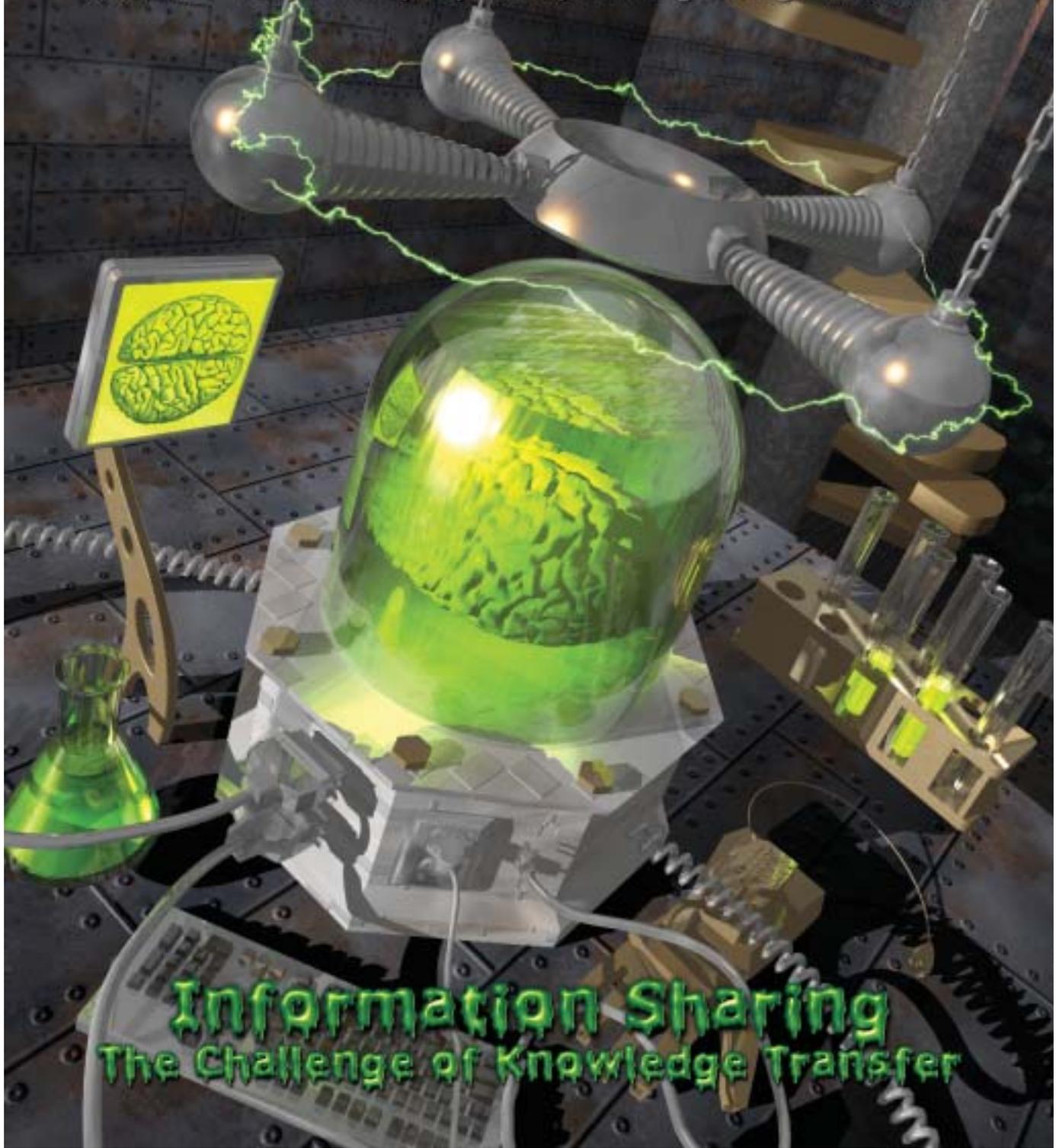


# CROSSTALK

October 2003 *The Journal of Defense Software Engineering* Vol. 16 No. 10



## Information Sharing The Challenge of Knowledge Transfer

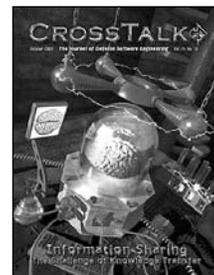
## Information Sharing

**4 An Information Architecture Strategy**  
Learn how to design an information architecture that provides the right information, in the right format, to the right person, at the right time, all through a secure access.  
*by John Wunder and Dr. Will Tracz*

**8 Warfighter's Access to Geospatial Intelligence**  
The National System for Geospatial Intelligence has met the challenge to integrate disparate information sources and provide a common user view into imagery data.  
*by Peter Winter*

**12 Effective Collaboration: People Augmented by Technology**  
This article uses a real-world example of collaboration between people and smart devices in a teaching environment that is adaptable to the global community.  
*by Richard L. Conn*

**16 Serialized Maintenance Data Collection Using DRILS**  
The Defense Repair Information Logistics System is a specialized Web-based application to collect, store, and retrieve Air Force depot and field maintenance data that especially benefits data entry personnel.  
*by Capt. Greg Lindsey and Kevin Berk*



**ON THE COVER**

Cover Design by  
Kent Bingham.

## Best Practices

**21 The Documentation Diet**  
This article discusses strategies you can use to make documentation work for you, including examples of project and process documentation.  
*by Neil Potter and Mary Sakry*

## Software Engineering Technology

**25 Software Architecture as a Combination of Patterns**  
These authors discuss how they used four design patterns to construct a radar system that can withstand the replacement of hardware and operating system software, and is adaptable to different requirements.  
*by Kent Petersson, Tobias Persson, and Dr. Bo I. Sanden*

## Online Articles

**29 Introducing Global Software Competitiveness**  
*by Don O'Neill*

**Data Warehouse: Your Gateway to the Information Age**  
*by Kelly L. Smith*

## Departments

- 3 From the Publisher**
- 11 Top 5 Award Nomination Information**
- 20 Coming Events Web Sites**
- 30 Letter to the Editor**
- 31 BACKTALK**

## CROSSTALK

**SPONSOR** Lt. Col. Glenn A. Palmer  
**PUBLISHER** Tracy Stauder  
**ASSOCIATE PUBLISHER** Elizabeth Starrett  
**MANAGING EDITOR** Pamela S. Bowers  
**ASSOCIATE EDITOR** Chelene Fortier-Lozanchich  
**ARTICLE COORDINATOR** Nicole Kentta  
**CREATIVE SERVICES COORDINATOR** Janna Kay Jensen  
**PHONE** (801) 586-0095  
**FAX** (801) 777-8069  
**E-MAIL** crosstalk.staff@hill.af.mil  
**CROSSTALK ONLINE** www.stsc.hill.af.mil/crosstalk  
**CRSIP ONLINE** www.crsip.hill.af.mil

**Subscriptions:** Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 24.

Ogden ALC/MASE  
6022 Fir Ave.  
Bldg. 1238  
Hill AFB, UT 84056-5820

**Article Submissions:** We welcome articles of interest to the defense software community. Articles must be approved by the CROSS TALK editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CROSS TALK does not pay for submissions. Articles published in CROSS TALK remain the property of the authors and may be submitted to other publications.

**Reprints and Permissions:** Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CROSS TALK.

**Trademarks and Endorsements:** This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSS TALK are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

**Coming Events:** We often list conferences, seminars, symposiums, etc. that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CROSS TALK Editorial Department.

**STSC Online Services:** www.stsc.hill.af.mil  
Call (801) 777-7026, e-mail: randy.schreifels@hill.af.mil

**Back Issues Available:** The STSC sometimes has extra copies of back issues of CROSS TALK available free of charge.

**The Software Technology Support Center** was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.



## Developers Meet a Variety of Complex Information and Data Sharing Needs



This month's CROSSTALK theme is centered on information sharing and data management. The relationship between data contained in a database and informational needs for today's complex computer-driven systems is the subject of our lead article from John Wunder and Dr. Will Tracz. In *An Information Architecture Strategy*, these authors explain how information architecture is a hierarchy of products and services organized to assure the timely and accurate delivery and storage of data and information across an entire enterprise. The article describes the components of an information architecture strategy using examples from the Global Combat Support System-Air Force's Enterprise Architecture, currently under development.

A system that is used heavily in the war against terrorism is described in the article *Warfighter's Access to Geospatial Intelligence*. Author Peter Winter explains the use of the Information Access Services tool developed by the Harris Corporation to provide timely, relevant, and accurate imagery, imagery intelligence, and geospatial information to our warfighters. Although the National Imagery and Mapping Agency has terabytes of imagery and intelligence information available, getting the right data, at the right time, to the right user is the challenge described in this article.

In his article *Effective Collaboration: People Augmented by Technology*, Richard L. Conn delves into the world of collaboration for development in today's world of increasingly complex software-intensive, mission-critical systems. He mentions how the development of the Joint Strike Fighter F-35 uses a Web-based information portal server as an example requiring collaboration of major contractors with subcontractors distributed across several countries. He goes into explicit detail with another example using classroom education of 100 students from 24 different countries to describe some of the current capabilities available for collaboration.

The article *Serialized Maintenance Data Collection Using DRILS* by Capt. Greg Lindsey and Kevin Berk describes a specialized Web-based application that has been successfully used to dramatically reduce the sustainment costs of the Air Force F-16 fighter. They explain that the key to successful cost reduction sustainment programs is to have a user-friendly, data-gathering capability that provides the supply chain manager with dependable part consumption data, as well as the information needed to identify and correct troublesome components in any complex weapon system.

Have you ever worked on a project where the document requirements did not make sense and seemed to be irrelevant, non-productive paper stacks? In *The Documentation Diet*, authors Neil Potter and Mary Sakry give an analysis of how inappropriate documentation can creep into and strangle a project. They elaborate on a list of several viable techniques for making your documentation strictly practical. If you hate project and process documentation, then read this and review how to put your documentation requirements on a diet.

In our supporting article, *Software Architecture as a Combination of Patterns*, authors Kent Petersson, Tobias Persson, and Dr. Bo I. Sanden recount their efforts in the creation of a unique software architecture to solve the requirement for a highly adaptable radar system. The radar had to handle the needs of different customer requirements in an environment of hardware and operating system replacement.

Lastly we bring you two online articles this month. Author Don O'Neill outlines a vision to achieve global competitiveness in *Introducing Global Software Competitiveness*. He explains that software competitiveness revolves around how the software work force is used to achieve customer satisfaction, how innovation is essential to delivering customer value, and how strategic software management guards against event threats. In *Data Warehouse: Your Gateway to the Information Age*, Kelly L. Smith gives new hope to those searching for a simple data storage solution, and describes the challenge of integrating today's data storage systems with those from the '70s and '80s.

We hope this selection of articles provides you with a new perspective on the management of increasing amounts of data in today's complex computer-driven enterprises. Our goal is that perhaps one or two good ideas on how to solve a current data dilemma are found in this issue.

H. Bruce Allgood  
Director, Computer Resources Support Improvement Program



# An Information Architecture Strategy

John Wunder and Dr. Will Tracz  
Lockheed Martin Mission Systems

*Information architectures play a strategic role in facilitating the efficient and effective storage, retrieval, and analysis of data in enterprise information systems. An information architecture consists of more than just data and the commercial off-the-shelf products in which it is stored. To provide the right information, in the right format, to the right person, at the right time, and to protect that information from unauthorized access, a robust information architecture strategy is necessary. This is also essential to establish data stewards, governance boards, and metrics to support the processes that describe sequences of information services that access the stored data.*

This article describes an information architecture strategy<sup>1</sup>, which by definition focuses on information, not data. Therefore, the scope of the architecture strategy extends beyond the physical architecture consisting of data, databases, data warehouses, and data marts, to the operational architecture focused on the processes that turn the data into information and facilitate the generation and manipulation of that information. The overarching goal of this strategy is to provide the end user with access to timely, accurate, and trusted information.

In the case of the Global Combat Support System-Air Force (GCSS-AF), the information architecture provides the tools to turn data from a myriad of systems into information, from information into knowledge, and from knowledge into power. This results in information superiority and reduced decision cycles across all Air Force echelons and operational theaters.

An information architecture provides easy storage, access, and retrieval of infor-

mation. For example, it enables authorized airmen to access, from any location, such diverse data as the budget information for Wright-Patterson Air Force Base, the engine status at Lakenheath Air Base, or all the personnel deployed in an aerospace expeditionary force, all through a Web browser connected to the Department of Defense infrastructure.

This article first describes the key terms *process* and *information services* and then defines the kinds of data that are stored in the various components that make up the information architecture. Next is an overview of components found in information architecture with special focus on the repository and integration services. Finally, the roles and responsibilities of those who govern the information architecture are described.

## Terminology

Operationally, from an information technology (IT) perspective, an information architecture supports *processes* that describe

sequences of *information services* that access *stored data*. This relationship is illustrated in Figure 1.

Processes are sequences of operations. For example, at a data/information level, processes might include Find, Fix, Target, Engage, and Assess. At the decision-support level, processes might include Readiness, Crisis Action Planning, Deployment, and Employment/Sustainment. Processes invoke information services in a specific order (i.e., in a sequence that becomes an operational flow).

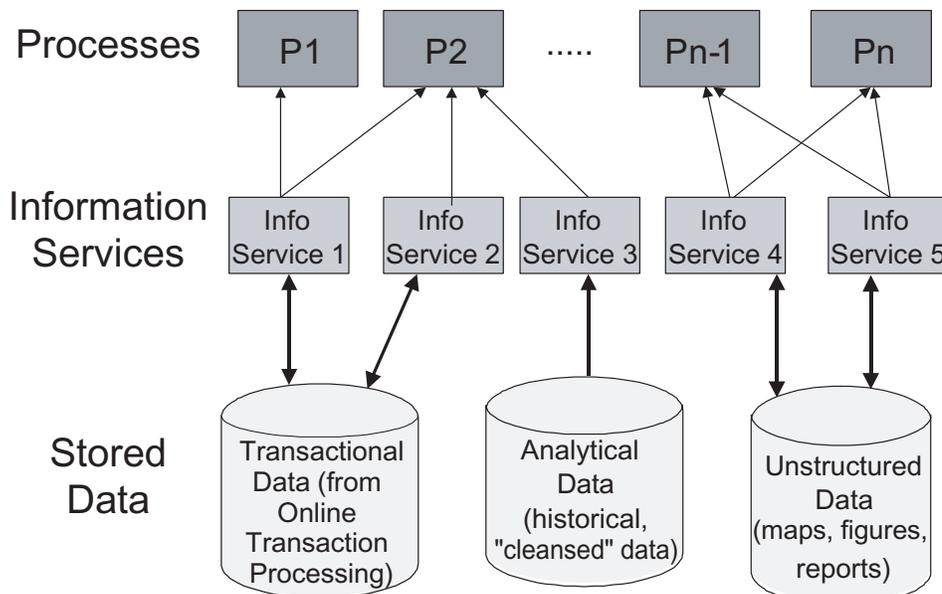
Information services provide intuitive access to information within the information architecture such as domain-relevant information on such things as aircraft, munitions, personnel, or bases. Information services also provide information on domain-specific capabilities such as sortie generation capacity, fuel consumption, skill training and certifications, or runway capacities. Information services apply domain rules, semantics, and syntax to convert raw data into information.

Two information services examples follow. The first illustrates the difference between raw *data* and decision-support *information* provided through an information service. The second highlights the data cleansing or scrubbing capabilities of the information architecture.

### Example 1: Munitions

In a combat situation, many users (e.g., from the joint commander, to the unit level commander, to the munitions supply clerk) need to know the *capacity* of an ammo dump. However, for safety reasons munitions are not active in the ammo dump but are built when needed. If direct access to munitions *data* were granted to end users, then the lack of munitions domain knowledge could lead to confusion rather than useful, decisional knowledge. Therefore, munitions information service could provide accurate *information* to authorized users

Figure 1: Relationship Between Processes, Information Services, and Stored Data



by applying the appropriate domain rules and semantics associated with the munitions data resulting, for example, in the required information about the munitions *capacity*.

### Example 2: General Ledger

All program managers need access to current expenditures. Accounting data is generated by many sources, potentially in many different formats. A general ledger information service assures that only complete, accurate, and audited expenditures of every budget line item are available to the user by eliminating conflicting or incomplete data (i.e., via data cleansing or scrubbing technology).

Information services accesses *stored data* to generate information. Generally, there are three types of data stored in information architecture. These types of data, as illustrated using the previous example in the general ledger information service, are as follows:

- **Transactional data.** These are day-to-day operations recorded as they occur. In the general ledger example, this is the recording of the thousands of day-to-day expenditures required to run an enterprise.
- **Analytical data.** These are historic transactional data that may be time-stamped and stored immediately after an event occurs, or are recorded later. Furthermore, the format of the data may be different than its originally recorded format. Analytical data are used to determine trends and make predictions. In the general ledger example, the previous year's expenditures would need to be recorded for later analysis to determine future budget requirements.
- **Unstructured information.** These are stand-alone documents that provide direct guidance/information for a task. These data include maps, weather reports, pictures or, for the general ledger example, expenditure guidance from Congress or the Department of Defense.

## Information Architecture in Context

Advances in IT in the areas of workflow and design notation/specification standards (e.g., Business Process Modeling Language and Unified Modeling Language) have reached the point where it is possible for mission processes to be composed, automated, and executed by end users without requiring IT professionals or traditional application development. In addition, IT Web services<sup>2</sup> [1] have simplified the creation, invocation, and aggregation

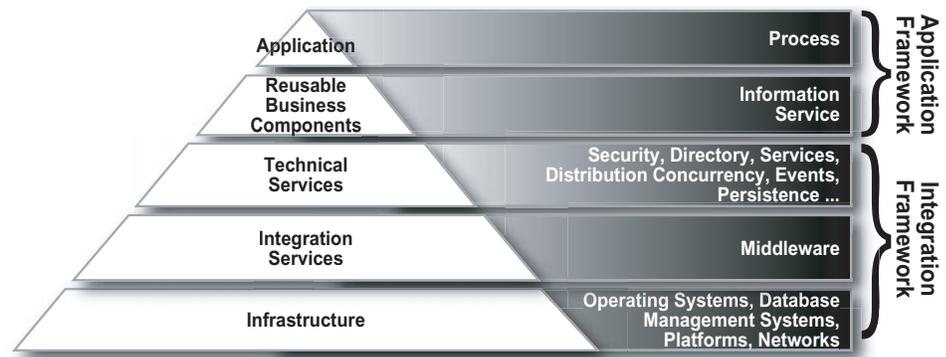


Figure 2: GCSS-AF Architecture, Frameworks (2), Layers (5), and Example Technologies

of domain-specific information services (e.g., aircraft, munitions, inventory, or budget).

In the Air Force, these advances allow warfighters to sequence and aggregate processes and information services to resolve unique, unanticipated situations as they occur. For example, the Regional Support Squadron (RSS) at Air Mobility Command may determine that they are having an unacceptable number of tire failures; however, no application was written to determine whether the tire failure was due to a particular lot of tires, type of aircraft, or runway. The information architecture allows the RSS member to easily use the Supplier, Aircraft, and Runway and Tire Failure Information Services to isolate the root cause of the failures and execute appropriate corrective actions without any new system or code development.

Finally, current IT standards and mechanisms for representing data patterns allow a clear separation between information creation and usage and data storage and retrieval. This allows implementers of core information services to use common, centrally administered and managed commercial tools to store and retrieve the data. These are the cornerstone concepts of information architecture strategy and enable the delivery of accurate, timely, and trusted information to the warfighter.

Information architecture is part of an enterprise information system's physical architecture. For example as shown in Figure 2, the GCSS-AF architecture consists of two frameworks comprising five layers. While Application Framework contains Air Force Information Services, the Integration Framework consists of information technology (commercial products) that enables integration and supports the capabilities in the Application Framework. Thus, in this case, the focus of the information architecture is primarily on the Application Framework. Processes form the top layer of the Application Framework and invoke information services that reside in the second layer. The data

store services are found in the Integration Framework (i.e., in the data warehouse or operational databases).

### Key Building Blocks

As shown in Figure 3 (see page 6), information architecture consists of the following components:

- A technology infrastructure that is comprised of the following:
  - Processes targeted to support the enterprise business processes (or in the case of the Air Force, Air Force missions).
  - Information services developed to support business-/mission-specific processes.
  - Data store services that are part of the enterprise infrastructure and consist of the following:
    - A repository that is a directory for looking up and calling processes and information services via the Web.
    - Online transaction processing (OLTP) databases for recording day-to-day transactions.
    - An enterprise data warehouse (EDW) for storing data that can be used for trend analysis and prediction.
    - Extraction, translation, and load (ETL) technology to move bulk data from databases to the EDW.
    - Enterprise application and integration (EAI) technology to provide data conversion services (wrappers) for current applications into information services and processes.
    - A portal to present browser-accessible, mission-focused information.
- An organizational infrastructure comprised of leaders (who define strategy/vision and performance metrics), stewards (who ensure information accuracy), and operational experts (who define process requirements and needs).

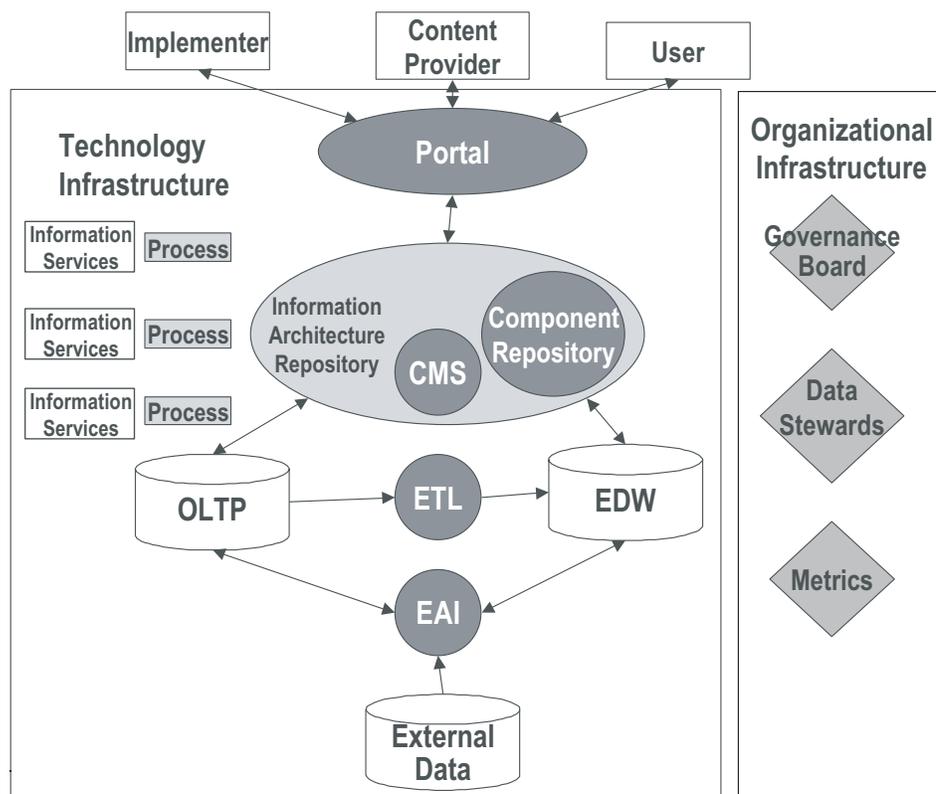


Figure 3: Information Architecture Strategy Components

For the Air Force, the information strategy leverages the GCSS-AF integration framework technologies while providing well-defined governance for the application framework implementation.

## Information Architecture Repository

The information architecture repository may be thought of as the *phone book* of the information architecture. For example, to make a connection to a process or an information service requires the following:

1. Look it up in the repository.
2. Use the resulting details for contacting the process or information service, including the information source, quality, format, and address.
3. Place the required calls to accomplish the mission.

Having the end user perform the above sequence of steps in his or her *operational* environment to create a new automated task removes the user's need to specify requirements for developers to implement the application in a traditional development environment. To grow these information architecture capabilities, end users or IT implementers must register new processes and information services in the repository providing all the characteristics required to enable automatic activation.

The information architecture repository is composed of two pieces: a content management system (CMS) for content

providers to manage unstructured data, and a component repository for content providers and implementers to manage structured data. The information architecture enforces a common information model (common categories and data attributes) on the CMS and the component repository. Standards such as the Dublin Core Standard for Metadata [2] (data about data) and the Universal Description, Discovery, and Integration (UDDI), when coupled with the domain-specific data classification categories of information architecture form the baseline data characteristics and assure common and easy access.

This dual repository defines, enables, and enforces the information architecture. The repository is not just a design artifact but also the production information access tool. Through registration in the repository, processes and information services are made available to authorized users within and without the enterprise. The repository enables information discovery, exchange, and flow.

## Integration Services

Figure 4 illustrates the flow of data through existing information architecture integration services (i.e., ETL and EAI) that support the composition of mission-specific information services.

### ETL

Extract, transform, and load technologies

are used to cleanse and normalize data. When stored in the data warehouse, these cleansed data allow for trending, aggregation across disparate sources, and general analysis of the data to support planned and unplanned investigations. The metadata-driven ability of ETL tools to enforce data definition is of particular importance to information architecture. Data extraction methods supported by the information architecture allow the timely extraction of newly created or updated data and can support scheduled as well as event-driven data-extraction operations. Efficient data extraction and filtering can be performed from databases and flat files without requiring access to application source code. This can also be performed without significantly affecting end-user response time.

### EAI

Many large enterprises have an extensive and effective information infrastructure in place. Unfortunately in many cases, this legacy information infrastructure was optimized for point solutions for segregated users (i.e., what has been referred to as *stovepipes*). EAI allows an enterprise to leverage the value of the existing enterprise stovepipe applications. In particular, EAI can wrapper current systems to appear as information architecture processes and information services, thereby garnering the enterprise information benefits without rewriting the current systems.

EAI initially will publish event information to the information architecture making it available to any authorized subscriber. As the information architecture matures, the EAI capability can be used to drive mission execution processes. For example, an event such as the reassignment of airmen will trigger such processes as user account updates, base housing activation, command notifications, or chaplain visits.

EAI is well suited to the publish/subscribe philosophy where data is cleansed and transformed into enterprise information at the source, in near real-time, on an event basis. EAI tools can be coupled with transaction monitoring tools capable of generating alarms when data or operational problems occur. Near real time information transfers and performance monitoring are obvious advantages in environments where the reliable delivery of information is important.

## Roles and Responsibilities

For any information architecture strategy to succeed, it must not only provide explicit guidance for the process and information service implementers as well as the content providers and end users, but also must pro-

vide oversight in establishing common goals and metrics, usually in the form of a governance body. In the Air Force's information architecture strategy, it is essential that the governance body consists of several *advocates*: 1) a doctrine operations advocate, 2) a technical systems advocate, and 3) an integration advocate, with doctrine advocacy taking precedence. The steward of the information must be the doctrinal owner of the operation. In order to institutionalize this process, an enterprise must support an information management board (IMB), which should comprise three members:

1. The chair is the transformational leader of the doctrinal operations and is responsible for all decisions, assuring that the operational architecture reflects current doctrine and concept of operations.
2. The information technology lead of the implementation (generally appointed by the chief information officer) is responsible for the system and technical architecture, assuring the operational architecture is implemented.
3. The integration scribe (appointed) is responsible for implementing and encoding the architecture in the GCSS-AF Integration Program.

The IMB is responsible for implementing the following:

1. The information architecture.
2. The definition of metrics and measures of effectiveness (MOE) of those metrics such as sortie generation capacity, mission capability, or cost per flying hour.
3. The documenting, prioritizing, allocating, authorizing, synchronizing, scheduling, and monitoring resolution of operational requirements.
4. The institution of lower-level IMBs to control information within doctrinal categories.

## Summary

Information architecture is more than just data in a database. It is a hierarchy of products and services organized to assure the timely and accurate delivery and storage of data and information across the enterprise. This article has described the components of an information architecture strategy using examples from the GCSS-AF Enterprise Architecture, which is currently under development as part of the GCSS-AF program. ♦

## References

1. Web Services Interoperability Organization <[www.ws-i.org](http://www.ws-i.org)>.
2. Weibel, S., J. Kunze, C. Lagoze, and M. Wolf. "RFC 2413 Dublin Core Metadata

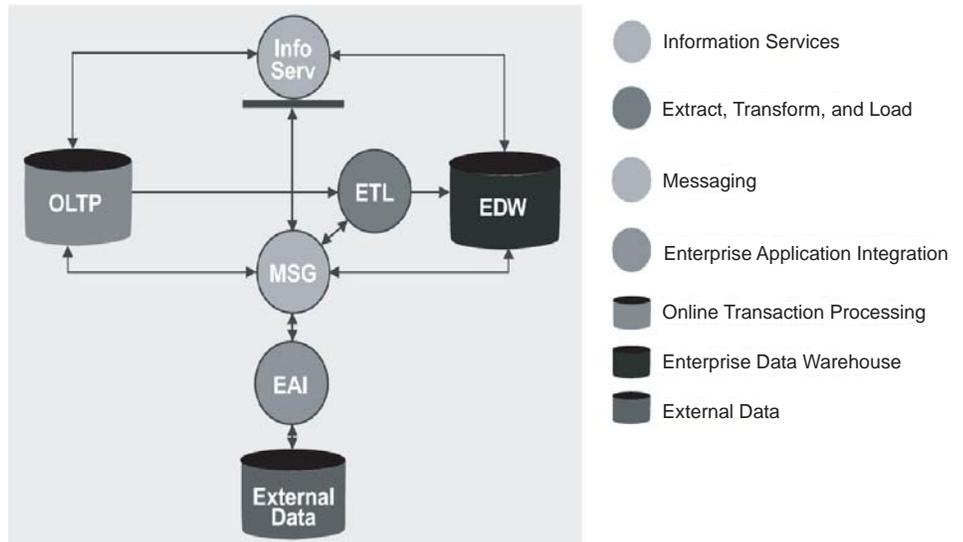


Figure 4: An Information Architecture Data Flow

for Resource Discovery." Sept. 1998 <[www.faqs.org/rfcs/rfc2413.html](http://www.faqs.org/rfcs/rfc2413.html)>.

## Notes

1. This article is based on the "GCSS-AF Information Architecture Strategy" White Paper, document number PROJ-2003-GCSSAF-0449, and is the result of

the efforts of several individuals on the GCSS-AF project.

2. A Web service is defined as a service that is available on the World Wide Web, is accessible via Web protocols, is self-defining for humans and machines, and is registered in a Web repository for easy lookup and activation.

## About the Authors



**John Wunder** is a certified Lockheed Martin architect and has been the lead system architect/composer on the Global Combat Support System-Air Force since 1999. Prior to this, he was lead architect for Dow Chemical Process Control, and software architect for the U.S. Army battlefield digitization project. Wunder has been involved in information technology for more than 20 years.

**Lockheed Martin Mission Systems**  
**1801 State Route 17C, MD 0605**  
**Owego, NY 13827**  
**Phone: (607) 751-6096**  
**Fax: (607) 751-2538**  
**E-mail: [john.wunder@lmco.com](mailto:john.wunder@lmco.com)**



**Will Tracz, Ph.D.**, is a principal research scientist in the Global Combat Support System-Air Force (GCSS-AF) Enterprise Engineering department at Lockheed Martin Mission Systems. He is responsible for investigating innovative applications and extensions of the GCSS-AF Architecture Integration Framework. Tracz is a member of Lockheed Martin's Corporate Advanced Software Technology Focus Group as well as an ad hoc member of the 1999 Air Force Scientific Advisory Board Commercial Off-the-Shelf study. He is a member of the Rochester Institute of Technology Software Engineering Advisory Board, editor of the Association for Computing Machinery SIGSOFT Software Engineering Notes, and the author of more than 100 technical reports and books.

**Lockheed Martin Mission Systems**  
**1801 State Route 17C, MD 0605**  
**Owego, NY 13827**  
**Phone: (607) 751-2169**  
**Fax: (607) 751-2538**  
**E-mail: [will.tracz@lmco.com](mailto:will.tracz@lmco.com)**

# Warfighter's Access to Geospatial Intelligence

Peter Winter  
Harris Corporation

*Imagery intelligence and geospatial intelligence information have become extremely critical to U.S. warfighters under the current demands of the war against terrorism. The National Imagery and Mapping Agency's National System for Geospatial Intelligence (NSGI) provides access to such information through standard browsers into online data holdings. This article explains how the NSGI provides this high-caliber imagery intelligence and geospatial information in real time to the warfighters when they need it, wherever it is within any of the data holdings to aid in mission success.*

The demands of the war against terrorism require timely, relevant, and accurate imagery intelligence and geospatial intelligence information. The National Imagery and Mapping Agency's (NIMA) National System for Geospatial Intelligence (NSGI) provides access to such information via standard browsers into diverse data holdings containing terabytes of online imagery, imagery intelligence, and geospatial products. The NSGI enables the warfighters to locate the data they need, when they need it, wherever it is within any of the data holdings to aid in mission success.

The challenge is to provide this high caliber imagery intelligence and geospatial information in real time. There are more than 300 product types and formats of imagery, imagery intelligence, and geospatial information each with a significant rate of change per day. Also, there are thousands of image segments and intelligence reports with terabytes of online products to query.

To overcome these challenges, the tool Information Access Services (IAS) provides quick access to imagery and geospatial information and products stored in many interconnected NSGI libraries around the world. Its Web-based forms

provide one tool with which to discover a wealth of information. IAS makes all underlying source and information types and library locations transparent to IAS users; it sees a single Web-based set of forms. The user selects the libraries he or she wants to query, creates a query or uses an existing query, and submits the query. IAS displays thumbnails and overviews of the query results, allowing the user to order only the products of interest.

## Mission

The NSGI is a worldwide network of data collection, data holdings, and specialized analysis tool sets, all supported by a confederation of systems developed by multiple contractors located throughout the United States. To distribute the right information at the right time, the NSGI enterprise was devised as an integration of technology and collection capabilities to enable geospatial intelligence analysis on whatever source imagery, imagery intelligence, and geospatial mission-relevant data is available, all to support the warfighter's mission without information overload.

The NIMA data holdings within NSGI include imagery, imagery intelligence, and geospatial information (maps, elevation, charts, and features). NSGI supports

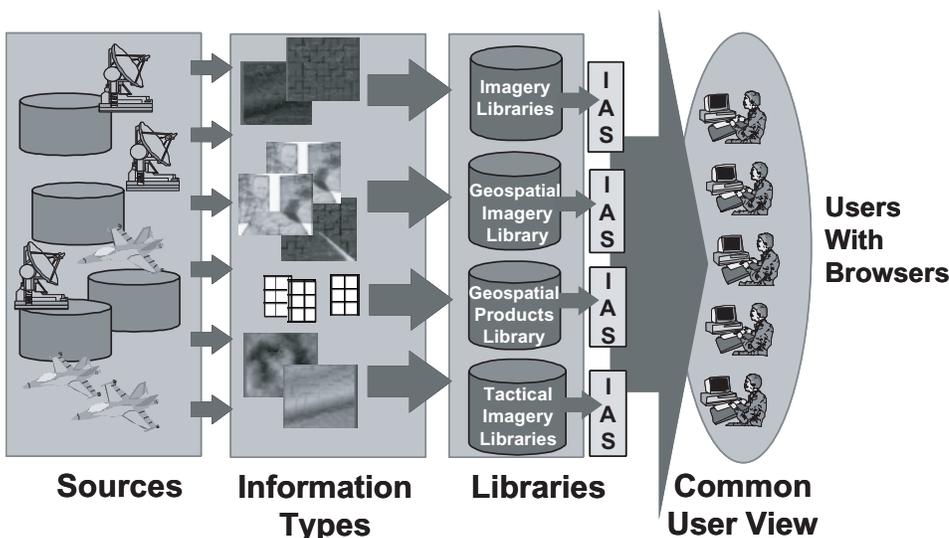
Department of Defense (DoD) initiatives to transition from standard-scale map and chart production to easily accessible digital information that satisfies military imagery, imagery intelligence, geospatial, mapping, charting, and weapons system requirements. The NSGI's goal is to provide a common view of this disparate information as shown in Figure 1.

All segments within the NSGI contribute to NIMA's vision of a seamless integration of its data information resulting in a community of interoperable NIMA information libraries that provide integrated geospatial intelligence to NIMA's communities. The information libraries comprise the information heart of the NSGI enterprise. Data from multiple sources are constantly stored and cataloged in the libraries based on a common logical view of the information, which is documented in the Discovery and Retrieval Interface Data Model (D&R IDM). Developed by Harris Corporation, the D&R IDM provides the data structure to define the relationship between the stored items (imagery, imagery intelligence, and geospatial products) and metadata attributes about the stored items.

Metadata is an informational fact that describes the geospatial, imagery, or imagery intelligence product. Metadata can include information about the stored items such as currency, accuracy, data content, source information, and coverage. The physical implementation of the D&R IDM data model by all libraries provides a common set of metadata attributes for the user to query, passing metadata attributes between client and information libraries. The common set of metadata attributes provides the user a common view of the available data holdings. Queries can be constructed using metadata attributes that are valid across all libraries, and query results can be sorted by resulting attributes to facilitate analysis.

An imagery analyst (IA) uses the results of the queries to perform image exploitation, which is the extraction of information to provide knowledge – or more

Figure 1: IAS Presents the Common User View From Disparate Information Sources



specifically intelligence – about an area on the earth. Extracted information can be a description of the image contents, measurements of features of interest, comparisons of new images to previous images to determine differences to assess damage or movement, terrain analysis, or precise location of objects of interest. The time needed to precisely locate objects of interest has been significantly reduced as a result of the NSGI.

A geospatial analyst (GA) uses the results of the queries to generate digital maps and charts. The resulting geospatial products are stored in the libraries for use by others. IAS provides several other critical components of the NSGI enterprise in support of the IA and GA.

## IAS Features

The Discovery and Retrieval Client 2001 provides a common user interface of the NSGI information libraries and is composed of application programs that can be executed from a user platform that is configured to meet community standards for communications and security compliance. The Protocol Adapter is deployed with Client 2001 for access to libraries that are not compliant with the most recent specification. Profile Services provides a single point of authentication for user access and a single point of storage for user communities. Stored information such as query results and public-saved queries can be shared within the user communities.

The Client 2001 is a powerful data access and retrieval tool that operates much like a Web-based search engine for imagery and intelligence data. With Client 2001, the warfighter or intelligence analyst can quickly locate and retrieve the information needed to perform his or her mission, even across distributed and dissimilar data holdings. The NSGI data holdings seek to provide the warfighter and intelligence analyst with the most current capability to store and catalog petabytes of mission data – more data than ever before available. The IAS enables the warfighter and intelligence analyst to locate the data they need at the right time wherever it is within any of these data holdings.

Both the Client 2001 and the information libraries make it easy for users to query the NIMA holdings for existing available geospatial, imagery, and imagery intelligence products. Queries are based on geographic areas of interest, product identifiers, and generalized parametric descriptors (for example, scale, resolution, accuracy, or currency). Users can focus in on areas of interest and only retrieve information that is truly relevant to their search

request. The IAS also enables users to browse online metadata, including browse views, thumbnails, and overviews of selected products; to graphically and textually view metadata; and to place orders. Users with sufficient hardware, communications bandwidth, and priority can receive orders online. Orders for physical media (maps, tapes, CD-ROMs, etc.) are delivered offline.

Those who repeatedly use the same query and result attributes and query the same library(s) can use Query Express. The Query Express page is pre-configured to show all the needed query fields with no user set-up required. It can be used as a template to be filled in with operationally significant values, or the values could have already been saved so that the user just reviews the query form and selects the submit button to initiate the query in a single button selection.

The Favorite Query option enables the user to open a query that was previously saved as a Favorite Query. This feature provides a quick method to open a frequently used query without having to access multiple pages. Favorite queries include Public Favorites (saved in a public workgroup folder) and Personal Favorites (saved to your personal Favorite Queries folder). Client 2001 provides an automatic query and order feature for advanced users that allows them to build queries that are set to automatically execute at specific times or intervals during the day and automatically forward ordered results. The results discovered are available to the user when the user logs on.

Information flows from the NIMA production systems to the information libraries for user access. IAS users also have the capability to store information in the libraries. The user-populated information is then made available to other users as value-added data for NIMA production purposes.

IAS users have unique user identifications, passwords, and account information that are specific to their use of the NIMA data and are consistent with network provider standards. The identification information, stored in Profile Services, is used to control access to library information and to provide for individual preferences and defaults. Preferences allows the user to tailor the default options that are used on many of the Client 2001 pages. After granting user access, the information libraries also control access at the data level within the library.

The Client 2001 online help capability provides NIMA users with accessible training information. It provides tutorials

on how to perform specific IAS tasks and the Online Learning Capability (OLLC). The tutorials are organized by pages and topics, and can be used to search for specific topics and words. The OLLC provides sectional links within Client 2001 pages. The sectional links (*Show Me*) display relevant cue cards for critical functions, providing the user with step-by-step instructions and examples to assist him or her during task performance.

The Protocol Adapter is deployed with Client 2001 for access to libraries that are not compliant with the most recent specification. The Protocol Adapter performs an interfacing function to make non-NSGI systems available to the NSGI enterprise. Such systems may include legacy systems, prototype systems, or systems that have not maintained compliance with the NSGI interfaces. The Protocol Adapter is a software service that acts as a translator allowing Client 2001 to communicate with non-NSGI systems. To a library, the Protocol Adapter appears as a client, using that library's native client Application Program Interface. To Client 2001, the Protocol Adapter appears as a library using the NSGI standard.

## Architecture

IAS consists of thin client-server architecture operating with the client's workstation-standard Web browsers. IAS supports the DoD, federal, and internal NIMA production users. Access is via defense and intelligence networks. The architecture is scalable to support a growing user community, flexible to incorporate new user services, and extensible to exploit emerging technologies. The architecture also includes mechanisms to ensure that access to classified and sensitive data is controlled and granted only to authorized users.

Figure 2 (see page 10) depicts the IAS architecture and also the physical location of the architectural component within the NSGI enterprise. In the IAS implementation, the presentation layer follows the lightweight client model of the World Wide Web. This model eliminates the need for specialization on the user's workstation. The only element required on a user's workstation is a Web browser that complies with Hyper Text Markup Language. The interface between the application layer – Client 2001 – and the presentation layer is the Web server, which is accomplished via a commercial off-the-shelf (COTS) product called Iplanet. The application layer, or Client 2001, implements the business logic needed to present information to the user and to interpret the user's actions. The application layer constructs

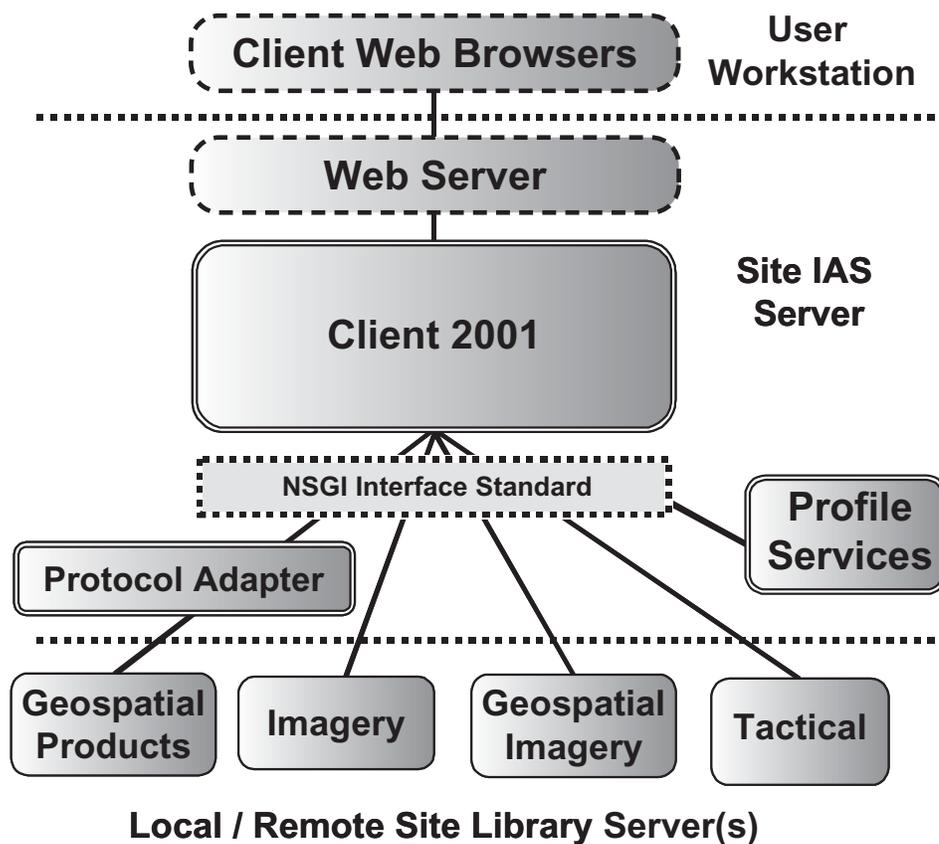


Figure 2: *The Information Access Services (IAS) Architecture*

the query as defined by the NSGI interface standard and distributes the query to the libraries, which represent the information layer of the NSGI architecture. The complexity of the IAS application layer is its interoperation with multiple instances of information layers, or libraries, providing an integrated view of information across a distributed enterprise to the user.

The NSGI interface standard documents the interfaces, data types, and error conditions that are expected to occur across the NSGI architecture. This specialized Interface Definition Language (IDL) allows clients access to data that have an association with a point or area on the earth. A NSGI-compliant library has interfaces that allow a client to search and discover information contained within the library, get details about a particular item stored in the library, and arrange for the delivery of the product.

For an enterprise as distributed as NSGI, it is necessary to define how systems can interoperate without being constrained by an operating system or language. The NSGI mandates the use of Common Object Request Broker Architecture (CORBA) middleware to provide this interoperability as part of the interface standard. CORBA is based on industry standards developed by the Object Management Group (OMG) to enable sharing of objects across layers. CORBA

requires that IDL be used to specify CORBA-transported objects and interfaces and make these objects and services accessible to other CORBA objects. This allows the object's services and information to be exposed to each other and to other systems' objects across the DoD Wide Area Networks. As required by the OMG, IDL specifications are generic enough to be compiled in several different languages, including C++ and Java.

The Protocol Adapter architectural component provides access to libraries not compliant with the most recent specification. As depicted in Figure 2, the Geospatial Products Library requires a Protocol Adapter because it is a library that is operating with an old, no-longer-supported version of the NSGI Standard. The Profile Services architectural component provides services to Client 2001 as a single point of authentication for user access and a single point of storage for information about the user, user-saved queries, and query results.

### IAS Deployed Configuration

The IAS is deployed worldwide, providing information access to thousands of users. The IAS-deployed environment generally consists of a 2-CPU Sun Enterprise E4500 server with T3 disk array. The COTS items hosted on the server include Java, C++, Netscape, Solaris, IONA, RogueWave,

Perl, Oracle, and iPlanet. The IAS software may also be deployed on a server as small as a laptop and up to an E6500 with 16 CPUs, depending on the deployment site workload and user base.

Most deployment sites consist of fixed location computing centers but may also include mobile environments such as ships, aircraft, and the High Mobility Multipurpose Wheeled Vehicle. Generally there is an instance of the IAS deployed with each library. These platforms exist in a secure distributed enterprise environment with users logging in from all parts of the world to obtain imagery data from the various NIMA imagery libraries. User login is through Hyper Text Transfer Protocol from a browser on their local workstation to the IAS server being accessed.

### Information Collaboration Critical to Successful Development

Information sharing and collaboration is also an important part of the development process. The IAS project has been executed with a close partnership of the customer and involvement of the user communities throughout the phases of release development. Requirements are developed in conjunction with the customer as one cohesive team and are reviewed by user representatives. In the development of requirements and design, IAS conducts daily on-site interface with the user communities providing feedback to the developers to ensure requirements and design reflect users' needs and concept of operations.

Requirements are captured and maintained in a database and linked with the following requirements analysis products: requirement interpretation, assumptions, issues, use cases, and test allocations. The project database is central to ensuring everyone involved in the development of IAS has a traceable linkage from requirements through design products to verification test procedures. IAS uses documented processes to build in quality and maintainability and to ensure completeness throughout product development and maintenance.

Adherence to the process ensures that quality is built-in not added-on. Project processes are documented in Process Directives (PD). These directives are based on Harris Government Communications Systems Division (GCSD) standards for executing projects called the division command media. The command media is tailored and elaborated for project execution. The PDs define in detail the project process throughout the

product life cycle, from requirements to product maintenance. Process is planned into a project master schedule containing detailed milestones, which provide the project management team and customer with detailed insight into the project execution.

Each function within the project acts as either a customer or a supplier. Customer-supplier checkpoints ensure the process is followed. The checkpoints and major process steps are detailed in the project milestone schedule and reported internally weekly and to the customer monthly. The process itself is open to quality improvement. PDs are sponsored and maintained by project functional groups. Improvements to process directives can be recommended by anyone on the team to the System Configuration Control Board.

The IAS project uses an Engineering Quantitative Management Plan to define the project metrics and control limits that are collected and reviewed monthly by the Project Management Team. The metrics are collected using the Harris GCSD standard metric tool (Web client/database server) [1]. The monthly review of the project metrics provides a quantitative view of the project product and processes, early visibility into possible trends, and actively identifies areas for improvement.

The IAS project collects and uses the trends of a variety of quality measurements to improve project execution and maintain or improve quality. Progress is tracked in the following areas: cost and schedule, risk assessment, resources (development environment and staffing), problems and/or defects with a product or process, system performance, stability of the degree of change, and completeness. The IAS Cost Performance Index and Schedule Performance Index remain at 1.0 or greater, demonstrating the adherence to process and good planning.

The metrics that the project monitors for adherence to plan or prediction include the following: defect work-off rate, defect discovery, defect closure status, defect severity, cost, schedule, engineering staffing, and risk assessment impact in dollars. These monthly metrics provide measure and insight into knowing how well the plan is being executed, detecting trends, and allowing decisions to be made on efforts that could be accelerated or need refocused resources to accomplish the project's goal.

## Conclusion

The NGSI has met the challenge of integrating disparate information sources and

providing a common user view into the data holdings of imagery, imagery intelligence, and geospatial information. The IAS discovers the data of interest, allows the user to browse views of selected discovered products, graphically and textual-ly views data about the products, and delivers orders of selected products either online or offline. The IAS enables warfighters to locate the data they need wherever it is within any of the data holdings at the right time. Through user feedback and process improvement, the NGSI will continue to implement user-driven enhancements to meet the evolving needs of the warfighter. ♦

## Reference

1. Natwick, Gary. "Integrated Metrics for CMMI and SW-CMM." *CROSSTALK* 16: 5 (May 2003).

## About the Author



**Peter Winter** is the chief system engineer for the Information Access Services project. He has 25 years experience in aerospace, information processing systems, and launch control. His experience includes requirements development, designing, integrating, and testing large-scale software and hardware systems. He has Bachelor of Science and Master of Science degrees from Florida Institute of Technology.

**Harris Corporation**  
**P.O. Box 37**  
**Melbourne, FL 32902-0037**  
**Phone: (321) 309-2442**  
**E-mail: [pwinter@harris.com](mailto:pwinter@harris.com)**

### TOP 5 QUALITY SOFTWARE PROJECTS

SPECIAL SOFTWARE QU

**CROSSTALK** ♦

January 2002 The Journal of Defense Software Engineering

2003 U.S. GOVERNMENT'S

**2003 U.S. Government's Top 5 Quality Software Projects**

The Department of Defense and CrossTalk are currently accepting nominations for the 2003 U.S. Government's Top 5 Quality Software Projects. These prestigious awards are sponsored by the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, and are aimed at honoring the best of our government software capabilities and recognizing excellence in software development.

The deadline for the 2003 nominations is December 5, 2003. You can review the nomination and selection process, scoring criteria, and nomination criteria by visiting our Web site. Then, using the nomination form, submit your project for consideration for this prominent award.

**FOR MORE INFORMATION OR TO ENTER,  
PLEASE VISIT OUR WEB SITE**

**[www.stsc.hill.af.mil/crosstalk](http://www.stsc.hill.af.mil/crosstalk)**

# Effective Collaboration: People Augmented by Technology

Richard L. Conn  
Microsoft Corporation

*We are entering a new decade in the world of computer technology, and objectives like the creation of trustworthy software, improvement in capability maturity, and the ability to “do more with less” by applying technology wisely are moving from dreams to reality. We now have the technology in most cases, and the issue we face is how to mold it to fit our needs and apply it. Collaboration is at the heart of most of our endeavors, and this article presents a number of ideas on how to collaborate more effectively by wisely augmenting people with technology. It briefly discusses the importance of collaboration for the success of a project, discusses the concept of collaboration, presents a case study in how collaboration in teaching has been augmented by technology, and provides pointers to the technologies which the reader may find useful in increasing the effectiveness of his/ her collaborations.*

The operation of today’s software-intensive, mission-critical systems can determine a company’s success or failure. Factor in a safety-critical element and these systems can make the difference between life and death. One does not have to look far in our society to see these systems. Communications, finance, air transportation, defense, and medicine are fields that often rely heavily on software-intensive systems, and the consequences of the failure of a key system can be significant.

In many cases, these software-intensive systems are far too complex for a single person or small group of people to create. Collaboration, sometimes on a large, international scale, may be required to create such systems. The greater the number of people involved in an effort, the greater the need to augment them with technology. The technology exists to capture raw data in many forms (written, audio, video, sensory, digital), to index this data to make it easier to retrieve later, to digest volumes of data into succinct and useful information, and to control the distribution of this data and the information derived from it. We need to apply our technology wisely to augment people and make their collaborations more effective.

## Collaboration Complexities

Collaboration can take many forms, particularly when you extend the concept to include smart devices as well as people. Let us first tackle the obvious – people.

People working together to achieve a common objective is how we normally think about collaboration. To make such a collaboration work, people require the following:

- A defined objective. This objective may or may not be clearly defined at first, but as the collaboration gets underway it becomes better defined.
- Willingness to collaborate. The collaboration will be less effective if the peo-

ple are not able (possibly due to legal or political constraints), willing, or motivated to collaborate.

- Ability to communicate. At the fundamental level, we communicate through our senses, so the more senses that are

---

**“The greater the number of people involved in an effort, the greater the need to augment them with technology ... We need to apply our technology wisely to augment people and make their collaborations more effective.”**

---

involved in the communications process, the better. Textual communication in a common language using a common set of terms (which may or may not be well defined) is a starting point. Images (photos, diagrams) can be added to enhance the effectiveness of the communication. Video (video clips, interactive dialog via cameras and sound) enhances it further. Full personal contact, where body language comes into play, brings all the senses (even smell) into play.

Collaboration can take place without anything else; but often it is not effective without adding a few enhancements such as the following:

- The objective must be clearly defined; success in achieving the objective must be measurable. For the collaboration to be effective, the team must be able to analytically determine when the objective has been achieved.
- Communication must be based on a language with as few dialects as possible using terms with as little ambiguity as possible. Communication must also be captured and made available for future reference in an organized form that is as easy as possible to search. Common threads must also hierarchically organize captured communication so the chain of reasoning can be followed.

Collaboration between smart devices involves a similar set of requirements. One good way to think of these devices (and people, for that matter) is to apply a Task-Object-Event model. In such models, the objective is couched in terms of a task to be performed, and this task may be divided into subordinate tasks, and so on. An object (such as a smart device, person, or team) is assigned to perform one or more tasks. Events (such as the click of a mouse or a manager’s order) trigger the objects to perform the tasks. For smart devices to collaborate, the devices require the following:

- A defined objective. The task(s) to be performed is the objective of a smart device. The task(s) may be expressed as a series of testable, measurable requirements, often with binary quality gates being the basis of deciding success or failure in the performance of the task(s).
- Willingness to collaborate. The interfaces of one object to another must be compatible, based on common standards, and precisely defined to the point where they can be compiled. Smart devices are motivated to collaborate – they are designed to reach out

and interact with other smart devices.

- Ability to communicate. The smart devices must be connected to each other and employ common communication protocols and data transfer standards to communicate. Smart-device communication based on international standards such as Transmission Control Protocol/Internet Protocol and eXtensible Markup Language (XML) can be extremely effective and rich, allowing a wide variety of data (textual, video, or binary) to be transmitted. Some form of network connectivity based on international standards (such as Universal Serial Bus, 802.11x, and Ethernet) is built into most smart devices.

Hundreds of different kinds of smart devices exist today based on common technologies like the .NET Framework, the .NET Compact Framework, and the Java Virtual Machine (JVM). These smart devices include personal computers (PCs) such as desktops, laptops, tablet PCs, personal digital assistants (PDAs), automobile PCs, and more; portable digital music players (often with voice recording capabilities); portable digital video players; cell phones; convergent devices (like PDA/cell phone combinations such as the Smart Phone); and a variety of customized devices now being embedded into other devices such as microwave ovens, televisions, and the walls of a house.

One of the key features of .NET and the JVM is that extensive collaboration facilities based on industry standards (such as XML and Web services) are built-in. This provides an infrastructure by which the smart devices may collaborate, and augments the collaboration capabilities of the people using them as well. Let us look at a few examples now on the drawing boards.

Imagine first-grade students in a classroom utilizing smart devices such as tablet PCs. The teacher is instructing the students in printing letters of the alphabet. They are writing on their tablet PCs while the teacher monitors their work from a console at the front of the classroom. As Johnny mistakenly forms a “b” backwards, the teacher sees it immediately and corrects Johnny’s efforts on the spot. The teacher also sees about half of the students making the same mistake, so he or she opens up a dialog with the entire class about it.

Imagine also a smart home. Johnny returns home from school gaining entrance through the front door by a reti-

nal scan. Signals are sent to unlock the door, turn on the entry lights, post messages to Johnny on a display screen inside the home (with emphasis placed on the priority messages), and begin playing Johnny’s favorite music. The home contains a collaboration of smart devices.

On a grander scale, consider the F-35 fighter aircraft program run by Lockheed Martin Aeronautics that incorporates smart technology. Sponsored by partners and supported by subcontractors in several countries, a Web-based information portal server is used to divide this large community into its interest areas and allow information to be exchanged, shared, and organized on a global basis. Information that should be visible to all participants is readily shared, and information particular to specific groups in this project can be shared without the other groups seeing it.

---

***“Information that should be visible to all participants is readily shared, and information particular to specific groups in this project can be shared without the other groups seeing it.”***

---

### **Technology-Based Collaboration in a Teaching Environment**

Microsoft not only develops a variety of technologies and programs to support collaboration for its customers, but also uses its own products internally. These collaboration tools are often used extensively internally and by its partners before the public at large sees them.

I recently attended a class at our main campus in Redmond, Wash., and it provides a good example of how collaboration between people (in this case, students and teachers) augmented by technology (such as smart devices) can work. The class was diverse, consisting of more than 100 students from 24 countries. English served as the common language, although the speakers had to be reminded from time to time to speak more slowly. The following sections describe this

environment.

### **Hardware and Class Environment**

The classroom was in Building 43, a controlled-access building. Students were given smart card badges containing a computer chip that could be scanned by the door entry mechanism to allow access. Students were cleared for access to some parts of the building and not others; their movements were tracked. The badges also plugged into the students’ laptops and tablet PCs to log them into their computers.

The cafeteria, with its waterfalls and garden, is located adjacent to Building 43, and also required smart card access during normal hours (cafeteria employees may use their smart cards to get in after hours). Building 43 has both Ethernet (at 100Mbps) and wireless access points (at 10Mbps). The classroom was filled with tables that provided power and Ethernet access points at every seat. Should a partial network failure occur (such as an Ethernet router failure), the alternate network was immediately available.

More than half of the students used laptops, while the other half used tablet PCs. Almost all the students had PDAs or Smart Phones (integrated PDA and telephone) as well. I found it interesting to note that the tablet PCs, with their handwriting and sketching input capability, made it easier and were less distracting for taking notes during class than the laptops. Finally, the instructor podium was like something out of Star Trek – videotapes, DVDs, and CDs (audio and digital) were fed to the computer in the podium and displayed to the students on a large screen and sound system at the front of the room.

The staff for the course included a main instructor (who stayed with the students as a common point of contact throughout the entire course), several guest instructors, an administrative assistant, and an on-call hardware/software support group. Course material was prepared using a variety of desktop publishing tools, database servers, Web-based information portals, and operating systems as well as the video production studios on the campus. Many of our partners also played a role, preparing sections of the course using their own resources.

### **The Software**

From the start of class, the main instructor was at the podium showing the students the Sharepoint Portal Server site and how to access the portal. Through the portal, the students could sign up for the

course e-mail distribution list, download slides and videos of the presentations and related material, download course programs and data files for their devices (laptops, tablet PCs, and PDAs/Smart Phones), perform evaluations of the course elements as they occurred, follow Web links to material related to the course, and upload material (such as homework) for the course. The main instructor also provided his e-mail address and the e-mail address of the administrative assistant.

As the course progressed, the portal was updated with information from the class; for example, after the introduction of the students, statistics on the demographics of the student body were posted on the portal. Prior to the class, the students' laptops and tablet PCs had been configured for secure access to the company's intranet (wired and wireless), and software had been installed to allow the students to play the videos and view the other class material.

### Communication Vehicles

The course was highly interactive, employing a wide variety of technologies to communicate with the students and help them learn. The students were already motivated to learn the material, and the technology greatly aided the learning process, making it both efficient and fun.

Presentations were vivid and highly animated, extensively employing color and animations effectively in both the slides to enforce the messages and in full-motion video to supplement the slides where reasonable. In some cases, the material was too large for a class of over 100 students to download efficiently, thus CDs and DVDs were distributed instead.

Video-oriented simulations were included on CD for some class exercises. Students ran the simulations, viewing videos and interacting with them, causing the sequence of videos and questions posed to the students to dynamically alter the simulation in response to the correct and incorrect answers they submitted.

### Technology Control

One key feature of teaching in this environment is that the instructor had to maintain firm control of the students' use of their devices (laptops, tablet PCs, and PDAs/Smart Phones) during the class. There was a distinct tendency for students to want to use them during lectures when their attention should be focused on the front of the room; this proved to be a distraction to the instructor and the other

students unless it was controlled.

Surprisingly, pen and paper were still used – particularly for those students without tablet PCs. The PDAs (using handwriting and voice dictation), the tablet PCs (using handwriting and handwriting recognition, speech recognition, and voice dictation), and pen/paper were the methods of capturing notes during the lectures. I imagine that future classes will be entirely based on pocket and tablet PCs; with digital ink, pen and paper can finally become a thing of the past.

### After the Course

From the first day, the instructor announced that the course portal and e-mail distribution list would be available to the students for up to a year after class completion. Special announcements of updates to the material would be sent to the students through the e-mail distribution list and a course newsletter. Updates

---

*“... collaboration is often required to develop many of our software-intensive systems for mission-critical, and sometimes safety-critical, applications.”*

---

to the material would be posted, and the students could use the portal to stay in touch.

Hybrid audio/digital CDs were distributed to the students so they could go over the material while driving their cars or working with their PCs (the digital part of the hybrid CDs included transcripts of the audio part). Updates to these CDs would be mailed to the students periodically.

Most students downloaded material from the company intranet to take back with them. In my case, I downloaded about 1.8G bytes of material knowing that I would have high-speed Internet access to these resources after I returned home. Other students had Internet access as well, but, due to their location, it was less reliable or not operating at a high speed. A student from South Africa, for example, brought in a USB 2 disk drive with 250G bytes of storage and down-

loaded over 130G bytes of material to take home.

With such a diverse group, the dual nature of the digital divide was evident – all students were on the positive side, having access to the Internet, but some students had slow-speed (56K baud or less) while others had high-speed (200K baud or more) access. With large volumes of material, the speed of access can make a difference.

Information was also provided in a format suitable for use on smart devices (particularly PDAs and Smart Phones), and the students installed this information during the class. The smart devices were also filled with task lists of ideas for the students to pursue when they returned home. Finally, the students were encouraged to take the follow-on class after returning to their jobs.

### Foundation Tools and Technologies

While the previous scenario took place in a classroom setting, the same technology can be applied in a virtual setting in which the participants are geographically scattered. Obviously, it would make collaborations much more effective in virtual enterprises where people are geographically distributed. In addition, while many large companies have extensive resources to call upon (such as video production studios), many tools are available that allow both small companies and individuals to set up their own collaboration mechanisms inexpensively.

Conferencing tools can be used to broadcast live video among several groups or individuals through the Internet. These tools also support an electronic white board that allows participants to draw diagrams using multiple colors (perhaps a different color for each participant) on a common board that all can see. Voice communication, of course, is included. To set this up, a server is required and common conferencing tools (or conferencing tools based on common protocols and standards) must be installed on each client.

Several video capture and editing tools are available to create video productions using just a PC with a Web camera or digital movie camera. Extensive editing and publication capabilities (including publication onto a DVD) are included. To set this up, a PC with an optional video capture card and the video capture/editing/production software is required.

Courseware authoring tools are available to create rich interactive presenta-

tions that the instructor can run in class to augment his or her presentation, and the student can run at home. These tools also allow the instructor to set up Web sites that allow the students to acquire homework and submit answers to homework, take exams, and interact with other students.

Desktop publishing tools provide the resources for creating the slides, documents, databases, and spreadsheets that may be needed. Free viewers are often available for download, so the clients do not have to have the full suite of desktop publishing tools installed on them. Tools for collaboration, providing e-mail, a calendar, task list, and contact list support, both locally and through the Web, are also needed.

Behind these tools are fundamental technologies that form a very effective infrastructure for collaboration:

- The .NET Framework (for larger smart devices like laptops and tablet PCs), the .NET Compact Framework (for smaller smart devices like PDAs and Smart Phones), and the JVM provide common virtual machines with a wide array of reusable components for application programs to exploit. These commercial frameworks from Microsoft and Sun are used by most developers to provide common virtual platforms that transcend the physical computers (Windows PCs, UNIX workstations, mobile devices) as well as the Internet (allowing the frameworks to extend to Web servers).
- XML provides a common way to store and transport data, retaining the rich context of the data in the process.
- Web services provide a common way to share resources and capabilities without concern about the location of those resources or capabilities (they may be on the same computer or on different computers without any impact to the application code). With Web services, a single cell on a spreadsheet can be tied to other cells on spreadsheets around the world, allowing a change at one location to be automatically viewed at other locations.

With these fundamental technologies in play, the door to more effective collaboration in both the enterprise and the classroom is wide open! But our current foundation tools can still be improved. Organizations such as Microsoft Research and Lucent Technologies (formerly Bell Labs) are investigating potential technologies in a wide variety of broad areas, including but not limited to the following:

- Management of digital photographs

and full-motion videos, including the ready extraction of useful information from them.

- Online communities.
- Next-generation smart devices.
- Mobile computing.
- Speech recognition and meaning interpretation.
- Signal processing.
- Databases and information mining.
- Ubiquitous computing (making the usage of computers transparent to humans).
- Intelligent reasoning and decision making.

## Conclusion

Collaboration among people augmented by today's technology is taking off like wildfire across the world, bringing the global community closer together in many ways. More and more effort is being poured into making this technology adaptable to different needs, 99.999 percent reliable and beyond, and secure to the point where the users can trust the technology to support and protect their privacy. Collaboration among large numbers of people, sometimes in different countries, is becoming more common, and collaboration is often required to develop many of our software-intensive systems for mission-critical, and sometimes safety-critical, applications.

Collaboration is taking place between both people and smart devices, often augmenting the people with smart devices and other more advanced technologies. A real-world example of collaboration in a teaching environment was presented in this article, and in this example, smart devices and technology significantly augmented people. This is happening today!

Many foundation tools and technologies are available today to support augmented collaboration. But we clearly do not have all the answers, and an increasing amount of research is being done to enhance our ability to collaborate by augmenting people.

Large volumes of material related to the topic of collaboration and technologies in support of collaboration are freely available to the public on several Microsoft Web sites:

- <[www.microsoft.com](http://www.microsoft.com)> is the main entry point for all information from Microsoft Corporation.
- <<http://msdn.microsoft.com>> is the Microsoft Developer's Network with detail upon detail about the technologies, including the .NET Framework, the .NET Compact Framework, XML, Web services, and the new open shared

source code.

- <<http://research.microsoft.com>> is Microsoft Research, with details and contacts for more information on Microsoft's research thrusts such as online communities, mobile computing, and much more.
- <[www.msnaa.net](http://www.msnaa.net)> is the Microsoft Academic Alliance, a vast resource of material and information for all educators from the Microsoft Corporation.
- <[www.mainfunction.com](http://www.mainfunction.com)> is a resource sponsored by Microsoft for educators in high schools.

Supplementary material related to this article can be found on my university Web sites at <<http://unicoi.kennesaw.edu/~rconn>> and <<http://cs.spsu.edu/rconn>>.◆

## About the Author



**Richard L. Conn** has more than 20 years of experience in software engineering, project management, and education. Conn is currently a university liaison for Microsoft, serving as an ambassador between Microsoft and many universities and participating on industry advisory boards for several universities. He also teaches as an adjunct professor for the Computer Science and Information Systems Department at Kennesaw State University and the Computer and Software Engineering Department at Southern Polytechnic State University. Conn has taught graduate school at the Air Force Institute of Technology in electrical and computer engineering, and at Monmouth University in software engineering. He has designed Capability Maturity Model Level 4 software engineering processes for Lockheed Martin Aeronautics, served as a government consultant working for The MITRE Corporation, served on the Federal Advisory Board for Ada, and contributed to the Department of Defense Software Reuse Initiative as a distinguished reviewer.

**Microsoft Corporation**  
**Education Solutions Group**  
**One Microsoft Way**  
**Redmond, WA 98052-6399**  
**Phone: (678) 521-3440**  
**E-mail: [rconn@microsoft.com](mailto:rconn@microsoft.com)**

# Serialized Maintenance Data Collection Using DRILS

Capt. Greg Lindsey  
Ogden Air Logistics Center

Kevin Berk  
Total Quality Systems, Inc.

*The Defense Repair Information Logistics System (DRILS) is a specialized Web-based application that collects, stores, and retrieves Air Force depot and field maintenance data. A Web browser is all that is needed to operate DRILS from any <.gov> or <.mil> Internet address to "carpe data" or, seize the data. One of the main goals of DRILS is to benefit those who enter the data. DRILS accomplishes this by providing streamlined and automated data entry that reduces maintenance documentation time while improving the data integrity. DRILS also provides historical analysis and other decision support tools that can be immediately used by the supervisor or technician.*

Supply chain managers (SCMs) work to continuously reduce the cost of sustaining their weapon systems through product and process improvements. The main cost-drivers must be prioritized to tackle the highest cost problems first, as not all problems can be solved simultaneously. To do that, the SCM must baseline the current repair activities using actual maintenance data collected at the point of maintenance by the repair technicians. The common repair technician complaint is that current U.S. Air Force legacy systems are too difficult to use for both inputting and retrieving data, and therefore, technicians do not regularly use them.

An F-16 SCM realized that he needed dependable and complete repair data to make and implement sound sustainable decisions. Manual data collection using spreadsheets started in a couple of avion-

ics repair shops to quantify known problems. This effort grew to a Microsoft Access database that was updated every four months. The repair technicians and shop management found the Microsoft Access database useful and wanted a real-time system that provided instant feedback.

A Web-based database was initiated using rapid prototyping, focusing on the technician inputting the repair data at the depot and in the field. Tracking the items being repaired by serial number, the warfighter in the field has visibility into the depot repair activities on those items and vice versa.

The remainder of this article will detail the challenges and how the Defense Repair Information Logistics System (DRILS) team meets or is planning to meet these challenges.

## Supply Chain Management Challenges

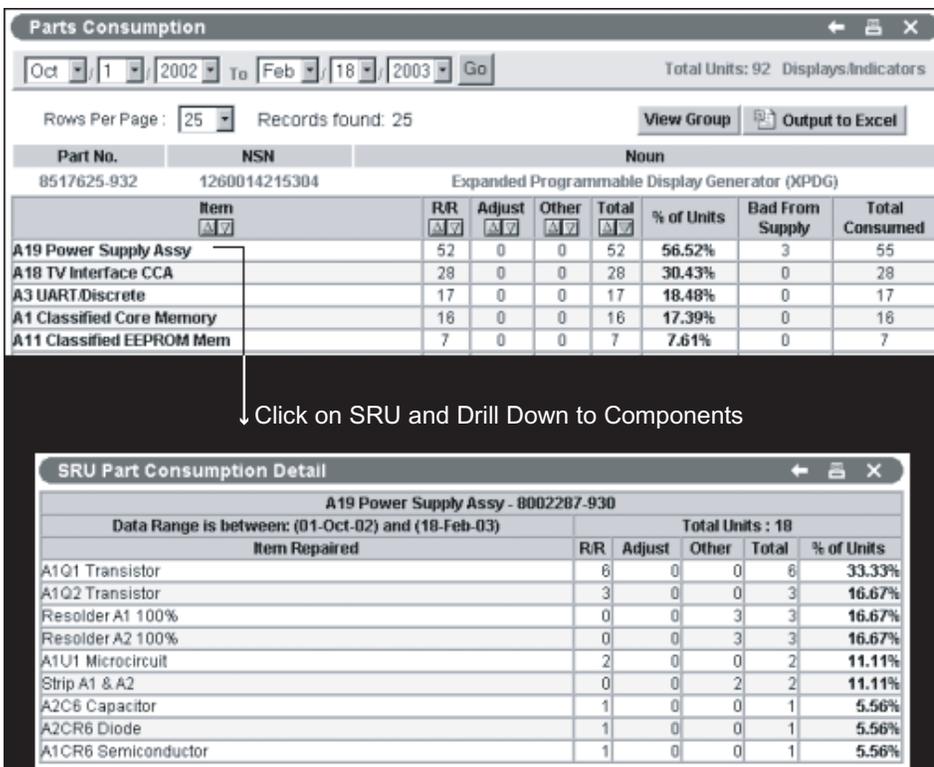
The Air Force SCMs are currently in need of a single dependable source of depot repair data. This became evident when the F-16 SCM at Ogden Air Logistics Center initiated the F-16 Flexible Sustainment program, otherwise known as Falcon Flex. This system is designed to analyze and attack high repair-cost drivers and determine the root causes of failures. These analyses are then used to drive performance-based acquisitions for hardware, software, and test equipment.

The SCM sought out part repair history like that shown in Figure 1 in order to feed the Falcon Flex analyses. Figure 1 is an example of DRILS parts consumption in a part number family sorted on the most replaced part. Various legacy data systems were queried to extract enough detailed repair data to perform an accurate analysis. The results of this very labor-intensive process were less than desirable, and the authenticity of the depot data that was available was suspect. Individuals in SCM organizations typically work around this issue by developing personal knowledge bases that are stored only in the brains and spreadsheets of program managers, item managers, material managers, engineers, etc.

Further research led the SCM to the shop floor to research technicians' personal log books and any other source of repair data at the depot repair facilities. Manually documented maintenance actions were also found on the work control documents used by the technicians, but were only saved for one year. Data were entered into the legacy system if the technician was willing and the system was available. However, data were difficult to enter and to retrieve using the user interfaces.

For example, it is well known that each technician will document the same maintenance action in different ways

Figure 1: Repair Parts Consumption



(e.g., R/R A2, R2 A2, Replaced A2, etc.). The SCM hired contractor labor to independently harvest these data and store them first in spreadsheets, then in a Microsoft Access database. These data were then *scrubbed* to group all similar maintenance actions together using a consistent notation and published quarterly to the SCM. This database was the precursor to the Web-based DRILS application.

Currently, DRILS data are being used by the F-16 SCM's Falcon Flex program to find high cost-drivers such as the Expanded Programmable Display Generator. The F-16 SCM has achieved \$43 million in cost avoidance as of January 2003 and a projected \$822 million in cost avoidance through 2020.

## Depot Maintenance Data

Challenges to supply chain management's ability to reduce total ownership costs are largely caused by the lack of dependable repair data, which is largely driven by a general distrust of the current legacy maintenance data systems. Antiquated user interfaces increase repair costs by requiring more time to input and retrieve data.

Another challenge is that depot repair facilities are inundated with many different computer systems performing similar functions that do not often interface with each other. This often results in shop-floor technicians being required to log into multiple applications with multiple user identifications and passwords. These applications often require duplicate data entry and cause frustration and errors between systems. Also, the designs of these systems often do not match the business process of the repair facilities, thus requiring the technicians to adapt their processes to the system.

In one example, the system software prohibited the technician from performing work on multiple items simultaneously. This forced the technician into a less efficient mode of serial production in order to accommodate the required software. The technician sees data entry time as impeding his or her primary goal, which is to produce serviceable units. What the depot technician really needs and wants is a flexible, single-user interface to quickly and easily enter all data documentation about the repaired unit.

Several initiatives by Air Force leadership to deploy obsolete information technology requiring extensive manual data entry with no additional worker compensation have died quick deaths. Thus, maintenance data documentation is regarded as

**Units Currently In Shop**

Rows Per Page: All Records found: 28

Nomenclature	Part No	Prod No	Total	INW	AWM	AWP	SERV	UNSERV	Notes
Azimuth Indicator	4030132-902	T7731N	0	0	0	0	2	0	(0)
DO HUD Display Unit (Blk 40)	79-081-13-02C	46589C	1	0	0	1	0	0	(0)
DO HUD Display Unit - NVIS (Blk 40)	179-081-13-02C	52559A	8	0	7	1	5	0	(0)
DO HUD Electronics Unit (Blk 40)	51-042-02-96A	55187H	4	0	4	0	9	0	(0)
Expanded Programmable Display Generator (XPDG)	8517625-932	59235A	43	0	42	1	8	0	(1)
Expanded Programmable Display Generator (XPDG)	8517625-931	59235A	8	0	5	1	1	0	(0)

Click to show serial numbers

**Displays/Indicators Units In-Shop List**

Rows Per Page: All Records found: 9

Nomenclature: Expanded Programmable Display Generator (XPDG) Part No: 8517625-931

Serial No	Status	Received	History	Discrepancy	Admin	Print Label
07187C0289	AWM on 01-Apr-03	01-Apr-03	PN SN SH	UNKNOWN		
07187C0319	AWM on 08-Apr-03	08-Apr-03	PN SN SH	UNKNOWN		
07187C0202	AWM on 14-Apr-03	14-Apr-03	PN SN SH	MFD 039, FAILED PROCESSOR AND VIDEO		
07187C0265	AWM on 14-Apr-03	14-Apr-03	PN SN SH	MFD'S BLANK		

Click to show in-work clock

Accumulated Hours			
AWM Hours	INW Hours	AWP Hours	Total Hours
368.78	0.00	0.00	368.78

Date	Status	Person	Hours
2003-04-16 16:14:10	In Work	Smullin, Fred	0.00
2003-04-01 07:26:28	AWM	Roeseler, Jo	368.78
2003-04-01 07:26:28	In Processed	Roeseler, Jo	0.00

Figure 2: Placing a Serial Number End Item in Work

an expense that can be easily eliminated from depot repair costs. However, by declining to fund maintenance documentation as the repairs are being performed, SCMs are setting themselves up for a larger expense later when a study must be contracted to determine why the repair facility is having problems with certain parts. It is the old *pay now or pay later* lesson with the latter needlessly expending many more times the cost of continuous maintenance data documentation.

We live in a data-rich environment, yet we are information poor. Today's war-fighting requirements are far different than they were 20 years ago when legacy data systems were initially designed and deployed. The fiscal year 1996-2001 "Defense Planning Guidance" states the following:

In order to support increasing functional requirements for information while implementing overall reductions in the budget, the Department must accelerate the pace at which it selects and deploys migration systems, identifies standard data, and conducts business process re-engineering across all functions. [1]

DRILS provides a Web-based, streamlined, user-friendly maintenance data documentation application for technicians at the point of maintenance. An example of how a technician can place an item into work is shown in Figure 2. By using the computer mouse to *point and click*, the part number family is selected from the "Units Currently in Shop" screen, which displays all the items currently inducted into the shop. The technician then selects the serial number of the selected end item and it is automatically placed into work thus initiating the in-work time clock, which tracks the labor time expended on that particular serial number.

DRILS also facilitates the collection and reporting of serialized maintenance data to Air Force worldwide data consumers for decision support thus filling a gap in legacy systems. An example of repair history for a serially tracked Line Replaceable Unit (LRU) is shown in Figure 3 (see page 18).

Specifically, DRILS provides much needed information and decision support both at the point of maintenance and in the SCM's organization. The technician at the point of maintenance is the only source that we have of obtaining this data.

Serial Number History						
Nomenclature			Part No	Serial No		
Expanded Programmable Display Generator (XPDG)			8517625-931	07187C0289		
Serial Number Summary						
Item	R/R	Adjust	Other	Total	% of Repair Actions	
A12/A13, Timer	1	0	0	1	33.33%	
A19, LV Power Supply	1	0	0	1	33.33%	
A6, Dual PT Memory	1	0	0	1	33.33%	
8517625-931						
Date	JCN	CND	Discrepancy	Corrective Action	Notes	Del
In Work LRPAG	023520210	No	BOTH MFD'S BLANK			
8517625-931						
Date	JCN	CND	Discrepancy	Corrective Action	Notes	Del
29-Mar-01 LRPAG	010120183	Yes	CAME IN AS A QDR, RAN GOOD, SENT OUT IN "A" CONDITION.	CND, CLEANED CONTACTS ON CARDS.		
8517625-931						
Date	JCN	CND	Discrepancy	Corrective Action	Notes	Del
25-Aug-99 LRPAG	991972605	No				
	Action	Reference Symbol	Serial Out	Part In	Serial In	Del
	R/R	A19, LV Power Supply	0479	8002287-930	2964	

Figure 3: Repair History of a Serial Numbered LRU

Both shop technicians and managers can use the Shop Status features as demonstrated in Figure 4.

### Warfighter Support

Today's warfighter does not have ready access to serialized depot repair data that could be used to diagnose faults in fielded weapon systems. The only option is to test the failed parts and then send them to the depot when they cannot be fixed locally. Once the parts are sent to the depot, the warfighter loses visibility of them and never receives feedback about the field diagnosis. This potentially results in wasted resources and an inability to investigate problems with training or equipment. The fighter wing's intermediate shops attempt to solve this problem by keeping detailed logs or local databases for trend analysis and bad actor identification.

To directly support the warfighter in the field, DRILS provides a fighter wing version of DRILS (using the same Web server as the depot version) as shown in Figure 5. Fighter wing intermediate shops can track items sent to the depot, and depot technicians can view the repair activities performed on the serialized item in the field. The entire repair history is

available to both the fighter wing intermediate shop and depot personnel. The warfighter version is being used currently by the 388th and 419th Fighter Wings.

### Legacy System Modernization

The military continues to attempt information system modernization with an emphasis on integrating commercial off-the-shelf (COTS) enterprise inventory and resource-scheduling products that promise shorter development and deployment time while improving supportability. However, the method in which the military performs and documents maintenance is quite different from anything currently in the commercial market. Thus, there are no readily available COTS solutions that can be employed without requiring massive changes to military maintenance documentation policy or to the COTS product.

Each previous attempt has failed because of the lack of focus to provide benefits to the technician. Each has attempted to *swallow the elephant whole* and provide a standardized, enterprise-wide solution where *one size fits all* yet the product satisfies no one. The repair activities vary from shop to shop, base to base, and

Figure 4: Shop Status

Displays/Indicators Units In-Shop										
Units You Have In Work										Shop: Displays/Indicators
Nomenclature	Part No	Prod No	Serial No	Date In Work	Review					
DO HUD Display Unit (Blk 40)	79-081-13-02C	46589C	K0656C0079	26/03/2003 20:04	REVIEW					
Expanded Programmable Display Generator (XPDG)	8517625-932	59235A	07187C1247	16/04/2003 22:40	REVIEW					
Expanded Programmable Display Generator (XPDG)	8517625-931	59235A	07187C0289	16/04/2003 22:21	REVIEW					
Units Currently In Shop										
Rows Per Page: All		Records found: 28		Output to Excel						
Nomenclature	Part No	Prod No	Total	INW	AWM	AWP	SERV	UNSERV	Notes	
Azimuth Indicator	4030132-902	T7731N	2	0	2	0	1	0	(0)	
DO HUD Display Unit (Blk 40)	79-081-13-02C	46589C	12	2	9	1	1	0	(0)	
DO HUD Display Unit - NMIS (Blk 40)	179-081-13-02C	52559A	12	1	11	0	2	1	(0)	

between military services.

The length of time it takes to develop this type of solution is counter-productive to the overall military mission. A development life cycle from concept to enterprise deployment historically has taken 10 years or more, ensuring that the systems are obsolete when they are deployed. Potential users are required to forecast requirements 10 years out and *compete* their requirements with other enterprise users. The military cannot wait 10 or more years for new *big-bang* information technology solutions. It is also not practical to discard many years and dollars invested in current legacy information systems when they can still be utilized.

Current legacy systems that house maintenance data such as the Enterprise Data Warehouse perform a very valid function. Their primary mission is to provide the capacity and processing power to store terabytes of maintenance and other data collected from worldwide sources. Their secondary mission is to provide weapon systems and enterprise-level decision tools to data consumers. However, their breadth and scope of mission do not allow them to be very responsive to point-of-maintenance changes and technology advances.

### DRILS Software Prototyping Process

As can be seen by the previous discussion, there are many challenges to collecting, storing, retrieving, and using maintenance data. To solve these challenges, DRILS used rapid joint application prototyping and evolutionary development techniques to elicit user interface requirements from shop-floor users and data consumers. The purpose of using this methodology was to field initial capabilities and quick refinements to all users faster than using traditional software engineering approaches.

The DRILS development team was able to learn and implement the users' likes, dislikes, and their processes, earning their trust early in the prototyping phase. The development team solicited requirements from users over several iterations of the software. Past mistakes such as developing a system without user involvement and *dumping* it on them, were avoided. The results of this prototyping are being used to complete a software requirements specification defining what is currently thought to be close to a *mature* application.

DRILS uses Macromedia's Cold Fusion MX as the front end while storing and retrieving data using Oracle. A Web-

based application was selected so that only the server software would need to be updated in one physical location. Since most people are familiar with Web-based applications and Web browsers, training time is focused on the work processes and maintenance data collection.

During user development and testing, defects were collected and the appropriate developer notified automatically by Cold Fusion as users exposed them. Defects that were discovered during testing prior to release were also collected and measured. The average time to fix a minor defect was about one hour.

### Future DRILS Enhancements

Future development and expansion into a more widely used product will be done with tighter process controls. A configuration control board will also be organized to evaluate and prioritize new and changed requirements as well as major defect fixes to form block releases for implementation and deployment. Releases will be coordinated to occur as often as feasible depending on the number, size, and complexity of the changes.

The Software Engineering Institute's Team Software Process<sup>SM</sup> (TSP<sup>SM</sup>) will be used to plan and track block upgrades and user-requested changes, and expansion to other organizations. Closer tracking of TSP software development metrics during future development phases and testing will provide more accurate estimates and on-time deliveries of more robust DRILS application releases.

Support costs are a concern of many users as DRILS is expanded into more shops. Computer-based training, online help, frequently-asked-question pages, DRILS usability feedback, and experimentation with training techniques are all being considered to help reduce the overall support costs without sacrificing data integrity and completeness.

### Lessons Learned

The following are just a few of the lessons learned so far:

- You must work directly with customers to understand and implement their processes. Design and implement applications to actually help users satisfy the *what's in it for me* yearning.
- Rapid prototyping works well to determine exactly what works and how well. Interface screens were developed *overnight* to incorporate the

Nomenclature	Part No	# Recvd	# NRTS	# Serv	# CND	% CND	# MICAP	# TCTO
Antenna (C/D,800)	758R800G01	3	3	0	0	0.00	0	0
AUDIO 1 (-3)	16F4419-3	1	0	1	0	0.00	0	0
Countermeasures Switching Unit	179250-0003	1	0	1	0	0.00	0	1
Data Transfer Unit (DTU)	7461000002	1	0	1	1	100.00	0	0
Data Transfer Unit (Night Vision)	7461000007	2	2	0	0	0.00	0	0
Digital Flight Control Computer (DFLCC)	3757528-1	8	3	5	4	50.00	1	0
DO HUD Display Unit (Blk 40)	79-081-13-02C	6	4	2	1	16.67	2	0
DO HUD Electronics Unit (Blk 40)	51-042-02-95A	2	0	2	1	50.00	0	1
DO HUD Electronics Unit (Blk 40)	51-042-02-96A	1	0	1	0	0.00	0	0
DO HUD Electronics Unit (Blk 40)	51-042-02-95B	2	0	2	1	50.00	1	1
Dual Mode Transmitter (DMT)	758R990G01	6	5	1	1	16.67	0	0
Enhanced Central Interface Unit (ECIU)	16E10080-827	1	0	1	0	0.00	0	0
Enhanced Central Interface Unit (ECIU)	16E10080-829	1	1	0	0	0.00	0	0
EXDEEU (-805)	16E10090-805	2	1	1	1	50.00	2	0
Expanded Programmable Display Generator (XPDG)	8517625-931	1	0	1	0	0.00	0	0

Figure 5: *Fighter Wing (Warfighter) View of DRILS*

- process owners' ideas and improvements.
- When a prototype *works*, most people think of it as production quality and want all the *documentation* that a software product normally contains. DRILS software documentation was delayed in order to focus on core functionality and easy-to-use interfaces.
- Design applications so that they can be easily modified to conform to process improvements.
- Solve current problems while architecting to include future requirements.
- Data must be captured at the point of maintenance and not estimated or *sampled*.

### Future Activities

Future activities include the following:

- Expand to 20th Fighter Wing Component Maintenance Squadron shop at Shaw Air Force Base.
- Participate in the U.S. Air Force serial number tracking effort.
- Continue to expand in Ogden-Air Logistics Center/avionics repair shops, Tobyhanna Army Depot, and Support Center Pacific.
- Expand to Ogden-Air Logistics Center airborne generator shop.
- Interface with Inventory Tracking System, Material Planning System, Core Automated Maintenance System, and Integrated Maintenance Data System.

- Participate in the Shop Service Center process improvement effort.

### Conclusion

The warfighter, depot repair facility, and SCM all need dependable part consumption and serialized repair history data. They need an agile, easy-to-use, maintenance data documentation solution that will facilitate and not hinder the collection of dependable serialized repair data at the point of maintenance.

Repair technicians both in the field and at the depot need a solution that aids rather than impedes their daily tasks. The solution should be beneficial to the repair technicians and the organizations that need to access the data.

The SCM needs dependable part consumption data to accurately forecast future part needs as well as identify those areas where efforts are likely to provide the greatest return on investment. DRILS fulfills these *needs* and is a key enabler for future supply chain process and product improvements. DRILS is a government-owned maintenance documentation system that is being developed by Total Quality Systems, Inc. For more information about DRILS, please contact the authors.♦

### Reference

1. Department of Defense. Defense Planning Guidance. Fiscal Year 1996-2001. Sec. III, Subsection F, Paragraph 4.

<sup>SM</sup> Team Software Process and TSP are service marks of Carnegie Mellon University.

## COMING EVENTS

**November 2-5**

*Amplifying Your Effectiveness Conference*  
Phoenix, AZ  
[www.ayeconference.com](http://www.ayeconference.com)

**November 5-7**

*Association for Computing Machinery  
SenSys '03*  
Los Angeles, CA  
[www.cens.ucla.edu/sensys03](http://www.cens.ucla.edu/sensys03)

**November 10-12**

*PDF Conference: Expanding the  
PDF Frontier*  
Anaheim, CA  
[www.pdfconference.com](http://www.pdfconference.com)

**November 15-21**

*Supercomputing Conference 2003  
Igniting Innovation Conference*  
Phoenix, AZ  
[www.sc-conference.org/sc2003](http://www.sc-conference.org/sc2003)

**November 17-20**

*4th Annual National Guard Bureau  
IT Conference*  
Las Vegas, NV  
[www.technologyforums.com/ngb](http://www.technologyforums.com/ngb)

**November 17-20**

*14th IEEE International Symposium on  
Software Reliability Engineering  
ISSRE 2003*  
Denver, CO  
<http://salieri.cs.colostate.edu:8000/>

**November 17-21**

*2nd International Conference on  
Software Process Improvement*  
Washington, DC  
[www.icspi.com](http://www.icspi.com)

**March 30-31, 2004**

*3rd Annual Southeastern Software  
Engineering Conference*  
Huntsville, AL  
[www.ndia-tvc.org/SESEC](http://www.ndia-tvc.org/SESEC)

**April 19-22, 2004**

*2004 Software Technology Conference*



Salt Lake City, UT  
[www.stc-online.org](http://www.stc-online.org)

## About the Authors



**Capt. Greg Lindsey** is the Defense Repair Information Logistics System program manager and is currently stationed at Hill Air Force Base, Utah. He began his Air Force career as an F-4 Tactical Aircraft Maintenance technician and progressed to KC-135 Maintenance officer. He was selected to participate in the Operational Exchange Program and was assigned to Wright-Patterson Air Force Base, Ohio, where he led an Integrated Logistics Acquisition Team at the F-22 System Program Office. He is a graduate of the University of Central Florida.

**OO-ALC/MAC**  
6090 Gum Lane  
Hill AFB, UT 84056  
Phone: (801) 586-6634  
DSN: 586-6634  
Fax: (801) 777-8159  
E-mail [greg.lindsey@hill.af.mil](mailto:greg.lindsey@hill.af.mil)



**Kevin Berk** is the Defense Repair Information Logistics System program manager for Total Quality Systems, Inc., Ogden, Utah. He has a wide range of experience in program management, managing software development, and process improvement in both the public and private sectors. He has a Bachelor of Science in physics from the University of Central Florida, a Bachelor of Science in electrical engineering from Michigan State University, and a Master of Science in computer engineering from the Air Force Institute of Technology.

**Total Quality Systems, Inc.**  
1990 West 2550 South  
Ogden, UT 84401  
Phone: (801) 731-2150  
Fax: (801) 731-4457  
E-mail: [kberk@tqsinc.com](mailto:kberk@tqsinc.com)

## WEB SITES

### American Society for Information Science and Technology

[www.asis.org](http://www.asis.org)  
Since 1937, the American Society for Information Science and Technology has been leading the search for new and better theories, techniques, and technologies to improve access to information. Membership includes some 4,000 information specialists from such fields as computer science, linguistics, management, librarianship, engineering, law, medicine, chemistry, and education.

### The Data Management Association International

[www.dama.org](http://www.dama.org)  
The Data Management Association (DAMA) International is a not-for-profit, vendor-independent association of technical and business professionals dedicated to advancing the concepts and practices of information resource management and data resource management. DAMA International's primary purpose is to promote the understanding, development and practice of managing information and data as a key enterprise asset.

### Boxes and Arrows

[www.boxesandarrows.com](http://www.boxesandarrows.com)  
Boxes and Arrows is an online journal dedicated to understanding the design of the architecture and structure of digital spaces, and often features articles on the craft of information architecture. Boxes and Arrows is a peer-written journal featuring the sharing of exemplary technique, innovation, and informed opinion of this field. The journal strives to provoke thinking among peers, to push the limits of the accepted boundaries of these practices, and to challenge the status quo by teaching new or better techniques that translate into results.

### DoD 5000 Series Resource Center

<http://dod5000.dau.mil>  
The DoD 5000 Series Resource Center provides a complete package of information on the DoD 5000 Series documents, including the official directive, operation of the defense acquisition system, interim defense acquisition guidebook, terminology, a defense acquisition tutorial, frequently asked questions, and more.



# The Documentation Diet

Neil Potter and Mary Sakry  
*The Process Group*

*We hate to do project and process documentation and are upset when it only seems like irrelevant paperwork that is done to please some external party. However, when we do not have any documentation, we can feel left in the dark because of a lack of direction and clear communication among colleagues. If your project is experiencing either too much or too little documentation, or your company improvement program is really no more than a documentation exercise, there is an effective middle ground; one that allows you to be productive in your projects, but not taxed with irrelevancy. This article discusses strategies you can use to make documentation work for you. The examples contain a mix of project and process documentation scenarios.*

During the 15 years we have been observing companies and helping them improve, a common cause of irrelevant or overwhelming stacks of paper has been a lack of purpose or objective for each document. When we ask the average project team member why he or she has so much documentation, a common answer is, “Because my organization requires me to fill out the templates.” If we ask about the usefulness of such documentation, a typical reply is, “I guess it will provide a trail of what has happened so that management can study my project later.”

In hindsight, few people ever go back and plow through *the stack*. With purposes as unclear as these, it is not surprising that people merely *fill out the templates*. What is the purpose of documentation? Here are two examples:

- Project documentation is a method of capturing and sharing critical project concepts, plans, and information as they are developed so that impacted parties can share this information, make informed decisions, and move the project forward.
- Process documentation is a method of capturing and sharing engineering and management practices so that an organization can remember, reuse, and refine its skills and avoid reinventing lessons learned and best practices for each new project. Process documentation can include templates, procedures, and checklists. Note that we did not include in our definition “evidence to please managers or auditors.” Defining the information required to manage a project effectively creates *natural* documents that provide ample evidence of certain practices occurring. For example, if we plan a project correctly and capture the details so they can be communicated to others, the natural document that results (the

plan) should be ample evidence that planning took place. Evidence is free when good practices are followed.

## Strategies for Making Documentation Practical

There are several techniques to make documentation concise and practical. They include the following:

---

**“Defining the information required to manage a project effectively creates natural documents that provide ample evidence of certain practices occurring ... Evidence is free when good practices are followed.”**

---

- Focus documentation on the organization’s needs.
- Merge duplicate work products.
- Remove redundancy in templates.
- Simplify “best practice” (process) documentation.
- Consider one representation.
- Consider one page per process or sub-process.
- Merge documented procedures and related work product templates.
- Use process descriptions as audit checklists.

The following sections describe these techniques in greater detail.

## Focus Documentation on the Organization’s Needs

Answer the following questions to determine the organization’s needs for each document:

- What goal are you trying to achieve; what role does this document play with respect to this goal?
- What problem are you trying to solve with this document?

These questions cause you to focus on the specific purpose of each document; your responses scope the document and provide you with an end point.

In one software development group, 50 percent of each requirements document contained information describing how the product was going to be built instead of focusing on what the product was going to do for the end user. The lack of a clear goal allowed the specification to become a *catchall* document with no end point. The following is an example of a goal for a requirements document:

Capture the needs of our customers by defining the tasks they need to perform and expectations they must have met in the solution we deliver (e.g., performance and reliability targets).

An example of a requirements template that captures user needs and other expectations is shown in Figure 1 (see page 22) [1].

## Merge Duplicate Work Products

When project documents contain similar information and there is little benefit in keeping them separate, consider merging them. For example, if there are three documents to complete – *Statement of Work*, *Product Requirements*, and *Contractual Requirements* – and each will contain the same information, consider one docu-

Requirements Template	
Instructions for Template Use ~~~~~ ~~~~~ ~~~~~	4. External Interface Requirements 4.1 User Interfaces 4.2 Hardware Interfaces 4.3 Software Interfaces 4.4 Communications Interfaces  5. Use Cases (UC) 5.1 UC Name 1 5.1.1 UC Details 5.1.2 Business Rules For UC 1 5.1.3 Functional Requirements for UC 1 5.x UC Name x  6. Other Nonfunctional Requirements 6.1 Performance and Reliability Requirements 6.2 Business Rules (Global)  Appendix A: Glossary Appendix B: To Be Determined List
1. Introduction 1.1 Purpose 1.2 Intended Audience  2. Business Requirements 2.1 Business Requirements 1.N 2.2 Product Scope (Context Diagram)  3. Overall Description 3.1 User Classes and Characteristics 3.2 Operating Environment 3.3 Assumptions and Dependencies	

Note: Adapted from IEEE Standard 830-1998

Figure 1: An Example Requirements Template

ment. In the document, cross-reference the other two templates that this document satisfies. If there are differences in the three documents but considerable overlap, write one set of requirements and label those items that are *Statement of Work* as deliverables, and those that are *Contractual* as requirements.

If you are using an improvement framework such as the Software Engineering Institute's (SEI) Capability Maturity Model® (CMM®), consider merging work products together to implement specific practices. For example, a *Software Configuration Management* (SCM) plan, *Software Quality Assurance* (SQA) plan and *Software Development Plan* (SDP) can be merged. Milestones and activities for SCM and SQA might be listed on the master schedule in the SDP.

If your project is required to write maintenance documentation so future teams can understand the program's internal workings, consider the overlap between this manual and the other work

products that have been created. A current design document already states the architecture, data flow, algorithms, methods and interfaces, so there should be no need to duplicate this information in a maintenance manual.

A current requirements document states the functionality and behavioral characteristics of the product, so the maintenance manual should not need to repeat this information either. Determine what unique information is needed for a maintenance person and scope the document to just this need. Cross reference existing work products to make the maintenance manual complete.

**Remove Redundancy in Templates**

Closely examine sections within each template that are redundant. The template might have looked sound when first created, but during use you might find that some of the sections contain the same information. Each use of the template is an opportunity to put it on a diet. For

example, the requirements template on the left in Table 1 can be slimmed down to the template on the right in Table 1 when it is realized that everything said in sections 1 and 4 have already been said in sections 2 and 3.

Figure 2 describes an example of a design template. In this example, items 5.6 and 5.7 are potentially overlapping and could be merged.

**Simplify Best Practice (Process) Documentation**

Process documents can also suffer from a lack of purpose clarity. For example, suppose you are on the Best Practices Definition committee, or your group is using the SEI CMM, and you have been chartered to develop a process for creating project schedules [2]. You might be tempted to build the world's greatest and most comprehensive schedule creation process, with all known bells and whistles. The document could restate the CMM text, include references to numerous books on the subject, and refer to Critical Chain Analysis (whatever that is!). The appendix could include three pages of cross-references to other models and standards.

Alternatively, ask the first question, "What goal are you trying to achieve; what role does this document play with respect to this goal?" For example, the goal could be to determine which product features could be completed by the established delivery deadline given the available resources. This process describes how to develop a schedule to help achieve that goal.

The second question, "What problem are you trying to solve with this document?" enumerates needs for the document to address. An example of a need is to prevent your project from chronically over-committing, causing financial loss to the company. Now write a small process to accomplish these two items. An example is shown in Table 2.

When do you stop defining this process? Stop when your goal has been achieved (e.g., scoping the project) and your problem solved (e.g., avoiding over-commitment). Refine the document further when it no longer meets the need.

**Consider One Representation**

Write processes using one representation. For example, if you are creating a process for risk management, it would be redundant to have one file of presentation slides, the same process formatted using a word processor, a version in HTML for

Table 1: Redundant Template Sections Are Removed

Requirements Specifications	Revised Requirements Specifications
1. Product Objectives 2. Business Requirements 3. Product Advantages 4. Value Proposition	1. Business Requirements 2. Product Advantages

Figure 2: Example Design Template

Design Template	
Instructions for Template Use ~~~~~ ~~~~~ ~~~~~	4.0 Software Architecture 5.0 Module Attributes 5.1 Module purpose 5.2 Performance requirements 5.3 Communication protocol 5.4 User interface (if applicable) 5.5 Local data structures 5.6 Algorithm 5.7 Flow diagram 5.8 Error handling
1.0 Purpose 2.0 Global Data Dictionary 3.0 Design Method 3.1 Function-oriented design 3.2 Data-oriented design 3.3 Real-time control-oriented design 3.4 Object-oriented design	

\* Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

browsing, and the same information again using a flow diagramming tool.

Instead, determine how the process document will be used (e.g., online use by developers during project execution, or in a classroom setting with 100 people being trained). Then consider one representation that can suit all needs. For example, a presentation slide format can be printed for reading, e-mailed for sharing, presented for teaching, and uploaded for browsing.

**Consider One Page Per Process or Sub-Process**

There are approximately 60 lines on a page and 10 words per line. That is quite a lot of information. So consider keeping process documentation to one or two pages (at least at the beginning).

Processes can be kept to one or two pages by limiting how much detail you allow yourself to write. Unless you plan on writing forever, you have to put some limit on the document, so start with one page. When you are tempted to add more explanation and detail, refine what you have defined; do not necessarily add more sections.

**Merge Documented Procedures and Related Work Product Templates**

Organizations using process improvement frameworks such as the SEI CMM and ISO 9001 might be tempted to write procedures *because the framework states that they are needed*. Creating a template to assist the procedure user (for example, a template for an SCM, SQA, or project plan) often follows procedure creation. Creating both a procedure and template can lead to redundancy. An alternative approach is to embed the instructions for completing a template in the template itself. The procedure and the template are the same document. For example, the CMM practice “Create an SCM plan according to a documented procedure,” can be implemented by developing a lightweight template with embedded instructions for use (see Figure 3).

**Use Process Descriptions as Audit Checklists**

If you have a process assurance function that audits projects for process compliance, use the process descriptions that the projects use; do not write a separate audit checklist. Write processes (for example, estimation, schedule creation, and change control) in a style that can be used for both project and audit purposes. It might be necessary to provide auditors with some additional guidance in conducting

Schedule Creation Process to Scope a Project and Avoid Financial Loss Due to Over-Commitment
<ol style="list-style-type: none"> <li>1. Determine project tasks.</li> <li>2. Determine project task dependencies:                             <ol style="list-style-type: none"> <li>a. For each pair of tasks (A+B), ask, "Must task A complete before task B starts, or can both tasks execute in parallel?"</li> <li>b. Draw dependency between tasks.</li> </ol> </li> <li>3. Add effort estimates for each task (uninterrupted time).</li> <li>4. Add resources to each task (people, equipment, resource assumptions).</li> <li>5. Add resource availability, i.e.:                             <ol style="list-style-type: none"> <li>a. Planned percentage each resource will be allocated.</li> <li>b. The dates each resource is available.</li> </ol> </li> <li>6. Overlay desired project completion deadline. If the deadline is impossible, ask:                             <ol style="list-style-type: none"> <li>a. What features fit within the deadline? Is this a satisfactory list?</li> <li>b. What options are available to achieve the deadline (e.g., make/buy tradeoffs, adding resources to the critical path, simplifying features, subcontracting work out, reusing existing code).</li> <li>c. What features should be demoted for later release?</li> </ol> </li> <li>7. Present data, schedule options, and risks to management and the customer. Agree on a schedule that has acceptable customer satisfaction and acceptable risk of failure.</li> </ol>

Table 2: A Schedule Creation Process

the audit and reporting the results. It is unnecessary to duplicate the same process information in a different format.

**Knowing When You Are in Trouble**

An organization is in *document trouble* when project team members create documents that have little use or value. This can occur when either the team members are

**“An organization is in document trouble when project team members create documents that have little use or value.”**

unclear about a document’s purpose or when a document is created to satisfy the needs of an external auditor or assessor.

In the first scenario, a committee is typically formed to define a specific phase of the software life cycle. The template is the committee’s deliverable. The template is successfully used on a few projects and is then made standard operating procedure. When the template contains more sections than needed, and when the larg-

er audience is not trained in the template’s purpose, too many project teams fill out the template with redundant information. At this point, the resulting document can be viewed as unnecessary.

In the second scenario, project team members believe they have to create additional documentation to prove to an external auditor or assessor that the project is being managed correctly. Memos capturing meeting discussions, statements-of-work documents summarizing product requirements, and design documents that are created after the code has shipped are produced to *pass the audit*. The team views documentation as an activity unrelated to building the product. It is a *keep management happy* tax.

In this second scenario, the cause can be due to poorly trained auditors who look for *paperwork* but do not really understand the fundamental engineering or management practices that are desired of the project teams. For example, the auditor looks for a statement-of-work document even though a detailed set of requirements exists that covers the same information. Minutes of meetings are examined even though the project is six months behind and none of the corrective actions during those six months have been implemented. Here, the auditor needs education, and the documents required of the team need tailoring.

*Pleasing an auditor* can also occur when the project team members have not ana-

Figure 3: Combined SCM Plan Creation Procedure and Template

SCM Plan Template	Instructions
Step 1: List Configuration Items - x, y, z	~~~~~ ~~~~~
Step 2: Establish File Naming Conventions - File-x<n>.doc	~~~~~ ~~~~~
Step 3: Establish Baseline File Structure - ~~~~~	~~~~~ ~~~~~



## Get Your Free Subscription

Fill out and send us this form.

**OO-ALC/MASE**

**6022 FIR AVE.**

**BLDG. 1238**

**HILL AFB, UT 84056-5820**

**FAX: (801) 777-8069 DSN: 777-8069**

**PHONE: (801) 775-5555 DSN: 775-5555**

Or request online at [www.stsc.hill.af.mil](http://www.stsc.hill.af.mil)

NAME: \_\_\_\_\_

RANK/GRADE: \_\_\_\_\_

POSITION/TITLE: \_\_\_\_\_

ORGANIZATION: \_\_\_\_\_

ADDRESS: \_\_\_\_\_  
\_\_\_\_\_

BASE/CITY: \_\_\_\_\_

STATE: \_\_\_\_\_ ZIP: \_\_\_\_\_

PHONE: (\_\_\_\_) \_\_\_\_\_

FAX: (\_\_\_\_) \_\_\_\_\_

E-MAIL: \_\_\_\_\_

**CHECK BOX(ES) TO REQUEST BACK ISSUES:**

**MAY2002**  **FORGING THE FUTURE OF DEF.**

**AUG2002**  **SOFTWARE ACQUISITION**

**SEP2002**  **TEAM SOFTWARE PROCESS**

**NOV2002**  **PUBLISHER'S CHOICE**

**DEC2002**  **YEAR OF ENG. AND SCI.**

**JAN2003**  **BACK TO BASICS**

**FEB2003**  **PROGRAMMING LANGUAGES**

**MAR2003**  **QUALITY IN SOFTWARE**

**APR2003**  **THE PEOPLE VARIABLE**

**MAY2003**  **STRATEGIES AND TECH.**

**JUNE2003**  **COMM. & MIL. APPS. MEET**

**JULY2003**  **TOP 5 PROJECTS**

**AUG2003**  **NETWORK-CENTRIC ARCHT.**

**SEPT2003**  **DEFECT MANAGEMENT**

**TO REQUEST BACK ISSUES ON TOPICS NOT**

**LISTED ABOVE, PLEASE CONTACT KAREN**

**RASMUSSEN AT <KAREN.RASMUSSEN@**

**HILL.AF.MIL>**

lyzed why an engineering or management practice (and its associated document) is required, and how they could benefit from it. In such cases, the team *reacts* to the process requirement without understanding why the requirement is there. Here, the team needs training on the purpose and correct use of each specific document. If the project is performing the required practices correctly, the natural documents produced should be essential for team operation and be adequate for any auditor or assessor. The goal is no *extra paperwork*.

## Summary

Software development is not about documentation. Software development is about creating solutions that help meet customer needs. Process improvement is not about documentation. Process improvement is about fixing critical problems in the organization and capturing the solutions for reuse and refinement.

Excess documentation is often the result of poor clarity of purpose and inadequate understanding of how each document should be used. When docu-

ments are written with a clear business goal and need in mind, they become important and useful. ♦

## References

1. Institute of Electrical and Electronics Engineers. Recommended Practice for Software Requirements Specifications. Piscataway, NJ: IEEE Computer Society Press, 1998 <<http://shop.ieee.org/store/product.asp?prodno=SH94654>>.
2. Software Engineering Institute's Capability Maturity Model (V1.1) Software Project Planning Activity 12.

## Additional Reading

1. Potter, N., and M. Sakry. Making Process Improvement Work – A Concise Action Guide for Software Managers and Practitioners. ISBN 0-201-77577-8. Reading, MA: Addison-Wesley, 2002.
2. Paulk, Mark C., Charles V. Weber, and Bill Curtis. The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA: Addison-Wesley, 1995.

## About the Authors



**Neil Potter** is co-founder of The Process Group, a software-engineering process improvement consultancy. He has 19 years of experience in software and process engineering. For six years, Potter was a software engineer at Texas Instruments (TI). For two years, he managed a TI Software Engineering Process Group spanning America, England, and India. Potter is a Software Engineering Institute-authorized assessor for CBA-IPI and SCAMPI<sup>SM</sup> appraisals and an "Intro to CMMI" instructor. He has a Bachelor of Science in computer science from the University of Essex (United Kingdom) and is co-author of "Making Process Improvement Work – A Concise Action Guide for Software Managers and Practitioners."

**The Process Group**

**P.O. Box 700012**

**Dallas, TX 75370-0012**

**Phone: (972) 418-9541**

**Fax: (972) 618-6283**

**E-mail: [help@processgroup.com](mailto:help@processgroup.com)**



**Mary Sakry** is co-founder of The Process Group, a software-engineering process improvement consultancy. She has 28 years of experience in software and process engineering. For 15 years she was a manager and software engineer at Texas Instruments (TI). In 1989, she led TI's worldwide process assessment effort and was a member of a Software Engineering Process Group. Sakry is a Software Engineering Institute-authorized assessor for CBA-IPI and SCAMPI<sup>SM</sup> appraisals and an "Intro to CMMI" instructor. She has Bachelor of Science in computer science and a master's degree in business administration. She is co-author of "Making Process Improvement Work – A Concise Action Guide for Software Managers and Practitioners."

**The Process Group**

**P.O. Box 700012**

**Dallas, TX 75370-0012**

**Phone: (972) 418-9541**

**Fax: (972) 618-6283**

**E-mail: [help@processgroup.com](mailto:help@processgroup.com)**



# Software Architecture as a Combination of Patterns

Kent Petersson and Tobias Persson  
*Ericsson Microwave Systems*

Dr. Bo I. Sanden  
*Colorado Technical University*

*Ericsson Microwave Systems in Sweden was confronted with the problem of constructing a radar system that could withstand the replacement of hardware and operating system software and be adaptable to different customers' functional requirements. This was accomplished by means of software architecture with highly independent and flexible components that is a combination of four design patterns: Layers, Pipes and Filters, Observer, and Model-View-Controller.*

The computer science community has introduced design patterns as a means of capturing and documenting solutions to common software problems [1, 2]. An important aspect is the combination of patterns. In "Design Patterns" [1], the authors discuss how patterns *dovetail and intertwine* in good software. Nevertheless, combining patterns into a good overall architecture has received less attention than the individual patterns. In much of the literature, each pattern is often presented separately along with a discussion of how it fits into a resulting class diagram. This assumes that the class diagram exists when you present your patterns.

In this article, we present not only how different patterns are used in software architecture, but also how we evolved the architecture by successively integrating more patterns to deal with particular design problems. We present the individual patterns, their interaction, and the questions that arise when they are combined.

The example we use is a software subsystem of a new generation of surveillance radar constructed by Ericsson Microwave Systems (EMW) in Mölndal, Sweden. This subsystem contains the modules that handle tracking, communication with external devices, threat evaluation, etc. A major problem with the software of an earlier radar system was that the modules were too interdependent. If you wanted to use only one part for a new product, you often had to include all modules even though the functionality was not needed.

When the new architecture was constructed, one premise was to maximize reuse. The main reason was the business situation for the surveillance radar system. The trend was that each customer bought a small series but still required considerable tailoring to his or her needs. To encourage reuse, the different components had to be made more independent and flexible. A function had to be coupled

to as few other components as possible, and direct dependencies had to be eliminated or minimized.

An important guideline in the architectural design was the requirement for modifiability. The following sources of modifications were especially considered:

1. Replacement of hardware and operating system.
2. Different customers' functional requirements.

---

***"A principle of the Layers pattern is that the components in each layer only know of and use components in lower layers ... the components in the application layer only know and use components in the support layer ..."***

---

In the following, we first discuss how the system was made adaptable to new hardware and software, and then how it was designed to accommodate changes in customer requirements in general and requirements for presentation and control in particular.

## **Adaptability to New Hardware and Operating System**

To facilitate future changes of hardware and operating system, a layered architecture was introduced. The system was structured in terms of components, which were placed in three layers with applica-

tion-oriented components at the top and hardware and operating system dependent components at the bottom. This is a time-honored architectural style with the Open Systems Interconnection protocol stack as a classic example [3].

## **The Layers Architectural Pattern**

The *Layers* pattern is described in [2]. In it, the system is structured as a number of layers that represent different levels of abstraction. We introduce the following three layers from top to bottom:

- The application layer containing the functionality required of the system.
- The support layer containing parts shared by the applications. An important role for this layer is to act as data storage for the applications. For maximum independence between applications, their input and output data is stored in the support layer. That way, the applications need not be directly aware of each other but instead depend on the database components in the support layer.
- The core layer containing components that depend on the operating system or the hardware.

A principle of the Layers pattern is that the components in each layer only know of and use components in lower layers. In our case, this means that the components in the application layer only know and use components in the support layer (and possibly the core layer). This principle automatically makes the components in the application layer independent, but sometimes forces you to introduce additional layers, potentially complicating the solution. For simplicity, it is sometimes reasonable to compromise and let some components in a layer know of each other. Still, a component should never need to know of a component in a higher layer.

## **Architectural Components**

The layered architecture is shown in

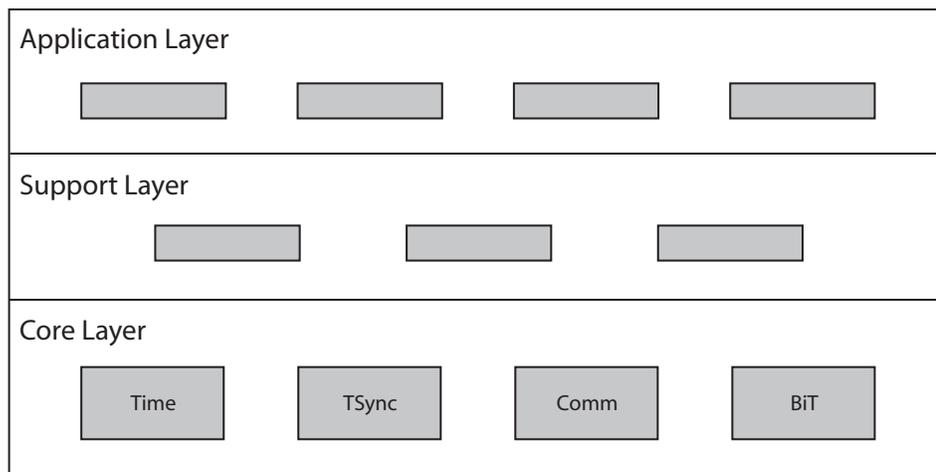


Figure 1: Layered Architecture With Some of the Core Layer Components

Figure 1. The following are some of the components in the Core layer:

- System Time (*Time*). This component allows the system to run at a user-defined time in a simulation mode. With multiple instances of this component, real time and simulated time can be used concurrently.
- Time Synchronization (*TSync*). With different parts of the system running on different hardware nodes, synchronizing the system time on each node is critical.
- Communication Protocols (*Comm*). This component includes components dealing with low-level communication protocols tailored for specific applications.
- Built-in Test (*BiT*). These modules handle hardware testing.

### Adaptation to Customer Requirements

Customers have differing functional requirements for at least two reasons. First, different customers need more or less functionality depending on how they intend to use the radar system in their overall system structure. Some want a basic system while others want a deluxe version.

Second, customers must integrate the system they buy into a larger system whose interfaces vary considerably. This has led to a lot of reconstruction of the radar system over the years. The interface to the human operators is particularly variable.

To solve the problem with differing customer requirements, we designed architecture for a complete system with

full functionality. It defines how the entire system would work even if EMW would only realize parts of it. This architecture is broken into subsystems that are sufficiently independent to allow for all reasonable variations. Designing a system that includes all possible variants is not a design pattern but a widely applicable design principle.

**“The flexible and independent structure of the architecture was very useful when combining components to meet the customers’ different needs and resulted in reduced cost as well as faster delivery of the projects.”**

### The Data Flow Principle

The decomposition into subsystems was done according to the Data Flow Principle. In this common architectural style, you study how data flows through the system and divide it into subsystems at suitable points in the data flow. For example, radar can be divided into a

Figure 2: The Pipe and Filters Pattern



Note: Arrows indicate the data flow between components.

transmitter, antenna, receiver, signal processor, data processor, or presentation, reflecting the path of a radar pulse through the system. Radar data processing has traditionally been structured along the same lines. One common structure is as follows: target tracking, data fusion, threat evaluation, and engagement planning. Most of those involved know and understand this structure.

### The Design Pattern: Pipes and Filters

The pattern that supports data flow is *Pipes and Filters* [2]. It is intended to solve the problem of structuring a big system that transforms a stream of data. Designing such a system as a single component is unwise and makes the system inflexible and vulnerable to future changes. For this reason, the system is divided according to how data flows as shown in Figure 2. We primarily use this pattern to structure the application layer, but it also has implications for the support layer.

### Composition of Layers and Pipes and Filters

The problem is how to fit both the Layers pattern and the Pipes and Filters pattern into the architecture. The Layers pattern structures the system vertically while Pipes and Filters structure it horizontally. We had to find an architecture that retains the desirable features of each pattern. Using Pipes and Filters directly on the application would make those application components that represent the filters aware of each other. This defeats one of the goals of the Layers pattern, namely application independence.

The solution was to put data storage components that work as buffers between the different applications – that is, the pipes in the Pipes and Filters pattern – in the support layer. Figure 3 shows the architecture with the Layers and Pipes and Filters patterns combined. This solution radically reduces the coupling between applications. By also making the components independent in terms of synchronization, we minimize the coupling between the components that produce and consume data. It is the case that a real-time system sometimes produces data faster than it can consume it, and a direct coupling between producer and consumer may then lead to the use of stale data. It is often better to discard the old data and continue with the relevant information.

With a separate data storage component we can isolate the problem and store valid data only without involving the producer or consumer. The producer pro-

duces data at one rate, and the consumer consumes it at a different rate. The data storage components ensure that the most current data are used.

**Event Handling**

The simple model with the components in the application layer decoupled by means of data storage components in the support layer works for data that are produced and consumed continuously. Other data, which may be associated with different events or operator controls, do not fit in this model. This is because changes occur rarely, but the application must react very quickly to them. In a data flow solution, an application that is dependent on a certain control would have to query the data storage component for its status quite often. Most of the time, the control would be unchanged. This is inefficient, but on the other hand, allowing the data storage component or some other application to send the information directly to another component in the application layer would create a strong dependence between components.

The solution is to introduce an event handling mechanism. Applications subscribe to an event by calling an event handler. When another application generates an event, which may carry with it other (changed) information, the subscribers are informed of the occurrence and can retrieve the additional information from the data storage components.

This kind of event handling reduces the coupling between the event-generating component and the subscribers; they need not be aware of each other. The approach is quite resilient to system changes. The event-handling mechanism works even if a certain subscriber is absent in a given variant of the system. Not even the event generator has to be present in all system variants, which will then lack certain functionality. A drawback is that event handling is an unstructured and dynamic way of information exchange comparable to exception handling and should be used restrictively.

**The Design Pattern Observer**

A design pattern that captures the idea behind the event handling described earlier is called *Observer* [1]. It is also referred to as *Publish/Subscribe* [2] and is used to synchronize the state of cooperating components. The component that detects the state of change publishes a message that is then forwarded to any number of subscribers.

The pattern solves the problem where many different components must be

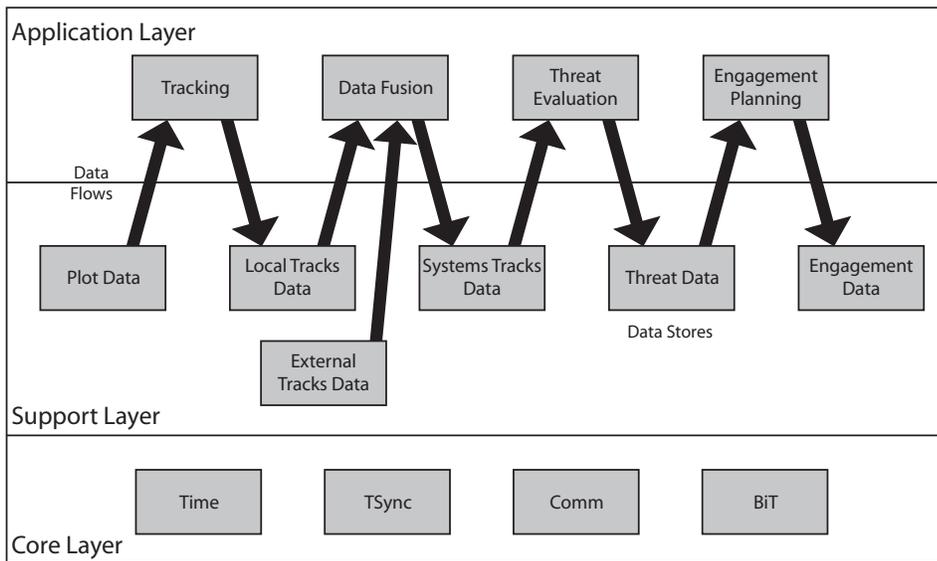


Figure 3: *Combining the Layers Pattern with Pipes and Filters*

informed of a relatively rare event in a flexible way. The salient feature of Observer is the reduced coupling between the publisher and the subscribers.

**Presentation and Control**

Presentation and control are particularly susceptible to rework due to different customer requirements. Changes are difficult to predict because the requirements for presentation and control tend to be unique to each customer. The structuring of the user interface is often fundamental to the architecture of any application of which it is a part. There are two possibilities:

1. Separating presentation and functionality with the rationale that the presentation represents one cohesive unit, and functionality another. That is, certain functionality is considered more

closely coupled to another functionality than to its own presentation. The *Model-View-Controller* (MVC) pattern captures this approach [2].

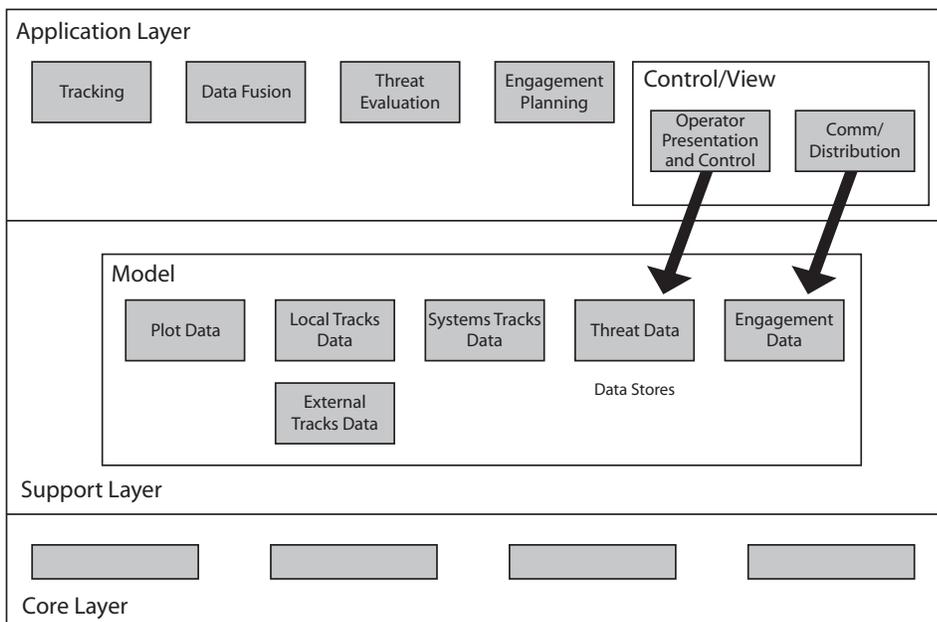
2. Tying functionality more closely to its presentation than to another functionality. A pattern that captures this view is *Presentation-Abstraction-Control* [2].

In our application, the concrete presentation and control are collected in one component: Operator Presentation and Control (OPC). This component is the entire program's interface to the human user. Changes concerning the concrete presentation and control are localized in one component according to MVC [2].

**The Design Pattern MVC**

The MVC pattern divides the world into a model, a presentation part, and a control

Figure 4: *The Architecture After Applying the MVC Pattern*



part. For compatibility with our own earlier designs, we used a variant, also described in [2], where the presentation and control parts are combined, while the model remains separate and consists of all the data storage components in the support layer.

The problem that MVC intends to solve is that user interfaces are especially vulnerable to change requests, and lead to many program modifications. A change to the presentation of one part of the system often forces a change to another part's presentation. It is hard to build a presentation with the required flexibility. Instead one can accept the situation and just structure the system so that all parts that handle the presentation are separated from the model.

### Combining MVC With the Earlier Patterns

In the final architecture, MVC must obviously be combined with the earlier patterns. Figure 4 (see page 27) shows how it fits. The control/view part consists of two components: OPC for the actual presentation and Communication/Distribution for the distribution of data to external systems. The MVC pattern supports the idea that external communication is just another form of presentation. This means that external communication, which also tends to vary drastically between projects, can be handled in the same way as presenta-

tion. Figure 4 shows how the components for presentation and external distribution are incorporated into the architecture. The data that are presented and distributed are in the data support layer, which is in accordance with MVC.

### Conclusion

The resulting architecture shows that the four patterns, Layers, Pipes and Filters, Observer, and Model-View-Controller, can be combined quite elegantly in radar software. We have also shown that it is possible and beneficial to use design patterns in a large and very complex application. The same overall architecture has been used in seven different projects with quite different functionality in the human-machine interface, external communication, and command-and-control parts. In five of these projects, the design and implementation of the software is now completed. The flexible and independent structure of the architecture was very useful when combining components to meet the customers' different needs, and resulted in reduced cost as well as faster delivery of the projects.

We believe that you can expect to continuously improve and refactor the software architecture during its entire lifetime. For example, the OPC needs to send commands and controls to the application components in a more effi-

cient way. Another example is the inner structure of the OPC component. Finally, it may be desirable to split the view and the controller of the MVC pattern into separate components.

In this article, we have only described a logical view of the components and ignored the mapping to physical processes and machines. Further work has been initiated to see how components can be structured and delivered as distributed applications that can execute on different nodes in a network. ♦

### Acknowledgment

The patterns work was partially supported by the Swedish Defense Materiel Administration, Försvarets Materielverk through the FOTA project.

### References

1. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
2. Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-Oriented Software Architecture: A System of Patterns. John Wiley, 1996.
3. Day, J. D., and H. Zimmermann. The OSI Reference Model. Proc. of the IEEE. Vol. 71, Dec. 1983: 1334-1340.

## About the Authors



**Kent Petersson** is currently working on software architectures at Ericsson Microwave Systems in Mölndal, Sweden. He has a background as associate professor at the Department of Computer Science at Chalmers University of Technology in Gothenburg. His research interests include program verification, type systems, and functional programming. Petersson received his degree in mathematics and computer science from the University of Gothenburg, Sweden.

**Ericsson Microwave Systems AB**  
**Surveillance and Communication**  
**Systems**  
**SE-43184 Mölndal**  
**Sweden**  
**E-mail: kent.petersson@**  
**ericsson.com**

**Tobias Persson** is manager of a software design group for Ericsson Microwave Systems and has worked at the company as a software engineer in radar projects. Persson has a Master of Science in computer science from the University of Gothenburg, Sweden.

**Ericsson Microwave Systems AB**  
**Surveillance and Communication**  
**Systems**  
**SE-43184 Mölndal**  
**Sweden**  
**E-mail: tobias.persson@**  
**ericsson.com**



**Bo I. Sanden, Ph.D.**, is professor of computer science at Colorado Technical University in Colorado Springs. He spent 15 years as a software architect with UNIVAC and Philips. His main research interest is software design. Sanden has a Master of Science in engineering physics from the Lund Institute of Technology, Lund, Sweden, and a doctorate in computer science from the Royal Institute of Technology, Stockholm.

**Colorado Technical University**  
**4435 North Chestnut St.**  
**Colorado Springs, CO 80907-3896**  
**Phone: (719) 531-9045**  
**Fax: (719) 598-3740**  
**E-mail: bsanden@acm.org**



# Introducing Global Software Competitiveness

Don O'Neill

Independent Consultant

*The vision to be globally competitive in software is accomplished by setting the enterprise direction, providing the fuel, and controlling the business environment, including suppliers, customers, competitors, and event threats. Software competitiveness revolves around how the software work force is used to achieve customer satisfaction, how innovation is essential to delivering customer value, and how strategic software management guards against event threats and even exploits change. In association with the Center for National Software Studies, I have identified the enterprise capabilities needed to achieve global software competitiveness in several stages. In addition, leading indicators have been identified to guide assessment and improvement of these capabilities.*

The prosperity of the nation depends on achieving global competitiveness in critical industries, including telecommunications, transportation, manufacturing, finance, utilities and energy, medical systems, and defense. Each critical industry contains strategically essential value points that are increasingly dependent on software. Achieving global competitiveness then depends on achieving excellence in software in each industry.

The Center for National Software Studies has identified the enterprise capabilities needed to achieve global software

competition. In addition, leading indicators have been identified to guide assessment and improvement of these capabilities along with models to reason about competitiveness.

## Global Competitiveness Maturity Levels

The strategic management of global competitiveness in software calls for setting direction, providing fuel, and controlling the business environment including the supplier, the customer, the competition,

and change and event threats. Accordingly, enterprise maturity in global software competitiveness is achieved in five levels.

- Level 1 is the absence of expectation, achievement, and engagement in the conversation on global software competitiveness.

---

*Due to space constraints, CROSSTALK was not able to publish this article in its entirety. However, it can be viewed in this month's issue on our Web site at <[www.stsc.hill.af.mil/crosstalk](http://www.stsc.hill.af.mil/crosstalk)> along with back issues of CROSSTALK .*

---

# Data Warehouse: Your Gateway to the Information Age

Kelly L. Smith

CASEware Technology, Inc.

*Many people consider data warehousing their gateway to the information age. The right approach to this process can deliver great rewards and bypass the pitfalls experienced by many. In this article, you will discover what a data warehouse is, why you should use one, and what steps should be involved when implementing one.*

The concept of data warehouse has given new hope to providing a solution to the ever-increasing need for more information. A data warehouse is a collection of data designed to support management decision-making. It contains snapshots of summarized and extracted data from operating systems that are stored in a warehouse database system that provides flexible access to information. CASEware Technology Inc. has applied the technology of Computer Aided Software Engineering (CASE) tools and concepts to forge the way to implement the data warehouse.

Today, most corporations are finding the legacy systems of the '70s and '80s do not provide the path needed to compete in today's market. The technology available for development of these systems did not

allow them to be designed from a global view. As a result, the systems were highly specialized and focused on specific functional areas of the business such as finance, human resources, material management, etc. Information between these systems was not integrated or shared. As a result, individuals cannot access complete information. The age of interfaces grew out of the need for complete information. These interfaces, however, are very time consuming, complex, and difficult to change. New systems are even more difficult to implement because of the interface issues of unplugging the old system. As a result, old systems never go away.

When interfacing could not fulfill the need for more information, power users resorted to downloading data from the legacy systems to provide them with the

necessary information. In an isolated incident, this works fine. However, as the need for information increases, more and more information is required until these requirements become unmanageable. It is no wonder that as the need for more interfaces, downloads, reports, etc. increases that information systems soon became backlogged with no hope for digging out. This backlog of work prevents the information technology resources from focusing on providing new integrated systems, rather than increasing the complexity of the environment.

---

*Due to space constraints, CROSSTALK was not able to publish this article in its entirety. However, it can be viewed in this month's issue on our Web site at <[www.stsc.hill.af.mil/crosstalk](http://www.stsc.hill.af.mil/crosstalk)> along with back issues of CROSSTALK .*

## LETTER TO THE EDITOR

Dear **CROSSTALK** Editor,

The February issue of **CROSSTALK** printed an article I wrote on choosing a software language. At the beginning of the article, I give some examples of how language choices are being made. In the original submission, one of the examples was the bookstore method, in which the number of books in the bookstore were counted and that was deemed to be the best language. However, this was edited out of the final article. Perhaps the editor didn't believe it, or maybe it was cut to save some space.

At the 2003 Software Technology Conference in April, one panel for a military program presented a brief on their language choice for their program. One of the slides they presented was a study of the number of books published on the various languages. To understand why this is a bad metric, I invite the reader to tour my home library. I want to show you books on three of the over 10 languages represented.

First, I have three books on Java. One is obsolete, because the language has changed since I bought it. The one I want you to look at closely is by Deitel and Deitel; it is a well written college textbook targeting the first software language crowd. These days when I hear Simon and Garfunkel singing about the words of the prophets being written on the subway walls, I think I see them scribbling the word Java. There will be many more additions to this language in the next couple of years, like the addition of generics. I think I will be buying more books on Java.

My older C++ books say that C++ is a superset of C and so there is already a large base of programmers using it. This is a blatant lie. It wasn't true then and it is not true now. One book is specific to Borland C++. Two books are specific to Visual C++, but they are obsolete as that platform has changed since I got them. My favorite is like three books in one because it covers three types of C++: switch (pick your version) {case version\_one: do it this way; break; case version\_two; do it that way; break; case version\_three; do it like so; break;}. Old C++ books are not useful because they don't have information on exceptions or namespaces or other additions to the language.

I have two stacks of magazines that are devoted to this language, and they are full of articles that tell me how to use it safely by not using some of the features, or how to be careful using other features. An interesting note about these magazines is that a lot of the code in them is actually Delphi, a language that was advertised as C++ without the problems. C++ claims to be a friend to object-oriented programming, but the friend function violates the paradigm; unless you believe the book that says it doesn't. These books claim that C++ is a strongly typed language, but I disagree. Besides my books and magazines, I have a notebook full of coding standards that tell how to make this language readable and safe to use, but they are full of contradictions.

I feel that these books and the C++ language have deceived me. My theory is that this language is still being used because of some weird Abilene paradox phenomenon. In the Abilene paradox, a group of people went to

town to eat and when they got back, they discovered that none of them really wanted to go, but they all went because they thought the others wanted to go. In more than one discussion with fellow engineers we all agree that C++ is a poor language, but we all use it because everyone else is using it. Engineers have to learn to read this language. For example, of about 20 books in my library on neural networks, 10 of them have example code in C++, even though Ada tasks would be more suited for building neuro nets. I don't plan to buy any more books on C++.

So, let's look at my Ada books. I have about 25 of them. Two of them I don't like. Five of them are now available on the Internet for free. There are three on the Internet that I don't have in hard copy, and I'm not counting them. Seven of them are Ada95, and the rest are Ada83 books. The old Ada books are still useful because of the backwards compatibility of the language and the fact that many of them are also about software engineering principles as well as the language. The one on distributed real-time systems is still a good book on distributed real-time systems, even though Ada95 fixed many of the issues that were raised.

These books compliment each other. Some authors are better at explaining access types, others at explaining tasking, and still others at explaining scope rules. These books are honest; they don't contradict each other. Unfortunately, our universities are slower to change and adapt to new technology than our militaries, so not many schools teach this language.

At the head of my linked list of want-to-have books is "Real-Time Systems and Programming Languages: Ada95, Real-Time Java and Real-Time Posix," by Alan Burns and Andrew J. Wellings. Should I file this with my Ada books or my Java books?

It is not the number of books that are published that count; it is their honesty, consistency, and the fact that they are endurable that counts. This is a much harder metric to measure. Last week in the bookstore, the number of books on C# and Java were each more numerous than the number of books on C++. Still, neither of these languages is as powerful or as stable as Ada. They are just more popular. It pleased me to see that there were three or four vendors at the STC from which I could buy Ada compilers and programming environments.

Dennis Ludwig  
*Simulation and Analysis Facility  
Wright-Patterson Air Force Base*

---

*CROSSTALK invites readers to submit their thoughts, comments, and ideas on its themes and articles as a "Letter to the Editor." Simply e-mail letters to <crosstalk.staff@hill.af.mil>.*



# CrossTalk Terminology Invitational

Each year the *Washington Post's* "Style Invitational" asks readers to take any word from the dictionary, alter it by adding, subtracting, or changing one letter, and supply a new definition. Some past winners include the following:

**REINTARNATION:** Coming back to life as a hillbilly.

**OSTEOPORNOSIS:** A degenerate disease.

**KARMAGEDDON:** It's like, when everybody is sending off all these really bad vibes, right? And then, like, the Earth explodes and it's like, a serious bummer.

**GLIBIDO:** All talk and no action.

**DOPELER EFFECT:** The tendency of stupid ideas to seem smarter when they come at you rapidly.

Not to be outdone, I beseech CROSSTALK readers to contribute to CROSSTALK's "Terminology Invitational" by taking any software engineering term, alter it by adding, subtracting, or changing a letter (or two), and supplying a new definition. To get you started, here are a few easy ones:

**AGUILE SOFTWARE DEVELOPMENT:** A software development method that favors skillful deceit and trickery over customer collaboration and working software.

**APPLECATION:** A group of positively charged programmers who still believe Apple Computers will achieve global domination.

**BEER REVIEW:** The review of after-work beverages for optimal obliteration after a long day.

**BONDWIDTH:** A software engineer's capacity to participate in pair programming.

**CAPABILITY MANURITY MODEL:** A model that delineates the characteristics of a process that stinks.

**CODEING:** Transforming a design into a program language while under the influence of a narcotic derived from opium.

**COMPUTER-RAIDED SOFTWARE**

**ENGINEERING:** The automation of methods for pilfering software designs.

**CUSSTOMERS:** Clients who use profanity to tell you what they really think about your software.

**DEVILERABLE DOCUMENTATION:** A wicked user's manual that torments its readers.

**DONFIGURATION:** Re-arrangement of programmer body parts when a racketeer-ing application crashes.

**DORK BREAKDOWN STRUCTURE:** A listing of all activities and attributes that renders a software engineer a dork.

**FAUX TOLERANCE:** The number of mistakes a manager can withstand before sacking a software engineer.

**FUNCTION PINT ANALYSIS:** A measurement process that focuses on the number of pints of ale required to start a project.

**FUNCTIONAL DECOMPOSTITION:** The filtering of system functions to remove decayed code and improve structure.

**HYPOTEXT MARKUP LANGUAGE:** A programming tool used to develop squalid spam and pop-up messages.

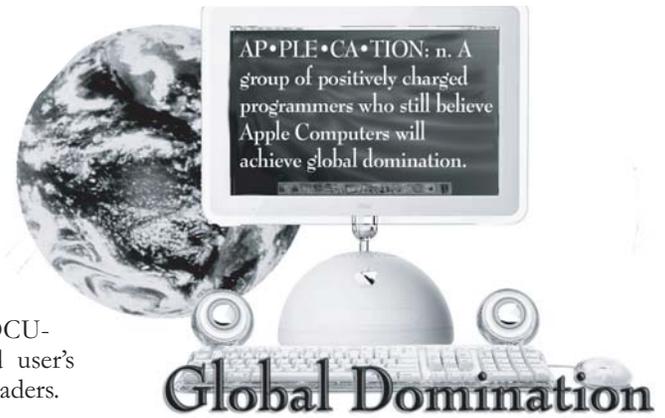
**JOINT SUPPLICATION DESIGN:** A design technique that humbly beseeches users for requirements and money at the same time.

**KINSPECTIONS:** An analysis technique that relies on visual examination of close relatives to detect errors and family issues.

**METADATE:** Something rare for a software engineer.

**PEST-CLOSURE ACTIVITIES:** Activities that occur after a software system has been formally accepted and problems begin.

**PROTECT MANAGEMENT:** A system of practices and know-how that shields a manager from blame on a failed project.



**RABID PROTOTYPING:** A tumultuous, uncontrollable system built inexpensively for demonstration so that end-users can determine what they don't want.

**RELEASE VEERSION:** A software application that has been tested and found to be completely off course from the original design but released for use anyway.

**RETIREMENTS ELICITATION:** The process through which a manager and engineer discover, articulate, and understand the conditions by which the engineer will leave a project and retire.

**OBJECT-ORIENTED DESIGN:** A software development technique in which a system or component is designed to fleece people on the Internet.

**SOFTWARE ENGINEERING PETHICS:** A gullible unsophisticated pair programmer kept for amusement.

**TOST ESTIMATION:** What 99 percent of software cost estimates become.

**USER WINTERFACE:** The look you get from a customer when your program freezes up during acceptance testing.

**WHISK MANAGEMENT:** A quick, nimble, rapid decision-making management style that whips a project into shape and works well with agile software development but may leave egg on your face if not done right.

Vote for your favorite term at <garyp@shiminc.com> or send your own term and definition, and it may get published.

— Gary Petersen  
Shim Enterprise, Inc.

# The Sixteenth Annual Software Technology Conference

19 - 22 April 2004 • Salt Lake City, UT

## STC → 2004



### Technology: Protecting America

#### Reasons to Exhibit at STC 2004

- Participate in the premier software technology conference - endorsed by the Department of Defense (DoD)
- Interact with over 2,500 leaders and decision makers in the DoD, government, industry, and academia
- Choose your level of involvement by becoming a platinum, gold, or silver trade show sponsor

**Register to exhibit today!**

**www.stc-online.org**  
**or 800-538-2663**

Source Code: CTD

**Co-sponsored by:**

United States Army	United States Air Force
United States Marine Corps	Defense Information Systems Agency
United States Navy	Utah State University Extension

**Co-hosted by:**

Cyber Air Logistics Center (CC)  
Air Force Software Technology Support Center



Sponsored by the  
Computer Resources  
Support Improvement  
Program (CRSIP)



Published by the  
Software Technology  
Support Center (STSC)

**CrossTalk / MASE**  
6022 Fir Ave.  
Bldg. 1238  
Hill AFB, UT 84056-5820

PRSR STD  
U.S. POSTAGE PAID  
Albuquerque, NM  
Permit 737