14067

Paper Number IVSS-2004-

Intelligent System Controller for the Full spectrum active protection Close in Layered Shield

Mark J. Salamango
National Automotive Center
U.S. Army Tank-automotive Research, Development, and Engineering Center

Brian E. Rathgeb
U.S. Army Tank-automotive Research, Development, and Engineering Center

ABSTRACT

Active protection may become heavily relied on for survivability of ground combat vehicles as the Army shifts to lighter platforms. Recent events in southwest Asia have also demonstrated a need for active protection on tactical vehicles. The Full spectrum active protection Close-in Layered Shield (FCLAS) system is an active protection system capable of intercepting threats fired even extremely close to a vehicle. The system was designed to provide significant protection for future vehicles while being applicable to the legacy fleet. FCLAS boasts elegant simplicity, which makes it an attractive solution to anyone looking to add protection to their vehicles.

The FCLAS system controller will offer both passive and active means to minimize fratricide when intercepting incoming threats. The system operators can actively disable individual tubes to prevent countermeasures from launching in the direction of dismounted troops; these areas are known as exclusion zones. A novel tracking system of dismounted troops allows the controller to dynamically create exclusion zones to maintain optimal protection while minimizing fratricide. The architecture used to implement the controller functionality provides a multitude of further benefits in the field. Among the benefits are diagnostics, prognostics, and logistics support.

Because enterprise networks are becoming distributed, the central data center has given way to a network environment containing distributed server clusters, edge servers, and a new tier of network-enabled devices that provide ubiquitous access. In essence, the network is expanding outward and embracing a series of new processing nodes (e.g., PDAs, cell phones, vehicles, system controllers, MP3 players, consumer appliances, etc.). These network nodes are called pervasive devices.

The goal of pervasive computing is to make data and application services available to any authorized user anywhere, anytime, and on any device. Technologies such as the Java Virtual Machine (JVM), the Open Services Gateway initiative (OSGi), and a host of communication standards have enabled the pervasive computing vision. Since a controller on a vehicle has similar configuration, security, and scalability needs as many pervasive devices, it makes sense to leverage these tools on the vehicle's embedded controllers.

This paper will explain the software architecture we have chosen to implement the FCLAS system controller in a secure, scalable, and reconfigurable way.

INTRODUCTION

As the U.S. Army continues to fight battles against a wide variety of enemies in a plethora of environments, it is vital to maintain technological superiority to keep the Soldier safe. A key aspect in protecting soldiers is to defend them from close range incoming threats. This paper will:

- Describe common threats.
- Introduce the Full spectrum active protection Close-in Layered Shield (FCLAS) system.
- Explain the software architecture for the FCLAS controller.

BACKGROUND

MODERN THREATS

Today's battlefield is host to a wide breadth and depth of threats intended to disable or destroy combat vehicles and/or combat support vehicles. Some threats have inception dates from the pre-Vietnam era while others have just recently come to fruition and possess the latest sophistication technology has to offer. In addition to age characteristics, threats can also be classified by how the weapon is utilized. Artillery shells can launch a bus full of submunitions from ranges in excess of 15 kilometers from the target. The shell releases its payload over a preprogrammed area of the battlefield at which time submunitions rain down on targeted vehicles.

Some threats are fired from the main gun of a battle tank. Two of the more common tank-fired threats are categorized as High Explosive Anti-Tank (HEAT) or kinetic energy rounds. A HEAT round will travel approximately 750 m/s and contain a fuze that is

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
26 MAY 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Intelligent System Controller for the Full spectrum active protection Close in Layered Shield	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Mark J. Salamango; Brian E. Rathgeb	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
USA TACOM 6501 E 11 Mile Road Warren, MI 48397-5000	**14067**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	TACOM TARDEC
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**8**	

triggered upon impact with a target vehicle. The fuze triggers the detonation of the onboard warhead, which forms a liquefied metal jet able to penetrate tens to hundreds of millimeters of armor. Kinetic energy rounds are transformed from dead weight into highly lethal projectiles when launched at speeds in excess of one kilometer per second. Resultant energies can exceed nine megajoules, which is enough to penetrate hundreds of millimeters of armor.

Anti-Tank Guided Missiles (ATGMs) can be launched from combat vehicles, portable launchers, or from the shoulder of a soldier. ATGMs employ a similar warhead to HEAT rounds, but have embedded systems that allow them to be guided into the targeted vehicle. Some ATGMs are actively guided by the operator into the targeted vehicle. An example of this is the American TOW (Tube-launched, Optically tracked, Wire guided) missile. The operator must command steering corrections of the ATGM via the fiber optic cable from launch until impact. Other ATGMs are considered "fire-and-forget" weapons. The gunner needs to only point and shoot the weapon for it to be effective. This capability allows some ATGMs to be fired from beyond the line of sight of the gunner, yet have a high probability of kill. Using a laser designator to identify the target is a common way to employ this type of system. A soldier, independent of the gunner, shines a laser at the target. The ATGM locks on the laser energy reflected off the target and steers itself into that target.

An additional threat gaining the attention of the public is the Rocket Propelled Grenade (RPG). This threat combines a rocket motor, impact fuze, and warhead into a small, easily transportable package. The launcher and RPG can usually be carried on the back of an infantry fighter or a collection can be transported in the bed of a pickup truck. The threat does not use any sophisticated equipment, which makes it easy to use for any trained or untrained soldier or insurgent. Records of RPGs date back to World War II at which time the Union of Soviet Socialist Republic began massive production efforts. Designs and reverse engineered derivatives of this threat have since been scattered throughout the world for a variety of countries to produce and purchase. This has brought available quantities into the hundreds of thousands and decreased purchase costs to hundreds of dollars. Adding to the threat's popularity is its effectiveness. A skilled shooter can hit a vehicle from 200-300 meters, puncturing a whole in a tank, personnel carrier, or truck and potentially ending the lives of its crew and passengers. Among the enthusiasts of RPGs are terrorists. Heavy armor can significantly degrade an RPG's effectiveness, but not all vehicles have the luxury of supporting such massive amounts of weight and space. The next generation of U.S. combat vehicles will not rely on this option either, based on the Army's efforts to build a lighter, more deployable force. One technology that may protect our Soldiers from these large caliber threats is known as Active Protection (AP).

ACTIVE PROTECTION

Active protection is the process of detecting, tracking, and physically defeating a threat at a distance sufficiently far from the defended vehicle to assure its survival. The process begins with knowing a threat is coming at a friendly vehicle. Sensors capable of this include Electro-Optic (EO) missile detectors, Infra-Red (IR) launch detectors, and radar search sensors. These "cueing" sensors look into the battlefield for a phenomenon typical of an incoming threat. This might be the escaping gasses from a chemically induced threat launch, emission of a bright flash from a gun barrel, Doppler shift of a radio frequency caused by threat flight, or some other threat signature. Battlefield clutter can cause false alarms by these sensors, so a high-level of signal processing is typically done by the cueing sensor to verify the existence of an incoming threat. Once the cueing sensor is confident a threat is approaching, the responsibility is handed off to a tracking sensor.

A tracking sensor follows the threat to acquire further information. The type of AP system determines what information is critical to successfully protect the vehicle. It is common for the tracking sensor to determine the threat's exact target. If the threat will pass by the vehicle, it may be best not to engage the threat. Other typical information tracking sensors determine is range from the vehicle and velocity. These help determine when to attempt threat intercept. Radar and Ladar (a sensor similar to radar, except that a laser is used rather than a radio frequency beam) are common tracking sensors. The tracking sensor passes the critical data to a processing unit that commands the countermunition to take action against the threat.

Numerous countermunition approaches have been researched. Airbags are one approach to catch the threat. Firing a gun from the vehicle back at the threat can potentially break the threat into pieces. Using a pure-blast of explosive is another approach. The blast creates enough energy to destroy the threat and/or induce a shift in the threats path for it to harmlessly pass by the defended vehicle or into the ground prior to target impact. Blast-fragmentation countermeasures utilize the same theory, but add fragmentation to break the threat into pieces with a higher probability. Some AP systems detonate their explosive from the vehicle hull. However, intercepting threats at a significant standoff distance from the vehicle can, in some cases, increase their likelihood of success. This allows more flight time for the threat to break into pieces and divert from its intended path. However, an accurate, stabilized launcher coupled with a fast delivery system can be necessary. The Integrated Army Active Protection System (IAAPS) has demonstrated successful end-to-end intercepts of Rocket Propelled Grenades, Anti-Tank Guided Missiles, and tube launched High Explosive Anti-Tank rounds. This system uses an EO cueing sensor, radar tracking sensor, and a rocket delivered, spin-dispersed buckshot warhead to predetonate the incoming threat. Development is continuing on this system to improve its defeat capabilities. One drawback to a system relying

on a rocket flying out to the threat is the timeline necessary for the processing, aiming, and rocket fly-out. This leaves the vehicle vulnerable to very close-in attacks. The most common threat employed in these scenarios, excluding Improvised Explosive Devices, is the Rocket Propelled Grenade. The U.S. Army is developing the Full Spectrum Active Protection Close-in Layered Shield (FCLAS) to fill this gap in protection.

FULL SPECTRUM ACTIVE PROTECTION CLOSE-IN LAYERED SHIELD

The Full spectrum active protection Close-in Layered Shield (FCLAS) is an active protection system capable of defeating shoulder fired rockets and ATGMs fired at even very close ranges. An active protection engagement is considered a success if the protected vehicle is not penetrated by the threat. This requirement is especially challenging with the U.S. Army's objective combat system weight of only 20 tons. The FCLAS system is being developed to pose a minimum integration burden on a multitude of platforms. Potential platforms that can benefit from the system are the Stryker, High-Mobility Multi Wheeled Vehicle (HMMWV), Family of Medium Tactical Vehicles (FMTV), Unit of Action variants, and others. The U.S. Army has historically developed active protection systems that intercept threats farther away from the vehicle. The need for FCLAS became apparent in recent years with the proliferation of RPGs around the world and the influx of urban warfare scenarios experienced by U.S. troops.

COMPONENTS

Three components make up a fully functional system: countermunition, launcher, and system controller. Each component processes data and communicates with the other pieces of the system. Onboard processing at each component spreads the computational load across multiple processors. This allows for parallel processing of critical functions, such as threat tracking, while other processing can still occur, such as handling user commands.

Countermunition

The countermunition has the responsibility of searching for, tracking, and intercepting threats. Figure 1 shows a mockup of an FCLAS countermunition. An sensor staring into the battlefield looks for potential threats in the immediate area of the vehicle. If a threat is detected, the signal processor begins tracking the round and predicting whether the threat poses a danger to the host vehicle. The threat will be ignored if it is predicted to miss the vehicle, i.e. pass over or around the platform. Otherwise, the countermunition makes a request to the system controller to engage the threat. This is done to prevent multiple FCLAS rounds from attacking the same threat. After permission is granted, the countermunition launches itself towards the incoming threat. The forward looking sensor is turned off and the side looking proximity fuze is activated once the countermunition leaves the launcher. This sensor stares radially out from the countermunition looking for the threat. Ground clutter is filtered out and once the threat is aligned with the countermunition, a fire signal is issued to the warhead. The warhead detonates and creates high-velocity fragments traveling in a tight pattern radially away from the countermunition. These fragments are designed to penetrate the warhead casing of the threat and sever it from the flight body (see Figure 2). This side attack method virtually eliminates the chance of triggering the crush fuze of the threat, which can cause the threat warhead to detonate and still be lethal to the crew. The countermunition is fire-and-forget, which means once is leaves the launch tube, it no longer needs commands or a data link from the vehicle.

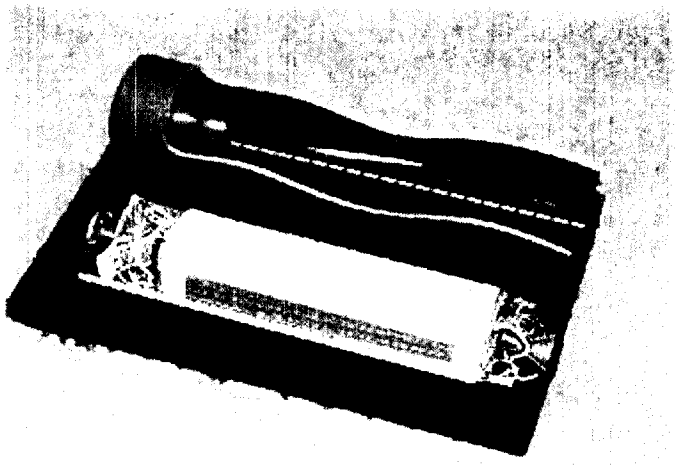

Figure 1. Mockup FCLAS countermunition.

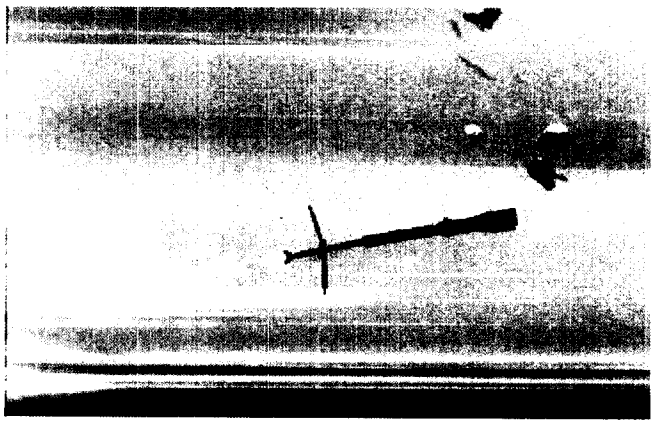

Figure 2. RPG after intercept by FCLAS countermunition.

Launcher

The launcher houses the countermunitions and is a communication node between the controller and countermunition. Each tube is 66mm in diameter enabling the FCLAS launchers to be downward compatible to launch standard smoke grenades. Figure 3 shows how an FCLAS launcher may look. Since multiple defensive weapons can be loaded in these

tubes, the launchers detect what type of round (FCLAS, smoke, or other device) is in each tube and relay that information to the system controller. Launcher design is flexible and can be adapted to meet the needs of each platform integrating FCLAS. Tubes can be aimed with tight angle separations for more overlap of coverage from countermunition to countermunition or wide angles to reduce the number of countermunitions needed for 360 degrees of coverage. A layered approach can add redundancy to the system. Aiming two tubes in the same direction allows the system to react to one threat fired at the vehicle and then to a second threat fired from the same direction. This requires more countermunitions, but has benefits that may outweigh its burdens. These options are under the discretion of the program manager using the system.

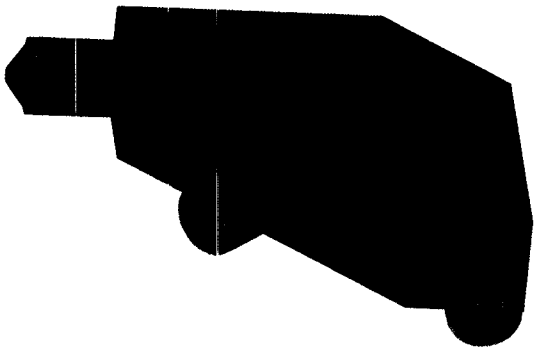

Figure 3. Conceptual FCLAS launcher design.

The FCLAS launchers currently utilize a Controller Area Network (CAN) bus to communicate with the system controller and other launchers. The 40 mega-bit/sec data rate used by the system is sufficient to perform common communication tasks with the system controller (i.e. launcher health status, munition identification, and enabling/disabling tubes). The most time-critical function performed by the launcher is passing a request to launch from a countermunition to the system controller and the concurrence or denial message returned to the countermunition. This requires a high-speed mode with data rates yet to be determined.

System Controller

The system controller maintains central control over the FCLAS launchers and countermunitions. A graphical user interface (GUI) view of the FCLAS system on the vehicle is used to present relevant data to the user and accept user commands (see Figure 4). The GUI shows which tubes are loaded and what type of munition is in each tube. Should a munition be launched, the countermunition being launched will be highlighted and the direction of the incoming threat is shown to the user. This data can also be made available to the fire control system of the vehicle for "slew-to-cue" operation. An automated gun system can slew itself in the direction of the threat to return fire. The now vacant tube is shown on the GUI after a short time delay. The crew can then operate knowing what zones are not protected by FCLAS and choose to reload the tube at the appropriate time. Exclusion zones can be setup using the GUI if the vehicle crew knows where supporting dismounted troops or friendly vehicles will be operating relative to the vehicle.

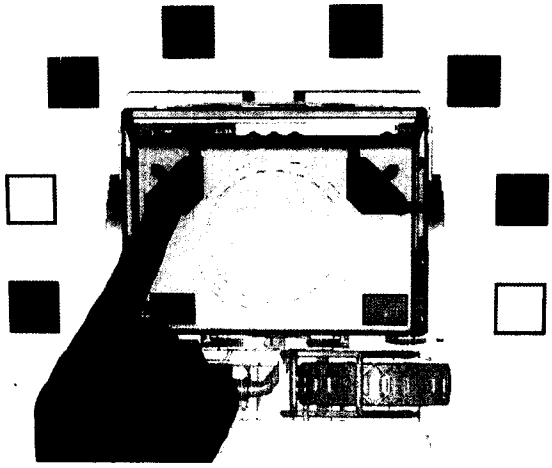

Figure 4. Graphical User Interface used for FCLAS controller.

Exclusion zones are areas around the vehicle where FCLAS is prohibited from engaging threats. This is done by enabling or disabling individual tubes or launchers using the GUI. Exclusion zones help prevent fratricide and collateral damage. For example, all tubes on the right side of the vehicle may be disabled if infantry soldiers will be clearing buildings on that side. Another example is that exclusion zones may be appropriate if a vehicle will be traveling in a linear convoy formation to prevent FCLAS from launching to the front or rear of the vehicle. Commands to launch munitions can also be directed via the GUI.

If smoke grenades are loaded in FCLAS launchers, their status will be displayed on the interface. The user can press a button to launch all or certain smoke grenades using the FCLAS controller interface. This operation is critical to obscure the vision of the enemy or the vision system on a guided missile. The GUI then provides feedback to the soldier by indicating which tubes have launched grenades and their empty status after a short time delay. The system controller also performs functions transparent to the user.

The controller requests a Built In Test (BIT) be performed by each FCLAS countermunition upon being loaded into a launcher. If a BIT fails, the user is informed of the malfunctioning munition via the interface. The controller is also responsible for preventing multiple FCLAS rounds from engaging the same threat. Each FCLAS munition sends a request to the controller to intercept a threat prior to launching. Tracking data of the incoming round is embedded in the message, so the controller can determine if multiple FCLAS rounds are

tracking the same threat. Once the controller concurs with a round to launch, it then prevents successive rounds from launching against the same threat if their tracking data indicates they are detecting the same incoming threat. All of these functions are critical to proper operation of the active protection system and performed without user guidance. The user may have the option of enabling a dismounted soldier tracking system to further minimize the chance of fratricide.

The FCLAS controller includes functionality to accept future warfighter capabilities to track dismounted troops. The locations of blue force troops in the immediate vicinity of the host platform can be displayed on the controller's GUI in real-time. Seeing the locations of friendly forces can enhance a vehicle commander's ability to fight. Tracking soldiers around the vehicle facilitates dynamic exclusion zones. This feature can turn tubes on and off depending on the locations of the dismounted troops. A dynamic exclusion zone improves on the capabilities of programmed exclusion zones in that only the very minimum number of tubes will be disabled at any time, thus protecting the vehicle to the highest level without endangering friendly forces.

The remainder of this paper will describe some of the general technologies used in the FCLAS controller, and then will go into detail on the specifics of how the application is constructed as a whole. It first goes over the requirements of the controller, then the Model-View-Controller (MVC) Pattern, Open Services Gateway initiative (OSGi), and P3ML. Then the paper discusses the operation of the controller and gives specific information on the software architecture chosen.

CONTROLLER REQUIREMENTS

The requirements for the FCLAS controller were basic at first glance:

- Create a graphical interface that could be used to arm and disarm individual munitions, or entire launchers.
- Dynamically update controller code as Java bundles.
- Configure the FCLAS system for different operations such as a supply convoy and "walled" troop protection.
- Leave a small size footprint.
- Be processing resource conscious.
- If possible, track soldier's movement around the vehicle and determine whether or not munitions should be disarmed if a friendly soldier is nearby.
- Connect to a bus to control the FCLAS launchers in any configuration.

Various subtleties became apparent upon further scrutiny. One such case is in the logic that determines when munitions should be armed and disarmed automatically. One such example is when a friendly soldier is outside a vehicle and there are no occupants inside. In this example it makes sense to automatically disarm the countermeasure with the hope that the incoming round or the outgoing counter-munitions will not hurt the soldier. However, if there are many people in the vehicle and only one soldier outside, it may make sense to fire the countermeasure to save the vehicle crew. Doctrine will have to decide the rules for a production controller. For now, the controller has the architecture appropriate for inserting the logic as it is developed.

In designing the application, great care was taken to make the system flexible so that as new requirements were added, the software could adapt quickly. In many cases throughout the application, Java *interfaces* were used and implemented as services. A Java interface is a "contract" among independent Java objects that allows them to interact with each other. It establishes a protocol of behavior for any object that wishes to communicate to a particular interface.

MODEL-VIEW-CONTROLLER (MVC) PATTERN

The Model-View-Controller (MVC) Pattern is a way of dividing up software into distinct functional components to allow for flexibility and easier maintenance. This pattern breaks out the business logic (model), the user interface (view), and the controller.

The model contains business logic and is usually looked at as the internal "state" of the system. The view is simply the interface where the user navigates through the application. The controller receives requests from the view and decides which model it will execute next. The controller sets the next stop for the interface when the business logic completes.

An MVC pattern is a good design method because it allows developers to focus on specific pieces of the overall system. If a section has to be replaced, the controller only needs to change the flow of control when the appropriate action is needed. If the user interface needs updating, it will make appropriate calls to the controller. Thus, the loosely coupled sections become easier to update and maintain.

OPEN SERVICES GATEWAY INITIATIVE (OSGI)

While there are many wide-area network and home networking standards, there has been no service delivery specification. The Open Services Gateway initiative (OSGi) specifications provide the 'glue' in this new value chain, through an open-platform independent framework and API's that allows for the dynamic delivery of managed services with secure, scalable and reliable metrics.

OSGi is an alliance between companies such as BMW, Nokia, Motorola, Sun Microsystems, and IBM

Corporation as well as many others. Their common goal as reported on www.osgi.org is to "specify, create, advance, and promote an open service platform for the delivery and management of multiple applications and services to all types of networked devices in home, vehicle, mobile and other environments." In a nutshell, the plan is to come up with a platform to deliver Java software to networked devices "on the fly."

There are several implementations of the OSGi specification. IBM created the Service Management Framework (SMF), Gatespace Telematics has the Gatespace Telematics Distributed Service Platform (GDSP), and the open source community has Oscar. There are several other open source and commercial companies that offer versions of the OSGi spec.

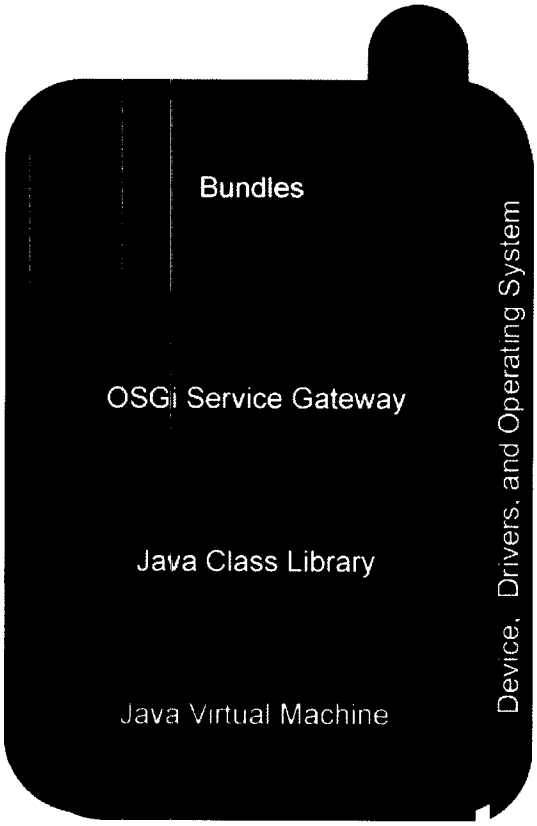

Figure 5. OSGi Software Stack.

OSGi takes special Java packages and makes them into one of the core OSGi building blocks called a "bundle." Bundles are distributed through a "bundle server."

A bundle must be able to start up and shut down cleanly. That is, it must initialize itself properly and clean up its resources when it terminates.

Bundles make extensive use of Java interfaces, exposing an Application Programming Interface (API) for other bundles to use. The interface signifies the notion of a "service." By abstraction through an interface, a specific class can *implement* the interface in any way the programmer wants.

In essence, a bundle is no more than a Java JAR file and a MANIFEST.INF file which is simply a file that describes the bundle, services it imports, services it exports, and what the bundle contains.

Bundles can be installed or uninstalled dynamically as they are needed by either devices, vehicles, or by other bundles.

A real benefit of using a distribution method such as OSGi is that bundles can be updated on the fly. A technician can remotely install bundles onto multiple vehicle telematics devices or on many handheld devices at once. Once more efficient or intelligent algorithms are developed, they can be sent to remote devices quickly.

When a new configuration of launchers is put on a vehicle, bundles can be sent to the vehicle to adjust accordingly.

P3ML – THE VIEW

In the case of the FCLAS controller, the view is handled by P3ML. P3ML is a lightweight graphical toolkit that was developed by OTI and is now included in the IBM WebSphere Studio Device Developer Integrated Development Environment (IDE).

With P3ML, it is possible to build custom Graphical User Interfaces (GUIs) made up of bitmaps and lightweight widgets. Skinning is possible to change the look and feel of the GUI.

P3ML uses eXtensible Markup Language (XML) to describe the layout and resources for an application. The XML is then parsed and rendered for use by a Java application.

A screen shot of the actual FCLAS controller can be found in Figure 6 below.

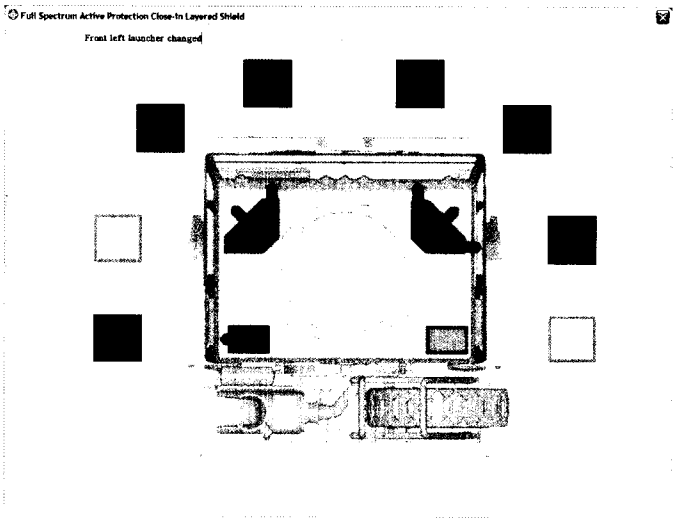

Figure 6. P3ML Graphical User Interface.

CONTROLLER OPERATION

The controller consists of an embedded touch screen computer and a GUI. The touch screen computer connects to either a bus or another communication method such as a wireless network. The launcher is then connected to the embedded computer through the bus or wireless network.

There is a view of the vehicle from the top where launchers and munitions are depicted. There is a color scheme to denote when launchers and munitions are armed or disarmed. Blue signifies a launcher or countermeasure is disarmed, red is for the armed condition, and white or gray means the launcher or launcher tube is empty. Smoke grenades can be signified by adding another color.

When the screen is touched, the P3ML interprets what action to take. If a launcher is touched, it will toggle all munitions within that launcher from armed to disarmed, or alternatively from disarmed to armed. The user may also arm or disarm individual countermeasures by pressing one of the squares surrounding the vehicle. If the system detects a friendly soldier outside the vehicle, a signal is sent to automatically disable the countermeasure. The doctrine of who has precedence over the operation of the FCLAS system, the signal from the soldier outside, or the vehicle operator, must be better defined for a production system.

SPECIFIC ARCHITECTURE

The FCLAS controller makes use of the MVC pattern where applicable. In this particular system, the controller used a Model-View pattern instead of the MVC pattern. Because the business logic and the controller logic fit so well in one class, it made more sense to include them together. The Java code bundles sit on top of the OSGi Framework, in this case on top of Service Management Framework (SMF). The FclasP3mlHmi bundle takes care of the user interface as seen in Figure 6. It makes the proper calls to the P3ML bundle to create the graphics on the screen. Figure 7 depicts the stack that has been implemented. The specific roles of each bundle can be found below.

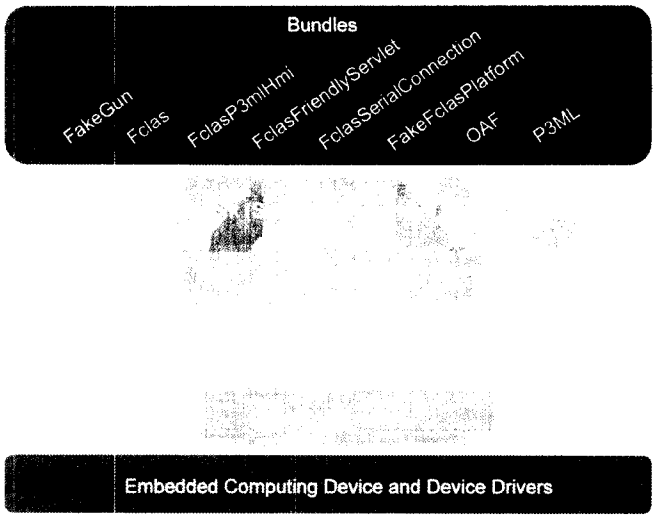

Figure 7. FCLAS Software Architecture.

The bundles such as FakeGun and FakeFclasPlatform implement interfaces; FakeGun represents the FCLAS munitions and the FakeFclasPlatform represents a vehicle. By creating bundles this way, it is possible to create a bundle that simulates the behavior of real FCLAS components without having the hardware. Then the fake bundle can be replaced later when the hardware is available. Many different types of FclasPlatforms and guns can be developed as long as they adhere to the interface specifications, their implementations can be hidden from the application.

FclasFriendlyServlet is a bundle that receives positional information from the friendly soldiers as they move around the battlefield and near vehicles. The Servlet accepts soldier identification, latitude, and longitude as its parameters. The Servlet sends the information to the launchers so they can arm or disarm particular rounds.

FclasSerialConnection does the "talking" between the embedded computer and the FCLAS launchers via a serial cable. It implements a service called FclasConnectionService. FclasConnectionService is an interface that specifies how the controller talks to the launcher. This allows the application to connect to the launcher via any communication such as a serial cable or a vehicle bus. When it is time to connect to a J-1939 data bus, an FclasJ1939Connection class can implement the FclasConnectionService, and the rest of the application won't have to change.

The Fclas bundle binds to the chosen real or fake platform. It also monitors the platform location and the locations of friendly soldiers around the vehicle. It unbinds the platform when the bundle terminates.

The OAF (OSGi Application Framework) bundle is a layer that sits on top of the OSGi Service Gateway that is used to make programming to the OSGi platform easier. OAF simplifies the creation and destruction of

bundles, increases reliability and predictability, reduces development time, and reduces training costs.

CONCLUSION

Today's battlefield continues to introduce increasingly lethal threats and tactics meant to harm U.S. Soldiers. A key technology to assuring their survival is Active Protection. The Full spectrum active protection Close-in Layered Shield (FCLAS) system is an application of this technology meant to provide a significant increase in protection without the weight and space burdens posed by conventional armor. The countermunition and launchers are critical to the success of the system, yet can not function without the added capabilities of the controller.

The controller makes use of real and emerging technologies such as Java, OSGi, and P3ML, as well as the MVC pattern for software development. The architecture simplifies development, allows fleet-wide application distribution with minimal effort, and provides platform independence. Because of this architecture, fleets will be more agile in and out of theater, maintenance time and cost will be reduced, and new applications will be fielded more quickly.

REFERENCES

www.osgi.org

OSGi Topics by Simon Archer

P3ML Topics by Simon Archer

CONTACT

Mark Salamango (salamanm@tacom.army.mil) is the Chief Pervasive Architect working for the U.S. Army Tank-automotive Research, Development, and Engineering Center - National Automotive Center in Warren, MI. Currently working in the area of Pervasive Computing (PvC) he has obtained a BS in Computer Science from University of Michigan in Ann Arbor. Mark has held various prestigious positions in industry such as IT/IS Director, Level II Technical Consultant, and is the President of his own consulting corporation.

Brian Rathgeb (brian.rathgeb@us.army.mil) is a Project Engineer at the U.S. Army Tank-automotive Research, Development, and Engineering Center in Warren, MI. Brian earned a BS degree in Electrical Engineering from Kettering University in 2001. He has five years of experience working on Active Protection systems for the Army.