A Review of Requirement Aware Systems in Dynamically Adaptive Systems

Aniket Lawande
North Carolina State University
Raleigh, NC
aglawand@ncsu.edu

Anisha Shetty
North Carolina State University
Raleigh, NC
aashetty@ncsu.edu

Sushil Ganesh
North Carolina State University
Raleigh, NC
sganesh4@ncsu.edu

1.1.1 ABSTRACT

Papers prior to the work published by Welsh et al. proposed the idea of requirements-aware systems that could introspect about the extent to which their goals were being satisfied at runtime. When combined with requirements monitoring and self-adaptive capabilities, requirements awareness should help optimize goal satisfaction even in the presence of changing run-time context. In this paper we present an overview of the progress made in the field of requirement analysis by presenting the research conducted by various researchers in the years 2008 through 2013. We have documented the key features introduced in each paper which gives the reader a broad overview on the subject.

KEYWORDS

Reflective middleware, Adaptation Infrastructure, Requirements monitoring, Self-adaptive systems, dynamically adaptive system, Dynamic Decision Networks

1. INTRODUCTION

Requirements aware systems attempt to build resiliency, flexibility, robustness and reliability into software applications by modelling requirements into the design and implementation of systems. The essence of Requirements Aware systems is that requirements should not just be design time realizations but tied in with the static and dynamic representations of the system. Traditionally systems implemented these mechanisms using rule-based decisions, where rules are loose representations of business rules or requirements.

The authors of the following papers present new ways in representing resilient systems. They involve building formal representations of the requirements into the system, identifying factors that affect these requirements, the consequences of changes to requirements and the decision to make such a change in the presence of influencing factors.

Modelling decisions is not quite a simplistic process as modelling requirements. There are several factors such as tradeoffs, side-effects, dependencies, costs, priorities and timing that make up a decision to change system focus for a requirement or a set of requirements. Formal specification of these decisions is also needed to effectively build Requirements Aware systems.

2. BACKGROUND AND CONCEPTS

2.1.1 DAS (Dynamically Adaptive Systems)

Self-adaptation is emerging as an increasingly important capability for many applications, particularly those deployed in dynamically changing environments, such as ecosystem monitoring and disaster management. As applications continue to grow in size, complexity, and heterogeneity, it becomes increasingly necessary for computing-based systems to dynamically self-adapt to changing environmental conditions. We call these systems dynamically adaptive systems (DASs).

2.1.2 Requirement Aware Systems

One overarching challenge in developing DASs, therefore, is how to handle the inherent uncertainty posed by the respective application domains. To date, several notable goal-oriented approaches have been proposed for modeling DAS requirements. A DAS has three main types of RE concerns: what are the conditions to monitor for adaptation, what adaptations are needed to achieve a desired new behavior, and what decision-making procedure should be used to associate the monitored conditions to the appropriate

adaptations. Self-repair refers to the ability of such systems to restore fulfillment of their requirements by relying on monitoring, reasoning, and diagnosing on the current state of individual requirements.

3. SAMPLING PROCEDURE

We started with the seed paper i.e. Towards Requirements Aware Systems: Run-time Resolution of Design-time Assumptions [1]. This paper served as an introduction to the concept of self-adaptive systems and runtime requirement analysis. After summarizing this paper, we selected the papers that cited our seed paper. We selected the most popular ones and summarized the past papers which laid out a background for the seed paper.
As the next step, we selected papers that were written post 2011. This gave us an insight into the advancements made in the field of the study and conceptualization of self-adaptive systems.

The following sections present a chronological overview of the papers that were reviewed and studied through the semester. The keywords and key features of each paper have been documented and explained in the proceeding sections.

4. TRACKING HISTORICAL CONTEXT (Papers published before 2011)

The below papers provide an overview of the research that was published prior to the seed paper by Welsh et al. For each of the below papers, we have described the keywords and listed the key features of each paper.

4.1 Goal-based modeling of dynamically adaptive system requirements

4.1.1 Keywords

Dynamically Adaptive Systems:
Systems capable of self-adaptation at run-time so that the target goal of the system is achieved in spite of the dynamic change in requirements at execution time.

MDD:
MDD stands for Model driven development which refers to the methodology of using models (artifacts of the proposed solution) to develop the software.

LoREM:
LoREM refers to Levels of Requirements Engineering for Modeling described by the authors in such a way that provides separation of concerns for the developers of a dynamically adaptive system.

Adaptation Infrastructure:
It is set of mechanisms that enable adaptation in order for the system to react to the changes in the requirements to occur.

4.1.1 Feature Extraction

Related work:
Fickas et al. in their paper [10] spoke about how requirements monitoring could help aid the maintenance of code that runs in dynamic environments. In a subsequent paper [17], they discuss an architecture and a development process for monitoring system requirements at runtime to reconcile the requirements and the system's behavior. Yijun Yu et al in their work [18] speak about how Aspect Oriented Development needs to address the earlier phases of SE through goal oriented requirement analysis. In later publications [19] [20] they talk about how this could result in the design of software that supports high variability.

Future work:
The authors of the paper point to the limitations of the current system and suggest that more work needs to be done to evaluate if DASs could perform RE at runtime. Also new techniques to help system developers identify and generate models of candidate target systems that handle potentially adverse conditions in a robust manner need to be developed. Further they suggest human-generated behavioral models need to be incorporated into the current research.
Sampling procedures
The current hypothesis is studied by implementing it for a flood warning system called GridStix. A detailed explanation of the technology-driven process for performing RE is presented.

Patterns
Using RE process models Application-Driven RE Process or Technology-Driven RE Process offers three key benefits.

1. Separation of concerns for the developers
2. Produces models that can be used to guide the design of a DAS
3. This approach provides more flexibility in defining process models for performing RE for a DAS.

4.2 Genie: Supporting the Model-Driven Development of Reflective, Component-Based Adaptive Systems [21]

4.2.1 Keywords

Variability Management:
An approach to structure, implement and document software variability with repeatability.

Domain Specific Language (DSL)
Computer language specialized for a particular domain.

Reflective middleware
Systems that dynamically self-adapt to changing domains and environment conditions.

Dynamic Variability
Variations in context and structural and environmental conditions at runtime.

4.2.2 Feature Extraction

Commentary:
The authors mention the two methods used to test the genie tool: The first one being a sensor network used to predict flooding of a river valley in North-West England. This case study is in the context of mobile computing environment applications which need to dynamically discover services in highly heterogeneous environments.

Related Work:
Coulson et al. speak about the benefits of a generic yet tailorable approach to component-based systems-building that offers a uniform programming model that is applicable in a wide range of systems-oriented target domains and deployment environments[22]. Coulson also co-authored papers that talks about the future of grid computing, the "divergent grid"[23] and how the need for a grid that could handle heterogeneous nodes and networking components has sprung up in the recent years. A useful application of divergent grid was the flood prediction system using embedded chips and wireless network components [24].

Future Work:
The authors of the paper wish to make the genie tool reconfigurable at runtime. The authors also wish to enhance Genie with validation capabilities so it can return a list of conflicts that need to be resolved.

4.3 A goal model for adaptive complex systems. [24]

4.3.1 Keywords

Goal classes:
Goals that are specified by the system designer to model the goal interactions within the organization.

Goal instances:
The runtime instantiation of a goal class with specific parameters.

Goal Specification tree:
A GMoDS goal specification tree (GSpec) specifies how the goal classes are related to one another.

Goal Triggers:
A set of relations within the tree structure to specify how runtime goals may interact.

4.3.2 Feature Extraction

Future Work:
The authors propose to extend the presented work by taking into account soft goals, goal preferences and goal metrics into their GMoDS goal model.

Informative Visualizations:
The authors have illustrated their concepts of goal specification, instantiation and realization through visualizations like graphs and trees. These media are powerful instructors because relationships between entities like depends, precedes and consists of, are best explained visually.

Related Work:
i* [27] framework is a modeling language suitable for an early phase of system modeling in order to understand the problem domain. i* modeling

language allows to model both as-is and to-be situations [28]. The name i* refers to the notion of distributed intentionality which underlines the framework. KAOS [29], is a goal-oriented software requirements capturing approach in requirements engineering. It is a specific Goal modeling method. It allows for requirements to be calculated from goal diagrams. In PRACTIONIST, a framework for developing agent systems according to the Belief-Desire-Intention (BDI) model, goals play a central role [30]

Motivational Statements:
The purpose of the paper, which the authors highlight in the beginning of it, is to allow a designer to specify goals during requirements and then use those same goals throughout system development and at runtime. Through the descriptions and walkthroughs that the authors have presented, they propose to have developed a model (GMoDS) for ensuring design time goals are translatable to runtime properties of the system.

4.4 RELAX: a language to address uncertainty in self-adaptive systems requirement.

4.4.1 Keywords
Non-invariant requirements:
Requirements that cannot be compromised or traded off for a variant requirement.

Uncertainty factors:
The environment and monitored properties and their relations.

Fuzzy set:
A set whose members possess degrees of membership and not necessarily boolean.

4.4.2 Feature Extraction
New Results:
The authors of the paper have proposed and described a new language for describing requirements for a self-adaptive system. This new requirement specification language addresses the problem of describing requirements in a way to account for changing environment and runtime factors.

Tutorial Materials:
The authors include an in-depth walkthrough through a sample application problem domain. They instruct the readers in the procedure of applying the new RELAX language in specifying requirements for the system and accounting for the various uncertainties involved. By following this walk-through they propose, the RELAX language can be applied to other problem scenarios as well.

Related Work:
Cheng et al provided a roadmap for the designing of self-adaptive software systems [24]. Their work categorizes self-adaptation into the four stages of: dimensions, requirements, engineering and assurances. Lapouchnian et al [25] studied the use of requirements based goal models as the foundation of the adaptive software engineering process. The importance of having appropriate software engineering methodologies in place to aid the design and modeling of software that adapts to environmental changes at runtime has been stressed upon [26]

Future Work:
The authors propose to investigate work in apping RELAX requirements to implementation using adaptive infrastructure and frameworks. They also propose to investigate requirements analysis techniques for RELAX requirements to gauge how effective and consistent RELAX requirements are for a system.

5. Requirements Aware Systems in ASE

5.1 Towards Requirements Aware Systems: Run-time Resolution of Design-time Assumptions [1]

5.1.1 Keywords
Requirement awareness:
Requirement awareness is the ability of a system to optimize the strategy used for goal satisfaction in presence of a change in context.

Self-adaptive systems:
Self-adaptive systems are systems that are capable of improving their ability to react to a scenario to produce desirable goals using some kind of feedback mechanism after each iteration.

Goals:
The goal represents the main functionality of the system under consideration.

Claims:
Claims represent the rationale for selecting a particular strategy to achieve the goal amongst numerous alternative strategies that exist in order to achieve the same goal.

5.1.2 Feature extraction

Study Instruments:
LoREM : A goal driven method for deriving requirements for self-adaptive systems
REAssuRE: An extended version of the goal driven method LoREM.
RELAX: A non-goal based approach to address uncertainty in self-adaptive systems requirement

Sample models:
The current hypothesis is studied using GridStix, an experimental flood warning system. This offers an executable form of the hypothesis by applying REAssuRE with the help of goal models and Claim Refinement models to the GridStix system.

Baseline results:
The authors of the paper were able to conclude that there was an improvement of 5% in the longevity when Grid-stix used claims. This result can be used as a baseline for further studies to determine whether the added complexity of the run-time model is justified by the improvement in longevity.

6. TRACKING ADVANCES (Papers post 2011)

The following papers track the progress made in the field of requirements-aware systems. For each of the below papers, we have described the keywords and listed the key features of each paper.

6.1 Stateful requirements monitoring for self-repairing socio-technical systems [6]

6.1.1 Keywords

Self-Repair:
Self-repair refers to the ability of the systems to restore fulfillment of their requirements by relying on monitoring, reasoning, and diagnosing on the current state of individual requirements.

Requirements monitoring:
Maintaining the state of each requirement of the system and checking if the system is correctly fulfilling the requirement during runtime.

Goal Models:
Refers to a model-based approach of visualising goals in order to support monitoring and repairing of the proposed stateful goals (end state of the system).

Socio-technical systems:
It is a term used to refer to systems that consist of human, hardware and software components that work together in order to fulfill requirements of the system.

6.1.2 Feature extraction

Motivational Statements:
Existing research on self-repairing has proposed several methods for issues related to requirements monitoring [11], reasoning [12], and diagnosing [13] and self- reconfiguration [14], [15], [16]. However, little is published on how to bridge the gap between the deviations detected and the repairing actions required by specifying precise application-specific self-repairing policies. Nor are there any proposals for an integrated and reusable infrastructure for developing such self-repairing systems. Also, existing proposals generally do not support decentralized requirements monitoring and self-repairing for socio-technical systems. To address the above problems, a fine-grained and stateful requirements monitoring approach towards goal fulfillment is proposed in this paper.

Informative Visualizations:
The Order & Delivery scenario involves two agents, i.e. Commodity Ordering System and Regional Distribution Center, and the simplified requirements of which are described by the goal model are described by Partial Order & Deliver Goal Model figure. Stateful process of requirements fulfillment in terms of goals of all kinds can be generalized by the figure of Elementary Goal State Machine. Hierarchical repairing process illustrates an ideal example of the self-repairing process for a requirements deviation rooted in the goal “Transport by Motorcycle”. Goal “Deliver Commodity” is initially delegated to one Regional Distribution Center. Monitoring and repairing approach is overviewed in in the figure: An overview

of our approach. States, actions and transitions for monitoring and repairing, is shown in Extended Goal State Machine for Requirements Monitoring and Repairing. Goal state machine interactions are embodied in event propagation among different goal instances and relevant state reasoning is described in the figure - State Propagation Example. An example of goal state interactions by general event mapping rules is described in Food Preparing Goal Model

Future work:
Towards the end of the paper, the authors point out that the effectiveness of their current hypothesis and propose the integration of the current work with the runtime modeling of the behaviors of physical world domains. They also mention that there is scope for more experiments with the socio-technical systems in the real world.

6.2. A Taxonomy of Uncertainty for Dynamically Adaptive Systems [7]

6.2.1 Keywords

Dynamically adaptive system (DAS)
Self-reconfiguration enables a dynamically adaptive system (DAS) to satisfy requirements even as detrimental system and environmental conditions arise.

Taxonomy
This paper proposes a classification of potential sources of uncertainty at the requirements, design, and execution phases, and identifies existing techniques for mitigating specific types of uncertainty.

Uncertainty
Refers to the unpredictable system and environmental conditions that a DAS will encounter as it executes.

Requirements Engineering
Requirements engineering (RE) refers to the process of defining, documenting and maintaining requirements and to the subfields of systems engineering and software engineering concerned with this process.

6.2.2 Feature Extraction

Hypothesis
Self-reconfiguration enables a dynamically-adaptive system (DAS) to satisfy requirements even as detrimental system and environmental conditions arise

Patterns:
The authors have provided a detailed description of the steps to keep in mind while describing the taxonomy of uncertainty. They have started by defining the various kinds of uncertainties, provided a template for describing an uncertainty and finally providing a detailed description of the uncertainties in the requirements, design and run-time phase of the development cycle.

Checklists:
The authors provide a checklist in the form of a table summarizing the taxonomy of uncertainty in a DAS.

New results:
This paper also introduces a template to facilitate the organization and reuse of the proposed taxonomy of uncertainty.

Future work
Towards the end of the paper , the authors point the need to include integrating the techniques presented along with existing techniques for resolving combinations of uncertainty, as well as further investigations into avoidance and/or mitigation strategies for those types of uncertainty that have not been as extensively studied.

6.3 Dynamic decision networks for decision-making in self-adaptive systems: a case study [8]

6.3.1 Keywords

Dynamic Decision Networks:
Dynamic Decision networks (DNs) extend Bayesian networks to provide a mechanism for making rational decisions by combining probability and utility theory over changes in variables over time.

Bayesian network
Bayesian network is a probabilistic model that represents a set of random variables or chance nodes and their conditional dependencies.

Decision Node
A decision node D represents the decision taken when specifying the topology to be used in time slice t.

Evidence Node
The evidence node E represents the information observed by monitorables.

6.3.2 Feature Extraction

Baseline results:
The paper describes the procedure and provides results for applying the Dynamic Decision Network methodology to the Remote Data Mirroring application. It features the process of analysis of the problem domain in the DDN terms, deriving a Bayesian Network that describes the Functional and Nonfunctional Requirements and providing the results of application of the same. This procedure and results can be used as comparison in other applications.

Delivery tools:
The experiments have been carried out using the Netica development environment. The tool is available at http://www.norsys.com

Statistical tests:
Extend Bayesian networks are used to provide a mechanism for making rational decisions by combining probability and utility theory. To decide among alternative decisions, the probability- weighted expected utility of each decision given evidence is calculated using a conditional probability method

Commentary:
An important property of BNs is Bayesian inference, which refers to the fact that probabilistic beliefs about random variables can be updated automatically as additional evidence e is learned. The probabilistic models BNs and DNs do not offer mechanisms for representing temporal relations between and within the random variables. Instead, DDNs can be used to represent variables that change over time.

Data:
The data consists of the results of the computation of the property being evaluated for each configurations chosen during time slices are plotted using a graph.

7. CONCLUSION

In earlier work, the idea of requirements-aware systems that could introspect about the extent to which their goals were being satisfied at runtime. When combined with requirements monitoring and self-adaptive capabilities, requirements awareness should help optimize goal satisfaction even in the presence of changing run-time context. Welsh et al describe initial progress towards the realization of requirements-aware systems with REAssuRE. REAssuRE focuses on explicit representation of assumptions made at design time. When such assumptions are shown not to hold, REAssuRE can trigger system adaptations to alternative goal realization strategies.
Later on additional factors were taken into consideration when modelling a requirement aware system. Lingxiao Fu et al proposed a stateful requirements monitoring approach by maintaining an instance of a state machine for each requirement, represented as a goal, with runtime monitoring and compensation capabilities. Ramirez et al synthesized uncertainty concepts from other disciplines, by revisiting the concepts of uncertainty from the perspective of a DAS, proposed a taxonomy of potential sources of uncertainty at the requirements, design, and execution phases, and identified existing techniques for mitigating specific types of uncertainty. Finally Bencomo et al discussed the case for the use of BNs, specifically Dynamic Decision Networks (DDNs), to support the decision-making of self-adaptive systems.

8. FUTURE WORK

Stateful, goal-based monitoring and self-repairing framework needs to be integrated runtime modeling of the behaviors of physical world domains and conducting more experiments with real socio-technical systems. More research is required in the techniques for resolving combinations of uncertainty, as well as further investigations into avoidance and/or mitigation strategies for those types of uncertainty that have not been as extensively studied. Further work is required towards systematic techniques for studying the value of the probabilities that change over time and their impact on the evaluation of the alternative decisions when the Bayesian learning provided by the DDN-based approach is under consideration. Further study on how the quality of the infrastructure monitoring, using the level of confidence of sensors, affects the decisions made by the DAS needs to be looked into. Development of tools to help the requirement engineer to design a DDN would be certainly very helpful as the current tool support imposes limitations. Also there are not many tools that support DDNs.

9. ACKNOWLEDGEMENT

This paper has been written as a part of the Automated Software Engineering course under the guidance of Dr Timothy Menzies, North Carolina State University

10. REFERENCES:

1. Kristopher Welsh, Pete Sawyer and Nelly Bencomo. 2008. Towards Requirements Aware Systems: Run-time Resolution of Design-time Assumptions. 26th IEEE/ACM International Conference Automated Software engineering (ASE 11), IEEE, 2011.
2. Whittle, Jon, Pete Sawyer, Nelly Bencomo, Betty HC Cheng, and Jean-Michel Bruel. "RELAX: a language to address uncertainty in self-adaptive systems requirement." Requirements Engineering 15, no. 2 (2010): 177-196.
3. H. J. Goldsby, P. Sawyer, N. Bencomo, D. Hughes, and B. H. Cheng. Goal-based modeling of dynamically adaptive system requirements. 15th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS), 2008.
4. DeLoach, Scott A., and Matthew Miller. "A goal model for adaptive complex systems." International Journal of Computational Intelligence: Theory and Practice 5.2 (2010): 83-92.
5. N. Bencomo et al., "Genie: Supporting the Model-Driven Development of Reflective, Component-Based Adaptive Systems," Proc. 30th Int'l Conf. Software Eng. (ICSE 08), ACM Press, 2008, pp. 811-814
6. Lingxiao Fu, Xin Peng, Yijun Yu, John Mylopoulos, Wenyun Zhao. 2012. Stateful requirements monitoring for self-repairing socio-technical systems. Requirements Engineering Conference (RE), 2012 20th IEEE International.
7. A Ramirez, A. Jensen, and B. Cheng. 2012. A Taxonomy of Uncertainty for Dynamically Adaptive Systems. Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop.
8. Bencomo, Nelly, Amel Belaggoun, and Valerie Issarny. "Dynamic decision networks for decision-making in self-adaptive systems: a case study." Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on. IEEE, 2013.
9. J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "Relax: a language to address uncertainty in self-adaptive systems requirement," Requirements Engi- neering, vol. 15, no. 2, pp. 177–196, 2010.
10. A.R.Dingwall-Smith,"Run-time monitoring of goal-oriented requirements specifications", Ph.D. dissertation, University of London, 2006.
11. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Reasoning with goal models", in ER, 2002.
12. Y.Wang,S.A.Mcllraith,Y.Yu,and J.Mylopoulos,"An automated approach to monitoring and diagnosing requirements", in ASE, 2007.
13. Y. Wang and J. Mylopoulos, "Self-repair through reconfiguration: A requirements engineering approach", in ASE. IEEE Computer Society, 2009, pp. 257–268.
14. M. J. Khan, M. M. Awais, and S. Shamail, "Enabling self- configuration in autonomic systems using case-based reasoning with improved efficiency", in ICAAS, 2008.
15. F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "An architecture for requirements-driven self-reconfiguration", in CAiSE, 2009.
16. M.S.Feather, S.Fickas, A.V.Lamsweerde, and C.Ponsard, "Reconciling system requirements and runtime behavior". In IWSSD '98: Proceedings of the 9th International Workshop on Software Specification and Design, 1998.
17. Y. Yu, J. C. S. do Prado Leite, and J. Mylopoulos "From goals to aspects: Discovering aspects from requirements goal models". In Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004), 2004.
18. Y. Yu, J. Mylopoulos, A. Lapouchnian, S. Liaskos, and J. C. Leite. "From stakeholder goals to high-variability software design". Technical report csrg-509, University of Toronto, 2005.

19. A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu. “Towards requirements-driven autonomic systems design”. In DEAS ’05: Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software, 2005.
20. N. Bencomo et al., “Genie: Supporting the Model-Driven Development of Reflective, Component-Based Adaptive Systems,” Proc. 30th Int’l Conf. Software Eng. (ICSE 08), ACM Press, 2008, pp. 811-814
21. G. Coulson, G. Blair, P. Grace, A. Joolia, K. Lee, J. Ueyama, and T. Sivaharan. A generic component model for building systems software. ACM Transactions on Computer Systems, February, 2008.
22. P. Grace, G. Coulson, G. Blair, and B. Porter. Deep middleware for the divergent grid. In IFIP/ACM/USENIX Middleware, France, 2005.
23. DeLoach, Scott A., and Matthew Miller. "A goal model for adaptive complex systems." International Journal of Computational Intelligence: Theory and Practice 5.2 (2010): 83-92.
24. Cheng et al. (2008) Software engineering for self-adaptive systems: a research road map, Dagstuhl-seminar on software engineering for self- adaptive systems. In: Software engineering for self-adaptive systems. Lecture Notes in Computer Science, Springer, Berlin, p 5525.
25. Lapouchnian A et al. (2006) Requirements-driven design of autonomic application software. In: Proceedings of CASCON
26. Morandini M, Penserini L, Perini A (2008) Modelling selfadaptivity: a goal-oriented approach. In: Proceedings of second IEEE international conference on self-adaptive and self-organizing systems (SASO), pp 469–470
27. Yu, E. 1995. Modeling Strategic Relationships for Process Reengineering. PhD thesis, University of Toronto.
28. I*. (2015, December 1). In Wikipedia, The Free Encyclopedia. Retrieved 00:02, December 15, 2015, from https://en.wikipedia.org/w/index.php?title=I*&oldid=693345491
29. Van Lamsweerde, A. and Letier, E. 2000. Handling Obstacles in Goal-Oriented Requirements Engineering. IEEE Trans. on Software Engineering. 26(10): 978-1005.
30. Morreale, V., Bonura, S., Francaviglia, G., Centineo, F., Cossentino, M., and Gaglio, S. 2006. Goal-Oriented Development of BDI Agents: The PRACTIONIST Approach. In Proceedings of the IEEE/WIC/ACM international Conference on intelligent Agent Technology (IAT), 66-72. IEEE Computer Society.