2023 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MOSA TECHNICAL SESSION
AUGUST 15-17, 2023 - NOVI, MICHIGAN

Demystifying Selection Criteria for Embedded Military Vetronics Software

Achieving Determinism in a MOSA-based "Project Convergence" Environment

John Warther

Green Hills Software, Gaithersburg, MD

ABSTRACT

Selecting component software for next generation vehicular and payload electronics is an increasingly difficult challenge. There are many culprits, including increased complexity at the silicon level that can ultimately enable the software defined "tank" of the future. This paper will address software criteria and development processes required to deploy a standards-based, net-enabled military ground vetronics capability and provide demonstrable foundational technology.

1. INTRODUCTION

Today's military ground vehicle platforms, which are at the heart of safeguarding NATO and allies around the world, are faced with a myriad of challenges to ensure warfighters can defend or take ground against an increasingly competitive peer adversary. With a military vetronics budget "projected to reach $8.22 billion by 2027" according to Fortune Business Insights, investment in new C4ISR capability and vehicle electrification will make providing the embedded software supply chain with choice paramount to success due to the platform dependencies.

In addition to traditional challenges of ensuring lethality and delivery of precision guided munitions, next-generation platform vetronics and payload weapons are increasingly reliant on embedded software that must be designed, developed, and deployed with Modular Open Systems

Architecture (MOSA). This requires embedded software design teams to think in entirely new ways and make difficult trades in the process. Stove-piped systems will no longer be considered in future programs. Further, next-generation open architecture-based systems MUST be resilient to cyber-attacks and utilize proven technologies that have demonstrated they are not vulnerable to nation state malicious actors intent on denying warfighter capability. This demands that any software run-time environment have certain the following fundamental characteristics if they are to be successfully deployed in theatre.

1. **The underlying technology must have security built in from the ground up using tools and processes based on high-assurance design principals.**

In an address to Carnegie University on February 27th, 2023, Jen Easterly—director of US Homeland Security's Director of the Cybersecurity and Infrastructure Security Agency made the following comment regarding her third pillar:

> *"The leaders of technology manufacturers should explicitly focus on building safe products, publishing a roadmap that lays out the company's plan for how products will be developed and updated to be both secure-by-design and secure-by-default."*

Leaders in the government have recognized that retrofitting security is NOT a viable option going forward. An integrated development environment (IDE) that is proven in past use (pedigree) and current to latest technologies is required for optimum productivity.

2. **The IDE must keep pace with hardware complexity**

Current hardware complexity, e.g., complex systems on chip (SoCs) or electronic control units (ECUs), requires that military vetronics developers be able to piece together multiple development utilities that can operate across disparate environments. This in turn not only requires that an IDE be tightly integrated, it must also include sophisticated developer tools that go well beyond a typical compiler, debugger, and editor. This means having an IDE with embedded features such as a static source code analyzer, run-time error checking, and an instruction set simulator.

3. **The run-time environment provides a foundation that is certified to be reliable and safe (across multiple standards)**

New ground platforms and upgrades to existing platforms requiring run-time software (payload or on-board vetronics) should require the underlying baseline be based on a foundation that has a security pedigree and service history that can meet unique military requirements. It must ensure the run-time environment can support cross domain information transfer, host type-1 encryption algorithms, or host applications in real-time that are partitioned from one another and mitigate threats.

Suppliers must have a record of being capable of certifying their software on the most sophisticated CPUs or MPUs from our most trusted silicon supply base. The CHIP Act, recently passed by Congress in a bipartisan manner, will ensure that military vetronics prime contractors have a trusted supply chain available in the future, built on US soil, with state-of-the-art capabilities. On February 15th, 2023, Texas Instruments (TI) announced the following:

DALLAS and SALT LAKE CITY, Feb. 15, 2023 /PRNewswire/ -- Texas Instruments Incorporated (TI) (Nasdaq: TXN) today announced plans to build its next 300-millimeter semiconductor wafer fabrication plant (or fab) in Lehi, Utah. The new fab will be located next to the company's existing 300-mm semiconductor wafer fab in Lehi, LFAB. Once completed, TI's two Lehi fabs will operate as a single fab.

4. **Middleware solutions must be tested and integrated for common as well as unique interfaces and be proven to operate in a deterministic manner.**

   The embedded community has a plethora of vendors, all claiming to have the latest gadget to meet a new requirement. Without proper device drivers and testing, the capability may be meaningless. As system integrators layer additional third-party software into their command-and-control baseline, not only must they think about the capability it brings, they must also verify whether it meets fundamental tenets that can lead to an ATO (Authority to Operate) and avoid costly lessons learned from the past.

2. PAST LESSONS LEARNED: FUTURE COMBAT SYSTEMS

We must never forget the lesson of Future Combat System (FCS) in the early 2000s. At that time, to ensure no single dependency arose with respect to operating systems and hardware, a large middleware capability was developed called SOSCOE (System of Systems Common Operating Environment). Its goal was to be a service layer to provide isolation between application services, operating systems, and computer hardware and to be the single development environment and deployment environment for the Tactical Domain (Figure 1). It was also to be the critical information assurance piece for Army FCS.

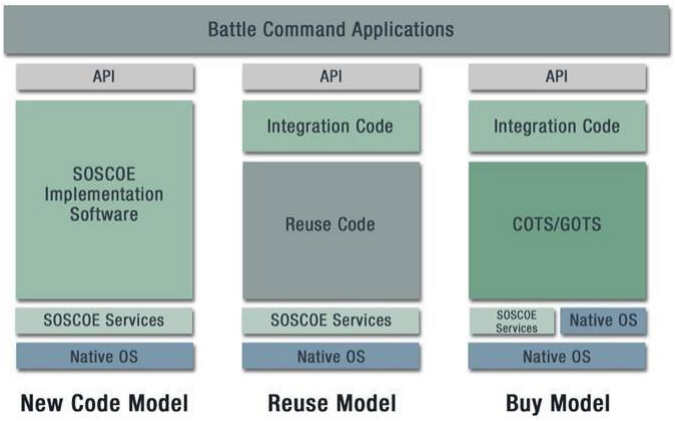

Figure 1: Early Goal and Army Vision (SOSCOE)

During the development of SOSCOE, it became increasingly clear that despite a noble objective, the sheer size of the code base (Figure 2), coupled with dependency on a single prime contract, made moving forward an unaffordable task. The result of the program was documented in a GAO report as follows:

"DOD concurred with this recommendation and stated that expectations for the 2009 milestone review would include an analysis of network technical feasibility and risks. However, in 2009, DOD decided to terminate major portions of the FCS program and to execute the remaining portions, including the network, in drastically different ways. Hence, this recommendation is no longer applicable."

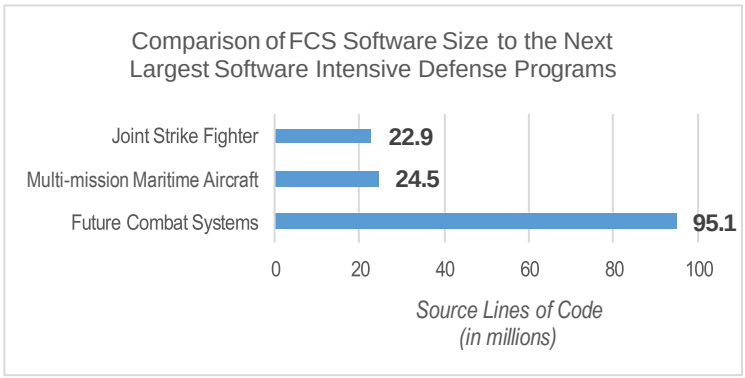

Figure 2: FCS Software Size Comparison

Right idea, wrong implementation. The middleware layer itself can be a monolithic code base with little capability to certify at any meaningful level.

3. LOOKING TO THE FUTURE

The US Army's vision of project convergence cannot afford to make the same mistakes. With the DOD's overall goal of building a joint all-domain command and control concept that is the basis of military doctrine as well as a secondary goal of linking manned and unmanned platforms that communicate a plethora of sensor data to

weapon system platforms, a divide-and-conquer methodology, with small certifiable components, is needed.

As these systems will operate across domains of land, sea, and air—as well as in cyberspace and multiple electronic spectrums, determinism again is an essential bedrock requirement to make the network function in a viable manner.

During the 2021 COMEX 2, *Inside Defense* highlighted the following key implementation attributes that feed what requirements the Army's vision of a Joint All Domain Command and Control (JADC2):

- What technologies enable the joint force to penetrate and exploit positions of advantage within enemy anti-access/area-denial?
- What emerging technologies contribute to how the joint force fights in joint all-domain operations?
- How do we incorporate artificial intelligence, machine learning, autonomy robotics, common data standards and architectures to enhance decision speed and multidomain maneuver at the joint tactical edge?
- How do we establish a joint network with required bandwidth that is sufficiently resilient and responsive in a delayed/disconnected, intermittent and limited (D/DIL) environment?

The common thread throughout vehicular electronics, their payloads, and their need to communicate in a JADC2 environment, is determinism.

4. DETERMINISM IN EMBEDDED SYSTEMS DEFINED

Defining and measuring determinism is achievable across all domains when the correct processes are invoked utilizing the best software development tools targeting a trusted (secure) and reliable (safe) run-time environment.

Getting this foundation right matters as multiple applications running at different safety and/or security levels on the same processor (either on the same core or different cores) will require sophisticated, mature capability. Support for resource allocation, fault detection, and fault isolation to prevent unintended interactions between independent applications is fundamental to support military requirements going forward.

5. INTEGRATED DEVELOPMENT ENVIRONMENTS FOR MILITARY VETRONICS

Army vetronics programs continue to be faced with SWaP constraints. Today's platforms must host new sensors, in-vehicle networks, onboard video distribution, onboard processing, safety critical systems, all which are part of a C5ISR (command, control, communications, computers, cyber (C5) intelligence, surveillance, and reconnaissance) systems framework. More than likely, these applications will be running on a disparate set of processing architectures from multicore CPUs to edge based ECUs. This requires system integrators to manage software development with the utmost visibility from cradle to grave. Depending on the vetronics functionality needed, the runtime environment may require a mix of operating environments (operating systems).

It is critical, given this complexity, that developers be outfitted with the best tools available. Software talent is a major expense and providing tools that enable software developers to be as efficient as possible is paramount. Just as you would never pay a professional roofer to use a hammer over a nail gun, software developers need best in class tools. For example, Green Hills Software has developed sophisticated

development tools that can step forward and backward in time during the debug process, allowing for software development teams to quickly identify vexing coding bugs that are difficult to uncover using "printf" methodologies. Years of internal research have resulted in tools that can now show how the program got to the current state and what the system was doing (History® viewer) and save a debugging session that can be easily shared with coworkers (Debug Snapshot) to increase productivity further.

6. COMPILER TECHNOLOGY

Generating efficient code for a plethora of target processors in the embedded vetronics environment is challenging given the limited space, weight and power constraints. This amplifies the need for optimizing compilers that generate efficient code. Green Hills Software has spent decades ensuring our compilers meet the demands of the military vetronics developer. Tools like CodeFactor™ optimization speeds your program's execution and reduces its size by removing redundant segments of code via subroutine calls and tail merges. Static basing provides the same benefits (faster speed, smaller size) by grouping data items to significantly reduce the number of load address operations.

Compiler technology must assist the software developer by considering security issues all through the development process. For example, the compiler should help identify potential integer overflows and format string vulnerabilities. Also, the compiler should help ensure control-flow integrity using "stack canaries" and "shadow stacks." Both capabilities help detect buffer overflows. But, due to performance-based contracting law, it is difficult to bake them into the requirement. In fact, how many times does a major defense program have requirements around the integrated development environment?

Also worth considering is the importance of mitigating the attack surface. Best-in-class compilers ensure code is usable and performs in the most optimal reliable manner to meet warfighter requirements.

7. DEBUGGING

Finding bugs early in the process is critical to any program (Figure 3). The impact on military programs has been documented many times. One study from IBS showed that catching bugs early as opposed to later in the process has a significant impact.

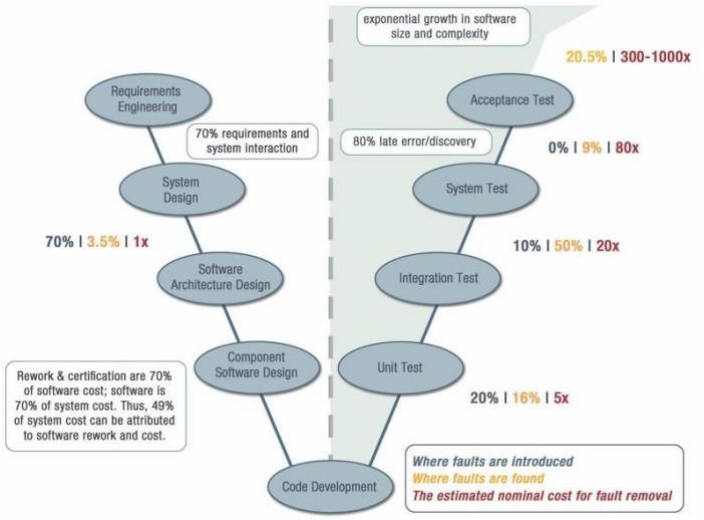

Figure 3: Impact of Software Rework on System Cost
Major cost savings can be realized by avoiding software rework. A $10K architecture phase correction saves $3M.

The US Army's "project convergence" will only exacerbate and expose these costs due to increasing software dependencies. Programmer efficiency that provides deep visibility into coding and finds the most complex bugs is critical to success. At a minimum, the following debug capabilities should be required for software-based ground development programs:

1. Make your software run faster than you thought possible by examining program execution history to find previously unknown bottlenecks.
2. Leverage gigabytes of execution history to verify that your program has been fully tested.

The above state-of-the-art development tools allow software development teams to address key areas where development performance often suffers:

1. System understanding from broad component behavior and interaction to small-scale multi-process interactions
2. High-level performance analysis
3. Bug triage, efficient handoff, and clear communication about technical issues

Performance-based contracting has made mandating or specifying these requirements a challenge to the supply chain. As military ground vetronic requirement planners consider how we bake in resiliency in MOSA based implementations, the software development tools used by our nation's most capable prime contractors should be up to the task.

8. RUN-TIME AND OPERATING SYSTEM FOR MILITARY VETRONICS

There are five architectural principles that military program managers and platform integrators should require when selecting an operating system capability onto a ground vehicle as follows:

1. Minimum implementation
2. Componentization
3. Least Privilege
4. Secure Development process
5. Independent Expert Validation

Commonly known as a separation kernel, these kernels have proven themselves viable in many high-assurance systems, where minimizing attack surface is a priority. Subsystems being integrated onto military platforms would benefit from these architectural principles.

9. MINIMUM IMPLEMENTATION

The earlier example of FCS showcases how easily a large military program can suddenly be mired in spaghetti code whereby vulnerabilities run rampant, and complex, convoluted code becomes the body of work that now must be implemented, usually without success. The hard work is in implementing a simple, eleqant solution that is evaluable and limits attack surface. Said another way, requirements should demand code implementations have as few source lines of code as possible.

10. COMPONENTIZATION

Figure 4 illustrates how today's state of art platforms will inevitably have to support a large body of software. That software in turn should be componentized in such a manner that a single engineer can understand the body of code. For the military, this is particularly important for the following reason: O&M (operations and maintenance).

By componentizing software, the Army will have well-documented interfaces between components and the issues of security functionality can be placed into separate components. Not only does this provide information assurance and risk management from adversaries but can aid in warfighter operations. Maintenance crews can still have access to pieces without the entire platform being unavailable due to issues around security access levels of maintenance personnel. Lastly, a properly componentized system will improve testability, auditability, data isolation and data limitation. It will also

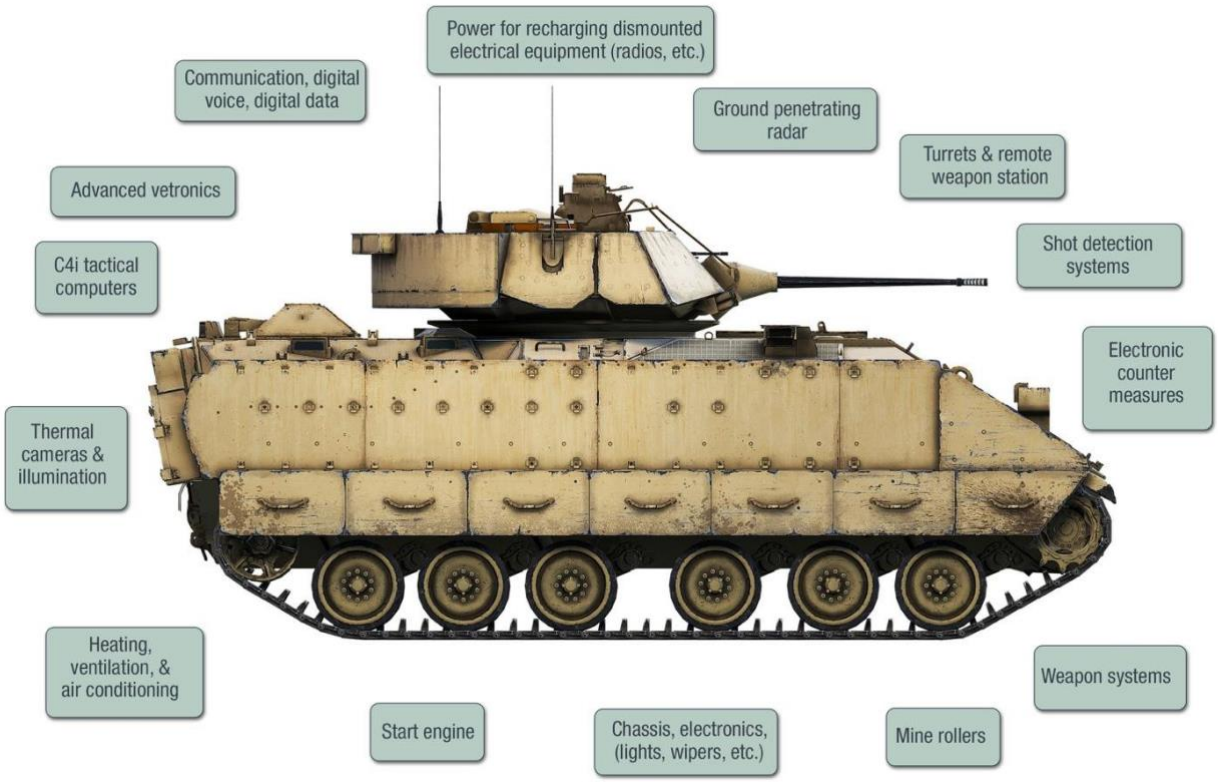

Figure 4: Military Ground Platform Components

mitigate a failure in one component from affecting another and ultimate system failure.

11. LEAST PRIVILEGE

Military ground platforms depicted above have many components and communication pathways. It is imperative that these systems are not only deterministic, but that access control for critical resources be mandatory. Information must be controlled. Monolithic operating system implementations treat access in a way that file systems, program access, and dependent system devices are too broadly accessible. Without applying the principal of least privilege, simple vulnerabilities may result in a buffer overflow that a peer enemy can compromise and then exploit with adverse consequences. As a refresher, least privilege is defined and properly implemented when a user is given minimum levels of access – or permissions – needed to perform his/her job functions. Military vetronics and systems on board are widely considered to require cybersecurity practices that protect privileged access to high-value data and assets.

Today's military ground platforms also must deal with mixed-critical environments for the microcontroller environment. Historically, many microcontrollers have used home-grown "bare-metal" schedulers with simple capabilities and without an actual operating system. However, as the ECUs and vehicle networks continue to get more complex, there is a growing need for such capabilities as networking stacks and more complex peripheral drivers. In addition, the functions operating on these simple microcontrollers have included critical tasks

that require safety certification as well as non-critical tasks.

As the number of tasks and their complexity and criticality has increased, those bare-metal schedulers cannot provide sufficient capabilities or hard real-time deterministic response times. Also, these microcontrollers are operating in environments that are constrained on such parameters as power, memory, and compute capability. This has brought about the need to use a proven safety-certified real-time operating system (RTOS) with minimal footprint and efficient scheduling. A safety-certified RTOS not only provides schedulers for the tasks, but also software interfaces for peripherals and networking stacks.

As the Army ground vetronics community continues on a path to implementations based on a CMOSS (C4ISR Modular Open Suite of Standards) architecture (Figure 5), the amount of control units will continue to grow. These edge devices may rely on smaller, single function ECUs and the commercial automotive industry has been a trailblazer in implementing these capabilities with safety and affordability in mind.

As more software capabilities get added to these ECUs, there is a trend to replacing some of the microcontrollers with more powerful microprocessors, and even multicore processors. These microprocessors not only have more compute capability, but also offer such features as a memory management unit (MMU) that allows for address virtualization. This virtualization capability allows individual tasks to be separated into isolated memory spaces that ensures that tasks operating in one virtual address space (VAS) cannot interfere with the operation of tasks in any other VAS, nor with the kernel of the operating system.

By combining a safety-certified RTOS and virtualization, it is possible to have an operating system that offers both critical deterministic real-time operation for safety-certified systems, but also offers a security-certified environment that mitigates interference for mixed-criticality systems.

12. MIDDLEWARE

Meeting the demand of military ground vetronics will also require a host of middleware solutions. To meet these demands, the underlying operating system must provide maximum security, reliability, and real-time performance as well as a comprehensive ecosystem of file system and networking stacks, protocols, and drivers to speed time-to-market.

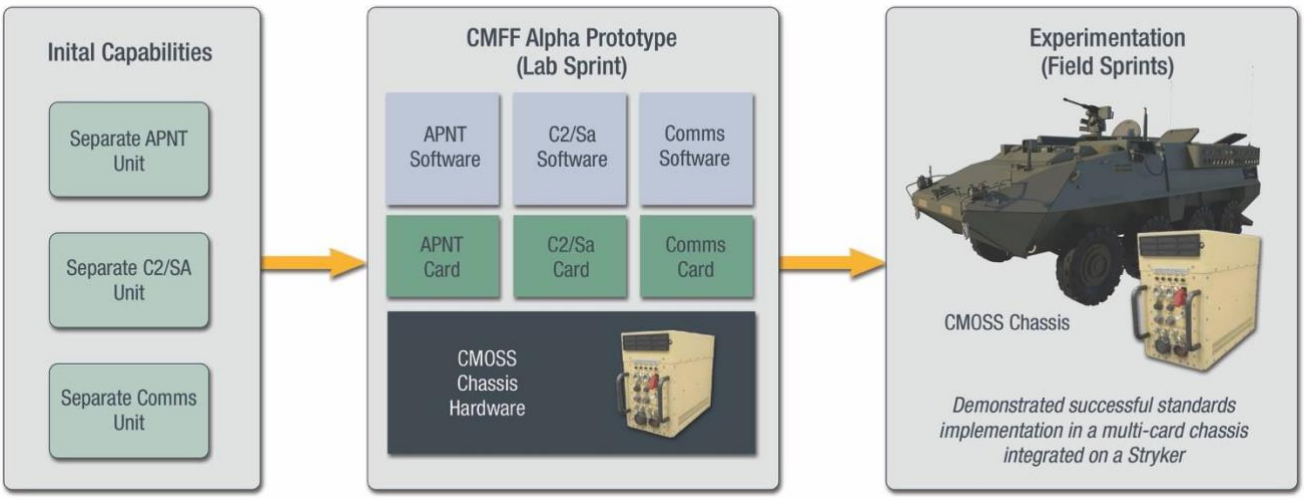

Figure 5: CMOSS Architecture

Green Hills Software remains a proponent of virtualizing device drivers and middleware capabilities. By doing so, ground vehicles can be assured that with a virtual device driver paradigm, the core kernel (run-time) is impervious to the bugs that are inevitable in the increasingly complex software found on military processing environments. For example, military vehicles, like in commercial automotive solutions, have a dependency on Controller Area Network (CAN) drivers. These device drivers can be integrated into the INTEGRITY® device driver model and support proprietary higher-level CAN-based protocols as well as standardized protocols such as DeviceNet and CANOpen; all while protecting the integrity of the underlying hardware resources. This maintains and assures reliability, security, safety, and determinism that the military platform will not be adversely affected. Even if a virus manages to get into the system, existing applications cannot be starved of memory or CPU time. Similarly, a bug or design flaw in a less critical component is isolated from affecting another independent component.

13. SUMMARY

Systems engineering and design requirements supporting project convergence and JADC2 in mixed-critical systems should, at their requirements level, ensure that a resilient system be architected on a foundation that is secure, safe, resilient, and deterministic. This requires that an embedded software supplier have a service history in delivering the tools, run-time, and middleware capability based on independent proven and verified accomplishments. Green Hills Software has over 40 years of delivering success into the community.