - 2002.
2. Cysneiros, L.M. and Yu, E.; Requirements Engineering for Large-Scale Multi-Agent Systems Book chapter in Software Engineering for Large-Scale Multi-Agent Systems – Research Issues and Practical Applications. A. Garcia, C. Lucena, F. Zambonelli, A. Omicini and J. Castro (eds.) LNCS 2603, Springer Verlag. 2003. (Revised and extended version of [SELMAS02]).
3. Deloach, S. et al.; Multiagent Systems Engineering. International. In: Journal of Software Engineering and Knowledge Engineering, 11(3): 231—258 - 2001.
4. Estrada, H; Martínez, A; Pastor, O; Mylopoulos, J.; An Experimental Evaluation of the i* Framework in a Model-based Software Generation Environment; E. Dubois, K. Pohl (Eds.); CAISE 2006, LNCS 4001, pp.513-527, 2006. Springer-Verlag, Berlin Heidelberg, ISSN: 0302-9743; ISBN: 3-540-34652-X, 978-3-540-34652-4.
5. Leite, J.C.S.P, Yu Y.; Liu Y.; Yu E.S.K., Mylopoulos, J.: "Quality-Based Software Reuse" CAiSE-05, LNCS 3520, pp.535-550, 2005, Springer-Verlag .
6. Oliveira, A. Padua A.; Cysneiros, L. M.; "Defining Strategic Dependency Situations in Requirements Elicitation" The IX Workshop on Requirements Engineering; Rio de Janeiro, Brazil - July/2006.
7. Pastor, Oscar; Estrada, Hugo; Martínez, Alicia; The Strengths and Weaknesses of the i* Framework: an experimental evaluation i*, its Applications, Variations and Extensions. Eric Yu et al. (eds.) MIT Press (accepted for publication in 2006)
8. Potts, Colin; Takahashi, Kenji; Antón, Annie I.; "Inquiry-Based Requirements Analysis", IEEE Software, Volume 11, Issue 2 (March 1994), Pages: 21 - 32
9. Yu, E. Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada - 1995.
10. Moody, D.L. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions, Data & Knowledge Engineering, 55 (2005) 243-276.