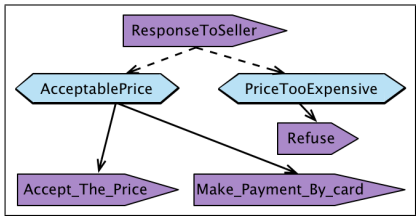

Figure 10: Buyer Agent Overview Diagram

For example, in 'Sale Transaction 2', the correction of the missing message and the two long traces from the first iteration revealed an additional missing message in the next iteration, but also removed 12 of the 13 message ordering related warning messages.

As mentioned in Section 4.2, there will be some failures that do not represent defects in the agent designs. In 'Sale Transaction 2', there are still two problems raised in the third iteration (Table 4) that are categorised as unknown. Figure 10 shows one of these. The participant designed both messages 'Accept_The_Price' and 'Make_Payment_By_Card' to be sent by one plan. Our approach considered that these messages could be sent in any order. However, 'Accept_The_Price' must be sent first to correspond to the protocol. Despite this, we have not classified this as a true positive defect, because it is unclear if the participant assumed that the ordering is implicit.

Table 5 shows the time-cost for extracting traces from the reachability graph and executing them against the protocol's Petri Net. The cost is low for all iterations, except for iteration 1 of 'Sale Transaction 2'. Although the time for extracting such a large number of traces was relatively low, a defect in that design caused an explosion in the number of traces, which consequently took over five and a half hours to execute on the Petri Net. This indicates a potential for explosion in execution time, and that further work is needed to mitigate this problem.

6. CONCLUSIONS

In this paper, we proposed an approach and tool support for finding defects in agent designs with respect to interaction protocol specifications. This approach involves generating the set of possible traces permitted by the agent detailed designs, and checking whether the sequences of messages in these traces are valid with respect to a given protocol specification. Although our tool supports only designs written using the Prometheus methodology, we believe the approach is general enough to work with other AOSE methodologies that follow the BDI model of agency.

We evaluated our approach on four designs developed by relatively experienced developers. The results 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.

In future work, we will refine the approach to reduce false positives. One step would be to allow context conditions in BDI plans, which could then be used to filter out some traces that currently lead to false positives. Our evaluation indicated that some designs can result in an explosive number of traces, and therefore, long execution times. To mitigate this problem, we plan to *prioritise* traces, with the goal of finding and reporting defects early in the execution.

7. REFERENCES

[1] J. Botía, J. Gómez-Sanz, and J. Pavón. Intelligent data analysis for the verification of multi-agent systems interactions. *IDEAL*, pages 1207–1214, 2006.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent oriented software development methodology. *AAMAS*, 8(3):203–236, 2004.

[3] L. Cernuzzi and F. Zambonelli. GAIA4E: A tool supporting the design of MAS using Gaia. In *ICEIS*, pages 82–88, 2009.

[4] B. Cox, J. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *First USENIX Workshop on e-Commerce*, pages 77–88, 1995.

[5] M. Dastani, J. Brandsema, A. Dubel, and J.-J. Meyer. Debugging bdi-based multi-agent programs. *Programming Multi-Agent Systems*, pages 151–169, 2010.

[6] S. A. DeLoach and J. C. Garcia-Ojeda. O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. *IJAOSE*, 4(3):244–280, 2010.

[7] L. Dennis, M. Fisher, M. Webster, and R. Bordini. Model checking agent programming languages. *Automated Software Engineering*, 19(1):5–63, 2012.

[8] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. Model checking early requirements specifications in tropos. In *Proceedings of 5th IEEE International Symposium on Requirements Engineering*, pages 174–181. IEEE, 2001.

[9] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Eng. App. of AI*, 18(2):159–171, 2005.

[10] J. J. Gomez-Sanz, R. Fuentes, J. Pavón, and I. García-Magariño. INGENIAS development kit: a visual multi-agent system development environment. In *AAMAS*, pages 1675–1676. IFAAMAS, 2008.

[11] T. Miller, L. Padgham, and J. Thangarajah. Test coverage criteria for agent interaction testing. *AOSE XI*, pages 91–105, 2011.

[12] S. Munroe, T. Miller, R. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: Lessons, experiences and challenges in commercial applications of agents. *KER*, 21(4):345, 2006.

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

[14] C. Nguyen, S. Miles, A. Perini, P. Tonella, M. Harman, and M. Luck. Evolutionary testing of autonomous software agents. *AAMAS*, 25(2):260–283, 2012.

[15] J. Odell, H. Van Dyke Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *AOSE*, pages 201–218. Springer, 2001.

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

[17] L. Padgham and M. Winikoff. *Developing intelligent agent systems: a practical guide*, volume 1. Wiley, 2004.

[18] D. Poutakidis, L. Padgham, and M. Winikoff. Debugging mass using design artifacts: The case of interaction protocols. In *AAMAS*, pages 960–967. ACM, 2002.

[19] R. Pressman. *Software engineering: a practitioner's approach*, volume 7. McGraw-Kill New York, 2009.

[20] A. Rao and M. Georgeff. BDI agents: From theory to practice. In *AAMAS*, pages 312–319, 1995.

[21] L. Sterling and K. Taveter. *The Art of Agent-Oriented Modeling*. MIT Press, 2009.

[22] M. Winikoff. Towards making agent UML practical: A textual notation and a tool. In *Quality Software, 2005.(QSIC'5)*, pages 401–406. IEEE, 2005.

[23] M. Wooldridge, N. Jennings, and D. Kinny. The Gaia methodology for agent oriented analysis and design. *AAMAS*, 3(3):285–312, 2000.