SOFTWARE PRODUCT LINE TECHNOLOGIES FOR MILITARY SYSTEMS

For The

SOFTWARE DESIGN AND PRODUCTIVITY COORDINATING GROUP

WORKSHOP ON NEW VISIONS FOR SOFTWARE DESIGN AND PRODUCTIVITY: RESEARCH AND APPLICATIONS

Abstract:

In 1995, an initiative was launched at Boeing (then McDonnell Douglas) to assess the potential for reuse of operational flight program (OFP) software across multiple fighter aircraft platforms, and to define and demonstrate a supporting system architecture based upon open commercial hardware, software, standards and practices [1]. The following year, this became a key element of the Bold Stroke Open System Architecture avionics affordability initiative which applied these techniques to the broader tactical aircraft mission processing domain. The Bold Stroke architecture, application components, middleware framework, and development processes have been leveraged on an increasing number of military platforms of national importance including manned and unmanned vehicles for the USAF and USN.

Our experiences on this effort have demonstrated the dramatic increases in software development productivity possible through use of cross-platform reuse and highly portable and standardized run-time subsystems. They have also highlighted weaknesses in several areas which significantly impact overall system cost, quality, and timeliness, many of which are unique to large-scale distributed real-time embedded (DRE) systems. Embedded system product lines depend on suitable domain specific architectures, run-time frameworks, application component libraries, and component development and integration tools and processes, and stress these capabilities beyond what would be sufficient for single system development approaches. In weapon systems, key remaining technology hurdles include (1) integration of thousands of software components with real-time deadlines; (2) integration of hard and soft real-time components; (3) enlargement in scope of quality of service resource management services from single processors to widely networked systems; (4) establishment of safety critical and mixed criticality run-time frameworks; (5) interoperability with closed legacy systems; (6) retention of high security levels while opening up strike assets to greater and greater levels of tactical network connectivity; and (7) infusing new technologies into fielded systems. In addition to these “design-centric” challenges, additional important challenges are contained at the front and back ends (i.e. requirements definition and verification) of the embedded system development process.

31 October 2001

Mr. David Sharp
The Boeing Company
PO Box 516
St. Louis, MO 63166
Phone: (314) 233-5628
Fax: (314) 233-8323
Email: david.sharp@boeing.com

Mr. Don C. Winter
The Boeing Company
PO Box 516
St. Louis, MO 63166
Phone: (314) 233-2743
Fax: (314) 233-8323
Email: don.c.winter@boeing.com

1 Introduction

Development of reusable large-scale distributed real-time embedded systems requires a vast array of technologies. To efficiently and effectively establish the technology foundation necessary for this approach requires careful assessment of suitable commercial technologies and development of domain and architecture specific technologies.

To maximize returns on investment, our overall approach has been to (1) influence COTS where possible; (2) leverage and adapt COTS where applicable; and (3) fill in technology voids where necessary. Each of these thrusts requires careful consideration of existing practices and needs on military system programs, broad understanding of available commercial technologies, and vision into technology advances which could improve existing development and operational capabilities.

This paper is organized to describe each of these thrusts. Section 2 describes the potential benefits of embedded system product lines. Subsequent sections describes our use, influence, and adaptation of commercial technologies, and some of our efforts at filling in commercial technology gaps.

2 Benefits of Embedded System Product Lines

While novel in 1996, and still unusual at the size of systems Bold Stroke targets, the use of product lines is increasingly popular in embedded software systems. The primary benefits to be gained by a shared software architecture are shared or reused products, and the associated increases in affordability and quality that result. The degree to which the members of the product line are common dictates the level of reuse that can be gained. If the members share only a common domain without common requirements, very little application reuse is possible, and the underlying architecture and supporting infrastructure may represent the bulk of potential reuse.

Where common requirements are available or can be created, then the members can begin to share them across the product line. Newly developed requirements can be built around the existing shared requirements, which may enable them to be used in the future by other members for whom they were not originally created. Development and configuration of common and product specific requirements is a challenging area.

Common detailed functional requirements, however, are not necessary for a common software architecture to be developed. As long as the types of requirements that act upon the architecture are shared, an architecture can be developed to address these requirements. Once this software architecture is in place, it can be implemented using common middleware (or infrastructure) elements. Once a common middleware is in place, application components can be developed that can be easily reused across the product line. In a broader sense, reuse of run-time frameworks across products and domains is an example of this.

Beyond direct reuse of software and its related products, there are several additional benefits that can be gained by having a common software architecture. Most of these benefits are less easily quantified, but revolve around communication between the development organizations of the product line. This enables developers to leverage their experiences on one product when they move to another, and results in more flexible development organizations.

3 Use of Commercial Technologies

Many critical areas for DRE systems have historically been custom crafted for each military system which needed them. Processors, networks, languages, protocols, run-time systems, even assemblers, compilers, emulators, and other development tools were project specific. The rapid advance of commercial technology alleviates the need for custom approaches, while reducing military budgets cannot sustain redundant development of highly reusable capabilities. In an era where commercial investments greatly exceed military ones, military technologies *must* focus on areas which do not compete with commercial technologies. Barring this, military investments are wasted, contractors work with substandard design environments and run-time systems, and adversaries leverage superior commercial development capabilities. Compounding these concern are academic and personnel issues: students are not trained to work with military specific technologies, and engineers do not desire to use them. In addition to these forces which motivate their use, commercial advances in these listed technologies are sufficiently general and capable that they meet many DoD system needs.

4 Influencing Commercial Technologies

Given the relative small percentage of the market represented by military systems, the influence possible by military system developers is diminishing. Our experience has been that influence is greatest during the earliest phases of technology and specification development. In collaboration with Washington University in St. Louis, we engaged the RT CORBA community during specification development and significantly influenced it in directions important for military embedded systems. Once large numbers of specifications and/or implementations are available for a particular technology, the opportunity to influence the marketplace significantly declines. In alignment with this approach, as part of the AFRL Real-Time Java for Embedded Systems program we are a member of the specification committee for Distributed Real-Time Java. It is critical that military stakeholders engage in these specification efforts to influence the industrial base when inputs are most highly leveraged. Commercial specifications represent a culminating transition opportunity, and potentially exit criteria, for research investments.

5 Adapting Commercial Technologies

Even with the best and most relevant commercial technology, additional effort will be required to adapt it for military systems. Commercial systems, especially in the embedded systems arena, typically do not reach the scale of military systems. Avionics systems typically exceed 1M lines of source code, span 10-100 processors, and require strict adherence to timing requirements of 10 – 100 Hz periodic rates. Other military systems greatly exceed these processing rates. These requirements stress the capabilities of commercial systems, and frequently require optimization to be met. On the development side, large teams of engineers, many of which are frequently accustomed to decade-old software development environments, must collaborate in system development. Development teams are frequently split between sites and companies. All of these contextual forces require adaptation of both run-time and design-time technologies available from industry. The adaptation of commercial technologies itself represents a critical technology area for DoD interest. The DARPA Program Composition for Embedded Systems is a good example, where commercial aspect oriented software approaches are being adapted and refined for military embedded systems.

The rapid advance in commercial technologies is well known and represents a two-edged sword. On

the one hand, we seek to exploit rapid escalation of capabilities, and on the other we seek a stable development environment. Process and architectural characteristics must be carefully crafted to isolate application developers from the variabilities of low level system elements including hardware, networks, and operating systems. For instance, some of the architectural attributes applied on Bold Stroke represent commercial technology adaptation techniques. For instance, the use of layered architectures allowed the use of many commercial technologies while containing the impact when they change. Specific examples include:

1. Layered middleware and operating system infrastructures allowed migration to faster hardware platforms as them became available. Whereas legacy systems frequently required locking in hardware early in the program which became obsolete before systems were fielded, the layered approach to COTS hardware usage allows fielded systems to use the latest "just-in-time" hardware technology available.
2. Operating System wrappers allowed the execution of flight software application components on commercial PC platforms (Pentium and Windows) for non-real-time functional debugging. This greatly speeded application development by providing a dedicated testing resource to each developer and allowing the use of leading commercial tools such as Rational Purify, Quantify, and Rose, and Microsoft Visual C++ within an environment that also strongly supported organizational tasks such as email and document preparation.
3. The Operating System wrappers also provided high levels of portability to new operating systems which were not part of the original development goals.
4. RT CORBA middleware approaches supported changing network protocols without impact on application developers. During initial system test the system uses TCP/IP on Windows platforms. For the flight system, VME and Fibre Channel are used. In addition, CORBA supported highly configurable distribution approaches—especially important in our product line with greatly variable hardware and networking configurations.
5. Domain specific services developed specifically for Bold Stroke isolated application developers from CORBA specifics and supported portability between different ORBs and alleviated the need for large numbers of application developers to become intimately versed in associated technologies.

Given the longevity of military systems, completely new military DRE systems are rare. While the aforementioned adaptations were described within the context of a new system design, many opportunities exist for incrementally infusing commercial technologies into existing systems. In addition, even new systems must typically interoperate with a wide range of existing legacy systems and protocols. Working with legacy systems is especially difficult, but it represents the majority of technology transition opportunities. We have been working on associated techniques for several years as part of the AFRL Incremental Upgrade of Legacy Systems program.

6 Filling In Commercial Technology Voids

There are several categories of commercial technology voids. Within a product line paradigm, they can be split into suitable domain specific architectures, run-time frameworks, application component libraries,

and component development and integration tools and processes. While architectures, frameworks, and development environments are often perceived as enabling technologies, application component libraries represent the core of many businesses. Given their highly product specific nature, application components are typically best left to development organizations, while research advances in the enabling technologies are well-suited to DoD sponsored research. Important new classes of systems are emerging as those associated with the DARPA Networked Embedded Software Technology program which require establishment of new domain specific reference architectures. Subsequent sections describe in more detail challenges in the areas of run-time frameworks and development tools and environments.

6.1 Run-Time Frameworks

Establishment of standard run-time frameworks represents a key opportunity for cross-military platform leverage, and one which has been largely untapped. Operating systems, distributed object component middleware, schedulers, fault tolerance services, and many other capabilities are broadly applicable across systems. While projects such as Bold Stroke have demonstrated the applicability of COTS-based run-time frameworks, many limitations remain. DARPA and AFRL sponsored programs have begun to extend capabilities with the integration of hard and soft real-time components and enlarging of scheduling and resource management services beyond a single processor. One of the complexities associated with product line approaches is a potentially large run-time system configuration task. Increasing levels of self and automated adaptation promises to significantly ease this challenge. Integration of highly deterministic and verifiable run-time systems (e.g. TTA) with capabilities necessary for more dynamic systems (e.g. adaptivity) poses strong challenges. To address vulnerabilities associated with network centric warfare, network security and multi-level security requirements must be addressed as well.

6.2 Development Tools and Environments

Development tools and environments are highly reusable across multiple programs and contractors and yet must address unique military DRE system requirements, making them ideally suited to DoD investment. Important military DRE system failures are well-known and impact the operational effectiveness of warfighters. Many types of analyses are available in theoretical contexts but fail to meet scalability and other real-world demands which have greatly inhibited their use on production systems. To address the real needs of component-based product line system developers, tools must address integration of thousands of software components with widely variable and modal real-time deadlines. Components must be highly configurable to support variation between products. Tools must support modeling of component configurations, automate many scheduling and allocation decisions currently handled by highly skilled system experts, employ standard interfaces to support tool integration, and be themselves highly configurable to domain specific uses. In addition to software development issues such as these, embedded system tools must fundamentally address the *physical* nature of embedded systems: their interaction with the real-world and their quality attributes associated with issues such as timeliness, safety, reliability and fault tolerance, and distribution. These are key issues that distinguish the needs of embedded military system developers from those in mainstream commercial industry. Many of these needs are being targeted by the DARPA Model-Based Integration of Embedded Systems program.

7 Conclusion

Military DRE systems pose a large number of challenges and opportunities. Their complexity requires careful application of a wide range of technologies, and invention of many more. Their cost and importance to both our economic standing and national security necessitates a concerted research effort towards their resolution. While each individual technology described herein represents an important capability, it must be recognized that their integration into an end system represents a culminating and daunting task. The system integrator's task is not to independently infuse technologies, but to combine them in operationally significant ways to transcend the capabilities supported by any single technology while balancing the forces and tensions presented by each. Only by furthering capabilities in individual transitionable technologies within the context of production system requirements and technologies can real-world DRE system development capabilities be incrementally advanced and the challenge of emerging military DRE systems be met.