Integrating the Operational Architecture,
Position Paper for WESAS 2000

Although the past two decades have seen great advances in computer architectures, few large-scale systems now being deployed and used actually reflect this advancement. Software development methodology, tools, processes have matured, with positive effects on the quality of large software systems; yet, the incorporation of new architecture patterns, commercial-off-the-shelf building blocks is rare. This is especially true for mission-critical or safety-critical command-and-control systems. The current generation critical systems now in deployment builds upon the architecture styles, components, and techniques of a decade or more ago. They do much low-level, local software reuse, modularity and structure, they depend upon modern hardware and network platforms, and they are built to exacting standards. Nevertheless, the application software tends to be completely handcrafted and uniquely designed, with few COTS software subsystems, components, no hint of middleware, and few hints of other architecture design technology or architectural standards.

Looking ahead to the next generation of these systems which are now in development, they certainly attempt advancement, but they still lag behind present architectures by up to a decade in the technology they will introduce. They will enter production in three to five years. This lag is not merely a result of conservatism, dinosaur blundering of huge corporations and government agencies, bureaucracy, or even the appropriate caution stemming from mission-critical and safety-critical nature of the systems. The growth and software technologies follows a kind of "Gresham's Law" (good technology drives bad technology out of circulation) that results in the big, slow, and unmaintainable simply being bypassed. This has not happened.

The following discussion employs the idea that a complete system architecture description provides information relative to three perspective views, term, following the DoD's C4ISR Architecture Framework Version 2.0: Operational Architecture, the Technical Architecture, and the System Architecture. In terms of these concepts, I take the position that:

1) Retarding and defeating factors at work in the large-scale system application domains which I believe relate to the Operational aspect of system architecture persist and have an increasing effect as systems get larger and more complex.
2) To reduce the effects of these factors, it is necessary to focus on the Operational Architecture. This entails developing a better understanding of how the system is employed operationally in various domains.
application domain. It means development of tools and techniques to
and evaluate the relationships between the operational knowledge and
functions, capabilities, features, and requirements of the system. Operational Architecture to drive the Technical Architecture without
restraints now encountered.

The remainder of this paper presents examples to support the position
example of a current generation system gives a picture of the domain
state of the art within it. The second exemplifies a next generation
now in design, and some of the problems being faced in attempting to
architecture-based designs.

The Raytheon C3I Systems unit in Fullerton CA (formerly Hughes Aircraft
Company, Command and Control Systems Division) develops medium to lar
tscale automation systems for air traffic management, satellite navig
ttraffic surveillance, air defense systems, strategic planning systems;
operations control communications computer and intelligence systems. Over the past 15 years the architectures for these systems
evolved remarkably. At the beginning of the period the architecture
characterized as monolithic, specialized, flexible and extensible on:
severely prescribed bounds, and implemented on unique, non-commercial:
militarized hardware. This past generation architecture has been re:
by systems developed wholly on commercial-off-the-shelf hardware and
software, in a commercial communications matrix, using commercial
development languages and tools.

The current generation of Raytheon architectures, now in production :
developed in C or C++, is component based, is highly adaptable and e:
and has undergone some form of productization. Productization in th:
means a system, originally one-of-a-kind, that has been purposefully
engineered so that it can be much more easily reproduced, enhanced,
customized, and adapted for many new applications and customers.

For example, the TracView ™ Air Traffic Management (ATM) system is b:
a system originally built in the early 1980's and written in the JOV:
for Hughes built computers. For TracView it was translated to C, por:
class hardware (Intel), marketed and produced as a low-cost, low-end
capability. It has since been ported to high-end Unix workstations
enhanced, restructured, and re-architected. It is deployed in over 4
with over 140 installations now in operation. Its uses range from p:
single-workstation, airport surveillance systems for technologically
ations, to being a key subsystem within massive systems like the FAA
system, now entering deployment.

While revolutionary at its inception in the late 1980s, specifically
language, X Windows, COTS hardware and OS, network distribution, and
modular structure, this type of architecture is now typical at Raytheon:
good system, but note, the reuse is purely local; its adaptability a:
are specific to a very narrow domain, itself alone. Moreover, as an
reusable software product, TracView is one of the few to succeed, at and elsewhere, against a background of many unsuccessful or much less successful attempts. One of the known reasons for the success is the simplicity of its operational concept, the caution with which new features technologies are added. A close working relationship developed between Operations and Technical Engineers. Through much of TracView's product history it was employed as an emergency replacement; which forced close problem-solving engagement between the end users, operations engineers, system engineers, and software engineers. The Technical engineering were usually involved from project inception to start of live operations often as short as 90 days.

The generation of Raytheon architectures that is now in the design process much closer to the present state of the art. For example, the Common C4I system attempts to incorporate an extensive set of internet-based technologies: multiple distributed client-server database systems, web and browsers, standard office automation COTS integrated through mid-office work flow and collaborative decision making. In theory development work is to be done involves very little traditional coding: table design, form design, configuring and adapting COTS software and hardware, scripting, and construction of small VB and Java applications. Development team seeks to repeat much of the TracView experience. They started with small, simple systems that fit the immediate needs of a group of users; and work with those users to make the technology, the architecture, conform to the operational circumstances as the users experience it.

Unfortunately, at this moment the wide range of possibilities, architectural patterns, competing technologies, COTS components and frameworks make the decision process difficult. The techniques and technology, and their capacity for adaptability and customization do satisfy the operational technical requirements, the resulting assemblage frequently does not provide operationally suitable systems. The effort, and cost, to resolve each issue soon becomes the program driver.

A second, frequently arising problem is that the COTS frameworks and components are usually developed for a different application domain. Significant amount of development effort is devoted to bending the COTS to the new domain to deal with fitness and suitability issues which arise or after system integration, only after the new system has revealed processes and assumptions of which no-one had been aware previously.

Raytheon's experience in system development of large-scale, domain specific command and control type systems repeatedly suggests that more attention to operational analysis is needed. Effective use of the proliferation of architectures, frameworks, and technology for adaptation and customization of standard parts requires development of better means for analysis evaluation of the effects of operational roles, and interactions on the technical