ADDRESSING REALISM IN DETERMINING REQUIREMENTS FOR SIMULATION BASED LEARNING ENVIRONMENTS

Jerry Gordon, Scott Casey, Dr. John Burns
Sonalysts, Inc Orlando, Florida

Dr Joseph Cohn
Naval Air Warfare Center Training Systems Division
Orlando, Florida

Research has shown that simulations that are more realistic provide better knowledge transfer and richer educational experiences. Consequently, the challenge for simulation designers has been to "create a more realistic model." Unfortunately, the concept of realism is a subject of much debate and differing understanding. At the lowest level, realism deals with the physical representation of entities within the training space, i.e. do they look right. Simulation designers have attacked this level of realism with success. However, higher order concerns of realism are often overlooked in simulation design. Entities may look real, but they must also act real, and act real for realistic reasons. Unfortunately, these first and second order effects are often hard to detect during requirements analysis. Simulation designers have not yet established a structured method for determining them. However, they are critical to produce a good simulation for use in a context based learning environment, especially for those used in training decision-making skills. The identification of realism concerns will have an impact on a number of design, control, and interoperability concerns as well. This paper proposes a structured approach for identifying and organizing training system requirements that address the physical, first, and second order concerns of realism.

ABOUT THE AUTHORS

Jerry Gordon is a systems engineer/analyst for Sonalysts, Inc. He has worked on a number of instructional and simulation technology research projects for the US Navy and DoD. He served 8 years as a nuclear trained naval officer, and managed the wargaming center for excellence at the Expeditionary warfare Training Group, Pacific. He has a B.S. in Marine Engineering from the US Naval Academy.

Scott Casey is a Software Engineer for Sonalysts, Inc focused on research and development of a PC-based simulation systems for training. Mr. Casey is also involved with the investigation into the application of fuzzy logic and Data Drive Knowledge Engineering (DDKE) to understanding the actions of a Conning Officer for Underway Replenishments and Pier Work ship handling tasks. He has a B.S in Electrical Engineering from Rensselaer Polytechnic Institute and has several years experience in developing embedded software and hardware applications.

Dr. John J. Burns is a senior scientist at Sonalysts, Inc. working in Orlando, FL. He has been working closely with NAWC/TSD in research and development of Naval training technologies. He is the Sonalysts lead in collaboration with NAWC/TSD investigating the application of VE technology to Naval training needs. He has worked extensively on the development of performance measurement methodologies and technologies. Dr. Burns received his Ph.D. in psychology from the University of Massachusetts in Amherst and has 8 years of experience in team training research and development

Dr. Joseph Cohn is an Aerospace Experimental Psychologist at NAWC/TSD, working as a lead in the Virtual Training Technologies Team (VTTT) program. He received his Ph.D. from Brandeis University's Ashton Graybiel Spatial Orientation Laboratory, where he studied the effects of VE on human motor control. He recently completed a post-doctoral fellowship with Dr. J.A.S. Kelso, at Florida Atlantic University's Center for Complex Systems, where he explored the spatio-temporal aspects of motor learning.

ADDRESSING REALISM IN DETERMINING REQUIREMENTS FOR SIMULATION BASED LEARNING ENVIRONMENTS

Jerry Gordon, Scott Casey, Dr. John Burns
Sonalysts, Inc Orlando, Florida

Dr Joseph Cohn
Naval Air Warfare Center Training Systems Division
Orlando, Florida

INTRODUCTION

Major advances in computer processing power and graphics rendering have enabled the creation of virtual reality simulations. These simulations depict an "out the window" high fidelity representation of the world, and enable the user to interact with objects in this world. An important application for this technology is in providing context based learning environments. Context-based learning environments must, of course, provide realistic renditions of the problem domain that the trainees will interact with and within. As with any engineering problem, it is impossible to maximize all aspects of design, since increasing one aspect (in this case, realism and detail) will decrease another aspect (real-time computation). This paper will introduce a three-part model for discussing "What defines realistic?" and the associated technical concerns with implementing this in software. Following that is a discussion for relating the requirements of realism to pedagogical need in instructional design. Finally, we discuss a general approach for optimizing realism and instructional need in computer simulations.

Research has shown that learning transfer is increased when the learning exists in context to practical application (Gagne, Briggs, & Warner, 1993). Constructivist learning theory dictates that individual learners build their own concept of reality and truth (i.e. the "learned" material) as they analyze, deconstruct and make abstractions of experiences. The Radical constructivist paradigm requires an instructional environment where the learner is provided "cognitive tools" that they will manipulate, and in fact learn as they create their own learning environment with these tools (Jonasson, 1994).

While it is possible to train skills using real world equipment or in the real world operational environment, it is often too expensive, dangerous, or politically impossible to do so. For example, close order maneuvering of navy ships is a difficult training problem because of the high cost of failure (collision), but is a necessary skill for every surface warfare officer. Another excellent example is provided by the nuclear power industry, where initiation of real nuclear emergencies for training casualty control procedures is not only very dangerous, but also politically unacceptable. Decades of aviation training with simulators have consistently shown a 10:1 cost savings over live training.

As much as computing power has increased, and the applications that are possible today are amazing even by the standards of a few years ago, they are not completely perfect. Any model is still just that, an artificial construct that bears some similarity to a real world artifact, but with some aspects that do not look or behave exactly as they would in the real world. Extremely high fidelity physics models, which calculate all of the underlying processes and depict photo quality visual representations of many objects are possible, but computationally expensive. Computer Graphic Imaging (CGI) movies such as "Final Fantasy" or "Toy Story" are extremely realistic, but each frame takes hours to compute and render. Obviously this is not acceptable for any real time training application.

Computer scientists struggle to find innovative algorithms or techniques for representing real world objects in more computationally compact ways, enabling the rendering of rich visual environments in real time. Of course, these techniques necessarily make approximations or discard "unnecessary" information as the cost of computations possible in real time. The depiction of the real world is increasingly faked, as it runs closer to the same time frame as the real world. It falls to the instructional technologist to design the learning environment such that the faked aspects of the virtual environment are outside of the concepts and skills that are the learning focus of the experience, or that the differences are reinforced in some other way.

The mechanism used by the instructional technologist to dictate the scope of training is

requirements development. Unfortunately, task analysis and requirements development often focuses too much on the physical representation of objects within the simulation, assuming that more "realistic" objects will provide a more realistic simulation and hence better training. The disconnection occurs in the definition of "realism". Realism is more than just "looking right." Realism is also acting right, and acting right for the right reasons. This paper calls this static representation, dynamic representation and behavioral/relational representation.

We suggest that task analysis should be more focused on the specific training objectives that will be satisfied by training with the simulation system. The simulation designers have many different ways that they can model objects and control the simulation, but only some of the characteristics exhibited by these models are observable to and controllable by the trainees, and these characteristics should be those that are reflected in the training objective needs. The next sections will discuss the various methods employed by designers to make simulation models more or less "realistic" and the technical tradeoffs associated with them.

CATEGORIES OF REALISM

Static Representation Modeling

The most straightforward aspect of modeling addresses the question "What do the objects look like?" The static representation of entities within the simulation provides information about the structure and appearance, stored as a series of polygons and textures. Modern simulation designers, leveraging from the commercial gaming and motion picture industries have been incredibly successful in raising the standard of static modeling to incredibly realistic levels.

Computers render objects by describing them as a mesh of *polygons* (triangles and quadrangles) with applied *textures*. Before each frame is rendered, the computer must first calculate the changes in position, orientation, and lighting of the polygonal objects in the environment and then determine which polygons are visible to the user. Extremely high fidelity models with differentially small polygons and multiple light tracing algorithms yield photograph-like images, but take a long time to compute. Designers normally cull out extra polygons, deleting details not significant or visible, and incorporate a number of features as part of a texture, which is a photo or picture of the polygon's surface. The desire is to lower the polygon count, easing the computational requirements to render the scene. One such common method is to change the complexity of the polygonal mesh as a function of distance from the observer, typically called level of detail switching.

Additionally, polygons are eliminated from calculations by performing hidden line routines. These prevent the calculation or subsequent drawing of parts of objects that are not visible to the observer. Textures further reduce this complexity by incorporating small details as flat pictures. Lighting is needed to show areas of light and dark, but reflections and shadowing are often incorporated as part of the texture picture. The game "Duke Nukem 3D" used a set of portals and mirrors to establish the rooms in a given level that were potentially visible, as well as any objects or reflections that were present in those rooms (Luebke & Georges, 1993).

Now revisiting this discussion of methods for representing physical objects, think about the potential requirements of a military training simulation and how these techniques would impact the ability to adequately represent the context of a training mission. Soldiers fighting building to building often rely on visual cues such as shadows and window reflections to spot snipers, scouts and enemy troops. If the Military Operations in Urban Terrain (MOUT) simulation in which the soldiers are training represents shadows and reflections as static textures, the enemy troops and snipers will never cast any reflections off of windows in that environment, and the users will be deprived of this visual cue, but more importantly, will not transfer the knowledge on how to incorporate this cue. Collectively, the considerations of what to include and how are called *level of fidelity*. The level of fidelity can change depending on user viewpoint, but it must support the training needs of the simulation.

The combination of detail into textures may provide a realistic "ten foot rule" appearance for the object, but it could also remove an important detail. A virtual environment designer could build a very nice tractor model that includes lifting lugs and tow pads as part of a texture, that make it look very real to the user, but incorporating this model in a logistics loading simulation would not satisfy the training needs, as the users cannot connect tie town chains or towing hitches, since the connections are not modeled as objects, only as pictures.

Naturally, polygon culling is a function of the designer when producing the simulation, or included as run-time algorithms. Techniques such as anti-aliasing and per pixel shading exist to reduce the artificialities of the aforementioned techniques.

However, those techniques come with a computational cost, and computational power, while getting better every day, is still a fixed resource, and the designer must balance static representation against other behavioral considerations.

Any one of these techniques can reduce the computational complexity of a simulation, as well as shorten development time and cost. Not all of them will be appropriate in every circumstance. The framework advocated by this paper, focusing on the needs of the training objectives, should be the driving factor in technology selection.

This section discussed the various ways in which simulation designers can address the question "What do the objects look like?" The next section will discuss approaches to answering "How do they act?"

Dynamic Representation Modeling

Given a set of objects that are available for placement within a virtual environment, they must be able to move and interact. The kinematics of objects deals with their freedom of movement about relevant axes, their velocity and acceleration. Dynamic modeling also addresses other aspects of traditional or Newtonian physics, such as collision detection (when the objects "hit", do they bounce, deform, or pass harmlessly through each other).

Just as static representation could have several levels of detail on a scale of computational cost, so can the dynamic modeling. At the highest level of detail is the finite element analysis technique. The objects are split into logical pieces and the dynamic behavior of each piece is calculated using physics equations. An example of this would be to calculate the forces on each individual piston in an engine for a ground vehicle. A lower level of detail uses stochastic models, which are probabilities of a specific behavior occurring. These are far easier to calculate, and for aggregate systems are generally very accurate. An example would be calculating the arrival time of a platoon of ground vehicles based on their expected performance over specific terrains. The fastest, and least extendable, method is via empirical emulation, which is essentially a value look up table. Dependent upon the requirements of the virtual environment, an emulation of the dynamic representation most likely will be more appropriate than trying to simulate a complex, dynamic process. Empirical emulation is not only cheaper in computational cost, but the knowledge discovery during simulation design is also easier. In fact, there are techniques for discovering the empirical relationships from data runs of real systems (Casey, Cowden & Burns, 2000).

Imagine an aircraft simulator that uses look-up tables to establish flight profiles. If the users want to force the aircraft into speed/altitude profiles outside of the normal envelopes, the lookup table may not reflect these values, nor will the simulation user experience the stalls, oscillations, or structural failure that would result from operating in these regimes. Not experiencing this threat to safety is potentially fatal negative training.

Dynamic representations are also prone to level of detail and fidelity considerations. An aggregate simulation may only require four degree of freedom (DoF) aircraft models that are viewed as 2-D icons. For a strategic level wargame simulation, this is sufficient. 4 DoF models are not sufficient, however, for a flight simulator with precise handling characteristics and out-the-window views. This dynamic level of detail problem becomes apparent when combining or reusing models in a composable simulation. The training requirements, including dynamic modeling requirements, must be well understood, before selecting models to combine and reuse.

Every behavioral model considers a set of possible input variables, and generates a set of interesting output variables. These inputs are orders from the simulation users, or from other objects in the simulation. The output variables represent those things that are directly or indirectly observed by the user, and form the substance of information that is analyzed and constructed by the user as part of training transfer. The modelers must ensure that all the required input variables are accounted for, and that the output variables are correct over the range of normal expected user activities within the simulation context. In behavioral modeling, Occam's razor applies; the simplest (which equates to fastest running) model that satisfies those criteria is the correct model to use (Garey & Johnson, 1979). A model more complicated needlessly increases computational cost and provides limited, if any at all, additional training benefit.

However, a corollary to Occam's razor here is that this minimum model cannot be extended to include input variables that are not part of its model, nor consequently can it exhibit outcomes that are affected by these variables. As such, the use of a behavioral model that is of insufficient detail to include all factors that should be part of a contextual construct will provide incomplete or erroneous information to a trainee using this model (that is, in a way it was not designed to represent) and may

result in negative training transfer. Again, it is the training objectives and the associated experiences, concepts and cues to be trained that will drive the minimum acceptability criteria of the models.

The considerations involved in "look and feel" of objects in a simulation, discussed above in the sections on static and dynamic representation are well considered by software designers, and now much less difficult for designers due to a host of off-the-shelf design tools (i.e. Vega Marine, 3D Studio Max) and rendering engines (i.e. WAD, LithTech, Quake) available for reuse.

Behavioral / Relational Modeling

The most challenging aspect of modeling is making sure that objects act the way they do for the right reasons. This behavioral or relational modeling incorporates the relationship between the user or synthetic "people" and the entities within the environment. It also includes the storyline, a string of happenings or prompts within the simulated space that will stimulate the user and synthetics to action. The root of the challenge in relational realism is that it inherently is affected by the actions of the observer or user.

Behavioral and Relational modeling incorporates the Human computer interface, the overall storyline or scenario, and the emergent behaviors of dynamic interaction between the simulation objects, including any synthetic humans. The factors controlled by the simulation designers are the efficacy of the Human Computer Interface (HCI), the scenario, and the design of synthetic humans.

HCI design addresses a number of important human factors consideration such as ease of use, but for military training simulations in particular, it must also balance expected training benefit of the system against cost and complexity. A full mockup control system will provide the best haptic fidelity to the real world system, but if the intention of the simulation is to train other skills (i.e. cognitive skills such as decision making), it is needlessly expensive. In fact, it may be more desirable to intentionally build a more simplified or even different HCI, e.g. when the simulation is teaching a spatial relationship concept. A simulation that provides a training experience of out the window visual cues for craft piloting might be best suited with a dashboard style interface, rather than a high fidelity mockup of the actual controls. While there is value in handling the controls and getting a "feel" for the response of the ship, many of the other cues of the real world system or full motion trainer may be missing. Again, negative training may occur as a result of the missing motion feedback. For example, if we develop a ship handling training system that relies totally on a visual interface (no motion cues), we might be promoting the development of skills that have no real-world application. That is, in the real world, motion might provide a cue that our training doesn't allow the student to integrate into his skill set.

It is important to understand that the virtual training environment is a total system, and that this system incorporates a *virtual environment* and a *training management system*. Often the reason something must exist within the virtual environment, or why something will happen in this environment is in support of pedagogical goals as managed by this training management system. From instructional technology theory, a training management system must have 5 components: scenario generation, scenario control, data collection, training assessment and instructional delivery. These five components are incorporated into or interact with the training environment and manage the curriculum. In this case the curriculum is the visual representation and the scenario.

The scenario is a rough description of the storyline, that will unfold, as a set of *events*. In a training simulation, the scenario is providing the curriculum. While knowledge discovery by experimentation is an important product of the training experience, an unstructured free-play is not likely to accomplish all of the intended training objectives, and may allow the trainee to stumble into situations that the models simulate to a lower fidelity, again possibly with negative training transfer. A structured scenario provides events as observation and performance assessment opportunities that are linked to the training objectives. (Oser, Cannon-Bowers, Dwyer & Salas, 1997)

Control of the scenario can be tricky, however. Events are usually selected such that the trainees have only a few common sense choices that will guide them on the right path, but sometimes their actions can drift away from the intended structure. Strictly scripted scenarios guarantee that all objectives are met, but there are drawbacks, especially from a constructivist viewpoint.

Firstly, a strict script greatly restricts the trainees' opportunity for exploration and self-discovery. Moreover, the flow of events may not appear causally realistic in context with the trainees' actions and results, and therefore impact negatively on the trainees' construction of mental models.

Less constrained scenarios provide more opportunity for exploration and self-discovery, but there is always a possibility for the trainee to do something that precludes a later planned event. Artificially forcing conditions to facilitate the event (sometimes called a "magic move") causes a disruption in causal realism, which as before has potential negative effects on the trainees' construction of mental models. A good scenario based training simulation will balance the expected training objectives with the structure imposed on the scenario and the benefits of trainee exploration and self-discovery (Thurman, 1993)

Behavioral / Relational Modeling

Artificial intelligence (AI) has been a holy grail of scientists and philosophers for a very long time. A number of influential authors have been positing a blurring of man and machine since the 1930's (Weiner, 1948), and the first artificial neuron (nerve cell) was constructed in 1947. Recently, the commercial video game market has made great leaps in AI as well as graphics rendering, with a number of truly innovative techniques. Artificial entities within the simulation interact with simulation objects and the human trainee. These artificial entities must act such that the user believes they are other humans, acting as humans would under the circumstances. In some cases the entities are controlled by intelligent behaviors that mimic the functions of teams of people, but these must, as well, appear very similar to what common sense or experience dictates is a "realistic" behavior.

Behavioral models for artificial intelligence, just like static and dynamic representations, exist at several levels of fidelity and detail. Often (almost always) they do not represent the cognition or thinking of individuals modeled, but rather the effects of the expected individual actions. The obvious example here is production rules based systems, which use a series of "if this is true then do this" statements to model human or organizational behavior. These systems exist as a black box around the organism modeled; it cannot perceive or act upon its environment in ways different from those initially programmed. While a real human, or a more advanced AI construct may perceive and adapt to an innovative action or exploration on the part of the simulation user (i.e. trainee), the rules based system cannot generally adapt in this way.

This limitation in rules-based models will cause the system to respond in unrealistic, possibly unstable, ways. In turn, this can lead to a situation in which erroneous assessment is provided to the trainer, who in turn, provides mis-guided feedback to the trainee. The final result might well be that the trainees construct expectations that deviate from most probable real world occurrences. Additionally, a rules-based system that models the effects of an entire tank crew, cannot later allow the trainee to interact with an individual member of the crew. That individual does not "exist" within the model, and the effects of interaction with the member cannot be evidenced without re-examining and reprogramming the model. Training requirements must be examined to determine the level of interaction between human users and synthetic human representations, and the training environment likewise bound to prevent the trainees from interacting, exploring and discovering "information" too far outside the scope of a realistic representation.

Discussing the aspects of realism gives rise to the question "So what?" Why is realism necessary, and what effect does the artificialities of computer simulations really have? How does realism impact training and learning transfer? How does a modeler identify acceptable and unacceptable departures from realism? This next section will discuss the relationship between realism and pedagogy.

WHY REALISM?

As stated before, the primary reasons for utilizing computer simulations versus real world evolutions or exercises is cost, safety and political necessity. Virtual environments provide a low cost of failure venue for trying new ideas, as well as a low danger environment for training or exploring potentially dangerous evolutions. They also allow for experimentation and training where manipulation of the real systems may cause permanent damage to people or the natural environment (such as climate models or nuclear casualty trainers). Decades of training in vehicle simulators consistently show a cost savings over live training.

The purpose of any context based learning environment is to provide a rich training experience by giving the trainee a "cognitive toolset" that can be explored and manipulated. In this manipulation, the environment responds and provides feedback to the trainee as to the effects they cause, and with sufficient exploration and training, allow the user to construct robust mental models of the underlying structure that describes this cause and effect. The advances in computer technology and virtual environments enable the development of an immersion experience, a willing suspension of disbelief, where the trainee "thinks" they are performing actions in the real world. Research has shown that active participation provides greater learning transfer than passively receiving information, such as from a lecture hall. (Thurman, 1993)

Naturally, then, any glaring inaccuracies risk shattering the willing suspension of disbelief, and therefore negatively impacting on training transfer. For trainees with less experience in the problem domain (familiarity with real world system), encountered inaccuracies may transfer as incomplete or erroneous parts of mental models; a situation which the instructor must identify and correct (Thurman, 1993).

As trainees are more and more familiar with the problem domain, particularly from experience with the real world systems and real world operations, they are more likely to identify faults in the static and dynamic representations or improbable/impossible cause and effect relationships. The simulation designers must then minimize the inaccuracies and artificialities visible to the user. One way is by using only extensive, high fidelity physics based models and extremely high detail graphic models. Unfortunately, that approach is rendered ineffective because of the computational cost, or left incomplete due to inadequate developmental resources. How then to balance realism and efficiency?

The secret to solving this dilemma lies in the requirements derivation. The expected training objectives must be outlined and bounded. Viewing software as a system of interrelated components, the designers must ensure that the components model reality at a level at least as detailed as the trainees ability to observe or interact with it. This is especially important in selecting off the shelf components to combine to satisfy a training requirement. Object oriented programming facilitates this use of off the shelf components, and it is becoming increasing popular as a cost saving measure, but models and components cannot extend their usefulness indefinitely. The training objectives and respective requirements for interaction and observation drive the level of granularity and level of detail needed for component models. The scenario and instructional control must be selected to keep the trainees focused on the "realistic" aspects of the simulation, and prevent them from observing or interacting with features that are at insufficient levels of fidelity or detail.

THE APPROACH

Requirements specification starts any software design process. Depending on the system, and the presence of any legacy systems to be replaced, the primary sources for determining requirements are interviews with users and review of operating manuals/literature regarding the use or procedures that the software system will augment. Requirements shape the final vision of the project by defining what it should do.

The real driving force for the requirements should be the training objectives that will be taught in curricula using the simulation. Every job, enlisted rating or special designation in the service, associated with a school curriculum or job assignment has an associated set of terminal and enabling training objectives that match to needed knowledge and skills. Review of these training objectives help to scope the follow-on knowledge engineering. Knowledge engineering is the process of discovering what and how tasks are done and how long it takes to do them, so that they may be replicated in software. The training objectives dictate the specific tasks that will be of interest. Common methods for performing knowledge engineering are the Cognitive Task Analysis (CTA), a set of structured interviews with subject matter experts (SME), and data driven knowledge engineering (DDKE), which uses data search techniques to find patterns and rules from data collected during controlled experimental runs.

Both DDKE and CTA's are typically performed on to identify information about how users interact with the real world systems. The training simulation that will eventually be built is just that, a simulation, meaning that it has inherent artificialities. The training objectives help scope those aspects of its appearance, operation and interaction will be under greater scrutiny (from the trainee) and therefore must be built to higher

levels of fidelity. The other aspects must be included for visual appeal or context, but can be reproduced at lower levels of fidelity/granularity. All the while, the knowledge engineer must consider the three aspects of realism, and identify the required levels of granularity: What does it look like? How does it act? Why does it act that way? The training objective analysis and knowledge engineering are performed iteratively, using an approach similar to the spiral design process. The simulation designers must continue to structure their findings in view of the three questions.

An excellent analogy to this is the Hollywood sound stage. The actors and the objects that they are interacting with are replicated in the center stage to a high level of detail, while the rest of the scenery is painted on the walls (or digitally reproduced) in the background. In the recent film "Gladiator" the actors fought on the center stage of the coliseum, but it would have been cost and resource prohibitive to reproduce all of Imperial Rome as a movie set, thus it was digitally "painted" in with 3D computer models and computer generated special effects. The movie audience is engrossed in the action, and only peripherally notices the background. The training simulation must reproduce the objects and interactions that are experienced directly by the trainees during the conduct of tasks covered by the training objectives to a high level of fidelity and the rest can be "painted in" as lower fidelity approximations that will still meet the specific training requirements for the simulation.

How is it that viewers cannot notice the difference between high detail mockups and simpler models in background scenery? Compact disks approximate music by digitally sampling and reproducing clear tones. The music appears to our ears as being very real, but it is not. Real music is a continuous analog signal; digital music is discretized with quantization errors. How then does the music that we listen on our CD players appear "real" to our ears? The continuous analog signal is sampled at such a rate that our brain cannot discern the quantization errors or frequency differences present in the digitization process. The same is true with motion pictures, which are still frames animated at 24 frames per second, presenting the movement of illusion. Simulation objects and interactions are likewise not "real", but if they are modeled such that the users can't observe behaviors or attributes at a level where they can detect the differences, then the experience of using the simulation will be real enough. The aspects of interacting with the objects, the understood context of the experience, and the eventual learning transfer, will all be equivalent to using the real world system.

For example, an LCAC trainer used to train the Navigator and Craftmaster (pilot) in vessel handling operations. The Craftmaster knows that when he pushes the engine start button that an electric starter will move a compressor turbine, and that fuel will start flowing and that exhaust gas will start moving a turbine. Observable to him within the cockpit a yellow light will turn on and he will see some engine indicators for power and speed. Based on these indications, the Craftmaster will know that he now has the ability to start maneuvering the ship. In this simulation for this use, and set of training objectives; training the Craftmaster how to steer, the designers need not have a physics based model of the engines. If the expected users will never access the engine, or leave the cockpit. Then visual, structural or behavioral models of the engine would not be appropriate for this model.

This methodology stresses a systems approach to analysis and design. The system is composed of components, user interfaces, users, synthetic environment, synthetic entities, and instructional control, depicted in figure 1. Components within this system will also interact with each other. The interactions will be effected through a process that may be observable (such as collision detection and resolution with another object, or feedback to the user when they press a button), and cause something to happen within the simulation. The cumulative effect of these actions over time must be controlled to accomplish some instructional purpose, and to limit the "free wandering" of the trainee to perform activities which are outside the scope of the intended training uses, and therefore possibly not very realistic. As the training objectives are compared with the results of the knowledge engineering, the components are identified, and the requirements of interactions are identified. Each component must be analyzed in the abstract, looking for common features that will associate it with other components (consistent with object oriented analysis and design). The process should start at the macro level and work down in detail through a hierarchy of abstractions. Figure 2 depicts an example hierarchy. The required model is then that which satisfies the most restrictive requirements at the lowest level of detail for each of the subsystem.

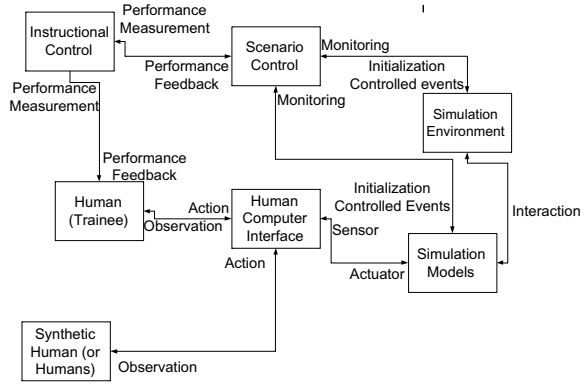

Figure 1. Training System Components

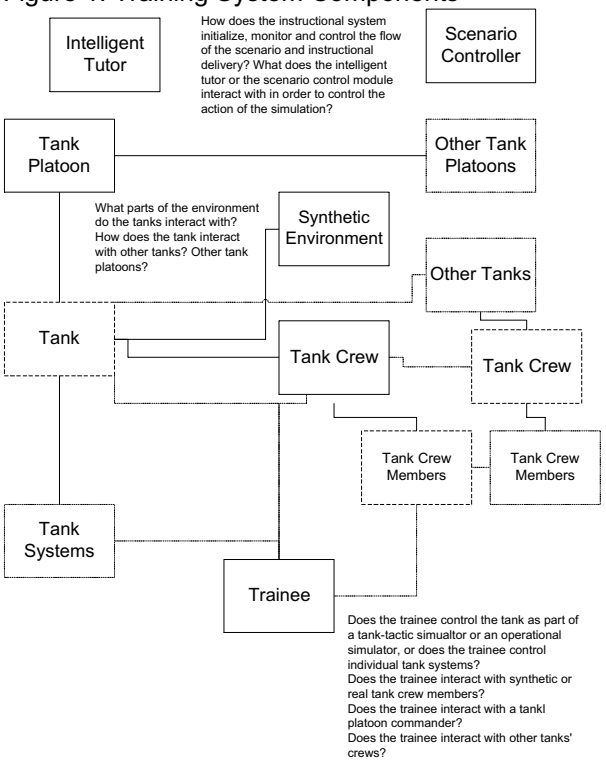

Figure 2. Abstraction Hierarchies

While it is true that sound product development starts with requirements derivation, "requirements" can exist at many levels, and much of the requirements documents that are currently produced are at too high a level to provide designers with sufficient guidance to make consistently appropriate design decisions. The lack of linkage between representation and pedagogical need combined with the unfamiliarity of subtle aspects of the problem domain (not otherwise discovered during KE, or incorrectly applied) obfuscates this requirement derivation. In developing a training system, it is all too easy to center on simulation technology, and forget that the needs of pedagogy should drive all decisions, as they should be the root of all requirements. The designers are left to make design decisions based on their incomplete understanding of the problem domain, in the face of maximizing performance (which as stated above, is sometimes at the expense of fidelity, and often will place limitations on scalability of models). Appendix A describes decomposition from Training Terminal Objective to UML modeling for the desired virtual environment. The resultant data structure was intended to conform to the open standard Synthetic Environment Data Representation and Interface Specification (SEDRIS).

CONCLUSION

We suggested at the outset of this paper that the driving force in creating simulations has been to create the most "realistic" environment for the trainees to interact within. Experience and observations from the authors has shown that this is usually interpreted to mean higher fidelity lower level of detail models for all objects within the simulation. Experience has also shown us that it is usually not possible to deliver all required functionality on time and budget using this mindset. The result is simulations with some aspects delivered at levels of fidelity much higher than that which is observable to the user, but with glaring gaps in functionality elsewhere.

We have provided the beginnings of a framework that dictates a different approach. We advocate a renewed focus on the training objectives, viewed in terms of the characteristics of static, dynamic and relational representation that are observable and controllable by the trainee, and therefore required for context. This framework, we believe, will provide very realistic impressions of those aspects that are central to training, and for those aspects not central to the training objectives, a lower level of realism that will not be as noticeable. This seems to be the best trade-off, allowing computational and financial resources to be applied where they are needed.

The most important points to take away from this approach are:

1) Training Objectives drive the requirements, answering "How realistic is realistic enough?"
2) That answer of "realistic enough" will then dictate the system requirements and design.

3) It is necessary to ask, “What is realistic enough?” because no practical engineering solution can be all things to all people, nor can it use all physics based 100% fidelity models and execute in real time.
4) “Realistic enough” must address the physical form and function (static and dynamic) of objects within the simulation, but things must also happen for a realistic reason.
5) Realistic reasons create a believable story and serve a valid pedagogical purpose.
6) The requirements of instructional technology are part of the overall system, and must be considered when determining requirements for the virtual environment
7) There should be a direct linkage between Training Requirements, Task Analysis/Knowledge Engineering, and Systems Requirements and Design

Think about a number of critically reviled special effects heavy movies. Where they may have excelled in visual appeal (static and dynamic representation), they lacked a basic believable and compelling story (Relational/Behavioral representation). With that lack of story, we do not like them, nor do we generally take much away from them. Many great character driven movies were made with minimal scenery and effects (and low budgets), but they are powerful experiences for us, as they had important and believable story and character interaction, with enough included scenery to provide needed context.

Clearly our framework needs to be further developed. In particular, it would be useful to provide a complete ontology for decomposing training objectives. Cognitive task analysis methodologies have known deficiencies as well, and this framework relies heavily on the CTA information to identify the needs of realism. In this approach, it is difficult to separate the concerns of the cognitive psychologist, instructional technologist and simulation designer, and as the three disciplines mature there will no doubt a need to mature our thinking.

Though our framework is incomplete, we think it is important to the modeling and simulation training community. Good training mandates a linkage between pedagogical need and simulation design, but no repeatable process exits yet to ensure it. Operational military forces need good training systems now, and they need to be trained right, and the development community is under pressure to deliver better functionality for less money.

References

Bloom, B.S. (1956) Taxonomy of Educational Objectives: The Classification of Educational Goal:Hanbook I, Cognitive Domain, New York: Longmans, Green.

Casey, Giebenrath, Shearouse and Burns, (2001) Fuzzy Cognitive Maps: Bridging the gap between Cognitive Task Analysis and Systems Design ; I/ITSEC Proceedings 2001.

Funge (1999), Representing Knowledge within the Situation Calculus using Interval-valued Epistemic Fluents , Gamasutra online.

Garey and Johnson, (1979), Computers and Intractability.

Jonasson, (1994), Technology as Cognitive Tools: Learners as designers.

Jonassen, Tessmer, Hannum (1999) Lawrence Earlbaum Assoc. Task Analysis Methods for Instructional Design

Luebke and Georges, (1993), Portals and Mirrors: Simple , fast evaluation of potentially visible sets

Oser, R., Cannon-Bowers, J., Dwyer, D., & Salas, E. (1997). Establishing a Learning Environment for JSIMS: Challenges and Considerations. *Proceedings of the 19th annual Inetrservice/Industry Training, Simulation and Education Conference, Orlando, FL* 144-153.

SAFE ENGINEERING AND OPERATIONS MANUAL Landing Craft, Air Cushion (LCAC) Navy Sea Systems Command S9LCA-AA-SSM-010.

TASK AND SKILL ANALYSIS REPORT FOR LANDING CRAFT AIR CUSHION OPERATOR TRAINING (1988) prepared by TSM Corporation

Thurman, R. (1993) Instructional Simulation from a Cognitive Psychology Viewpoint. *Educational Technology Research and Development* 41(4), 75-89

Wiener, N. (1948), Cybernetics or Control and Communication in the Animal and Machine, New York Wiley and Sons, Inc.