Managing healthcare workflows in a multi-agent system environment

HILL, R., POLOVINA, S. <http://orcid.org/0000-0003-2961-6207> and BEER, M. <http://orcid.org/0000-0001-5368-6550>

Published version

HILL, R., POLOVINA, S. and BEER, M. (2005). Managing healthcare workflows in a multi-agent system environment. In: Proceedings of the Agents Applied in Healthcare Workshop, Nineteenth International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland.

Copyright and re-use policy

Managing Healthcare Workflows in a Multi-Agent System Environment

Richard Hill, Simon Polovina and Martin Beer
Web & Multi-Agents Research Group
Faculty of Arts, Computing, Engineering & Sciences
Sheffield Hallam University
Sheffield, United Kingdom
r.hill, s.polovina, m.beer@shu.ac.uk

Abstract

Whilst Multi-Agent System (MAS) architectures appear to offer a more flexible model for designers and developers of complex, collaborative information systems, implementing real-world business processes that can be delegated to autonomous agents is still a relatively difficult task. Although a range of agent tools and toolkits exist, there still remains the need to move the creation of models nearer to code generation, in order that the development path be more rigorous and repeatable. In particular, it is essential that complex organisational process workflows are captured and expressed in a way that MAS can successfully interpret. Using a complex social care system as an exemplar, we describe a technique whereby a business process is captured, expressed, verified and specified in a suitable format for a healthcare MAS.

1 Introduction

Multi-Agent Systems (MAS) are proving a popular approach for the representation of complex computer systems. The many emerging approaches and tools for Agent Oriented Software Engineering (AOSE) assist the generation of MAS models, enabling the translation of specifications to program code, but there still remains a gap between abstract initial requirements and MAS design specification. Abstract models are assembled and are iterated into a series of design models using the Unified Modelling Language (UML) or more recently, Agent-oriented UML (AUML) [Bauer *et al.*, 2000].

To prevent significant disparities between program code and the more abstract models, AOSE methodologies such as Gaia [Zambonelli *et al.*, 2003], Prometheus [Padgham and Winikoff, 2002], Zeus [Nwana *et al.*, 1999] and MaSE [DeLoach, 1999] have emerged and attempt to provide a unifying development framework. Except for Tropos [Bresciani *et al.*, 2004] however, little work has been published that encompasses the whole cycle from initial requirements capture through to implementation of MAS. Tropos attempts to facilitate the modelling of systems at the knowledge level and highlights the difficulties encountered by MAS developers, especially since notations such as UML force the conversion of knowledge concepts into program code representations [Bresciani *et al.*, 2004]. As a methodology Tropos seeks to capture and specify 'soft' and 'hard' goals during an 'Early Requirements' capture stage, in order that the Belief-Desire-Intention (BDI) architectural model [Georgeff *et al.*, 1999] of agent implementation can be subsequently supported. Whilst model-checking is provided through the vehicle of Formal Tropos [Fuxman *et al.*, 2001], this is an optional component and is not implicit within the MAS realisation process.

Extensions to the UML meta model such as AUML [Bauer *et al.*, 2000], have simplified the design and specification of agent characteristics such as interaction protocols, yet the process of gathering and specifying initial requirements is often limited to the discipline and experience of the MAS designer, using established notations such as UML's use case diagrams [OMG, 2005].

This paper therefore describes an improved MAS design framework that places a greater emphasis upon the initial requirements capture stage by supplementing the modelling process with Conceptual Graph notation [Sowa, 1984]. Section 2 describes the proposed process before an exemplar case study in the social care domain is explained in Section 3. A MAS model is built and checked to explicate the process in detail, illustrating the significance of the results. Finally the verified model is translated into the AUML design notation, permitting MAS code generation onto the platform of choice.

2 Modelling Process

In order to successfully capture system functions, an initial capture process should incorporate the following:

- A means of modelling the concepts in an abstract way, that facilitates the consideration of qualitative issues.
- An ability to reveal more system requirements to supplement the obvious actor-to-agent mappings.
- An explicit means of model-checking before detailed analysis and design specification.
- Improved support for capturing domain terms, with less reliance upon domain experts.

This approach enforces a rigour upon the requirements capture stage that is currently lacking from a range of AOSE design methodologies such as Gaia [Zambonelli *et al.*, 2003]

and Prometheus [Padgham and Winikoff, 2002]. The process is described in [Hill *et al.*, 2005a; 2005b; 2005c; 2004; Polovina *et al.*, 2004] and is as follows:

1. *Use Case Analysis* - Requirements are gathered initially and represented as use case models.
2. *Model Concepts* - The high level concepts are modelled and used to describe the overall scenario.
3. *Transform with Transaction Model and Generate Ontology of Types* - The high level model is transformed with the Transaction Model (TM)[Sowa, 1984], which ensures that a balance-check rigour is imposed upon the model, plus a rudimentary hierarchy of ontological terms is generated.
4. *Model Specific Scenarios* - Specific instances of the system are modelled.
5. *Inference with Queries and Validate* - The model is tested by inferencing queries to elicit rules for the ontology and refine representation.
6. *Translate to Design Specification* - The model is transformed into a design-level specification such as AUML.

3 A Community Healthcare Case Study

3.1 The Environment

Most older, frail or disabled older people prefer to receive care in their own homes [Beer *et al.*, 2001], in preference to the care provided within hospitals or residential home environments. UK National Health Service managers also recognise that hospitals can achieve a higher throughput rate of surgical cases if routine, rehabilitative care is delegated to the local community, removing it from the hospital ward. Home-based community care requires a variety of healthcare services to be delivered within the recipient's own home, allowing them to continue to live independently, and maintaining the best possible quality of life. Healthcare services include help with domestic cleaning, 'meals-on-wheels', medical treatment and assistance with basic bodily functions. Whilst the aims of community care address humanitarian issues, it is a challenging and expensive task to manage the organisation, logistics, quality assurance and efficiency of these services [Beer *et al.*, 2001].

The exponential rise in costs has, in recent years, led to UK Local Authorities creating policies that off load the delivery of care to private sector care providers. The community care scenario is thus an immensely complex and politically charged healthcare market place, freely trading care services in a competitive environment [Beer *et al.*, 2001]. It is apparent that the community care environment is comprised of a number of autonomous agencies, such as local authorities, different care providers, medical staff and the care recipients themselves.

As each element of care is often delivered by independent agencies, the number of autonomous command and control systems quickly increases, leaving the overall managers of the care (the Local Authority) to protect the individual bodies from disclosing sensitive and irrelevant information. Although information technology is established in community care management, it is clear that in many instances the vast quantities of disparate heterogeneous information repositories can lead to the undermining of effective system operations. It would seem that using collaborative intelligent agents, which overcome the difficulties of integrating disparate hardware and software platforms, queries can be mediated to the most appropriate information source, suggesting that agent technologies have the potential to build effective co-ordinated healthcare management systems. Groups of agents such as private care providers, routine care services, nursing and medical staff, which are managed by local authorities, private care managers, and health trust agents, exchange a vast number of messages within the community. As such the benefits of multi-agent systems, and examples of the types of improvements offered for community healthcare management, are documented by [Zambonelli *et al.*, 2003] and Beer et al. [Beer *et al.*, 2001].

Integration of all the disparate information repositories would assist the agents in their abilities to coordinate and control the logistics of home-based community healthcare. Current systems employ a 'needs-led' approach, documented using an Individual Care Plan (ICP), which is the key artefact of a package of care for each individual. The real-world implementation of the ICP can vary from paper-based forms through to electronic repository; however it is the management of the ICP process, together with the complexities of community healthcare logistics that offers substantial potential for efficiency improvements. Unfortunately the initial assessment and subsequent monitoring of ICP is too cumbersome for most Elderly Persons, resulting in a reduction in the quality of life and loss of personal dignity. This quality of life argument makes agent solutions more compelling, as it is common that the recipient's health condition deteriorates rapidly towards the end of their life.

Human agents regularly engage in transactions and Beer et al [Beer *et al.*, 2001] demonstrated the transactions involved when processing emergency alarm conditions in a community care environment (INCA - INtelligent Community Alarm). These apparently innocuous transactions proved much more difficult to deploy than first envisaged, though there were distinct advantages in favour of the agent approach. Upon raising the alarm, the message from the Elderly Person required brokering to determine the correct type of service, as well as locating the nearest, most available source of help. The multi-agent approach assisted the development of such a prototype demonstrator using the ZEUS Agent Building Toolkit [Nwana *et al.*, 1999], and it appeared relatively straightforward to 'map' actors to agents. However this work recognised that the payment transactions were rather more complicated and proved difficult to represent and implement faithfully.

It is feasible that an increased number of people currently receiving residential care could continue to live independently as part of their local community if the available technologies were utilised better to manage delivery of their care needs. It is also feasible that substantial efficiency improvements could be made if an architecture existed that successfully and robustly undertook the management of the myriad of transactions that exist between each party. It is important

to illustrate that excessive external monitoring and data capture is not a prerequisite of such a system. Each individual must receive a timely, flexible and efficient service, which is tailored effectively to meet the needs of the care recipient. Current financial and resource pressures mean that there is a requirement to actively monitor and direct care provision to where it is most needed, and delegate the task of managing payment transactions between the disparate agencies.

Since community healthcare includes an increased proportion of private sector health care services, together with the associated competitive marketplace, there is a much larger requirement for processing transactions between each of the separate agencies. Human agents in a commercial environment have long established transaction protocols, enabling the successful trading of goods and services. Multi-agent systems however are not as well developed, and it is essential that we establish a means of representing transactions in a realistic way, if we are to confidently delegate our transactions to autonomous agents. It follows that there is a need for a deployment framework that embodies the requirement for a robust transaction architecture, enabling the design and development of multi-agent systems that can be relied upon.

To further illustrate our approach, we shall describe the modelling of a multi-agent system for community healthcare management. We consider an established transaction model in relation to the multitude of payment transactions within the community care management environment, and propose a robust transaction-based framework for the deployment of agent-managed community care systems, that attempts to address the gulf between abstract concept and low-level, agent system implementation. The first stage is to examine the use cases within the system.

3.2 Use Case Analysis

The scenario is represented at the highest level with a use case model. Figure 1 illustrates the mappings evident between the use case model and the concepts expressed in the Conceptual Graph (CG) variant.

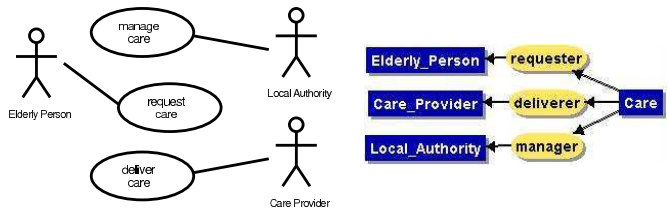

Figure 1: Use Case and Conceptual Graph models of healthcare scenario.

- *Elderly Person* - An infirm, elderly person that chooses to continue to live in their own home and request care support from the Local Authority.
- *Local Authority* - A localised representative body of the UK Government that manages and administrates the delivery of healthcare services.
- *Care Provider* - A private organisation that delivers care services into the Elderly Persons' home environment on behalf of the Local Authority.

3.3 Model Concepts

A illustrated in Figure 1, the healthcare scenario concepts are modelled with Conceptual Graphs (CG) [Hill *et al.*, 2004], [Polovina *et al.*, 2004].

3.4 Transform with Transaction Model and Generate Ontology of Types

As described in previous work [Hill *et al.*, 2005a], [Hill *et al.*, 2004], [Polovina *et al.*, 2004], the Transaction Model (TM) is a useful means of introducing model-checking to the requirements gathering process [Sowa, 1984]: pp 110-111. This capture of requirements at the outset ensures that the model-checking is not considered as an afterthought. Each process model is represented by debits and credits, in terms of the economic resources required for the transaction to take place. It follows that the models are incomplete until both sides of a transaction 'balance', and this has been shown to lucidly represent qualitative transactions such as 'quality of care received' [Polovina *et al.*, 2004]. For instance a simple healthcare transaction might be the payment of money (economic resource credit) for the delivery of a care package (incurring an economic resource debit). Such a transaction would be considered complete when the debit matches the credit and thus a balance is achieved. A more complicated transaction however might be the investment of time for training (debit), opposed by an improvement in the quality of care received (credit). The specialisation of the generic TM CG of Figure 2 onto the community healthcare scenario is illustrated by the CG in Figure 3. As Figure 4 shows, this specialisation serves two fundamental objectives:

1. The concepts identified within the care scenario are represented as a transaction thus 'economic events' and 'economic resources' are balanced;
2. Each concept is classified in terms of type, therefore a hierarchy of types, which is an important element of an ontology, is derived.

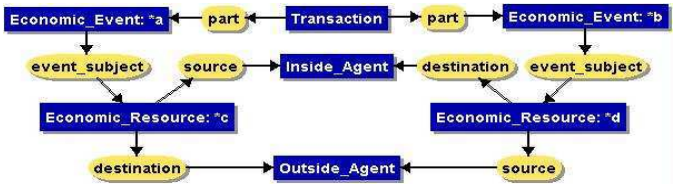

Figure 2: Transaction Model (TM)

It is not clear from the outset (Figure 1) which party pays the bill for the care, or who was the 'source' of the money. The UK Welfare System has three particular scenarios:

1. The Local Authority pays for the care in full.
2. The Elderly Person pays for the care in full.
3. The Local Authority and the Elderly Person make 'part payments' that amount to 100% of the care cost.

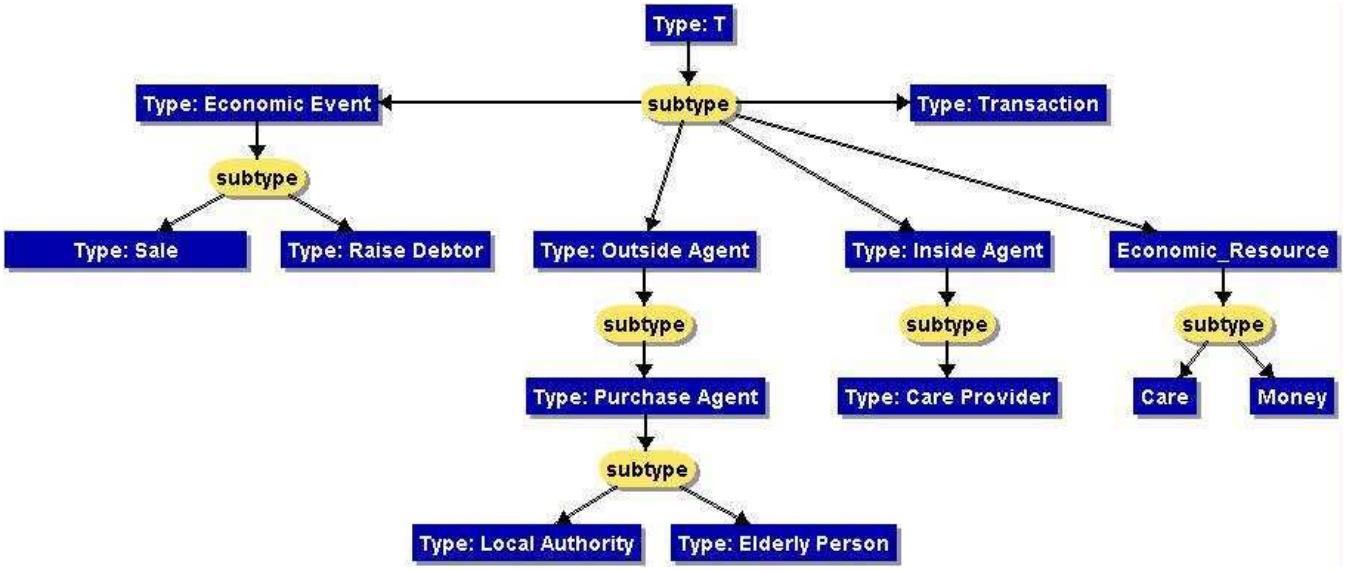

Figure 4: Type hierarchy after transformation with TM.

In order to satisfy the TM we therefore derive 'Purchase_Agent' as the supertype of 'Local_Authority' and 'Elderly_Person'. Determining terms for the ontology (Figure 4) is an important step during the agent realisation process. Whilst it is feasible to depend upon existing processes for the most part, the most significant contribution of this stage is the implicit 'balance check' that immediately raises the developer's awareness of the need for appropriate terminology. The type hierarchy in Figure 4 is deduced from Figure 3.

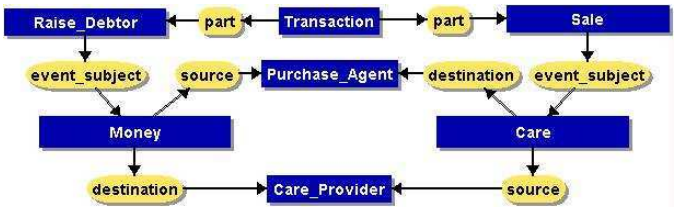

Figure 3: Healthcare scenario after application of TM.

3.5 Model Specific Scenarios

Once the generic model has been created, it is tested with some general rules. We first explore the specific scenario whereby an Elderly Person has been assessed and is deemed to be eligible to receive care at zero cost. In this particular case (highlighted in Figure 5), we see that the 'source' of the money to pay for the care is the Local Authority 'Sheffield City Council (SCC)', who also manage the provision of the care. The care package is not delivered by the Local Authority however; this is sold to them by private organisations, hence the need for a 'Care Provider', in our case, 'Meals on Wheels'. Since the Local Authority incurs the cost of the care package, that is its destination. Note that each concept in this figure now has a unique reference, denoting a specific instance. Conversely, the scenario exists where the Elderly

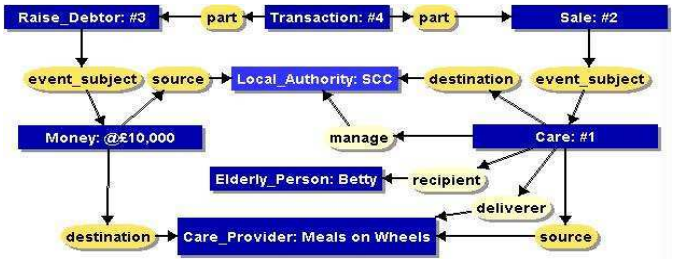

Figure 5: Local Authority (LA) pays for healthcare (Assets low).

Person is deemed to have sufficient monetary assets not to warrant a free package of care (Figure 6), where it can also be seen that the care package is still managed by the Local Authority.

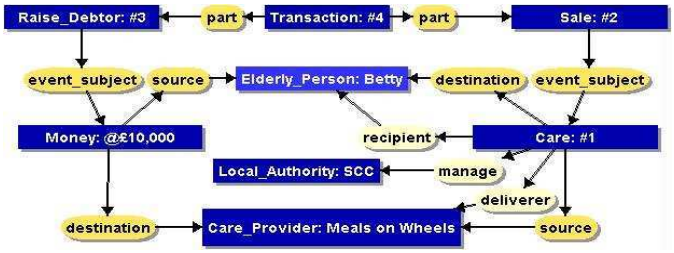

Figure 6: Elderly Person (EP) pays for healthcare (Assets NOT low.

3.6 Inference with Queries and Validate

From the preceding figures the general CG pattern (Figure 7) emerges. To evaluate this scenario we examine the case where the Elderly Person's 'Assets' are deemed to be less than

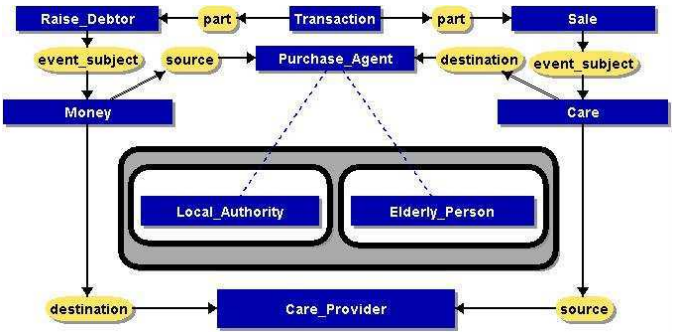

Figure 7: Emergent general CG pattern for this TM.

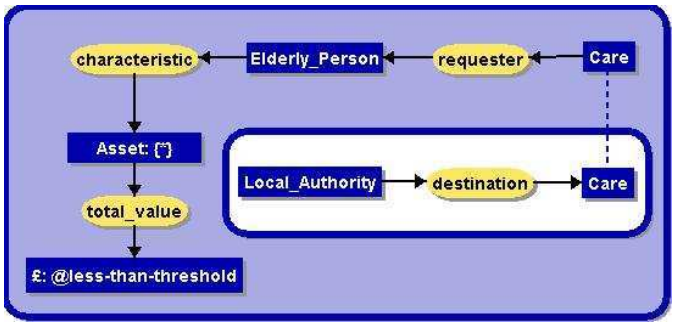

Figure 8: Elderly Person receives package of care at zero cost.

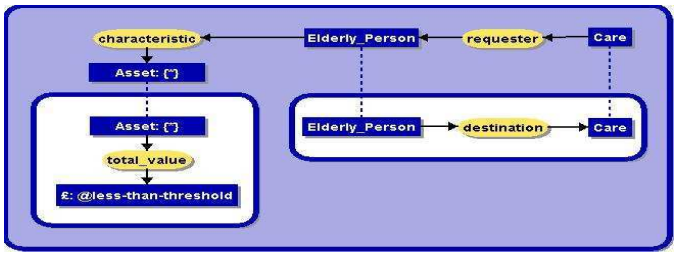

Figure 9: Elderly Person pays for care package in full.

a particular threshold set by the Local Authority, who would therefore be the destination of the care. Figure 8 shows this case. Figure 9 illustrates the alternate situation, depicted by 'less-than-threshold' asset test being set in a negative context. Here the Elderly Person would be the care and cost destination, as he or she is deemed to have assets that are not below the threshold. The part-payment model in Figure 10

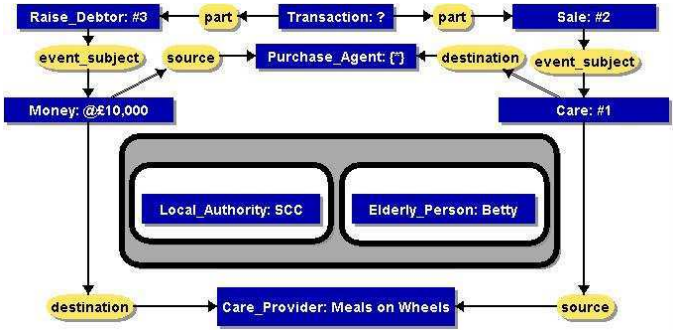

Figure 10: Incomplete TM illustrating two purchasing parties awaiting association with 'Purchase Agent'.

comprises Local_Authority and Elderly_Person, plus the Purchase_Agent derived earlier in Figure 9. However, Figure 10 does not allow joint parties to be the Purchase_Agent. Therefore we re-iterate the model further to support Figure 11. Here the Local_Authority and Elderly_Person have a split liability that is variable depending upon an individual's circumstances whilst ensuring that the total cost adds up to 100%.

The Elderly_Person and Local_Authority agents are no longer sub-types of the Purchase_Agent as originally illustrated, but are instead associated via 'liability' relations. Referring back to the hierarchy of types defined in Figure 4, we can now create a rule to supplant the ontology for the model. Figure 12 thus depicts an ontological component that is no longer valid, hence set in a negative context (or Peirce cut [Sowa, 1984]). Given the refinements discovered, the community care ontology is updated in Figure 13 and the TM in Figure 14 to show the liability relationship. The co-referent links are now valid thus the model can now be completed, enabling all three of the payment scenarios to be accommodated in one model.

3.7 Translate to Design Specification

Once the CG representations have been verified against the TM, it is then possible to perform a translation to a design specification. The 'inside' and 'outside' agents in the TM serve to provide direct mappings as follows:

Figure 11: Part-payment situation with shared liabilities.

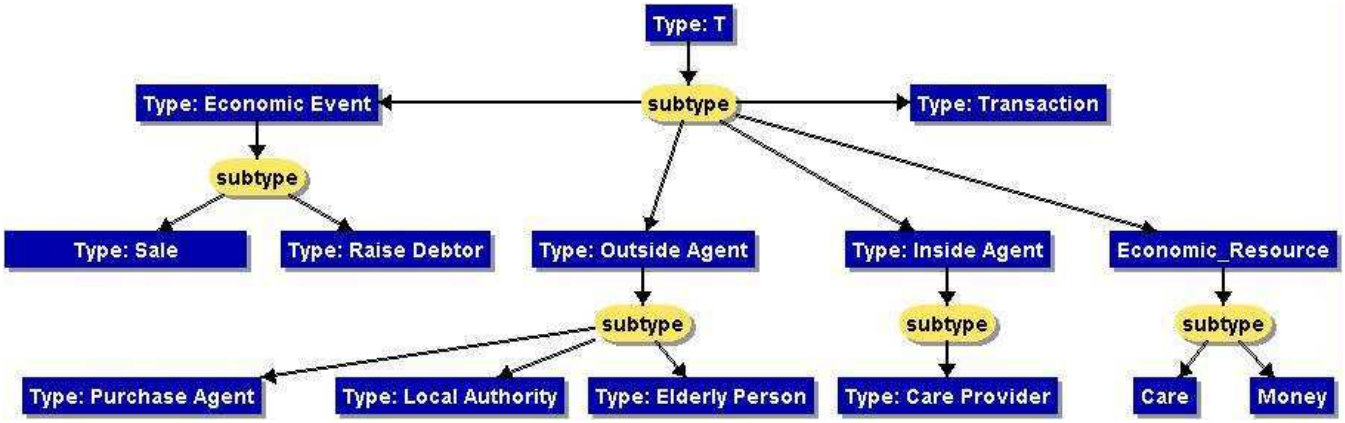

Figure 13: Updated Ontology.

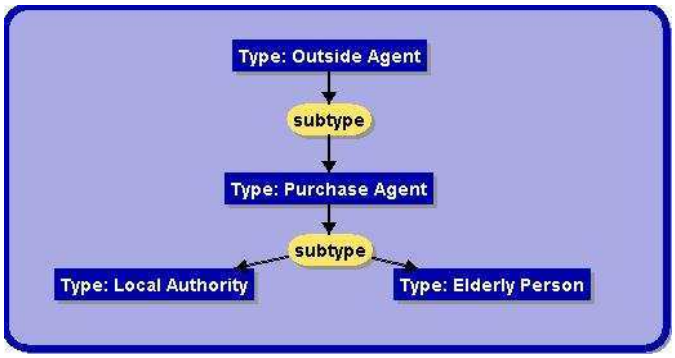

Figure 12: Ontological component that is no longer valid.

- Inside Agent: Purchase_Agent, with liabilities jointly satisfied by Local_Authority (SCC) and Elderly_Person ('Betty')
- Outside Agent: Care_Provider ('Meals_on_Wheels')

Further iterations and graph joins (omitted for brevity) would illustrate the following additional agents (where LA represents Local Authority):

- Care Request Agent:

  `[Elderly_Person]`
- Purchasing Agent:

  `[Local_Authority]->(sub-agent)->`

  `[LA_Procurement_Agent]`
- Care Assessor Agent:

  `[Local_Authority->(sub-agent)->`
  `[LA_Social_Worker]`
- Finance Agent:

  `[Local_Authority]->(sub-agent)->`
  `[LA_Finance_Assessor]`

From these direct translations we can construct agent bodies, to which specific sub-tasks can be assigned. Each of the behaviours is informed by the relations specified within the TM. For instance, referring back to Figure 1 the key abstract definition is that:

```
Management of Care is Local Authority.
```

Further analysis of the models results in the 'manage' role of the Local Authority Agent through its sub-agents identified above being decomposed into:

```
Assess_care_needs;
Confirm_financial_eligibility;
Procure_care_package;
Manage_care_delivery.
```

The process of revealing the agent behaviours is informed and contextualised by the business protocols that the TM has identified and the developer needs to apply across the many agent protocols. For instance, the 'Procure_care_package' behaviour can be represented by the FIPA Iterated Contract Net protocol [FIPA, 1999], thus devolving the task of obtaining the cheapest care package available to that protocol to which a given task may be best suited. This approach thus creates a situation whereby the method of requirements capture concentrates on what the MAS must deliver from the outset to implementation, assisting the developer in determining the extent to which the solution is influenced by the business model. Further refinement of the model with other methodologies is not precluded however, as the core transactional behaviours have now been established, verified and available for inclusion as needed.

4 Conclusions

Our approach has enabled the early elicitation of domain knowledge, and subsequent ontology specification, whilst incorporating a robust transaction model from the beginning. This has allowed representations of agent-managed healthcare workflows to be assembled at a much faster rate, especially since we have greater confidence that the underlying design is based upon a solid framework. This approach does not compromise further development with AUML, rather it ensures that the qualitative issues have been captured and considered prior to detailed system specification. The key features of this approach are as follows:

1. CGs represent the workflows in a more abstract way, and provide a foundation for modelling the knowledge exchange within a system. The abstraction is such that

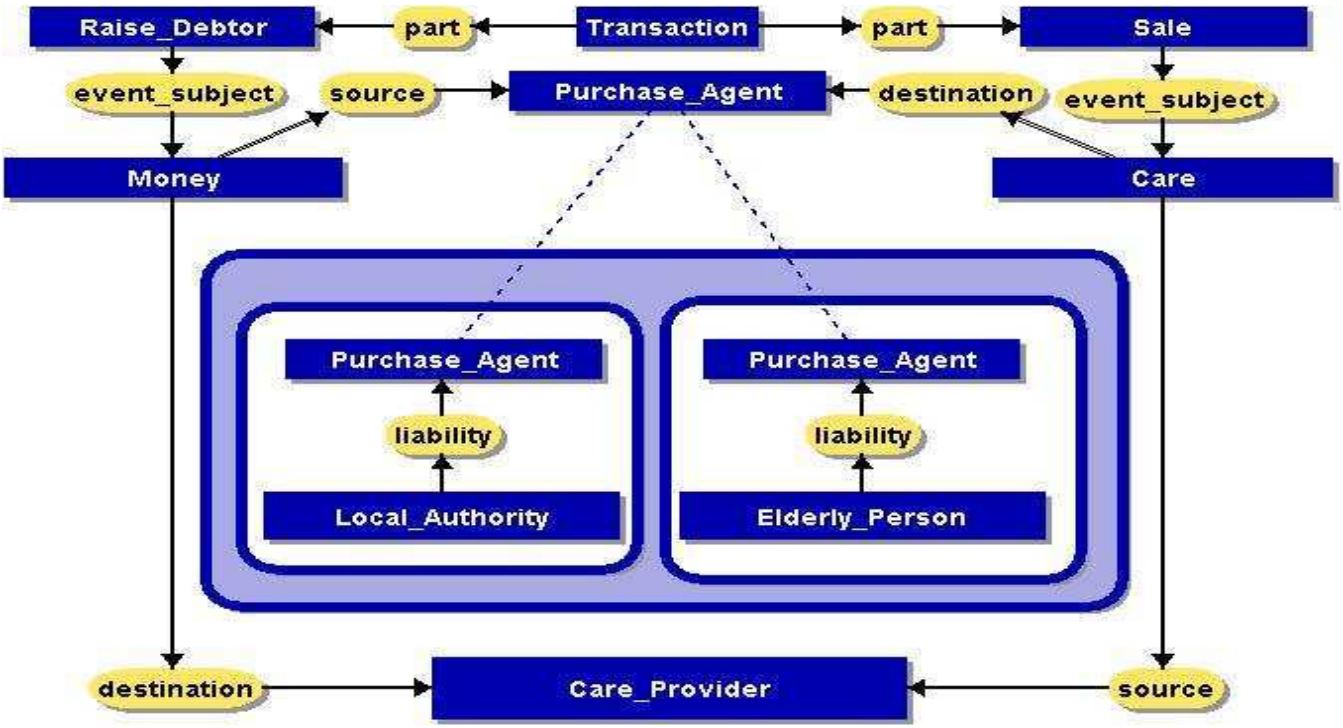

Figure 14: Refined model to accommodate part-payment scenario.

high-level, qualitative issues such as 'quality of health care received' are addressed, so it is feasible that the system is questioned from the point of view of concepts, rather than relying on an individual's prior experience.

2. CGs are similar to AUML in that there are some obvious mappings from concepts to agents, however there are also subtleties that CGs appear to reveal more consistently. Our experiences with AUML illustrated that actors translated to agents, though further analysis work suggested that the actors did in fact consist of several agents. More iterations were conducted until it was deemed that the tasks were distributed in an equitable way. The important point to note here is that the models are derived after considering issues at a much higher abstraction, thus resulting in a somewhat different view at micro-level.

3. Once the initial CG model has been produced, test scenarios can be evaluated using inferencing, and it is also possible to 'project' graphs onto other CGs. Using the transaction architecture, it becomes possible to force a set of rules (checks and balances) upon a model before it is represented in AUML.

4. Ontological terms are derived from the transaction model during the process of capturing requirements. Again, the inherent balance check of the model ensures that terms are agreed upon before the model is complete. This process ensures that debates about slot names are conducted sooner rather than later, having the immediate benefit of specifying more of the system detail before further model development.

5. The transactions approach makes model verification implicit as any missing nodes (concepts or relations) renders the model out of balance and thus unable to satisfy both sides of the transaction.

Using the design, specification and implementation of an agent-managed community healthcare system we have illustrated the development of a complex multi-agent system that embodies the notion of robust workflow management. AUML has illustrated how the enhanced richness of this notation can assist the expression and description of agent behaviours, interactions and architectures, especially when a multi-agent system designer needs to produce software specifications. We also recognise the limitations of AUML, particularly at the requirements gathering stage, where it is necessary to capture complicated, qualitative transactions that may not readily appear initially. Community healthcare management exposes a vast number of qualitative issues in the widest sense, and it is apparent that AUML has limitations when attempting to elicit these conceptual issues. Our experiences with CG illustrate that this notation appears to offer the multi-agent system designer a considerable advantage when it comes to assembling a specification of requirements that captures the essence of real-world representations, particularly when used in conjunction with AUML. It is also apparent that CG models lack the detail necessary to specify agent program code, unlike the comprehensive representations that can be expressed with AUML.

We believe that the combination of CG, AUML and an established transaction model is a first step towards providing

a unified framework for community healthcare information system deployment. We have demonstrated the lucidity of the CG approach, particularly with regard to the capture of concepts and the 'softer' aspects of a system. Subsequently, it is possible to derive the multi-agent interactions and behaviours required using AUML, assisting the specification of multi-agent systems that incorporate robust transaction management and real-world traits. It is of vital importance that such models are realistic if the potential benefits of multi-agent systems architectures are to be realised. Whilst multi-agent systems can appear an attractive solution to many complex domain problems, we have hitherto been hindered by a tool set that makes the description of realistic architectures difficult.

Increasing interest in agent and service-oriented architectures, together with the convergence of technologies in pursuit of the Semantic Web also illustrates the requirement for accurate and responsive descriptions of domain knowledge. The combination of a robust transaction model, with CG and AUML as representation vehicles, exemplifies an improved method for deriving ontologies for multi-agent managed systems, thereby overcoming the significant compromises that are often made when deploying agent solutions. Trading agent mechanisms for negotiation and payment are generally viewed as primitive, but our approach to modelling community care payment workflows for INCA is now permitting the development of robust, business-focused protocols. We believe that the use of CG at the requirements gathering stage enables the multi-agent system designer to overcome the difficulties of using AUML from the outset, by providing a robust means by which complex concepts can be modelled. It is now possible to exploit the enhanced functionality of INCA to support the continued application of multi-agent architectures that include robust transaction management as inherent agent behaviours.

5 Further Work

We have now established a route from abstract requirements gathering through to design specification that incorporates two significant events:

1. The high-level model is represented as CGs and then verified using a process of de-iteration and double negation upon the transactional model.

2. The resulting model is then translated into an AUML design specification.

We have now re-developed the INCA model using this approach and are currently deploying a demonstrator. The next step is exploring the use of this technique in other domain areas, in order to validate and enrich the approach further.

6 Acknowledgements

This project is in receipt of an AgentCities Deployment Grant from the European Union AgentCities.rtd Project (IST-2000-28385).

References

[Bauer *et al.*, 2000] B. Bauer, J. Muller, and J. Odell. Agent UML: A formalism for specifying multiagent interaction. *Agent-Oriented Software Engineering*, 1957:91–104, 2000.

[Beer *et al.*, 2001] Martin Beer, Iain Anderson, and Wei Huang. Using agents to build a practical implementation of the inca (intelligent community alarm) system. In *Proceedings of the Fifth International Conference on Autonomous agents*, pages 106–107. ACM Press, 2001.

[Bresciani *et al.*, 2004] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8:203–236, 2004.

[DeLoach, 1999] S. DeLoach. Multiagent systems engineering: A methodology and language for designing agent systems. 1999.

[FIPA, 1999] FIPA. Specification part 2 - agent communication language, 16th April 1999.

[Fuxman *et al.*, 2001] A. Fuxman, M. Pistore, J. Mylopoulas, and P. Traverso. Model checking early requirements specifications in tropos. In *Proceedings of the 9th IEEE International Requirments Engineering Conference*, Toronto, Canada, 2001. IEEE, IEEE.

[Georgeff *et al.*, 1999] Mike Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Mike Wooldridge. The belief-desire-intention model of agency. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10. Springer-Verlag: Heidelberg, Germany, 1999.

[Hill *et al.*, 2004] Richard Hill, Simon Polovina, and Martin D. Beer. Towards a deployment framework for agent-managed community healthcare transactions. In *The Second Workshop on Agents Applied in Health Care, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 13–21, Valencia, Spain, August 2004. ECCAI, IOS Press.

[Hill *et al.*, 2005a] Richard Hill, Simon Polovina, and Martin D. Beer. From concepts to agents: Towards a framework for multi-agent system modelling. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Utrecht University, Netherlands, July 30 2005. In press.

[Hill *et al.*, 2005b] Richard Hill, Simon Polovina, and Martin D. Beer. Improving aose with an enriched modelling framework. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Agent Oriented Software Engineering (AOSE) Workshop*, Utrecht University, Netherlands, July 30 2005. In press.

[Hill *et al.*, 2005c] Richard Hill, Simon Polovina, and Martin D. Beer. Managing community healthcare infor-

mation in a multi-agent system environment. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) - BIOMED Workshop*, Utrecht University, Netherlands, July 30 2005. In press.

[Nwana *et al.*, 1999] Hyacinth S Nwana, Divine T. Ndumu, Lyndon C. Lee, and Jaron C. Collis. Zeus: A toolkit and approach for building distributed multi-agent systems. *Applied Artificial Intelligence Journal*, 13(1):129–186, 1999.

[OMG, 2005] Object Management Group OMG. Uml resource page, 2005.

[Padgham and Winikoff, 2002] L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. In *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering*. AAMAS, 2002.

[Polovina *et al.*, 2004] Simon Polovina, Richard Hill, Paul Crowther, and Martin D. Beer. Multi-agent community design in the real, transactional world: A community care exemplar. In Heather Pfeiffer, Karl Erich Wolff, and Harry S. Delugach, editors, *Conceptual Structures at Work: Contributions to ICCS 2004 (12th International Conference on Conceptual Structures)*, pages 69–82. Shaker Verlag, 2004. ISBN 3-8322-2950-7, ISSN 0945-0807.

[Sowa, 1984] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Zambonelli *et al.*, 2003] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.