

# Paladin's Digital Fire Control System –a Technology Upgrade

**AdaRose Inc., a Service-Disabled Veteran-Owned Small Business (SDVOSB), was a key player for a major technology upgrade to an Army artillery system - the Paladin Digital Fire Control System.**

**AdaRose performed many system level integration tasks; resolved numerous hardware problems; and was the lead software developer on this program which included substantial modifications to over 230,000 lines of Ada, C, and Assembler language code.**

**Working with cost and schedule constraints, AdaRose used system-level risk management techniques, metrics, and numerous feature-set deliveries to keep the Integrated Product Team abreast of the development effort.**

**Other near-term technology upgrades to Paladin, that AdaRose engineers have been involved with, are also discussed.**

## The Challenge

Developing military battlefield systems can be challenging and rewarding, but seldom easy. Developing a system with a floating hardware and software baseline can be quite difficult. Add to this the parallel development of test tools, simulators, and emulators by third parties, then place schedule and cost constraints on the entire project and you challenge even the best and the brightest.



## The Task at Hand

Paladin is the Army's most advanced, fielded, self-propelled howitzer. The task was to port software from Paladin's legacy system to a simplified hardware architecture incorporating three single board computers in one control unit to provide navigation, command and

control, automatic fire control, situational awareness, and real time diagnostics.

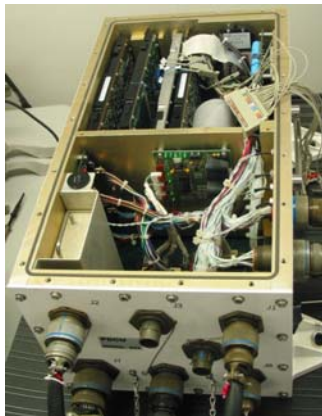
The IPT consisted of Northrop Grumman - a major defense contractor; Sechan Electronics, Inc. - a small business hardware manufacturer; AdaRose, a small business software integrator; the Army Artillery end user; Army R&D and Test engineers from Picatinny Arsenal, NJ, and the Paladin Product Manager.

**IPT members all had** prior Paladin experience developing hardware and software. AdaRose and Sechan engineers, led by UDLP, were the prime software and hardware developers for AFCS XXI. This bold experiment, (initiated by General Cartwright and brought to fruition by Carroll Gagnon) resulted in the first tactical weapon system to run on a common PC architecture using a commercial operating system. A subsequent upgrade to Paladin saw AdaRose engineers, working with Northrop Grumman engineers, to integrate an enhanced display and a situational awareness capability with the Force XXI Battle Command Brigade and Below (FBCB2) application.

All members of the IPT were experienced professionals with in-depth knowledge of the Weapon System from both a functional and operational perspective. The software consisted of 230,000 lines of Ada code with some specialized modules and drivers written in other high-level languages. The upgraded system is known as the Paladin Digital Fire Control System (PDFCS).

We'll first provide an overview of "what's new" in the Paladin PDCS system and then discuss how the program was managed from a risk management/mitigation perspective, and finally discuss other near term technology improvements to Paladin that will transform this weapon system into a mean, lean, fighting machine with awesome firepower.

## PDFCS Technology Improvements:



PDFCS is quite a different animal than Paladin's legacy Automatic Fire Control System (AFCS). Today's faster processors, smaller form factors, new communication devices, enhanced radar measuring units, enhanced FBCB2, and fire-control software with more robust internal communications, have made it possible to replace three Line Replaceable Units (LRUs) with one composite LRU, named the Paladin Digital Control Unit (PDCU.) The PDCU is a ruggedized hardware

unit containing three single-board computers, one each for the DFCS; the FBCB2 application; and a Prognostic Diagnostic Interface (PDI) application. There is also a flash drive for the DFCS; a TacLink communication module; and a Servo Input/Output (SIO) Card. Here are the major features and improvements of the new system.

**Improved Processing Capacity and Bus Speed:** The PDCU architecture is based on the industry standard Compact PCI bus. This provides significant performance improvement over the ISA bus of the AFCS system. Additionally, the single board computer features an 800MHz Pentium processor with enhanced graphics capability and increases in memory and disk space over the legacy system. Both the Fire Control and FBCB2 single board computers are identical and interchangeable.

**Display Unit (DU) Switch:** In currently fielded Paladin systems containing the Enhanced Display, a fatal flaw causes AFCS to go down when FBCB2 becomes inoperative. This will no longer occur with PDFCS. A DU switch on the display panel keypad lets the operator select either FBCB2 or DFCS video output from either of the two single board computers in the PDCU. Now, if FBCB2 goes down it will no longer take DFCS down, and vice versa.



**Built in Test (BIT):** BIT now provides more meaningful status messages to assist the soldier in diagnosing to the LRU level. Operator displays show the most likely source of the failure. Another added feature occurs at 'Start Up', when so-called 'Fatal Bits' no longer cause the System to power down completely. Instead, for example, the PDCU subsystem status may be displayed as OUT but FBCB2 will continue to operate.

**Replacement of Gun Movement Tachometers:** Analog tachometers have been experiencing high failure rates in the field. PDFCS removes the Azimuth and Elevation Tachometers that controlled the rate of gun movement and replaces them with a software solution, developed by AdaRose, that substitutes analog rate data with digital rate data

obtained from the DRU-H Inertial Measurement Unit. With rate movement algorithms developed by AdaRose, gun pointing accuracy is now better than that of the legacy system. The worst case error in the legacy system was predominantly in the azimuth plane at high gun tube elevations, sometimes as large as 0.7 - 0.8 mils. The new tachless system implements a systematic error reduction routine to improve this error to less than 0.2 mils on average.

**Device Drivers:** For PDFCS, AdaRose developed dedicated Windows device drivers to interface with the DRU-H; the PDFCS Keypad; the servo



weapon controller functions; and the SIO discrete functions (e.g., switch settings and indicator lights). These dedicated drivers replaced a single

generic driver in the legacy system where all data processing was being performed within the application software. The benefits of the new approach are: 1.) device-specific code has been moved into the driver, cutting down on the hardware dependencies in the application code; 2.) application code has been simplified; and 3.) overall system performance has been increased due to the dedicated nature of the drivers. In addition to these devices, the Paladin system contains two more ASYNC channels and three more SYNC channels for future expansion. The AdaRose driver structure provides a simple mechanism to take advantage of these devices in the future with only minimal software effort.

**TacLink:** The TCIM in the legacy AFCS and TacLink in the new DFCS both support communications employed on the emerging digital battlefield. Whereas TCIM was connected to the host computer via a SCSI



interface, and weighed several pounds, TacLink now interfaces with PDFCS via an internal PCMCIA connection and weighs only 4oz, and yet has a 32-bit microprocessor with 1Mbyte of RAM and

512K of flash memory. AdaRose developed the Ada software bindings to integrate TacLink with AFCS.

**Prognostic Diagnostic Interface:** Residing on the 3<sup>rd</sup> single board computer (i.e PC/104 card) within the PDCU is the Prognostic Diagnostic Interface (PDI). The PDI provides maintenance support for the Hydraulics, Cab Electric, Gun Positioning and the DFCS. This replaces the former Prognostic Diagnostic Interface Unit (PDIU), which was a separate LRU. The old 1553 bus structure has been eliminated from the PDFCS system and communications between DFCS and the PDI has been changed from a 1553 protocol to Ethernet TCP/IP Socket communication.

This change has made the communication a lot faster and more ruggedized.

Input/Output (I/O) Control was modified extensively and performs all operations necessary to obtain samples of analog inputs, digitize them and make them available to the PDI application. A new I/O driver allows the application to access privileged I/O facilities. This driver also performs Direct Memory Access (DMA), read/write to a COM port and read/write to an address. The PDI software was upgraded from Ada83 to Ada95 making the source code simpler and more manageable. The addition of the Windows Operating System (NT) was a big advantage due to the addition of the WIN32 API. PDI development was a joint effort between Sechan, ARDEC, and AdaRose. AdaRose converted Assembler language code in the legacy PDI system to Ada95.

**Muzzle Velocity Radar System (MVRS):** The new MVRS has a number of unique and improved features: an advanced trigger detection to ensure correct operation even with several weapons fired at once; advanced jamming protection; self calibration; a motion compensation for gun jump; accuracy better than  $\pm 0.05\%$ ; and it can store up to 1000 muzzle velocity results in non-volatile memory. AdaRose developed the software to integrate MVRS with Paladin.



## The PDFCS Integrated Product Team (IPT)

Although each project has its own requirements, the fundamentals of effective risk management at the system level remains the same. By identifying risks and developing solutions before and during the development process, should maximize the team's efficiency and the quality of the finished product. This was a fast-paced project that required solid teamwork and daily hands-on support from the IPT. The IPT is a valuable adjunct to containing costs and reducing risks, especially those that might effect scheduling. The Paladin IPT facilitated problem solving, enabling the team to rapidly respond to changing requirements, and prompted everyone to work on schedule.

### Identifying the Risks

From the beginning the IPT was informed that this project would have significant risk drivers, i.e. 1.) It would have cost and schedule constraints, 2.) It would require software development before the hardware was built, and 3.) Tools such as Lab simulators and emulators would have to be developed during the tactical software development. What was initially perceived as a straightforward port of software to a new hardware environment turned out to be a non trivial undertaking.

## System Level Risks

The challenges on this project soon became evident. On the one hand, software could not wait for completion of hardware due to the schedule constraint. This required the project to proceed with software design and development without access to a hardware target platform. Additionally, there was a requirement for building simulators and emulators for both development and testing. Some of these tools, being built by Army engineers, would need to be certified before use but certification required running on hardware and software which was still under development. These parallel development efforts would require a unique approach to development and risk management. At the Macro or Program Level we identified the following risk drivers.

**1. Schedule:** The schedule would be constrained and success oriented and the highest priority was placed on meeting schedule to allow for early fielding of the system. Many tasks that would normally be done sequentially would have to be done in parallel.

**2. Funding:** Limited funding was available for the software portion of the program. AdaRose would plan to make maximum use of available funding by multiple tasking of full time engineers and by utilizing part time labor for engineering support elements such as Configuration Management (CM), Quality Assurance (QA), Lab Technicians, and Network Maintainers.

**3. Technical:** A number of engineering challenges were evident. The FBCB2 Situational Awareness (SA) Computer had to be integrated to ensure that any SA failure would not impact the primary mission computer. FBCB2 risk was minimized because Northrop Grumman, developer of the FBCB2, chose a very strong Program Manager, with prior Paladin integration experience as the overall system integrator for the project.

Sechan engineers had built computer hardware components for the legacy system but putting three computers, running three separate applications, in one box had not yet been tried on the Paladin system.

AdaRose engineers had past familiarity with the tactical software, as well as prior experience with integrating situational awareness functionality and third party products. Two third party products in particular on this program were an MVRS Radar Measuring Unit and a TacLink Tactical Communication Module that needed to be integrated.

Therefore, technical risk, although not discounted, was placed third in priority as it did not appear to be a “show stopper” for the program.

## Risk Scenarios

At the start of the program AdaRose developed a number of risk scenarios to determine those events or trigger points we would have to watch, to warn us if and when the risk became imminent. Even though Technical was not a serious risk driver, our number one risk scenario was that the hardware, when delivered, would differ substantially from the specifications we had worked from. If so, it could entail software re-work and impact cost and schedule.

Our number one concern was the Communication Interface between the Tactical application and the Inertial Measurement/Navigational Unit – the DRU-H. In the legacy system, this had been a Direct Memory Access (DMA) interface. Any change here was very risky because this unit was at the heart of the system and failure here meant the system could be dead in the water. The trigger point we watched for in this risk scenario was any change to that interface – and sure enough it occurred as the project evolved.

As a result of the change in bus architecture to Compact PCI, the legacy protocol to the AFCS system had to be changed to an Interrupt Driven interface on the high speed CPCI bus from the legacy DMA interface on the older slower bus. This in turn, drove additional requirements to develop four new drivers to replace a single generic driver contained in the old architecture. This risk was mitigated somewhat by the fact that AdaRose engineers had prior formal training in developing software drivers for this operating system.

## Controlling Risks

As a baseline to accommodate top level program visibility, AdaRose normally uses the typical Stair Step development process consisting of 1) requirements analysis (RA), 2) design, 3) code & unit test (CUT), 4) system level integration & test (SIT), and 5) formal qualification test (FQT). Then, depending on the type of software to be developed (e.g. new development, re-host, block update, prototype, etc.) and the constraints placed on the program (e.g., cost, schedule, technical), this baseline is modified/augmented for best program performance.

For this program we decided to modify the baseline process with a Spiral development approach to obtain maximum productivity from our software developers and to mitigate major risk areas. At any point in time, programmers would be coding and unit testing in some areas while requirements analysis or design would be proceeding in others. We could also move out in those areas where the software was not yet dependent on hardware availability. For example, we decided early on to develop application-specific software simulators and communication protocol

simulators to test the software -- especially in those technical areas where rapid prototyping for proof-of-concept or early risk mitigation was warranted. The threads AdaRose used to maintain coherence throughout the development effort became known as “Feature Sets.” We found these to be invaluable risk mitigators.

## Feature Sets

As defined by AdaRose, a “Feature Set” is a block of executable software that contains pre-defined features/requirements that make up a subset of the entire program/application. A Feature Set can be anything from rapid prototype for proof-of-concept to a fully integrated and tested baseline. The purpose of Feature Sets is: 1.) to put before the user periodic drops of executable code to gain early concurrence and feedback of the included features/requirements; 2.) to conduct early testing to reduce program risk and provide relatively “bug free” software prior to entering FQT; and 3.) to keep the development effort moving by allowing developers to move forward on those sets of features that are not dependent on other events -- events such as delivery of target hardware, special tools, or 3<sup>rd</sup> party products.

Ideally, as the program progresses and the software matures, periodic drops of Feature Sets would consist of the most current feature set along with all previous sets until such time that the final set is incorporated and the application is ready to enter FQT. Most of the early Feature Sets were tested using the developed software simulators.

## Prototyping

Exploratory prototyping is an excellent risk mitigator if project requirements are ill-defined or likely to change before project completion. It’s also an excellent way to clarify requirements, identify desirable features of the target system, and promote the discussion of alternative solutions. Prototyping should answer two questions that are fundamental to system development and risk management—“Is the concept sound?” and “Is it worth proceeding further?” If the answer is not a clear “yes,” you may have a false sense of what can be accomplished. Better to know that up front.

**Digitizing Gun Movement:** For example, on this project we needed to determine whether or not a viable software solution could be found to replace the aging analog tachometers that controlled the rate of movement of the weapon system. AdaRose discovered that rate data was obtainable from the Inertial Measurement/Navigation Unit. They then proceeded to develop prototype algorithms that substituted this rate data for the tachometer data. The next step was to “prove the concept.” This required just enough re-coding to make it work on the legacy system hardware. This proved successful and as a result we were able to mitigate this risk early-on in the program.

Answers and risks were found to be satisfactory in the exploratory prototyping phase. AdaRose then moved on to evolutionary prototyping, which offers several benefits. It enabled AdaRose's software team to quickly and efficiently build on proven aspects of the software. As a result, the core of the software's foundation was tested and proven early in the project, significantly reducing exposure to unknowns. This became an important contributor to "Feature Sets".

## Process Improvement:

### HALT Testing – Building Ruggedized Components

Highly Accelerated Life Testing (HALT) tests the hardware well past normal operating conditions of temperature and vibration. It stresses components to the point of failure. It goes beyond hardware specifications and is designed to detect design defects prior to the production phase. This test method allowed Sechan, the hardware manufacturer, to evaluate failures and make necessary corrective measures. It not only improves product quality but reduces potential defects in the field. The results of the HALT testing yielded the following hardware design changes for improved reliability:

- Improved mechanical heat sink interface between the PDI microprocessor and the chassis
- Improved electrical and mechanical interface between the Fire Control Single Board Computer and the TACLINK.

### Dynamic Process Improvement

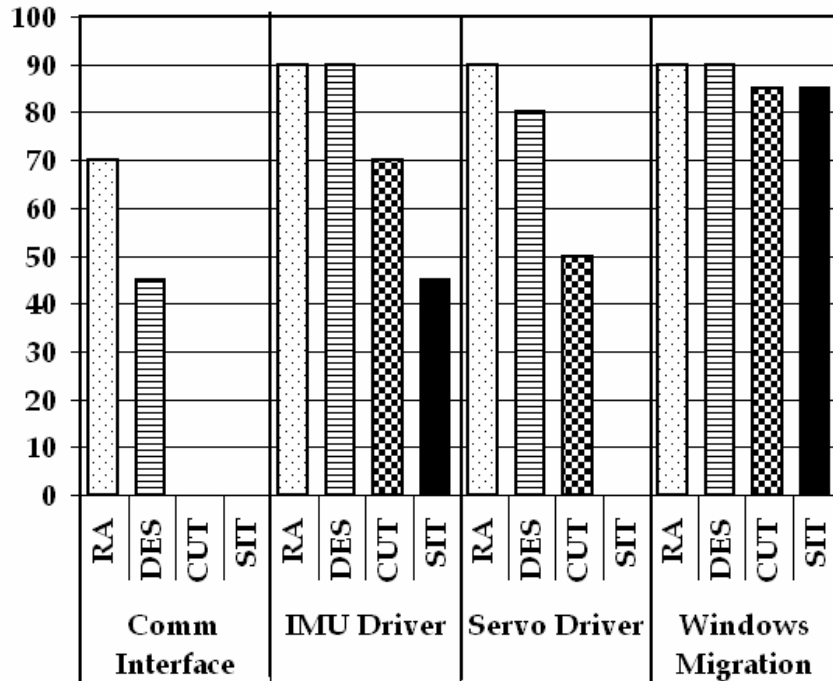
Improving processes should be ongoing throughout the project. For example, this project required a dual display mode on the operator's console. Rather than hold up development, while waiting on hardware to arrive, the software developers did the necessary design and coding and used a dual monitor graphics card to test out and "prove" the design.

We found it important to continually ask, "Is there a better way to get the job done?" AdaRose management met with the teams' engineers on a regular basis for focused but informal discussions. While these meetings were exceptionally valuable, we guarded against extended meetings that might cut into the teams' work time.

We found that a better alternative to lengthy meetings was to develop and distribute weekly status reports. These gave each engineer as well as the IPT insight into the progress of the entire project and a clear view of the big picture. Remember that you can have the best processes in place and

still fail miserably in software development. A motivated, goal-oriented, and knowledgeable workforce will succeed even when the process is lacking. An example of one metric we used on a weekly basis is displayed below:

**% complete**



**RA = Requirements Analysis      DES = Design**  
**CUT = Code & Unit Test          SIT = System Integration & Test**

This metric provided top level insight to the stage of development across blocks of functional requirements. We've shown here, only four of the thirteen major functional areas. Note that work in the Communication Interface area had not yet entered the Code & Unit Test phase due to non-availability of hardware being developed by a third party. However Windows Migration was well ahead of the curve because it was not hardware dependent.

Also included in the weekly reports were more detailed descriptions of the major risk areas, for example:

#### **IMU DRIVER**

Complete, integrate and deliver with Feature Set #7a.  
Test the integrated build at the system level.

**Risk Priority Ranking = 2**

Schedule: Moderate/High

Technical: Moderate

Cost: Low/Moderate

**Risk Mitigation:** Develop Dedicated IMU Driver & Modify Application Code IAW Established DRU-H Driver Development Plan. Apply Best People. Utilize Appropriate Tools (SoftICE DDK). Develop software communication simulator. Use logic analyzer & oscilloscope for low-level debugging.

**Action Officer(s):**Primary, Tom SW Engineering, Frank & Ilya

**Suspense:** IAW Driver Development Plan, Until Removed from Risk List

Risk rankings were continuously re-evaluated and re-prioritized throughout the program. As higher priority risks were worked off others would move up to take their place. Risk Mitigation became a dynamic real-time process.

### **Formal Qualification Testing – a Distributed Approach**

This program took two different approaches to Formal Qualification Testing (FQT). First it utilized engineers from both Northrop Grumman the overall system integrator and AdaRose the software developer, to run portions of FQT test procedures. Second, the testing was performed at three geographically separated locations as far as 3000 miles apart -- the Northrop Grumman Carson City, CA, facility, the Picatinny Arsenal, NJ, facility and the AdaRose Lexington, MA laboratory. This “virtual” test environment using development engineers with government oversight as well as government testers and IV&V personnel, was required because of limited availability of government test personnel due to test requirements of other on-going programs. Live fire tests were also conducted at Yuma AZ.

### **Third Parties: A Mixed Blessing**

If a product fails, it is common for many developers to blame the project’s failure on third parties. In some cases they are correct. At times you will have no choice but to elicit their help. The key is to minimize how much you depend on them.

Anytime you rely on a product or service from someone outside of your group, your risk of failure or delay increases. Your team may do everything right, but if a crucial third party does not, your work may be in vain. Use daily communication with them to keep them in the loop and make them a part of your team. AdaRose established an early-on working relationship with L-3COM engineers, by providing them with a subcontract so their engineers could contribute to the successful integration of tactical communication modules with TacLink.

## System Level Cohesion

Needless to say, systems, especially tactical battlefield systems, cannot be developed in a vacuum. In an ideal world, one would hope to have qualified hardware, emulators/simulators, and all design and interface documents delivered at project start. But that's not real, especially with military tactical systems. One finds that software and hardware are an integral, non-separable entity.

Quite often participants in system development efforts will “finger point” and blame the other guy for lack of progress. However, on this project we all realized the many challenges and knew that a successful outcome depended on a strong team effort. As a result we witnessed almost daily instances of engineers supporting each other – often putting aside their own work to help move forward a higher priority effort. We saw a close working relationship develop between our software developers and the hardware developer. Hardware engineers developed a strong cohesive bond with the software developer. AdaRose software engineers quite often diagnosed hardware anomalies and provided workable solutions. And at the IPT level, all were well aware of the risks and a “helping hand” rather than a “pushing hand” was the norm.

## Results

The PDFCS project successfully completed Formal Qualification Test and is now being fielded. The schedule was always paramount, but hardware prototypes came on line early, and software development was able to proceed in advance of production hardware by identifying and mitigating those critical risk areas that could be worked off early. We did this up front, through rapid software prototyping and by using Feature Sets to show proof-of-concept and provide executable code to qualify the hardware and help certify the simulation/emulation tools. This required a tailoring of our process and maintaining a viable and dynamic risk management program at the system level.

## Summary

System development will always include risks, but none are insurmountable if you are prepared to face them at the start. Risk management is an excellent way to prepare for daily challenges. Risk

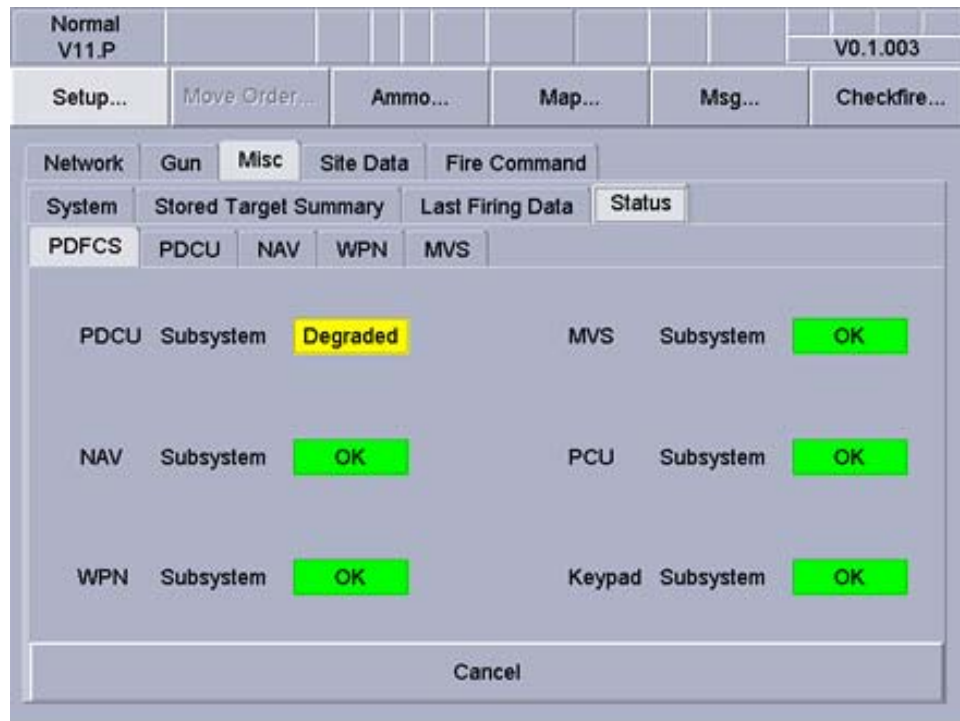
management must not only be implemented but continually re-assessed throughout the life of the project. Don't blindly follow any particular process, but tailor your process to the job at hand.

A viable risk management plan can mean the difference between success and failure. Above all it should be flexible and encourage initiative. Remember to always look ahead, use rapid prototyping as necessary, develop simulators if necessary, follow a defined program to minimize and manage risks, use a good set of metrics, keep the customer in the loop, and always follow the fundamentals of sound hardware and software development – from the system level perspective.

## On-Going Technology Improvements

### An Improved Graphical User Interface:

Current AFCS screen displays are in much need of improvement. Currently, the Soldier has to sequentially step through numerous ASCII Text menus in a non-intuitive manner to input data to the system. A more user friendly Paladin Soldier Machine Interface should substantially improve soldier/system performance. New graphical displays and touch screen technology combined with faster processing, in a windows environment, will now provide the opportunity to realize this advantage.



Northrop Grumman Mission Systems' has developed a Graphical User Interface front end and is working with AdaRose, to marry this Java GUI front end with the existing Ada DFCS application. Current efforts will result in a prototype and demonstration of the new GUI, integrated with the DFCS application. AdaRose made major modifications to the Paladin DFCS code to successfully integrate the underlying functionality of the DFCS with this new GUI.

This code is being delivered to ARDEC for future Paladin Software Block Updates.

### Excalibur Integration:



The Excalibur is a 155mm Precision Guided Extended Range Artillery Projectile. It uses GPS and inertial guidance to navigate its way to the intended target. Fired into a relatively high trajectory its glided flight is then controlled through base fins, achieving high accuracy and extended range beyond 30 km.

Adapting Paladin to fire the Excalibur would extend Paladin's range by 30 percent. More importantly, first round accuracy should improve from 370 meters for traditional artillery projectiles to 10 meters, or less, for Excalibur. In terms of combat fire power this means that many more targets could be engaged with the same ammo load.

Paladin will incorporate an inductive fuze setter to transfer target and fuze data to Excalibur. AdaRose engineers had earlier developed prototype software to enable Paladin's Automatic Fire Control System to communicate with the Portable Inductive Artillery Fuze Setter (PIAFS.) It was shown that the Paladin Fire Control System was capable of sending commands and targeting data through PIAFS / EPIAFS to Artillery Rounds, such as Excalibur, which are capable of having their fuzes set inductively.

Currently, Excalibur integration is being performed by ARDEC engineers at Picatinny, Arsenal, NJ. Most recently, AdaRose provided ARDEC engineers with the Windows Device Driver Tool Set that AdaRose used to develop dedicated drivers for PDFCS.