ShaMAN: An Agent Meta-model for Computer Games

Steve Goschnick, Sandrine Balbo, and Liz Sonenberg

Interaction Design Group, DIS, University of Melbourne, 3010, Australia
{stevenbg,sandrine,l.sonenberg}@unimelb.edu.au

Abstract. In this paper, we detail recent research on agent meta-models. In particular, we introduce a new agent meta-model called ShaMAN, created with a specific focus on computer game development using agent systems. ShaMAN was derived by applying the concept of Normalisation from Information Analysis, against a superset of agent meta-model concepts from the meta-models investigated. A number of features are identified, including human-agent *locales* and *socialworlds*, that might be usefully added to a generic AO meta-model.

Keywords: Agent-oriented, Agent Architecture, Multi-Agent Systems, Meta-model, Agent Meta-models, Agents in Computer Games, HCI.

1 Introduction

Agent-oriented (AO) architectures and methodologies are the main interest area of the research outlined here, with a focus on the application domain of computer games. While we are specifically interested in extending current AO concepts to further facilitate game specification and development, a consequence of this study identifies possible generic features to add to an AO meta-model.

1.1 Motivation

Computer games invariably have a graphic user interface (GUI) whether they are on PCs, dedicated game consoles or mobile phones. Additionally, many games are multi-user over either a proprietary network or the Internet, and as such, some data is often shared between multiple users. Neither graphic interfaces nor their associated event models, nor distributed data are well considered in the current AO architectures and frameworks, but computer games make heavy use of all three.

There is a precedent early on in the Object-oriented (OO) paradigm for an under-appreciation of these same facets of application programs, which ought to be instructive for the newer AO paradigm. At about the time that mainstream developers moved to OO languages, in particular C++ (early 1990's), GUI interfaces became the default in mainstream operating systems (OS). GUI and mouse/pointer interfaces made it necessary for application programmers to handle non-sequential *event-handling*, a significant change in programming practice from sequential processing in most character-based applications. Prior to more modern OO languages such as Java, both the GUI and event-handling was *not* a part of the language proper, e.g. C++. For

P. Forbrig and F. Paternò (Eds.): HCSE/TAMODIA 2008, LNCS 5247, pp. 237–245, 2008.

exam-ple, on the Unix OS events were handled via X-Windows and Motif *class libraries*. Thus, the application programmer in the early 1990s moved to an OO paradigm in language constructs, but their dealings with the GUI and event-driven programming, initially happened outside of the OO paradigm. So, an *event-driven paradigm* of pro-gramming happened concurrently by necessity, but it initially went unheralded in the shadow of the OO language paradigm.

1.2 A Gap in Agent Architectures

From the start AO has been socially-oriented such that *inter-agent communication* – a form of event - is typically allowed for with an Agent Communication Language (ACL). However, the AO paradigm has followed the initial OO programming languages, in not doing anything within the architecture or the constructs of the languages themselves, with regard to the GUI interface or non-agent event-handling.

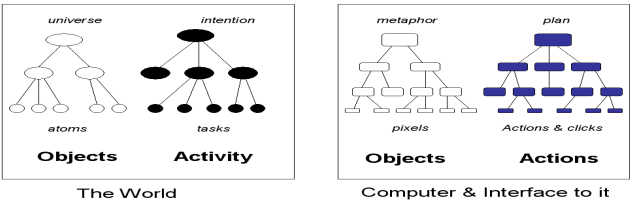

Fig. 1. Real world and the Object Action Interface Model

Interaction events and GUI interface objects are at the core of all mainstream computing platforms today, whether it be workstation, desktop, laptop, PDA or mobile phone. Figure 1 is an adaption of Shneiderman's Object Action Interface Model [20], showing a high-level representation of the physical world and what is done on a computer to supplement it, when a user interacts with an application program, via a computer screen and input mechanisms (e.g. mouse and keyboard).

The gap addressed by our research, is to achieve an AO architecture that engages with the user at the level of a GUI metaphor rendered down to the pixel level (left-hand side of right box in figure 1), with events down to the keyboard and mouse-click level, (right-hand side of figure 1). Our architecture is expressed as a meta-model.

1.3 Meta-models

Much of the research discussed here is centred around meta-models expressed in UML class diagram notation. Meta-models expressed in UML as such are now commonly used in both AO [1,12,13] and OO [17] research and development domains: to represent state-holding entities; to communicate base ideas; and as a useful means to compare different agent systems or architectures [6,12].

1.3.1 Agent Concepts

Given that there is currently no universally accepted single meta-model for AO systems, when we first looked to agent concepts and architectures with computer games

in mind, we examined the meta-models of several agent architectures and methodologies - AAII [16], GAIA [22,23], Tropos [1,11], TAO/MAS-ML [5], ROADMAP [13,14], ShadowBoard [8,9] - to explore the commonalities and differences between them. In addition, given our identification of a gap in the AO paradigm at the input device event level, we studied several well-known meta-models from the *Task Modelling* field, with its roots in the interaction between human users and computational devices, covered elsewhere [10].

1.3.2 Normalisation

A technique from Information Analysis (IA) used to improve ER models [2] that did not crossover into the later OO paradigm is the concept of *Data Normalisation* [15]. In this process derived from relational mathematics by Codd [3], the ER model is put into *normal form*. The model resulting from normalising a preliminary model, is considered to be in a state ideal for future change, and one that causes the least anomalies to operations upon the state held in the current entities. It is usually applied in IA to a model as a quality control procedure, however, Normalisation can also be used as a *bottom-up design technique* enabling the analyst to methodically deduce a well-formed model from a set of relevant concepts. In this research we applied it to a superset of the agent concepts found in agent and task meta-models, and arrived at a normalised agent meta-model named ShaMAN. From the perspective of a multi-agent system at *runtime*, a normalised meta-model is best for *insertion*, *update* and *deletion* of state information as it is happening in real-time.

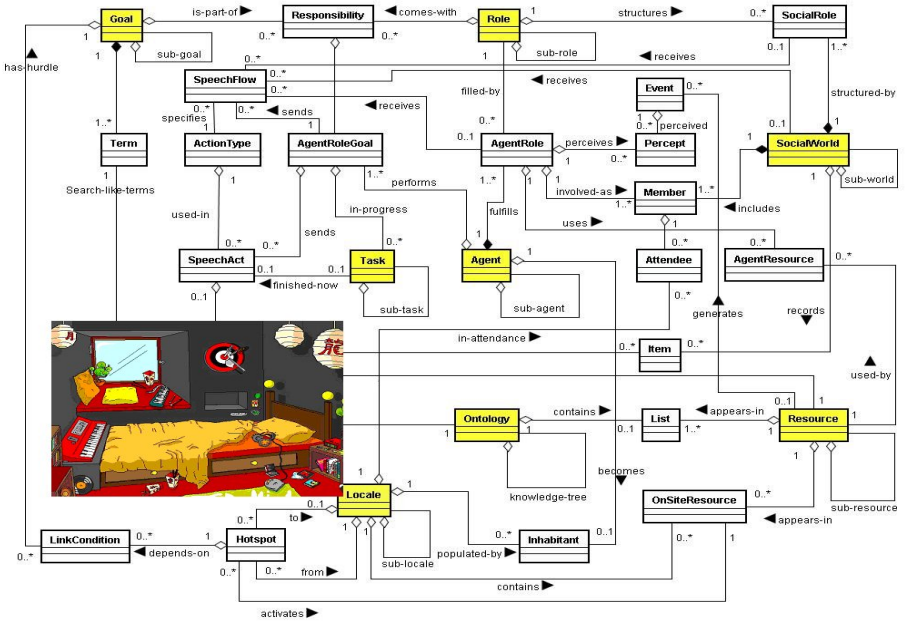

Fig. 2. The ShaMAN Agent Meta-model (with insert of a concrete game Locale)

1.3.3 Overview

In Section 2 we introduce the ShaMAN agent meta-model. To explain some of the entities in it, we present two groupings of the entities from the meta-model in detail, and then describe the flexibility it brings to building applications. In Section 3 we compare the concepts from the ShaMAN meta-model with those other agent meta-models investigated. In Section 4 we conclude and look to future work related to ShaMAN.

2 The ShaMAN Meta-model

We arrived at the ShaMAN meta-model depicted in figure 2 by taking concepts from a number of existing AO meta-models and a number of Task Analysis meta-models [10] – analysed them for similarities and differences, added some extra requirements from the games application genre, and then normalised the resultant set of entities. The following sections describe some aspects of the ShaMAN meta-model in more detail.

2.1 Locales for Computer Games

Computer games invariably interact with the player through the usage of a human-machine interface, for example a screen of one size or another. The Locale sub-section of ShaMAN lets us model the visual metaphors and the screen interaction between player/user and screen characters of a game, right in the AO model itself, rather than leaving it to some other paradigm such as OO. While some agent meta-models do have constructs for the agent *environment*, none of those investigated specifically model the computer screen as the primary representation of that environment.

In ShaMAN, this screen representation of an agent's environment is called a *Locale* - in homage to Fitzpatrick's [7] definition of a Locale as a generalised abstract representation of where members of a Social World [21] inhabit and interact. Figure 3 represents the sub-section of the ShaMAN meta-model that represents Locales within games.

A *Locale* entity may have sub-locales within hierarchies of Locales. Locale is a generic concept representing some spatial construct presentable on the screen, e.g. room, outdoor area, sections of a board-game - suitably broad enough for novel game interfaces.

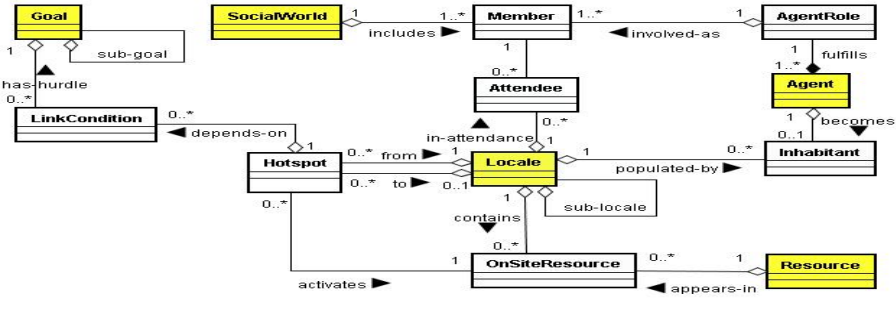

Fig. 3. The Locale sub-section of the meta-model

The insert in figure 2 is a concrete example of a Locale. It depicts the bedroom of a player's character within a game, which is represented as a Locale in ShaMAN. The *HotSpot* entity represents any area on the screen that is interactive, in the sense that whenever the user either clicks or passes over that area on the screen (or has *the focus*, from a keystroke point-of-view), certain interaction between the user and the game may take place. Whether the game presents a 2D or 3D scene, or an abstraction, the interaction with a standard display is 2D and involves *area*. The HotSpot entity has two relationships with Locale, one named *to* and the other named *from* – enabling navigation between Locales.

A HotSpot may also link to an *OnSiteResource* entity. These are *Resources* that live in the Resource entity (which may involve a hierarchy of Resources). Resources are typically programmed entities that are not Agent-oriented. E.g. clicking on the digital clock on the bedside table opens a window that displays a fully-functioning *clock* object, which is a Resource. OnSiteResource is an associate entity – a representation that allows the same Resource to be used in multiple Locales, e.g. a clock in many rooms drawing upon the same programmed code. Resource may also represent real objects in the real world, such as in a robot or a sensor application based upon the ShaMAN meta-model.

A HotSpot may also have a relationship with the entity *LinkCondition*, which in turn links to a *Goal* via a relationship called *has-hurdle*. This allows the game developer to enforce conditions to be met. Locale is also linked to the entities *Attendee* and *Inhabitant*. Attendee is an associative entity that records all occupants in a particular Locale over time, retaining a record of when agents (or human avatars) entered and left a Locale. It is linked to the agent's Role during that occupation via *AgentRole*, and also to the *SocialWorld* they were engaged in when they did so. This *history* aspect of the Attendee is usefull in providing and/or recording a back-story for any particular agent-oriented game character – a necessary aspect of realistic game creation.

2.2 The Goals, Roles, Responsibilities and Tasks of Agents

Computer games often have the need for intelligent, intentional, proactive and autonomous game characters that interact both with the human players and with other char-acters in a game. These properties are the harbingers of AO systems, and the sub-group of entities from ShaMAN meta-model in left and centre of figure 4,

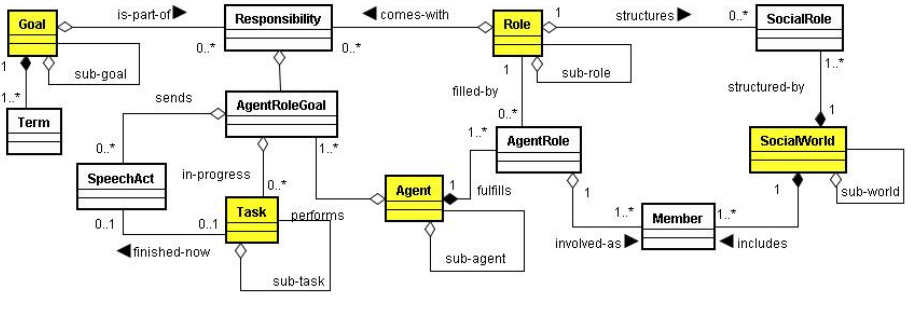

Fig. 4. Goals, Roles and Tasks in ShaMAN

represent the entities that appear most frequently (but not consistently) in one form or another, in many of the agent meta-models that we examined.

Figure 4 shows five entities in this sub-model of ShaMAN that have hierarchies of sub-elements of the same type, namely: *Goal, Role, Agent*, *Task* and *SocialWorld*. The associate entity between Goal and Role called *Responsibility* represents the responsibilities of a particular Role. A given Responsibility instance is fulfilled via an

Table 1. ShaMAN meta-model comparison with other agent architectures and meta-models

ShaMAN	KGR /BDI	GAIA V1	GAIA V2	RoadMap	Tropos	MAS-ML	DigitalFriend
AgentRole		Role	Role	(pointers)		Play	AgentRole
Percept							Percept
Event						Event	Event
SocialWorld (tree)	Acquaint-ance	System	Organisation, Pattern		Actor/ Social Agent	Organisa-tion	
SocialRole		System			Actor/ Position	Owner-ship	
Member							
Item							
AgentRole-Goal	Capability, Service	Services, Activity	Service, Activity	Services	Depend-ency	Actions	AgentRoleGoal
Task (tree)		Activity	Activity		Plan		Task
SpeechAct	Interaction	Activity				Message	SpeechAct
ActionType	Action, performa-tive						Action
SpeechFlow	Acquaint-ance, Permis-sions, Protocols	Protocol, Acquaintance model[5]	Protocol			Protocol	MessageFlow
Resource (tree)	Resource	Resource	Resource	Protocol (tree)	Resource	Object	Resource
Agent-Resource	Service	Permissions	Permissions		Depend-ency		AgentResource
Ontology (tree)				Knowledge-Component			Ontology (tree)
List	Knowledge						ResourcList
Locale (tree)			Environment	Environment-Zone		Environ-ment	
Attendee							
Inhabitant						Inhabit	
OnSite Resource							
HotSpot							
Link-Condition							
R,A,D,I,T,Rt	A,D,I	A,D	A,D	R,A,D,Rt	R,A,D,I	R,A,D	R,A,D,I,Rt

Note 1. R,A,D,I,T,Rt lifecycle phases: Requirements, Analysis, Design, Implementation, Testing, Run-time.
Note 2. The DigitalFriend V1 tool [9] is an implementation of the ShadowBoard agent architecture [8].
Note 3. AO models for Prometheus [18] and GoalNet [19] were in the study but not here, for space rea-sons.

instance in the *AgentRoleGoal* entity, by being enacted or performed by an Agent that takes on that Role. An Agent may have many Roles via *AgentRole*.

Goals will often have sub-goals in a hierarchy of goals to be achieved. One such sub-goal will be associated with a matching sub-role, and an agent will be assigned via an instance of the AgentRole entity. During execution of a ShaMAN application, sub-agents can be called upon in a downward direction via the need to achieve the sub-goals of parent goals, which is termed *goal-driven execution*. Or, they can be called upon from below, where a SpeechAct has been sent from further down the sub-agent chain, and the upper level goal has to be solved or rerun, termed *data-driven execution*. Data-driven execution often eventuates when a sub-agent retrieves new information from an external service such as a Web service, or from another agent across agent hierarchies or across Social Worlds.

2.3 Social Worlds in ShaMAN

Individual *Agents* can be members of one or more *SocialWorlds*. Their membership begins with an instance in the *Member* entity. Agents are related to the Member entity via the *AgentRole* entity. SocialWorld's have a number of SocialRoles, such as 'Captain' or 'Treasurer', which is useful in the design phase before specific agents are instantiated.

3 A Comparison of Agent Meta-models

Our motivation for collecting and comparing agent meta-models was for their agent concepts, as the primary input into a normalisation process, to arrive at a well-formed agent meta-model. Hence our initial interest in the comparison was analytic only.

Table 1 is a comparative format representing a sub-set of agent concepts that we used as input into the meta-model normalisation process in deriving the ShaMAN entities (the first table column). All models have some entity similar to the ShaMAN's *Goal, Role, Responsibility* and *Agent*, so these have been excluded from the table in this paper. Even so, a particular comparison (e.g. ShaMAN's *Goal(tree)* and Tropos's *Soft Goal/Hard Goal*) only *approximately* equates the concepts. Sometimes a comparison is close in meaning, other times it is close in name but distant in meaning, and sometimes there is wide variance in both name and the semantics. In the full study we did examine each twin comparison of concept in detail, but it cannot be presented in this paper for space reasons.

What is more useful in this paper is to highlight where ShaMAN has entities that have little or no comparison across the other agent meta-models examined. The darker shaded cells in the table shows that entities around *Locale* are unique to ShaMAN. Similarly, the lighter shaded cells show several of the entities related to *SocialWorld* are unique to ShaMAN. These entities were discussed explicitly above in the discussion of figures 3 and 4.

4 Conclusions and the Future

Agent-oriented architectures and frameworks lend themselves well to Human-Centred Software Engineering, given that several of them are derived from branches of

psychology and mentalistic notions (e.g. BDI – from Folk Psychology; ShadowBoard – from Analytical Psychology). We set out to extend current AO concepts to further facilitate game specification and development. While the entities unique to ShaMAN were introduced specifically for that purpose, most of them have a more generic usage, particularly for intelligent applications, with multiple users, many agents and rich user interfaces. It has not been our intent to develop a generic agent meta-model, however others are endeavouring to define an all-inclusive agent meta-model: Hahn et al [12] demonstrate the usefulness of the MDA (model driven architecture, see OMG [17]) approach to software development with AO tools. Fischer et al [6] propose that a *unified agent meta-model* is a worthy goal and could provide interoperability between many of the current disparate agent meta-models, methodologies and technology platforms. Theirs is a work-in-progress that we intend to align ShaMAN development with, as much as possible.

References

1. Bresciani, P., Perini, A., Giorgini, P., Guinchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, 203–236 (2004)
2. Chen, P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems 1, 9–36 (1976)
3. Codd, E.F.: A relational model of data for large shared data banks. Communications of the ACM 13, 377–387 (1970)
4. Cossentino, M.: Different perspectives in designing multi-agent systems. In: AgeS 2002, workshop at NodE 2002, Erfurt, Germany (2002)
5. Da Silva, V.T., De Lucena, C.J.P.: From a Conceptual Framework for Agents and Ob-jects to a Multi-Agent System Modeling Language. Autonomous Agents and Multi-Agent Systems 9, 145–189 (2004)
6. Fischer, K.: Agent-oriented software engineering: a model-driven approach. International Journal of Agent-Oriented Software Engineering 1(3/4), 334–369 (2007)
7. Fitzpatrick, G.: The Locales Framework: Understanding and Designing for Wicked Problems. Kluwer Academic Publications, London (2003)
8. Goschnick, S.B.: ShadowBoard: an Agent Architecture for enabling a sophisticated Digital Self. Thesis, Dept. of Computer Science, University of Melbourne, Australia (2001)
9. Goschnick, S.B.: The DigitalFriend: the First End-User Oriented Multi-Agent System. In: OSDC 2006, the third Open Source Developers' Conference, Melbourne, Australia, December 5-8 (2006)
10. Goschnick, S., Balbo, S., Sonenberg, L.: From Task to Agent-Oriented Meta-models, and Back Again. In: Tamodia 2008, Pisa, Italy (2008)
11. Guinchiglia, F., Mylopoulos, J., Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In: AAMAS 2002. ACM, New York (2002)
12. Hahn, C., Madrigal-Mora, C., Fischer, K., Elvesaeter, B., Berre, A., Zinnikus, I.: Meta-models, Models, and Model Transformations: Towards Interoperable Agents. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2006. LNCS (LNAI), vol. 4196, pp. 123–134. Springer, Heidelberg (2006)
13. Juan, T., Sterling, L.: The ROADMAP Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environments. In: 4th International Workshop on Agent Oriented Software Engineering, Melbourne (2003)

14. Juan, T., Pearce, A., Sterling, L.: ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In: Autonomous Agents and Multi-Agent Systems AAMAS 2002 (2002)
15. Kent, W.: A Simple Guide to Five Normal Forms in Relational Database Theory. Communications of the ACM 26(2), 120–125 (1983)
16. Kinny, D., Georgeff, M., Rao, A.: A Methodology and Modelling Technique for Systems of BDI Agents. In: Van de Velde, W., Perram, J.W. (eds.) The Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Springer, Berlin (1996)
17. OMG: MDA Guide Version 1.0.1 (2003), http//www.omg.org/docs/omg/03-06-01.pdf
18. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In: AOSE Workshop, AAMAS-20, Bologna, Italy (2002)
19. Shen, Z., Li, D., Miao, C., Gay, R.: Goal-oriented Methodology for Agent System Development. In: International Conference on Intelligent Agent Technology, IAT 2005 (2005)
20. Shneiderman, B.: Designing the User Interface, Strategies for Effective Human-Computer Interaction, 3rd edn. Addison-Wesley, USA (1997)
21. Strauss, A.: A Social World Perspective. Studies in Symbolic Interaction 1, 119–128 (1978)
22. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3, 285–312 (2000)
23. Zambonelli, F., Jennings, N.R., Wooldridge, M.J.: Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12, 417–470 (2003)