Modelling and Reasoning about Remediation Actions in BDI Agents

(Extended Abstract)

João Faccin
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Brazil
jgfaccin@inf.ufrgs.br

Ingrid Nunes
UFRGS, Porto Alegre, Brazil
TU Dortmund, Dortmund, Germany
ingridnunes@inf.ufrgs.br

ABSTRACT

Remediation actions are performed in scenarios in which consequences of a problem should be promptly mitigated when its cause takes too long to be addressed or is unknown. Existing approaches that address these scenarios are application-specific. Nevertheless, the reasoning about remediation actions as well as cause identification and resolution, in order to address problems permanently, can be abstracted in such a way that they can be incorporated to agents. In this paper, we present a domain-independent approach that extends the *belief-desire-intention* (BDI) architecture, providing means of modelling and reasoning over causal relationships and remediation actions.

Keywords

BDI architecture, software agents, remediation action, cause-effect, goal generation, plan selection, software reuse

1. INTRODUCTION

Performing *remediation actions* is a strategy typically applied when we face problems in which consequences, or *effects*, must be mitigated before their *cause* is addressed. In this scenario, a problem can only be considered solved when both cause and effect are also solved. There are many issues related to the process of carrying out such problem-solving strategy. On the one hand, mitigating a consequence without considering its cause would result in resources and effort spent ineffectively. On the other hand, immediately dealing with a cause disregarding its consequences would worsen the effects if the cause solution is not provided instantly. There are also situations in which the cause is *unknown* and should be investigated so that the real problem can be identified and resolved; otherwise, effects will likely reappear.

In Computer Science, many instances of this scenario can be observed in various contexts, such as self-healing systems [1] and network resilience [5]. However, despite being able to properly deal with consequences and its causes, existing proposals addressing such domains provide solutions that are individually analysed and implemented, and in which the actions of mitigating effects and searching for causes are all explicitly modelled and *hard coded*. The agent technology is an alternative to the flexible implementation of these proposals. Nevertheless, existing agent-based approaches that deal with the proposed scenario also implement the reasoning over causes and effects manually.

In this paper, we propose a domain-independent approach that extends the *belief-desire-intention* (BDI) architecture [4] by allowing agents to select suitable, possible remediation, plans to solve a problem and deal with its possible causes, also considering agent preferences and constraints. This extended architecture comprises a set of components to capture the required domain knowledge to support agents in making such decisions. Such knowledge is used in a customised reasoning mechanism, which can select remediation plans when needed, and to generate goals to search and deal with problem causes.

2. BDI AGENT MODELLING

Agents based on the BDI architecture [4] are structured in terms of the mental attitudes of beliefs, desires and intentions. Their goals are explicitly specified, and plans able to achieve these goals are provided at design time. Many concepts of our target scenario can be associated with existing components of the BDI architecture, e.g. remediation actions and problems, which can be related to plans and goals, respectively. Other domain-independent concepts, however, have no corresponding components and were incorporated into our extended architecture to make agents able to mitigate problems through the use of remediation plans, as well as to identify and solve their causes when needed. We detail such components as follows.

A goal represents a state of the world the agent wants to bring about. When attempting to achieve a goal, the actions performed by the agent can consume *resources*, which are any consumable supplies of assets whose consumption can be measured by the agent, e.g. processing time and allocated memory. There are situations in which these resources are limited to a given amount. In such scenarios, there is no advantage for the agent to execute actions that will require more resources than available. Constraints on the resource consumption while achieving a goal are specified using *operation constraints*. Moreover, when there are different ways to achieve the same objective, there may be restrictions on how resources must be spent, i.e. when it is more valuable to maximise or minimise the resource consumption. The specification of such restrictions is given by an *objective function*. We named *constrained goal* the goal that encompasses the

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

notions of objective functions and operation constraints. A *plan required resource* specifies the amount of a given resource that a plan is expected to consume when trying to reach a goal. A set of plan required resources together with sets of preconditions, achievable goals and actions define a plan. Finally, our *extended BDI (EBDI) agent* comprises a belief base, a plan library, a set of goals, and a *preference function*, which specifies the importance given by our agent to each resource.

Although providing the information required to consider the constrained resource consumption during the process of goal achievement, the described components do not encompass the knowledge that allows the reasoning over effects and their causes. In our approach, a goal is a problem to be solved, which may be either an effect or a cause, or even both at the same time. While an effect is composed of a single fact, a cause can comprise a set of them. A *cause-effect relationship* thus specifies the association between the effect of a problem and the possible facts that comprise its cause. We distinguish mandatory, optional and alternative facts.

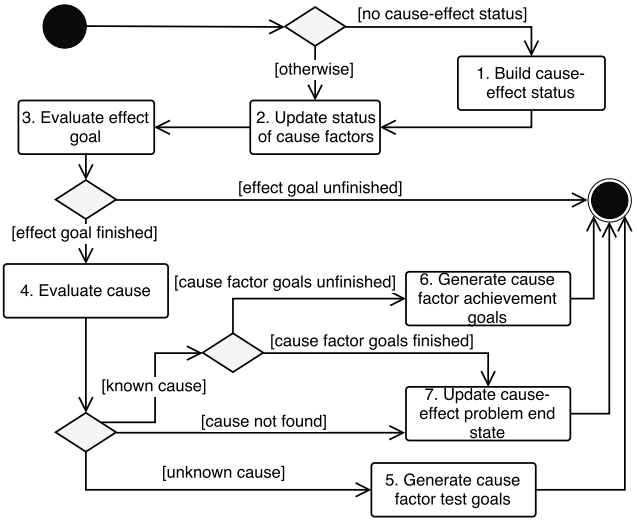

Figure 1: Option Generation Activity Diagram.

3. CUSTOMISED BDI REASONING CYCLE

The behaviour of a typical BDI agent is defined by four abstract functions, which are integrated into a reasoning cycle. The first of them, named belief revision function, is responsible for updating agent beliefs based on internal or external events perceived. The option generation function, in turn, maintains updated the set of agent goals, adding or dropping goals when needed, while the filter function selects, from the set of agent goals, those that will become intentions. Finally, the plan selection function chooses, from a set of predefined suitable plans and according to a given criteria, a plan to achieve a goal that is a current intention.

To make agents able to deal with our target scenario, functions of this reasoning cycle must be customised. Our customised plan selection function is implemented to perform two key steps. First, for a given goal, a set of candidate plans is selected from the agent's plan library. These candidate plans are those capable of achieving the goal while their corresponding plan required resources satisfy the goal's operation constraints, if any. The second step consists of selecting, from the set of candidate plans, the plan that best satisfies agent preferences over resources. Such satisfaction is calculated according to the information provided by plan required resources and the goal's objective function. Considering the case in which the goal being addressed is part of a cause-effect relationship, the plan selected for execution may fully address the effect and its cause, or may be a remediation action that only deals with the effect. In this last situation, the current cause of the goal must still be identified and solved. This task is carried out by our customised option generation function.

The option generation function (see Figure 1) has three key roles in our work. Given a goal that is an effect in a cause-effect relationship, this function is responsible for identifying the possible cause factors, evaluating such factors to find the current cause of the problem, and keeping track of them to assess whether they are solved. The identification of possible cause factors is made directly from the information provided by existing cause-effect relationships. Evaluating these factors, however, requires the execution of additional steps. The current status of a factor may be already known by the agent, i.e. be present in its belief base, or completely unknown. The latter situation requires the creation of a test goal (from the AgentSpeak language [3]) in order to discover the current factor status. When all possible cause factors have their status known, cause-effect relationships are consulted again to define which of these factors comprise the current cause of the problem. Our option generation function thus creates an achievement goal for each of them.

The tracking of effects and cause factors is performed through the use of an internal reasoning cycle structure, which we called *cause-effect status*. A cause-effect status is able to register changes performed in the status of each factor comprising the cause of a given problem, as well as to identify if a cause factor (1) remains unaddressed, (2) is currently being tackled, or (3) is already solved. When an effect is remediated, and every factor that comprises its corresponding cause-effect status is resolved, the problem is considered solved.

4. CONCLUSION

Remediating consequences before identifying and solving causes of a problem is a problem-solving strategy typically applicable in several domains. Existing approaches that adopt such strategy are application-specific and requires manual implementation. In this paper, we introduced an approach that exploits the BDI architecture to provide a domain-independent solution both able to mitigate problems and address their causes. We evaluated our approach by taking an existing network resilience scenario [2], which is implemented in an application-specific way, and developed it with our approach. Results show that we are able to reproduce the original behaviour in a domain-independent way using our solution, without impacting the performance of the system, as well as reducing the development effort.

Acknowledgments

This work receives financial support of CNPq 442582/2014-5. The authors would like to thank CNPq 141840/2016-1 and 303232/2015-3, CAPES 7619-15-4, and Alexander von Humboldt, ref. BRA 1184533 HFSTCAPES-P.

REFERENCES

[1] D. Breitgand, M. Goldstein, E. Henis, O. Shehory, and Y. Weinsberg. PANACEA towards a self-healing development framework. In *Integrated Network Management (IM)*, pages 169–178. IEEE, 2007.

[2] I. Nunes, F. Schardong, and A. Schaeffer-Filho. BDI2DoS: an application using collaborating BDI agents to combat DDoS attacks. *Journal of Network and Computer Applications*, 2017.

[3] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-agent World : Agents Breaking Away: Agents Breaking Away*, MAAMAW '96, pages 42–55, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.

[4] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *First International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, 1995.

[5] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245–1265, 2010.