A Systematic Approach for Detecting Defects in Agent Designs

(Doctoral Consortium)

Yoosef Abushark*
Supervisors: John Thangarajah, James Harland and Tim Miller†
RMIT University
Melbourne, Australia
yoosef.abushark@rmit.edu.au

ABSTRACT

Multi-agent systems are increasingly being used in complex applications due to features such as autonomy, proactivity, flexibility, robustness and social ability. Thus, techniques to detect and avoid defects in such systems are valuable. In particular, it is desirable to detect issues as early as possible in the development lifecycle. This research aims to find ways in which to ensure the correctness of agent design artefacts against different point-of-reference: requirements and interaction protocols. The proposed approach will be applicable at design time not requiring source code or a formal design model.

Keywords

AOSE Methodology, Goal-Oriented Requirements, Interaction Protocol

1. BROADER RESEARCH CONTEXT

Autonomous agents are widely-used for developing systems that are highly dynamic in nature in a broad range of domains [6]. Most existing work on verifying BDI agent systems has focused on formal verification (e.g. [2]), particularly using model checking techniques (e.g. [3]) and theorem proving (e.g. [10]), or on runtime testing (e.g. [8]). Such work tends to focus on the verification of complete agent programs, requiring source code or a formal design model. However, it is long established that early detection and resolution of software defects saves time and money [[1], Page 1466]. Our aim therefore is to develop a suite of lightweight techniques, supported by tools, for detecting defects at the design phase, prior to implementation. One of the barriers to the widespread adoption of the agent technology in the industry is its reliability [4]. In order to increase the trust of such systems, many testing and debugging techniques have been proposed. Some of these techniques (e.g. model-based testing) rely on the design artefacts. As a result, it is vital to check these artefacts and ensure their correctness.

The questions that need to be addressed by this research are as follows:

1. What is the **scope** of the static verification in the context of the **BDI**-model of agency?
   The answer to this question states all possible checkable artefacts in the context of BDI agent-based systems. Also, it identifies the point-of-reference and the design units to be considered in the proposed approach. This question is part of the necessary background knowledge, I consider it as a first step towards the final aim of this research.
2. What is a suitable **executable model** for agent-based design artefacts?
   This question investigates different formalisms with the aim to be used as an executable representation for the point-of-reference artefacts. Since the point-of-reference encompasses two artefacts: requirements and interaction protocols, this question has two sub-questions as follows:
   (a) What is a suitable executable model to represent **requirements**?
   (b) What is a suitable executable model to represent **interaction protocol**?
3. What is a suitable **executable model** to represent **agents' detailed models**?
   It is more likely to have the design of given point-of-reference (requirements specifications or interaction protocol) scattered across multiple agents. In this question, I investigate how to merge all the relevant entities of a given point-of-reference into one coherent structure.
4. How can an **automated checking framework** be developed to **effectively detect design defects**?
   This question is addressed through addressing the following two questions:
   (a) How to extract all possible behavioural runs out of the agent detailed deigns?
   (b) How to check the behavioural runs against the point-of-reference models?

Evaluation of the proposed approach

The evaluation aims to ensure that the effectiveness of the framework proposed through answering the following questions:

(a) Can the framework detect design defects?
(b) What is the level of false positives that are generated by the proposed approach?
(c) How scalable is the proposed checking framework?

*Acknowledges King Abdulaziz University for Scholarship.
†University of Melbourne, Melbourne, Australia.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

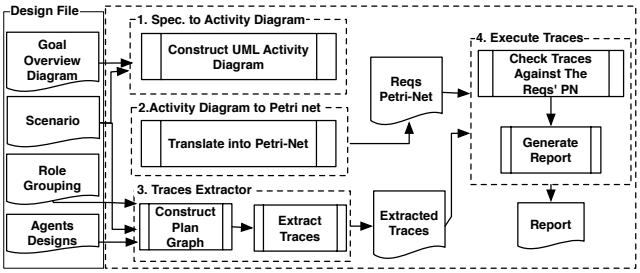

Figure 1: Proposed Framework.

2. METHOD

This research is intended to find a systematic way (Figure 1), supported by algorithms, to ensure the correctness of the detailed design artefacts (events and their associated plans) with respect to the specified point-of-reference (i.e. **requirements** or **interaction protocols**).

In order to verify the design with respect to a given point-of-reference, the point-of-reference should be expressed in a formal notation. Thus, the work of Poutakidis et al. [9] is adopted, as it proposes a way to translate AUML sequence diagrams into more formal representation, *place/transition* Petri-Nets (PN). Then, a plan graph, as proposed in [5], is constructed from the detailed design artefacts of the involved agent in the point-of-reference to act as a coherent structure that merges the designs. The correctness check is performed through running the extracted behavioural runs out of the plan graph over the point-of-reference's PN and logging the existence of any violations.

A plan graph may capture many execution flow fragments including: sequential, selection, loop and parallel. Due to the non-deterministic nature of MAS, plan graphs must be traversed in a way that guarantees that the behavioural runs generated cover all possible combinations. The existing graph traversal algorithms do not fulfil such a purpose. The intention is to transform the plan graph into *place/transition* PN, since it can be used to study and describe systems that have the non-deterministic nature [7]. Then, a reachability graph (RG) for that PN is calculated, since reachability is one of the PN's proprieties that is used for studying its dynamic behaviour [7]. Since that RG captures all possible ways of executing the PN, behavioural runs that cover all possible combinations can be extracted through traversing the RG. The two Petri nets are then compared for similarity through running the behavioural runs extracted from the RG over the point-of-reference's PN. To maintain the computation complexity of the approach, especially in situations when the RG includes too many behavioural runs, the approach considers one run at a time.

3. PROTOTYPE IMPLEMENTATION

I implemented the proposed approach as an eclipse plug-in that integrates with the Prometheus Design Tool (PDT). The tool takes the design file in an XML format, and tokenises the point-of-reference out of the design file. Then it translates them into PNs. Regarding the agent detailed design notations, the tool extracts the entities that are related to the specified point-of-reference. Then, it applies a set of merging rules on the specified point-of-reference to generate the abstract description of the intended plan graph. Then, it uses the abstract description to generate a DOT Graph source script for the plan graph. The tool then extracts all possible behavioural runs and executes them against the specified point-of-reference's PN, and reports any inconsistency.

4. EVALUATION

To the best of my knowledge, there are no existing benchmarks that can be used in evaluating the proposed approach. Thus, the plan was to conduct an empirical evaluation through applying the proposed approach on the artefacts of complete agent-systems (21 cases). Some of these systems were developed as final projects in the Agent-Oriented Programming and Design course, offered at RMIT University, whereas others were developed within the agent group at the university. Also, there are two case studies were included in the evaluation: oil production simulation and an air traffic management system. Regarding the scalability of the approach, a scalability analysis has been conducted on 24 synthesised plan graphs to measure how scalable the approach is. The generation of these plan graphs was systematic with varying the size and the amount of parallelism up until the time taken to extract the traces are still "reasonable". I define "reasonable" to be within 24 hours.

Both analysis methodologies were used: quantitative, by measuring the number of errors detected through applying the proposed framework and qualitative in terms of judging the criticality of the errors and their categories: false and true positives.

The results of the empirical evaluation conducted showed that the proposed approach is able to detect defects in agent designs, with a low number of false positives and generally in a reasonable amount of time. The scalability analysis performed showed that the proposed approach can verify large plan graphs in under 24 hours, despite the exponential explosion in traces as designs get larger.

REFERENCES

[1] B. W. Boehm. Understanding and controlling software costs. *Journal of Parametrics*, 8(1):32–68, 1988.

[2] M. Dastani, K. V. Hindriks, and J.-J. Meyer. *Specification and verification of multi-agent systems*. Springer Science & Business Media, 2010.

[3] L. A. Dennis, M. Fisher, M. P. Webster, and R. H. Bordini. Model checking agent programming languages. *Automated software engineering*, 19(1):5–63, 2012.

[4] N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[5] T. Miller, L. Padgham, and J. Thangarajah. Test coverage criteria for agent interaction testing. In *AOSE*, pages 91–105. Springer, 2010.

[6] S. Munroe, T. Miller, R. A. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: Lessons, experiences and challenges in commercial applications of agents. *The Knowledge Engineering Review*, 21(04):345–392, 2006.

[7] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[8] L. Padgham, Z. Zhang, J. Thangarajah, and T. Miller. Model-based test oracle generation for automated unit testing of agent systems. *Software Engineering, IEEE Transactions on*, 39(9):1230–1244, 2013.

[9] D. Poutakidis, L. Padgham, and M. Winikoff. Debugging multi-agent systems using design artifacts: The case of interaction protocols. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 960–967. ACM, 2002.

[10] J. Sudeikat, L. Braubach, A. Pokahr, W. Lamersdorf, and W. Renz. Validation of bdi agents. In *Programming Multi-Agent Systems*, pages 185–200. Springer, 2006.