# Dynamic change impact analysis for maintaining and evolving agent systems

# (Extended Abstract)

Hoa Khanh Dam and Aditya Ghose
School of Computer Science and Software Engineering
University of Wollongong
New South Wales 2522, Australia
{hoa,aditya}@uow.edu.au

## ABSTRACT

In contrast to an increasing number of agent-based applications in various domains, there has been very little work on maintenance and evolution of agent systems. This paper addresses this gap with a focus on change impact analysis, i.e. estimating the potential effects of changes before they are made as an agent system evolves. We propose a technique for performing impact analysis in an agent system using dynamic information about agent behaviour. Our approach builds a representation of an agent's behaviour by analyzing its execution traces which consist of goals and plans, and uses this representation to estimate impacts.

## Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Distribution, Maintenance, and Enhancement

## General Terms

Design

## Keywords

change impact analysis, multi-agent systems

## 1. INTRODUCTION

Complex agent-based applications will evolve and will need to be maintained throughout their life, which would require substantial costs. The focus of this paper is on *change impact analysis* of agent systems – predicting the potential consequences of a proposed change. Change impact analysis [1] usually starts with the software maintainer examining the change request and determining the entities initially affected by the change (i.e. the *primary changes*). The software maintainer then determines other entities in the system that have potential dependency relationships with the initial ones, and forms a set of impacts. Those impacted components also relate to other entities and thus the impact analysis continues this process until a complete impact set is obtained. Change impact analysis plays a major part in planning and establishing the feasibility of a change in terms of predicting the cost and complexity of the change (before implementing it). This help reduce the risks associated with making changes that have unintended, expensive, or even disastrous effects on an existing system. Furthermore, change impact analysis can be used to predict or identify parts of a system that will need to be retested (i.e. regression testing) as a result of changes.

Although notions and ideas from a large body of work addressing change impact analysis for classical software systems (e.g. [1]) can be adapted, agent systems with their distinct characteristics and architectures introduce new problems in software maintenance. For instance, while object-oriented software deals with classes, methods and fields, a typical agent-based software, e.g. the Belief-Desire-Intention (BDI) [4] agents, consists of agents, plans, events/goals and beliefs.

A recent effort [2] has proposed a change impact analysis technique specifically for agent systems. It is however based on static analysis of agent source code, which can safely estimate the impact of changes, but its conservative principle leads to a large impact set which may contain many unnecessary entities. This is because static analysis considers all possible behaviours of a software system while only a subset of such behaviours may be executed in practice.

Therefore, this paper takes a dynamic approach to change impact analysis for agent systems: we propose an impact analysis technique using dynamic information about agent behaviour. Our dynamic impact analysis technique focuses specifically on agent systems, in particular the well-known and widely-used BDI agents. We identify the essential information needed to perform dynamic impact analysis on a BDI agent system. Such dynamic information is collected from execution data for a specific set of agent executions (e.g. executions based on an operational profile or executions of test suites) which contains two key aspects determining the behaviour of a BDI agent system: the *goals* an agent pursued and the *plans* it deployed to achieve those goals. We further define a technique to analyse that information to determine when a plan or goal is changed, what other plans and goals are potentially impacted by the change.

## 2. DYNAMIC IMPACT ANALYSIS

The hierarchical structure of BDI plans which determine the run-time behaviour of a BDI agent can be viewed as a goal-plan tree where each goal has children representing the

**Appears in:** *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

relevant plans for achieving it, and each plan has children representing the subgoals (including primitive actions) that it has. This goal-plan tree can be seen as an "and/or" tree: each goal is achieved by a successful execution of one of its plan ("or"), and the success of each plan relies on all of its sub-goals being resolved ("and"). Figure 1 shows an example of such an goal-plan tree. Goal G can be realised by either plan P1 or P2. Plan P1 has two subgoals G1 and G2 in which G1 can be achieved by one of plans P3, P4 and P5, and G2 can be achieved by plan P6. Plan P2 has only one subgoal G3, which can be realised by either plan P7 or P8.

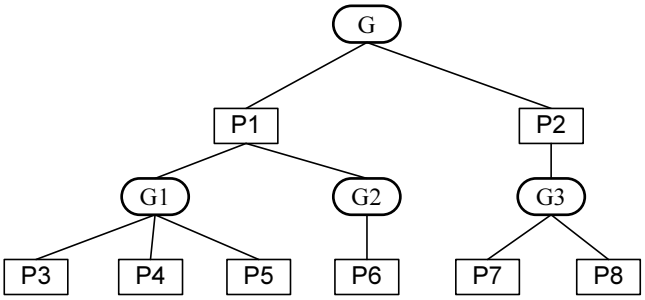


**Figure 1: A goal-plan tree for agent $A$**

As an example, suppose we have a single execution trace $t$, shown by a string of letters in figure 2, for an agent $A$ whose a goal-plan tree appears in figure 1. Note that $G_p$ denotes goal G being posted, whereas $G_s$ indicates goal G being successfully achieved. Similarly, $P_e$ denotes plan P beginning execution and $P_s$ indicates a successful completion of plan P. As can be seen, the execution trace in figure 2 demonstrates that after goal G is posted, plan P1 executes, goal G1 is posted, plan P3 executes and successfully completes, goal G1 is successfully resolved, goal G2 is posted, plan P6 executes and successfully completes, goal G2 is successfully resolved, plan P1 successfully completes, and goal G is successfully achieved.

$G_p\ P1_e\ G1_p\ P3_e\ P3_s\ G1_s\ G2_p\ P6_e\ P6_s\ G2_s\ P1_s\ G_s$

**Figure 2: A typical execution trace $t$ for an agent $A$**

Assume that we propose to change plan $P6$ in the above example, an impact analysis technique needs to determine the other plans and/or goals that are potentially affected by the change (i.e. the impact set). The static analysis technique proposed in [2] computes the impact set by considering static (direct and indirect) dependencies between $P6$ and other goals or plans in the agent system. It works under the assumption that a change in $P6$ has potential impact on any node reachable from $P6$ in the goal-plan tree for agent $A$. Therefore, an impact set of plan $P6$ returned by the static technique in [2] contains all entities in the goal-plan tree in figure 1. This would result in highly inaccurate impact set, as evidenced by the experimental result (i.e. precision of approximately 0.3–0.4). We will now show that our dynamic analysis technique which relies on information from execution traces can predict impact sets that are more accurate than those computed by static analysis.

Our dynamic analysis technique relies on execution traces such as the one in figure 2 rather static goal-plan trees. Given a set of changes, we adapt the PathImpact technique [3] to perform forward and backward walks of a trace to identify the impact set of the changes. The forward walk determines all plans executed and all goals posted after the changed goal/plan, whereas the backward walk identifies plans/goals into which the execution can return. More specifically, for each changed entity E (which can be either a plan or goal) and each occurrence of $E_e$ (if E is a plan) or $E_p$ (if E is a goal), we will do the following. Note that we will illustrate our technique using an example of trace $t$ in figure 2 and a change set {P6} (i.e. only plan P6 is modified).

- In the forward walk, we start from the entity immediately following $E_e$ (if E is a plan) or $E_p$ (if E is a goal), add every plan executed or goal posted into the impact set (i.e. every entity $F$ such that the trace contain an entry $F_e$ or $F_p$ after the occurrence of $E_e$ or $E_p$), and count the number of unmatched successes. In our example, in the forward walk we start at $P6_s$ and add nothing to the impact set since there is no plan executed or goal posted after P6. We however count 3 unmatched successes (i.e. $G2_s$, $P1_s$, and $G_s$)
- In the backward walk, we begin from the entity immediately preceding $E_e$ (if E is a plan) or $E_p$ (if E is a goal), and add into the impact set as many unmatched plans or goals as the number of unmatched successes counted in the forward walk. In our example, we add G2, P1, and G to the impact set.
- Add E to the impact set if it is not already there. Therefore, the impact set in our example would be {P6, G2, P1, G}.

The above trace is an example of a typical, successful execution. An agent's execution may however contain parallelisation (e.g. achieving two goals concurrently), interruption (e.g. suspending an executing plan to deal with higher priority events), and failures handling (e.g. trying alternative plans in pursuing a goal). We can apply the same technique described earlier to determine impact sets from traces derived from those agent behaviours. In practice, there are usually multiple execution traces of an agent system. In this case, we process each single trace and compute the union of the impact sets returned by each execution traces.

# 3. REFERENCES


[1] R. Arnold and S. Bohner. *Software Change Impact Analysis*. IEEE Computer Society Press, 1996.

[2] H. K. Dam and A. Ghose. Automated change impact analysis for agent systems. In *Proceedings of the 27th IEEE International Conference on Software Maintenance*, ICSM '11, pages 33–42, Washington, DC, USA, 2011. IEEE.

[3] J. Law and G. Rothermel. Whole program path-based dynamic impact analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 308–318, Washington, DC, USA, 2003. IEEE Computer Society.

[4] A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. R. Lesser and L. Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, pages 312–319. The MIT Press, 1995.