Using Decomposition to Model Multi-agent Interaction Protocols in Event-B

Elisabeth Ball and Michael Butler

Dependable Systems and Software Engineering, Electronics and Computer Science,
University of Southampton, UK
{ejb04r, mjb}@ecs.soton.ac.uk

Abstract. This paper outlines a practical approach to the formal development of multi-agent systems (MAS). Rigorous design practices are needed if MAS are to be used for critical applications. Event-B is a formal method that can be used in the development of reactive systems. Decomposition is used as part of the approach to reduce the complexity of modelling MAS. The experience of modelling MAS Interaction Protocols in Event-B using a decomposition method is described. Future work is required to further model MAS.

1 Introduction

Distributed problems exist in dynamic environments. Software systems that can solve this type of problem need to be able to respond dynamically to change in both the problem and the environment. Agents are software entities that encapsulate their behaviour and can respond, both pro-actively and reactively, to changes in their environment. Predefined rules and plans guide behaviour that is motivated by internal goals [1].

MAS are complex distributed systems that need to be developed using rigorous design practices when being created for critical applications. The B-Method is a Formal Method used to model and reason about systems [2]. Event-B is an adaptation of the B-Method that is more suitable for reactive systems [3]. Software development in Event-B involves abstractly specifying the requirements of the system and then refining these requirements through several steps to create a concrete description of the system that can be translated to programming code. The correctness and consistency of each model can be proven, as can the relationship between abstract models and their refinements.

Decomposition and abstraction are techniques for coping with complexity. In an Event-B development abstraction is used to deal with the complexity of the system. As the model is refined it becomes less abstract and more complex. Decomposition is ideal for simplifying the increasing complexity of the model.

2 Motivation

When MAS are used to solve critical problems the agents need to be relied upon to work within boundaries. Formal development provides a functional trust for

distributed critical systems that can be applied to MAS. The goal of this research is to make formal development easier to integrate into Agent Oriented Software Engineering (AOSE). This will also show the potential of Event-B when it comes to modelling complex distributed systems.

3 Progress

MAS are complex systems with easily identifiable first class components. Decomposing a MAS into components reduces the complexity of modelling the system. The first step in modelling MAS using Event-B was identifying a suitable pattern for decomposition.

A case study was used to develop the decomposition method using the B4Free tool [4] and the Click 'n' Prove [5] interface. The abstract model defined a banking system and customers that can query their balance. The transaction was given a state, each event changed the state and the next event was triggered.

During refinement the bank and customer were logically separated, each side maintained their own record of the transaction state. Middleware was added to model the coordination between the bank and customer. The middleware was developed over several refinements due to the complexities of the proof.

For the decomposition method to be useful it should take advantage of the capabilities of the development tools. A final refinement was added to the development to recompose abstract models of the components. The INCLUDES clause was used to access the events of the components. This model was then proven to be a refinement of the original abstract model. Each component can then be refined further to model their individual functionality in greater detail without needing to change the overall system specification.

This method allows component reused as each component is an abstract machine. The middleware is a good example of a component that could be used in several different applications.

A shortcoming of this approach is the need to fully model any shared functionality before decomposition can take place. However, the method does allow the composition of the abstract component models to be proved to be a correct refinement of the original abstract model.

With the decomposition method proven it can be used for modelling MAS in Event-B. Agents need to be able to coordinate their behaviour to be able to function as a MAS. One method for coordinating agent behaviour is to use interaction protocols to structure agent negotiation and decision making. The Contract Net protocol [6] is a decentralised protocol that allows agents to bid on a contract. This was used as a case study for modelling interaction in MAS.

The abstract specification modelled the initiation of the contract, the generation of proposals, the commitment between the agents and possible failures. The first refinement modelled the responses of agents to the initiation of the contract and the proposals. The second refinement separated the shared knowledge

about the contract and proposals to model the distribution of the system. The third refinement introduced the messages passed between agents to coordinate their knowledge. The fourth and fifth refinement developed the middleware into a generic model. At this point the functionality of the agents and middleware that support the interaction protocol was fully modelled and the decomposition method could be applied. This approach to modelling interaction protocols allows the interaction to be specified first and the internal behaviour of the individual agents is modelled separately. This simplifies the overall development process. There is also the possibility of reusing the models in other developments.

4 Conclusion

Event-B is ideal for the formal modelling of reactive systems. The use of abstraction and decomposition reduces the complexity of modelling MAS. Abstraction models the system as a whole in an understandable way. Decomposition allows the system to be broken down into abstractions of easily recognised agent-based components. Event-B provides an accessible and rigorous development method that is suitable for reactive systems. The progress described above is the first step towards an approach for the development of MAS using Event-B.

5 Future Work

Further work is needed in order to formally model MAS using Event-B. The modelling of interaction has led to the decomposition of the system into several components. The agents need to be modelled fully, as do their roles and other agent-level entities. The supporting platform infrastructure for MAS also needs to be investigated. The results of this research can then be used to produce a method for the practical formal development of MAS.

References

1. Jennings, N.R.: On agent-based software engineering. Artificial Intelligence **117** (2000) 277–296
2. Abrial, J.R.: The B-Book, Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996)
3. Métayer, C., Abrial, J.R., Voisin, L.: Rodin deliverable 3.2. Event-B language. Technical report, University of Newcastle upon Tyne, UK. Available From : http://rodin.cs.ncl.ac.uk/Delivarbles/D7.pdf (2005)
4. ClearSy: B4free home page. Available From : http://www.b4free.com/. Accessed 3rd September (2005)
5. Abrial, J.R., Cansell, D.: Click'n prove: Interactive proofs within set theory. In Basin, D., Wolff, B., eds.: Theorem Proving in Higher Order Logics, 16th International Conference, TPHOLs 2003, Springer-Verlag (2003) 1–24
6. FIPA: Fipa contract net interaction protocol specification. Available From : http://www.fipa.org/specs/fipa00029/sc00029h.pdf. Technical report, FIPA (2002)