Reengineering: An Affordable Approach for Embedded Software Upgrade

Kenneth Littlejohn
Air Force Research Laboratory

Michael V. DelPrincipe, Jonathan D. Preston, and Dr. Ben A. Calloni
Lockheed Martin Aeronautics Company

Within the Department of Defense, embedded information systems found in aging aircraft are facing readiness, supportability, and upgrade challenges due to diminished manufacturing sources (DMS), which impacts both embedded processing hardware and software development tools. In many cases, however, the embedded software functionality within these systems is still highly viable. Under the Embedded Information System Reengineering project, the Air Force Research Laboratory Information Directorate and Lockheed Martin Aeronautics Company have matured a legacy software reengineering capability that eliminates software tooling DMS, permitting affordable application support and upgrade. A successful flight demonstration aboard a U.S. Air Force F-117 stealth fighter aircraft was recently conducted to verify correct performance of reengineered weapon system software components generated using high degrees of automation assistance.

Maintaining the viability of embedded information systems is a key technical and economic problem facing operators of aging aircraft. Within the Department of Defense (DoD), many currently fielded embedded information systems face readiness challenges imposed by evolving missions and extended service life spans. For example, emerging requirements for global situational awareness and rapid strike capabilities necessitate increased information processing and information exchange between command and control (C2) and weapon system platforms. However, the ability to overcome these challenges is constrained by such factors as shrinking budgets, limited computational capacity reserves, and the effect of diminished manufacturing sources (DMS).

Wholesale redevelopment is often cost prohibitive, particularly since large portions of embedded applications continue to fulfill mission requirements. In fact, most present-day upgrade programs involve incremental changes to an established design baseline, the majority of which is reused as is. Even when major functional overhauls are performed, budget and schedule realities usually dictate a phased approach. These realities underscore the need for an efficient means to carry forward, modernize, and exploit usable functionality within legacy software.

Solutions must maximize the recapture of prior design investments, provide efficient pathways for continued technology refresh, and accommodate changing technologies and economies of scale over decades-long service life spans. Purposeful migration toward insertion of commercial components mandates changes to existing business practices. The challenge, then, is to offer developers affordable methods of leveraging existing embedded information system applications to provide a foundation on which to base future systems. The Embedded Information System Reengineering (EISR) solution assumes that the end user is actively migrating to commercial hardware and operating systems.

The Air Force Research Laboratory Information Directorate (AFRL/ID) and Lockheed Martin Aeronautics Company have matured an integrated set of technologies that facilitate affordably maintaining and upgrading legacy systems and software. The EISR project has developed an automation-assisted JOVIAL-to-C reengineering capability that permits simultaneous modernization of both the structure and source language of legacy embedded applications.

The EISR environment has several key features: support for detailed analysis of legacy software, visualization of critical execution sequences and complex data dependencies, rapid source conversion, and a high-percentage source construct conversion rate. Using this capability, developers can rapidly characterize the overall legacy software architecture, perform incremental or wholesale source language conversion, and upgrade selected components and structures. Engineers can apply the proven labor-saving visualization and analysis features provided in modern commercial Computer Aided Software Engineering (CASE) tools to legacy JOVIAL applications. Following conversion, legacy applications can then be imported into other mainstream commercial graphical CASE environments that allow visual reconstruction and automatic source code generation.

To summarize, today's system developers face many general and high-level obstacles impeding evolution and modernization of these systems:

- Greatly extended platform service life spans.
- Rapidly changing mission scenarios, system roles, and threats.
- Increased information-processing requirements.
- Desire for cross-platform commonality of capability and architecture.
- Shrinking budgets.
- DMS affecting both application and software engineering environment hardware and software elements.

In addition to these obstacles and challenges, there are additional detailed design-level issues that must be dealt with in order to derive maximum benefit from large-scale reuse of legacy software.

In this paper, we assume upgrade scenarios where developers will migrate from military specific programming languages and development environments toward mainstream commercial replacements. Successful migration requires dealing with specific aspects of legacy applications and their development, which are outlined below.

Outdated Methods

Legacy systems commonly contain hierarchical, functionally decomposed, time-slice scheduled software architectures targeted to uniprocessor platforms. These designs are often highly coupled through global data pools as opposed to modern data-encapsulated object-oriented analysis/object-oriented design (OOA/OOD) forms. Such coupling hinders selective isolation and capture of proven functionality.

Lack of Modern Integrated Analysis Capability

Legacy development environments often consist of a patchwork of standalone textual/command line-based tools. Although there is support for symbol and dependency tracing, this capability is not comprehensive or integrated and is generally cumbersome, involving the use of several separate tools. As a result, it is difficult to determine the scope of a design change. This can result in increased risk and reduced confidence, thereby eroding the motivation for refreshing the structure of an existing embedded application.

Platform Coupling

Frequently there is no clear separation of operating system and pure application functionality. Legacy code often contains interspersed code sequences that perform input/output (I/O) device control, data formatting, and interrupt handling.

Hardware performance (temporal knowledge) is often encoded into legacy algorithms in the form of time constants. This was done to achieve and maintain performance and computational accuracy as the system matured. Thus changes or updates to the processing hardware have unintended detrimental effects on *unchanged* software. Successful migration and reuse of applications containing such characteristics depends on up-front investigation, identification, and understanding of low-level platform coupling to fully understand top-level design constraints and performance impact aspects.

Structural Degradation

Many currently fielded embedded systems are extensions of custom designs that have evolved during an extended upgrade and maintenance lifetime. Engineering decisions made in the development of these early systems were often specific to the *task at hand*, resulting in system architectures that were not designed for direct reuse. As the product is maintained and upgraded over time, the original architectural design often degrades due to multiple sets of small modifications. Local optimizations are made without adherence to the overall architectural policies that drove the initial design.

As a result, many legacy designs are costly to update because of non-uniformity and brittleness. The structural degradation combined with hidden platform coupling often creates a significant perceived risk in reusing an application, eliminating the consideration of this as an option. The application is perceived to be *spaghetti code* that is overwhelmingly complex and unsuitable for reuse.

Resource Constraints

The limited memory, I/O, and processing capacities of legacy militarized electronics units often drove developers to make design decisions favoring efficiency over quality of design. Often mechanisms peculiar to the inherent programming language were used to provide more efficient but not necessarily extensible designs. As a result of design tradeoffs due to the resource constraints and the extended maintenance/upgrade cycle, the applications often evolve characteristics such as a large number of complex threads of control that cross processing segment boundaries and result in complex segment coupling.

Implicit data dependencies shared across processing segments result in data-driven segment coupling and degradation of individual segment cohesion. This combination of tightly coupled segments of code designed using specific programming language constructs can result in a complex system that is difficult to dissect into reusable and migrational segments. Often such an undertaking requires extensive manual analysis and redesign, thereby increasing update costs.

COTS Exploitation

In contrast to past decades, the defense industry is no longer the driving force behind the development and production of computing electronics and software engineering environments. Although the selection of purely commercial off-the-shelf (COTS)-based architectural approaches has yet to overcome many of the inherent problems facing airborne embedded systems, the industry must now rely on the commercial marketplace for large-scale procurement of select processing architecture elements.

The incorporation of commercially available processing elements promises to provide increased throughput, memory reserves, and I/O bandwidth. However, this advance in technology brings with it potentially greater DMS concerns, as commercial components typically have a two-year refresh cycle, forcing application developers to plan for much shorter hardware lifetimes. In addition, legacy design and development tools are often not available for commercial systems, further stressing the development organization.

Available Knowledge and Experience Base

The knowledge and experience level of a development team is critical to the ability to maintain and upgrade existing embedded systems. The original design criteria for an older system may be missing or inadequately documented. Taking into account that the current development team may have no contact with the original designers, capturing the in-depth knowledge encapsulated in the design becomes critical.

Without this information, developers are often unable to overcome the impacts of structural degradation and evaluate the impact of resource constraints and hardware/software coupling. Migrating entire or selected portions of legacy applications to new platforms or architectures may require so much time to understand the impacts that this approach becomes no longer cost-effective.

Despite these problems, the reuse of legacy software functionality in both fielded and future systems is programmatically attractive. Therefore, we must develop a strategy to overcome the barriers of limited budget, increased integration and production costs, shortened cycle time, COTS insertion, and DMS. The failure to do so may adversely affect the DoD's ability to provide and sustain quality products within available funding profiles and scheduled need dates.

Following is a brief description of the EISR project, its accomplishments, and results from the F-117 flight demonstration.

Technical Approach

The EISR program focused on maturation and integration of two capabilities: structural visualization and construct transformation. Combined, these capabilities provide maximum utility for users interested in wholesale or incremental upgrade. The visualization system provides graphic depictions of data dependency, control flow, and program component interaction. Transformation capability focuses on complete and accurate construct coverage, with certain caveats. For example, it was known in advance that certain JOVIAL constructs had no equivalents in C, and that 100 percent construct coverage was not achievable. However, conversion of even 95 percent of JOVIAL constructs was deemed highly successful as this minimizes the amount of manual reengineering work required to obtain an operational converted application.

The EISR program was aware of several past reengineering environment development efforts that attempted to provide automated restructuring and design aids. One of the problems encountered with earlier efforts was accommodating a variety of target design styles. Different embedded information systems often employ different architectural styles as maintainability requirements vary from application to application. Programming a design environment with expert knowledge of each

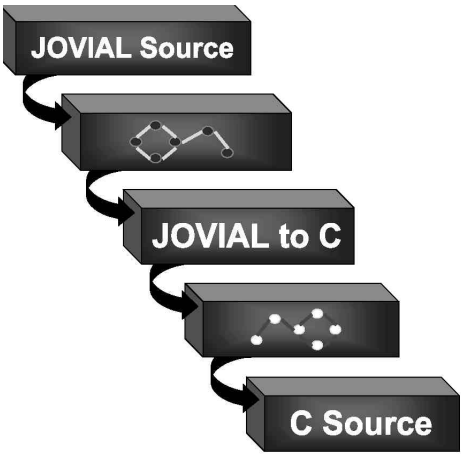


Figure 1: *Transformation Process*

desired target design style is a highly challenging problem. After careful consideration, it was decided to defer investment in automated restructuring and design aids as the combination of visualization and transformation promised the greatest initial return on investment.

Project Results

The EISR technical product is a desktop software-reengineering environment. This system operates on the JOVIAL source files making up an application, parsing and converting them into language-neutral graph-based representations of their operation (see Figure 1). The use of this internal representation yields several benefits. First, since it is language-neutral, this form provides common basic semantics to facilitate integration of *back-end* code generators for a variety of specific target source languages. Second, the form abstracts away language syntax specifics and other peculiarities. Procedural and data elements can then be represented in a common *fundamental* graph-based form that captures inherent interdependencies (see Figure 2).

Figure 2 provides a realistic example of complex data and component dependencies found in typical legacy software artifacts. High legacy software maintenance costs are due in part to the inability of developers to rapidly trace the effects of desired software changes. The EISR graphical analysis suite remedies this by replacing past manual and text-based dependency tracing methods with modern visual tools and search engines. The developers can thus manipulate this graphical representation of the program structure directly, and apply filters and navigational tools to assist in rapid interpretation and restructuring.

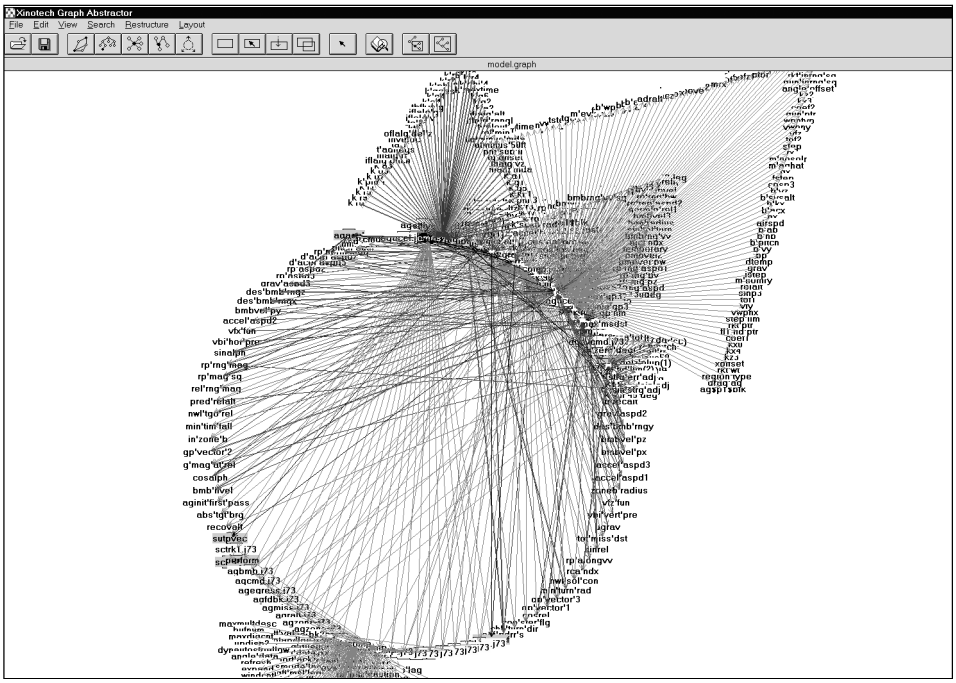
This combination of EISR features gives developers the ability to visualize and trace couplings and structural features of the legacy code. As a result, software engineers now have a capability for understanding and visualizing legacy software artifacts that is on a par with modern graphical CASE tools. A feature of the EISR tool-set is extensibility. The environment is designed around an intermediate representation form that provides a common framework for integrating new *front end* parsers and *back end* target source code generators.

Base Experiments Results

A set of base experiments was conducted under EISR to evaluate the semantic performance and conversion speed of the EISR tool-set using actual DoD application software. The following list summarizes the base experiment results:

- EISR matured capability results in a source code conversion rate of 10K source line of code (SLOC)/minute (PC/NT based).
- Comparative manual conversion rates varied from 20 SLOC/hour to 67 SLOC/hour depending upon experience level of developers and legacy application complexity.
- An initial experiment involving a 4,000 SLOC application required less than 24 hours of *clean-up* touch labor.
- Within the EISR tool-set on initially selected test programs, 100 percent JOVIAL construct coverage was achieved. The following caveats applied:
  - Constant tables became full-fledged structure variables in C.
  - Highly convoluted variable overlays were flagged for manual reengineering since this would be the best overall solution in light of improvements in computing resource availability.
  - Compiler directives were excluded from this statistic since they are not part of the military standard.
- Initial JOVIAL functional structure was retained in the resulting C code with no loss of design partitions or structural understanding.
- Detailed *before and after* code inspections validated that the process of statement-to-statement transformation was l*ossless*.

Figure 2: *EISR Graphical Analysis View (model illustration only)*



Extended Experimentation

The results of the EISR project provided the opportunity for further experimentation and evaluation using the converted artifacts from the original experiments. Engineers were curious to study the ease with which converted JOVIAL artifacts could be migrated to modern commercial object-oriented CASE environments.

In these experiments, a legacy application with a functional structure was converted to C using both manual and automated processes. The resulting code was imported into an object-oriented, Unified Modeling Language (UML)-based commercial CASE tool. The code then underwent a mild restructuring to migrate the original functionally decomposed design to a medium-grained object-oriented structure. The CASE tool was then used to autogenerate C++ source code from the reengineered representation. The results are summarized below:

- Test Case 1: Transforming Functional Decomposition to object-oriented

design (OOD). Consisted of manual JOVIAL-to-C code conversion and manual transformation from functional decomposition into C++ OOD.
- Test Case 2: Transforming from Functional Decomposition to OOD. Consisted of automated JOVIAL to C code conversion using the EISR technology and manual transformation into C++ OOD.

The result of each test case was a legacy JOVIAL-based application transformed into an object-oriented C++ base application (defined using UML notation) targeted for a PC-based processing platform. In comparing Test Case 1 and Test Case 2, we found that the use of the EISR tool-set to capture and transform the JOVIAL to C reduced the level of effort for the overall process (JOVIAL to UML/C++) by approximately 75 percent.

This experimental set was of particular significance because it examined likely and desired future migration goals. We believe that developers will want to move legacy artifacts into modern graphical CASE environments in order to take advantage of automation-assisted testing features, improved software understanding, commercial standard notation benefits, and cost savings due to economies of scale. EISR technology is thus a *key enabler* for full COTS exploitation, as it forms a bridge to modern commercial practices and tool-sets.

Flight Demonstrations in Summer 2001

A *first ever* flight demonstration of reengineered avionics application code took place on July 12, 2001 over Edwards AFB aboard a USAF F-117 Nighthawk stealth fighter. This significant flight demonstration successfully verified correct performance of reengineered components generated by using high degrees of automation assistance. In this demonstration, a small component of the aircraft navigation application was converted from JOVIAL to C++ using the EISR suite and installed in a Power PC-based mission computer prototype.

This application component ran accurately and continuously through a 1.5-hour flight, providing critical data processing in support of the aircraft system navigation solution. This functionality is part of the F-117 precision navigation suite that the pilot relies on extensively throughout the entire mission. Subsequent tests involving delivery of practice and precision munitions were accomplished during the latter part of July 2001.

This demonstration provided a key confidence point, proving operational viability of reengineered application code. This work illustrated an affordable upgrade strategy for the F-117 mission computer using reengineering and computer emulation technology developed at AFRL/IF. EISR thus allows the DoD to affordably recapture previous investments in proven legacy software artifacts and create a migration pathway for exploitation of COTS economies of scale.

EISR Significance

- EISR technology is estimated to significantly reduce design time-span. The technology is mature. Resulting applications have been operationally proven in several flight demonstrations performing realistic missions.
- EISR capability is currently available *off the shelf*.
- The completeness and robustness of the EISR tool-set resulted in a low risk of losing design content (key algorithms, behavior) during the reengineering process.
- EISR technology reduced the effort associated with the coding phase by an order of magnitude when compared to manual transformation. This, in turn, resulted in a 20 percent cost reduction when considering the *overall* software development process (i.e., design, code, test, etc.).
- EISR technology has enabled a paradigm shift and new programmatic process for legacy information system upgrade (see Figure 3).
- Robust, automation-assisted reengineering capabilities enable affordable structural refresh.
- EISR technologies provide a low-risk bridge for migration to modern commercial CASE tools – which in turn enables even greater cost savings potentials.

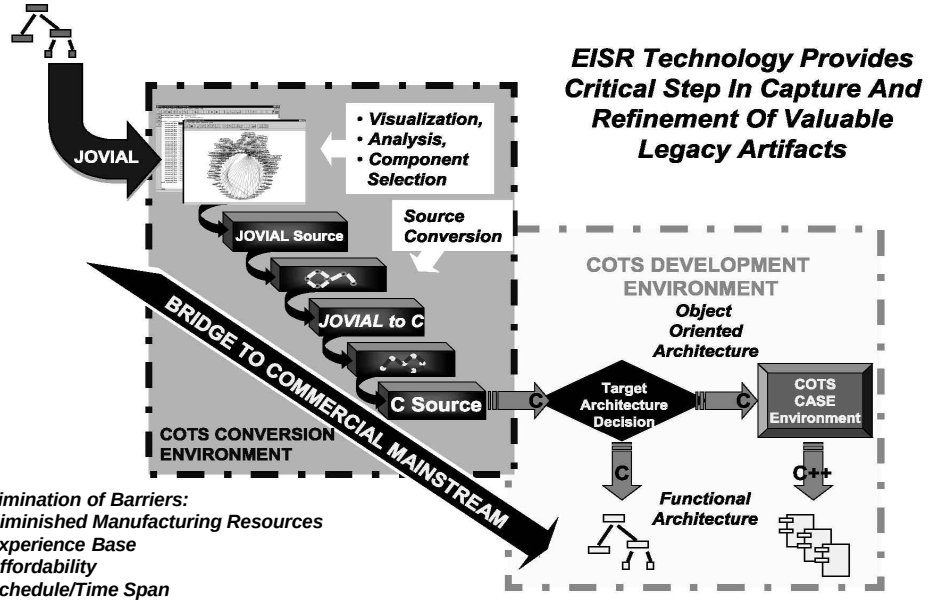
Summary

The EISR program has been successful in putting legacy DoD application software on a convergent path with mainstream software engineering tools and practices. The resulting EISR tool-set has been proven to reliably and wholly capture and transform legacy application content. Legacy software transformation is performed quickly and affordably, drastically improving design cycle times when compared to manual methods. In addition, the resulting artifact forms allow application of a variety of COTS tool-sets for continued development and maintenance. The following list summarizes EISR benefits:

- Modern CASE visualization and analysis capability for legacy designs.
- Minimal introduction of human errors during the process of manual transformation.
- Order of magnitude improvement in SLOC conversion per labor hour expended rate.
- Bridge to mainstream commercial products and practices.
- Maintainability and supportability improvements.
- Affordability increases by leveraging economies of scale.
- Higher availability of skilled developers.

The EISR technology has successfully completed engineering proof-of-concept evaluations. AFRL/IFTA, on behalf of the Computer Resources Support Improvement Program Office (CRSIP), has acquired

Figure 3: *Bridging the Gap*



Coming Events

January 27-31, 2002
2002 Western MultiConference
San Antonio, TX
www.scs.org

February 4-6, 2002
International Conference on COTS-Based Software Systems (ICCBSS)
Orlando, FL
www.iccbss.org

February 11-15, 2002
Application of Software Measurement (ASM 2002)
Anaheim, CA
www.sqe.com/asm

February 25-27, 2002
15th Conference on Software Engineering Education and Training (CSEE & T)
Covington, KY
www.site.uottawa.ca/cseet2002

April 28–May 2, 2002
Software Technology Conference 2002 "Forging the Future of Defense Through Technology"
Salt Lake City, UT
www.stc-online.org

May 13-17, 2002
Software Testing Analysis and Review (STAREAST 2002)
Orlando, FL
www.sqe.com/stareast

June 4-7, 2002
8th IEEE International Symposium Software Metrics (Metrics 2002)
Ottawa, Ontario, Canada
www.software-metrics.org

ownership of the EISR technology, including the source code. In October 2001, the EISR technology was transitioned to the CRSIP Program Office that will provide long-term support and maintenance and will facilitate distribution of the technology. Source code for the EISR tool-set is currently owned by the Air Force. The CRSIP Program Office is located at Ogden Air Logistics Center, Hill Air Force Base, UT. Parties interested in licensing use of the tool can contact Gerald L. White at (801) 775-6713, or via e-mail at <gerald.white @hill.af.mil>.◆

About the Authors

Kenneth Littlejohn is a project engineer in the Embedded Information Systems Engineering Branch, Information Directorate, Air Force Research Laboratory. He has more than 19 years research experience related to affordable design, development, and support of real-time embedded software for Air Force weapon systems. Littlejohn currently serves as the project engineer for the Embedded Information System Reengineering project. He earned a bachelor's degree in electrical engineering in 1987, and a master's degree in computer science in 1994, both from the University of Dayton.

AFRL/IFTA
2241 Avionics Circle
WPAFB, OH 45433-7334
Phone: (937) 255-6548 ext. 3587
Fax: (937) 656-4277
E-mail: kenneth.littlejohn@wpafb.af.mil

Michael V. DelPrincipe has more than 15 years experience in the design, development, and test of embedded real-time avionics fire control, weapons management, and display software applications for airborne weapon systems. He is currently responsible for the technical and programmatic management of multiple Air Force Research Laboratory-sponsored research projects related to legacy embedded information system modernization. DelPrincipe earned a bachelor's of science degree in computer science with honors from the State University of New York, Brockport campus.

Lockheed Martin Aeronautics Company
P.O. Box 746
Fort Worth, TX 76101 MZ 6295
Phone: (817) 777-3667
Fax: (817) 777-3121
E-mail: michael.v.delprincipe@lmco.com

Jonathan D. Preston is a technology program manager within the Advanced Development Programs branch of Lockheed Martin Aeronautics Company. Throughout his career, he has managed several government-funded technology programs that have transferred technologies to major weapon system programs. Preston earned a bachelor's degree in electrical engineering from the Pennsylvania State University and an master's degree in computer science engineering from the University of Texas at Arlington.

Lockheed Martin Aeronautics Company
P.O. Box 746
Fort Worth, TX 76101 MZ 2411
Phone: (817) 763-2740
Fax: (817) 763-2967
E-mail: jonathan.d.preston@lmco.com

Ben A. Calloni, Ph.D., is a research program manager for multiple software research and development efforts at Lockheed Martin Aeronautics Company, Fort Worth, Texas. He is leading the investigation into commercial off-the-shelf solutions for legacy avionics software. Dr. Calloni earned a bachelor's of science degree in industrial engineering from Purdue University, and master's and doctorate degrees in computer science from Texas Tech University. He is a licensed professional software engineer in Texas.

Lockheed Martin Aeronautics Company
P.O. Box 746
Fort Worth, TX 76101 MZ 2859
Phone: (817) 777-4345
Fax: (817) 763-2967
E-mail: ben.a.calloni@lmco.com