Rich Socio-Cognitive Agents for Immersive Training Environments – Case of NonKin Village

Barry G. Silverman, PhD, David Pietrocola, Nathan Weyer, Oleg Osin, Dan Johnson,
Ransom Weaver, Ben Nye

*Ackoff Collaboratory for Advancement of the Systems Approach (ACASA), University of
Pennsylvania, 220 South 33rd Street, Philadelphia, PA 19104-6315*

Phone: 215-573-8368   basil@seas.upenn.edu

Abstract: Demand is on the rise for scientifically based human-behavior models that can be quickly customized and inserted into immersive training environments to recreate a given society or culture. At the same time, there are no readily available science model-driven environments for this purpose (see survey in Sect. 2). In researching how to overcome this obstacle, we have created rich (complex) socio-cognitive agents that include a large number of social science models (cognitive, sociologic, economic, political, etc) needed to enhance the realism of immersive, artificial agent societies. We describe current efforts to apply model-driven development concepts and how to permit other models to be plugged in should a developer prefer them instead. The current, default library of behavioral models is a metamodel, or authoring language, capable of generating immersive social worlds. Section 3 explores the specific metamodels currently in this library (cognitive, socio-political, economic, conversational, etc.) and Section 4 illustrates them with an implementation that results in a virtual Afghan village as a platform-independent model. This is instantiated into a server that then works across a bridge to control the agents in an immersive, platform-specific 3D gameworld (client). Section 4 also provides examples of interacting in the resulting gameworld and some of the training a player receives. We end with lessons learned and next steps for improving both the process and the gameworld. The seeming paradox of this research is that as agent complexity increases, the easier it becomes for the agents to explain their world, their dilemmas, and their social networks to a player or trainee.

Keywords: socio-cognitive agents, model driven architecture, explainable agents, game AI, immersive training.

1 Introduction

A growing number of producers and consumers of immersive simulators, virtual worlds, and game environments are interested in setting up and running cultural and cross-cultural training and rehearsal experiences. Their goal is that trainees could gain experience in foreign cultures and learn to be sensitive to local norms, values, and issues prior to arriving in the country or region where they must interact with and possibly influence and assist natives in that culture. By making the learning experience immersive, the trainees should hopefully have an easier time transferring their experiences to the real world.

This is useful for many types of users such as, but not limited to, multinational corporations tutoring their sales force, international aid organizations training their field representatives, and diplomatic advisors and military forces needing to learn how to handle counter-insurgency issues. As an example of the later, we shall show a case study in this article of military players entering a simulated foreign village and having to go through the three stages of counter-insurgency: inventory the population and befriend the stakeholders; isolate issues driving stakeholders apart and coopt the agenda; and identify mechanisms for a self-sustaining peace to take hold. However, this case study is only one example of the broader range of immersive cultural simulation needs just mentioned.

This approach to training and orientation is mostly done today with humans playing all the roles in the virtual worlds and simulators. That is, human trainers play the roles of the native stakeholders in order to give trainees the desired immersion and learning experiences. Still other trainers design the activities and scenarios that the role players will enact. This human role-playing is costly and time consuming and thus there is significant interest in replacing the humans with agent-based sims that can carry out the pattern of daily life and express the various stakeholder issues and behaviors autonomously.

This is challenging since agents for such immersive training games, simulators, and gameworlds would need to have aspects of their behavior be guided at various times by sociological, psychological, economic, political, etc. fields of concern. This implies that they are inherently very complex. Until now the agents that have been made for such worlds were a kind of one-off implementation that would incorporate some aspects

from different fields in one application/domain specific architecture. Therefore they are not reusable. In order to get to more reusable/maintainable frameworks we propose a model driven approach where one starts with a model of the different models from the sciences contributing to the functioning of the agents. Let us explore the specific research goals in the next section, after which we will more fully survey the current practice.

1.1) <u>Objective and Goals</u>

The objective of this research is to improve the realism of the social behaviors of agents in gameworlds in order to enhance player and trainee engagement or immersion in the experience. Immersion, as used here, means the human users feel a sense of connection to the social actors, and feel that the actors' concerns and the actions that can be taken in this world are reflective of dynamics from the real world. Since this is for training, we want to be careful to reflect what is important about a given ethno-political situation from a particular scientific stance. Specifically, our intent is to try and achieve this aim by making use of social scientists theory and data about real-world situations and their dynamics. While scientists and pollsters may not agree on specifics, and while subjects might shift their issues rapidly, one can still model communities meaningfully and deeply so that trainees will be appropriately challenged. While we use gameworlds and discuss similar concepts (eg, engagement, immersion, etc.), we are not pursuing this for entertainment purposes.

This objective leads to three goals:

1. **<u>Model Driven Behaviors</u>** – We are interested in culling best-practice theories from the social behavior literatures (eg, psychology, sociology, political science, etc) and implementing these as parameters, metrics, and models to drive the agents in the virtual environment. The early stages of the agent field concerned multi-agent simulations which captured fairly simple rules and showed how these cause agents to shift behavior in unanticipated ways leading to emergent phenomena and new equilibria (eg, Schelling, 1978). In general, these offered powerful displays of emergence, but involved over-simplification of human decision making and stakeholder issues. This over-simplification would cause loss of trainee immersion in a 3D simulation. Likewise, videogame agents generally have minimal content and instead rely on misperception, anthropomorphism, etc to trick the players into believing artificial life exists. But this fails on "up close" interaction tests. To better support trainee immersion and suspension of disbelief, our goal is to explore how to model the sim agents more explicitly and to make use of the growing social science theories and data models. The first goal is thus to explore a rich base of models to replace humans having to drive the avatars and to allow the agent sims to be able to mimic social dynamics and faithfully express real-world stakeholder concerns through action choices and conversational interactions.

2. **<u>Synthesis Across Models</u>** – The social sciences often tend to be fractured into disciplinary silos, yet social dilemmas and cultural concerns cut across such silos. Stakeholder issues often include aspects from all disciplines at once. As a result, we do not want a single model of behavior, but the ability to synthesize or plug together a range of the relevant models. From a scientific viewpoint, we want to explore best-practice models of the day, plug them together in meaningful ways, and readily unplug and replace them subsequently as new, better theories and models are derived. If done well, this synthesis might also help to identify gaps in the science and suggest new research directions. From an engineering viewpoint, we want a standards-based architecture in which to do this, so that it is widely understood, reusable, and maintainable/extensible. The second goal is thus to explore if the model driven architecture (MDA) standard from the Object Management Group and related software design patterns offer a useful pathway to synthesize, run, and manage a models collection.

3. **<u>Synthesis Across Functions</u>** – The ultimate aim of an immersive training environment is to improve trainee performance on specified pedagogical objectives. This raises the need for several added functions and components. The first component we wish to connect is a platform that runs the animations in a 3D gameworld. Since such gameworlds permit free play, there is a risk that the trainee will miss some of the pedagogical objectives. A common solution is to add a "stage director" agent or commander who is able to send the player to the proper situations (and set up more situations if the player misses some training). During play it also is useful and common to offer the player a place to store, inspect, reflect on, and analyze the items they accumulate. Often this is done in a virtual backpack or rucksack which might have a map that grows as players learn more of the world and that shows what happened where and who said what when. On the back end, the players also need to receive assessments of their performance in the gameworld. For this to work, one needs to have an assessment system with metrics provided by training developers and that can access and assess model parameters to discover what the NPCs (non-player characters) think of the player's behavior. All of these diverse components (3rd party gameworld, director/coach agent, rucksack suite of tools, assessment system, etc.) also need a standards-based way to interact with the sims and the sim world. Our third and final goal is to explore if the same MDA standards based approach will support these added components.

In sum, the goal of this research at present is to explore answers to these design questions. This is a significant undertaking and the current paper illustrates some of the progress made to date. Specifically, Section 2 surveys the literature and explores design trends to date and how and why we are trying to shift the state of the practice. That section concludes with a review of the MDA standard and an example of how it helps to plug models

together. Section 3 then delves more deeply into the library of models assembled across the social sciences, for handling agent choices and social dynamics as well as for transactional processes (eg, dialogs, interactions, etc). Section 4 provides a case study of bridging to drive the agents in an immersive 3D platform. Finally, Section 5 returns to discussion to the three goals and examines lessons learned about the state of design for immersive training systems. A related topic is whether a given design improvement translates into positive training value. However, assessing training impact is a future question for subsequent research and papers. In sum, the MDA approach has helped our thinking and code organization. Since goal 1 is to use models, the MDA approach facilitates goals 2 and 3, and the case study at the end of this paper is the proof-of-concept test that this works.

Before moving on, it is worth mentioning that the complexity of the agents we present here is in direct contrast to the parsimony and simplicity of agents often pursued in game theory, in artificial societies, and elsewhere where the KISS (Keep It Simple, Stupid) principle is followed: eg., see Edmonds & Moss (2005). Just as a bicycle will not do what a car can do, so too a simple agent will not do what a complex one can do. Rich socio-cognitive agents exhibit more realistic behaviors for enhancing immersion. In addition, this leads to what might be seen as an apparent paradox, though it is by design. That is, as agent complexity grows, there is an inverse decrease in the effort to get the agent to carry the burden for the dialog authoring. An agent that has many models and can explain its models will be capable of autonomously dialoguing about its world, its grievances, the networks and organizations its involved with, the conflicts and their potential solutions. Let us now explore this further.

2 State of the Art

As just explained, our aim is to improve the social realism of the agents in gameworlds for training purposes. We want to have agents that improve player immersion by reflecting greater depth and breadth. Specifically, this translates into better agent decision making and interactivity, two dimensions that we plot as the two axes in Table 1. Across the top of Table 1, it is useful to categorize sims or NPCs into three generations of agents. First generation agents (left column) are either human-played avatars or hand-scripted finite state machines of limited capability (eg, FSMs, cellular automatas, or social nets). In the second generation of agents (center column), the designers adopt ONE of many possible approaches to enhance the thinking and reasoning of the agent FSM (eg, cognitive model, planning algorithm, or cognitive attributes atop social nets), but do not offer a well-rounded capability. Thus these tend to be agents that are strongly autonomous in one dimension or another. The third generation (right column) seeks to unify capabilities so that the agents are reasonably good at cognitive as well as social tasks. This third generation of agent models do not typically have as great a strength in each dimension as second generation architectures, but they are better balanced and not as brittle.

Table 1 – Three Generations of Agent Thinking and Conversing in Immersive Training Worlds

Increasing Agent Understanding in Social Terrains

	1st Generation: Finite State Machines (and scripted actions)	2nd Generation: Discipline Thematic	3rd generation: Meso-Level Socio-Cognitive Agent Theories & Models (Trans-Discipline)
Humans play roles, supply dialogs (numerous players needed each session)	•Most Virtual Worlds, •Most MMPOGs •Most Wargames/SAF •MAS/Cellular Automatas, Social Nets		
Procedural Dialogs: Scripted branching dialogs (many 100s of trainer hours required to author each case)	•VW/MMPOG agents •Crowd Models/BOIDs •Most Videogames, eg, RTSs, FPS, Force More Powerful, Palestine, etc •Story/Narrative Agents (Branching dialog graphs)	•Goal Planning Algorithm Games, Black-White, GTA?? • Sociol. Sim (SocNet Envmnt, NetLogo, Repast, etc) •Cogn Models (ACT-R, SOAR, PsychSim, etc) •Embedded Conversational Agents (ECAs)– eg, REA, Tact.Iraqi, ICT games…	•Nurture Games (SIMS, Sim City) •Federated Models (cognitive, sociologic, economic, political, etc) •InsurgiSim (Red Force) •PMFserv-FactionSim
Declarative Dialogs: Humans profile inter-action model parameters & ethno-poli-cultural situation (a few hours to produce each case)		•Interactive Drama/Fiction Management (Oz, FAÇADE, Etc,..) •PMFserv & AESOP •Athena's Prism	•NonKin Village (daily life & culture) •Precursors: •MDA Standard •SW Design Patterns •Social Science Models

Decreasing Reliance on Narrative Developers

The vertical dimension or rows of Table 1 are intended to capture 3 levels of sophistication in the narrative dimension of a simulated world. In the first row are the simplest agents where humans must play all the roles and supply the dialogs. In the second row, we classify procedural dialogs – or – the case of developers having to write out each line of a fixed branching script. Finally, the third row involves declarative dialogs where the developers only need to declare high level agent parameters and the dialogs get dynamically generated at runtime as situations evolve. This row becomes possible as agent complexity grows.

Within the body of this 3x3 classification in Table 1 we can plot the progress of a large number of fields that use agents: videogame artificial intelligence (AI), artificial-life, virtual world folks, social science modeling, cognitive modeling, and narrative/conversational systems. In terms of the first row of the table, only the first column is populated. However, that is the vast bulk of the field in terms of quantity of agents built and operated. That is, the vast majority of "agents" one encounters today in role playing games, virtual worlds (VWs), massive multiplayer online games (MMPOGs), in mission rehearsal environments, and in military and commerce wargames are either human-played avatars or hand-scripted finite state machines of limited capability. The reason for this is twofold: (1) so much effort is required to set up and maintain the 3D immersive environments, that little budget or effort is generally left when it comes to adding AI, and (2) today's immersive AIs or agents are too easily perceived as mechanistic automatons, causing users to experience frustration, inappropriate expectations, and/or failures of engagement and training. Reliable pathways for creating more realistic and believable agents could ultimately help reduce barriers to interacting with as well as to creating behaviors of empathetic avatars, electronic training world opponents and allies, digital cast extras, wizard helper agents, and so on.

Moving down to row 2, most agent developers add dialog into their FSMs by some form of procedural scripting. This often takes the form of cue cards or hand written dialog lines inserted into procedural code and triggered by rules. This is used in innumerable gameworlds where players encounter the level "boss" or some other NPC that has a few choice lines to throw out to setup a scene, give a clue, or steer the player to a location. The basic design assumption, however, is that the gameworld is about players roaming about freely and stories constrain the players too much. Likewise, in social science modeling (eg, crowd models, social nets, cellular automatas), the interest is in the analysis, not the dialog with the agents. When these techniques are used for movies and games they need only minimal lines for their swarming armies, crowds, and extras. But some story games occupy this cell of the table that are interested in culture games (eg, Force more Powerful, Palestine, etc), or very clever dialog games like the Monkey Island series. The more powerful of these scripting approaches include very large branching dialog graphs for the characters to use in the diverse situations that come up. The hope is that there is a dialog branch for each situation the player might move into, however, this ultimately leads either to constraints on player movement (and creativity), or perceived brittleness of the agents. Still, the more well done of these do have their fan base.

There are several agent communities that operate at the center row and column of Table 1. For videogame AI and a-life developers, they tend to migrate here by "rewiring" their FSMs into a minimalist goal planning algorithm. The trick is to look smart or realistic long enough to escape the player's critical eye, so they only need a short amount of re-planning of goals, and for limited circumstances (Rabin, 2008). In contrast, a number of other communities co-exist in this cell that strive for greater autonomous reasoning in one direction or another. For example, as reviewed in Zacharias et al. 2008, the cognitive modeling community may be subsumed here with their many scientific models of human reasoning (eg., ACT-R, SOAR, PsychSim, PMFserv) as can the second generation social science models that add greater reasoning to the agents in their social nets, cellular automata grids, swarms, and so on. These cognitive vs. sociologic models tend to be opposite ends of the spectrum. On the one hand, (a) many of the cognitive models are quite detailed about a single person's rationality and reasoning, but are incapable of managing relationships, thinking about groups and collectives, or behaving in culturally relevant ways. On the other hand, (b) the sociologic models often handle network transactions, collective behavior, and relationships well, but have limited individual agent reasoning, though sociologic modeling environments like NetLogo, Repast, and so on now come with scripting languages so developers can create added (one of a kind) reasoning by agents. Some of these types of agents also can understand a narrative (dialog graph) as a plan and handle and manage conversations with a user as branching to diverse steps along the plan or toward better or worse relationships during the course of carrying out a task: eg, Bickmore & Cassell (2008). The games coming out of the Institute for Creative Technology (ICT) of USC appear to fall into this mid-level cell of Table 1 and amply illustrate the divide between the social science modelers. For example, ICT has produced a number of immersive 3D, branching dialog graph/conversational agent-based applications for training the US military in cultural sensitivity topics – eg., (ICT, 2010). Some of the games appear to use a goal planning approach (Full Spectrum Warrior), other use the cognitive science model in PsychSim (BILAT Game), and still others see use a sociologic model or social net (UrbanSim Game). These games illustrate that despite game industry trepidation, a lot can be done with a wide array of second generation AI.

Sticking with this middle column of Table 1, if one wishes to move to the bottom row this introduces a whole new story or drama management capability atop whatever social or cognitive AI the agent already has. The idea of drama management is to provide the player a chance for free play while simultaneously exposing

him/her to the intended plot points and story elements. This is often done with the aid of a drama manager or director agent that makes sure that the scenes and NPCs move to where the player is and try to involve the player in the intended plot regardless of what else the player is engaged in: eg, see Oz (Kelso et al., 1993)'s drama manager, Mateus and Stern (2003) with Façade and their beat architecture for managing storyline threads, or Sharma et al (2010) using a case based reasoner approach. Note, drama management does not require a natural language understanding, but an ability to have autonomous, interactive dialog in the form of context-triggered questioning and answering. In the current paper we seek to implement these ideas with the aid of a beat-oriented approach we call Authoring Electronic Stories for Online Players (AESOP) as well as the other components mentioned under Goal 3 at the outset (eg, Commander agent, Player Assessment and Feedback System). As will be explained we seek to make the narrative arc and dialog models responsive to the parameters of the other models driving the agents' behaviors, which in turn are directly influenced by the players' actions and/or inactions.

If we back up to the second row of Table 1 and move to the last column, this is now the territory of the third generation agents, albeit with traditional branching dialog graphs. In the videogame industry, the nurture games probably best typify the merger of sociologic and cognitive models, though there is no scientific basis to the models driving the SIMS or Sim City. Still, these are examples of agents that must balance both extremes of modeling behaviors. Another videogame example is cited in Bostan (2009) which has about 1,500 agents carry out a daily life set of activities and appear to socialize, though they are largely unaware of, and have only limited dialogs for socializing with, the players. Pew & Mavor (1998), Bjorkman et al (2001), Sun (2005), Graesser et al. (2008), and Zacharias et al. (2008) all point out that there are almost no examples of scientific models in this category, but a federation approach could help to preserve the investment in legacy simulator and game environments, while helping to synthesize across the social sciences, and making newer behavioral model innovations available. This path has been advocated by the US Department of Defense, among others, who has identified a need for interoperability of human behavior models to help improve the realism of agents in legacy simulators. This is, essentially, the argument of column 3, that many different types of social science models should be synthesized if one is to effectively model social systems. Thus in the social simulations, in general, "model driven" must mean not just one model, but a library of interacting models – a model of models. This pluralization is an important 'extra' and the topic we turn to next in discussing a standards-based way to manage multiple models.

In order to progress to the bottom right corner of Table 1 (3rd generation in thinking and interacting), one wants to have NPCs that use a science-based federation of models to drive their behaviors and so they can dynamically explain themselves, their wants and aspirations, and their grievances and complaints. This means they can converse about their own thinking as well as their relations, their affect/cognitions and their socio-politico-economic concerns. The games mentioned above in the other table cells do not achieve this. For instance, a minority of the ICT games use the science models in PsychSim (Marsella, et al, 2004), but this omits socio-politico-economic aspects and is thus far narrower then our focus here. Likewise the drama manager games have no social science models behind them, but instead illustrate social settings for entertainment value. For instance, Façade is about a divorce but has no social science model of marriage relations or the divorce dilemma.

2.1) The Model-Driven Architecture (MDA) Approach

An immersive simulated community created from models would facilitate countless avenues of inquiry and training in fields as diverse as education, public health and relief, city planning, security operations training, and policy analysis, among others. However, the major complication facing anyone who attempts a social systems modeling library is that there are no mature scientific theories (and certainly no first principles) about what to model and how models inter-relate. For example, how do memes migrate? How do people convey norms? How does alternative media and messengers play a role or not at the micro-processing level (and how does this differ upon each type of listener)? Why are some groups happy in impoverished conditions while others rebel? What is the correct theory behind terrorism (sacred values, religious extremism, genocide, patrimonialism, poverty, jealousy, mimicry and copying of fashionable but deadly youth movements, etc.)? There are innumerable questions like these that have no single answers. The result is that a scientific and social system testbed must be assembled and so alternative competing model hypotheses can be studied. Real world social systems and cases need to be recreated in the gameworlds and improved over time as new evidence is uncovered. The breakthrough we hope to occur from this effort will happen only with a socio-cognitive agent framework that can serve as a theory testbed to study the intersection of psychological and sociological theories and phenomena. Such an architecture, to be effective, also may need to model the organizations, economics, infrastructure, and institutions that service and support the individuals in a region.

In particular, we are interested in training and analysis of ethno-political conflict in diverse cultures. In order to support study of this problem, we attempt to provide a modeling suite that allows one to configure diverse theories of cross-cultural conflict and their resolution. Specifically, culture is often thought of as (1) communicative practices involving language and gestures; (2) systems of regulation external to the individual

agent, including formal laws, religious tenets, and norms of practice for different stakeholders; (3) key beliefs that individuals and groups hold -- personal mindsets and collective worldviews; and (4) cognitive processing differences such as, for instance, Middle Eastern quam or trust building vs. Western contract and decision-centric. In this research and paper, we explore a testbed for implementing models of the latter three of these in order to simulate artificial societies and cultures.

Once one decides to go down this pathway with a model base, it tends to explode in size and one winds up with many dozens of models that need to be managed. Fortunately, there are several model-oriented standards, design patterns, and best practices to help out. We discuss the model-driven architecture (MDA) standard here, and add two others in Section 2.

So what is the model-driven approach and how does a standard support it? Researchers and software engineers have long sought effective paradigms for enabling domain experts to write their own software and, for all intents and purposes, without a need to write their own "code." Schmidt provides an overview of such efforts by describing computer-aided software engineering (CASE) and explains its lack of commercial success due to restrictive domain constraints (Schmidt 2006). Similarly, the multiagent systems field has asked the same questions by looking to mature agent-based modeling tools for end users in domains of interest. For example, a skills trainer may have a mental model of how certain interactions should proceed in a domain, or context; a model-driven architecture (MDA) would allow the user to express such a model in a natural way and produce an application. Despite the obvious benefits for such a method, reaching the full ideal has remained an elusive goal, and a "one-size-fits-all" representation for such domain models has been difficult, if not impossible.

Still, many in industry have struggled with this idea and the MDA is now a standard (see Object Management Group. 2003). MDA came out of the software industry's attempts to handle increasingly complex projects and systems that become weighed down by programming code specifics. MDA's goal, essentially, is to separate business system requirements and domain details from the specifics of any one technology or platform. The MDA standard specifies that one should create three layers:

1. **Platform Independent Meta-MetaModel (PIMMM)** – This is the high level authoring environment that supports domain specialists in specifying their social system requirements, mental models, and user applications without needing to write code or program models. This also includes the components that the library of models will be assembled from.

2. **Platform Independent MetaModel (PIMM)** – This is the suite of models in code that are exposed in the PIMMM for domain specialist usage. The programmers must author the PIMM in a way that it can be readily maintained and updated as new models come out, yet still seamlessly support the PIMMM specialists. Further, PIMM programmers must develop and maintain middleware bridges so the lowest layer (PSM) can also seamlessly interoperate.

3. **Platform Independent Model (PIM)** – This is the instantiation of a single scenario out of the PIMM. This is independent of any platform that will animate the scenario, but it is a commitment to a configuration that will then work as a server that interacts with a platform-specific client.

4. **Platform Specific Models (PSMs)** – These are the actual platforms where users (trainees, analysts, and other) interact with the domain specialists' simulated worlds and agent applications. These might be any number of 3D environments or VWs running on diverse machines.

A reasonable question is who has used the MDA ideas and what have they achieved for domain specialist programming of applications? The difficulty is that even though this standard was published in 2003, it still remains an elusive goal to implement it in any domain, including the agent-based modeling field. So any survey in the agent field, is more a survey of what portions of the MDA have been omitted and why. Let us examine several examples from the literature.

a) A prime (though non-agent) example of this approach would be using the Unified Modeling Language as a PIMM (Schmidt 2006). One can then run language-specific code generators to produce skeletal PSM code that can be filled in to run on a given platform. But this example only highlights the difficulty since the PSM application cannot be produced by the UML author without programmer support. Further, UML authoring is not truly a domain-ready PIMMM, since one must conquer the UML language and concepts to use it. It requires Object-Oriented Analysts to use it, not domain specialists.

b) The NetLogo environment for agent-oriented applications is a great example of a multi-agent systems environment with broad applicability (Surhone et al. 2010). The applications run over web-browser plugins so they are, in fact, PSMs that run on any machine that can support Java plugins (though they also require NetLogo to display their applications). As a PIMM, NetLogo includes an agent authoring environment and generic screen objects to display the emergent results. However, it relies on the programmer to implement the social science model of interest (economics, social behavior, psychology, etc.). There is no library of social science models built into it, though they have a public website where one can inspect the models that various students and users created. Further, there is no inherent support for a PIMMM usable by domain specialists not interested in learning the NetLogo language and agent based modeling and simulation topic.

c) Another strain of recent research is the early attempts at MDA prototyping in multiagent systems: eg, see Bezivin 2005; Hahn et al. 2009, among others. Hahn et al. provide a survey of approaches and then describe a prototype of a platform-independent metamodel (PIMM) for developing agent applications in a

generalized manner. By using high-level domain concepts of agents in interactions and organizations, they follow the MDA standard to transform platform-independent models (PIM) into platform-specific models (PSM) in two agent environments using mapping rules. There is no attempt to describe a PIMMM. Further, the example agents that get generated are for workflow tasks and not social simulation (our objective).

2.2 Synthesis of Two Illustrative Models

There are many ways to federate models with each other and with explanation engines that can foster conversation with users. To help the reader visualize what we will be scaling up throughout this research, it is helpful to show some details about how two illustrative models might inter-operate. Specifically, we describe the collaboration between an agent's default perception model, and its default decision model. These models represent the core of a PMFserv agent's reasoning ability (see Sec. 3.1.1) and their connections may be generalized to all models in PMFserv to generate the PIMMM under discussion. Instead of a rigid standard for plugging models into the framework, PMFserv facilitates several design patterns, including traditional object-oriented programming practices and the publish-subscribe pattern, which will be discussed here.

Figure 1 illustrates the three steps to including a model in the framework. These three components feed into the PMFserv simulation, which resolves all models in a scenario through a model manager and controls agent steps. On the left-hand side, PMFserv includes a model repository that is maintained by the model manager, and each model must be registered with the model manager. New models may be added by subclassing from the base model class to create a plugin class and simply including the model in a designated plugin directory. The PMFserv plugin manager evaluates these plugin classes and adds them to the overall model repository, which contains all models native to PMFserv. Once registered as a PMFserv model, a new model may require data or parameters from other models that exist in the collection. For example, the decision model of this example is dependent upon an emotion model, a values tree (or multi-attribute payoff function as will be described in Sect. 3.1), and a perception model. A perception model is defined as any model that examines the simulation environment and outputs a list of afforded actions along with how taking such action would affect an agent's values tree. As the middle component of Figure 1 shows, these dependencies are specified in a dependency graph. The model manager then resolves all mandatory and optional dependencies, ensuring that the appropriate model references will exist at agent creation time.

Figure 1 – Example Illustrating How the Model Plugin Framework Works in PMFserv
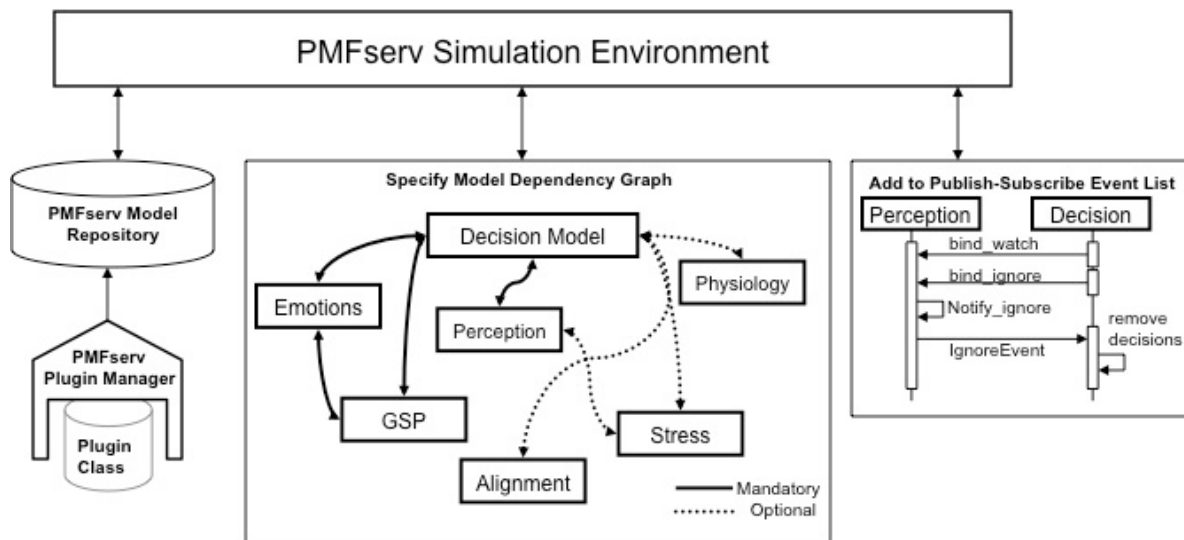
Let us now describe the interaction of perception and decision-making in an agent (the right-most component in Figure 1). The default decision model in PMFserv, which performs a subjective expected utility calculation, asks the model manager for a reference to the perception model and then subscribes (binds) to events that perception publishes (notifies). For example, a chosen perception model may define perception scope of the agent as complete omnipotence, where the agent perceives all entities in the world and their actions, or perhaps only a field of view if the agent were operating in a virtual world. However, the decision model should not care about these details and instead is only updated with events of importance to the decision-making process. Specifically, entities that are being watched or ignored dictate which entities may afford actions to the agent. The decision model then asks a separate emotion model (also in Sect. 3.1) to evaluate how taking each available action would change its own emotional state. This emotional activation (or decay) is used by the decision model to calculate a subjective expected utility and the highest ranked action-target pair is then selected to be performed by the agent.

3 Model Base Management Practices

In this section we overview the library of models that we are currently synthesizing and federating to simulate villages and other regions. In particular, as already mentioned, we are interested in creating and autonomously running an immersive artificial society on the scale of a village, small town, or other community. In general, we need two broad model categories: product and process. Product models are things like agents, groups, buildings, infrastructure – anything that will have a physical presence, things that might be teleological and take actions in the world, or things that might be the object of actions. One can imagine the range of what is needed for products to be realistic would have to straddle: (1) the minds/bodies of the agents that live there and what motivates their daily existence; (2) the influential organizations and groups in the region of interest ranging from familial to commercial to governmental and so on; and (3) the structures, assets, and infrastructure of the organizations that service the residents. Agents might decide to go home, show up at work, or buy some goods, but for that to be visualized they need a way to carry out the associated processes. Process models thus are needed of the workflows, interactions, conversations and the like that we want users to be able to visualize. Specifically, (4) in a drama theoretic sense, one also needs models that can be reused and instantiated to run the array of interactions, transactions, and conversations that can transpire between individuals as well as the types of transgressions and grievances (or good will) that might arise. The important point is that these elements are the building blocks, or syntax, for any downstream metamodel development. We will describe product- and process-oriented modeling packages that satisfy these features in Sections 3.1 and 3.2, respectively.

Before explaining the models, it is worth pointing out that several software engineering practices – model-driven architecture (MDA), model factory pattern, and product vs. process models – have helped us manage the complexity of the resulting model library. These three practices are summarized in Figure 2 as will now be explained. First, Figure 2 reflects the three big components of the MDA across its breadth. This segments the metamodels from the platform model. It also segments the specs for the metamodels from the metamodel library. That allows one to separately manage and control these differing components. The idea of moving the models library out of the PSM and into a metamodel might initially seem like a lot of effort. However, it is the nature of social/behavioral models today that one often must embed behind a client's legacy simulator. This is a challenge. In a recent survey of five legacy combat simulators (JSAF, ModSAF, OneSAF, DISAF, JCATS), it was found that (1) one often cannot discover if a given behavior exists or what level of fidelity it is modeled at; (2) the software is growing constantly; (3) verification and validation needs of the legacy software make it prohibitive for anyone other than the prime contractor to add updates (Lavine et al. 2002). This study indicated the need to find novel ways to off-load behavior modules and agent software to external servers where they can be separately maintained and validated. When needed they could be dynamically federated (i.e., interoperated) through a mediating service. This article investigates MDA as one such federation approach.

Second, the center of Figure 2 also shows a Model Factory. The factory is a software design pattern (Cohen & Gil, 2007) defined as a facility where the user provides the specs (as in a Dell computer website order) and pre-canned parts are simply snapped together at assembly time to satisfy the order. In the case of software, this is the "baking process". This is consistent with the MDA concept (separating PIMMM from PIMM). The factory of models we adopt is shown in Figure 2 as an extensible model of models architecture where cognitive, social, and experimentation layer models and tools all can be plugged in or out as needed for the region being studied. The "glue" for synthesizing the many models together involves the model registry and publish-subscribe process defined in earlier Section 2.2.

3.1) Socio-Cognitive Agents (Product Models)

As already mentioned, the vast majority of agents in immersive social training worlds are played as avatars with the second highest fraction being simple finite state machines that have low level AI (navigation, collision avoidance, simple artificial life functions) and are meant to be hand-scripted for any higher level behaviors. The goal of our research is to try and synthesize models across the social and behavioral sciences that will be needed to support autonomous agents carrying out their lives in an artificial society. Since we are interested in cultural training and ethno-political conflict resolution, we also have a constraint that the models help to recreate real world places and/or close facsimiles that are archetypical of that real-world culture. We seek to do this by assembling a collection of product models including agents, groups/organizations, and natural and manmade structures. The following three sub-sections explore these items.

Figure 2 -- Overview of How the Model-Driven Architecture, Model Factory Pattern, and Product vs. Process Model-View-Controllers Help to Manage the Models Library
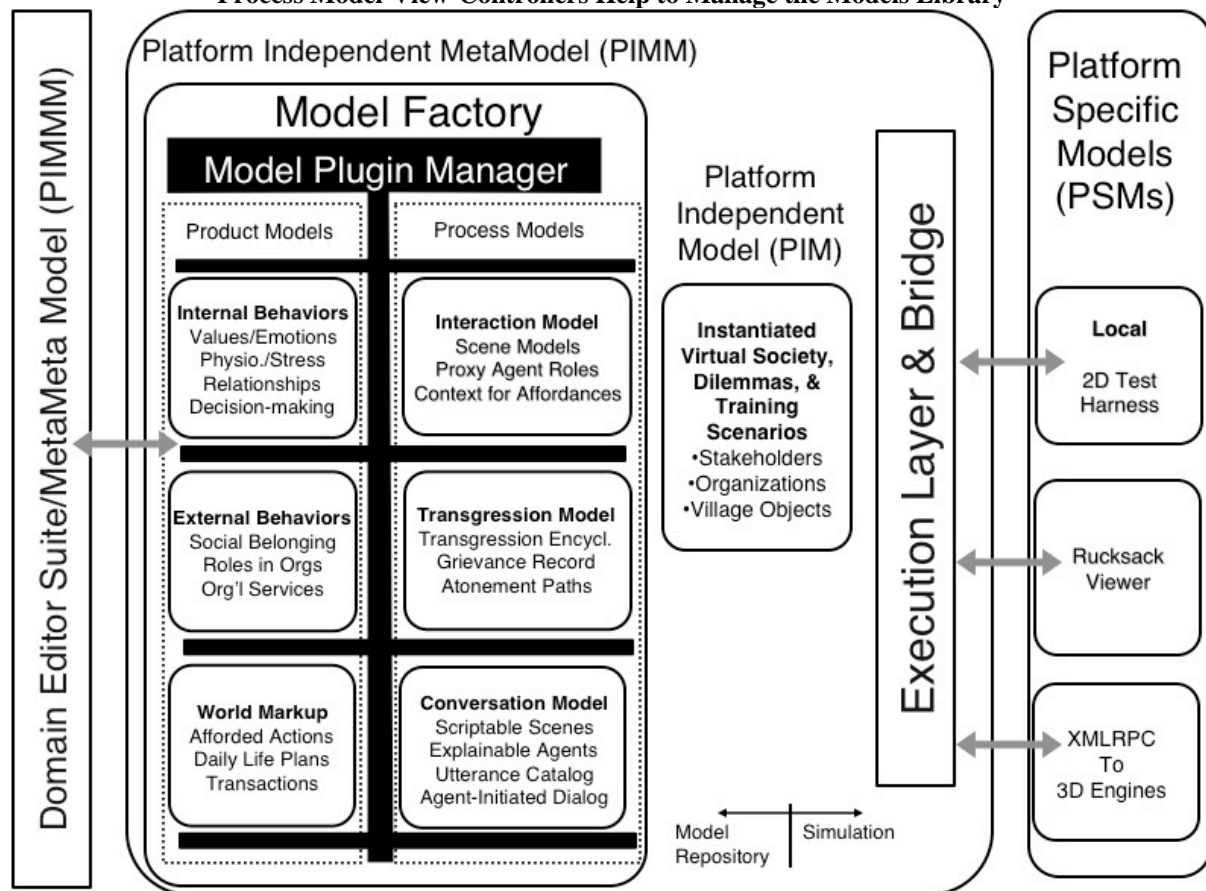
3.1.1) Agent Cognition: PMFserv

Since our goal is to study purposeful or teleological agents, we want the agents in the artificial society to have significant cognitive depth to them. We could adopt any of a number of cognitive models for this purpose, but a constraint is that they must support modeling individual differences, personality, and cultural factors. For this purpose, in this section we introduce PMFserv, a COTS ("Commercial off the Shelf") human behavior emulator that drives agents in simulated game worlds. This software was developed over the past ten years at the University of Pennsylvania as an architecture to synthesize many best-of-breed models and best practice theories of human behavior modeling. PMFserv agents are unscripted, and use their micro-decision making, as described below, to react to actions as they unfold and to plan out responses.

A performance moderator function (PMF) is a micro-model covering how human performance (e.g., perception, memory, or decision-making) might vary as a function of a single factor (e.g., sleep, temperature, boredom, grievance, and so on.). PMFserv synthesizes dozens of best-of-breed PMFs within a unifying mind-body framework and thereby offers a family of models. None of these PMFs are "home-grown"; instead they are culled from the literature of the behavioral sciences. Users can use the PMFserv IDE to turn on or off different PMFs to focus on particular aspects of interest (or add their own PMF models). PMFserv is an open, plugin architecture[1].These PMFs are synthesized according to the inter-relationships between the parts and with

---

[1] It is worth noting that because our research goal is to study best-of-breed PMFs, we avoid committing to particular PMFs. Instead, every PMF explored in this research must be readily replaceable. The PMFs that we synthesized are workable defaults that we expect our users will research and improve on as time goes on. From the data and modeling perspective, the consequence of not committing to any single approach or theory is that we have to come up with ways to readily study and then assimilate alternative models that show some benefit for understanding our phenomena of interest. This means that any computer implementation we embrace must support plugin/plugout/override capabilities, and that specific PMFs as illustrated in Figure 3 should be testable and validatable against field data such as the data they were originally derived from.

each subsystem treated as a system in itself.

The unifying architecture in Figure 3 shows how different subsystems are connected. For each agent, PMFserv operates what is sometimes known as an observe, orient, decide, and act (OODA) loop. PMFserv runs the agents perception (observe) and then orients all the entire physiology and personality/value system PMFs to determine levels of fatigues and hunger, injuries and related stressors, grievances, tension buildup, impact of rumors and speech acts, emotions, and various mobilizations and social relationship changes since the last tick of the simulator clock. Once all these modules and their parameters are oriented to the current stimuli/inputs, the upper right module (decision-making/cognition) runs a best response algorithm to try to determine or decide what to do next. The algorithm it runs is determined by its stress and emotional levels. In optimal times, it is in vigilant mode and runs an expected subjective utility algorithm that re-invokes all the other modules to assess what impact each potential next step might have on its internal parameters. When very bored, it tends to lose focus (perception degrades) and it runs a decision algorithm known as unconflicted adherence mode. When highly stressed, it will reach panic mode, its perception basically shuts down and it can only do one of two things: cower in place or drop everything and flee. In order to instantiate or parameterize these modules and models, PMFserv requires that the developer profile individuals in terms of each of the module's parameters (physiology, stress thresholds, value system, social relationships, etc.).

As an illustration of one of the modules in Figure 3 and of some of the best-of-breed theories that PMFserv runs, let us consider "cognitive appraisal" (Personality, Culture, Emotion module)—the bottom left module in Figure 3. This is where an agent (or person) compares the perceived state of the real world to its value system and appraises which of its values are satisfied or violated. This in turn activates emotional arousals. For the emotion model, we have fully implemented the one described in Ortony et al. (1998). The OCC model uses appraisals of actions to compute relationships between each pair of agents based on two primitive variables: valence (varies from -1 to +1) and degree of agency or humanness (varies from 0 to 1). An agency of 0 implies the viewer views the other as an object and its values need not apply. This accounts for the ability of agents who believe "thou shalt not kill", to commit murder and genocide without violating their values.

Figure 3 – PMFserv, an example Cognitive Architecture
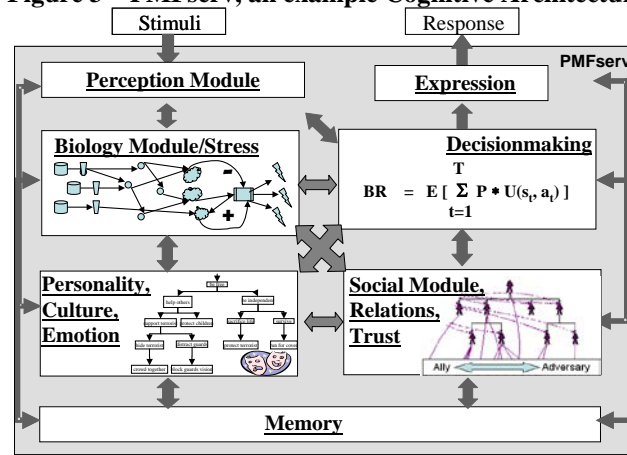

To implement a person's value system, this OCC model requires every agent to have goals, standards, and preference (GSPs) filled out, though they give no constraints on how to implement that. In our implementation, GSPs are multi-attribute value tree structures where each tree node is weighted with Bayesian importance weights. A GSP Tree is shown partially opened on the left of Figure 4. A GSP tree encapsulates an agent's long-term desires for world situations and relations (Preferred states or preferences), standards that govern its behavior and how it judges behavior of others, and goals for short term needs (e.g each day or tick). Each agent must have the identical tree structure, but their weights are individualized to capture personality and cultural differences. So the left of Figure 4 shows the weights for a villager who is relatively pro-government, while the right shows the weights of a committed Taliban Jihadist. Also, the far right of Figure 4 illustrates how one of the metrics models of the next section (motivational congruence) is computed from lower level agent models. This is an example of the output of a cognitive model being used to help provide the inputs for a sociology model. Agent values similarity to leader values is part of the reason that agents keep membership in a group. The agent's values are oriented in the OODA loop, then used for decision making as described in earlier Sect. 2.2, and now the reader sees they also play a role in helping the agent figure out their social network relations as the next section will elaborate.

We generally elicit the GSP weights from experts in the culture of interest, but in theory, one could automatically infer them from a corpus of past actions by real world agents. At runtime, this set of values is used by each agent to Orient and compute a weighted hierarchy of the current state the agent is in. For the Decide step, the agent infers another weighted hierarchy for each possible new state that an action could precipitate. By comparing each possible action available, the agent can compute its next action by selecting the one (or several) action with best possible next step utility. Utility is simply the roll up of the expected activations from that

action times the Bayesian weights on the relevant branches of the GSP tree structure. The resultant is a Subjective Expected Utility. It is expected since action choices might not lead to the actual outcome. In some cases FactionSim will run them through the Lanchester equations (see below) to compute likelihood of success, while in other cases, other agents in the society might be taking countermoves that alter the outcome of the original agent's actions.

**Figure 4 - GSP Value Tree Structure, Weights and Emotional Activations for Opposing Leaders**
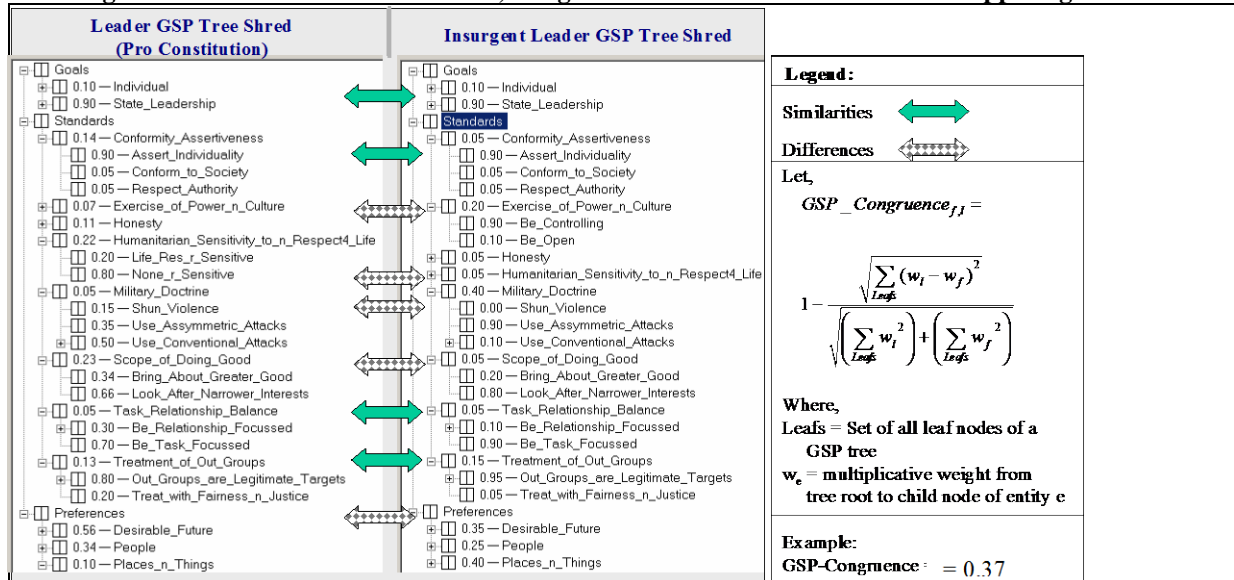
3.1.2) Social Framework: FactionSim

One suite of models we have been assembling offers a generic game simulator to social scientists and policymakers so that they can use it to rapidly mock up a class of conflicts (or opportunities for cooperation) commonly encountered in today's world. Simply put, we have created a widely applicable game generator (called FactionSim) where one can relatively easily recreate a wide range of social, economic, or political phenomenon so that an analyst can participate in and learn from role-playing games or from computational experiments about the issues at stake. Indeed, this game generator and its suite of social science models have successfully been applied in the past to recreate communities in SE Asia, the Mid East, and Horn of Africa. In over 200 statistical correlations with real world communities it has been shown to be over 80% accurate at recreating the conflict and cooperation decisions of leaders for and against the groups they manage and follower membership action choices: eg., see Silverman et al. (2009). These statistics hold for decisions aggregated and averaged over 3 month intervals, and FactionSim should not be interpreted as capable of predicting what a specific leader or follower will do on a given day or even in any given week.

Factions are modeled where each has a leader, various sub-faction leaders (eg, loyal and fringe), a set of starting resources (Economy, E, Security, S, and Politics, P), institutional ministers who manage the flow of public good and services, and a set of N follower agents. A leader (and possibly a leader council as well as the institution ministers) is assumed to manage the faction's E- and S- tanks so as to appeal to the followers and to each of the other tribes or factions they wants an alliance with. Each of the leaders of those factions, however, will similarly manage their own E and S assets in trying to keep their sub-factions and memberships happy. Followers determine the level of the P-tank by voting their membership level (a topic discussed later in this paper). A high P-tank means that there are more members to recruit for security missions and/or to train and deploy in economic ventures. So leaders often find it difficult to move to alignments and positions that are very far from the motivations of their memberships.

As an example, consider Figure 5 which shows the main groups and members of an artificial society. Here we are showing a village of two Pashtun clans (center of Figure), the Afghan government on the left, the Taliban on the right, and the US-led International Forces in the top center. In Section 4 we shall explore what happens when a human plays the US forces. One can also see in Figure 5 that each of the main groups has subgroups; ego, social and economic networks; and there are resources and services that are included as well. As indicated by the dashed arrow and box in the lower left, we are showing that there are little services currently for those in the rural villages (the Government of Afghanistan primarily supports the urban centers). Creating those local services and fostering jobs will be the way to counter the competing services that the Taliban does provide in that region.

Figure 5 – FactionSim Layer of an Example Artificial Society
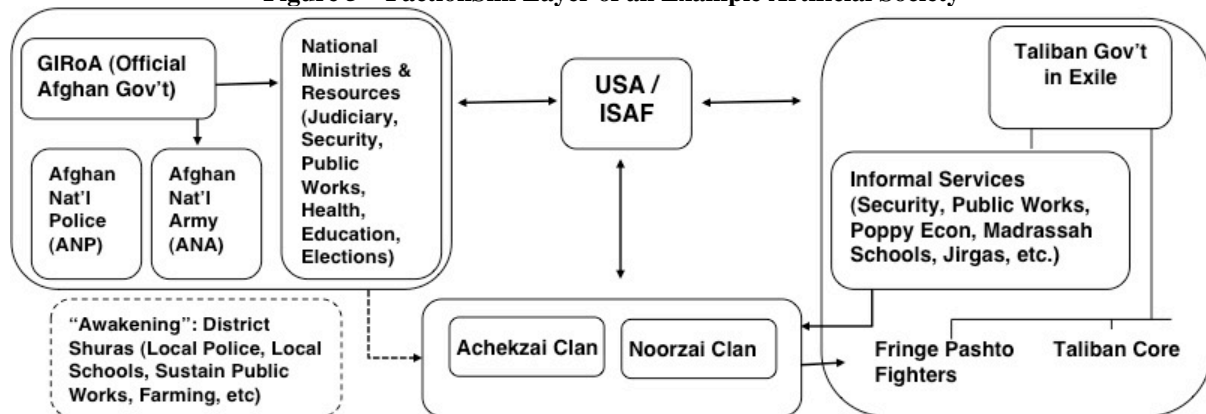
In order for this to work, there must be models behind the scenes that run this society. Indeed, FactionSim includes a suite of default, first approximation models which are extensible and can be improved and/or replaced by developers who have their own models they wish to explore using. These include:

- Security Model (Skirmish, Modified Lanchester) – carries out attack action choices of diverse leaders:
  - Power-vulnerability Computations (conducted by each group and leader)
  - Skirmish Model (force size, training, etc.)
  - Modified Lanchester Model (probability of kill, damage) – used for economic attacks as well
- Economy Model (LMR model) – gives each agent a wallet and each group a treasury (econ tank), carries out many economic actions, and updates the flow of services, goods, jobs, taxes, and related economic transactions for the following sectors:
  - Black Market – poppy economy, smuggling
  - Formal Capital Economy – farming, marketplace, taxes, and related jobs and enterprises
- Socio-Political Model (loyalty, membership, injustice, etc.) – handles governmental, institutional, and group management actions carried out by leaders as well as followers:
  - Institution Sustainment Dynamics – handles minister agent allocations and distributions, bribes
  - Group Membership model – computes numerous averaged metrics using inputs from members of each group such as resource satisfaction levels, group collective beliefs (vulnerability, injustice, distrust), mobilization and votes, and motivational congruence (members vs. leader). The latter of these was illustrated in earlier Figure 5 and discussed in Sect. 3.1.
  - Group Alignment and Familiarity models – computes numerous individual metrics based off of agent to agent relations (valence and agency parameters – see next section) such as alignment, strength of relationship, authority/legitimacy level, InGroup influence, CrossGroup Alignments and Influence levels, and individual agent reputation/familiarity (a merger of rapport and trust).

FactionSim is thus a tool where you set up a scenario in which the factional leader and follower agents all run autonomously and are free to use their micro-decision making as they see fit. One can study what emerges and explore what-if questions, or use it immersively in role playing games and training sessions. The models provided are default PIMMs only, and the PIMMM facilitates the codification of alternative theories of factional interaction and the evaluation of policy alternatives. At present, however, the PIMMM only supports parameter experiments on existing models and requires knowledge of Python to use it for substituting in new models.

3.1.3) Affordance-Based Perception Model and World Entity Markups

While FactionSim specifies the resources of and services provided by and to each group (eg, family, business, political party, religious sect, etc), this is done in the abstract as statistical properties. For 3D worlds it is important to map these onto actual natural and manmade structures. For example, if a given group's institutions provide health services that only reach 30% of the people, then this should appear in a 3D world as a small clinic with limited capacity. And, conversely, if that facility is damaged in the 3D world, this needs to be reflected back to FactionSim as a drop in capacity. Thus, in addition to managing agents, FactionSim and PMFserv also manage perceivable entities (representing both agents and non-agents, such as a car, location, a business, etc), including when and how they may be perceived and acted on by agents.

PMFserv manages agents' perception of perceivables by implementing affordance theory, meaning that each entity (ie, object, group, institution, and agent) applies perception rules to determine how it should be seen by each perceiving agent. Entities then reveal the actions (and the potential results of performing those actions)

afforded to the agent. For example, an object representing a car might afford a driving action which can result in moving from one location to another. A business might afford running it, working there, purchasing goods, and/or attacking and damaging it. To make it easier to edit a virtual world, a great many entities are being pre-encoded in the PMFserv libraries so that training scenario developers need not fill in the markups, but only need to link them to structures, areas, organizations, etc. of that town or region. Section 3 will illustrate this further.

These entity affordance markups permit the PMFserv agents to perceive and reason about the world around them. Thus the PMFserv agents operate their OODA loop and notice their own needs after the Orient step.

The afforded actions convey how much satisfaction of needs and/or activation of utility they would provide to the agent (eg, kcal of eating a unit of food, utility from hitting an opponent, or utility from aiding a friend). The Decide step of the OODA loop merely appraises and ranks all possible actions and selects the optimum relative to the agent's current needs. The GSP trees of earlier Figure 5 show two small tanks to the left of each node on the tree. They are white or empty in that figure. When a given afforded action succeeds the left tank activates, if it fails, the right tank activates. Succeed and Fail activations cause the agent to want to satisfy that GSP node less or more on the next tick. Thus if the agent just ate and satisfied its Stay-Healthy Goal, it won't need to eat again until the succeed tank activation decays (or the stomach grows empty due to exertion).

Using this implementation of affordance theory, we have developed a taxonomy to categorize and build up complicated entities that would be needed in an immersive virtual society with socio-cognitive agents and sufficient player interaction capability (Pietrocola et al, 2010). Ultimately this serves the purpose of allowing agents to reason about the environment, including objects like buildings that may afford services such as going to work, or purchasing food. This taxonomy also includes numerous multi-step plans for carrying out daily life functions (eg, go to work, pray, socialize with friends, carry out an attack on a target, etc.). FactionSim models described earlier make a set of actions available in the abstract, whereas these actions are the lower level, individual agent counterparts that permit them to carry out a Faction leader's higher level plan should he communicate it to them.

Considering our desire to expose culture in our models, we also build upon universal affordances like the examples just outlined, and embed culturally-affected "collective perceptions." Specifically, we use a culture's systems of regulation and core beliefs to influence how and when structures and environmental features are perceived. For example, behaviors related to food purchasing may be driven by norms like market days or religious observances. Similarly, gender differences may affect accessible locations and interactions between NPCs.

Additionally, the decentralization of perception rules to the objects themselves provides an engineering advantage along the lines of modularity and reusability (Cornwell et al, 2003). Each way in which an object, group, institution, or agent may be perceived is called a perceptual type, or p-type. A grid of p-types is created for each perceivable entity. Rules on a p-type allow a modeler to establish appropriate contexts for the object to be viewed in that way. When active, p-types afford actions to the perceiving agent and the decision-making process can proceed.

3.2) Process-Oriented Models: Interactions, Transgressions, and Conversations

Process models were earlier defined as the abstract models of social interactions that can happen between agents, groups, agents to objects, etc. In most game environments, one expends significant manual labor in producing these types of "models" as one-off, hand- scripted and hard-wired dialog graphs, limited Markov action chains (and rules), and preset spatio-temporal aspects (timelines, workflows, routes). An alternative is to approach all of these as potentially reusable models. For that to work, one wants such models to be relatively well done, maintainable, and extensible. Ideally, such models could interact with product models. Thus there should be some synergistic effect where many of the interactions and process dynamics freely emerge from the product models. One should not have to hand-author every utterance, ever action sequence, and every scene if the agents are autonomously thinking of their own grievances, desires, mobilizations, etc. This section begins by introducing our Interaction Model, then explains Transgression and Atonement, and finally winds up with the types of Conversation Models that are being supported. All of these models are supported with an editing tool we call Authoring Electronic Stories for Online Players (AESOP). AESOP is a first draft of a PIMMM suitable for programmer usage.

3.2.1) Interaction Model

In carrying out the actions and daily life plans of earlier Section 3.1.3, agents often must interact with each other or with the player(s). Interactions can be defined as a condition existing between two or more agents such that each exhibits one or more behaviors affecting the other during a finite period (Biddle 1979). Common interactions may include socializing and conversations, ordering coffee at a café, searching a residence on

security patrol, holding a council of elders meeting, and so on. We each have our own mental model of interaction types, many of which fall under a few general domains such as workplace, transactions, or classroom. The composition of an interaction model and the usage of its semantic layer to create a platform-independent model is now described.

Consider a villager agent "Ubadah Sabih" in some NonKin scenario. Ubadah is a Medical Supplies Importer by trade. There are a number of situations related to this occupation that this person might become involved with such as ordering or selling supplies. Ubadah is also a head of household, so within the context of being home, visitors have a number of actions afforded to them in interacting with him that they would not have when he was in his office or on the street. We have these various situations related to roles he fills, with each role potentially providing a different set of properties and actions to Ubadah.

Suppose a domain expert wished to create a platform-independent interaction model of searching a house in a law enforcement operation. In the case without interaction models, we would need to map out all the possible things that can happen in a search house and then apply these actions to every agent who may be involved in a house search. We would also need to switch off actions that are unrelated to the event currently happening. From an efficiency, modularity, and authoring perspective this would be both labor intensive and error prone, causing an explosion in lines of code required and maintenance costs. This lack of reusability discourages the authoring of complex interactions.

Interaction models avoid this dilemma by creating a "simulation within a simulation". Each interaction contains a full simulation model, complete with scenario and agent instances. Each agent is a proxy (representing an assumed role) for agents back in the main scenario with limited access to their state and models. Agents are passed in like parameters so that not only can they act like themselves as village entities, but have different capabilities depending on various situations they are in and what role they fill within that situation. This gives us a reusable container object to encapsulate various situations independent of who is involved.

Figure 6 - Interactions as sub-simulation models
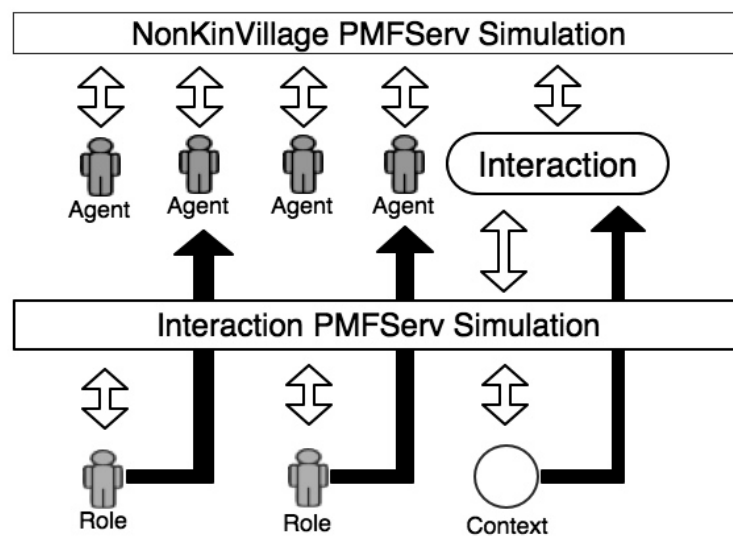
Interactions are made up of a number of interconnected but independent objects that together combine to create the interaction model itself. These components are:

- Interaction Scene -- The Interaction contains a Scene based off the one executing in the main Simulation Model. This scene contains only the agents and object that are relevant to the interaction.
- Internal Simulation -- In order to execute the Interaction's Internal Scene, it also must contain its own Simulation Model. This model is based off the same PIMM as the one the Interaction exists within.
- External Interaction -- The external interaction is the interaction's face to the rest of the world. It sits in the main simulation model alongside all other agents and objects in the village. Just like other perceivable objects it has actions that can be performed on it and information that can be drawn from it. For example, one of the default actions they support is "Join", which allows agents to join an interaction of their own volition.
- Internal Context -- The internal context sits inside the scene contained within the interaction's simulation. It acts as a proxy to the external interactions, permitting sharing of state information between the inside (within the interaction's simulation model) and outside (within the simulation that contains the interaction) worlds. For example it contains its own potential actions like "Leave" which allows an agent to exit an interaction.

- Roles -- Roles are where the bulk of the unique capabilities of interactions come from. Each exists as an incomplete agent that then proxies for an agent in the main scenario. This proxy causes requests for a subset of the role's models to actually return models on the main simulation agent instead. This allows the internal models of the normal agent to be visible on the proxy agent while it maintains its own perception and decision model thus tying it to the local simulation. Because of this the agents combine a local-level knowledge of their interaction with the global state and emotions of the main agent, creating a hybrid object that can act as an abstract role while maintaining the characteristics that make the agent unique in the village.

3.2.2) Social Rules: Transgression and Atonement

When interactions occur, social and cultural norms might be violated. Earlier in Section 3.1.2, we looked at the value systems that people (and agents) might use to appraise their situation. In addition to internally held values, most societies also have social obligations or cultural norms that individuals are expected to conform to. The social rules of a given culture need to be captured in a potential norms catalog or encyclopedia. An example of this in our Afghan village is shown in the left column of Figure 7. These norms are packaged in NonKin as a Social Norms Catalog that one can transgress against.

By "social transgression" we mean an offense an agent can commit against social norms or rules. When transgressions are committed they are instantiated as PMFServ objects representing an offense between an agent and another agent or group. Transgressions are usually created dynamically as a side effect of an action. For example, in NonKin Village, they are used to represent offensive behaviors executed by the user (often not intentional!) and to represent historical grievances of the village.

We utilize a simple, yet comprehensive taxonomy of the possible types of transgressions (Knight et al, 2008):

- Faux Pas - refers to the etiquette related transgressions (e.g. rude remarks, eating with left hand)
- Taboo - refers to any symbolistic related transgressions (e.g marrying your sister, defiling a holy relic)
- Materialistic - refers to destroying or stealing someone's possessions (e.g. vandalism, fraud, damaging property)
- Violence - refers to injuring, killing, or threatening bodily harm.

A transgression could have any single or combination of the above categories violated to different degrees. The example transgression highlighted in Figure 7 causes three categories of violations simultaneously.

Figure 7 - Overview of a Transgression Catalog Summarizing Social Rules of a Culture
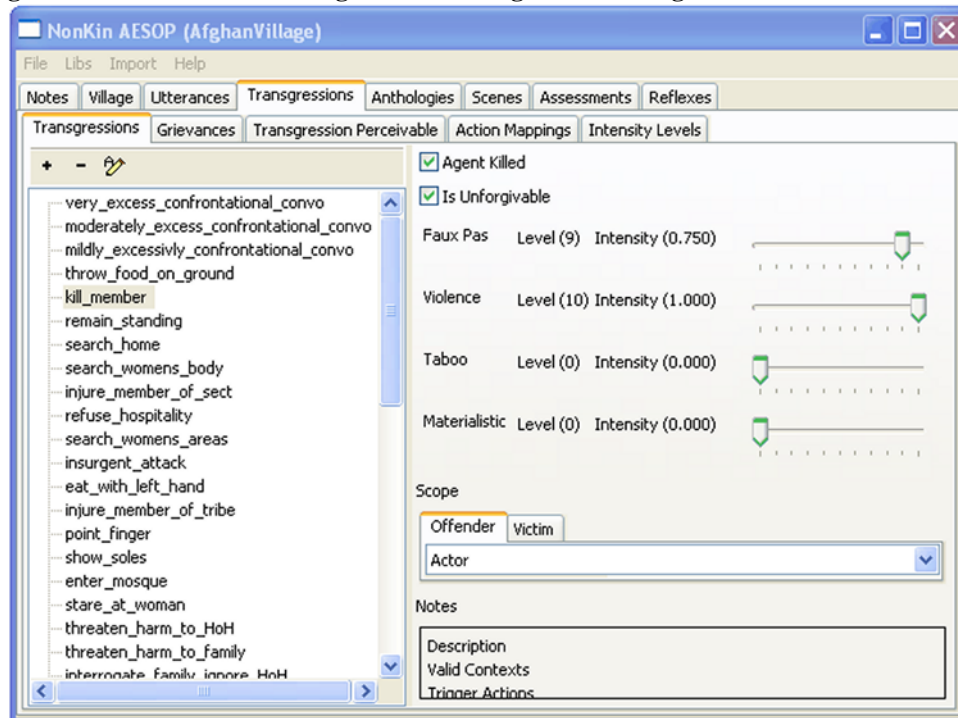
All transgressions have a transgressor, a set of victims, and a set of effects. Effects are the direct effects of the offending action, not the emotional activations on observers. Those events or effects are handled internally by the PMFserv agents. Beyond these basic properties, our transgression objects keep track of some

15

relations with the transgressor, relations with observers, properties of the effects, and relations between the transgressor and observers.

Each agent automatically perceives any transgression immediately after it's created. Transgressions are configured to invoke reactionary and usually negative emotions when perceived. Some transgressions are unforgivable, meaning that their emotional effects never lessen. However, most transgressions are forgivable and hence decay over time. The rate of forgiveness depends on the agent's "grudge factor". Other factors effecting forgiveness are apology and the removal of the negative effects of the transgression (e.g. returning stolen property, or a healed wound). In NonKin Village, serious offenses require a formal apology and possible compensation according to local customs and tribal law before forgiveness can occur. In this document, we refer to steps required to rectify an offense "Paths to Atonement".

3.2.3) Scenes Having Conversations and Transactions

Success in the NonKin environment depends on learning the human terrain, and to do that the player needs to talk to the village inhabitants. Many games implement conversation, but it is almost always based on finite-state branching structures or simple rule-based systems. The conversation system in NonKin needs to go well beyond this to support agent expressiveness about its physical and emotional state, and to allow the agent to take (verbal) actions based on the affordances of those actions.

To achieve this NonKin Depends on several structures:

- Utterance Catalog for each model in the collection
- Dialog graph for each generic, reusable type of interaction/transaction
- Interaction contexts (Scenes)

The Utterance Catalog is a library of statements that enable the agent to express something about a part of their internal models and sub-models (eg, physiology, emotions, etc) and/or their models of others they like, dislike, etc. Thus, without any further scripting an agent can state what's happening to and around it, like: I am very angry at you, I feel joyful about my daughter's health, or I am full. Each utterance has an Activation Rule associated with it. The activation rule is the logic that looks at the agent's internal models and decides the appropriateness of speaking the statement. The statements in the Utterance Catalog also have replacement rules, e.g. the agent can say " @subject is a member of @group".

It is important to note that the responses generated by the Utterance Catalog are dynamic, in the sense that they are generated from a static set of responses as an expression of the socio-cognitive models of the agent, which will differ from agent to agent, and which may change over time. So the Utterance Catalog is a place to find answers to questions. How can the player pose these questions? The answer is to provide them with dialog options, and to do that NonKin makes use of a graph-based Dialog system. These superficially resemble finite-state systems, but through modularization, contextualization and connection to the Utterance Catalog, they are used to provide needed structure and narrative "arc" to the player-agent conversations.

The main function of the Dialog graph is to give the player a list of statement options, directed at the currently engaged agent. The attributes of a statement include the verbal statement, a possible transgression, and an activation rule. These statements reside in a node of the graph, called a "beat". Each statement leads to another beat. Typically that destination beat contains a statement by the agent in response. The author of the conversation may choose for the agent to respond using one of four types of statement:

General	General statements are the default type and contain the markups common to all statement types
Expand-able	Expandable statements allow for a simple macro expansion which creates multiple general statement off an author created function. For instance a statement that expands on Groups, in a scenario with groups A,B, and C, would become 3 different statements, one for each group.
Jump	Jump statements pause the current conversation and jump control to a newly created conversation. This allows chaining of small reusable conversation fragments into larger structures.
Answer	Answer statements expose the utterance engine to the author, allowing agents to answer questions about themselves and others. These form the glue between Beats/Statements and the Utterance Catalog component of the Conversation Model.

Not all conversation can be adequately represented by the Utterance Catalog. To attempt that would mean giving the player a list of all possible conversation choices they might want to make, and link each into the Utterance Catalog. This is not practical. Instead context dictates what conversational snippets (dialog graphs) are available. Within each dialog graph, some statements may lead to other dialog graphs (through jump statements) or request the agent's opinion (through an answer statement). Meanwhile the general structure of the conversation can be maintained with general statements, used for player questions and agent responses
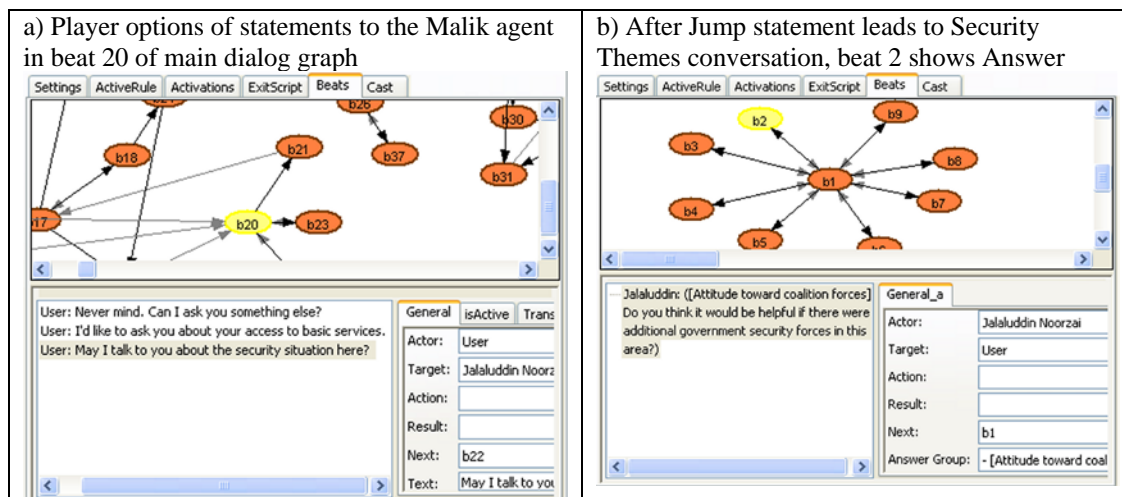
where no agent model consultation is required. Figure 8, from the AESOP scenario authoring tool, shows this sequence at work:

In Figure 8a the primary dialog graph "Tea ceremony" establishes the narrative arc of the conversation. When the player chooses the selected option, the agent responds with a Jump statement "what would you like to know?" (not pictured), which leads to the new dialog graph in Fig8b. There the player makes a choice in beat 1 (represented by the text box contents) which leads to an agent Answer statement accessing the appropriate response in the Utterance Catalog.

The final component in creating a useful engine for synthetic conversation is establishing the context of the conversation. That is the role of the NonKin "Scene": a "simulation within a simulation" (ie, earlier Figure 6). A scene is a reusable container in which agents and objects within the game can be mapped to roles within the scene. Agents can inhabit roles in multiple scenes simultaneously. Scenes may exist statically in the game or be created dynamically via player/agent actions.

Imagine a scene called "Dwelling". The player enters the home of an agent and steps into the Dwelling scene, and is mapped to the role of "visitor". The player now has available to him all the actions and narrative choices accorded to a visitor. The head of household's perception of the player is altered as well, compared to how they would perceive him outside the home. The player might choose to detain the agent. This action would dynamically create an "Arrest" scene, in which the player and targeted agent assume the pertinent roles of that scene.

Figure 8 – Dialog using a scripted scene (Malik has invited player in for tea)

a) Player options of statements to the Malik agent in beat 20 of main dialog graph

b) After Jump statement leads to Security Themes conversation, beat 2 shows Answer

Within any scene conversation can be initiated as an action, and by either a player or an agent. In our customary affordances based approach (Silverman et al. 2007), conversations are stored on agents like any other action (along with an Activation Rule and Activations, allowing agents to reason about how starting any particular conversation will impact them) and thus can be started at any time if an NPC agent decides to. Conversations also track the 'subject' of dialog, allowing the agents to have discussions relative to an arbitrary and changeable 3rd party. This concept is especially powerful when combined with utterances, allowing agents to not only talk about themselves but also speak about others.

The reasoning of NonKin agents is supported by three further models: The State Model, the Memory Model, and the Familiarity Metric. The State Model sits on an agent and tracks what is true about them such as age, sex, employment, coordinates, family and faction affiliations, etc. The Memory Model tracks (in a very simple way) the relationship between State Models, and tells us not only what the agent knows about other agents, but what the agent knows about what other agents know. Familiarity is computed and maintained by the agent as a summary of its models affecting its relationship with the user. Familiarity allows an agent to decide how much information to answer based off how comfortable they are with the listener.

3.3) Authoring the Artificial Society: NonKin Village

At the start of Section 3, Figure 2 presented the MDA and a model factory, most of whose components were described in Sections 3.0 and earlier 2.2. Our stated goal is to use the factory to author an artificial society for cultural training. Specifically, we want to create a village where trainees may learn its culture through immersion and problem discovery and resolution. More generally, our intent is to have a village generator. A

suite of PIMMs that may be used to produce many villages from differing cultures. We refer to this generator as NonKin Village.

If we were able to follow the MDA standard fully, we would have cultural domain specialists use very high level editors to produce the specs of a given village, fill in all the model parameters, complete the scenes and training vignettes and conversations. At present we do not yet have a PIMMM that satisfies this MDA standard. Instead, we rely on using the PMFserv IDE and AESOP to do this, editing tools which are more relevant for in-house modelers. With these tools we have fleshed out two villages to date – one in Iraq consisting of about 120 agents and that recreates a village that the US Marine Corp erected and staffed with Iraqi role players in the California desert. The other is a fictional Afghan village of about 50 agents in six families described in the next section.

The pipeline of the MDA process treats this instantiated village as a platform-independent model (PIM) which we next must migrate to platform-specific versions in order to provide immersive experiences for users. The PIM is converted into XML files through a process called baking, which aggregates all of the meta-models together into one file. This baked file can then be imported into our generator, called NonKin, which is capable of creating concrete instances of the meta-models, the PIM via the execution layer (see right side of earlier Figure 2).

Up to this point, our models and meta-models have been completely platform independent. They could easily be adapted to work in any environment. It is at the client plug-in stage that we encounter platform specific models. The client plug-in expands upon the platform independent models provided by the client API and creates platform specific versions that can be integrated into the plug-ins respective client platform. We illustrate one example client plug-in in the next section. This uses VBS2, a 3D game engine that successfully displays our agents acting and conversing through their counterparts that are animated by the 3D world.

We should conclude by mentioning that the bridge provides a number of services required for our village to work smoothly with a real-time 3D world. In PMFserv, a tick consists of an agent performing an OODA loop, or perceiving the world, reasoning about their afforded actions, and taking the decided action. This happens in a turn-based approach with each agent reasoning and acting one after the other. This works well for a simulation world, but fails to represent actions realistically in a continuous play 3D interactive environment where actions take real amounts of time (e.g. walking to a location). Occasionally, an action dictated by NonKin will be incompletable in the 3D world due to path-finding restrictions or physics-based interruptions. Indeed, we routinely use a 3D engine's low-level AI to perform actions such as path-planning, obstacle avoidance, and animations.

In order to include these additional world constraints, we push action completion and result choice on to the client, or 3D engine. Any client of the NonKin server is responsible for telling the server when and how an agent completed an action, and only when the client is satisfied that the agent has successfully completed its action will the results be propagated throughout to the simulation side (NonKin). Also, the rule that all agents must sequentially take actions is removed. As various actions will take different amounts of time, agent decisions are queued asynchronously (i.e. clients ask for the agent's next decision once it has completed the current one or another action has become more attractive).

All implementations of our 3D clients have a potential list of actions that the client is capable of performing (and animating). These actions are usually ones that require a time and/or spatial component (e.g. moving to a location, praying, attacking a target, socializing). Behavioral markup language is an emerging XML standard designed to be used when communicating information about the behavior of realistic, emotional agents (Vilhjalmsson et al. 2007). It is designed to be embedded within a larger xml document and passed as metadata along with things such as conversation nodes or actions. It describes the exact motion and action that a character can take in a 3D simulation; eyebrow movement, mouth shape, and hand gestures, which allow the character model to convey emotion that would be recognizable by a human observer.

The inclusion of BML into our bridge is in its infancy. We currently use it to convey behavior intention (e.g. the name of the emotion to display) rather than direct physical characteristics that denote that emotion (wide eyes for happiness, furrow for distress). However in later iterations, a full fledged behavior representation model may be developed that would be capable of mapping emotion combinations (10% envy, 30% joy, 5% distress etc) to exact behavior (0.1 mouth curl up, eyelids 0.5 down, eyebrows 0.2 pulled in).

BML is designed so that the receiving engine can use as much of it as it deems necessary. For example, an engine with full-fledged procedural animations (rigged character models that can perform dynamic and unique animations) could leverage every bit of information and specifically place the different model structures (mouth, hands, eyebrows etc) in the positions indicated by the BML. However, an engine that only supported discrete animations (rigged character with preset animations, like "run" "walk" "attack high" etc) would only look at the name of the emotion and perhaps its severity, and call a specific animation based on that.

4 Gameplay in the Platform Specific Model (PSM) of NonKin Village

The resultant Platform Specific Model (PSM) for NonKin is shown in this section for a fictional Afghan village called Baja. Baja is located in the rural Nad Ali district of the Helmand River valley. The founding of Baja dates from the development of the irrigation system in the late 50's. It is a small grouping of about 20 residences, with the total number of families being 6, divided between the Achekzai (4 families) and Noorzai (2 families) clans of the Durrani Pashtun tribe.

The village is situated on a small knoll in the agricultural plain. Each family occupies a mud-walled compound, each containing 4-6 dwellings and other small structures. The main water source is a pit well on the edge of the village. The local irrigation system was a project organized by the USA in the 1950s. Consequently the locals are not unfamiliar with or naturally ill-disposed to Americans. The system is in disrepair.

The two main groups have complicated relationships with each other and with rulers past and present. The Noorzai own marginally the most land, and are invested in poppy cultivation. The Taliban banned poppy in 2000 and the Noorzai were consequently helpful in ushering out the Taliban in 2002, but have since become intertwined with them through the opium trade. The Achekzai are a mixed group, divided between supporters of the communists during the Soviet occupation and the mujahidin resistance, and later some became Taliban. Some are Afghan national army soldiers. Historical grievances over land persist between the two groups.

The non-local groups in the village are the Marines, the Afghan National Army (ANA). The ANA operate alongside the Marines and are based from the same FOB, about 2 miles distant from the village. The attitudes towards the ANA is mixed, but generally positive, with the exception of the few hardcore Taliban supporters. Your initial tasks as the player are to go patrol the village, inventory the population, and build rapport.

4.1) Rapport Building

Consider how gameplay proceeds in a NonKin Village. The goal is for small unit training (eg, squad, platoon), in general, and small unit leader training, in particular. The idea is to successfully get through each of the three phases of the counter-insurgency (COIN) theory as espoused in Army FM 3-24 – survey the human terrain and learn all the stakeholders, coopt the agenda/befriend the locals, and convert the local institutions and governance to a self-sustaining, equitable, peaceful one (see Petraeus et al. 2006). To guide the player, there is a Captain who assigns missions in each of these three phases (levels) of the game. In Figure 9a, he is tasking the player with a standard Phase I mission – to go on patrol and learn who is out there and build rapport. There is also after action feedback, but more on that shortly.

In Figure 9b we see the platoon leader taking his squad out on an afternoon patrol, a presence mission that increases the local villagers' sense that there is security in the area and that permits the Marines to build rapport and learn what are the issues in the village between the diverse stakeholders. In Figure 9c, the player arrives at the house of the Malik (leader) of the local Achekzai clan. The player is accompanied by his other 8 squad members who stay outside to guard the compound, by an interpreter, and by an Afghan National Army (ANA) member. Another goal is to help train the ANA and foster their seeing a model of how soldiers treat civilians. In this scenario, all these other agents are NPCs, but the environment is multi-player and these could be played by other marines or soldiers.

In Figure 9c and 9d we see shreds of the conversation the player is having with the Malik. In the former the player is invited for tea: an opportunity to build rapport and familiarity. The latter shows an example of the Malik exposing his inner models and being queried about his internal state. These make use of the utterance catalogs so the Malik can explain his model parameters.

The outcome of both of these activities – having tea and inquiring about the agent's condition – lead to an increase in the Malik's familiarity with and trust in the player. As discussed in Section 3.1.1, there are many dimensions of interaction and trust/rapport building in a socio-cognitive model suite. The care action directly alters the relationship from neutral to friend. Likewise, the time spent having tea shifts the player to a high level of familiarity in this Malik's eyes since this is the 3rd time the player has stopped by for tea.
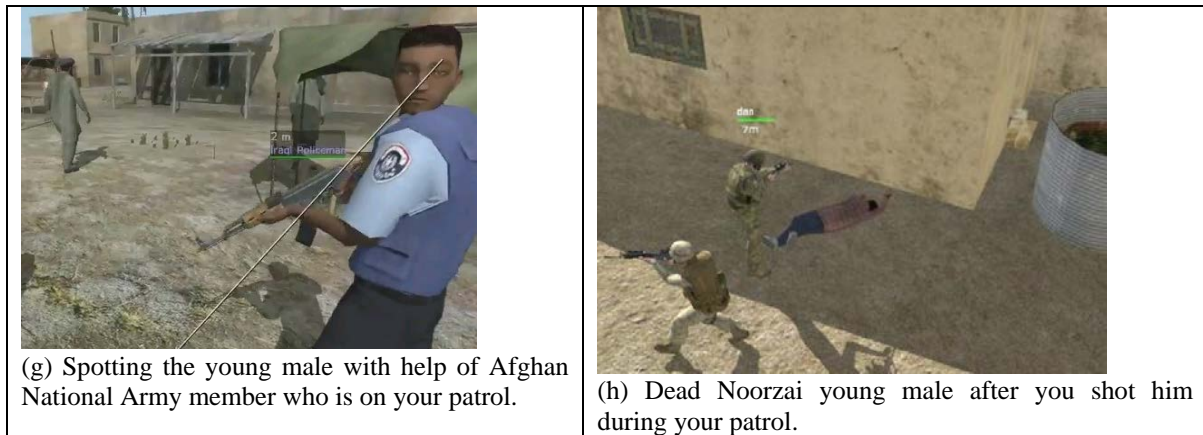
A secondary effect also occurs when the player departs from the Malik's compound. Specifically, FactionSim runs the relationship and familiarity changes up to the group level and propagates them across groups and back down to group members. The first of these shifts is in group to group alignment. Since the leaders of the Achekzai and Platoon Groups are now friends, the group alignment also shifts from neutral to friend. This is further strengthened since the Platoon is promising to provide other services as well that will help the village (security, projects to improve the irrigation, and education). If these are done in a way to provide jobs for locals this will shift the Player to Phase II (coopt the agenda) of the game. If the player gets the locals to in fact take over managing and running such projects in a way that is equitable to both tribal clans, this would be

the end-game and the player would have successfully reached Phase III. But that is many hours of play yet to happen. Returning to the current moment, a second effect of this recent cup of tea with the Malik is the increase in familiarity to a high level. Since the Malik has a respected position in his clan, this causes the familiarity of all other clan members to also be increased as discussed in the network algorithms of earlier Section 3.1. Of course it might also have some adverse impacts on alignment with groups that are adversaries of the Malik's group.

Figure 9 – Scenes from Gameplay in NonKin Village

(a) commander giving you orders to go on patrol

(b) Going on patrol and stopping at the Malik's compound

(c) Malik inviting you in and offering tea. Dialog graph offers the player three options.

(d) Malik responds to about his health. Displays response from Utterance Catalog.

(e) On patrol the next day, seeing the bustling market

(f) Shopkeeper of Achekzai clan warns player about Noorzai young male who is possibly planting IED

(g) Spotting the young male with help of Afghan National Army member who is on your patrol.

(h) Dead Noorzai young male after you shot him during your patrol.

The platoon returns to their FOB for the evening and the next day goes out on patrol once again. This time they head toward the market area. As Figure 9e shows, there is a lot of activity with many villagers there carrying out daily life, working the booths, shopping for supplies, and so on. In earlier Sect. 2.1.3 and 2.2.1, we explored the many routines and plan sequences that agents may opt to carry out. Here we see a number of agents crossing paths. None of these are strictly pre-scripted, but are the result of each agent assessing his or her needs and deciding to carry out the highest utility plan. Since most agents sleep, eat, pray, go to work, and/or carry out errands at roughly the same time of day, it is not unusual to see some bustle at key times. In fact this is a pattern the player should learn to recognize. The lack of bustle at a time when it should be there is often a warning sign of some impending calamity or attack.

In "walking the beat" into the market area, the player has the opportunity to fraternize and further build rapport (or commit transgressions) with villagers he may meet there. In Figure 9f, the player is flagged over by a shopkeeper. The shopkeeper is another village elder and a respected person in his own right. The effect of the relationship and familiarity from the visit to the Malik are evident in the trust and alignment assumptions of the Shopkeeper. He assumes the Player is friend and ally and will provide security. Due to a night letter he received, he is concerned about the immediate safety of his shop and conveys that, naming a suspicious member of the village. Here the transgression model also influenced his choice of utterances. While the young Noorzai male that he names is also Pashtun, he is of another clan and, further, he is disrespectful of the elders since he has joined up with the Taliban who hope to take over running things. Having his ally stop the young Norzai male would thus lead to multiple benefits for the shopkeeper and this is the highest utility utterance for him to select at this point in the game.

Our player up to this point in the game appears to have been fairly even-tempered and succeeded at building rapport with several villager elders and others. Sadly though, in his efforts to detain and interrogate the young Noorzai male, things deteriorate quickly, our player thinks he has to defend himself and he winds up shooting and killing the suspect (see Figure 9f). This event triggers worst case transgressions (violence) which propagate across the Noorzai group. At this point the player has ruined his chances at befriending the Noorzai and advancing the game. There are atonement pathways for causing a death, but they are not simple.

Figure 10 - Player's Rucksack Showing The Assessment of his Training Level Thus Far.
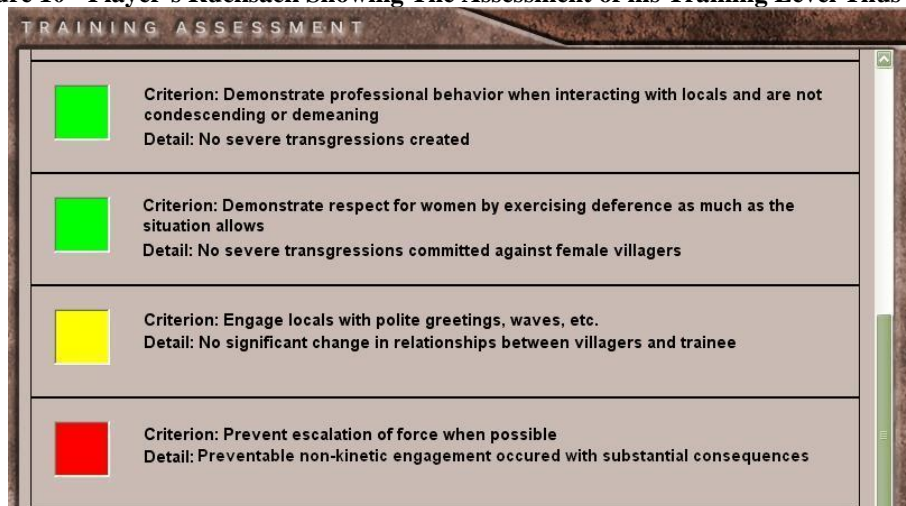
4.2 Rucksack Tool and Training Level Assessment

After completing an assigned mission, the player is presented with an assessment of his/her performance based on criteria defined by the trainer. A criterion consists of two components: events or changes of interest during game play, and the grading 'rubric' to assess them. A collection of criteria that forms the assessment is executed by running the rules against the simulation from mission start to completion. For example, in the mission we have worked through in this paper, the assessment examines seven criteria, six from the U.S. Marines' Tactical Cultural Awareness Training (TCAT) series: eg., see Solmani et al (2005). These assesses cultural awareness and rapport building. There is also an assessment rule for inappropriate escalation of force. Based on the player's actions and simulation changes, the scorecard (Fig. 10) provides a grade that follows the Marine scheme: green for 'Trained', yellow for 'Partially Trained' and red for 'Ineffective'. Optional detail information may give additional personalized feedback to the trainee. Since the player fired on a civilian in an unprovoked manner, the escalation of force obviously returns an ineffective grade.

5 Conclusions and Next Steps

This article has described a capability for producing virtual, immersive villages that enhances agent realism by synthesizing a wide array of social science models. Called NonKin Village, this is a gameworld generator that brings life to agents of all factions in sort of a SimCity style of play although it supports street level interaction and dialog with agents to learn their issues, needs, grievances, and alignments. The idea is to create realistic cultures and ethnic groups so one can learn and train about their cultural concerns and inter-factional conflicts. More broadly, this capability is potentially applicable to any cross-cultural training concern.

One result of this approach is that it leads to rich socio-cognitive agents – agents that are very complex. The overall objective of this research (sect. 1.1) was to learn how to use standards and modern software design patterns to help manage this complexity so as to ensure reusability, replace-ability, and maintainability of the models collection. To support this objective, we examined three research goals: to explore how diverse social science theories and models could best be used (goal 1) and synthesized (goal 2) to drive agent behaviors so that autonomous socio-cognitive agents would populate the village society. Their micro-decision making would lead to emergent macro-behaviors relevant to their cultural norms, value systems, and collective perceptions. Also of interest (goal 3) was to synthesize added components such as diverse 3D viewers, a training assessment module, etc. A seeming paradox and possibly the main contribution of this research was to notice that as more models were synthesized across the social sciences and as agent complexity grew, the easier it became for the agent to explain its world and the situations it found itself in. Early in this research, we only had internal mind-body models and the agent could not discuss social issues. Subsequently, we added numerous social science models. Because of this, the agents can now explain their dilemmas in terms not just of psychology, but also from from perspectives of sociology, politics, economics, and other sciences. They also can explain their situation relative to the range of organizations and networks that impact them – kinship, ethnic, ego, commercial, religious, and so on.

As we encountered and benefited from this complexity paradox effect we learned how to move beyond the current practice in immersive worlds (Sect. 2). That practice is largely based on either human-played avatars or simple finite state machines that one scripts by hand (1st generation agents). There are also a growing number of cases of 2nd generation agent approaches in use that rely on one or a few narrowly focused social science models. A broad ranging model library such as we implemented represents a move to 3rd generation agents and is a first, as far as we know of. Certainly we have found no other immersive agent cases to guide us, and borrowed from the model driven (and related) standards to derive the framework presented to support the 3 goals.

As a result, there are a host of further research directions and challenges that the framework for NonKin opens up as we spiral toward the next version. We mention but a few of these here. In striving to satisfy the research goals, we encountered an explosion of models, straddling both product models (ie, cognitive agents, social organizations, and artifact affordances) and process models (ie, interactions, transgressions, and conversations). In order to manage this model proliferation problem, we turned to several model-based software design patterns including the Model Driven Architecture (MDA) standard, the model factory pattern, and others. Among other things, the MDA standard helped us to better organize our code base as platform independent models (PIMs). This facilitates separating the modeling code from the 3D platforms that handle the immersion, and thus makes the behavior models more maintainable and better reuseable for plugging in to drive the agents of other 3D worlds or platforms. The standard further calls for meta-level PIMs (or PIMMs) that can be instantiated to create a PIM of a given society or village. Segmenting the behavioral models into a PIMM server (and PIMM programming language) also facilitates reaching our third goal of federating or integrating added components – eg, a developer's higher level editor (PIMMM), any 3D viewer of the world, a training assessment module, and so on. As a test of this, we did successfully plug in to both the VBS2 platform and a

Rucksack module for assessment as illustrated in Sect. 4. We further tested this by plugging in to a different 3D world not shown there (ie, IRLICHT).

On this last point, we are evolving taxonomies and markups for the range of models mentioned in this paper – transgressions, events, interactions, etc. – that will support migrating the village to new settings. In addition, our goal was to instantiate sufficient content in the models of the NonKin generator so that it will be largely a matter of turning things off when trying to set up NonKin for a new village or region of the world. Our approach for doing that relies on filling NonKin with the current best-of-breed social science theories about descriptive and profiling methods, realizing that these should be updated and replaced downstream as better models are discovered. Thus the PMFserv architecture is an extensible one that holds many dozens of default models that can be unplugged and replaced. There is much to be done, but ultimately, we will add editors that permit non-programmers to readily turn off features not needed and to adjust others to the setting of interest. This is consistent with the MDA standard and factory pattern, which both suggest that a non-programmer's editor suite (PIMMM) should be added that allows domain specialists to provide specs of the artificial society and training dilemmas and goals.

As a final note, the validity of the current collection of models is not entirely testable, but efforts were mentioned where the model parameterizations have been statistically tested and shown to correlate over 80% with specific community behaviors in North and South East Asia, Africa, and the Mid-East. Conducting validity assessments for any given artificial society is an ongoing challenge of this type of research.

References
1. Bezivin, J. (2005). On the unification power of models. *Software and System Modeling*, 4(2):171-188.
2. Bickmore, T., Cassell, J. (2005). Social Dialogue with Embodied Conversational Agents. In: J. van Kuppevelt et al. (eds): *Advances in Natural Multimodal Dialogue Systems*, v30 pp. 23-54, Berlin: Springer-Verlag.
3. Biddle, B.J. (1979). *Role Theory: Expectations, Identities, and Behaviors*. New York: Academic Press.
4. Bjorkman, E. A., Barry, P. S., & Tyler, J. G. (2001). Results of the Common Human Behavior Representation and Interchange System (CHRIS) Workshop. (Paper 01F-SIW-117). Proceedings of the Fall Simulation Interoperability Workshop, Orlando, Florida.
5. Bostan, B. (2009) "Requirements analysis of presence: Insights from a RPG game." *Computers in Entertainment (CIE).* 7(1), 1-17. Feb 2009. ACM.
6. Cohen, T, Gil, J (2007). "Better Construction with Factories". J. of Object Technolog y, July, 6, 6, 109-130
7. Cornwell, J., O'Brien, K., Silverman, BG, Toth, J., "Affordance Theory for Improving the Rapid Generation, Composability, and Reusability of Synthetic Agents and Objects" *12th Conf on Behavior Representation in Modeling and Simulation (BRIMS, formerly CGF)*, SISO, 12-15 May 2003.
8. Edmonds, B. Moss, S. (2005) From KISS to KIDS – an 'anti-simplistic' modeling approach. In P. Davidsson et al. (Eds.): Multi Agent Based Simulation 2004. Springer, *LNAI*, **3415**:130–144.
9. Graesser, A, King, B, (2008). "Technology Based Training," in JJ. Blascovich, CR. Hartel (eds), *Human Behavior in Military Contexts*, Washington: National Academy Press.
10. Hahn, C., Madrigal-Mora, C., Fischer, K. (2009). A platform-independent metamodel for multiagent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 18:239-266.
11. ICT (2010), *Projects: Games & Simulation*, Playa Vista: USC/Institute for Creative Technologies, website at http://ict.usc.edu/projects/games_simulation
12. Kelso, M., Weyhrauch, P., Bates, J. (1993). Dramatic Presence. *PRESENCE*, 2(1): 1-15.
13. Knight, K., et al.: Transgression and Atonement. In: Dignum, V. (ed.) (2008). *AAAI Workshop Proc.*, Chicago.
14. Lavine, N., Peters, S., Napravnik, L., & Hoagland, D. (2002). An advanced software architecture for behavioral representation within computer generated forces. In *11th Conference on CGF & BR*, Orlando, FL: SISO, May, pp. 169-180.
15. Marsella, SC., Pynadath, DV., and Read, SJ. PsychSim: Agent-based modeling of social interactions and influence. In *Proceedings of the International Conference on Cognitive Modeling*, pp. 243-248, 2004.
16. Mateas, M. and Stern, A. (2003). Façade: an experiment in building a fully-realized interactive drama, *Game Developers Conference (GDC '03),* San Jose, CA, USA, March 4 – 8, 2003.
17. Mateas, M., Stern, A. (2003). Integrating plot, character, and natural language processing in the interactive drama Façade. In: *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*.
18. Ortony, A., Clore, G.L., & Collins, A. (1988). *The cognitive structure of emotions*. Cambridge: Cambridge University Press.

19. Object Management Group (OMG) (2003). MDA guide version 1.0.1, Document omg/03-06-01, June 2003,
20. Petraeus, H., Nagl, J. Amos, J., (2006). *U.S. Army/Marine Corps Counterinsurgency Field Manual.* Available: Amazon.com
21. Pew, R.W., & Mavor, A.S., (1998). *Modeling human and organizational behavior: Application to military simulation.* Washington, DC: National Academy Press.
22. Pietrocola, D., Silverman, B.G. (2010) : Taxonomy and Method for Handling Large and Diverse Sets of Interactive Objects for Immersive Environments. In: *19th Annual Conference on Behavior Representation in Modeling Simulation*, Charleston, SC
23. Rabin, S. (2008). *AI Game Programming Wisdom 4*. Hingham, Massachusetts: Charles River Media.
24. Salmoni B, Holmes-Eber, P, (2008), *Operational Cultures for the Warfighter: Principles and Applications*. Quantico: Marine Corp Univ Press
25. Schelling, T. (1978). *Micromotives and Macrobehaviors*. New York: W.W. Norton & Company.
26. Schmidt, D. (2006). Model-driven engineering. *IEEE Computer Society*, Feb: 25-31.
27. Sharma, M., Ontanon, S., Mehta, M., Ram A. (2010). Drama Management and Player Modeling for Interactive Fiction Games. Computational Intelligence, 26(2) 183-211.
28. Silverman, BG, Johns, M, Cornwell, J, O'Brien, K, 2006a, "Human Behavior Models for Agents in Simulators and Games: Part I – Enabling Science with PMFserv," *PRESENCE*, April, 15:2, 139-162.
29. Silverman, B.G., Bharathy, G.K., Nye, B., Eidelson, 2007. "Modeling Factions for 'Effects Based Operations': Part I – Leader and Follower Behaviors", *J. Comp'l & Math'l Organization Theory*. Dec.
30. Silverman, B.G., Johns, M., Weaver, R., Mosley, J. (2003). Authoring Edutainment Stories for Online Players (AESOP). In: *Internat'l Conference on Virtual Storytelling*, Toulouse, FR: Springer.
31. Sun, R. (2004). *Cognition and Multi-Agent Action*. Cambridge: Cambridge Univ. Press
32. Surhone, LM, Timpledon, MT, Marseken, SF (Eds), *NetLogo: Multi-Agent System, Programming Language, Integrated Development Environment, Emergence, Science*, Mauritius: Betascript.
33. Vilhjalmsson, H., Cantelmo, N, Cassell, J., et al. (2007). The Behavior Markup Language: Recent Developments and Challenges. In: C. Pelachaud et al. (Eds.): *Intelligent Virtual Agents 2007*, LNAI 4722, pp. 99–111, Berlin: Springer-Verlag.
34. Zacharias, G.L., MacMillan, J., and Van Hemel, S.B. (Eds.) (2008). *Behavior Modeling and Simulation: From Individuals to Societies*. Washington, DC: National Academies Press.

Repeat of Figure 5