TURNING A CORNER: GAMES AND THE SOCIAL CONTENT

Don Anderson[a], Stephen E. Arnold[a], Dennis Jacobi[a]
Quasim Mehdi[b], Norman Gough[b]
[a]Intellas Group, LLC, 15424 Beckley Hills Drive, Louisville, KY 40245
[b]University of Wolverhampton, School of Computing and Information Technology,
35/49 Lichfield S Wolverhampton, WV1 1SB United Kingdom
danderson@intellas.com

KEYWORDS
Games, Social Content, Simulation, Modeling, Artificial Intelligence

ABSTRACT

Electronic games have moved to the mainstream. With this change has come a new set of challenges for engineers, developers, and funding entities. Third-generation game warfare goes beyond the reflex-reaction of the first and second-generation games, particularly with regard to warfare. The task today is to blend a data rich environment, near real time updates, and cognitive change within a context. Funding agencies, particularly for government-sponsored projects, require two different approaches to engineering design. The first is the need for architecting so that one or more elements can be easily repurposed. Repurposing means that the cost of developing a function or feature can be spread across multiple event delivery platforms. The second is the need for cost reduction and even cost recovery. The outcome of these two different engineering boundaries is a change in the way second-generation games and third-generation games are planned, constructed, implemented, and repurposed. The outlook for games is more robust than for some other types of applications but the opportunities come with greater costs. Changes include the need for online support, use of game-like functions outside of a game environment on analysts' desktops, and an increased discipline with regard to code, team composition, and engineering tactics.

INTRODUCTION

The challenges of modern game development were encountered in a recent project in which Intellas Modeling and Simulation worked with a noted board game designer, Vance von Borries, to create a concept for a third-generation training and battle simulation for the United States Air Force. Until the present, war games and simulations have been primarily attrition based and are centered on the concept of "force on force," and have been designated as "second-generation" war games. So-called "first generation" war games were focused on strategy with the primary concept of "mind on mind." This effort views "third generation" war games and battle simulations as concentrating on effects based operations with the primary focus being "system on system."

The new system will take into account all the factors of the previous generations such as strategy, tactics, and attrition, but will also include logistics, cascading effects, doctrine (both military and social), command and control, and differentiation between allied, enemy and coalition forces. (Pew et al. 1998) The revised system will be designed to be flexible and scalable with the capability of modeling a variety of scenario types that will include peacekeeping operations, homeland security and police actions in addition to the typical military combat scenarios. Details of the framework for the project can be found in Jacobi et al (2003). A macro view of a possible framework is illustrated in Figure 1.

ENGINEERING ISSUES

The primary engineering issues for a project of this scope and magnitude can be categorized in three main areas: Scalability, modularity, and system intelligence.

Scalability has generally been absent in the development of most game and

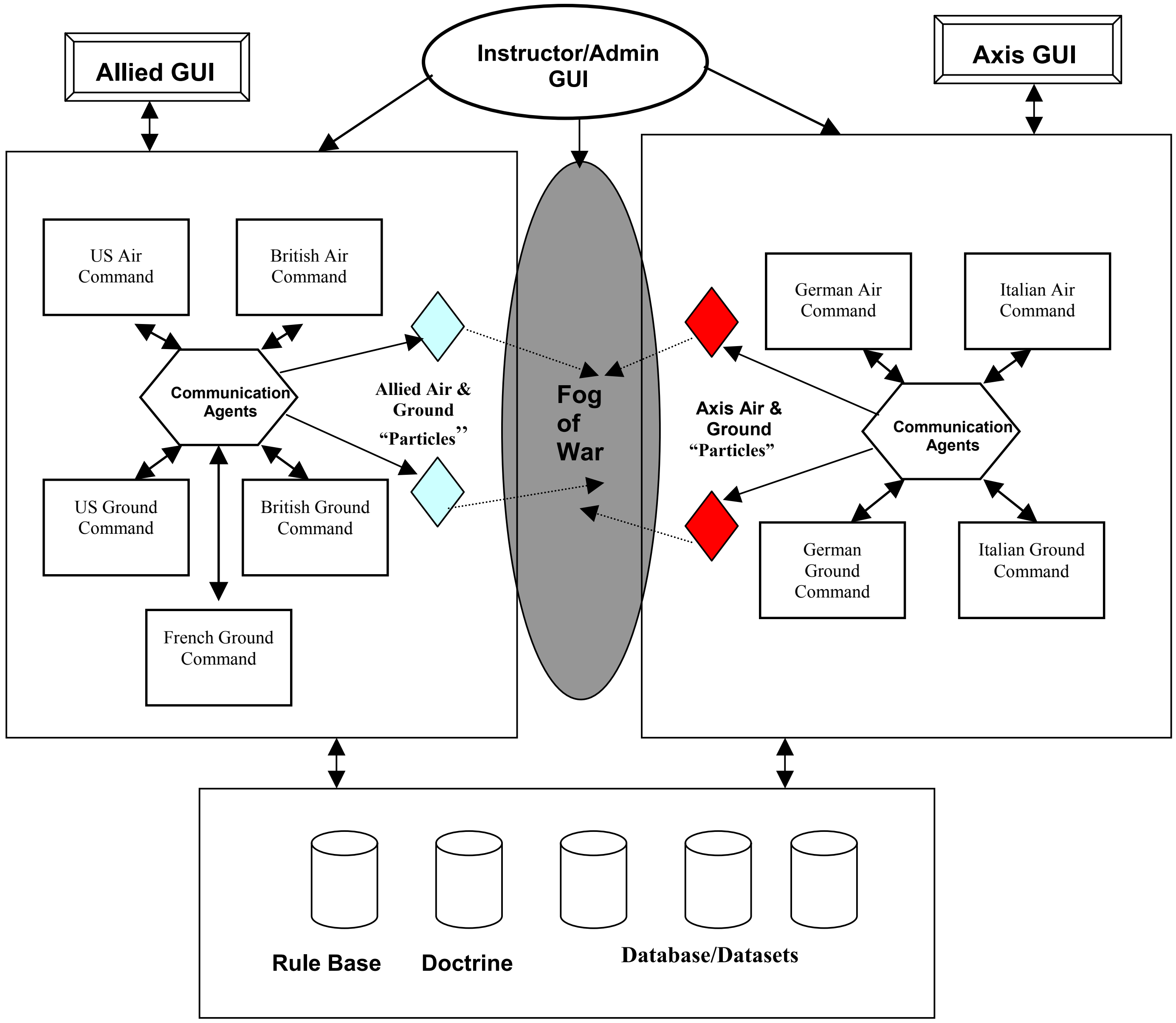

Figure 1: Simulation Macro View

simulation software. There is a vast gap between systems designed to run on standard personal computers (PCs) and those designed to run on large scale and super computers. The middle ground is barely touched at present. The on-line gaming paradigm has gone a long way towards making users aware of the need for a system that can scale to many users, but in government applications in particular, this creates security issues. For training purposes the scalability problem can be addressed by creating a distributed system (Anderson et al. 2002) that can run in a closed loop network or across secure lines between installations. Also a properly designed scalable system should lend itself to the ability to run on a single PC, multiple PCs or other suitable platforms, and also the ability to have higher level features that can be implemented and connected to on larger scale systems, clusters, and/or supercomputers.

Modularity, typically in the use of object-oriented design, has become the standard in the software industry as a whole. Developers have in general moved from the older methods of top down programming to the use of objects and modules of code. However the challenge is to take this concept to the next level through the creation of distributed systems in which the objects or modules can reside on a variety of PCs, platforms and other devices. Also the use of agents (Weiss 2000; Chen and Wragg 2000, Mehdi 2002) as well as newer methodologies such as particle swarm technology are needed to move to the next level of development and deployment of these systems to bridge the gap between games, training tools and higher level analytical simulations.

This new paradigm also requires new methods of system intelligence to be devised. Most games and simulations have used scripted intelligence, which is only as good as the scripts that are referenced. This need is the most diffficult challenge in this arena, to develop systems that are capable of learning, and beyond that to be able to mimic the imperfect decision making of human intelligence to add reality to the game or simulation.

PROGRAMMING COSTS

The cost of software development has not benefited from Moore's Law. For example, the cost of developing a game for the Sony PS1 was in the mid-six figures. The cost for developing a game for Sony's PS2 rose to about \$3 million per title. The cost of an original game for the Sony PS3 is likely to hit \$10 million. The Xbox game cost parallels the costs of developing for the PlayStation platform. The likely savings for the Xbox2 will come from Microsoft's use of a modified PowerPC chip, also used by Nintendo, and development tools that use the DirectX technology. The benefit for developers is that costs of developing for the Xbox2 versus the Sony PS3 is that overall development costs are likely to be somewhat lower. Sony has invented a new chip, a new graphics subsystem, and, therefore, new software development tools.

The same situation exists in virtually every sector of the commercial or enterprise software arena. Costs for license fees are flat or drifting downwards. The costs for programming, maintenance, and support are rising faster than any other cost associated with modern systems.

There are notable exceptions, and these warrant mentioning:

1. An individual not charging his time to a project with the requisite technical skills can program a winner. Whether one points to the success of Tetris or the handiwork of Shawn Fanning, it is possible to make millions at very low cost.

2. Open source programming provides a viable alternative to branded network operating systems. The open source revolution is likely to persist; however, for certain enterprise applications the fear of rogue code or security vulnerabilities effectively keeps certain open source software out of some organizations.

3. Recycling "old code" with today's programming tools can reduce the cost of migrating certain applications from one platform to another. Microsoft's new approach to Xbox development is that the SDK allows the programmer to compile for specific devices, including wireless platforms. By making repurposing faster and easier, programming costs are comparatively lower than approaches that don't use the most modern tools.

4. Modular structure, the use of ANSI standard C, and Extensible Markup Language can shave time from a programming project, thus reducing costs.

Nevertheless, overall game development costs are rising and there is little evidence that development costs will trend down in a significant way in the near to mid term.

The market has driven a change in development methodologies as illustrated in Figure 2. Efficiency and modularity are required to reduce development costs and provide a platform for future efforts. The organization of the project has become as important as the product to be developed. Since time has become increasingly an important factor, parallel development is essential to expedite the testing and release of the product both for the government and private sector markets. Investors are looking to see a quick return on their investments.

Old Development Methodology		New Development Methodology
1	Unstructured code	Structured and modular code
2	Assembler	C, C++
3	1 to 3 developers	2 to 4 teams, each with two to four developers
4	Graphics done ad hoc	Graphics specialists working in a way similar to the design team on a motion picture
5	No antecedent	Based on antecedents or a motion picture parallel shoot
6	Serial development	Parallelized development; teams may be dispersed
7	No documentation	Automatic documentation plus special notations for proprietary elements
8	No or casual source code control	Configuration management
9	Ad hoc compiles and tests	Engineering best practices

Figure 2: Comparison of Old and New Development Methodologies

CONCLUSIONS

For government game development projects, the goal is to create a code base that can meet the needs of the government client and cross over to generate commercial revenue. America's Army has become the model for that type of development. There are, then, some general guidelines that game developers will want to keep in mind.

OUTLOOK

What's ahead for government-funded game development?

1. Increasing pressure for commercializing certain games or components in order to generate cost recovery

2. Games will be engineered in the same way that other high performance government systems are designed and built

3. Reuse of graphics and code will expand beyond the "game application"; for example, recruiting commercials

4. Online is no longer an option. Games must run locally and support Web services.

5. Costs will continue to increase.

REFERENCES

Anderson, D., Belknap, M., Cui, X., Elmaghraby, A., Jacobi, D., Kantardzic, M., Ragade, R. (2002), "A Distributed Agent Architecture for Human Operations in Space," for the International Society for Computers and their Application 11th International Conference on Intelligent Systems on Emerging Technologies, Boston, 2002.

Chen, J.R., Wolfe, S.R., and Wragg, S.D., "A Distributed Multi-Agent, System for Collaborative Information Management and Sharing," Proceedings of the 9th ACM International Conference on Information and Knowledge Management, 2000, 383-388.

Jacobi, D; Anderson, D.; Borries, V; Elmaghraby, A; Katardzic, M; Ragade, R; "Building Intelligence in Third Generation Training and Battle Simulations," for the International Society for Optical Engineering AeroSense Conference, April 21-25, 2003

Mehdi, Q., Gough, N., Sulliam, H., "Virtual Agent Using a Combined Cognitive Map and Knowledge Base System," for the International Society for Computers and their Application 11th International Conference on Intelligent Systems on Emerging Technologies, Boston, 2002.

Pew, Richard and Mavor, Anne, editors, "Modeling Human and Organizational Behavior: Application to Military Simulations," National Academy Press, 1998.

Weiss, Gerhard, editor, "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence," The MIT Press, 2000.