

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/286810296>

Virtual Environments for Computational and Analytical Modeling: A Telemedicine Application

Article in *Lecture Notes in Business Information Processing* · January 2012

DOI: 10.1007/978-3-642-27612-5_8

CITATIONS

0

READS

19

4 authors, including:



Tung Bui

University of Hawai'i System

143 PUBLICATIONS 2,314 CITATIONS

SEE PROFILE



Alexandre Gachet

Université de Fribourg

25 PUBLICATIONS 190 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



A Neural-Network-Based Behavioral Theory of Tank Commanders [View project](#)



Social Proof and EWOM [View project](#)

Virtual Environments for Computational and Analytical Modeling: A Telemedicine Application^{*}

Tung Bui¹, Daniel Dolk², Alexandre Gachet¹, and Hans-Jürgen Sebastian³

¹University of Hawaii
{Tung.Bui, Gachet}@hawaii.edu

²Naval Postgraduate School
drdolk@nps.edu

³Deutsche Post Chair of Optimization of Distribution Networks
RWTH Aachen University
Sebastian@or.rwth-aachen.de

Abstract. Virtualization is commonly known in computer science as an abstraction technique of computer resources – physical platforms and resources – so that applications or end-users can seamlessly interact with these resources without the needs to deal with physical requirements. Going beyond the simulation of computer environments and resources, this paper proposes a paradigm for designing complex information systems based upon the concept of virtual modeling. The idea is to allow modelers use a virtual environment that is composed of real modeling platforms to replicate complex real problems, and explore new and virtual problems that might have high potential for real-life applications. Modeling here is not just an effort to find (new) solutions to an existing problem, rather it is also a discovery process seeking to create new (problems). We see this approach as vital in addressing the applications emerging from service science, management and engineering (SSME), which will rely upon computational modeling approaches as much, if not more, than traditional supply-chain based analytical modeling. We illustrate our design methodology with a telemedicine application using Brahms, a multi-agent programming language developed by the NASA.

Keywords: Information Systems Modeling, simulation and decision support, service system, SSME, virtual environments, telemedicine.

1 Introduction

With the advent of high performance computing systems, modeling has become both a process of explanation (e.g., knowledge representation, business process engineering and re-engineering formally coded in executable code) and a process of exploration (e.g., action-driven artificial intelligence software, decision support systems, agent-based simulation). Carley [3] argues that organizational computing has

^{*} The authors would like to thank Stephen Kimbrough, Wharton School, and Murray Turoff, NJIT, for their inspirational work on virtuality and their suggestions for this paper.

evolved toward an inquiry process related to information, knowledge and computation, and this process has led to a wide range of advanced I.S. applications (e.g., flight simulators, remotely-controlled robots for task-specific applications, and virtual classrooms). Within this expanded scope, modeling can be viewed as a process that not only replicates models of realities but is also capable of creating life-like situations that appear real, yet have no correspondence in reality [29,37]. For example, a computerized business game is modeled after the realities of competition. However, as it becomes a tool for exploring new competitive strategies, the emerging rules of virtual competition might nevertheless lead to real actions. Another classic example is SimCity.

In this paper, we discuss the evolution of both virtuality and modeling in the context of modeling for problem solving with a focus on services science, management and engineering (SSME). We argue that virtuality has evolved to a state in which, only under certain modeling objectives and conditions, should reality be used as a reference. We also contend that modeling under virtual environments requires a new paradigm that views computation as both an experimentation and explanation process. We advocate a paradigm shift in developing a computational model. A negotiated reality – an application environment that situates within the reality-virtuality continuum – requires both the modeler and the users of the model to: (i) use a formal language for describing and explaining the functions and behaviors of a phenomenon and its environment, and (ii) use the power of information technologies to search and experiment with new environments that would best address the problem at hand. Negotiated reality may be a particularly relevant context in which to consider SSME applications since service activities require a much higher degree of cooperation and coordination between provider and consumer than is typically the case with commodity-based economic transactions. We will elaborate this thesis by emphasizing the concepts of computational experimentation and computational explanation.

The paper is organized as follows. Section 2 discusses the evolution of the concept of virtuality. It serves as a foundation to explore in Section 3 how computational modeling and problem solving converge thanks to the joint consideration of reality and virtuality (Fig. 1). In Section 4, we propose a software modeling methodology called Virtual Environment for Computational and Analytical Modeling (VECAM) that applies model management design principles for analytical modeling to generate requirements desiderata for computational modeling environments. Through the discussion of a telemedicine service application, we demonstrate how VECAM can be implemented as a design methodology, one that is particularly relevant to service science, management and engineering (SSME).

2 Evolution of Virtuality

We examine the concept of virtuality in progressive stages, looking first at the role of modeling and simulation in scientific inquiry to see how virtual environments are becoming more integral to that process. We then posit a virtuality spectrum which

includes the key concept of *negotiated reality* which we see as one of the distinguishing aspects of services-based applications. We discuss how negotiated reality in concert with virtual environments support SSME concepts and applications.

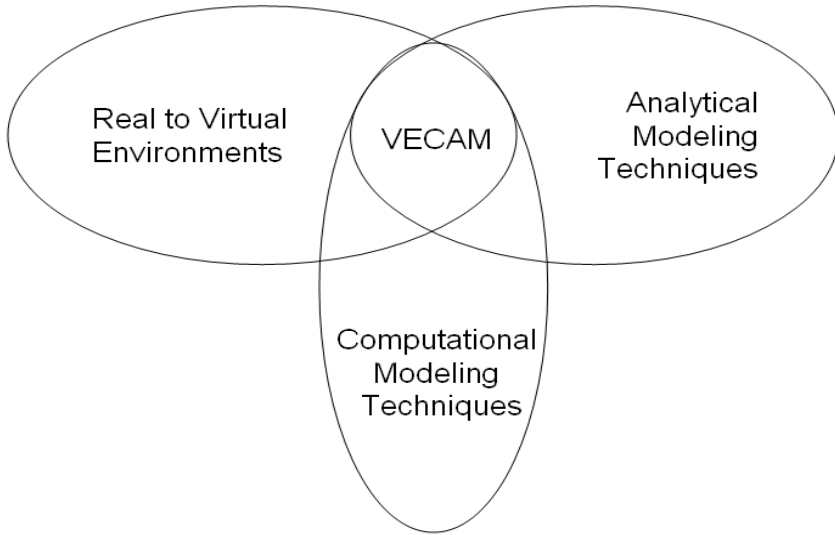


Fig. 1. VECAM framework

2.1 Modeling and Simulation in the Scientific Method

We begin by examining the role of modeling and simulation in the scientific method as characterized by [16]. Fig. 2a shows the traditional interplay between theory and experiment which has earmarked science since the beginning of the Enlightenment until the advent of digital computers. Fig. 2b shows the emergence of modeling and simulation in this process as a result of the use of digital computing. In this scenario, a Model is a formal representation of reality which implements a Theory, and a Simulation elicits the behavior of the Model, usually over time, thus corresponding to an Experiment. Models in this context have largely been what we call analytical models in that they are primarily mathematical in nature, for example systems of partial differential equations, mathematical programming, and the like. We use the term simulation in the larger sense of an experimental design for solving and analyzing a model using various forms of sensitivity analysis and/or goal-seeking, as opposed to the more specific context of various simulation technologies such as discrete or continuous event simulations. Not all models are dynamic, for example, a mathematical programming model for determining the optimal location of a warehouse is spatially, rather than temporally, oriented. Nevertheless, it makes sense to think of an experimental design, or simulation, for testing and analyzing such a model even though it may be time independent.



Figure 2a. Scientific method, 15th-20th centuries

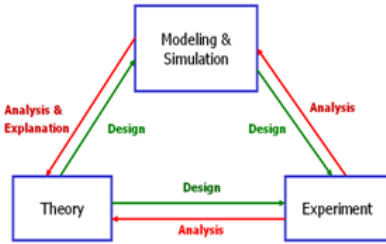


Figure 2b. Scientific method, 1950 - present

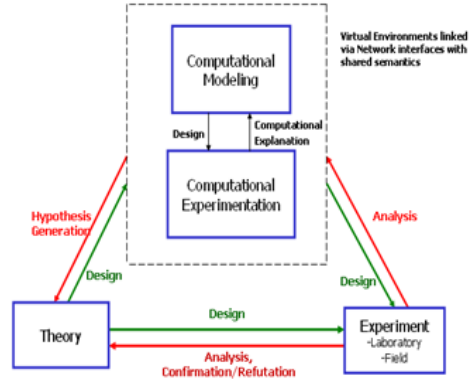


Figure 2c. Scientific method, emerging 21st century

Fig. 2. Role of modeling and simulation in scientific inquiry [16]

Fig. 2c shows the next stage in the scientific method which moves beyond analytical modeling and squarely into the arena of virtual environments. In this scenario, Hamming envisions networks of virtual environments that eventually can be linked via shared semantics. This is model integration in the large, and relies heavily upon computational modeling and the methodology of computational experimentation for the creation and maintenance of virtual environments. We take pains to emphasize that analytical modeling is not rendered obsolete in this context, but rather subsumed and integrated under the umbrella of computational modeling. We will indicate ways in which this can be achieved, specifically through a VECAM architecture.

2.2 The Virtuality – Reality Continuum

According to the Oxford English Dictionary, “virtual reality” is a state or an object that is “not physically existing, but made by software to appear to do so from the point of view of the program or user”. This definition depicts a fundamental characteristic of virtuality. According to Turoff, it is a process of *negotiated reality* in which the artifacts of computer systems are adopted by their users as “agreed-upon” reality [37]. In this process, the reality becomes simply more and more artificial [10] while the virtuality becomes more and more real. The virtuality-reality continuum can be explained as a constant search for truth using analytical, experiential, conflictual, synthetic and pragmatic approaches [28].

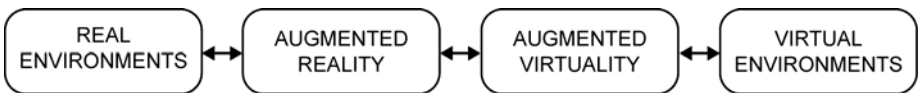


Fig. 3. The evolution of virtuality - from real to virtual environments

Table 1. The Reality-Virtuality continuum with some application examples

REALITY-VIRTUALITY CONTINUUM	APPLICATIONS	CHARACTERISTICS	OBJECTIVES	EXAMPLES
AUGMENTED REALITY	Simulation	Replicate a business behavior	Generate lifelike behavior in either cost-effective or accelerated simulated environment	Queuing applications; flight simulator; technological forecasting; futures research
AUGMENTED VIRTUALITY	e-environments	Create a task in an electronic environment that is different than reality	Use agent-based systems for complex applications on the Internet	e-classroom, Internet-supported distributed teamwork
VIRTUAL ENVIRONMENTS	Virtual communities (e.g., users' groups)	Social connectedness in cyberspace	Create new communities (e.g., virtual cities with avatars)	MUD (Multi-User Dungeons), MOO (Object-oriented MUDs); Second Life; Facebook groups

Yet, *negotiated reality* implies a number of implicit but fundamental assumptions about modeling that depend on the way the modeler approaches the “reality-virtuality” continuum. As evidenced by several recent systems [22], the reality-virtuality continuum spans from real environments to augmented reality to augmented virtuality and to virtual environments [27] (see Fig. 3 and Table 1). One contrasting dimension of this continuum is the unity of space, time and place in a real environment compared to a dislocation by information and communication technologies and the omnipresence of the WWW in a virtual environment (for example, the traditional classroom versus the distributed e-classroom).

There are two notions that may be relevant here in the discussion of real and virtual environments: *coherence* and *correspondence*. Coherence is a measure of how well a virtual environment holds together internally. Does it exhibit consistent behavior

within the boundaries of its environment? Is it believable by the users who interact with it? Does it conform to its internal “laws”? A computer game such as SimCity or a combat simulation would have coherence as a desirable property. Correspondence refers to how well the virtual environment corresponds to a real world counterpart. This is closely related to the concept of “external validation”. For example, if we build a synthetic economy, we may want that economy to emulate the real world economy to a specified degree of verisimilitude.

The same is true for many operations research and management science (OR/MS) models, including simulations (e.g., business games). Virtual worlds could also be categorized by the degree of correspondence that is required of them. A recreational computer game such as SimCity may have a low correspondence requirement. Other virtual worlds such as flight simulators may have a partial correspondence requirement that gives the user a sense of the “real world” but not necessarily align completely. An OR model on the other hand may require as complete a correspondence as necessary.

This issue arises in the field of artificial life (AL), for example. AL researchers build elegant virtual environments which mimic the outcomes of evolutionary and biological processes quite impressively. However, they are continually criticized by the scientific establishment for having little, or no, correspondence. The coherence-correspondence relationship has become intertwined and pervasive in the information-based world in which we live (Table 2), and scientists in a number of fields have captured it from a variety of perspectives (Tables 3 and 4).

Kimbrough [21] argues that to deal with complex problem solving, we need a modeling language that is capable of explaining the real world (i.e., extract the basic

Table 2. Characteristics of virtuality

CHARACTERISTICS OF VIRTUALITY	ATTRIBUTES	EXAMPLE
Visual	Unreal, but looking real	Optics: real and virtual picture of an object look the same, but the virtual picture can't be caught on photographic paper
Place	Immaterial, but provided by ICT	Virtual library, virtual database, virtual classrooms
Time	Potentially present	On-line or offline web services or e-communities
Evolution	Existing, but changing	Dynamic reconfiguration of adaptive systems

characteristics of the reality and explain how it works), and of experimenting with it (i.e., use the model that is derived from real-life and experiment with it using conditions that may not (yet) exist in reality). The Delphi method [24] could be interpreted as an example of such a modeling language. Using a structured approach to communications, Delphi could be described as a participatory rituals for reflection and imagination in a highly complex and uncertain scenario [24].

We see the trend towards experimentation as inevitable and it coincides with the growing complexity of the problems to be solved as well as the exponential progress in computer processing capacity. Experimentation in the context of negotiated reality is the iterative search for the virtuality configuration that finds the best interplay between perceptual-motor, cognitive and social aspects of people and computer systems. Thus computational experimentation is both a science of discovery and an engineering design methodology.

Table 3. Computing requirements for virtuality in the Reality-Virtuality continuum

	APPLIC- ATIONS	MODEL	DATA	INTERFACE	INFO- STRUCTURE
REAL ENVIRONMENTS ▼	Simulation	Replicate a close-to- real-life business behavior	Queuing applications; flight simulator	3D GUI; robotics	Stand-alone with advanced real-time sensor or high- performance computing
AUGMENTED REALITY					
AUGMENTED VIRTUALITY ▲	e-envirom- ments	Achieve a task in an electronic environment that is different than reality	e-classroom, Internet- supported distributed teamwork	Distributed multi-media platform	High bandwidth networks; cloud computing platforms
VIRTUAL ENVIRONMENTS	Virtual communi- ties	Social connected- ness in cyberspace	MUD, MOO	Instant- Messenger- like technology	Internet; Virtual- ization

Table 4. Definitions of virtuality – An inter-disciplinary perspective

DISCIPLINE	CHARACTERISTICS	SOME REFERENCES
Philosophy	High technology applications of the general principle that humans are self-defining creatures Inquiring systems and reality construction	[24] [36]
Management Science/ Operations Research	Conceptualization and abstraction of real-world via modeling and simulation Inquiring systems (e.g., Delphi)	[28] [29]
Computer Science	Property of a computer system with the potential for enabling a virtual system (in a computer) to become a real system; create model without coding	[23] [37]
Sociology	Departure from everyday reality to construct identity in the culture of simulation, thus eroding boundaries between the real and the virtual; create new forms of identities as they work and play with the new technologies	[34] [36]
Information systems	A new way of representing the world that is proving its value for understanding, monitoring and controlling natural processes; Scenario management	[19]

2.3 Modeling of Service Systems: Negotiated Reality and SSME

The relationship of virtual environments to service science, management and engineering (SSME) is one that has not yet been examined closely, most likely because the field of SSME is relatively new and still searching for guiding principles and concepts.

The formal representation and modeling of service systems is nascent, largely because of the complexity of modeling people, their knowledge, activities, and intentions. Service system complexity is a function of the number and variety of people, technologies, and organizations linked in the value creation networks. The challenge lies not simply in formally modeling the technology or organizational interactions, but in modeling the people and their roles as knowledge workers in the system [25].

Modeling requirements for service systems subsume conventional analytical modeling techniques. Although traditional operations research models, for example,

may still play an important role in service system analysis [11], the organic (versus hierarchical) perspective implied above suggests that computational experimentation approaches may be equally, if not more relevant, for capturing the people-based and knowledge-based dimensions of value creation networks.

Consider, as a very simple example, the well known MIT Beer Game simulation which demonstrates how local optimization of activities performed by each node in a beer supply chain (Factory->Distributor->Wholesaler->Retailer) leads to dysfunctional global system behavior such as the bullwhip effect where demand is cyclically over- and underestimated. Kimbrough et al. [20] show how an agent-based representation of the problem leads to a system optimum which is Pareto superior for all nodes in the chain. Although the optimization model can be formulated and solved at the overall system level, it is difficult to envision how it may realistically be implemented. What the Beer Game experiments and the agent-based model suggest is how a negotiated reality environment, one wherein each of the service providers plays the role actually corresponding to his/her “real life” role, may reveal improved service strategies in reality which benefit all players in the supply chain.

One of the salient features of service systems is that participating players must rely more heavily upon cooperation than competition. Inter-network dynamics may be competitive but intra-network processes are largely cooperative. This increased need for cooperation in turn relies heavily upon negotiation (e.g., service level agreements) as a critical element in service-based processes. We therefore return to the negotiated reality aspect of virtual environments as playing a key role in the modeling of SSME.

It appears that computational modeling and virtual environments are in the ascendant as instruments of exploration and analysis of reality, and that this may very likely be the case as well with service-based applications. However, we do not believe that this argues in any way for the obsolescence or decreased importance of the more conventional analytical modeling techniques and environments. As the Beer Game example demonstrates, there is substantive value to both approaches. What we would like to achieve is a synthesis in the form of an architecture for virtual environments which not only supports both analytical and computational modeling, but facilitates their integration in the spirit of Fig. 2c. In the next section, we outline guidelines and design principles for such an architecture.

3 Virtual Environments (VE) for Computational and Analytical Modeling (CAM): An Integrative Approach

This concept paper is an attempt to introduce the reader to the basic notions about virtual environments *for* computational and analytical modeling (VECAM). But what does the “for” really mean? At least two possibilities come to mind: “Virtual environments *in support of* computational and analytical modeling (VE→CAM)” and “Virtual environments *created by* computational and analytical modeling (CAM→VE)”. The former in our view emphasizes the *Science of Design* resulting in artifacts such as model management systems, collaborative environments, libraries of meta-heuristic solution procedures and grid arrays for solving systems of large-scale

simulation and optimization models. Many other examples could be cited. Broadly speaking, the purpose of these systems is to help modelers solve a wider array of more complex problems than they currently can address feasibly, including the integration of existing models.

The latter is oriented more towards decision-makers, and embodies the traditional view of DSS as “models in support of decision-making”. By its very nature, any model comprises a virtual world, by dint of the assumptions it makes about which details of the “real world” to emphasize and which to ignore. Certainly, the agent-based phenomenon plays a central role in this category embracing both the real and virtual worlds. We contend that the essence and potential of virtuality is such that modeling virtuality and using virtuality to model reality present contemporaneously rich opportunities and challenges for the scientific community.

In this section, we adopt the $VE \rightarrow CAM$ perspective, specifically examining agent-based modeling and simulation (ABMS) platforms for computational modeling in the context of model management research. We observe that ABMS environments are roughly at the same level of software maturity that analytical systems such as optimization modeling were twenty years ago. We apply design principles learned about analytical modeling environments from model management to generate requirements desiderata for computational modeling environments. In this way we hope to achieve a rapprochement that facilitates development of environments which support computational and analytical modeling simultaneously. In the Section 4, we take up the $CAM \rightarrow VE$ perspective and show how such a system can be used to create a virtual service environment in the telemedicine domain.

3.1 The VECAM Platform

To help bridge the gap between analytical and computational modeling environments, we advocate the definition of a bi-level, integrative framework inspired by the basic principles of analog transmission in the field of telecommunications. In analog transmission, data is transmitted using two components: a carrier wave and a signal. The carrier wave is a waveform that is modulated by the signal that is to be transmitted. This carrier wave is of much higher frequency than the modulating signal (the signal which contains the information). The reason for this is that it is much easier to transmit a signal of higher frequency, and the signal will travel further.

By analogy, the language that we propose consists of two components (Fig. 4). On the bottom level, a formal “carrier” middleware supports the modeling of all the social interactions in the virtual environment and offers the infrastructure needed to explain behaviors and describe environments. On the upper level, an unbounded set of formal languages (by analogy, a set of different signals) support the modeling of cognitive and reasoning operations.

We define this framework as integrative because it unites two families of existing platforms and languages. On the one hand, analytical modeling languages have been around for a long time (for example, mathematical, rule-based, heuristic, analog, and social modeling systems). However, just as an analog signal does not travel far on a

conductive medium without a carrier wave, the knowledge created by such languages does not travel far in a virtual environment without carrier middleware. On the other hand, agent-based languages can play the role of the carrier middleware. The next section focuses on the carrier middleware. Section 3.3 focuses on analytical modeling languages.

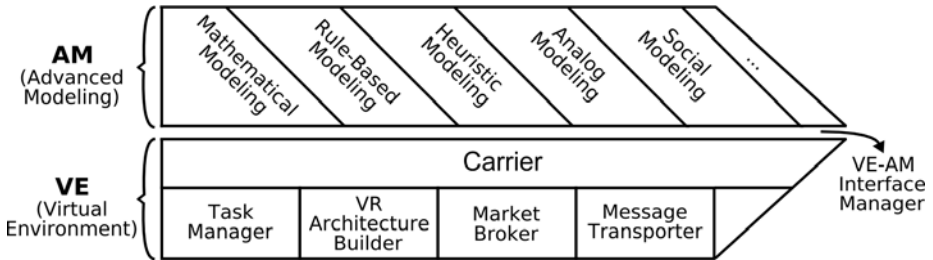


Fig. 4. The VECAM modeling framework

3.2 Carrier Systems to Support Virtual Environments

We use agent-based modeling and simulation (ABMS) platforms as exemplars of computational modeling environments. Recent years have seen a rapid growth in the number of multi-agent platforms, with a current total of at least 150 products¹. However, modeling software in the ABMS domain is currently at a relatively immature level when viewed from a model management perspective. For example, [Nguyen 2008] conducted a survey of six popular agent toolkits (SWARM, NetLogo, AnyLogic, Ascape, MASON, REPASt) focusing upon agent structure(s) and the technique(s) for representing agent behavior which each system employs (e.g., state transition graphs, programming language procedures, rules, etc.). Without exception, some level of software coding knowledge is required for the construction and execution of agent-based models using these systems. Further, there are very limited higher level agent representation schemas; of the six systems surveyed, only AnyLogic provides any capability in this realm in the form of state transition graphs for capturing agent behaviors. All systems, however, do provide libraries of reusable executable models which can be retrieved and modified.

This situation is similar in many ways to where modeling software for operations research and management science (OR/MS) applications was two or three decades ago. Linear programming systems, for example, used to employ matrix generators as the standard model representation. This required modelers to view and manipulate models as matrices at a machine level of representation instead of in a more natural mathematical form. Model representation formalisms such as structured modeling [14] and modeling languages such as AMPL were eventually developed to overcome this drawback [12]. As a result, the universe of model builders expanded from the highly focused specialist who had to know Fortran and mathematical programming in

¹ UMBC AgentWeb website, <http://agents.umbc.edu>, accessed September 20, 2010.

order to develop models. We would like to replicate the success of software evolution in the domain of OR/MS to the area of agent-based modeling environments, and incorporate some of these design advances into a broad conceptual architecture for a VECAM platform. In this vein, we see the following requirements as desirable, although not exhaustive, features of a computational modeling environment:

1. Generalized representations for agent structure and behaviors. The desideratum here is that model builders be freed from having to be Java programmers in order to build agent-based virtual environments. This, in turn, requires higher level ways of representing agent-based models. Static agent structures are relatively simple, often consisting of little more than attributes one might specify in a normalized relational table. Dynamic behaviors, however, are considerably more complex to represent and may require a portfolio of representation formalisms such as state transition diagrams, workflow diagrams, decision trees, and event diagrams. Higher level languages with corresponding graphical interfaces for specifying agents in these formalisms are necessary.
2. Separation of model representations from data. Agent-based models are typically less data intensive at run time than analytical models. Often the bulk of data processing in ABMS is at the front end of model building in the form of data mining to derive agent behaviors. Nevertheless, any data required for model development and execution should be logically separated from the agent representation.
3. Separation of model representations from solvers. Existing ABMS platforms are best viewed as potential solvers for an agent-based model representation, in the same way that OR algorithms and heuristics are solvers for various mathematical programming representations. Freeing model representations from any particular software platform protocols provides a powerful increase in generalization and flexibility. The cost of this generalization is the need to develop conversion engines for translating model representations transparently into specific solver formats (and vice versa for transmitting results).
4. Reusable model libraries. Models that have been developed, tested and used effectively should be documented and stored for future reuse.
5. Language(s) for experimental design. Specifying agent structure and behavior is only half the battle with agent-based models. As with any simulation technique, it is equally critical to set up experimental designs for analyzing the dynamic behavior of the model.

We re-emphasize that this is not intended to be a complete list of VECAM requirements but rather desirable features culled from decades of model management research into analytical modeling environments [Lindstone and Turoff 2002] which may be transferable to computational modeling systems. However, agent-based models have unique characteristics above and beyond analytical models that facilitate the cultivation of virtuality. We look at three more sophisticated and complex multi-agent ABMS environments: Cougaar, JADE and Brahms (Table 5) in order to extend our requirements list.

Table 5. Examples of platforms to implement the VECAM carrier

	Cougaar	JADE	Brahms
Main supporter	DARPA	Telecom Italia Lab	NASA
Implementation language	Java	Java	Java
Availability	Open Source (BSD-like)	Open Source (LGPL)	For research and non-commercial purposes. Licensed to NASA
Real-world use cases	Logistics DSS, military maneuver DSS (US Army); IT management software; Vulnerability analysis (e.g. Electrical grids)	Supply chain management; Holonic manufacturing; Rescue management; Fleet management; Auctions; Tourism	Human-robotic exploration; Modeling work practices onboard the International Space Station (ISS) NASA mission operations, planning, and scheduling
References	Helsingier, Thome et al, 2004 http://www.cougaar.org	Bellifemine, Caire et al, 2003 http://jade.tilab.com/	Sierhuis, J. et al, 2003 http://www.agentisolutions.com/

Cougaar is “an open-source Java-based agent architecture that provides a survivable base on which to deploy large-scale, robust distributed applications” [17]. Its extreme reliability makes it a very strong carrier middleware in the augmented virtuality area of the reality-virtuality spectrum (see Fig. 2). It offers the features required by any carrier sublanguage. The task manager is based on Cougaar applications. An application includes a set of domains (application data ontologies), a network of agents, and a society configuration (assigning agents to hosts and plugins to agents). The virtual reality architecture builder supports the development of highly resilient distributed architecture. The market broker relies on white and yellow pages repositories. Finally, the message transporter relies on a proprietary agent communication language (high level) and pluggable asynchronous protocols (low level).

With Cougaar, the VECAM interface with the upper layer of our bi-level framework (the modeling languages) is provided through plugins. “A *plugin* is a software component that is added to an agent to contribute a specific piece of application business logic. Each plugin adds domain-specific behavior to the agent” [17]. This definition is perfectly in line with our framework and Cougaar plugins naturally represent the VECAM interface manager shown in Fig. 4.

Table 6. Platform features in the context of the VECAM framework

VE	Cougaar	JADE	Brahms
Task Manager	Based on Cougaar applications	Based on ontologies and complex conversation skeletons	Based on Brahms workframes
VR Architecture Builder (with position on the VR spectrum)	Highly resilient distributed architecture (Augmented virtuality)	Peer-to-peer distributed architecture (Augmented virtuality)	Single VM simulation engine ² (Augmented reality)
Market Broker	White pages, yellow pages, service discovery	White pages, yellow pages	Single VM namespace
Message Transporter (low level protocols)	Pluggable asynchronous protocols, including RMI, CORBA, HTTP, and UDP, SSL, SMTP	RMI, JICP (JADE proprietary protocol), HTTP, IIOP	Relies on the KAoS middleware; CORBA
Message Transporter (high-level agent communication languages)	Proprietary agent communication languages	FIPA standard compliance Interoperability between J2EE, J2SE, J2ME, and .NET platforms	Proprietary agent communication language Open Agent Architecture (OAA) messages
VECAM Interface Manager	Using plugins	Opacity of the underlying inference engine	Using communication agents and/or Java activities

JADE is “middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm” [4]. JADE shares many similarities with Cougaar. However, it focuses more on mobility than on resilience, and offers better compliance with existing agent standards than Cougaar. The connectivity with the upper layer of our bi-level framework does not use the same plugin architecture. According to [4], “JADE is opaque to the underlying inference

² Even though Brahms is designed to produce a runtime system from a simulation, the publicly available version of the language is only meant to write simulation models.

engine system, if inferences are needed for a specific application, and it allows programmers to reuse their preferred system. It has been already integrated and tested with JESS and Prolog” (two rule-based languages belonging to the “rule-based modeling” category of Fig. 4). The other elements are described in Table 6.

The third platform, Brahms, is a multi-agent programming language developed by the NASA Ames Research Center [32]. Brahms relates knowledge-based models of cognition (e.g., task models) with discrete simulations and the behavior-based subsumption architecture. Unlike Cougaar and JADE, the publicly available version of Brahms belongs to the augmented reality area of the reality-virtuality spectrum. It is more a simulation engine than a runtime execution engine. The connectivity with the upper layer of our bi-level framework occurs via a specific kind of agent called communication agents. A *communication agent* is “a Java-based agent that interfaces between a Brahms system and other hardware or software components” [9].

Table 6 presents the features of these three platforms in the context of the VECAM framework shown in Fig. 4.

In the context of the VECAM framework, the carrier middleware must support the five requirements displayed in Fig. 4:

1. **A Task Manager**– this manager supports the representation in the virtual environment of tasks associated with the studied phenomenon. For example this representation can take the form of application data ontologies (e.g., for workflow-based planning or logistics), of typical interaction patterns to perform specific tasks (such as negotiations, auctions and task delegations), or to locate behaviors of people and their tools in time and space.
2. **A Virtual Reality Architecture Builder**– this builder supports the development and implementation of the actual virtual reality architecture adapted to the tasks of the studied phenomenon, as represented by the task manager.
3. **A Market Broker**– this broker supports the dynamic matching between entities that need to interact to solve a task or subtask during the study of the phenomenon.
4. **A Message Transporter**– this transporter supports the actual exchange of messages between entities brought together by the market broker. The message transporter manages both the high-level communication languages and the low-level network protocols.
5. **A VECAM Interface Manager**– this manager supports the connectivity between the carrier middleware and the upper layer of the framework, that is to say the Analytical modeling languages.

3.3 Analytical Modeling Languages for VECAM

Connected with the appropriate carrier middleware, modeling languages can be taken out of the often isolated and very domain-specific environments in which they currently reside. Table 7 groups existing modeling languages in five broad categories that could be used together on top of a carrier middleware to solve varied real-life problems. The purpose of this table is to illustrate the broad spectrum of modeling languages that can be integrated in the proposed framework. For example, rule-based

languages, such as JESS or CLIPS, can turn underlying agents into experts, injecting in their virtual incarnations knowledge traditionally found in isolated expert systems. Mathematical and heuristics-based languages, such as LPL, GAMS, or AMPL, can turn agents into number-crunching model solvers. Conversely, analog modeling languages can turn passive sensors usually considered as artifacts into reactive agents and group them in sensor networks [ACM 2004]. Finally, social modeling languages, such as ARBAS or ABEL, can lead agents to engage in negotiation and argumentation activities going far beyond the simpler communication patterns usually found in multi-agent platforms.

What we have shown in this section is a conceptual architecture for VECAM which facilitates a fusion of analytical and computational modeling in the service of virtual environments. Borrowing design principles from model management has the potential of liberating agent-based modeling environments from the sole bailiwick of programmers, and simultaneously putting both modeling paradigms on equal footing in terms of computer representation and executability. Meanwhile, the signal-carrier paradigm allows us to differentiate between agent-oriented languages and analytical modeling languages, while still allowing for the fruitful combination of the two in a powerful integrative way. Thus, we have taken a step in extending the notion of virtual environments to accommodate computational models without sacrificing the power and utility of more conventional analytical models.

Table 7. Modeling languages for VECAM

	Mathematical Modeling	Rule-based Modeling	Heuristic Modeling	Analog Modeling	Social Modeling
Methods	Linear and nonlinear programming, differential equations; game theory, queuing theory, linear regression, time series analysis, path analysis, and logistical regression or logic analysis	Forward chaining, backward chaining	Simulated annealing, tabu search, iterated local search, evolutionary algorithms, ant colony optimization, and other meta-heuristics	Sensors-based techniques (thermometer, speedometer, anemometers, barometer, hygrometer, accelerometer, etc.)	Negotiation, argumentation, discussion, articulation

Table 7. (continued)

Languages (examples)	AIMMS, AMPL, CPLEX, GAMS, Lindo, LPL, MPL, OPL Studio, Xpress; Prism, SPSS, R, S-Plus	JESS, CLIPS, OPS5, PROLOG	(ad hoc languages)	(ad hoc languages usually embedded with the physical devices)	ARBAS, ABEL
Input	Data	Facts	Data	Stimuli	Positions
Knowledge base	Models	Rules	Algorithms	Symbols	Social models
Reasoning	Optimization, mining, forecasting	Inference	(Sub)optimization,	Symbolic representation	Inference; optimization
Output	Data	Facts	Data	Data	Propositions
References	Huerlimann, 1999; Fourer, Gay et al, 2003; Castillo, 2002	Friedman-Hill, 2003; Giarratano and Riley, 1998; Sterling and Shapiro, 1994	Osman and Kelly, 1996; Voss, 1999	ACM and IEEE, 2004	Wooldridge and Parsons, 2000; Bui, Bodart et al, 1998; Anrig, Haenni et al, 1997

In the next section, we show notionally how to apply a limited version of VECAM architecture to a services-based application. We explore a telemedicine scenario as an example of negotiated reality and show how, using Brahms, we can create an appropriate virtual environment for examining and analyzing this situation. We believe that the computational modeling dimension of VECAM will play an increasingly important role in SSME applications.

4 Modeling Telemedicine Using VECAM

To illustrate the concepts advanced in this paper, we have modeled the virtual environment of a simulated telemedicine scenario [Bui 2000]. Telemedicine is not only a highly services-oriented application but it illustrates well the negotiated reality

aspect of SSME environments, and thus the suitability of VECAM for addressing these phenomena. Fig. 5 describes the workflow model of this scenario. Due to the simulation nature of this example, we have chosen the Brahms language as a carrier middleware. We note that Brahms suffers many of the same shortcomings as most ABMS platforms with respect to model management features, but is nevertheless sufficiently powerful to demonstrate conceptually the signal-carrier metaphor central to our VECAM architecture. A real-world implementation, however, would most likely rely upon Cougaar and its resilient architecture. A Brahms model can be used to simulate human-machine systems for what-if experiments, for training, for “user models”, or for driving intelligent assistants and robots.

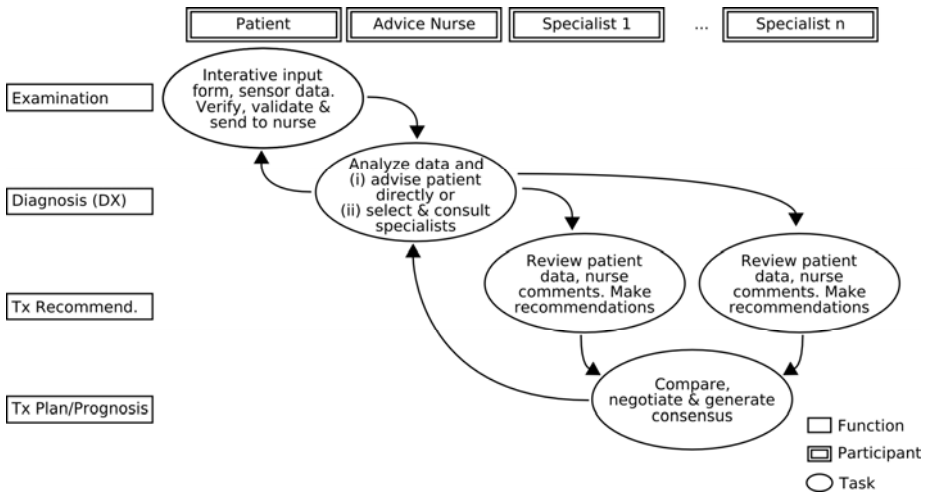


Fig. 5. The telemedicine workflow model

In a negotiated reality, we look for a model that includes aspects of reasoning found in an information-processing model, plus aspects of geography, agent movement, and physical changes to the environment found in a multi-agent simulation. Brahms makes this kind of model possible. Brahms relates knowledge-based models of cognition (e.g., task models) with discrete simulations and the behavior-based subsumption architecture. *Brahms* is centered on the concept of “agents.” Agents’ behaviors are organized into activities, inherited from groups to which agents belong. *Brahms* differs, however, from other multi-agent systems by incorporating chronological activities of multiple agents, conversations, as well as descriptions of how information is represented, transformed, and reinterpreted in various physical modalities. Activities locate behaviors of people and their tools in time and space, such that resource availability and informal human participation can be taken into account. Fig. 6 can be seen as an instantiation of Fig. 4 for the specific context of this telemedicine scenario.

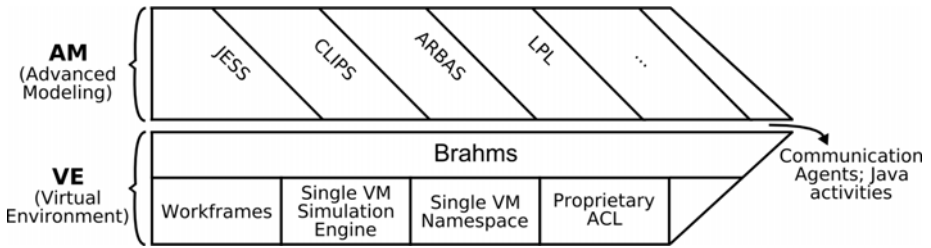


Fig. 6. The bi-level, integrated language adapted to the telemedicine scenario

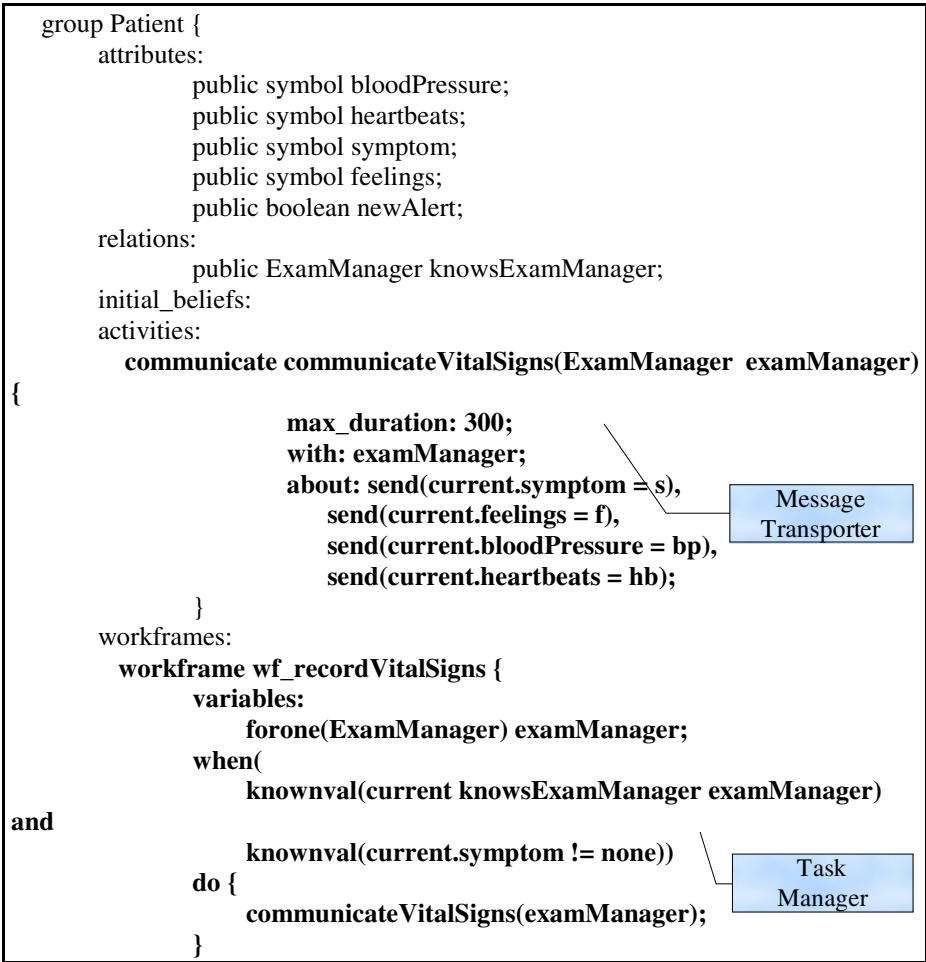
The Brahms language is built around constructs which can be related to one another according to the structure outlined in Table 8 (Acquisti, Sierhuis et al. 2002). This structure represents the backbone of the simulation engine built by the Brahms virtual reality architecture builder. With this structure, it is possible to accurately model complex man-machine, machine-machine, and man-man social interactions in a virtual environment.

Let's consider the telemedicine example. A specific patient (an agent belonging to the group "Patient") feeling sick (that is to say, having certain **beliefs** about his current health) can engage in a **communication** activity with a device monitoring vital signs (man-machine interaction). Beliefs about the patient's health condition are exchanged during this communication activity. The device in turn connects to the computer of the field nursing station (machine-machine communication), which will alert a nurse on duty (machine-man). In Brahms, the agent group Patient is modeled as in Table 9.

Table 8. An example of VR architecture builder

Groups of groups containing
Agents who are located and have
Beliefs that lead them to engage in
Activities that are specified by
Workframes that consist of
Preconditions of beliefs that lead to
Actions, consisting of
Communication activities
Movement activities
Primitive activities
Other composite activities
Consequences of new beliefs and facts
Thoughtframes that consist of
Preconditions and
Consequences

Table 9. An example using Brahms as a carrier platform



The `communicateVitalSigns()` activity of Table 9 exemplifies the use of the message transporter in the Brahms middleware. The workframe `wf_recordVitalSigns` is an example of subtask indicating that the `communicateVitalSigns()` activity must be triggered when the patient feels sickness symptoms (`knownval(current.symptom != none)`).

The agent group modeling the vital signs device (the `ExamManager` defined in the relations: section of the `Patient` group) defines a specific workframe to react to communication activities from the patient (Table 10).


This workframe exemplifies how the market broker of the Brahms middleware matches model elements together. The instruction `knownval(current knowsDxTxManager dxTxManager)` can be interpreted as “find in the virtual environment the `dxTxManager` element known to the current element”. This `dxTxManager` is later used by the message transporter during the `communicateVitalSigns()` activity (last instruction of Table 10).

Table 10. Finding other model elements with the broker

```

workframe wf_sendVitalSigns {
  repeat: false;
  variables:
    forone(Patient) patient;
    forone(DxTxManager) dxTxManager;
  when(
    known(patient.bloodPressure ) and
    known(patient.heartbeats ) and
    known(patient.symptom ) and
    known(patient.feelings ) and
    knownval(current knowsDxTxManager dxTxManager))
  do {
    conclude((patient.newAlert = true));
    communicateVitalSigns(patient, dxTxManager);
  }
}

```



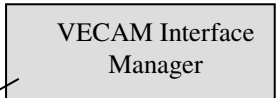
The computer at the field nursing station defines another workframe to react to communication activities from the vital signs device (Table 11).

Table 11. The Brahms middleware interacting with an Analytical modeling language

```

workframe wfr_processIncomingAlert {
  variables:
    forone(TreatmentPlan) treatmentPlan;
    forone(BaseAreaDef) loc;
  when(
    knownval(current.location = loc))
  do {
    broadcastIncomingAlert(loc, patient);
    createTreatmentPlan(treatmentPlan);
    conclude(
      (current knowsTreatmentPlan
      treatmentPlan));
    conclude(
      (treatmentPlan isDesignedFor patient));
    conclude(
      (treatmentPlan.shouldBeSentToSpecialist = false));
    broadcastTreatmentPlan(loc, treatmentPlan, patient);
  }
}

```



In this specific example, the createTreatmentPlan() instruction connects the Brahms middleware to a Java activity calling a rule-based language to establish a preliminary treatment plan.

Fig. 7 illustrates the communication between a (human) patient (John Smith), his exam manager agent in a dedicated device, and the diagnosis agent at the field nursing station (called the DxTxManager in the figure). The figure is a screenshot of the Agent Viewer, the visualization tool of the Brahms language. Each layer shows the active workframe(s) of the agent (“wf”) and its corresponding activities (“cw” for communicate activities, “ca” for compound activities, and “pa” for primitive activities). Note that a compound activity (for example, the processIncomingAlert activity of the DxTxManager) is broken down into subworkframes and subactivities. Reasoning activities are represented with light bulbs. In Fig. 7, light bulbs identify simple conclude statements (see Table 11). However, light bulbs in Fig. 8, which represent the negotiation activities between the cardiologist and the psychiatrist, facilitated by a negotiation software agent, represent reasoning activities performed by a dedicated negotiation sublanguage.

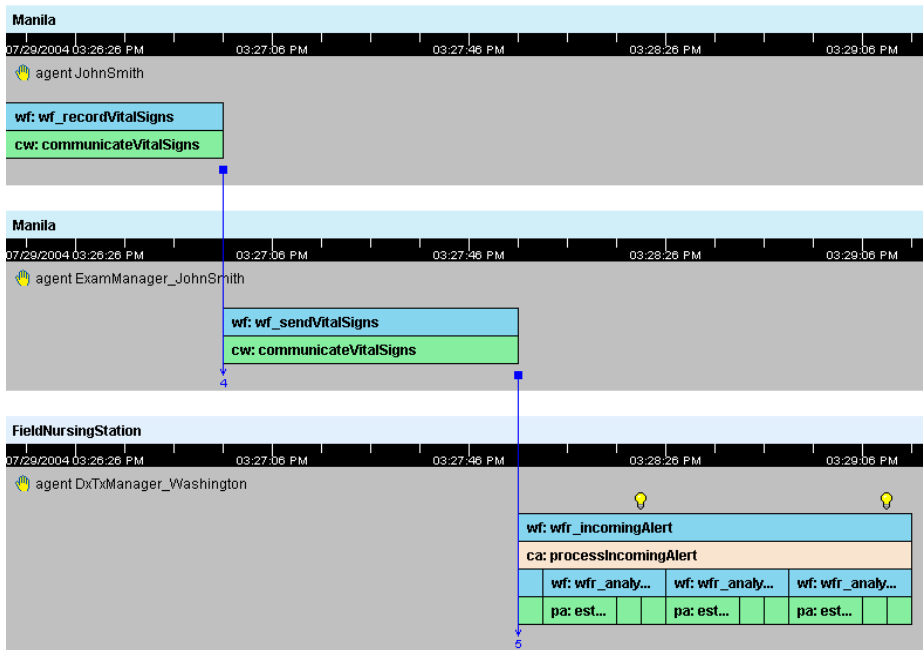


Fig. 7. Interaction between the Patient and the diagnosis agent, via the exam manager agent

Presenting in detail the complete scenario and the Brahms syntax goes beyond the scope of this paper. Tables 9 to 11 and Fig. 6 and 7 are only provided for illustrative purposes. However, it is important to understand that a model of activities in Brahms does not necessarily describe the intricate details of reasoning or calculation, but instead captures aspects of the social-physical context in which reasoning occurs. For

example, the activity createTreatmentPlan() in the workframe of the computer at the field nursing station does not contain any information about the actual operation consisting of establishing a treatment plan based on a preliminary diagnosis. The only information that the Brahms language associates with this activity is its duration and the artifacts (resources) used during the activity (if any).

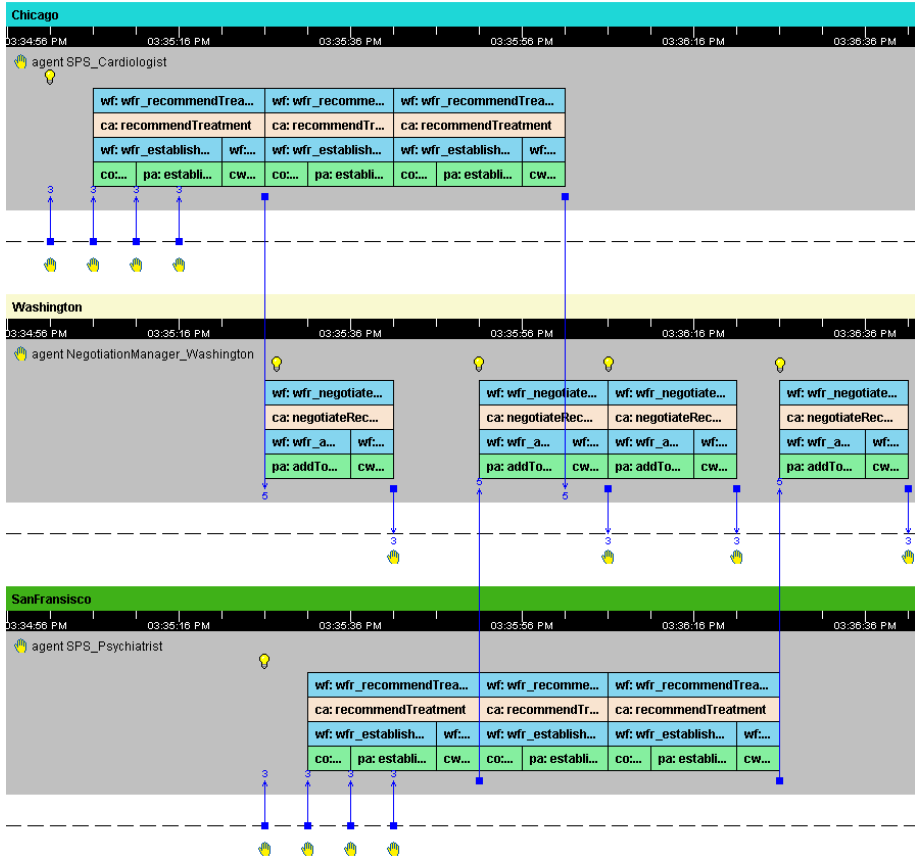


Fig. 8. Negotiation between the cardiologist and the psychiatrist facilitated by a negotiation agent

This is where the second level of our formal language comes into play. The carrier middleware must be “modulated” by other modeling languages or, in other words, the carrier middleware must offer hooks for other formal languages to do some reasoning and processing. For example, we might want to use a formal rule-based sublanguage (such as CLIPS or JESS) to model the actual reasoning happening in the createTreatmentPlan() action (with standard medical diagnosis rules). This language should have access to the belief set of the agent and to the fact set of the virtual environment. It should also be able to conclude new beliefs and facts. In Brahms, such hooks are currently provided in the form of communication agents performing

Java activities. Java activities are primitive activities, but their actual behavior is specified in Java code. Of course, this code could itself call native operations implemented in a different language (for example, the above mentioned rule-based language).

Our telemedicine example also illustrates the possibility to consult one or several specialists if an immediate diagnosis cannot be established by the nurse on duty. However, specialists might recommend conflicting treatment plans for a same patient. As mentioned above when describing Fig. 8, a special negotiation agent could rely on a structured communication language, such as the ARBAS language, to support argumentation and try to reach a consensus between the specialists. Once again, the reasoning operations modeled in ARBAS would be conveyed on the carrier sublanguage as beliefs and facts.

It is important to remember that the set of analytical modeling languages on the second level of our language is unbounded. Computational exploration might require various forms of reasoning and it is mandatory to give modelers and users the freedom to modulate the carrier middleware with the modeling language best adapted to the problem at hand. As such, the modeler would have a tool for computational explanation and experimentation.

5 Summary

With the increased use of virtual systems to make systems applications less dependent on hardware platforms, and the need for a framework to model highly complex situations, there is a growing recognition that the context in which modeling is required needs to be expanded from the traditional formalism of knowledge inquiry and problem solving. This is particularly true for service-based applications which we see as benefiting from advances in computational modeling. We propose a new paradigm – virtual environments for computational and analytical modeling (VECAM) – to help OR/MS/IS researchers explore new ways of modeling based upon negotiated reality. The framework consists of a Virtual Environment – the Carrier – that provides a virtual platform for applications to be built upon, and a Computational and Analytical Modeling module that contains a variety of modeling tools to allow the modeler to model and experiment with a virtuality.

We have successfully developed a virtual telemedicine application derived from a real-life concept of providing distributed, patient-centric emergency care. The migration of this reality to the VECAM platform allows the modeler to experiment with a number of new and virtual environments (virtual medical offices, virtual social network of medical professionals) and new and virtual business practices (medical advising procedures, semi-automated negotiation support). With VECAM as a conceptual basis, we believe we can begin to dramatically enhance, transform, and integrate the nature of modeling processes and environments.

The VECAM framework opens the door to several future research directions. From a model management perspective, an interesting avenue to explore is the model representation of agent-based simulations (ABS). Can higher level representations of

ABS be developed much like modeling languages were for OR/MS applications [Fourer et al 2003], which allow a more abstract and concise specification of models that not only free the modeler and end-user from the chains of learning object-oriented programming but also facilitate the integration of analytical and computational models? From a system design perspective, we might ask whether a different set of design principles is required for building a Carrier, or Virtual, Environment than for a Modeling module. The latter can be constructed from well known system life cycle methodologies, but as [Markus et al 2002] points out in discussing systems for emergent knowledge processes which have much in common with virtual worlds, the landscape may be much less crisp when the set of end users and their respective requirements are not known a priori. Virtual environments are much more likely to exhibit emergent properties which in turn may require a higher degree of dynamic configurability and “fuzzy” design, much like what has been envisioned for the Semantic Web. Adopting a network perspective, there is still much to be learned about combining virtual worlds as suggested in Fig. 2c and the network effects of doing so. As virtual worlds enter and leave such a network, what impact will this have on the process of scientific inquiry and how will this enhance or obfuscate the collaborative decision-making that will be required? How even does one “share” a virtual world in the first place? For example, how can a telemedicine virtual world be integrated with an emergency response counterpart so that medical assistance can be accelerated in times of crises? Also, implicit in the discussion, it seems that technologists understand well the distinction and interaction between virtual reality and reality, this understanding seems to be lost to post-modern sociologists. As an example, when extending the virtuality-reality continuum to the context of crisis management, the interplay between the emergent behaviors in the real worlds (e.g., on-site rescue vs. forum blogs offering resources) [Subba and Bui, 2009] has become an undeniable artefact. These and many more issues arise from thinking about fully idealized virtual environments. We hope that VECAM can be a preliminary step in addressing some of these intriguing questions.

References

1. Acquisti, A., Sierhuis, M., Clancey, W.J., Bradshaw, J.M.: Agent Based Modeling of Collaboration and Work Practices Onboard the International Space Station. In: 11th Conference on Computer-Generated Forces and Behavior Representation, Orlando, FL (2002)
2. Anrig, B., Haenni, R., Kohlas, J., Lehmann, N.: Assumption-based Modeling using ABEL. In: Nonnengart, A., Kruse, R., Ohlbach, H.J., Gabbay, D.M. (eds.) FAPR 1997 and ECSQARU 1997. LNCS, vol. 1244, pp. 171–182. Springer, Heidelberg (1997)
3. Association for Computing Machinery and Institute of Electrical and Electronics Engineers. In: Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, California, USA. New York, N.Y., April 26-27, Association for Computing Machinery (2004)
4. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A White Paper. Exp. 3(3), 6–19 (2003)

5. Bui, T.: Building agent-based corporate information systems: An application to telemedicine. *European Journal of Operational Research* 122, 242–257 (2000)
6. Bui, T., Bodart, F., Ma, P.-C.: ARBAS: A Formal Language to Support Argumentation in Network-Based Organizations. *Journal of Management Information Systems* 14(3), 223–237 (1998)
7. Carley, K.: Computational Organizational Science and Organizational Engineering. *Simulation Modeling Practice and Theory* 10(5-7), 253–269 (2003)
8. Castillo, E.: Building and solving mathematical programming models in engineering. Wiley, New York (2002)
9. Clancey, W., Sierhuis, M., Kaskiris, C., van Hoof, R.: Advantages of Brahms for Specifying and Implementing a Multi-agent Human-Robotic Exploration System. In: 16th International FLAIRS Conference, St. Augustine, FL (2003)
10. Deleuze, G., Guattari, F.: *Anti-Oedipus: capitalism and schizophrenia*. Viking Press, New York (1977)
11. Dietrich, B.: Resource planning for business services. *Communications of the ACM* 49, 7 (2006)
12. Fourer, R., Gay, D.M., Kernighan, B.W.: *AMPL: a modeling language for mathematical programming*. Thomson/Brooks/Cole, Pacific Grove (2003)
13. Friedman-Hill, E.: *Jess in action: rule-based systems in Java*, Greenwich, CT, Manning (2003)
14. Geoffrion, A.M.: The Formal Aspects of Structured Modeling. *Operations Research* 37(1), 30–51 (1989)
15. Giarratano, J.C., Riley, G.: *Expert systems: principles and programming*. PWS Pub. Co., Boston (1998)
16. Hamming, R.W.: *The Art of Doing Science and Engineering: Learning to Learn*. Gordon and Breach Science Publishers, Amsterdam B.V. The Netherlands (1997)
17. Helsinger, A., Thome, M., Wright, T.: Cougaar: A Scalable, Distributed Multi-Agent Architecture. In: International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands. IEEE (2004)
18. Huerlimann, T.: *Mathematical Modeling and Optimization, An Essay for the Design of Computer-Based Modeling Tools*. Kluwer Academic Publishers, Dordrecht (1999)
19. Jarke, M., Bui, T., Carroll, J.: Scenario Management – An Interdisciplinary Perspective. *Requirements Engineering Journal* 3, 3 (1998)
20. Kimbrough, S., Wu, D., Zhong, F.: Computers play the beer game: Can artificial agents manage supply chains? *Decision Support Systems* 33, 323–333 (2002)
21. Kimbrough, S.: Computational Modeling: Opportunities for the Information and Management Sciences. In: Eighth INFORMS Computing Society Conferences, Chandler, AZ (2003)
22. Kristensen, B.B., May, D., Jensen, L.K., Gesbo-Moller, C., Nowack, P.: *Reality-Virtuality Continuum Systems Empowered with Pervasive and Ubiquitous Computing Technology: Combination and Integration of Real World and Model Systems*, University of Southern Denmark (2004)
23. Lendaris, G.: Structural Modeling: A Tutorial Guide. *IEEE Transactions on Systems, Man and Cybernetics* 10, 12 (1980)
24. Linstone, H.A., Turoff, M. (eds.): *The Delphi Method: Techniques and Applications*, NJIT (2002)
25. Maglio, P., Srinivasan, S., Kreulen, J., Spohrer, J.: Service systems, service sciences, SSME and innovation. *Communications of the ACM* 49, 7 (2006)

26. Markus, M.L., Majchrzak, A., Gasser, L.: A design theory for systems that support emergent knowledge processes. *MIS Quarterly* 26(3), 179–212 (2002)
27. Milgram, P., Kishino, F.: A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems* E77-D(12) (1994)
28. Mitroff, I., Turoff, M.: Philosophical and methodological foundations of Delphi. In: Linstone, H.A., Turoff, M. (eds.) *The Delphi method: Techniques and Application*, pp. 17–36. Addison Wesley, Reading Massachusetts (2002)
29. Minsky, M.L.: *The society of mind*. Simon and Schuster, New York (1986)
30. Nguyen, C.: *GAME: Generalized agent modeling environment*. Naval Postgraduate School M.S. Thesis (2008)
31. Osman, I.H., Kelly, J.P.: *Meta-heuristics: theory & applications*. Kluwer Academic, Boston (1996)
32. Sierhuis, M., Clancey, W.J., van Hoof Brahms, R.: A multi-agent modeling environment for simulating social phenomena. In: *First Conference of the European Social Simulation Association (SIMSOC VI)*, Groningen, The Netherlands (2003)
33. Sterling, L., Shapiro, E.Y.: *The art of Prolog: advanced programming techniques*. MIT Press, Cambridge (1994)
34. Stone, A.R.: *The war of desire and technology at the close of the mechanical age*. MIT Press, Cambridge (1995)
35. Subba, R., Bui, T.: *Convergence Behavior in the Blogosphere*. In: *Proceedings of the Americas Conference on Information Systems (August 2009)*
36. Turkle, S.: *Life on the Screen: Identity in the Age of the Internet*. Simon and Schuster, New York (1995)
37. Turoff, M.: *Virtuality*. *Communications of the ACM* 40(9), 38–44 (1997)
38. Voss, S.: *Meta-heuristics: advances and trends in local search paradigms for optimization*. Kluwer Academic Publishers, Boston (1999)
39. Wooldridge, M., Parsons, S.: *Languages for Negotiation*. In: *Fourteenth European Conference on Artificial Intelligence (2000)*