From Task to Agent-Oriented Meta-models, and Back Again

Steve Goschnick, Sandrine Balbo, and Liz Sonenberg

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

Abstract. In the research discussed here, in addition to extracting meta-models from numerous existing Agent architectures and frameworks, we looked at several Task meta-models, with the aim of creating a more comprehensive Agent meta-model with respect to the analysis, design and development of computer games. From the agent-oriented perspective gained by examining the resultant extensive agent meta-model – named ShaMAN – we then revisit the Task Analysis research domain, and consider what benefits Task Analysis and Modelling may draw from the Agent-oriented paradigm.

Keywords: Agent-oriented, Task Models, Multi-Agent Systems, Meta-model, Agent Meta-models, Task Meta-models, Software Engineering, Computer game development, Agents in computer games.

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. In addition to extracting meta-models from numerous existing Agent architectures and frameworks (not covered in this paper), we looked at several Task meta-models, all with the aim of creating a more comprehensive Agent meta-model with respect to the analysis, design and development of computer games. From the agent-oriented perspective gained by examining the resultant extensive agent meta-model – named ShaMAN – we revisit the Task Analysis research domain, and consider what benefits Task Analysis and Modelling may draw from the Agent-oriented paradigm.

1.1 Motivation for Task Models in AO Meta-model Research

A modern sophisticated computer game can be characterised as a mixed-initiative multi-agent system – meaning that interaction happens between the human users/players, and various game-based synthetic characters, which have a high degree of proactive autonomous behaviour. In addition to being multi-agent in nature, such games also involve multiple users, playing alone or in guilds (teams). AO researchers have predominantly been intent on putting intelligence into artefacts (e.g. trading systems, robots, simulations, etc.), with only a small percentage concerned with mixed-initiative human-agent systems [9,11,12], such as computer games.

P. Forbrig and F. Paternò (Eds.): HCSE/TAMODIA 2008, LNCS 5247, pp. 41–57, 2008.

The specification of goals in agent systems usually begins in analysis with the identification of high-level, motivational goals. Lower level tasks and actions become a focus of endeavour in the AO SDLC when an agent system is being implemented.

Task Analysis (TA) began with a keystroke-level interest in the things that humans do with technology. While the purpose of TA is to understand the user's activity in the context of the whole human-machine system for either an existing or a future system [6], a task model helps the analyst observe, think, record and communicate the user's task activity [18].

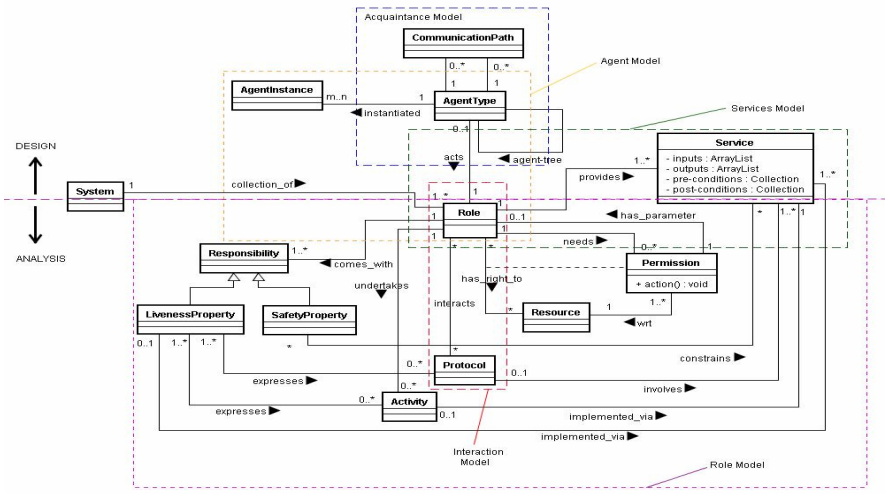

Fig. 1. Meta-model of GAIA V1 in UML, with models superimposed

In a strong sense TA and AO began at the opposite ends of the design spectrum with very different agendas, one bottom-up and people-oriented, the other top-down AI-oriented. While there are several task meta-models that draw from a cognitive top-down perspective, several of which we draw upon in this research – e.g. HTA, TWO, and TKS – there are few agent meta-models that include activities down at the user keystroke level. It seems intuitively obvious that TA ought to have lessons for AO, particularly in mixed-initiate AO systems, including computer games.

1.2 Meta-models

Much of the research discussed here is centred around meta-models expressed in either: UML class diagram notation, or Entity Relation (ER) notation [3], the adoption of which requires a little explanation up front.

Modelling in a Visual Notation, Comparing Meta-models in UML. In agent-oriented research and in computing in general, modelling expressed in some form of visual representation is used to communicate, build and document complex systems and artifacts. In complex systems no single diagram type can most effectively express and communicate all of the concepts and ideas encountered – a particular notation is

designed to best encapsulate one or two aspects of the complexity. The UML language for example has no fewer than 13 diagram types from which to choose a representation that best suits a particular aspect and phase of an object-oriented (OO) system or application. Meta-models expressed in UML class diagrams are now commonly used in both the AO [2,13,14] and OO [20] research domains: to represent state-holding entities; to communicate base ideas; and as a useful means to compare different agent systems or architectures [7,14]. To facilitate the process of comparison, it is useful to represent as many aspects of an architecture as possible in a single meta-model diagram.

In the well-known GAIA agent-oriented architecture [28,29] there are distinct models for: Roles, Interaction, Agent, Service and Acquaintances – several with unique notations. However, figure 1 is a single condensed meta-model of GAIA – the dotted lines superimposed over the entities, representing the Roles, Interaction, Agent, Service and Acquaintances models, each a subset of the meta-model.

Agent Concepts. Given that there is no universally accepted single meta-model for AO systems at present, when we first looked to agent concepts and architectures with computer games in mind, we examined the meta-models of several well-known agent architectures and methodologies (AAII [19], GAIA [28,29], Tropos [2,13], TAO/MAS-ML [5], Prometheus [21]) and several that are less well known (ROADMAP [16], ShadowBoard [10,11], GoalNet [22]), to explore the commonalities and differences between them. In addition, given our identification of a gap in the AO paradigm down at the input device event level, we included in our study several well-known meta-models from the Task Modelling field. For space reasons this paper discusses the Task meta-models, but not the agent meta-models.

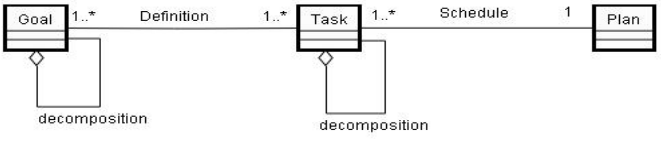

Fig. 2. HTA Task Meta-model in UML notation

Cognitive Task Meta-models Examined. According to Annett [1], HTA is best regarded as "a generic approach to the investigation of problems of human performance within complex, goal-directed control systems". HTA focused on system goals and plans, where other approaches at the time focused on observable aspects of performance. In contrast to a behaviouralist view, the cognitive approach taken by HTA envisaged human behaviour as goal-directed and controlled by feedback. HTA Tasks describe lists of actions needed to achieve a particular goal state. Goals may have subgoals in a hierarchical fashion. Tasks represent specific user actions in a goal-directed activity. They can be hierarchical with sub-tasks linked to sub-goals (see decomposition relationships in figure 2). Plans are like recipes to achieve a task, controlling sequence and timing.

The Task World Ontology (TWO) model from Groupware Task Analysis (GTA) [26,27], is a generic task analysis model that includes as its primary entities: Event,

Task, Goal, Agent, Role and Object, all of which except Object are like-named entities of primary interest to most Agent meta-modellers. In addition, TWO includes several many-to-many relationships, including Responsible and Plays.

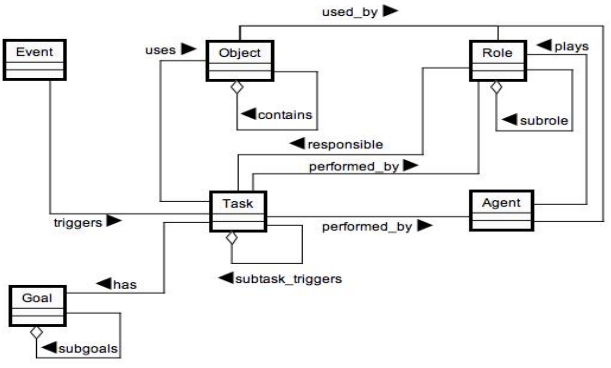

Fig. 3. TWO Task World Ontology model in UML notation (adapted from [27])

The GTA is of specific interest to us, as it stemmed from task analysis in the Computer Supported Cooperative Work (CSCW) field, thereby involving multiple users and complex contexts.

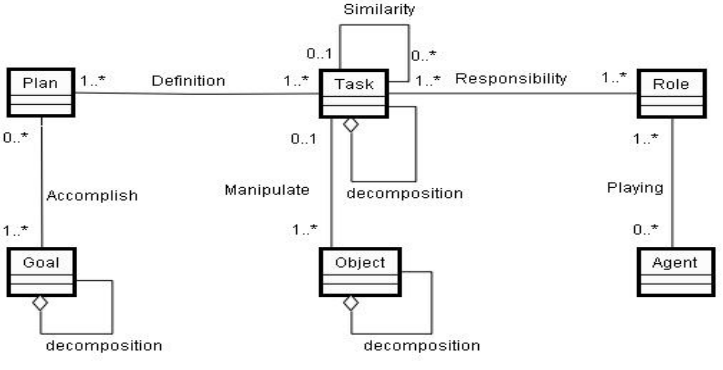

Fig. 4. The Task Knowledge Structure (TKS) Meta-model in UML

Johnson and Johnson [15] developed the Task Knowledge Structure (TKS) model to support the design phase of the SDLC. It details task hierarchies, objects hierarchies (the decomposition relationships in figure 4 represent hierarchies – e.g. subtasks of a task, recursively) and knowledge structures. It aimed to represent the knowledge a user has stored in their head regarding tasks. The user is represented by an Agent in the model. The Role/s that a user takes on, each have Responsibilities that are manifested in a set of Tasks to be done.

Additionally, we looked at GOMS [18] and the DIANE+ [25] task modelling language, which will not be introduced here for brevity sake.

Overview. In Section 2 we introduce the ShaMAN agent meta-model proper. In order to explain the entities in figure 5 efficiently, we present several groupings of the entities from the meta-model in detail, and then describe the flexibility they bring to building applications using the model. In Section 3 we compare some concepts from the ShaMAN meta-model with the Task meta-models investigated. In Section 4 we highlight some challenges that others have raised for Task Modelling. In Section 5 we describe how Task modeling may be enriched by AO concepts, addressing some of those challenges highlighted in Section 4.

2 The ShaMAN Meta-model

At a fundamental level, an agent meta-model is an entity model, thereby abiding by all the conventions of a notation that an application model adheres to. In this research, in a bottom-up design manner, we applied normalisation [17] to a superset of the agent concepts found in the agent meta-models (AAII, Gaia, Roadmap, GoalNet, TAO, Tropos, ShadowBoard and Prometheus), in several task meta-models (HTA, TKS, GOMS and the Task World Ontology (TWO) from Groupware Task Analysis (GTA) together with a few concepts needed by computer games, and arrived at a normalised agent meta-model named ShaMAN, presented in figure 5. (Nb. The normalisation process is not documented here for space reasons).

As a normalised model ShaMAN is both: flexible to ongoing requirements upon the meta-model itself (should it need future enhancements); and is a model least susceptible to anomalies arising from changes of state with respect to the existing concepts represented in it. The following sections describe the main sub-sections of the ShaMAN meta-model in more detail.

2.1 Locales for Computer Games

The domain of applications of interest to us in this research is computer games, which 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 the ShaMAN model 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. It is an abstraction that is suitable for games and other rich media applications, without modelling specific widgets in a UI library. While several of the agent meta-models investigated do have constructs for the agent environment, none specifically model the computer screen as the primary representation of the environment to the user.

In ShaMAN, this screen representation of a sub-section of the agent's environment is called a Locale in homage to Fitzpatrick's [8] definition of a Locale as a generalised abstract representation of where members of a Social World [23] inhabit and interact. Figure 6 represents that sub-section of the ShaMAN meta-model that represents Locales within games. Note: the simplified crows-foot (zero, one or many) ER notation for cardinality, is used in these detailed figures of sub-sections of the meta-model, to increase the general readability.

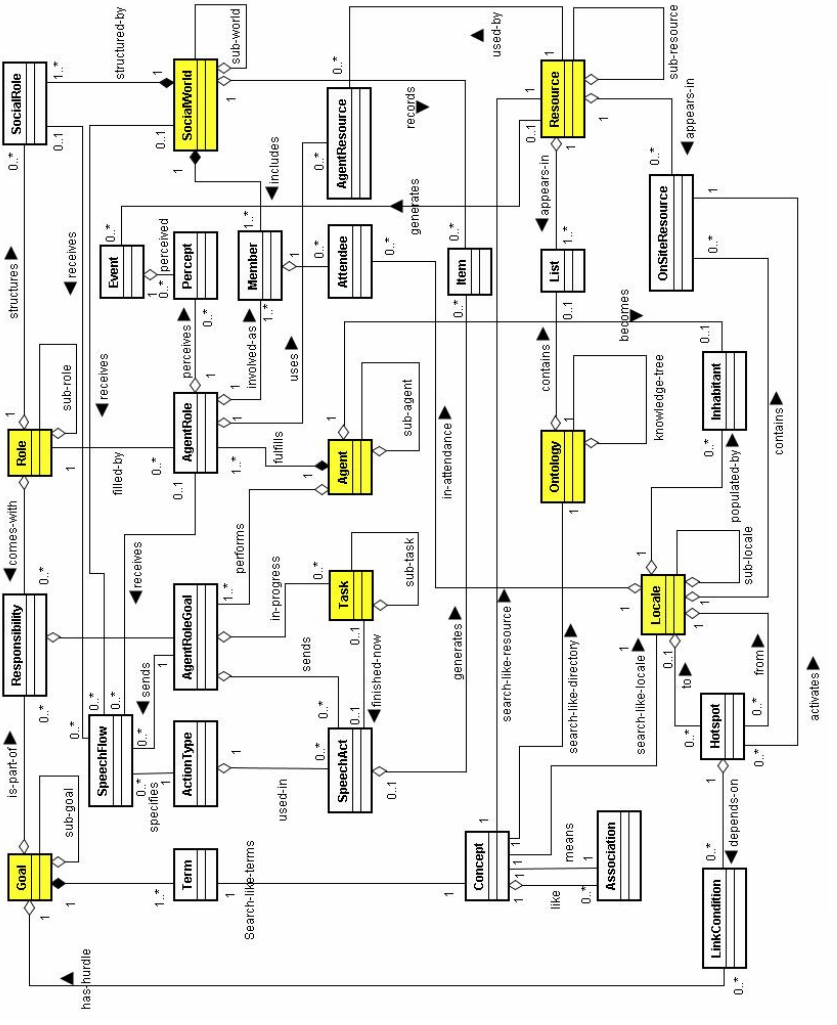

Fig. 5. The ShaMAN Agent Meta-model

The Locale entity may have sub-locales thereby allowing hierarchies of Locales. Locales are a generic concept representing some spatial construct presentable on the screen, e.g. rooms, outdoor areas, sections of a board-game, etc. - suitably broad enough for novel game interfaces. By way of describing the entities in figure 6, we will refer to the game screen in figure 7 as a concrete example of Locale.

The screen depicts the office of the player's character within the game BranchOut, and that room 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 if a HotSpot has the focus, from a keystroke point-of-view), certain interaction between the user and the game may take place – for example clicking on the filing cabinet draw opens a window that displays the contents of the draw. HotSpots are a generic concept for such screens, whether the game presents a 2D, 3D or something abstract, the interaction with a standard display is 2D and involves area. HotSpot has two relationships with Locale, one named to and the other named from – enabling navigation between Locales.

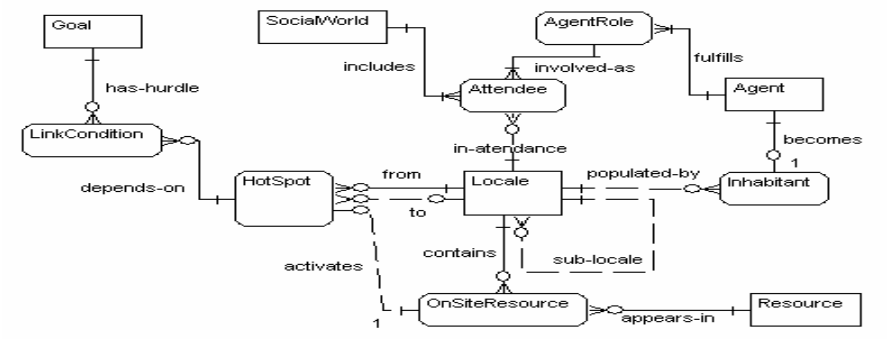

Fig. 6. The Locale sub-section of the meta-model

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 Agents in their conception nor development, but they may also represent real objects in the real world. The digital clock on the wall in figure 7 is a fully-functioning clock object, and when clicked-on, displays a fuller interface to the digital clock 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.

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: e.g. before the player may advance to another Locale, or before they may use a particular Resource.

Locale is also directly linked to two other 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 the Locale and when they left it, if they are no longer present. It is linked to the agent's Role during that

Fig. 7. Depiction of a Locale in a Computer Game being developed using ShaMAN

occupation via AgentRole, and also to the SocialWorld they were engaged in when they did so. This history aspect of the Attendee is usefully in providing and recording a back-story for any particular agent-oriented game character – a necessary aspect of realistic game creation. In contrast to Attendee, the Inhabitant entity represents the current occupants of the Locale. It has a direct link to the Agent entity, and is used when the overhead of SocialWorlds and agent Roles are not a concern.

2.2 Communication Via SpeechFlow

Some computer games have large numbers of agents and very often these agents are categorised by the roles they serve. When it comes to communication between agents, there is the need for communication at several levels: one-to-one; one-to-a-group of agents in a particular Social World; one to all agents filling a specific role (e.g. Captain). Note: Human players (human-in-the-loop) are considered to be agents, from the system's point of view. The SpeechFlow sub-section of the ShaMAN meta-model as portrayed in figure 8, addresses these three required levels of communication in gameplay. In addition, it addresses Events from non-agent Resources.

The SpeechFlow entity is at the heart of all interaction within ShaMAN – including events generated by Resources. An agent communicates to other agents or to humans-in-the-loop by generating an instance of SpeechFlow. It does so while acting in some Role, working upon some Goal associated with that Role. The entity that represents the instance of those two things for a particular agent is AgentRoleGoal. While the AgentRoleGoal instance may generate many SpeechFlow instances, each one of them is linked to just one ActionType, which are in turn determined by a particular agent communication language (ACL).

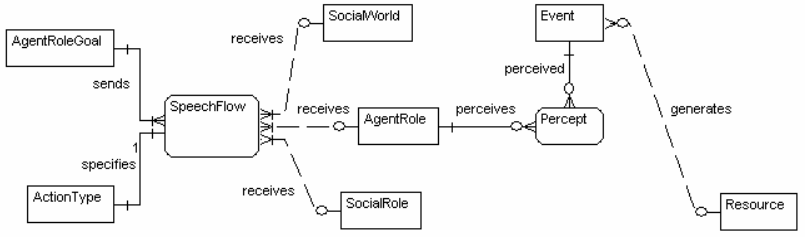

Fig. 8. Message flow within ShaMAN

While a particular instance of AgentRoleGoal is the source of any given instance of SpeechFlow it can have one or more of three possible receivers: AgentRole is a given individual recipient agent while acting in a particular role; SocialWorld means that the message will be broadcast to the whole membership of a social world however many members it has (via Member – see figures 5 or 10); or, SocialRole which is a particular role common to many SocialWorlds, such as 'Treasurer', or, a game example such as 'Captain' – e.g. All 'Captains' of SocialWorld's of type 'Ship'. The three receives relationships are all zero-or-one to many, because they may be either alternative or parallel receivers of a specific message.

Non-agent Events come from Resources (including from UI widgets), and are perceived by Agents through AgentRole as Percepts.

2.3 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 characters in a game. These properties are the harbingers of AO systems, and the subgroup of entities from ShaMAN meta-model in figure 9, 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 9 shows four entities in this sub-model of ShaMAN that have hierarchies of sub-elements of the same type, namely: Goal, Role, Agent and Task. The associate entity between Goal and Role called Responsibility represents the responsibilities of a particular Role. A given Responsibility instance is fulfilled via an instance in the AgentRoleGoal entity, by being enacted or performed by an Agent that takes on that Role. An Agent may have many Roles and the AgentRole entity represents this multiplicity.

In task modeling, a hierarchy of related tasks is performed to achieve a goal that sits at the head (or root) of a task hierarchy. In ShaMAN the AgentRoleGoal entity represents such a root Goal (via Responsibility), while such a task hierarchy is represented by Task together with the self-relationship (a unary relationship) called sub-task.

The completion of a Task may spark a SpeechAct. SpeechActs (zero, one or many of them) may also be generated directly by the AgentRoleGoal entity. Note that SpeechAct is related to the ActionType and SpeechFlow entities that were depicted and described earlier in Section 2.2.

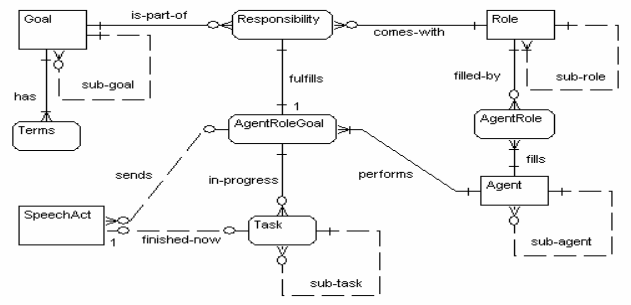

Fig. 9. Goals, Roles and Tasks in ShaMAN

Goals within ShaMAN are expressed as GoalX(term1, term2, … termN), like a predicate in the predicate logic sense. I.e. By setting one or more of the terms to a constant, and leaving one or more of the terms as variables, a logic language such as Prolog will accept such a predicate as a goal, and will set about solving it, without a formal plan. Even though this predicate format for goals conforms with logic languages, it is also an acceptable way to specify methods capable of achieving goals in imperative languages. Likewise, it is an equally acceptable format for expressing RPC-oriented (Remote Procedure Call) web services (WS) and furthermore, database queries in a Query-by-Example format.

In ShaMAN a Term can be a simple variable (or a literal/constant) but it need not be – it may also include constraints. E.g. While the variable Temperature is an acceptable term within a ShaMAN goal, the constraint: during(12 < Temperature < 100) is also acceptable as a term. A term expressed as such is an invariant constraint, meaning that it remains in force for the life of the goal to which it is associated. A second type of constraint can be expressed in a ShaMAN term as follows: before(12 < Temperature < 100) – which means that the constraint must hold before the goal can be begin to be solved. An example of a third form of constraint definition is after(Tempature = 100), representing a constraint that exists following a successful completion of the goal in question. This use of keywords before, during and after within constraints as temporal directives, was proposed by Goschnick & Sterling [12].

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. I.e. Even though the groupings of agents within ShaMAN are generally organized in hierarchies, communication and hence cooperation can happen between agents across different agent/sub-agent hierarchies.

2.4 Social Worlds in ShaMAN

Individual Agents can be members of one or more SocialWorlds. Their membership begins with an instance in the Member entity. See figure 10 below. Agents are related to the Member entity via the one-to-many relationship involved-as between their entry in the AgentRole entity and Member, while a SocialWorld includes multiple agents represented in Member via a one-to-many relationship.

Agents can fill multiple Roles via the AgentRole entity. The Roles that are available within a particular SocialWorld are listed as instances of the SocialRole entity, which sits between Role and SocialWorld. SocialRole is a useful entity in a number of ways: it can be used to specify all the roles that make up a SocialWorld in the design phase, before any Agents become members; and, as we saw in Section 2.2, at execution time SpeechFlow messages can simply be broadcast to all agents in a SocialWorld that occupy a particular SocialRole such as 'Captain'.

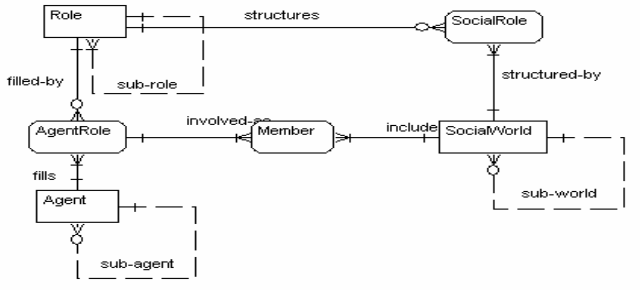

Fig. 10. SocialWorlds within ShaMAN

2.5 Knowledge Tree and Resources

The Knowledge Tree part of ShaMAN (see figure 11) consists of a hierarchy of concepts in the form of an ontology (the entity is called Ontology), but which is then related to lists of Resources (via the ResourceList entity) at each level in the ontology. In the functioning beta instantiated system based on ShaMAN, the Ontology is represented as a file directory structure and the resource list includes a multitude of file types, including image and video files used in a game, through to executable Java objects, such as the Filing Cabinet displayed in figure 7 above and discussed in Section 2.1. It differs from a conventional file directory structure in that any Resource can appear in multiple ResourceLists via the appears-in relationship between them.

The Resource hierarchy can store any group of objects or resources that are naturally composed in a hierarchical form. For example, a computer consists of motherboard, hard drive/s, memory cards, etc. – which could be logically represented in the Resource tree of ShaMAN. Similarly, a standard GUI screen is a hierarchy of on-screen components/widgets, that can be represented as a part of a Resource hierarchy used by a Locale that represents that screen, and linked by the OnSiteResource entity depicted at the bottom of figure 6.

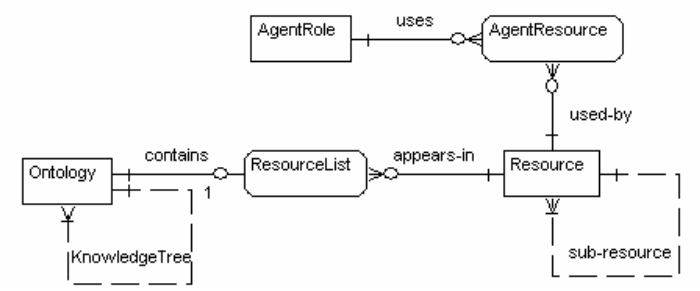

Fig. 11. The ShaMAN Knowledge Tree

3 A Comparison of ShaMAN with Task 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. So our initial interest in the comparison was analytic only.

Table 1 below is a comparative format representing the original set of agent concepts from the Task meta-models that were used as input into the meta-model normalisation process in deriving ShaMAN (see first table column). Nb. There is a similar comparitive table of ShaMAN concepts against the agent meta-models used as input, but it is not in this paper for space reasons.

While a particular comparison in the table, such as ShaMAN's Goal(tree) and TWO's Goal(tree) approximately equates the concepts, the comparison is a gross simplification. A TWO Goal is different from a ShaMAN Goal, in that a TWO Task has a Goal as an attribute, while a ShaMAN Goal leads to a number of Tasks. Sometimes the table 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. However, such a table is useful to begin the cross-model discussion.

What is common to ShaMAN and all the Task models examined is that task represents a unit of work. An analogy that we will use here, is a model-neutral concept - the programmed Method (Procedure) within an imperative language. We will distinguish between the Method Signature (its name plus parameters passed), the Method Body of the method (the logic that can resolve the goal), and an Executing Method. An Executing Method represents a task underway. In ShaMAN a method signature is the equivalent to a Goal, where the terms can equate to parameters. As mentioned earlier, in Section 2.3 this predicate format for goals is not only an acceptable way to specify method signatures capable of achieving goals in imperative languages, it is equally acceptable for expressing RPC web services (WS), certain database queries and other languages.

HTA does not attempt to represent cognitive systems (as BDI, ShaMAN and other MAS architectures do), but it does allow a functional analysis of goals, sub-goals, tasks, sub-tasks and plans. As with the ShaMAN Goal the HTA Goal is analogous to a Method Signature, while the HTA Plan is analogous to a Method Body – the specific logical recipe for achieving the goal. The Method Body may call upon other Methods

Table 1. ShaMAN meta-model comparison against selected Task Meta-models

ShaMAN	Hierarchical Task Analysis (HTA)	Task Knowledge Structure (TKS)	GOMS - Goals, Operators, Methods, Selection	Task World Ontology (TWO)	DIANE+
Goal (tree)	Goal (tree)	Goal (tree)	Goals(tree), Methods	Goal (tree)	Goal, Procedure
Role *(tree)*		Role		Role (tree)	
Responsibility		Responsibility (task) [2]		Is-responsible [2]	
Agent *(tree)*	Plan (tree)	Agent, Plan	Methods; Selection rules	Agent	Actor; Logical relationships
AgentRole		Playing [2]		Plays [2]	
Percept					
Event				Event	
SocialWorld*(tree)*					
SocialRole					
Member					
Item					
AgentRoleGoal			Operators	Performed-by [2]	
Task	Task (tree)	Task (tree)		Task (tree)	Temporal Relationships; Iteration
SpeechAct					Procedure
ActionType					
SpeechFlow					
Term					Data
Concept					
Association					
Resource *(tree)*		Object		Object (tree)	Object, Widget
AgentResource		Manipulate [2]			
Ontology *(tree)*					
List					
Locale *(tree)*					
Attendee					
Inhabitant					
OnSiteResource					
HotSpot					
LinkCondition					
R,A,D,I,T,Rt [1]	R,A	R,A,D	A,D,T[1]	R,A,D	R,A,D,I[2]

Note 1: R,A,D,I,T,Rt: Requirements, Analysis, Design, Implementation, Testing, Runtime.
Note 2: In the Task Meta-model, this is represented as a relationship rather than an entity.

(by Method Signature). I.e. The logic in a plan hierarchy can reference goals. So, a goal hierarchy can be specified without any procedural logic, although logical AND, ORs and XORs, are commonly used between predicates, as is the case in ShaMAN goal hierarchies. In application, HTA relies heavily on the feedback loop to: monitor progress of goals; to detect errors and reactive events, to call upon alternative plans in the event of errors and reactive events, In fact, the HTA feedback loop is very similar in principle to the functional event loop in a BDI agent.

The TWO version of a Task is two-fold: a unit task is the lowest level task that people want to consider in their work, while a basic task is system-oriented such as a single command in an applicaion. Tasks may be further subdivided into actions. The TWO Role is similar to that in ShaMAN, although it is seen as a collection of Tasks for which it is Responsible. Both have Agents playing/fulfilling Roles. TWO Objects are not OO-like, but are very similar to ShaMAN Resources in that they can be real

objects or abstract objects that exist in the knowledge context of an agent. What can be done with a TWO Object is specified by a list of actions. TWO Objects are strongly related to TWO Tasks, via the uses relationship.

The TKS Goal is very agent-like, in that it represents the goals in a human's head. Agents are synthetic intelligence with meta-models often mimicking a human psychology in the quest to achieve some level of intelligence. The concept of Role and Responsibility are directly equivalent to that used in several agent meta-models, including ShaMAN. However, in ShaMAN, Responsibility can be constructed at the analysis and design, as it is an associative entity between Role and Goal, rather than between Role and Task. Where the TKS Task manipulates TKS Objects, the ShaMAN Agent (which performs Tasks via AgentRoleGoal) has access to ShaMAN Resources (stored in a Knowledge tree), via the AgentResource associative entity, and so on. In the full study we do examine each twin comparison of each concept in detail.

4 Some Challenges for Task Analysis and Modelling

In Chapter 30, the final chapter of the Task Analysis handbook [6] editors Diaper and Stanton consider the future of Task Analysis. They make a case for the removal of two substantial concepts within Task Analysis, namely: Functionality and Goals:

"It (functionality) is a concept only applied to computer systems and not to the other main type of agents in HCI and task analysis, people (i.e., we do not usually discuss the functionality of a person, although we might discuss the functionality of an abstract role). The concept of functionality as what application software is capable of doing to transform its inputs into its outputs may well have been a reasonable one in computing's infancy, when programs were small and did very little, but today computer systems, particularly when networked and/or involving some artificial intelligence (AI), have outgrown the utility of the functionality concept. We believe that future computer systems will become further empowered as agents and that either we should happily apply the functionality concept to both human and nonhuman agents or drop it completely and use the same concepts we use for people when addressing the behavior of complex computer systems." (Note: the bolding is added).

They discuss the use of goal in TA: "What are goals for in task analysis? Chapter 1 identifies two roles for goals in task analysis: to motivate behavior and to facilitate abstraction away from specific tasks and thus promote device independence." Then outline ways to achieve those two things without the use of goals at all (using of attentional mechanisms), and conclude that goals ought to be dropped from TA.

"On this view, people, including task analysts, merely use goals as a post hoc explanation of system performance. Furthermore, attention, both selective and divided (see chaps. 14 and 15), is an existing cognitive psychological mechanism that controls the allocation of the massive, but still ultimately limited, mental information-processing resources, and we think that attentional mechanisms can entirely replace the motivational role of goals ... we suspect that goals in task analysis are similarly post hoc, whether elicited from task performers or inferred in some other way by task analysts. That is, we believe that goals are not part of the psychology that causes behavior but are used to explain it."

Both of these calls upon TA - to recede from the current inclusion of motivational Goals, and from the modelling of the ever growing functionality of complex modern computer applications - are one response to growing complexity and to interdisciplinary influences upon a discipline. Another response is to embrace and expand the field, inspired by the incursions of and into other disciplines and domains.

5 Enriching Task Modelling with Some AO Concepts

Today's users are multi-tasking much of the time and often have their computer systems running 24/7. The use of inputs and outputs in the earlier quotation highlights a different view of computational systems within the AO paradigm: Agents have percepts rather than inputs, and they act rather than just having an output - this underlines their perpetual operation in a task world or environment. They are designed to run 24/7, as are many current day mainstream applications, for example, an email client.

If AO concepts could influence TA with regard to the first recommended change to TA by Diaper and Stanton, it would advocate modelling of both users and agents – TKS and TWO have gone down this path to some degree. The AO approach is compliant with the idea of applying functionality concepts to both humans and agents – particularly within mixed-initiative systems. E.g. In ShaMAN an individual human is modelled as a hierarchy of sub-agents – a ShadowBoard agent - together representing the myriad of roles an individual user has [10]. Also note, that sub-agents in a multi-agent system are often much less than fully autonomous entities [2,5,7,10,11,13,14,22,24], and so functionality can and does get broken down into small definable units in these outwardly complex AI systems.

In response to the call to remove motivational goals, an Agent analyst would point out, that to restrict an agent to react to events only, in an attentional manner, would render it a simple reactive agent - the earliest form of agent architectures. Reactive agents were superseded by deliberative architectures, which, in addition to being reactive, are also proactive in pursuing their own goals, including motivational goals.

If, for example, you take either the TWO or the TKS task meta-model in figures 4 and 5, replace the agent entity with the SocialWorld sub-model in figure 10 above, replace the Object entity with the Knowledge Tree sub-model in figure 11, replace the Event entity with the MessageFlow sub-model in figure 8 – the result is a much expanded meta-model, which is as much an Agent meta-model as it is a Task meta-model. While it is large, it is also now capable of representing multiple users together with multiple agents, complex memberships and complex messaging, rich knowledge representations and complex goal, role and responsibility interrelationships.

The futures of Task Analysis and Task Modelling and the AO paradigm have much in common. AO researchers are predominantly intent on putting intelligence into artefacts (e.g. trading systems, robots, simulations, etc.), with only a small percentage of their number currently concerned with mixed-initiative human-agent systems. Task analysts and modellers are focused on people more than artefacts, and are therefore well-positioned and more inclined to embrace the modelling of people in mixed initiative human-agent systems, of which computer games are but demonstrative of the possibilities such systems hold for people in all walks of life.

References

1. Annet, J.: Hierarchical Task Analysis. In: Diaper, D., Stanton, N. (eds.) The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, Mahwah (2004)
2. Bresciani, P., Perini, A., Giorgini, P., Guinchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. In: AAMAS 2004, pp. 203–236 (2004)
3. Chen, P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems 1, 9–36 (1976)
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 Objects to a Multi-Agent System Modeling Language. In: Autonomous Agents and Multi-Agent Systems, vol. 9, pp. 145–189. Kluwer, The Netherlands (2004)
6. Diaper, D., Stanton, N. (eds.): The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, Mahwah (2004)
7. Fischer, K., Hahn, C., Madrigal-Mora, C.: Agent-oriented software engineering: a model-driven approach. International Journal of Agent-Oriented Software Engineering 1(3/4), 334–369 (2007)
8. Fitzpatrick, G.: The Locales Framework: Understanding and Designing for Wicked Problems. Kluwer Academic Publications, London (2003)
9. Fleming, M., Cohen, R.A.: User Modeling Approach to Determining System Initiative in Mixed Initiative AI Systems. In: 6th International Conference on User Modeling (1997)
10. Goschnick, S.B.: ShadowBoard: an Agent Architecture for enabling a sophisticated Digital Self. Thesis, Dept. of Computer Science, University of Melbourne, Australia (2001)
11. Goschnick, S.B.: The DigitalFriend: the First End-User Oriented Multi-Agent System. In: OSDC 2006, Open Source Developers Conference, Melbourne, Australia (2006)
12. Goschnick, S.B., Sterling, L.: Enacting and Interacting with an Agent-based Digital Self in a 24x7 Web Services World. In: Workshop on Humans and Multi-Agent Systems, at the AAMAS 2003 International Conference, Melbourne, Australia (2003)
13. Guinchiglia, F., Mylopoulos, J., Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In: Autonomous Agents and Multi-Agent Systems (AAMAS 2002) International Conference. ACM, New York (2002)
14. 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)
15. Johnson, P., Johnson, H.: Knowledge Analysis of Tasks: Task analysis and specification for human-computer systems. In: Downton, A. (ed.) Engineering the Human-Computer Interface, pp. 119–144. McGraw-Hill, London (1991)
16. 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, AAMAS 2003 International Conference, Melbourne (2003)
17. Kent, W.: A Simple Guide to Five Normal Forms in Relational Database Theory. Communications of the ACM 26(2), 120–125 (1983)
18. Kieras, D.: GOMS Models for Task Analysis. In: Diaper, D., Stanton, N. (eds.) Hand-book of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Inc., Mahwah (2004)

19. 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.) Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Springer, Berlin (1996)
20. OMG: MDA Guide V1.0.1 (2003), http://www.omg.org/docs/omg/03-06-01.pdf
21. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In: AOSE Workshop, AAMAS 2002, Bologna, Italy (2002)
22. Shen, Z., Li, D., Miao, C., Gay, R.: Goal-oriented Methodology for Agent System Development. In: International Conference on Intelligent Agent Technology, IAT (2005)
23. Strauss, A.: A Social World Perspective. Studies in Symbolic Interaction 1, 119–128 (1978)
24. Sun, R.: Duality of the Mind - A Bottom Up Approach toward Cognition. Lawrence Erlbaum Associates Inc, Mahwah (2002)
25. Tarby, J.C., Barthet, M.C.: The Diane+ Method. In: Second International Workshop on Computer-Aided Design of User Interfaces, University of Namur, Belgium (1996)
26. Van der Veer, G., Van Welie, M.: Groupware Task Analysis. In: Hollnagel, E. (ed.) Handbook of Cognitive Task Analysis Design, Lawrence Erlbaum Inc., Maywah (2003)
27. Van Welie, M., Van der Veer, G.: An Ontology for Task World Models. In: DVS-IS 1998, Adington, UK. Springer, Wein (1998)
28. 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)
29. Zambonelli, F., Jennings, N.R., Wooldridge, M.J.: Developing Multi-Agent Systems: The Gaia Methodology. ACM Transactions on Software Eng. and Methodology 12, 417–470 (2003)