Customized Software in Distributed Embedded Systems: ISOBUS and the Coming Revolution in Agriculture

J. Lenz[1], R. Landman[2], and A. Mishra[3]

[1]Deere and Co., Moline, IL, USA.

[2]Phoenix International Corporation, 1750 Research Park Drive, Fargo, ND 58102, USA.

[3]University of Minnesota, Minneapolis, MN 55445, USA.

E-mail: lenzjamese@johndeere.com

ABSTRACT

The electrification of agricultural equipment has been evolving for many years and in some ways is lagging behind other industries. However this strategy of following the lead of other industries now offers Ag the opportunity to move forward at a revolutionary pace. Network standards defined by the Society of Automotive Engineers (SAE) and the International Organization for Standardization (ISO) committees are the basis for defining a rulebook for this industry-standardizing worldwide electronics interoperability. ISOBUS (ISO 11783) which defines a physical standard between tractors and implements will be an important enabler for most new product definitions. The foundation of this coming revolution will be provided through software. This paper outlines the electronics hardware and software architecture for off-road vehicles that allows for implementation of customized machine control features. There are several key areas discussed. The first enabler for this revolution is a software development and delivery system that defines a design methodology for creating and delivering software modules for a distributed set of controllers. This design methodology presents two advantages that today's modern electronic technologies can deliver: 1) Customization with commodity hardware and 2) Service without replacing hardware parts anywhere in the world. The second enabler for this machine revolution is an 'agile' process to develop the software. Many product ideas are being valuated through a trial and error and continuous improvement process. Software will play an important enabler for these product definitions. A comparison between the worldwide trend for software processes, the Capability Maturity Model (CMM), and what type of process would fit the off-road industry is based around the maturity of the new product ideas. The strong supply chain link between dealers and customers for off-road machines, coupled with the emerging awareness of electronic functions and controls, sets a basis for a specialized software development process. An important enabler for this 'agile' process is the re-use of code and incremental testing with reviews.

The history of the off-road machine business has been based on proven designs and long times between model updates. However, the worldwide adoption of the ISOBUS standard is poised to change this history. ISOBUS is not only establishing an open system for interoperability, it is establishing a sequence of features for diagnostics, sequenced operations, and information management. As customers discover these capabilities, they will expect them to be further advanced and customized for their specific needs. This requires adding agility into the proven durable processes so that manufacturers can respond faster to these growing needs. Electronics, and especially well-planned software systems, offer an agile technology for meeting this coming

J. Lenz, R. Landman, and A. Mishra. "Customized Software in Distributed Embedded Systems: ISOBUS and the Coming Revolution in Agriculture". Agricultural Engineering International: the CIGR Ejournal. Manuscript ATOE 07 007. Vol. IX. July, 2007.

need. This paper presents the benchmarking of various embedded software development projects relating project content, project rigor, and quality. From this, insights into maintaining quality are gained in order to include agility into a durable development project. Also, risk and rewards of leveraging low cost country software development skills are addressed to stretch resources or even develop common resources for software systems.

Keywords: Agile, durable, embedded systems, interoperability, ISOBUS, off-road, software

1. INTRODUCTION

Electronics allow easier operation and customization of machines for various operator needs. The initial step of replacing mechanical operator controls, such as levers, with electronic switches and computers are optimizing agricultural machine control in real time. Figure 1 illustrates the general path and pace in which electronic controls have evolved.

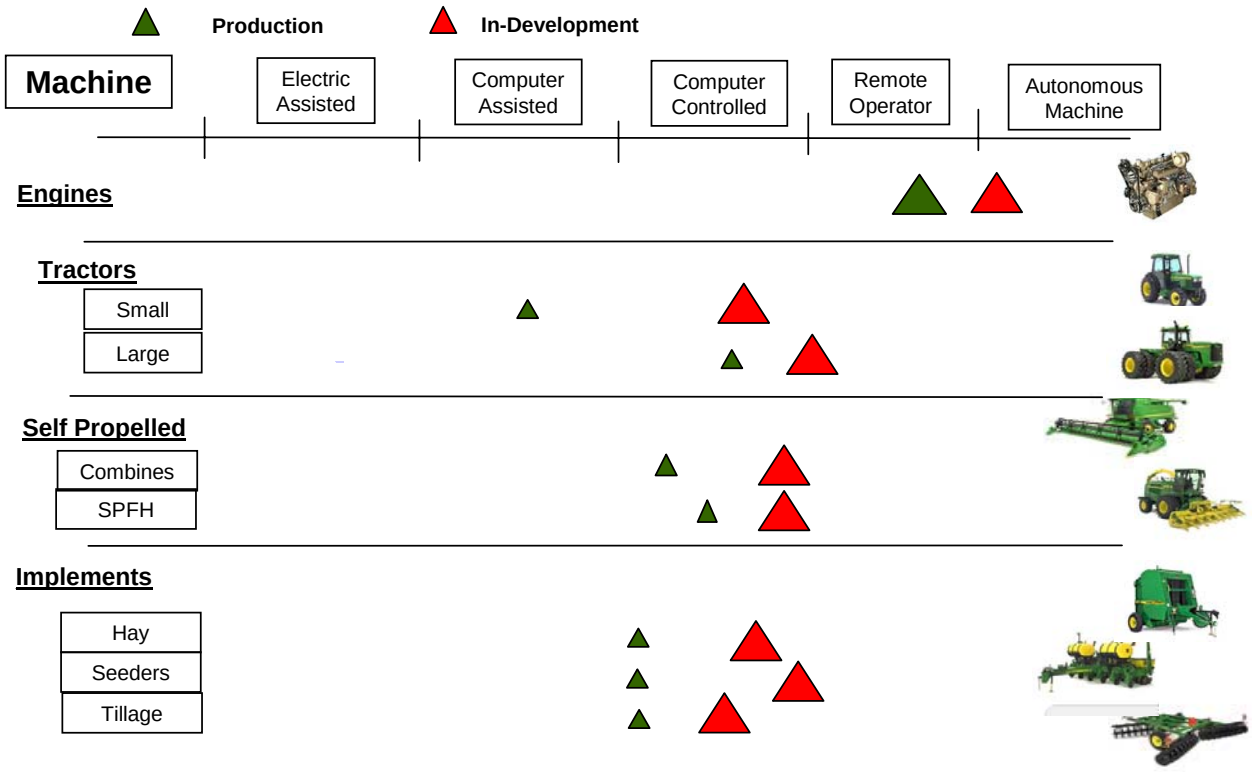

Figure 1. Comparison of machine electronics.

1.1 Electric Assisted

The first step in this evolution is defined as Electric Assisted. This step involves adding switches, wiring, indicators, and wire harnesses to the machine, such as electric start for engines. User controls can be more compact and sensors are used to monitor critical functions.

1.2 Computer Assisted

When machinery becomes Computer Assisted, logical and mathematical processing is added, such as the processing that takes place in engine position timing. The main advantage is added safety along with re-configurable labeling for fewer switches and more parameter settings.

1.3 Computer Controlled

Adding Computer Controlled electronics involves using sensor inputs to automatically control a function independent from the operator, such as engine fuel injection. Functions can be done repetitively with more precision.

1.4 Remote Operator

Introducing Remote Operator electronics into a vehicle involves adding self-reasoning to the control circuit, such as adaptive emissions-managed engine control. The control can diagnose its operation and adapt for optimum performance and safety.

1.5 Autonomous Machine

Creating Autonomous Machines involves adding awareness and intent to the decision process and control, such as an engine that runs on power demand.

Of the major ag-related machines, engine manufacturers have lead the way in adopting technology while tractors and combines have been close followers. With the coming ISO 11783 standards for ag equipment, implements are jumping to a level equal to tractors in sophistication. By incorporating ISO 11783 standards, an entire function and its precise, safe coordination within the machine, are transparent to the user who then can focus on larger needs. A brief discussion of the ISO 11783 standards is provided in the following paragraphs.

2. ISOBUS: THE EMERGING STANDARD FOR AGRICULTURE AND FORESTRY EQUIPMENT

ISOBUS is the term usually used to refer to the ISO standard 11783, the general standard for mobile data communication for the agricultural and forestry equipment industries (Benneweis, 2006). Equipment that is ISOBUS compliant promises to communicate seamlessly (i.e., plug and play) with other equipment to form systems of machines and implements that can be flexibly configured to meet user needs. A major piece of the standard deals with a virtual terminal, or VT, that can be used by various devices to display setup, status, and diagnostic information about a particular implement or subsystem. This permits vehicle manufacturers and third parties to build display devices that work with any number of implements regardless of manufacturer. This is a major departure from the proprietary systems often used in the past and creates major benefits for the end user. Competition will drive the market for best features and productivity enhancements via this standardized integration.

Another part of the standard addresses task controllers. This allows integration of the business (farm) management from the office to the mobile equipment. This enables on-demand and near real time use of precision farming. Location dependent prescription chemical application and

yield monitoring are just two examples. Still other parts of the ISOBUS standard deal with the details of network management, data integrity, data storage, and gateways between network segments. As a whole ISOBUS, has the potential to revolutionize the industries that adopt it. These industries will in turn deliver the benefits to users in the form of productivity and efficiency.

2.1 The Value of Electronic Controls

A typical architecture being encouraged by the ISO 11783 standards is illustrated in figure 2. It shows tractor controllers for five distributed control functions. A series of implement controllers can also be added to the system. A Virtual Terminal (VT) acts as a reconfigurable, shared interface to this distributed control system. This system approach creates a "plug and play" environment for tractor and implements (Lenz and Jensen, 2004).

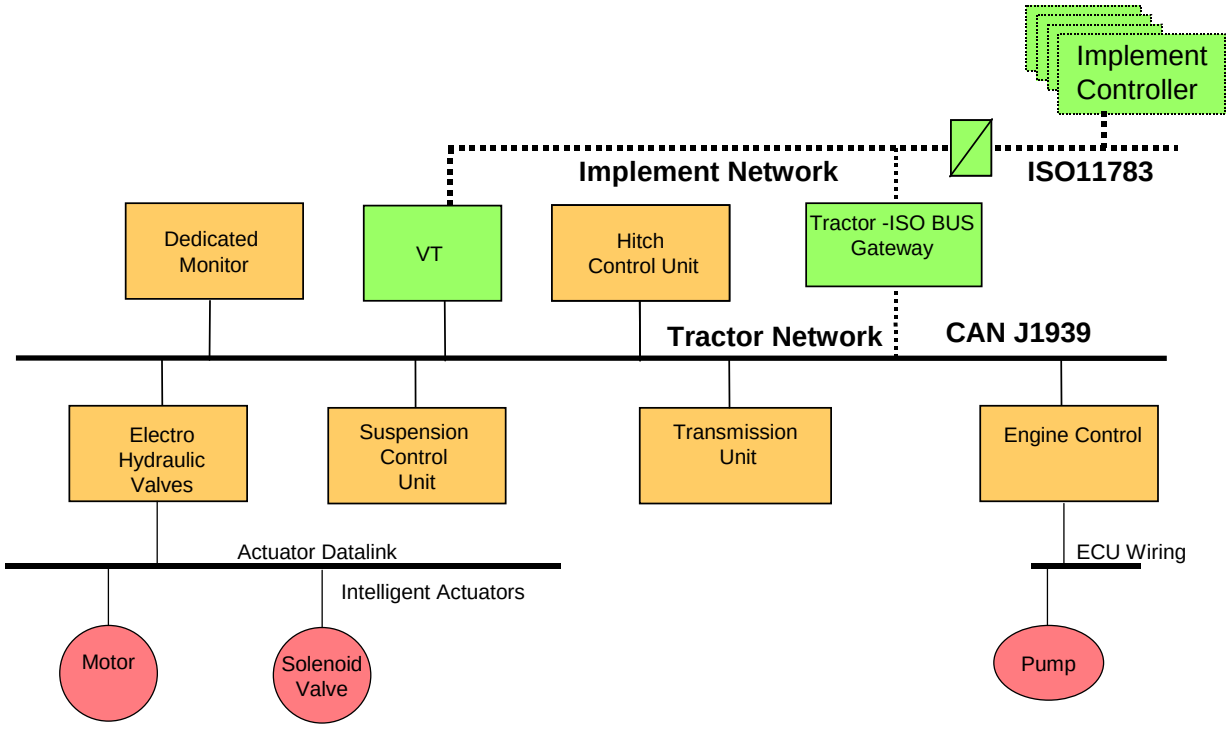

Figure 2. Distributed control architecture.

One challenge is to provide a system that is cost effective in comparison to the benefit it provides as well as to make it expandable for future needs. The benefits of electronics in machine control are best gained through use of common modules such as software, microcomputers, connectors, sensors, actuators, and housings. These common modules offer development and performance benefits such as:

- Easy customization of machines for each operators needs.
- Efficient service for customer updates and upgrades.
- Standard diagnostic procedures.

- Shorter development time enabled by modular development.
- Local real-time response and control.
- Isolation of faults allowing reduced but continued operations.
- Improved software quality and reliability.

The modular approach also reduces development and machine cost through:

- Reduced part numbers and hardware inventory.
- Reduced wiring and electrical connections.
- Standard processors (i.e. controllers) that can be configured through software.
- Modular hardware that is mass-produced even at relatively small off-road machine volumes.
- Modular software where re-use is maximized.
- Development workforce mobility.
- Reduced verification costs.

A key to maximizing the value of electronics in machine control is having one set of hardware modules that can be installed anywhere and software modules that can be configured and loaded at the end of the production line. The underlying architecture for this modular approach is distributed control (fig 2). The hardware modules, primarily displays and controllers, can be readily designed and maintained. However, developing modules in a way that they work seamlessly together is a significant challenge. Part of the problem is that multiple design teams - often in different locations, business organizations, or even separate companies - typically develop different modules. This is further compounded by the global nature of today's products.

2.2 Modular Structure of Embedded Software

The major software modules in a controller are illustrated in figure 3. There are two layers to the structure. The application layer is the specific software that causes the controller to offer unique, customizable features to the machine control. The infrastructure layer provides the standards for communication, security, memory management, processor function, and connections to the machine.

The infrastructure layer consists of four major modules:

1) Operating System,
2) Common Services such as Timing Management, On-board Debugger, and Fault Management,
3) Hardware Drivers such as Analog Acquisition, PWM Driver, and CAN Communications and
4) Security/ Re-programming Pass code.

For each controller configuration the application software module is linked with the selected infrastructure modules to form a controller payload. For a typical 128K byte payload, the application code will use 25% of the controller's memory and the re-useable infrastructure code will fill the remaining 75%.

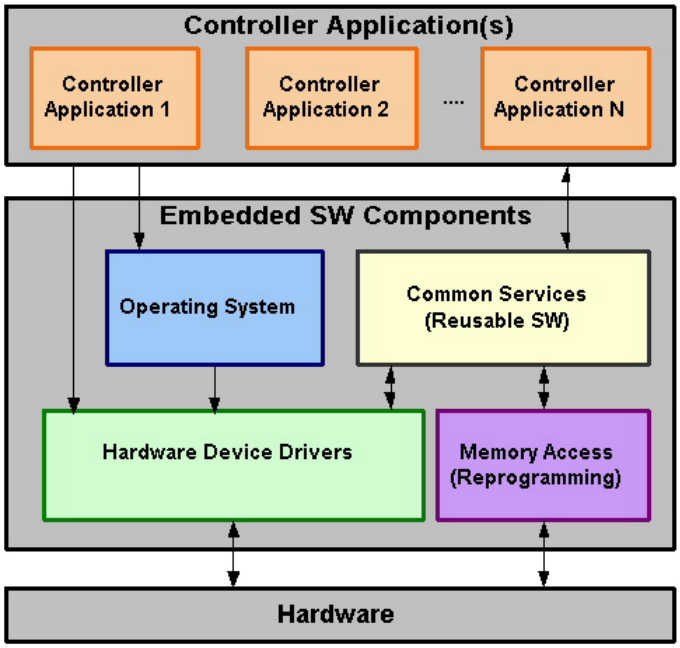

Figure 3. Modular structure for the controller software.

2.3 Software Architecture and Delivery

ISOBUS will require a capability for seamless development of software modules and delivery of these software modules to machines on the production line and in the field. This encompasses the four aspects of a life cycle of a controller software payload:

1) Final Programming - at factories,
2) Service Reprogramming - at the machine site,
3) Field Reconfiguration - at dealers or machine site, and
4) Aftermarket Features – direct to the machine.

Through use of the World Wide Web to deliver this customization, an IT infrastructure for embedded software for distributed controllers has been created. This Embedded Software Information Technology (ESIT) system consists of the following key parts:

- **Rulebook: CAN Management** — This allows every developer, regardless of organization or geography, to conform to the distributed control communication and data structures
- **Library** — This is the infrastructure layer software including the drivers and electrical specifications to the various switches, sensors, and actuators for our family of machines.

- **Memory Access/Pass Code** — This is the infrastructure layer software providing uniquely configurable security to prevent un-intentional access to re-writing or storing data or software code.
- **Formatter** — This desktop application allows engineering to build software payloads for a controller. Through parameter sets, payloads can be customized for nearly every user's requests.
- **Order Delivery: WebIT** — This web enabled server application allows for ordering and delivery of compacted software payloads for controller.

Figure 4 shows the flow of these parts and how they are used to create and deliver a controller software payload. This ESIT system has been designed to utilize the benefits of World Wide Web technologies and its global reach. It establishes a core competency for all future machine control systems. The latest features for a machine can be customized for an individual end-user if needed. The developers for new features have well-established infrastructure and programming guidelines. Ultimately this system provides the most efficient, most flexible, most re-configurable platform to developing controls and delivering them to customers for their specific needs.

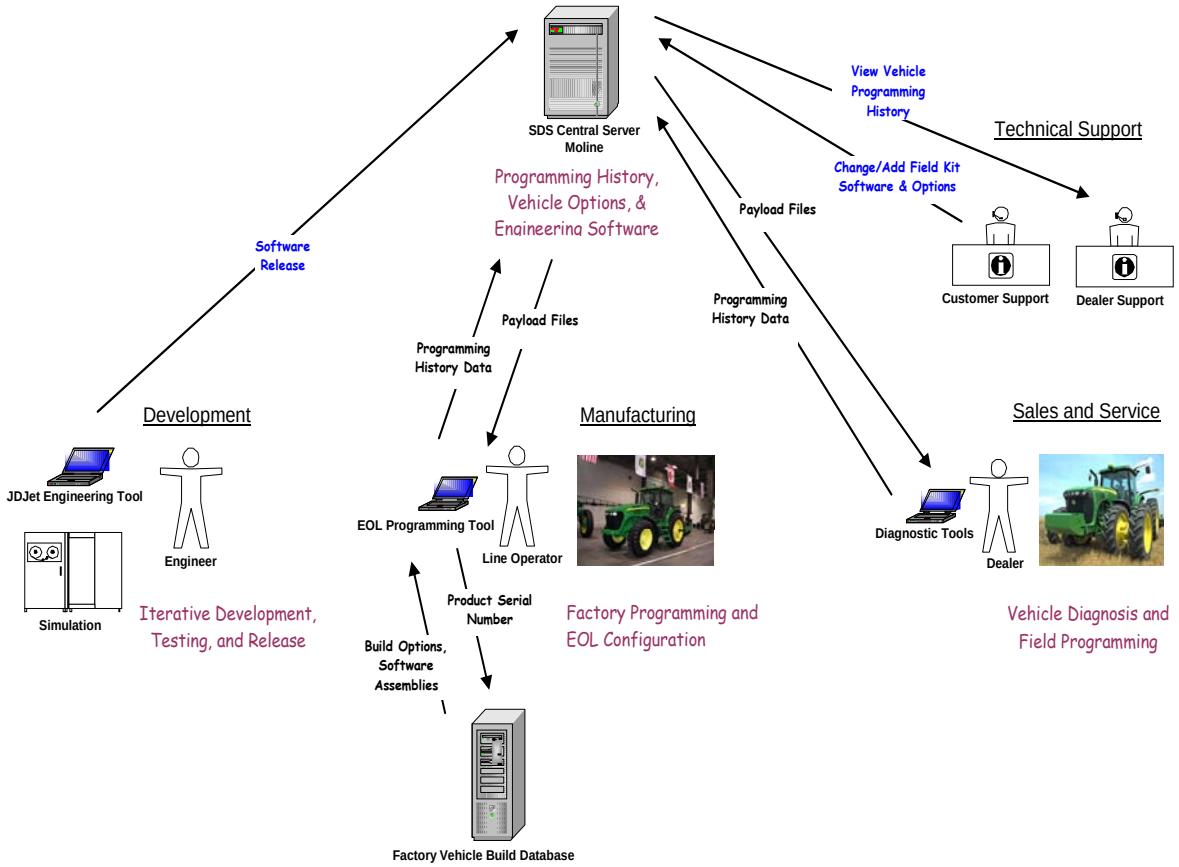

Figure 4. A critical function for an embedded software system is use of a single IT based infrastructure to connect development, manufacturing, and service.

The main objective of the software development methodology and delivery system illustrated in figure 4 is to tailor products to specific customer needs, both on the production line and in the field. Aftermarket sales of upgrades benefit from ease of software reconfiguration. Field support costs decline because technicians can remedy many machine problems through software updates rather than hardware replacement.

While the goal of the software system in an off-road industry is for long-term service and customization, this distributed architecture has other benefits. Substantial code re-use can result in a 75% reduction in software development time and cost while maintaining quality and efficiency. Also, because all software must conform to standards for interoperability, this single system drives a software development process leveraging a software service 'help desk' approach. Higher reuse and clear design guidelines have also translated into fewer software defects and smoother system integrations due to “plug and play” interoperability between controllers. The full value of this infrastructure can be achieved through shared usage similar to what is achieved with most IT technologies. It is expected that as ISO 11783 drives more interoperability between machines and suppliers there will be a need for a common infrastructure that is available to every developer and service center.

3. CHALLENGES FOR SOFTWARE DEVELOPMENT

Electronics is a growing part of the control and functions for off-road vehicles, and the use of software to configure the machines is an especially important capability. However the history of software development is often linked to projects having unmanaged costs and schedule overruns. As the off-road equipment industry relies on electronics to meet future needs, software development must be approached differently than other industries have done in the past.

3.1 Software Development Evolves to Meet New Business Models

In the 1980’s when software was being recognized as a recurring area of project management difficulty, there were two prevalent themes that underscored the problem: poorly defined and changing needs and a ‘discovery’ approach to software coding. Several major efforts to address this problem resulted in bringing maturity to the industries embracing electronics in the 1990’s. This maturity cycle is illustrated in figure 5. As the product is developed, the verification of the requirements and the validation the product needs is done early in the build cycle of the product. When customer needs are grouped and defined early in the development cycle, the product, verification, and customer expectation can be forecast and determined before the product launch even begins. Nearly all of the product specifications and requirements can be evaluated and optimized during the technical feasibility and market feasibility development steps. Once the product is launched there are few product specification unknowns. The resources can be focused on delivering to this specification. Most products fit into this type of business model. A characteristic of the type of business is that the learning curve and experience of the sales channel (dealership and customer) are not expected to contribute significantly to the product’s continuous improvement. In this case the software development can be defined like a component and developed accordingly. Today there are a number of standard processes for developing software where specifications can be defined and forecast with reasonable accuracy. Software still requires a ‘discovery’ process and as a result can attempt higher levels of complexity than hardware only approaches.

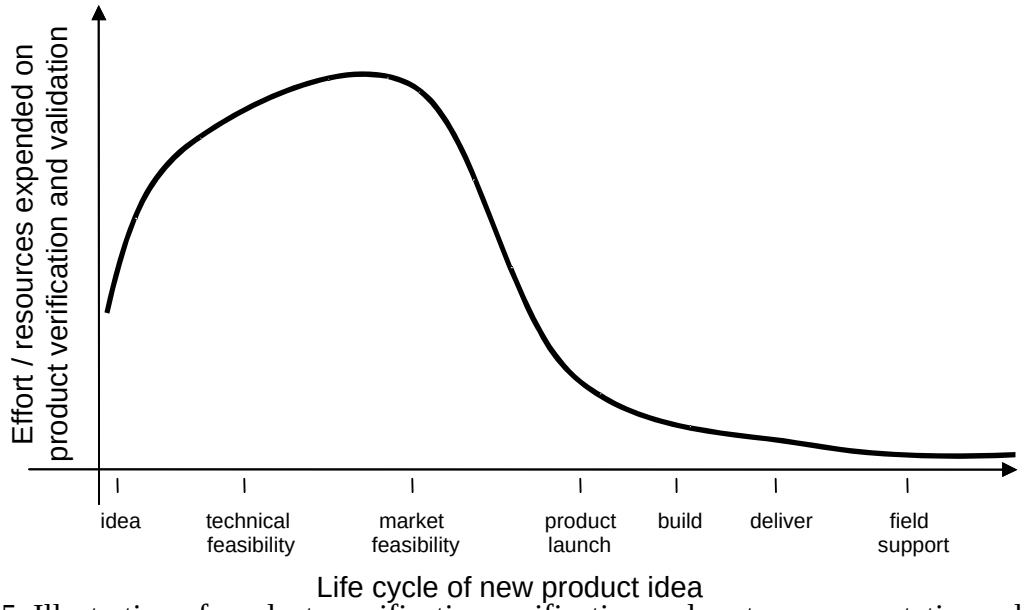

Figure 5. Illustration of product specification verification and customer expectation validation resources expended during the product development cycle for a definable need.

3.2 Setting Standards for Software Development

The growing standard for software development where needs are clearly defined has been developed by Carnegie Mellon® Software Engineering Institute (SEI). The Capability Maturity Model (CMM) (fig. 6) for software provides organizations with two sets of guidance:

1) Establishing processes for developing and maintaining software, and
2) Creating a culture of software engineering and management excellence.

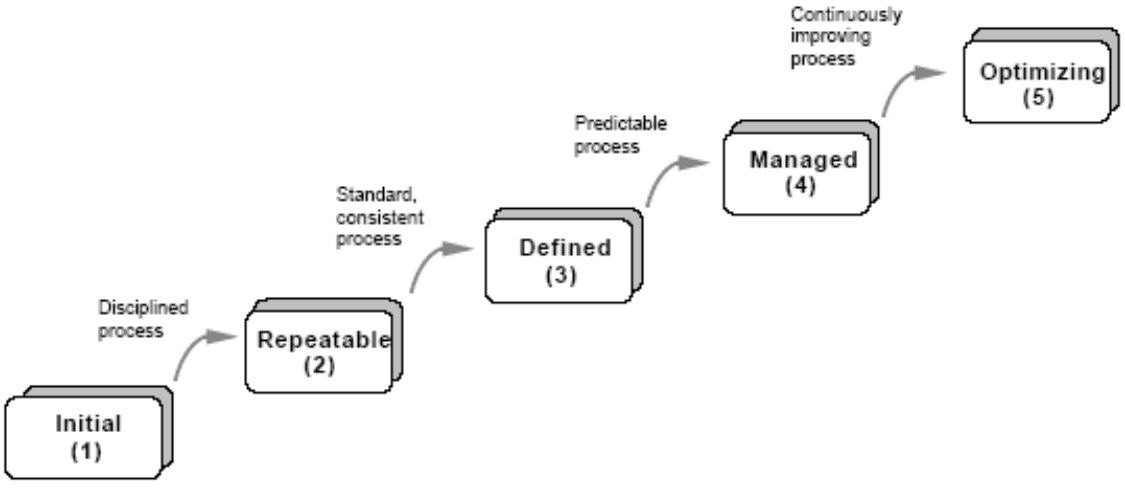

Figure 6. The five levels of software process maturity as defined by CMM.

The CMM (Paulk et al., 1993) was designed to guide software organizations in selecting process improvement strategies by focusing on a limited set of activities and working aggressively to achieve them.

The staged structure of the CMM is based on principles of product quality that have existed for the last sixty years. These principles have been adapted into a maturity framework that establishes a project management and engineering foundation for quantitative control of the software process, the basis for continuous process improvement. A characteristic of the type of business that benefits from CMM is when the learning curve and experience of the sales channel (dealership and customer) are not expected to contribute significantly to the product's continuous improvement.

However what happens when the experience of the customer is a main driving factor for new needs? How readily can an organization move up levels in this capability maturity model when the specifications for the product do not become evident until late in the life cycle? How can agility be offered with durable products? These are some of the questions facing the off-road equipment industry that drive a different approach to software development.

3.3 Durable and Agile Processes

In the off-road equipment industry the type of customer and their needs are quite diverse. They require precise performance for their individual applications. Since the market is fragmented, the supply chain (dealer and customer) has difficulty verbalizing general product requirements. This tends to drive the product verification and validation late in the product life cycle with much of it being done through the learning curves of the supply chain as illustrated in figure 7.

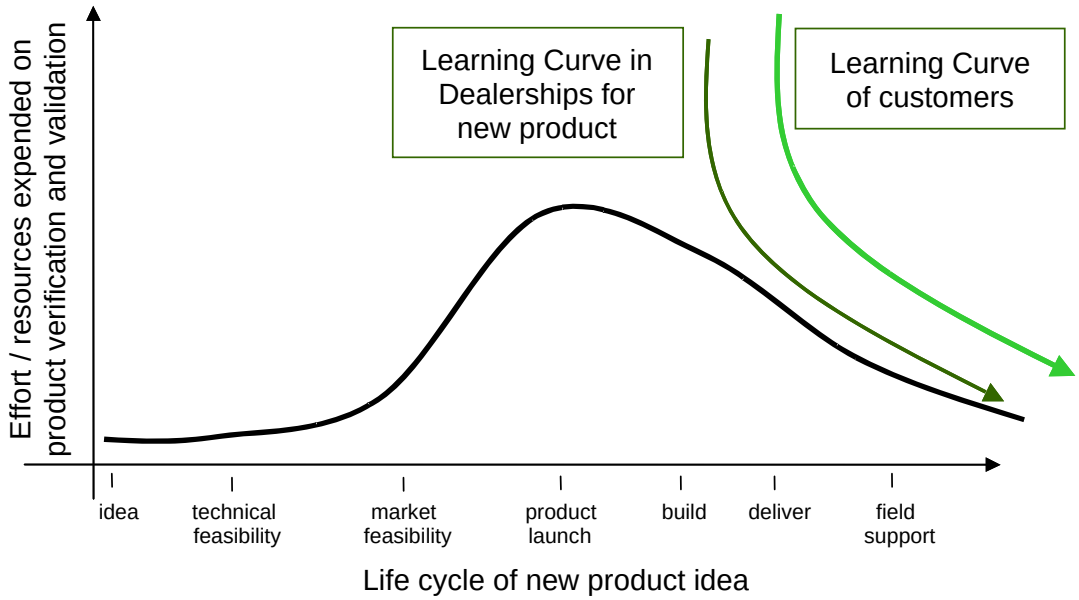

Figure 7. Illustration of product specification verification and customer expectation validation during the product development cycle for a fragmented market where final product specifications rely on the learning curves in the supply chain.

Another way to compare the differences between the durable and agile levels of market maturity is shown in figure 8. The 'room for discovery' as a metric is compared between the two types of software processes. A traditional product development process is based on a stage-gate method where innovation is managed as the product is defined for product launch. One can think of this as a Durable Software Development Process. However, software because of its virtual, re-programming structure can be adapted to a product readily if this has been anticipated.

This can be thought of as an Agile Software Development Process. The Agile process works for software because there is no extra tooling or manufacturing elements. Once the software component is developed and tested, it can be immediately inserted into machines. Figure 8 shows three software updates being done as the machine nears initial delivery.

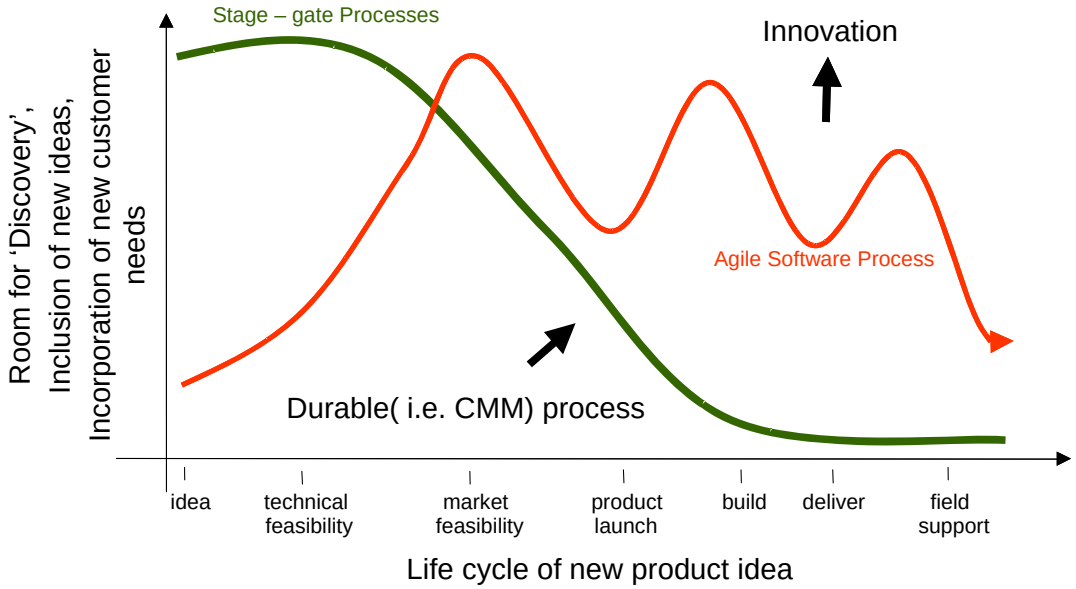

Figure 8. Software can deliver innovation late in the product life cycle.

The Durability Software Process, i.e. CMM levels 2 through 5 (fig. 6), has these characteristics:

- Reduced 'discovery', detail specification defined up front.
- Quickly drop 'poor' ideas in favor of 'good' ideas.
- Can control development schedule for substantial durability testing and meet product launch date.
- As components are defined can engage a competitive supplier bid process.
- Better manage costs, leverage use of low cost sources.

The Agile Software Process has these characteristics:

- Can deliver software very late in product build-innovation.
- Software quality must be achieved without long corner condition testing.

- Marketing and engineering works together to offer short time to market features.
- Difficult to outsource software production when the specifications are not fixed.
- Development costs are more difficult to manage.
- Ability to match the product to the true un-met needs is optimized when the un-met needs are difficult to verbalize before experiencing.

Many features of off-road vehicles must precisely fit the customer's need. The Agile Software Process provides a fast-learner method to respond to needs as they are discovered (Erickson et al., 2005). The main trade-off between the Agility and Durability approaches is if software quality is compromised. However, the customer does not readily distinguish between software quality and performance if the overall product does not meet their needs. But the true need is often difficult to specify until the operator, dealership, and the manufacturer experience a large statistical number of occurrences. When this discovery occurs, it has great value. The faster this discovery can be turned back into the product, the greater the reward.

3.4 Software Development

The five key steps to developing software modules for controllers are illustrated in figure 9. This diagram is referred to as the V chart. It depicts the software development process moving from a larger systems view of the software to the coding details and then back to the larger machine view. The left side of the V represents the systems engineering and the software specification and design. The right side of the V shows code verification to the specification and integration. The major effort is at the bottom in creating the software code. The key to managing costs using this process is to keep these five steps tightly coupled. This impacts the use of outside engineering resources. A key part to managing this process is the use of development tools such as document version control, requirement capture, and regression testing. However, one of the most important aspects of controlling costs and continuously improving software quality is establishing a code re-use system.

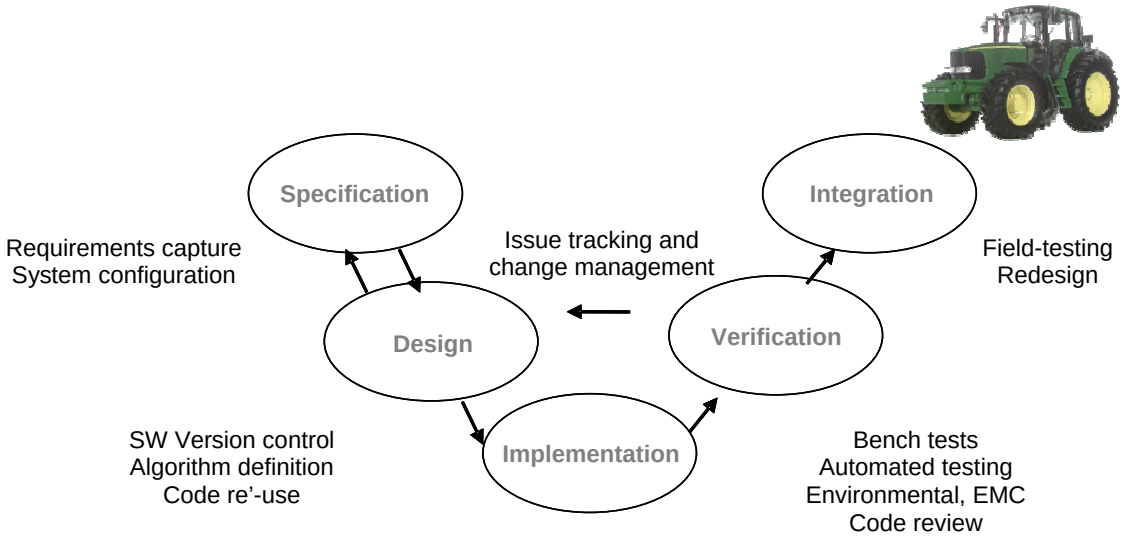

Figure 9. The five key steps to developing an embedded systems software payload.

These five major tasks, along with leveraging re-use architecture, defines an agile software development process. This is assisted through having a strong infrastructure for re-usable code and a tightly coupled development team. The Agile Process is a good fit for products where customer needs are emerging and difficult to specify until experienced.

CMM is portrayed as related to a durable but maybe not as an agile process. This reflects the general thinking as organizations have developed and improved their software development process. However the issue is not CMM, but adapting the process to what is most needed. Software in the off-road industry uses 'updates' and thus a process must have the rigor for durability but also the agility to serve these updates. The three general reasons for software updates are:

- Customer Needs: Software enhancements to address missing requirements, for example issues found through field use
- Design Issues: Software enhancements to fix hardware problems
- Software Quality: Software 'patches' to fix 'bugs': typos, formatting, timing, math errors

Figure 10 illustrates that the focus on an agile or durable process results from the level of the customer maturity. When the product and its software is serving an emerging use of electronics, having a 'fast learner' or incremental process is the best fit. When the value of electronics is well understood then the focus is on software quality. Software will always be used to fix hardware problems mainly because it can, but no one would set up a process specific for this purpose.

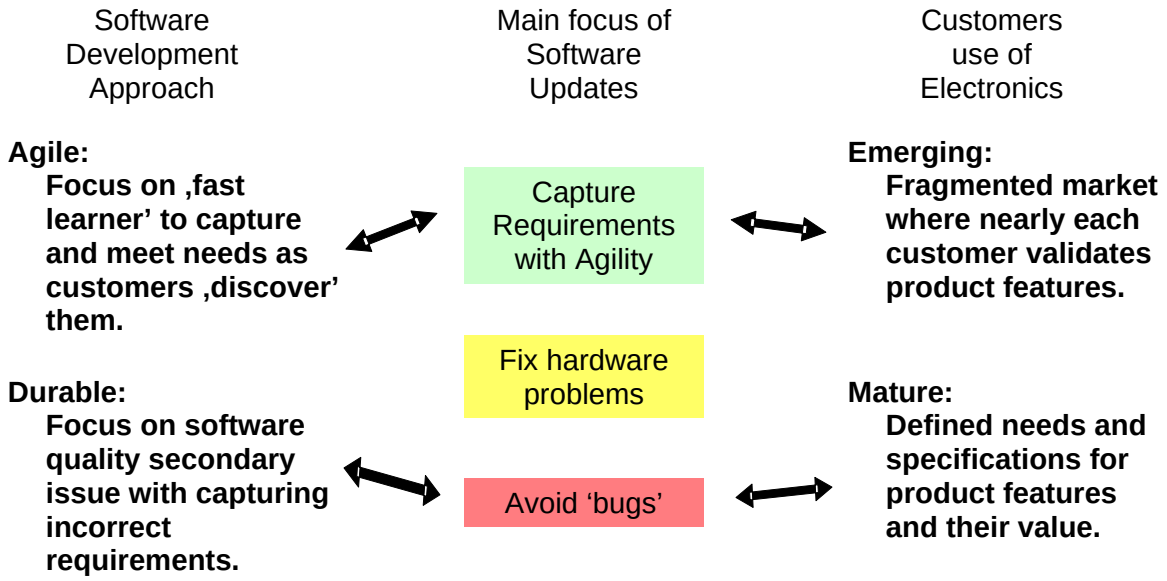

Figure 10. The maturity of the market being served is a main driver in having a software process that has its primary focus on agility or durability.

Many items required for CMM/CMMI (Requirement for Management, Process, Documentation, etc.) are also showing up in 61508, the IEC standard for safety relevant components. Thus the software development process for off-road machines will continue to be challenged between

delivering to very specific requirements and delivering into an emerging market where the value of the product feature is challenged and re-worked nearly every day.

3.5 The Challenge between Durability and Agility

Traditionally quality in electronics is built through long durability developments (Henning and Heide, 2004). These long development cycles serve the market need when the market pull for new features changes slowly (fig.11). Even this process tends to over-serve the market.

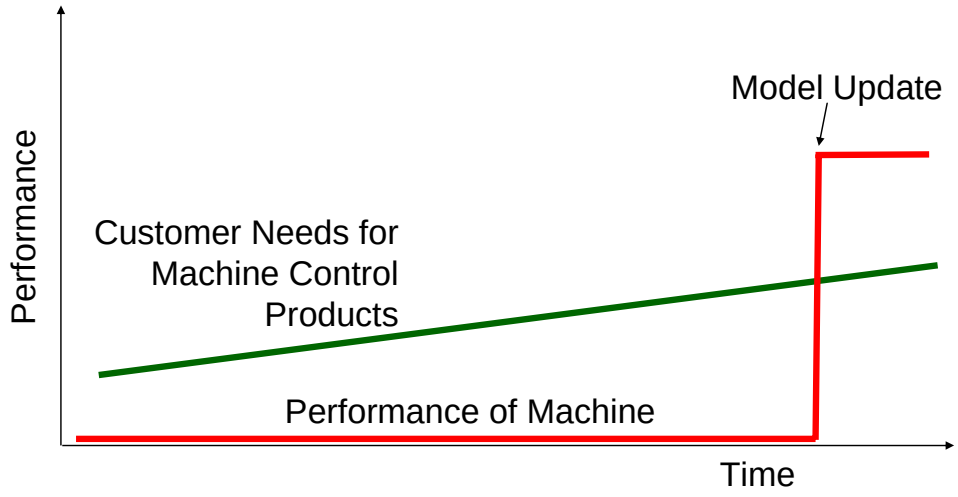

Figure 11. Durable Approach. A Durable Product Development approach serves the customer if their need for machine updates increase slowly.

Because of the length of time between updates, there is a strategy to go beyond what customers are asking for. Introducing a product that over-serves (provides more performance and features than the customer needs at the time) can produce a negative impression of the product. The concept of a durable process as defined in this paper represents the long time between feature updates. This includes time to define the requirements and validate to these requirements so that the product does not significantly over-serve the market. This durable process is then managed to produce a high quality product where the electronics are primarily performing machine control features.

With the coming of ISOBUS standards for sharing controls and interoperability, there is a growing need for features related to machine intelligence instead of just simple machine control. These intelligence features are emerging from many users and suppliers in the industry, fueled by the interoperability of ISOBUS. These features are primarily developed and delivered through software in the electronics.

To keep up with the growing demand, a more agile process is needed (fig. 12) which can more rapidly deliver when a durable process may mostly under-serve the current market needs. Our industry is being challenged to find a way to produce high quality, i.e. durable software in a faster, i.e. agile process.

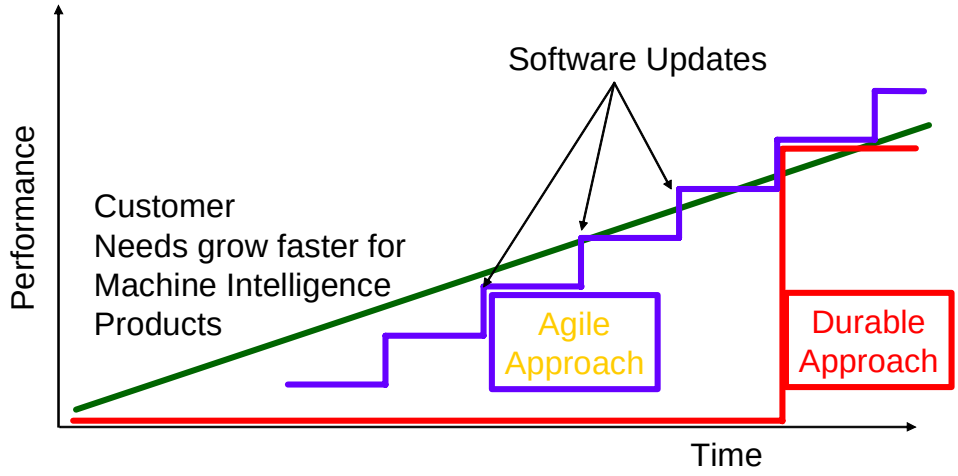

Figure 12. Agile Approach. An Agile Product Development approach serves the customer if their needs are growing faster than traditional machine updates can accommodate.

There are two basic foundations to adding agility to a traditional durability development process. The first is highly reliable reusable software components that can be quickly assembled to implement new features or completely new modules. Reuse of components such as vehicle network communications, reprogramming, fault management, user interface and many more give the machine designer a rapid development environment to build their individual module's capabilities (Lenz and Muench, 2005).

The second foundation is developer support. Developer support binds together the proven software architecture, standard software modules, and implementation details. Together these reduce development time and eliminate duplication of efforts. This approach is effective in avoiding repetition of costly mistakes that have already been solved. Developer support provides:

- A library of code software for selected microcontrollers that includes I/O drivers, hardware abstraction, and operating system.
- Re-programming software based on standard IT functions preconfigured for specific microcontrollers, memory, and hardware combinations.
- In-depth training on the above components.
- Application integration and debug support available by phone and email to all users.
- Continuous improvement of "core" software with needed features as determined by a user group.

3.6 Software Development and Quality

Quality is the motivation for a durable process. As developers face the demand of the schedule there are two options: add inexperienced staff, or tailor the process. In an emerging or 'immature' situation, it appears quality is achieved better by tailoring the process and maintaining a tightly integrated team. High quality is still achieved through four factors:

1) Using a software process,
2) Leveraging a high percentage of re-use in the interoperability software,
3) Help desk support that gives assistance for the coding details, and
4) Experienced staff that specializes in capturing systems domain knowledge in software.

The future goal is always perfect quality. As complex systems grow with still emerging values for electronics, even faster turns on ideas are needed while maintaining quality and customer expectations. Modeling and simulations are the next engineering practice to include Agility in a durable product development process.

4. OFF SHORING EMBEDDED SOFTWARE DEVELOPMENT

The tremendous growth in information technology hardware and software has opened up a number of alternatives for executing embedded software development projects. Teams of professionals armed with laptop computers, fax-modems, e-mail, voice mail, videoconferencing, interactive databases and frequent flyer memberships, are being sent out to conduct business in this global arena. With respect to this trend, offshoring of embedded software development work has increased significantly in the last decade.

While the subject of offshoring has received extensive coverage in business and academic press in recent times, a majority of these studies have focused on understanding this phenomenon in a setting that involves routine manufacturing and service activities (such as call center, medical transcription etc.) as opposed to knowledge-based work such as software development (Youngdahl et al., 2005). Offshoring in manufacturing and service industries typically involves handing off standardized tasks such as manufacture of component parts, administrative tasks, or back office operations to vendors, with little day-to-day interaction required between the local firm and the off-shore supplier firm. Yet understanding how to manage these activities is likely to be very different from knowledge-based work like embedded software development. As described above, the use of embedded controls in the off-road industry is an emerging capability where there is minimal routineness and a multitude of feature details. Finalizing the exact machine performance requires significant exchange of complex information between the customer, the service center, the factory, the systems engineer, and the software developer.

Offshoring of embedded software development projects present a unique challenge to the off-road equipment manufacturer (Rottman, 2006). On one hand, firms are likely to outsource their software development projects for reasons related to the low cost of labor in offshore destination countries as well the availability of manpower resources in these countries. On the other hand, given the uncertainty associated with software development and the increasing interdependence between development tasks, the entire development team needs to exchange information continuously. This brings forth the challenge of coordination and effective project management across geographical, temporal, and cultural boundaries (Overby, 2003). The difference in working styles across team members situated in difference geographical locations can also be a source of constant misunderstanding and frustration among the team. Herbsleb and Grinter (1999) found that members of distributed software development teams, regardless of the way they structured their work, were “constantly surprised” and confused about the activities of their distant colleagues. While there is some understanding of the strategic reasons (such as lower

costs of development, availability of technical expertise in popular offshoring destinations etc.) that motivate firms to offshore their development projects, it is not clear as to how such projects can be managed effectively. The above identified issues bring forth the need for more research on project management of offshoring teams especially in managing the relationship and the development of a shared understanding between the client and vendor teams.

5. SUMMARY AND FUTURE DIRECTIONS

The coming ISOBUS standard will revolutionize the Agriculture industry in many segments. This revolution will be driven by the customers pull for ease of use and operations that are more efficient. In order to deliver this solution across our industry there is significant complexity, as described above, which must be transparent to the end customer. The revolution will come from how the off-road industry deals with these complexities of which some are:

- **Single Service Capability** – Service electronics and finding root cause faults is challenging in distributed control systems. The ISOBUS standard starts with identification information but eventually common built-in test and data logging procedures may be needed (Darr and Hudson, 2004).
- **Software Interoperability** – The ISOBUS standard leads to strict formats for communication protocol (CAN J1939) and input/output commands (Virtual Terminals). It is software subroutines that deliver this interoperability (Hofmann, 2006). The lowest cost solution comes when a single source is used for this software. There is cost for each machine manufacturer to develop these software libraries with an even greater cost for each to maintain this code for years. This is a cost that the off-road industry cannot afford and if not managed will make ISOBUS unaffordable.
- **User Interface Commonality** – The most visible part of the ISOBUS standard to the customer will be the display and the control switches. The customer will want something that is intuitive and familiar across all machines (Haapala et al., 2006). The industry will need to work together to provide an ease of use format.
- **Hardware Common Modules** – Connectors are the beginning of the commonality and an obvious necessity for interoperability. However this need for commonality will grow to other hardware such as displays, controllers, sensors, wireless devices, etc. To gain confidence in the use and maintenance of this electrical system, the customers and service centers will want to become familiar with the components. Again as this commonality grows so will the pull for ISOBUS.

ISOBUS will help off-road industry manufacturers distinguish between their core competencies and the core values they offer the customer, and what just has to work 'behind the scenes'. The coming revolution will clarify this for each manufacturer in this industry.

6. REFERENCES

Benneweis, R. K. 2006. Facilitating Agriculture Automation Using Standards, Club of Bologna Session: Information Technology for Agricultural Machines, Sept. 2006, Bonn, Germany.

Darr, M., and K. Hudson. 2004. Standardization of Electronics in Machinery Systems - ISO 11783 nears completion for ag, construction, and forestry equipment, In: Resource - Engineering & Technology for a Sustainable World Vol. 11 No. 10, Dec. 2004, published by ASABE.

Erickson, J., K. Lyytinen, and S. Keng. 2005. Agile Modeling, Agile Software Development, and Extreme Programming. The State of Research, Journal of Database Management, Vol. 16 Issue 4, pp.88-100.

Haapala, H.E.S., L. Pesonen, and P. Nurkka. 2006. Usability as a Challenge in Precision Agriculture – case study: an ISOBUS VT. Agricultural Engineering International: the CIGR Ejournal. Manuscript MES 05 001. Vol. VIII. March, 2006.

Henning, K., and A. Heide. 2004. Embedded Software Failure Analysis. In: Bildverarbeitung - Safety and Security. 8. Symposium der Technischen Akademie Esslingen. Hrsg. v. Technische Akademie Esslingen.

Herbsleb, J. D., and R. E. Grinter. 1999. Architectures, Coordination, and Distance: Conway's Law and Beyond. IEEE Software, Vol. 16 No. 5, Sept.-Oct. 1999, pp. 63-70.

Hofmann, R. 2006. Software in Tractors: Aspects of Development, Maintenance and Support, Club of Bologna Session: Information Technology for Agricultural Machines, Sept. 2006, Bonn, Germany.

Lenz, J., and R. Jensen. 2004. Customized Software for Distributed Control, VDI Conference Agricultural Engineering 2004, Hannover, Germany, VDI-Berichte Nr. 1855, pp. 137-144.

Lenz, J., and P. Muench. 2005. Software Development for Off-road Machines, VDI Conference Agricultural Engineering 2005, Dresden, Germany, VDI-Berichte Nr. 1895, pp. 315-322.

Overby, S. 2003. The Hidden Costs of Offshore Outsourcing, *CIO Magazine,* 1 Sept. 2003. (http://www.cio.com/archive/090103/money.html).

Paulk, M., B. Curtis, M. Chrissis, and C. Weber. 1993. Capability Maturity Model for Software (CMM), Version 1.1, Technical Report, CMU/SEI-93-TR-024, ESC-TR-93-177, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Rottman, J. W. 2006. Successfully Outsourcing Embedded Software Development. IEEE Computer, Vol. 39 No. 1, Jan. 2006, pp. 55-61.

Youngdahl, W., K. Ramaswamy, R. Verma, and B. S. Sahay. 2005. Offshoring of Service and Knowledge Work. Journal of Operations Management, Jan. 2005, Vol. 23 Issue 1, pp. 109-110.