Computer Based War Gaming: Recollections After Twenty Years

Todd Mason

Kriegspiel Developments

masont@bigpond.com

Abstract. The Australian Army began transitioning from manual war gaming to computer assisted and computer based war gaming in the late 1980s. Since then, computer based war games have been used for a variety of training and capability development tasks. One of the most successful systems is Janus(AS), which, since 1990, has been continuously enhanced and used for a range of tasks. This paper will highlight some of the factors that contributed to its success and identify lessons learned.

1. BACKGROUND

Janus was initially developed by the Lawrence Livermore National Laboratory in the 1970s to model nuclear effects on the battlefield. A number of versions were subsequently developed until 1989 when TRAC WSMR (along with other agencies) developed a 'standard' version called Janus(T)[1]. In 1990, the Australian Army procured version 1.0 of Janus(T) as an analytical tool to support capability and force structure experimentation. This became known as Janus(AS).

Janus is a multi-sided, multi-user, event stepped, networked, stochastic, constructive simulation of land and joint combat aimed at the combat team to brigade level. Strictly speaking, Janus is both the simulation software itself and a suite of support tools including: a database editor, terrain editor, scenario editor and replay tool.

The first activity to use Janus(AS) was Exercise Cyclops Dilemma in October 1990. This activity was as much a trial of the system and staff as it was an actual analytical study. The scenario was based on an infantry battalion providing vital asset protection to RAAF base Tindal against a low level enemy raider group. A number of 'runs' of the scenario were conducted with the defenders equipped with differing mixes of surveillance equipment.

For that exercise, a number of enhancements were made to the software including: multi-stage observation, neutral forces, disguised enemy, explosive devices and filtering of post-processor data for analysis.

This determined the standard method for development in support of experimental exercises. Between 1990 and the end of 1992, a further three analytical exercises were conducted with another one scheduled for early 1993. Analytical exercises were planned well in advance and software modification and testing was conducted in a fairly orderly, linear manner, with considerable assistance from staff from DSTO and other Defence agencies.

However, at the beginning of 1993, following on from experience in the US, Australia decided to trial Janus(AS) as a training tool. This generated a completely new set of stresses on the development process. Initially, the database did not support the expanded variety of equipment required to support brigade level combat. Additionally, many of the models in Janus(AS) had never been used and needed to be tested and understood.

The first training activity, in early 1993, was in support of the School of Armour and an initial attempt at a combined arms course.[2] This activity highlighted many problems with the system, but was evidently successful enough to demonstrate the potential for further use. The most significant lesson, however, was the realisation that training activities would require a completely different approach to the development of the software.

From 1993 onwards, analytical studies and experiments continued to be conducted, including support for the Restructuring The Army (RTA) trials in 1997 and Headline Experiments from 1999 onwards. However, training became the dominant usage, with all arms corps schools using Janus(AS) by the end of 1996. Plans were also developed to establish regional simulation centres to support unit training, that would use Janus(AS) and other systems as they became available.

In 1999, Janus(AS) was briefly replaced by another US developed Janus variant, known as Simitar [sic]. However, this was found to have a number of serious problems, and the users chose to return to Janus(AS) in 2001 when it was ported to Linux.

2. USAGE

In the analytical role, Janus(AS) has been used to support studies examining mobility, surveillance, uninhabited aerial vehicles, infantry firepower, motorised brigade force structure, littoral operations, and urban operations.

In the training role, Janus(AS) has supported Infantry, Armour, Engineer, Aviation and Artillery corps training activities at the Regimental Officer Basic Course

[1] See 'History of Operations Research in the US Army'. Charles R. Shrader

[2] This activity eventually evolved into the current Combat Officers Advanced Course (COAC).

(ROBC) and Regimental Officer Advanced Course (ROAC) level as well as combined arms training within the COAC. It has been used to support staff officer training at the Army Staff College and at Canungra. It was used to support a multi-national brigade Command Post Exercise (CPX) during Exercise Kangaroo 95.

Throughout its life, Janus(AS) proved to be extremely flexible. Janus is designed around a series of data driven models. For example, the direct fire lethality model uses lookup tables holding probability of hit and probability of kill data. These tables allow a great variety of lethality curves to be represented in an easily understood manner. Assuming the data is available, adding new systems to the database is generally a trivial exercise. Additionally, many of the models are stochastic, or probability based, which makes converting expected outcome to input data relatively easy to understand. The ease with which the data models could be explained to users greatly helped their understanding of how the system worked and also helped them provide suitable input data.

3. SOFTWARE CHANGES

3.1 Community Development

From the start, it was expected Australia would develop our own version of Janus, or at least our own subsystems. The UK had already developed their own version and Canada acquired Janus at approximately the same time as Australia. Within the US, while WSMR maintained the baseline version, there were multiple versions that had been developed for different purposes at different laboratories. There was a desire amongst the international community to be able to exchange software, data and ideas, but the practicality of that was limited.

Australia sent representatives from the Army and DSTO to each of the international user group meetings between 1990 and 1999 and in 2003. It was a condition of the memorandum with the US that Australian code changes were made available to the US team.

The only Australian model that was ever directly incorporated back into the US baseline was the multi-stage observation level algorithm. This replaced the simple 'detected or not detected' model with a more nuanced version with three observations states: detected, recognised and identified; where each state provided slightly more information about the target.

Cooperation was not restricted to swapping code. Many ideas were exchanged between the various developer groups. Australia certainly benefitted from this arrangement. For example: the UK team developed a new method for passing information to and from the graphical user interface which was adapted for use in the Australian version; and the US team developed an urban terrain model that formed the basis of the model used in a series of Headline Experiments between 2003 and 2006.

In 1995, the Australian team installed Janus(AS) on the computers at White Sands Missile Range where it was demonstrated to the international community at the Janus User Group Meeting. This was the first time an international version was demonstrated to the group and was received with considerable interest. Subsequent meetings were renamed ‘user fairs’ and included a significant amount of time allocated to demonstrations.

3.2 Australian Development

The list of Australian developments is quite extensive. To date, they have only been published in the proceedings of the Janus User Group Meetings, but even that is only complete up to 1999. Internal ASW documents exist describing individual changes, and the user documentation was maintained describing the overall latest functional state. However, as ASW no longer reported changes back to the US, no consolidated change log was maintained.

Some of the uniquely Australian developments include:

Improved aircraft flight profiles.[3]

Multi-mode RADAR.

Active defence systems.

Complex minefields.

Formation movement.

Maritime/ littoral movement.

Suppression.

Activity nodes.

Area weapon effects.

Stimulation of BCSS and 3D display.

Agent based artificial intelligence module.

4. DEVELOPMENT METHOD

4.1 Analytical Development

The model used for development to support analytical activities was the traditional maintenance model where a new requirement was identified, designed, coded and tested at a relaxed pace. These types of exercises were planned many months in advance and a detailed development and test schedule was established. Even in later years, once training had become the dominant driver of change, analytical enhancements typically had the luxury of time.

4.1.1 Structural Improvements

The relatively long timelines available for development facilitated the development of standards in coding and engineering that greatly contributed to the longevity of

[3] The Australian version was selected by a student at the US Naval Post Graduate School to support his thesis because of the enhanced flight profile model.

the system. The three principal methods were modularity, data hiding and named constants[4].

A number of early problems found with Janus were due to side effects of changing code. A side effect, is where a change in one section of code, causes an unintended change somewhere else. To combat that, proposed functional changes were implemented by re-engineering the code so as to separate code into smaller functions that performed discrete tasks. The major models, such as the detection model and the direct fire model, were reviewed to ensure that they did not interact at a low level, but communicated at a high level via passing of semaphores and messages.

Whenever a new feature required a change to a data structure, rather than just changing the structure and any code that referenced it, a set of functions were created to allow interaction with the data. This meant that future changes to the data structure would require minimal code changes. For example, in 1992, we were required to make significant changes to the terrain model to support maritime and littoral operation. This resulted in all direct access to the terrain data model being replaced by an extensive set of functions. When, in 2001, a completely new terrain model was adopted from one of the US versions of Janus, only the functions in the terrain library needed to be changed, leaving the main simulation code unaltered.

Throughout the original code, literal numeric constants abounded. Whenever these were encountered, they were replaced with named numeric constants. This has the effect of making the code easier to understand, but also allows changes to be made with lower risk of mistakes.

The only independent review of the Janus(AS) code was performed in 1996[5]. Although it identified some areas for further improvement, it concluded that the effort to date represented an advance on the US supplied code and that future maintainability had been greatly improved.

4.1.2 Teamwork

The team that initially established Janus(AS) was drawn from a variety of backgrounds. It included military personnel from a variety of corps, DSTO scientists, Defence public servants and contractors. This team was very well supported by other Defence organisations. For example, scientists from what was then the Army Engineering Development Establishment (EDE)[6] assisted in the development of a high resolution mobility model that was used to support two mobility studies in 1991 and 1992.

The collegial approach set the pattern that was continued throughout the development of Janus(AS). It also established the credibility of the core team.

4.2 Training Development

In 1993, the Janus(AS) team spent over 5 months of the year at interstate training establishments supporting courses and building a customer base. Unlike the analysts, who had many months to experiment with Janus(AS) and build confidence and an understanding of its behaviour, the trainers were lucky to have a day or two of familiarisation before they needed to begin developing a scenario and start user training. Credibility issues gained an inflated importance as students latched onto any failure to model a capability important to their plan.

The pace of the training schedule, the variety in customers and requirements, the limited knowledge customers had of simulation and the limited knowledge the technical team had of customer business all resulted in a need to change the way the software was developed. A conscious decision was made to adopt an evolutionary development model, where small high priority changes would be made as frequently as possible. This was combined with a time box approach to managing delivery schedules. Prototyping and 'test as you go' were heavily relied upon to ensure the development was heading in the right direction and to maintain user involvement.

Evolutionary development acknowledges that many things will change during the life of a development project. The customer base is changeable in the best of circumstances, but the military posting cycle introduces considerable variation in staff, from senior leadership down. Changes in doctrine, equipment and operational emphasis drove changes in development priority. Additionally, each change to the system effectively constituted a change in the environment necessitating frequent re-evaluation of the priorities. Each change opened doors to new opportunities that may not have previously been apparent.

4.2.1 Tracing

In 1993, a major effort was applied to developing a credible combined arms database and testing existing models. As part of this testing process, an extensive system of in-code tracing was introduced. With this tool, the internal calculations being performed by the software could be checked.

Initially, this was used merely to gain an understanding of how the system worked. For example, in 1994, Janus(AS) was used to support the Army School of Engineers, but the mine clearing model had never been used before and no one knew exactly how it worked. The ability to create test scenarios and run the tracing system proved invaluable in understanding the existing functionality. More importantly, it provided the foundation for working with the subject matter experts to establish credibility and to develop additional functionality.

---

[4] These are all pretty standard engineering approaches. Their use here reinforces how important they can be. Their lack, is not an indictment on the original software, it merely reflects the memory and speed constraints imposed by earlier hardware limitations when Janus was first developed.

[5] A consultant from Adacel was contracted to produce a report comparing Janus(AS) with the latest version of Janus from the US.

[6] Now Land Engineering Agency.

4.2.2 Customer Focus

The military trainers and instructors, were central to the development process. It was critical that they understood how the software could be used to support training and what improvements were feasible. The ability of the development team to deliver promised improvements not only enhanced their credibility, but made it easier to negotiate difficult requirements.

The development was facilitated by frequent visits to the user site, such as the school of armour. Often, such visits included demonstrations of real equipment such as tank bridge laying equipment.

4.2.3 Prototyping

An important element used to capture user requirements was the use of prototyping. Often a prototype was built as soon as an enhancement request was made. The benefit of a prototype was that users could immediately identify crucial features and confirm that the developers understood the essential elements of the request. For developers, the prototype served as a means of reflecting back to the user their description of the problem. The fact that developers and users could be physically collocated so often was incredibly valuable in reducing the risk of misunderstandings.

4.2.4 Time Box

The time box development process is used when delivery schedules are immutable. Training exercises were planned months in advance and need to fit in with student (and their parent unit) schedules. Unlike an analytical activity, a given training scenario may only be run once with no scope for re-running it if a problem occurs.

Change requests were prioritised with the user and then worked on in order until the development window was reached. Typically, requests were categorised as either 'must have', 'desirable' and 'wishful thinking'. Then testing and other pre-exercise activities were conducted. This means that it was not always possible to deliver all of the desired functionality in a given build. However, in most development windows between 1994 and 2009, not only were 'desirable' functions often implemented, but 'wishful' ones were as well.

The time box process is also important in restraining scope creep, which can be a risk with iterative development methodologies.

4.2.5 Teamwork

A crucial element that contributed the success of Janus(AS) was teamwork. This not only included the members of the direct support team, but the extra staff who supported exercises. Most importantly it included the users and other subject matter experts who contributed their time and expertise to problem solving.

For example:

> In 1993, instructors from the School of Armour and School of Infantry assisted in developing a suppression model.
>
> In 1994, instructors from the School of Engineers worked with the developers to code new behaviours for the mine clearing model.
>
> In 1995, a group of cavalry officers helped develop and test a casualty evacuation model.
>
> In 1996, officers from the School of Artillery worked at the AWGC to define the requirements for the indirect fire user interface, develop a prototype model and test the final version.

4.2.6 Documentation

When Janus(AS) was purely used for analysis, it was typical for comprehensive technical documentation to be produced. Often, this was done by DSTO scientific and analytical staff for publication.

In the training domain, there was little need for complex technical documentation. Users did not require detailed documentation beyond a basic operator manual and training guides and hindsight has shown that what technical documentation was produced was rarely read.

The training team did not receive the same degree of scientific support that analytical exercises did. Therefore, DSTO did not produce any scientific papers based on training developments.

A basic change summary was maintained for the purposes of reporting to the Janus User Group. This was supported by the code revision control system (CMS) and software versioning. The internal tracing system was used to supply supporting technical documentation.

The user manual and training material were regularly updated by military training staff. A system manual, that included installation and setup procedures, was also maintained, but did not require frequent updating.

In hindsight, greater emphasis should have been placed on requirements tracing. A simple running log of 'outstanding requirements' was maintained by the team manager. The details of the user requirements themselves were usually only contained in post exercise reports. It was possible to continually review these reports while the team remained small. However, once these documents were 'archived', that historical record was effectively lost.

4.2.7 Metrics

A weakness in the management of Janus(AS) development was the lack of documented measurement of success. No records were kept regarding the number of user requests satisfied or the hours expended. To a degree, some of this information could have been extracted from the software revision control repository. However, its absence subsequently made it difficult to substantiate what had been done without recourse to minute technical detail.

4.2.8 Marketing

In 1993 and 1994, a significant effort was made to market Janus(AS) to the training community. This resulted in extensive use amongst a broad cross-section of the Army. This possibly remains the most successful promotion of Army computer based war gaming in Australia.

However, this effort was not sustained. As the incumbent system, Janus(AS) suffered the effects of vested interests attempting to promote a variety of alternatives. In some cases, the sheer longevity of Janus(AS) worked against it as many people were often heard to remark 'oh, you're *still* using Janus'.

The continued Australian enhancements were not advertised beyond the small group supporting the system and a handful of similar international teams. After 1999, even the international teams were not regularly kept abreast of Australian developments. For example, Janus(AS) developments have never been reported at SimTecT.

5. OBSERVATIONS

Teamwork was the most critical element that contributed to the success of Janus(AS). The involvement of users in all stages of the development process meant that solutions were focused on the actual user need. The use of prototypes and user testing greatly helped overcome communication issues between military subject matter experts and developers.

This was made possible by the efficiency inherent in a small organisation such as the Australian Army. It was possible to co-locate small work groups to focus on a problem and apply the insights that different disciplines brought to the situation. This created a fertile ground for exploring novel solutions and ensuring that the most important problems were being addressed. It also ensured that all ideas were subjected to broad scrutiny to minimize the effects of biases.

The involvement of users produced a sense of ownership of the solution. This meant they went out of their way to provide information and assistance. They were encouraged to offer constructive criticism to improve the software and they became actively involved in testing.

Similarly, the developers gained a sense of ownership of the exercise. The success of the activity became the measure of success for the whole project. This ensured that 'tees were crossed' and details were not overlooked.

The software design approach of 'engineering for change' facilitated the evolutionary development method. The foundation of the software lent itself to prototyping. The inherent flexibility generated confidence that the software and the team could deliver the required capabilities.

Developing simple models, often using lookup tables and probability values, proved both intuitive and simple to develop. They are easily explained to users and straightforward to enhance. The use of in-code tracing also greatly helped explain the models to users and highlighted their assumptions and limitations.

Minimising the effort expended on documentation resulted in more resources available to test and develop the software and support user activities. In hindsight, more effort should have been expended on consolidating user requirements. Additionally, the failure to publicise successes left critics free to set the agenda.

The evolutionary development methodology proved well suited to the situation where detailed long term requirements were undefined and rapidly changing. The nature of simulation based training is complimented by a user focused development approach.