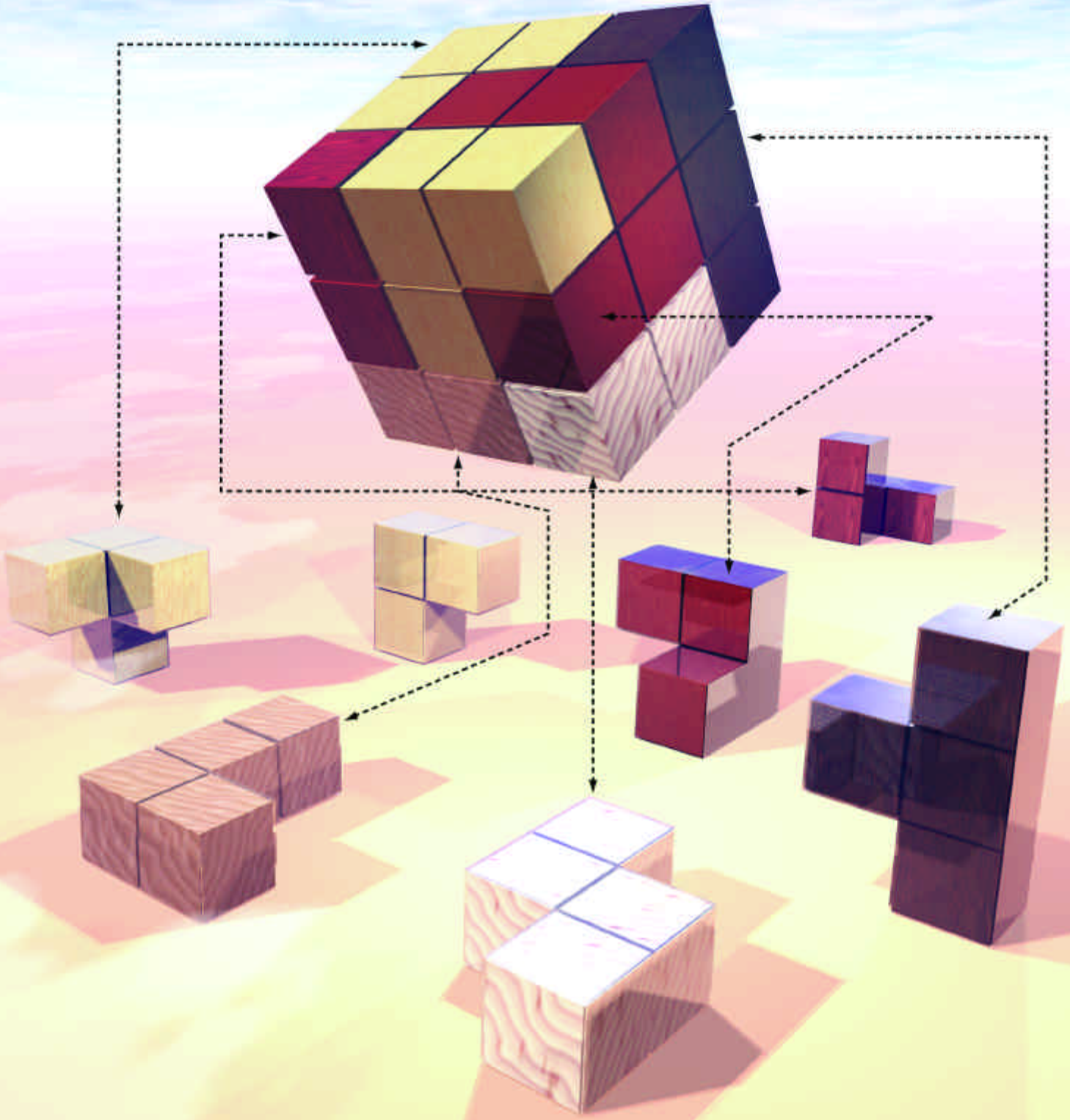
CROSSTALK

November 2001 The Journal of Defense Software Engineering Vol. 14 No. 11



DISTRIBUTED SOFTWARE DEVELOPMENT

Distributed Software Development

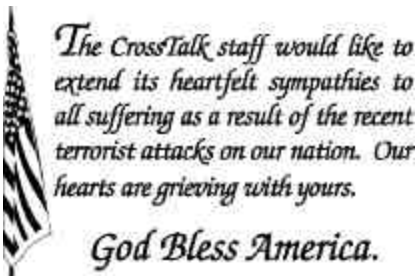
ON THE COVER

Kent Bingham, Digital Illustration and Design, is a self-taught graphic artist/designer who freelances print and Web design projects.

Software Engineering Technology

Open Forum

Departments



CROSSTALK

SPONSOR Lt. Col. Glenn A. Palmer

PUBLISHER Tracy Stauder

ASSOCIATE PUBLISHER Elizabeth Starrett

MANAGING EDITOR Pam Bowers

ASSOCIATE EDITOR Benjamin Facer

ARTICLE COORDINATOR Nicole Kentta

CREATIVE SERVICES COORDINATOR Janna Kay Jensen

PHONE (801) 586-0095
FAX (801) 777-5633
E-MAIL crosstalk.staff@hill.af.mil
CROSSTALK ONLINE www.stsc.hill.af.mil/crosstalk/crostalk.html

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. 9.

Ogden ALC/TISE
7278 Fourth St.
Hill AFB, UT 84056-5205

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CROSSTALK editorial board prior to publication. Please follow the Author Guidelines, available at www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf. CROSSTALK does not pay for submissions. Articles published in CROSSTALK 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 CROSSTALK.

Trademarks and Endorsements: This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSSTALK 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 CROSSTALK 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 CROSSTALK 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.

Which Distributed Development Concept Did You Have in Mind?

Are we developing software at distributed sites, or are we developing distributed software? These are obviously very different concepts. Until I saw the articles that are in this month's issue, I didn't realize there was so much about distributed software that I didn't know.

Distributed Development of Software: When we received Paul McMahon's article, *Distributed Development: Insights, Challenges, and Solutions*, it occurred to me that I wish I had known this information long ago. McMahon's ideas about seamless vision, virtual communication, and testing the virtual organization would have really been helpful to one of my past projects. My company worked with another company, trying to provide the end product to the customer; it was difficult at best. This was not a joint collaboration from two companies working as a team. The customer selected my company to develop most of the software, then informed us that we would let company B develop a portion of the software and use their product with ours.

There was no attempt to integrate product teams, create a shared vision, or communicate virtually. Company B simply sent numerous versions of their software, and each time I sent them back a list of errors I found while testing it. When the software was returned, some errors were corrected, some were not, and new errors were inserted. Finally, I just sent them a copy of the test I was running so they could verify that the software passed the test before sending it to me again.

However, I did not realize that company B did not use the same computer set-up as I did. My test listed keystrokes – not computer functionality. So they sent the software back with errors in it again because they could not run my test. The other company did not know what I was trying to accomplish with the test, and instead of calling and asking for clarifications, they just sent back another bug-filled software package with an unusable user's manual. Management finally just requested that I fix the software myself and rewrite company B's user's manual. That finally brought closure, but time and effort would have been saved using McMahon's insights.

Development of Distributed Software: In *Toward Adaptive and Reflective Middleware for Network-Centric Combat Systems*, Dr. Douglas Schmidt and several of his colleagues propose a vision for how coupling military sponsored research, industrial standards development, and commercial products can dramatically advance capabilities in the development of complex military embedded systems. They cite the Boeing Bold Stroke project as just one example of a real-world military system leveraging this technology toward the establishment of an open systems-based software product line. We will learn more about Bold Stroke in December with an article that describes their Air Force Research Laboratory-sponsored work in affordably upgrading legacy applications to COTS-based computing platforms.

Distributed Development of Distributed Software: What if you want to purchase distributed software middleware? Dr. Thomas Croak comes to the rescue with *Factors to Consider When Selecting CORBA Implementations*. If you are planning to buy a CORBA distributed object-computing system this article will help. Dr. Croak's analogy to buying a car is also applicable to any software purchases you may plan to make.

Dr. Lon Gowen's supporting article, *Predicting Staff Sizes to Maintain Networks*, provides helpful information to readers responsible for managing technical resources. Predicting staffing sizes is a problem for many organizations, and this article should help with the struggle.

Hillel Glazer rounds out this issue with *Dispelling the Process Myth: Having a Process Doesn't Mean Sacrificing Agility or Creativity*. Having worked on several small projects, I agree that tailored processes can indeed make agile processes more productive with the time saved by reducing rework.

The theme topics addressed in this month's CROSSTALK are not my areas of expertise. However, the articles are very informative, and I now know more than I did a few months ago. Maybe if these subjects come up in the future, I will now know enough to start asking the right questions to help with new endeavors. I learned a lot from reading this month's CROSSTALK. Of course, I never miss an issue. (As the associate publisher, I'm paid not to miss an issue.) I hope you never miss an issue either.

Elizabeth Starrett

Elizabeth Starrett
Associate Publisher

Distributed Development: Insights, Challenges, and Solutions

Paul E. McMahon
PEM Systems

Today, many organizations are facing difficulties competing for new work due to a critical shortage of engineering skills. Employing the power of distributed development can increase an organization's opportunities to win new work by opening up a broader skill and product knowledge base, coupled with a deeper pool of personnel to potentially employ. By distributed development, we mean development efforts that span multiple organizations and/or multiple physical locations. This article provides an overview of key issues and challenges managers and project engineers are facing today on distributed development efforts. Insights into root causes of difficulties and recommended solutions are provided. Within this article the terms distributed and virtual are used synonymously. Information presented in this article has been excerpted and condensed with permission from the newly published Virtual Project Management: Software Solutions for Today and the Future [1].

Other authors using the terms *virtual collaboration*, *virtual development*, and *distributed development* have addressed the subject of this article. Regardless of the name used, this subject is about the issues involved when multiple organizations and/or multiple physical locations join forces on an advanced technology software-intensive effort.

It is important to note that I am not referring to traditional subcontract relationships. Rather, this article focuses on the use of *virtual teams* operating more as a single *integrated* team employing some degree (to be discussed) of common processes, support services, and technical strategies driven through a streamlined management chain.

While just a few short years ago such a project would have seemed inconceivable, modern technologies like e-mail, the World Wide Web, NetMeeting, teleconferencing and videoconferencing are today providing new possibilities for distributed teams to work in a more integrated and productive manner.

Why Care About Distributed Development?

To understand the critical importance of succeeding in a virtual collaborative environment, one only needs to examine the changes taking place inside today's workforce. According to a recent study conducted by the U.S. Bureau of Labor Statistics, approximately 25 percent of all workers age 16 or over have been with their current employer 12 months or less. This same study indicates that the average worker is now expected to change jobs every five years [2].

These statistics paint a picture of an increasingly mobile work force. The 20-year employee holding a wealth of corporate knowledge inside his head may well be a corporate asset of the past. At the same time, corporations are experiencing an increasing demand for software-intensive solutions produced from a combination of existing products and new developments. This, in turn, is driving a greater demand for personnel with increasingly specialized software skills; that is, skills geared toward specific software products. Compounding this demand for key people is the seemingly never-ending shortening of cycle times to enter and succeed in new markets.

> "Within this demanding environment, even the largest mega-corporations are finding they can no longer maintain inside their own corporate walls all the critical skills necessary to compete in many new markets."

Within this demanding environment, even the largest mega-corporations are finding they can no longer maintain inside their own corporate walls all the critical skills necessary to compete in many new markets. As a result, more and more companies today are reaching out in a cooperating manner to organizations previously viewed only as competitors. While the rationale for embracing virtual operations is evident, many corporations today are struggling with implementation-related challenges.

Distributed Development Challenge

Research conducted by Booz-Allen [3] has identified four characteristics common to many of today's collaborative failures:

- Cultural incompatibility.
- Leadership struggles.
- Lack of trust.
- Inbred notions of competition.

Recently, a three-year collaborative development study was documented in *Virtual Project Management: Software Solutions for Today and the Future* [1]. The results indicate that beneath these symptoms lie a number of key relationships that include both technical and non-technical factors. On the positive side, this study also indicates that, once these key relationships are understood, practical and affordable actions can be taken to aid organizations in achieving successful virtual operations. The referenced study is based on experiences derived from real distributed projects that occurred between 1994 and 2000.

The Eight-Step Plan

In this article we employ an eight-step plan (see Figure 1) as a framework to assist our investigation of distributed project challenges. This framework can be used as an aid in setting up a new distributed project, or in instituting improvements to an ongoing one.

While the eight steps identified may appear traditional and relevant to any project, our focus in this article is on specific issues related to distributed operations. It should be noted that the use of the term *steps* is not intended to imply that virtual project success can be achieved through a *cookbook* approach. Nor do we mean to imply that these steps are easily achieved. Many of the challenges faced on

virtual projects are closely coupled to communication. Evidence of this fact can be seen throughout the eight steps starting with Step 1.

Step 1: Selecting Team Leaders

It should not be a surprise to anyone who has worked on a collaborative endeavor that one of the most important decisions to project success is the choice of team leaders. While strong conflict management skills, and a willingness to consider alternative approaches are desirable traits for leaders on any project, these leadership characteristics are particularly critical to distributed project success.

Unlike most traditional collocated projects, virtual projects face the added challenge of teammates with differing backgrounds, experiences, and technical expectations. This situation can give rise to frequent and often intense conflict. While most project conflicts faced are not insolvable, all too often timely and effective resolution falls short due to a breakdown in communication.

Conflict and Communication

Conflict is not unique to distributed projects, but it is not uncommon for traditional, collocated approaches to conflict resolution to fail in a distributed environment. To understand why, we must look closer at communication in the organization.

In the early 1980's, Alan Cox conducted a survey in which he found that more than 66 percent of middle managers believed that more than half of the communication in their organizations occurred informally [4]. Experience on distributed projects indicates this is not only true, but some of the most critical communication with respect to conflict resolution occurs in this manner. Unfortunately distance, differing experiences, and internal team competition often hinder informal communication on distributed efforts.

It should be noted that the term *informal* in this article means unplanned and undocumented.

A Partial Solution to Virtual Project Conflict

While distributed development provides the potential power of rapidly accessible personnel with key skills and key product knowledge, it often does so at a cost of interpersonal team bonds built over time through shared experiences. Although there does not exist a simple cure-all for the loss of shared experiences and traditional informal communication on distributed projects, a number of partial solutions do exist.

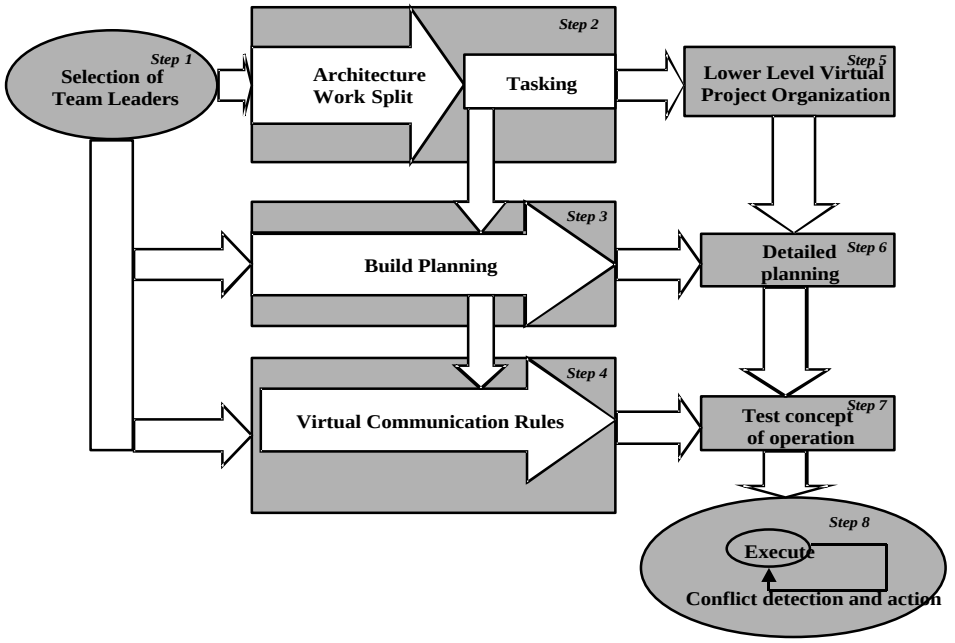


Figure 1: *Eight-Step Plan*

For example, experience indicates that collocating a small team of senior systems designers during the critical creative design stage of a distributed project can be effective. Studies have shown that when team members must interact frequently and for short durations, collocation offers the best opportunity for success [5]. This does not mean that full-time collocation is required for the life of the program. Cases where repeating cycles of intense collocated work followed by periods apart have worked well, especially during the early critical creative design stage.

It is also important to note that early collocation is multi-purpose. This activity supports the shared development of a project's common technical vision, but equally important it also starts the process of building interpersonal bonds among key teammates across project organizations/sites. Other recommendations in support of a shared leadership vision are discussed in Step 7.

Step 2: Architecture, Work Split, and Tasking

While collocation of key personnel early can go a long way to getting a distributed effort off on the right foot, a significant number of managers remain skeptical about the viability of distributed development. When surveyed on this subject, many managers have expressed fear in not knowing if a remote team member is "doing the right thing [6]."

When one considers how much a new engineer traditionally has learned about task expectations through informal means, this fear is understandable. Managers who have known only collocated operations often take informal communication for granted, but they intuitively know how much they rely upon it every day.

Even in organizations where task assignments are formally written, there is usually a significant reliance on informal communication to clarify and guide the new engineer. With respect to designing or coding a solution, this informal guidance often takes the form of an experienced engineer relaying examples of solutions patterns that he/she knows will fit that particular organization's accepted technical architecture. Although the process described is commonplace, the relationship being described among architecture, work split, and tasking has not always been well understood.

Definitions

When the term *architecture* is used, it means the "components" of a system and the rules defining how the components are connected, along with any constraints. When used, the term work split means the allocation of responsibilities across physically separated sites or organizations. Work split can also be thought of as *site-level tasking*.

Architecture as a Management Tool

Traditionally, many think of architecture as a technical issue, and work split as a separate and distinct management issue. But in practice, work split decisions can fracture a sound architecture. Furthermore, a

sound technical architecture can actually provide one of the best task communication and coordination techniques.

For example, think about how a senior technical mentor guides a new engineer. The most effective mentors guide by listening first and then providing feedback that ensures the approaches chosen fit within the range of an organization's acceptable solutions. This is another way of saying that effective mentors guide through the vision of a technical architecture.

When used appropriately, a sound technical architecture can go a long way to addressing a manager's concerns about whether a remote team member is in fact "doing the right thing." Often it is through informal architecture-centric discussions that teammates gain the real insight needed to accurately meet task expectations within a specific organization. But for architecture to be effective as a task communication aid, the work split definition across physically remote sites must follow the architecture definition, not the reverse.

Architecture First

Too often we see work split decisions made without due consideration to the technical architecture. When work split decisions are forced prior to architecture definition, we often find distributed projects suffering from fuzzy task responsibilities and technical leadership struggles. Without a well defined architecture, remote groups often find themselves heading down inconsistent paths leading to project conflict and control struggles.

For example, the choice of computer hardware platform has been a topic of intense inter-site battles on many distributed efforts. By documenting agreed to platform constraints early, unnecessary project conflict during a critical project stage can often be avoided.

There are many sources available that can provide more in depth information on the key role of architecture to project success [1, 7, 8, and 9].

Do Not Delay the Work Split

There is another side to this coin that must also be adequately considered. Delaying the definition of work split too long can have equally devastating effects on a distributed project. The right answer to the problem of fuzzy task responsibilities is not always simply delaying the work split definition until the architecture is defined. When work split decisions are delayed too long, internal teams' mistrust quickly builds. Be aware that the consequences of defining a work split that leaves tasking grey in certain areas has proven to be a poor solution to this challenging area. While your architecture needs to be in place when you define your work split, you should also know that architecture is an evolutionary product. Never wait for the architecture to be 100 percent complete, or you'll never get your work split defined.

> "Studies have shown that when team members must interact frequently and for short durations that collocation offers the best opportunity for success."

Step 3: Planning the Builds and Site-Specific Infrastructure

While a sound architecture can aid work split definition and task management, it does not convey when product functionality is available. This is the purpose of a build plan.

Today, many companies are moving toward incremental build approaches especially on distributed projects because a build approach reduces integration risk. You can think of a build as a set of hardware and software that meets a subset of the functionality of the final deliverable product. *Planning and coordinating the builds across distributed sites may be the single greatest challenge faced on virtual projects.*

One of the keys to effective build coordination on distributed projects is found in the technical infrastructure (hardware computer platforms and software tools). The criticality of the technical infrastructure to effective build planning can best be conveyed through two competing technical infrastructure visions often found on distributed efforts. These two visions are referred to as the *maximize use of company-owned assets* vision and the *seamless* vision.

Maximize Use of Company-Owned Assets

Maximized use of company-owned assets means the use of existing company-owned organizational assets (computer hardware and software tools) to the maximum extent possible to meet project needs. Those who demand that the project's direct infrastructure costs be kept to a minimum drive this vision. While driving hard toward this vision does reduce the up-front project expenditures, it can also increase the project's integration risk and the overall project cost since not all existing hardware and software will be the same.

Seamless Vision

On the other hand, those driving the seamless vision believe that an engineer should be able to log in and do 100 percent of his engineering work using identical tools and processes from any workstation at any remote project location. While the advantages of the seamless vision are evident, the cost of common hardware, common software tools, and software licenses can quickly become prohibitive for a single project. It is also important to keep in mind that the choice you make with respect to infrastructure (hardware, software tools) cannot be made independent from your project's process decision unless you keep your process definition high. But keep in mind that if the process definition is too high, it is more likely to lead to miscommunication. In the following section we discuss a common distributed project process pitfall.

Do Not Drive Process Commonality Too Deep

One of the most common pitfalls witnessed on distributed projects is referred to as the "let's use the most mature process available" pitfall. This pitfall usually starts with the project leadership's decision to mandate that all project sites use a common process. While at the appropriate level a common process makes sense, the pitfall is tied to what often happens next. Rather than define the common process at the appropriate level and allow individual sites the appropriate freedom to leverage site-specific procedures, oftentimes a mandate is sent across the distributed sites to drive procedure commonality (different from process commonality) as well. The common set of procedures chosen is usually supplied by the highest software maturity-rated organization on the project.

It is natural to look to your teammate with the highest process maturity for software guidance. However, procedures represent only a small part of the complete process maturity picture. They are often too site-specific to make sense for application across multiple organizations (each with their own culture and history) in conducting development activities.

When attempts are made to drive commonality too deep into a virtual

organization the lack of an enabling organization, supporting infrastructure, and supporting culture at each of the remote sites is almost certain to lead this initiative to failure.

Solution to the Common Process Initiative

A process freedom line is defined to be the point in the process where a site/organization is free to make process-related decisions. For example, a project level procedure may call for a design document to be produced with specific design artifacts, but may not require a specific document format. I recommend that virtual projects define process freedom lines at the point where products and people must come together across divergent sites/organizations. It is not recommended that the project attempt to dictate how specific sites/organizations accomplish their tasks internally.

The freedom line definition essentially tells each site where they are free to leverage site-specific procedures (that can include site-specific support organizations and company-owned assets) in implementing solutions. This strategy has proven effective at balancing the management of the project's integration risk, while at the same time leveraging the strengths of individual sites/organizations.

Step 4: Virtual Communication Rules

Architecture definition and planning are critical on all projects, collocated or virtual. Virtual communication, on the other hand, presents distributed projects with new challenges not previously faced in traditional collocated environments.

Today, virtual communication is in its infancy. We are just now starting to comprehend the implications of first generation lessons on using the World Wide Web, teleconferencing and videoconferencing, NetMeeting, and e-mail. We recommend that virtual project communication lessons drive written virtual project rules (guidelines) to aid engineers in the effective application of new tools.

For example, in the early stages of a large virtual project it is not uncommon for engineering personnel to receive 50-60 e-mails per day! Think about it. If you take just four minutes to process each e-mail, at this rate you could spend half of your day just handling e-mail correspondence. E-mail flooding is the result of personnel being given a new tool and insufficient training in its use. E-mail, voice mail, teleconferencing, videoconferencing and NetMeeting all require training in more than just the mechanics of their use.

I recommend that each project create its own guidelines as it moves forward. And don't ignore rules and lessons that may seem obvious. I challenge those who have been involved in distributed efforts with the following questions:

- Does your distributed project have rules for the use of e-mail and teleconferencing?
- Have your people been trained and are they following the guidance provided?

> "On virtual projects the building of trust requires a more proactive management stance."

It has been our experience that while few disagree with the concept of guidelines, most distributed projects in operation today are not doing the best job of deploying effectively virtual communication technologies, and the unfortunate part is that it could be costing you plenty in human resources. Furthermore, this recommendation is simply not that expensive to deploy!

For more examples of first-generation virtual communication lessons, see [1].

Step 5: Lower Level Virtual Project Organization

The recommendation for the lower level structure of a distributed project organization may not be a popular one. At the top end of the organization where a breath of issues must be addressed, the Integrated Product Team (IPT) structure tends to function well, and I recommend it. This is the level where *heads-up* activities exist. By heads-up activities I mean work that must look across the multiple sites and organizations of the project. But I have also observed – and many clients concur – that a strict IPT structure is weak when it comes to producing products that include detailed design, code, and test cases.

Where the *real* engineering, or what we refer to as heads-down work, occurs, I have found that on distributed efforts a *hybrid* of IPTs and functional groups is often more effective. When I use the term heads-down work I mean the engineering work associated with building and testing actual executable code.

An example of why we recommend this structure can be seen in the need for an infrastructure implementation group that provides common services that multiple product development teams may need across remote sites. Too often, when virtual projects try to drive a strict IPT structure deep into the organization, responsibility for critical common services is lost. This is because when a strict IPT structure is employed at lower levels of the organization, you often find that each of those lower level IPTs focus almost exclusively on their own specific product. As a result, each solves their own specific problem in their own specific way.

On the other hand, when organizations recognize the need for an infrastructure implementation group that is not focused on any single specific product, commonality across multiple sub-products can be more effectively achieved.

Step 6: Detailed Planning

Detailed planning is important on any project, but on a virtual project its critical relationship to work split is often misunderstood.

Often on virtual projects work split decisions get delayed. This can occur for a multitude of reasons – most are not technical. But, all too often, great pressure continues to be brought on the engineering team to complete the detailed project plan despite the uncertainties of where work will actually get done. What is too often misunderstood in these situations is the extent to which detailed planning directly depends on work location.

While some project planning can be done independent of location, think about the real issues an engineering manager faces when it comes to developing a really detailed plan that is actually executable. Here are just a few of the critical questions to be asked:

- Is the development hardware available?
- Have the software tools and licenses been procured and installed?
- Have we identified the engineering personnel that will do this job?
- Have the identified personnel been trained on the chosen platform, language, tools?

To develop a detailed plan that is executable, managers must make assumptions with regard to each of these issues. These are the real issues that truly impact project performance.

Now think about how the answers to these questions are affected when work is moved to a different location. Based on my experience, if you are doing detailed

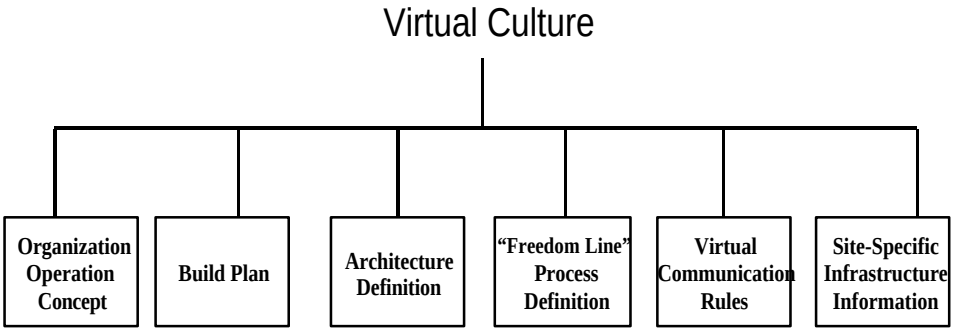


Figure 2: *Sample Virtual Culture*

planning, and the work location is still fuzzy, you can start planning right now on doing your detailed plan all over again!

Step 7: Test the Operation Concept of the Virtual Organization

It is unreasonable to expect new virtual project organizations to instantly operate as effectively as strongly cultured time-tested collocated operations. Effective organizations – collocated or virtual – do not just happen.

We recommend that newly established virtual project organizations set aside the time for project leaders to walk through key organizational scenarios that are most likely to cause leadership friction. When leaders take the time to discuss openly their visions of the virtual organization, potential problem areas can often be uncovered and resolved quickly.

Often, at the start of a new project, leaders are uncertain where the most likely trouble spots might be in the operation of the distributed organization. For example, experience has shown that when multiple organizations are involved, task management of remote personnel is a critical area. It is recommended that you walk through your task management model and your risk management model, and be sure to do it in a face-to-face setting with all your project leaders present. For more information on recommended organizational scenarios, see the referenced book [1].

Step 8: Execute

As discussed in Step 1, increased conflict is to be expected in a distributed development environment due to the lack of shared experiences and interpersonal bonds. Recognizing this fact, a key to effective process deployment on virtual projects is ensuring that leaders are aware of the warning signs of unhealthy team conflict. An example of an unhealthy conflict warning sign is what we refer to as the repeating issue warning sign. This is the case where:

- A valid issue is raised by a virtual team member.
- The issue is worked through by the team.
- A consensus is reached and the issue is put to bed.
- One week, or one month later, the same issue returns.

Does this sound familiar? Have you ever sat in a meeting and thought you were sitting through a rerun of an old movie? If you detect the repeating issue warning sign on your distributed project be sure to deal with it rapidly before it does permanent harm to your team. See reference for more examples of unhealthy team conflict warning signs, and recommended actions [1].

Deploy A Virtual Culture

In this article I have emphasized challenges being faced today on many distributed projects. I have stressed the impact of the loss of traditional informal communication in distributed environments, and have provided related recommendations. I also recommend the deployment of what is referred to as a virtual culture.

A virtual culture [1] is a simple, yet powerful concept that brings an information-age perspective to the notion of culture. Think of a virtual culture as a physical framework that supports effective communication across distributed project sites.

The virtual culture – unlike traditional collocated engineering cultures – is product oriented. It is not intended to replace past traditional collocated cultures. In fact, I don't think you should try to replace strong local cultures. Rather, my recommendations are based on leveraging the strengths of your teammates within their proven environments.

The virtual culture complements the existing site-specific cultures providing the critical information needed to coordinate and communicate key tasking information across distributed sites. This approach reduces the risk of rework when remotely developed products are integrated together. A sample structure of a virtual culture is provided in Figure 2. Virtual cultures can be implemented through a Web site or through a shared directory system.

It is worth noting that a key difference between a virtual culture and a traditional culture is found in its formality. Experience indicates that an effective virtual culture cannot be informal. In other words, it must be written down. We recognize that in today's world this emphasis on the written word may not be popular.

However, recommendations with respect to a more formal virtual culture should not be interpreted as a step backwards to the days of voluminous documentation. The virtual culture is not intended to include historical milestone-type documentation, but rather it focuses on those critical pieces of information that must be coordinated and communicated across distributed sites. Experience indicates that when you go virtual and utilize remote operations that more things do need to be written down in support of effective remote communication.

Conclusion

The potential gains of virtual operations are great. Nevertheless, implementation issues cut deep inside present-day engineering organizations. Inside traditional engineering environments, common cultures, common site infrastructures, and common experiences provide key ingredients supporting team trust.

In collocated environments, trust appears to just happen through little more than the passage of time. In reality, in these traditional environments there have always been numerous informal factors hard at work building trust on a daily basis. In the past these informal activities may not have received the attention they deserve. This is because in strongly cultured collocated environments the benefits of informality came to us essentially for free. In the virtual world this is no longer the case.

On virtual projects, the building of trust requires a more proactive management stance. It requires leaders who are willing to listen to alternative approaches put forth by team members who may

have very different backgrounds and experiences from their own. But listening is only the first step.

When alternative ideas are accepted they must also be effectively communicated to the full team. And on virtual projects, we now know we cannot rely on traditional collocated informal mechanisms to achieve this communication. Therefore, in the virtual world, the written word takes on new and increased importance.

Think about the information that is today conveyed through unplanned meetings in hallways, at lunch, casually in cubicles and over the tops of cubicles. While experience has shown that e-mail, teleconferencing, videoconferencing and NetMeeting are all powerful distributed development communication tools, they cannot replace what collocated organizations have taken years to mature. Consider deploying the virtual culture concept on your distributed project to aid communication to all your team members. It is not that expensive to implement, but the potential cost of not implementing one is.◆

References

1. McMahon, Paul E. Virtual Project Management: Software Solutions for Today and the Future. Boca Raton: St. Lucie Press, An Imprint of CRC Press LLC, 2001.
2. Gannett News Service, "Job Hopping by Young Workers Increasingly Common," 29 Aug. 1999.
3. Norton, Bob, and Cathy Smith, eds. Understanding the Virtual Organization. Hauppauge, NY: Barron's Educational Series, 1997. 68.
4. Cox, Alan. The Cox Report on the American Corporation. New York: Delacorte Press, 1982. 112-114.
5. Gindele, Mark E., and Richard Rumpf, eds. "Effects of Collocating Integrated Product Teams," Program Manager, July-Aug. 1998: 38.
6. Haywood, Martha. Managing Virtual Teams. Boston: Artech House, 1998.
7. Software Engineering Institute, <www.sei.cmu.edu/architecture/definitions.html>.
8. Shaw, Mary, and David Garlan, eds. Software Architecture: Perspectives on an Emerging Discipline. Englewood Cliffs, NJ: Prentice Hall, 1996.
9. Bass, Len, and Paul Clements, eds. Software Architecture in Practice. Reading, MA: Addison-Wesley, 1998.

Additional Reading

1. Karolak, Dale Walter. Global Software Development. Los Alamitos, CA: IEEE Computer Society, 1998: 35-46.
2. Mayer, Margery. The Virtual Edge. Newtown Square, PA: Project Management Institute Headquarters, 1998.
3. Lipnack, Jessica, and Jeffrey Stamps, eds. Virtual Teams: Reaching Across Space, Time and Organizations with Technology. New York: John Wiley & Sons, Inc., 1997.
4. Reifer, Don. Practical Software Reuse. New York: John Wiley & Sons, 1997.
5. Royce, Walker. Software Project Management. Reading, MA: Addison-Wesley, 1998.
6. Highsmith, James A. Adaptive Software Development: A Collaborative Approach To Managing Complex Systems, New York: Dorsett House Publishing, 1999.
7. Deeprose, Donna. The Team Coach. New York: American Management Assoc., 1995.

About the Author

Paul E. McMahon is an independent contractor providing technical and management leadership services to large and small engineering organizations. McMahon began his career in the early 1970's as a flight simulation programmer. Before initiating independent work at PEM Systems in 1997, he held senior technical and management positions at Hughes and Lockheed Martin. Today McMahon employs his 27 years of experience in helping organizations deploy high quality software processes integrated with systems engineering and project management. He has taught software engineering as an adjunct at Binghamton University, N.Y., and published more than a dozen articles and a book on virtual project management.

118 Matthews Street
Binghamton, NY 13905
Phone: (607) 798-7740
E-mail: pemcmahon@acm.org

Toward Adaptive and Reflective Middleware for Network-Centric Combat Systems

Dr. Douglas C. Schmidt, *University of California, Irvine*
Dr. Richard E. Schantz, *BBN Technologies*
Michael W. Masters, *Naval Surface Warfare Center*
Dr. Joseph K. Cross, *Lockheed Martin*
David C. Sharp, *The Boeing Company*
Lou P. DiPalma, *Raytheon*

Software is increasingly important to the development of effective network-centric Department of Defense combat systems. Next-generation combat systems such as total ship computing environments, coordinated unmanned air vehicle systems, and national missile defense will use many geographically dispersed sensors, provide on-demand situational awareness and actuation capabilities for human operators, and respond flexibly to unanticipated run-time conditions. These combat systems will also increasingly run unobtrusively and autonomously, shielding operators from unnecessary details while communicating and responding to mission-critical information at an accelerated operational tempo. In such environments, it is hard to predict system configurations or workloads. This article describes how adaptive and reflective middleware systems (ARMS) are being developed to bridge the gap between military application programs and the underlying operating systems and communication software in order to provide reusable services whose qualities are critical to network-centric combat systems. ARMS software can adapt in response to dynamically changing conditions for the purpose of utilizing the available computer and communication resources to the highest degree possible in support of mission needs.

New and planned Department of Defense (DoD) combat systems are inherently network-centric, distributed real-time and embedded (DRE) *systems of systems*. Combat systems have historically been developed via *multiple technology bases*, where each system brings its own networks, computers, displays, software, and people to maintain and operate it. Unfortunately, not only are these *stovepipe* architectures proprietary, but by tightly coupling many functional and quality of service (QoS) aspects they impede these DRE system features:

- *Assurability* is needed to guarantee efficient, predictable, scalable, and dependable QoS from sensors to shooters.
- *Adaptability* is needed to (re)configure combat systems dynamically to support varying workloads or missions over their life cycles.
- *Affordability* is needed to reduce initial nonrecurring combat system acquisition costs and recurring upgrade and evolution costs.

In recognition of the importance of enhancing affordability, recent DoD programs such as the Aegis destroyer program [1], the New Attack Submarine program [2], the Weapons Systems Open Architecture program [3], and the Unmanned Combat Air Vehicle (UCAV) program [4] have adopted strong open systems approaches to system design and commercial-off-the-shelf (COTS) refresh strategies. Ultimately, open systems approaches are more likely to be robust with respect to change over the long life cycles typical of military systems. For example, the affordability of certain types of DoD systems such as logistics and mission planning have been improved by using COTS technologies.

However, many of today's procurement efforts aimed at integrating COTS into mission-critical DRE combat systems have largely failed to support life-cycle affordability, assurability, and adaptability effectively since they focus mainly on initial nonrecurring acquisition costs and do not reduce recurring software life-cycle costs, such as COTS refresh and subsetting combat systems for foreign military sales [5]. Likewise, many COTS products lack support for controlling key QoS properties such as predictable latency, jitter, and throughput; scalability; dependability; and security. The inability to control these QoS properties with sufficient confidence compromises combat system adaptability and assurability, e.g., a perturbation in the behavior of a COTS product that would be acceptable in commercial applications could lead to loss of life and property in military applications.

Historically, conventional COTS software has been unsuitable for use in mission-critical DRE combat systems due to either of the following:

- It is flexible and standard, but incapable of guaranteeing stringent QoS demands, which restricts assurability.
- It is partially QoS-enabled, but inflexible and non-standard, which restricts adaptability and affordability.

As a result, the rapid progress in COTS software for mainstream business information technology (IT) has not yet become as broadly applicable for mission-critical DRE combat systems. Until this problem is resolved effectively, DRE system integrators and warfighters will not be able to take advantage of future advances in COTS software in a dependable, timely, and cost effective manner. Developing the new generation of assurable, adaptable, and affordable COTS software technologies is therefore essential for U.S. national security.

Although the near-term use of COTS software in DRE systems will be limited in scope and domain, the prospects for the longer term are much brighter. Given the proper advanced research and development (R&D) context and an effective process for transitioning R&D results, the COTS market can adapt, adopt, and implement the types of robust hardware and software capabilities needed for military applications. This process takes a good deal of time to get right and be accepted by user communities, and a good deal of patience to stay the course. When successful, however, this process results in standards that codify the best-of-breed practices and technologies and the *patterns* and *frameworks* that reify the knowledge of how to apply these practices and technologies.

Key Technical Challenges and Solutions

Today's economic and organizational constraints – along with increasingly complex requirements and competitive pressures – make it infeasible to build complex distributed real-time system software entirely from scratch. It has long been accepted that the use of commercial operating systems and communication support software is cost-effective for all but the most

resource-constrained DRE systems. Increasingly, this same logic is being applied to middleware, which is reusable service/protocol component and framework software that services end-to-end and aggregate combat systems' needs [6]. Middleware bridges the gap between these areas:

- Application-level requirements and mission doctrine.
- The lower-level, underlying, localized viewpoints of the operating systems and communications support mechanisms.

From the application perspective, when middleware and the services it constitutes are combined with traditional network and operating system components, it forms the new infrastructure for developing modern network-centric combat systems. In both commercial and military systems, middleware performs functions that are essential to meeting application-level requirements. In military systems, moreover, the qualities of the services provided by the middleware are critical to the qualities of service that are presented to the end users – the warfighters.

Thus, there is a pressing need to develop, validate, and ultimately standardize a new generation of adaptive and reflective middleware systems (ARMS) technologies that will be readily available and able to support stringent combat system functionality and QoS requirements. Some of the most challenging computing and communication requirements for new and planned DoD combat systems can be characterized as follows:

- Multiple QoS properties must be satisfied in real-time.
- Different levels of service are appropriate under different configurations, environmental conditions, and costs.
- The levels of service in one dimension must be coordinated with and/or traded off against the levels of service in other dimensions to meet mission needs, e.g., the security and dependability of message transmission must be traded off against latency and predictability.
- The need for autonomous and time-critical application behavior necessitates a flexible distributed system substrate that can adapt robustly to dynamic changes in mission requirements and environmental conditions.

Adaptive middleware [3] is software whose functional and QoS-related properties can be modified in either of these ways:

- Statically, e.g., to reduce footprint, leverage capabilities that exist in specific platforms, enable functional subsetting, and minimize hardware and software infrastructure dependencies.
- Dynamically, e.g., to optimize system responses to changing environments or requirements, such as changing component interconnections, power-levels, CPU/network bandwidth, latency/jitter, and dependability needs.

In DRE combat systems, adaptive middleware must make these modifications dependably, i.e., while meeting stringent end-to-end QoS requirements.

Reflective middleware [7] goes a step further in providing the means for examining the capabilities it offers while the system is running, thereby enabling automated adjustment for optimizing those capabilities. Thus, reflective middleware supports more advanced adaptive behavior, i.e., the necessary adaptations can be performed autonomously based on conditions within the system, in the system's environment, or in combat system doctrine defined by operators and administrators.

Middleware Structure and Functionality

Networking protocol stacks can be decomposed into multiple layers such as the physical, data-link, network, transport, session, presentation, and application layers. Similarly, middleware can be decomposed into multiple layers such as those shown in Figure 1.

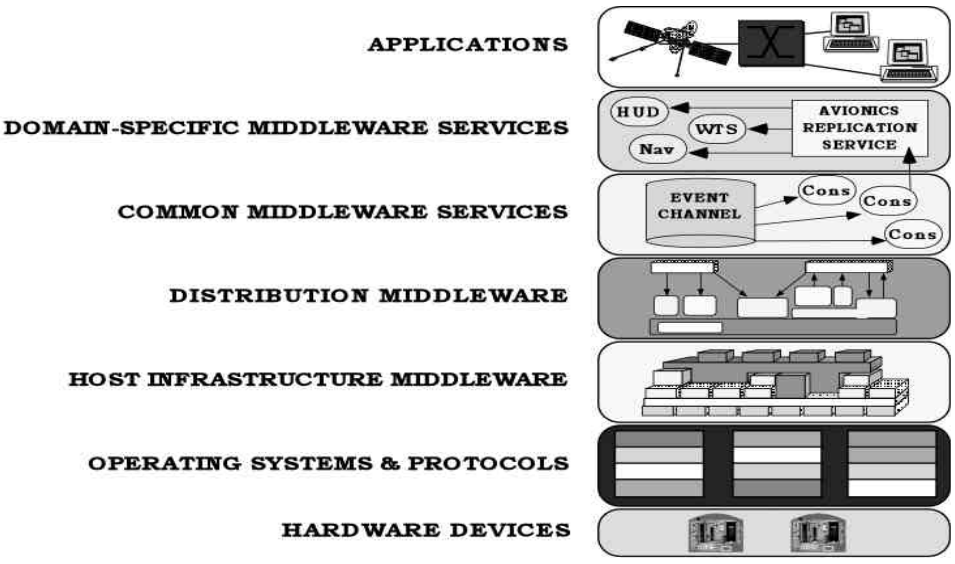
We describe each of these middleware layers and outline some of the COTS technologies in each layer that are suitable (or are becoming suitable) to meet the stringent QoS demands of DRE combat systems.

Host Infrastructure Middleware

Host infrastructure middleware encapsulates and enhances native operating system communication and concurrency mechanisms to create portable and reusable network programming components such as reactors, acceptor-connectors, monitor objects, active objects, and component configurations [8]. These components abstract away the accidental incompatibilities of individual operating systems and help eliminate many tedious, error-prone, and non-portable aspects of developing and maintaining networked applications via low-level operating system programming application program interfaces (APIs), such as Sockets or POSIX Pthreads. Examples of COTS host infrastructure middleware that are relevant for DRE combat systems include the following:

- The Adaptive Communication Environment (ACE) [9] is a portable and efficient toolkit that encapsulates native operating system network programming capabilities such as inter-process communication, static and dynamic configuration of application components, and synchronization. ACE has been used in a wide range of DoD DRE systems, including missile control, avionics mission computing, software defined radios, and radar systems.
- Real-Time Java Virtual Machines implement the Real-Time Specification for Java (RTSJ) [10]. The RTSJ is a set of extensions to Java that provide a largely platform-independent way of executing code by encapsulating the differences between real-time operating systems and CPU architectures. The key features of RTSJ deal with memory man-

Figure 1: *Middleware Layers and Their Surrounding Context*



agement and concurrency. Although RTSJ implementations are still in their infancy, they have generated tremendous interest in the DoD R&D and integrator communities due to their potential for reducing software development and evolution costs significantly.

Distribution Middleware

Distribution middleware defines a higher-level distributed programming model whose reusable application program interfaces and mechanisms automate and extend the native operating system network programming capabilities encapsulated by host infrastructure middleware. Distribution middleware enables developers to program distributed applications much like stand-alone applications, i.e., by invoking operations on target objects or distributed components.

At the heart of distribution middleware are QoS-enabled object request brokers, such as the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [4, 11]. CORBA is distribution middleware that allows objects to interoperate across networks without hard-coding dependencies on their location, programming language, operating system platform, communication protocols and interconnects, and hardware characteristics. In 1998 the OMG adopted the Real-Time CORBA specification [12], which extends CORBA with features that allow DRE applications to reserve and manage CPU, memory, and networking resources. Real-Time CORBA implementations have been used in dozens of DoD combat systems, including avionics mission computing [4], submarine combat control systems [13], signal intelligence and Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance systems, software defined radios, and radar systems.

Common Middleware Services

Common middleware services augment distribution middleware by defining higher-level, domain-independent, reusable services that have proven necessary in most distributed application contexts to deal with multi-computer environments effectively. In addition, these services provide components that allow application developers to concentrate on programming application logic, without the need to write the plumbing code needed to develop distributed applications using lower level middleware features directly.

For example, whereas distribution middleware focuses largely on managing end-system resources in support of an object-oriented distributed programming model, common middleware services focus on allocating, scheduling, and coordinating various end-to-end resources throughout a distributed system using a component programming and scripting model. Developers can reuse these services to manage global resources and perform recurring distribution tasks that would otherwise be re-implemented by each application or integrator.

Examples of common middleware services include the OMG's CORBAServices [14] and the CORBA Component Model (CCM) [15], which provide domain-independent interfaces and distribution capabilities that can be used by many distributed applications. The OMG CORBAServices and CCM specifications define a wide variety of these services, including event notification, naming, security, and fault tolerance. Not all of these standard services are sufficiently refined today to be usable off the shelf for DRE combat systems. However, the form and content of these common middleware services will continue to mature and evolve to meet the expanding requirements of DRE.

> "Today's economic and organizational constraints – along with increasingly complex requirements and competitive pressures – make it infeasible to build complex distributed real-time system software entirely from scratch."

Domain-Specific Middleware Services

Domain-specific middleware services are tailored to the requirements of particular combat system domains, such as avionics mission computing, radar processing, weapons targeting, or command and decision systems. Unlike the previous three middleware layers – which provide broadly reusable horizontal mechanisms and services – domain-specific middleware services are targeted at vertical market segments. From a COTS perspective, domain-specific services are the least mature of the middleware layers today. This immaturity is due in part to the historical lack of distribution middleware and common middleware service standards, which are needed to provide a stable base upon which to create domain-specific middleware services. Since they embody knowledge of a domain, however, domain-specific middleware services have the most potential to increase the quality and decrease the cycle time and effort that DoD integrators require to develop particular classes of DRE combat systems.

A mature example of domain-specific middleware services appears in the Boeing Bold Stroke architecture [4]. Bold Stroke uses COTS hardware and middleware to produce a non-proprietary, standards-based component architecture for military avionics mission computing capabilities, such as navigation, data link management, and weapons control. A driving objective of Bold Stroke was to support reusable product-line applications, leading to a highly configurable application component model and supporting middleware services. The domain-specific middleware services in Bold Stroke are layered upon common middleware services (the CORBA Event Service), distribution middleware (Real-Time CORBA and the tactical air operations object request broker [16]), and infrastructure middleware advanced computing environment, and have been demonstrated to be highly portable for different COTS operating systems (e.g., VxWorks), interconnects (e.g., VME), and processors (e.g., PowerPC).

Recent Progress

Significant progress has occurred during the last five years in DRE middleware research, development, and deployment within the DoD, stemming in large part from the following trends:

Maturation of Standards

During the past decade, middleware standards have been established and have matured considerably with respect to DRE requirements. For example, the OMG has recently adopted the following DRE-related specifications:

- Minimum CORBA removes non-essential features from the full OMG CORBA specification to reduce footprint so that CORBA can be used in memory-constrained embedded systems.

- Real-Time CORBA includes features that allow applications to reserve and manage network, CPU, and memory resources predictably end to end.
- CORBA Messaging exports additional QoS policies such as timeouts, request priorities, and queuing disciplines to applications.
- Fault-Tolerant CORBA uses entity redundancy of objects to support replication, fault detection, and failure recovery.

Robust and interoperable implementations of these CORBA capabilities and services are now available from multiple vendors. Moreover, emerging standards such as Dynamic Scheduling Real-Time CORBA, Real-Time CORBA publish-subscribe services, the Real-Time Specification for Java, and the Distributed Real-Time Specification for Java are extending the scope of open standards for a wider range of DoD applications.

Dissemination of Patterns, Frameworks

A substantial amount of R&D effort during the past decade has also focused on the following means of promoting the development and reuse of high quality middleware technology:

- Patterns codify design expertise that provides time-proven solutions to commonly occurring software problems that arise in particular contexts [17]. Patterns can simplify the design, construction, and performance tuning of DRE applications by codifying the accumulated expertise of developers, architects, and systems engineers who have already confronted similar problems successfully.
- Frameworks are concrete realizations of related patterns [18] that provide an integrated set of components that collaborate to provide a reusable architecture for a family of related applications. Middleware frameworks include strategized selection and optimization patterns so that multiple, independently developed capabilities can be integrated and configured automatically to meet the functional and QoS requirements of particular DRE applications.

Historically, the knowledge required to develop predictable, scalable, efficient, and dependable mission-critical DoD DRE combat systems has existed largely in programming folklore, the heads of experienced researchers and developers, or buried deep within millions of lines of complex source code. Moreover, documenting complex systems with today's popular software modeling methods and tools, such as the Unified Modeling Language (UML), only capture how a system is designed, but do not necessarily articulate why a system is designed in a particular way, which complicates subsequent software evolution and optimization.

Middleware patterns and frameworks help address these problems by systematically capturing combat system design expertise in a readily accessible and reusable format, thereby raising the level at which systems engineers and application developers approach the decision making and implementation of their systems. Two efforts to provide suitable guidance for the development of military systems are the New Attack Submarine (NAS) [2] and the Aegis Shipbuilding Program. NAS developed a guidance document detailing allowable standards for the NAS C3I system, and the Aegis program developed a guidance document for Baseline 7 phase I [19]. These documents were instrumental in guiding the design of these systems.

> "Given the proper advanced R&D context and an effective process for transitioning R&D results, the COTS market can adapt, adopt, and implement the types of ... capabilities needed for military applications ..."

Much of the pioneering R&D on middleware patterns, frameworks, and standards for DRE combat systems has been conducted as part of the Defense Advanced Research Projects Agency's (DARPA) Information Technology Office Quorum Program [20], which played a leading role in the following:

- Demonstrating the viability of host infrastructure middleware and distribution middleware for DoD combat systems by providing the foundation for managing key QoS attributes such as real time behavior, dependability, and system survivability from a network-centric middleware perspective.
- Transitioning a number of new middleware perspectives and capabilities into DoD acquisition programs [4, 21] and commercially supported products.
- Establishing the technical viability of collections of systems that can dynamically adapt [3] their collective behavior to varying operating conditions, in service of delivering the appropriate application level response under these different conditions.

The Quorum program focused heavily on CORBA open systems middleware and yielded many results that transitioned into standardized service definitions and implementations for the Real-Time [4] and Fault-Tolerant [22] CORBA specification and production. Quorum is an example of how a focused government R&D effort can leverage its results by exporting them into, and combining them with, other on-going public and private activities by using a common open middleware substrate. Prior to the viability of standards-based COTS middleware platforms, these same R&D results would have been buried within custom or proprietary systems, serving only as an existence proof, rather than as the basis for realigning the DoD R&D and integrator communities.

Successful DoD technology transition most often results from a partnership between technology developers and technology users. One of the most successful examples of such partnerships is the joint DARPA/Aegis High Performance Distributed Computing program (HiPer-D). Through the use of prototyping and system-scale experiments, this program has demonstrated the effectiveness of a number of DARPA and standards-based COTS technologies for building DRE combat systems that are efficient, scalable, fault tolerant, and flexible in their design and operation.

Looking Ahead

Due to advances in COTS technologies outlined earlier, host infrastructure middleware and distribution middleware have now been demonstrated and deployed in a number of mission-critical DRE combat systems. Since off-the-shelf middleware technology has not yet matured to cover the realm of large-scale dynamically changing systems, however, COTS DRE middleware has been applied to relatively small-scale and statically configured embedded systems. To satisfy the highly application- and mission-specific QoS requirements in network-centric *system-to- system environments*, DRE middleware

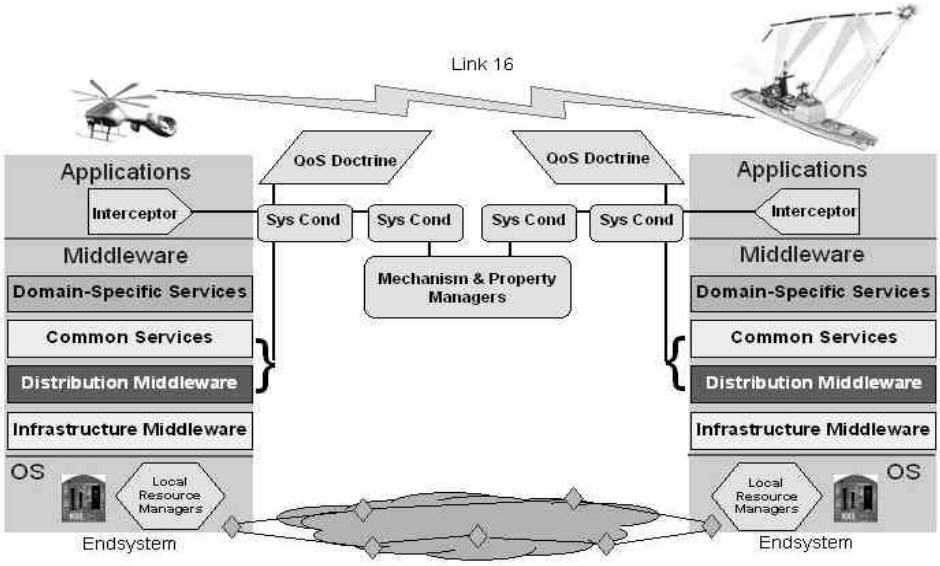


Figure 2: *Decoupling Functional and QoS Attribute Paths in QuO*

must therefore be enhanced to support common and domain-specific middleware services that can manage the following resources effectively:

- Communication bandwidth, e.g., network level status information and management services, scalability to 102 subnets and 103 nodes, and dynamic connections with controlled and reserved bandwidth to enhance real-time predictability.
- Distributed real-time scheduling and allocation of DRE system artifacts (such as CPUs, networks, UAVs, missiles, torpedoes, radar, illuminators, etc), e.g., fast and predictable behavior of widely dispersed components that use the managed communication capabilities and bandwidth reservations.
- Distributed system dependability, e.g., policy-based selection of replication options to control footprint and reactive behavior to failures.
- Distributed system security, e.g., dynamically variable object access control policies and effective, combined real-time dependability, and security interactions.

Ironically, there is little or no scientific underpinning for QoS-enabled resource management, despite the demand for it in most distributed systems [23]. Today's system designers develop concrete plans for creating global, end-to-end functionality. These plans contain high-level abstractions and doctrine associated with resource management algorithms, relationships between these, and operations upon these. There are few techniques and tools, however that enable users, i.e., commanders, administrators, and operators, and developers, i.e., systems engineers and application designers, and/or applications to express such plans systematically, reason about and refine them, and have these plans enforced automatically to manage resources at multiple levels in network-centric combat systems.

To address this problem, the R&D community needs to discover and set the technical approach that can significantly improve the effective utilization of networks and end-systems that DRE combat systems depend upon by creating middleware and distributed resource management technologies and tools that can automatically allocate, schedule, control, and optimize customizable – yet standards-compliant and verifiably correct – software-intensive systems. To promote a *common technology* base, the interfaces and (where appropriate) the protocols used by the middleware should be based on established or emerging industry or DoD standards that are relevant for DRE combat systems. However, the protocol and service *implementations* should be customizable – statically and dynamically – for specific DoD DRE combat system requirements.

To achieve these goals, middleware technologies and tools need to be based upon some type of layered architecture along with QoS adaptive middleware services such as the one shown in Figure 2 and based on empirical investigations of this type of capability [3].

The Quality Objects (QuO) [24] project is an example of such a layered architecture designed to manage and package adaptive QoS capabilities as common middleware services. The QuO architecture decouples DRE middleware and applications along the following two dimensions:

- Functional paths are flows of information between client and remote server applications. In distributed systems, middleware ensures that this information is exchanged efficiently, predictably, dependably, and securely between remote peers, and in a manner that scales well to large configurations. The information itself is largely application-specific and determined by the functionality being provided (hence the term *functional path*).
- QoS attribute paths are responsible for determining how well the functional interactions behave end to end with respect to key DRE system QoS properties such as the following:
  1. How and when resources are committed to client/server interactions at multiple levels of distributed systems.
  2. The proper application and system behavior if available resources are less than the expected resources.
  3. The failure detection and recovery strategies necessary to meet end-to-end dependability requirements under anomalous conditions.

In next-generation combat systems, the middleware – rather than operating systems or networks in isolation – will be responsible for separating DRE system QoS attribute properties from the functional application properties. Middleware will also coordinate the QoS of various DRE system and application resources end to end. The architecture in Figure 2 enables these properties and resources to change independently, e.g., over different distributed system configurations for the same application.

The architecture in Figure 2 is based on the expectation that QoS attribute paths will be developed, configured, monitored, managed, and controlled by a different set of specialists (such as systems engineers, administrators, operators, and perhaps someday automated agents) and tools than those customarily responsible for programming functional paths in DRE systems. The middleware is therefore responsible for collecting, organizing, and disseminating QoS-related meta-information that is needed to do the following:

- Monitor and manage how well the functional interactions occur at multiple levels of DRE systems.
- Enable the adaptive and reflective decision-making needed to support QoS attribute properties robustly in the face of rapidly changing mission requirements and environmental conditions.

Researching and developing these middleware capabilities is crucial to ensure that the aggregate behavior of future network-centric combat systems is dependable, despite local failures, transient overloads, and dynamic functional or QoS reconfigurations.

To simultaneously enhance assurability,

adaptability, and affordability, the middleware techniques and tools developed in future R&D programs increasingly need to be application-independent, yet customizable within the interfaces specified by a range of open standards such as these:

- The OMG Real-Time CORBA specifications and The Open Group's QoS Task Force.
- The Java Expert Group Real-Time Specification for Java (RTSJ) and the emerging Distributed RTSJ.
- The IEEE Real-Time Portable Operating System (POSIX) specification.

Conclusions

As a result of much previous R&D and transition experience, network-centric systems today are constructed as a series of layers of intertwined technical capabilities and innovations. The main emphasis at the lower layers is in providing the core computing and communication resources and services that drive network-centric computing: the individual computers, the networks, and the operating systems that control the individual host and the message level communication.

At the upper layers, various types of middleware are starting to bridge the previously formidable gap between the lower-level resources and services and the abstractions that are needed to program, organize, and control systems composed of coordinated, rather than isolated, components. Key capabilities in the upper layers include common and domain-specific middleware services that provide the following:

- Enforcing real-time behavior across computational nodes.
- Managing redundancy across elements to support dependable computing.
- Controlling end-to-end adaptive behavior in responding to changes in operating conditions while continuing to meet application needs.

These new middleware services make the coordinated use of multiple computing elements feasible and affordable by controlling the hardware, network, and end-system mechanisms that affect mission, system, and application QoS delivery and tradeoffs that are needed to deliver the right QoS at the right time under the prevailing conditions.

Adaptive and reflective middleware systems (ARMS) is a key emerging paradigm that will help to simplify the development, optimization, validation, and integration of DRE middleware in DoD combat systems. In particular, ARMS will allow researchers and system integrators to develop and evolve complex combat systems assurably, adaptively, and affordably through the following:

- Devising optimizers, meta-programming techniques, and multi-level distributed dynamic resource management protocols and services for ARMS that will enable DoD DRE systems to configure standard COTS interfaces without the penalties incurred by today's conventional COTS software product implementations. Many network-centric DoD combat systems require these DRE middleware capabilities.
- Standardizing COTS at the middleware level, rather than just at lower hardware/networks/operating system levels. The primary economic benefits of middleware stem from extending standardization up several levels of abstraction so that DRE middleware technology is readily available for COTS acquisition and customization.

As COTS implementations of middleware standards mature in their functional quality and QoS, they are helping to lower the total ownership costs of combat systems. For example, Real-Time and Fault-Tolerant CORBA implementations are creating a common base of COTS technology that enables complex DRE middleware capabilities to be reconfigured and reused, rather than reinvented repeatedly or reworked from proprietary stovepipe architectures that are inflexible and expensive to maintain, evolve, and optimize. Additional information on middleware for DRE systems is available at <www.ece.uci.edu/~schmidt/TAO.html>.◆

References

1. Holzer, R. "U.S. Navy Looking for More Adaptable Aegis Radar," Defense News 18 Sept. 2000.
2. New Attack Submarine Open System Implementation, Specification and Guidance, Aug. 1994.
3. Loyall J. L., et al. "Comparing and Contrasting Adaptive Middleware Support in Wide-Area and Embedded Distributed Object Applications." Proceedings of the 21st IEEE International Conference on Distributed Computing Systems. Phoenix, AR. 16-19 Apr. 2001.
4. Sharp, David C. "Reducing Avionics Software Cost Through Component Based Product Line Development," Software Technology Conference. Salt Lake City, Apr. 1998.
5. Clapp, J., and A. Taub, eds. A Management Guide to Software Maintenance in COTS-Based Systems MP 98B0000069. Bedford, MA: The MITRE Corporation, Nov. 1998.
6. Schantz, R., and D. Schmidt. "Middleware for Distributed Systems: Evolving the Common Structure for Network-Centric Applications." Encyclopedia of Software Engineering. Wiley & Sons, 2001.
7. Blair, G. S., F. Costa, G. Coulson, and H. Duran, et al. "The Design of a Resource-Aware Reflective Middleware Architecture." Proceedings of the 2nd International Conference on Meta-Level Architectures and Reflection. St.-Malo, France: Springer-Verlag, LNCS, Vol. 1616, 1999.
8. Schmidt D., M. Stal, H. Rohnert, and F. Buschmann F., eds. Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. Wiley and Sons, 2000.
9. Schmidt D., and S. Huston, eds. C++ Network Programming: Resolving Complexity with ACE and Patterns. Reading, MA: Addison-Wesley, 2002.
10. Bollella, G., and J. Gosling. "The Real-Time Specification for Java." Computer June 2000.
11. Object Management Group, The Common Object Request Broker: Architecture and Specification Rev. 2.4. OMG Technical Document formal/00-11-07, Oct. 2000.
12. Schmidt, D., and F. Kuhns, eds. "An Overview of the Real-Time CORBA Specification." IEEE Computer Magazine June 2000.
13. DiPalma, L., "The Infusion of CORBA into the U.S. Navy's Submarine Fleet," Software Technology Conference. Salt Lake City, May 1999.
14. Object Management Group. CORBAServices: Common Object Service Specification. OMG Technical Document Formal/98-12-31.
15. Object Management Group, CORBA Component Model Joint Revised Submission. OMG Document orbos/99-07-01.
16. Schmidt D., Levine D., and Mungee S., eds. "The Design and Performance of the TAO Real-Time Object Request Broker," Computer Communications Special Issue on Building Quality of Service into Distributed Systems 21.4 (1998).
17. Gamma E., R. Helm, R. Johnson, J. Vlissides, eds. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.
18. Johnson R., "Frameworks = Patterns + Components." Communications of the ACM 40.10, Oct. 1997.
19. Guidance Document for Aegis Baseline 7 Phase 1 and II Specification Development: Information Architecture and Baseline Applicability, Ver. 1.0, 20 Mar. 1998.
20. The Quorum Program, Defense Ad-

vanced Research Projects Agency, <www.darpa.mil/ito/research/quorum/index.html> 1999.
21. Guidance Document for Aegis Open Architecture Baseline Specification Development, Ver. 2.0 (Draft), 5 July 2001.
22. Object Management Group. Fault Tolerance CORBA Using Entity Redundancy RFP. OMG Document orbos/98-04-01 edition, 1998.
23. Narain S., R. Vaidyanathan, S. Moyer, W. Stephens, K. Parameswaran, and A. Shareef, eds. "Middleware For Building Adaptive Systems via Configuration," Workshop. ACM Optimization of Middleware and Distributed Systems (OM 2001), Snowbird, UT. June 2001.
24. Zinky, J.A., D.E. Bakken, and R.E. Schantz, eds. "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems 3.1, Apr. 1997.

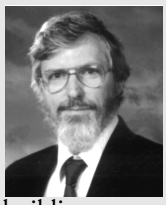
About the Authors

Douglas C. Schmidt, Ph.D., is an associate professor in the Electrical and Computer Engineering Department at the University of California, Irvine. He currently serves as deputy director of the Defense Advanced Research Projects Agency's Information Technology Office, where he leads the national effort on distributed object computing middleware research and development. His research focuses on design patterns, implementation, and experimental analysis of object-oriented frameworks that facilitate the development of high-performance, real-time distributed object computing systems on parallel processing platforms running over high-speed networks and embedded system interconnects.

Electrical and Computer
Engineering Department
University of California, Irvine
Irvine, CA 92697-2625
Phone: (949) 824-1901
Fax: (949) 824-2321
E-mail: schmidt@uci.edu

Richard E. Schantz, Ph. D., is a principal scientist at BBN Technologies in Cambridge, Mass. His research has been instrumental in defining and evolving the concepts underlying middleware since its emergence in the early days of the Internet. He was directly responsible for developing the first operational distributed object computing capability and transitioning it to production use. More recently, he has led research efforts toward developing and demonstrating the effectiveness of middleware support for adaptively managed Quality Of Service control, as principal investigator on a number of key Defense Advanced Research Projects Agency, Information Technology Office projects.

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Phone: (617) 873-3550
Fax: (617) 873-4328
E-mail: schantz@bbn.com

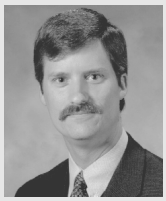


Michael W. Masters serves as chief scientist for the U.S. Navy's High Performance Distributed Computing program (HiPer-D), a joint effort between the Aegis shipbuilding program and several Defense Advanced Research Projects Agency, Information Technology Office programs. HiPer-D is defining a new distributed real-time computing architecture for shipboard use. Masters is co-inventor of a technology called dynamic resource management, an enterprise-wide system control capability that allows large-scale real-time systems to dynamically reconfigure themselves to adapt to varying environments, changing mission demands and current resource availability.

NSWC Dahlgren Division
17320 Dahlgren Road,
Dahlgren, VA 22448-5100
Phone: (540) 653-1611
Fax: (540) 653-6415
E-mail: mastersmw@nswc.navy.mil

Joseph K. Cross, Ph.D., is a senior staff system engineer at Lockheed Martin Tactical Systems, Eagan, Minn. He is currently serving as principal investigator of the Meta-Interfaces for Embedded Real-Time Systems project in Defense Advanced Research Projects Agency, Information Technology Office. His other activities focus on middleware for Navy standard products and mechanisms for automatic configuration of complex communication systems.

Lockheed Martin Tactical Systems
P.O. Box 64525 MS U2N27
St. Paul, MN 55164-0525
Phone: (651) 456-7316
Fax: (651) 456-2078
E-mail: joseph.k.cross@lmco.com



David C. Sharp is a Technical Fellow at Boeing Phantom Works in St. Louis, Mo. As lead architect and core architecture team leader for Boeing's Bold Stroke product line avionics software initiative, Sharp spearheaded the development, documentation, and presentation of the Bold Stroke Software Architecture. This reusable product-line software architecture is used as the basis for avionics program work on a range of Boeing production and experimental aircraft programs, and as the foundation for several U.S. government-sponsored research and development programs. Sharp serves as principal investigator for a number of Defense Advanced Research Projects Agency, Information Technology Office and Air Force Research Laboratory programs.

The Boeing Company
P.O. Box 516
St. Louis, MO 63166
Phone: (314) 233-5628
Fax: (314) 233-8323
E-mail: david.sharp@boeing.com

Lou P. DiPalma is the manager of the Sub-Surface Warfighter Information Center Systems Engineering Department of the Portsmouth, R.I. headquarters of the Naval & Maritime Integrated Systems Operation of the Raytheon Electronic Systems Company. DiPalma has been involved in the design and development of Submarine Combat Control Systems including the New Attack Submarine CC, the Combat Control System Mk 2 and AN/BSG-1 Weapon Launching System Programs. He has been actively involved with the infusion of new technology into the aforementioned systems, including Common Object Request Broker Architecture (CORBA) and Real-Time CORBA.

Raytheon Electrical Systems
Naval and Maritime Integrated Systems
1847 West Main Road
Portsmouth, RI 02871
Phone: (401) 842-5592
Fax: (401) 842-5232
E-mail: louis_p_dipalma@raytheon.com

Factors to Consider When Selecting CORBA Implementations

Dr. Thomas J. Croak
Computer Sciences Corporation

Performance and flexibility are old rivals in computer architecture. Common Object Request Broker Architecture (CORBA) certainly provides flexibility, but at what cost to performance? This article summarizes middleware research resulting in the identification of "10 dimensions of variability" in software architecture that can cause each implementation of CORBA to behave differently in a given system. The resulting factors can be used to choose a CORBA Object Request Broker implementation, and then tune it to the specific architecture.

All distributed software architectures require a mechanism to exchange data between the nodes of the architecture. A generic term for this class of software is *middleware*. Common Object Request Broker Architecture (CORBA) is an open standard for object-oriented middleware developed by the Object Management Group, a consortium of nearly 800 companies. A CORBA-compliant Object Request Broker (ORB) can facilitate data communication between dissimilar hardware, dissimilar operating systems, and modules written in different languages. CORBA is a Joint Technical Architecture (JTA) endorsed standard, and is a part of the Defense Information Infrastructure (DII) Common Operating Environment (COE).

Command and control (C2) systems must be able to provide decision support information in response to the battle as it is taking place and therefore must be very efficient. The ability of a system to meet performance requirements depends heavily on the underlying architecture selected for the software system. Middleware is a critical performance component. Most C2 systems being built to JTA and DII COE standards have CORBA as their middleware. There are many choices of CORBA ORB implementations, and they can each affect performance differently. The basic question then is how does the architect or designer select the best implementation of a CORBA ORB given the performance needs of the system?

This article summarizes middleware research resulting in the identification of 10 *dimensions of variability* in software architecture that can affect a CORBA ORB's behavior for a given system [1]. The results of this research can aid in determining what factors of a specific architecture affect performance when using CORBA. These factors include aspects of the selected hardware architecture that would favor one CORBA implementation over another, and details about how CORBA works that affect the performance of the architecture. The understanding of these factors can help quantify the needs of the architecture, and in turn help evaluate candidate implementations of CORBA.

A secondary benefit of this research is that knowledge of these factors can be used to tune existing systems or systems under construction that have already selected a CORBA product. Most CORBA products have parameters that can be tuned by the system designer to complement the system's architecture and improve performance.

It Is Like Buying a Car

The analogy to this situation is purchasing an automobile. Some people select a vehicle based on features or options without much consideration of how the vehicle will be used. Many CORBA ORBs are purchased the same way. This is possibly because both cars and CORBA ORBs have extensive marketing literature focused on features and options.

A better approach is to perform a needs analysis to determine which vehicle factors are most important, and then compare vehicle choices directly using common units of measure for those factors. The first step of a needs analysis would be to determine what the primary purpose of the vehicle will be. What is secondary? How many people should it hold? What else will be carried in the vehicle? Should it be able to tow a camping trailer or a boat trailer? The next step is to select factors for comparison such as acceleration, gas mileage, towing capacity, and range. There are a series of standard units of measure that can be used to help make this comparison. Acceleration can be compared using the time to go from 0 to 60 mph. Fuel efficiency can be compared using the estimate of mpg. Towing capacity can be expressed in terms of both dead weight and tongue weight. Range is simply the fuel tank size times the mpg estimate.

How can all of these factors be optimized? Basically, it cannot be done. The only possible result would be a vehicle that was not good at anything. The best solution is to determine which factor is the most important to the need and optimize that factor. Fortunately for car buyers, there is a whole series of generally understood units of measure for comparison. The factors to consider when selecting CORBA are not nearly as well understood, and there are few if any generally accepted units of measure for comparison.

Command and Control System Needs Analysis

The first step toward selecting a CORBA implementation should be performing a rudimentary needs analysis to understand the constraints on the design of the system. While a family vehicle must satisfy many different needs, often loosely defined, there is a much better situation for C2 systems. A C2 system is often designed to support a single mission or a closely related group of missions. The requirements for such a system are generally clearly stated in terms that are quantifiable and testable. This statement of need is often referred to in computer science terminology as the *service policy* for the system.

To better understand the service policy for a C2 system, it may be useful to compare it with the service policy for a business system. This is done for the simple reason that most CORBA implementations are actually designed for the business environment. A rudimentary understanding of this environment is important to selecting a CORBA implementation for C2 systems.

A typical service policy at a bank might say: "Ninety-five percent of the transactions must be processed within 10 seconds." Notice that there is no stated requirement for the other 5 percent, and that the time constraint is fairly loose and could be easily met by several architecture

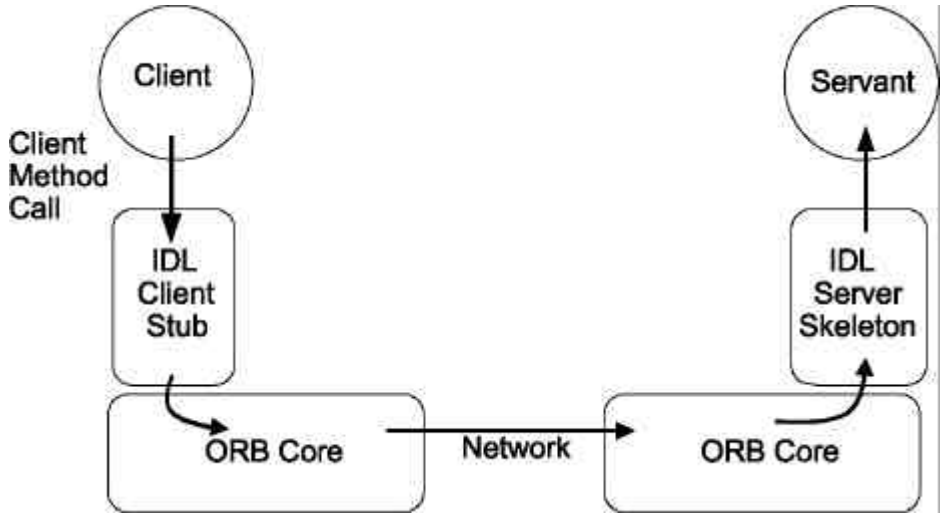


Figure 1: *Generalized Flow of a CORBA Transaction*

configurations. A bank's service policy would most likely be derived from statistical analysis of marketing data. It would show customer arrival rate and how long customers are willing to stand in line, or how long they are willing to stand in front of an ATM while waiting for the transaction to be authorized. Cost per transaction seems to permeate many of the calculations. The bank is looking to reduce the cost per transaction without losing customers.

The bank's service policy might seem very foreign to someone used to working with C2 systems. A service policy for a C2 system might be stated: "All transactions must be processed within 80 milliseconds (msec) while using no more than 50 percent of available processing capacity." This statement is different from the bank's service policy in three important ways. All transactions are covered, not just most of them. The time constraint is far shorter. And perhaps most importantly, a built-in reserve capacity has been stated.

The cost reasonableness for these systems is not based on a per transaction basis, but rather the cost of regrets if the mission fails due to an inefficient computational suite. The advantage the C2 system designer has over the business system designer is that the focus can be on optimizing the system to process a single transaction. That transaction does not even have to be a typical one. It may be the one that is most time-critical or has the highest use of compute power. This is an advantage because the designer can break down that transaction into its component parts and find the processing bottlenecks without having to consider arrival rate statistics. These component parts include the processing steps within the CORBA transaction.

Components of the CORBA ORB Transaction

The basic building block in CORBA is the CORBA object. A CORBA object models a real-world object and consists of the data and the methods that it may invoke. The object's public interface is through the CORBA Interface Definition Language (IDL). The purpose of the IDL is to hide the underlying object's implementation details. Any client of the object can use this interface to invoke the methods of the object without knowing any of the details of the implementation, the platform it is on, or the location of the object [2]. There are IDL implementations for Ada, C++, JAVA, Smalltalk, and several other languages.

The heart of CORBA is the ORB. The ORB acts as the middleware component that implements the conceptual bus between the client and the servant. The ORB ensures delivery of the client's request, while hiding the location and implementation details from the client. Additionally, in its broker role, if there are multiple server components that can perform the method, the ORB performs a load balancing function between them [3]. ORBs are optimized and tested thoroughly resulting in the virtual elimination of many tedious, error-prone aspects of creating and managing distributed applications, while increasing the portability and reusability of the service components [4].

While the ORB is far more complicated than the simple illustration in Figure 1, it contains the principal components involved in the transaction. For more detail and a look at the entire CORBA reference model, check out the Object Management Group's Web site at <www.omg.org>. The general flow of an ORB transaction consists of the following steps. The client object invokes a method call on the servant as if it were performing the call directly to another object. The IDL Stub is the public representation of that method and intercepts the call. The ORB core performs the data conversion functions, the brokering function, and the communications function. The IDL skeleton represents the client to the servant method call.

The CORBA IDL uses the concepts of stubs and skeletons as the glue between the client and servants respectively, and the ORB. Stubs provide either a strongly typed static invocation interface or a more weakly typed dynamic invocation interface, that is used to marshal the client data into the ORB's common packet-level representation of the data. The skeleton does the reverse by taking the packet-level representation and demarshals it back into typed data that is meaningful to the servant.

This stub/skeleton approach hides the complexity of the low-level communication between the client and the server and facilitates the interaction between modules that may have been coded in different languages and between dissimilar hardware representations of the data. An IDL compiler for the language of the component automates the transformation between the CORBA IDL definitions and the target programming language. By performing this automated step, the IDL compiler also greatly reduces the potential for inconsistencies between the client stubs and the server skeletons. Additionally, the compiler eliminates common sources of network programming errors and provides opportunities for automated compiler optimizations [5] and [6].

The CORBA ORB Core provides the brokering role as well as all the communication facilities for sharing resources among processes. The core translates the logical address for the method call presented to it by the client into a physical address and performs all the steps required to ship the data to the corresponding ORB core on the server. That portion of the core performs the other half of the communications protocol and delivers the information to the skeleton. Some ORB cores are optimized to know that if the called servant is on the same computer, many of the communications steps can be skipped.

Dimensions of Variability

The length of time to perform an ORB transaction varies greatly with the implementation of the ORB, how it is tuned, and the architecture it runs on. In order to compare ORB implementations, the architect or designer must first look at the architecture of the C2 system. How distributed is it? How many components are there? How fast are the Local Area Networks and Wide Area Networks? What are the distances traveled? How fast are the servers? How many processors are there? Where are the expected bottlenecks in the architecture?

These are all aspects of the architecture that can affect the performance of an ORB. Various ORBs will also respond differently to various architecture situations. To make an intelligent selection of an ORB, the architect or designer must understand all the factors that will affect the performance of the ORB, and which factors can be tuned to reduce their effect. In addition, the long-term effect of the tuning choices must be considered with regard to the system's maintainability, flexibility, and scalability.

During this research, more than 15 dimensions of variability were identified. Only 10 of these are discussed here. The others such as which language, operating system, and compiler, have less affect and can easily be held constant for the purposes of this evaluation. The remaining 10 will be discussed either independently or in combination. They are discussed roughly in the order that can cause the most impact on the transaction performance. Each is followed by some advice on how that factor can affect the choice of an ORB implementation.

Grain, Bandwidth, and Distance Combined

The first three dimensions of variability are addressed together to show the interplay, then addressed separately to show how they each affect the architecture's performance.

Communication delays are calculated as a function of bandwidth, distance, and data packet size. The reason is that while the distance determines when the first bit arrives at the destination, the bandwidth determines when the nth (last) bit arrives. The architect must choose a grain size that maximizes the combination of the distance and bandwidth. On the surface, it would seem a simple problem that as the bandwidth increases, so should the grain size. It is much more complicated than that.

For example, you want to ship a 1 Mbit file across the United States and receive a reply. At 64 Kbps, about 50 percent faster than a typical home modem, it takes nearly 16 seconds to get the data sent. The 30-msec latency delay for the distance does not add much. Today's standard ATM rate is 622 Mbps (OC-12)[1], with 5 Gbps (OC-96) already in use. At 622 Mbps it takes only 1.6 msec to deliver the message, so the 30 msec latency delay to wait for the reply ends up taking 95 percent of the time. As bandwidth increases, the time-to-reply for this example asymptotically approaches 30 msec [7].

There are other problems that the designer must face in this decision, such as buffer size, efficiency of the computation operation at each end, as well as the time required to checkpoint the data to permanent storage. From this description, it is easy to see why the time to perform the transaction is a function of the object's grain size, the distance traveled, and the bandwidth of the circuit.

> "Philosophically, it can easily be seen that any length of chain has a weakest link. Likewise, there is always a bottleneck in any process. One of the key roles of the software system architect is to understand, be able to detect, and manipulate the location of the bottleneck."

Object Grain

Grain refers to the size and layout of the data object being sent to the servant. The size, type of data, and layout of the data can each have an effect on transaction efficiency. Most of the computation time expended during the ORB transaction is applied to the marshalling and de-marshalling of the data. If the data type is a complex record consisting of a mixture of data types, the impact is higher. Large grained data objects also impact the data transport time, especially if the bandwidth is low [6].

So what? If your grain size is large/complex, look for an ORB with efficient marshalling and de-marshalling for the language you intend to use.

Communications Bandwidth

Bandwidth is described in terms of the clocking mechanism of the path and ultimately relates to the bit density of the data to be passed. Bandwidth can vary by several orders of magnitude with a corresponding affect on transaction performance. For a given distance traveled and a given data packet size, the time it takes to deliver that packet is a function of the bandwidth of the communications path.

So what? Take advantage of high bandwidth by finding a CORBA implementation that efficiently handles large data objects.

Distance Traveled

With the advent of giga-bit networks, we have transitioned from being bandwidth capacity constrained to being latency constrained [8]. This fact requires a change in thinking about how much data should be shipped at once. With low-speed networks, the bandwidth drove the decisions on object size. Now the distance traveled can cause the biggest difference in performance. The speed of electrons through a packet-switched network is about two-thirds the speed of light. Use 1msec for every 200 miles as a rule of thumb. If you are doing a method call on a server on the other side of the country, you have 30 msec of latency for the round trip before you add in any of the other performance factors.

So what? For short distances, optimize for an ORB that efficiently handles high volumes of transactions.

Dynamic vs. Static Invocation

Deciding how to invoke the method call also affects the efficiency of the transaction. As stated earlier, static invocation is strongly typed and dynamic invocation is weakly typed. Any Ada programmer knows that strong typing is a good thing and weak typing is not, so this should be an easy choice. However, this line of reasoning ignores one of CORBA's real strengths: the ability to hide the implementation details of the distributed environment from the programmer. Some of those implementation details include in what

[1] *The OC designation stands for Optical Channel, a unit of measure for fiber networks with a single Optical Channel, OC-1, delivering approximately 54 million bits per second.*

programming language the invoked method is coded, and what operating system is running on that server.

Dynamic invocation allows you to make calls for service at runtime without prior knowledge of language or data format requirements of the called service [9]. Static invocation makes little or no changes to the data object prior to shipping it, and requires little or no change prior to presentation to the servant. Obviously, this technique will improve transaction performance, but at the cost of flexibility. Static invocation can be used if the language, compiler, and operating system (and hardware) are known to be the same on both client and server.

So what? If using static invocation, optimize for a CORBA ORB that efficiently handles your data types.

Number of Processors — Number of Threads

As the cost of a processor continues to come down and as the architecture of the servers and the operating system allows, the number of processors in a typical C2 system continues to increase. Often 20 or more, and even up to 256 processors can be found in today's servers. Parallel processing of the ORB transactions greatly reduces the queuing affect at each stage of the processing; therefore, the time required for the transaction is also a function of the number of processors. The number of processors can sometimes exceed the number of threads.

So what? Look for a CORBA ORB that works well with a context-switching operating system.

Modern operating systems permit multiple threads of control within the same session. Like the number of processors, the number of threads can change the queuing affects at each stage of the transaction; therefore, the time to perform the transaction is also a function of the number of threads [10]. The number of threads often exceeds the number of processors.

So what? Look for a multi-thread capable ORB that can efficiently manage a thread pool (including protection against priority inversion).

Backplane Speed

The discussion on backplane speed is subtly different than bandwidth. Multiprocessor-capable servers use a high-speed (to extremely high-speed, i.e. 6 GHz) bus, generally referred to as a backplane, to facilitate communication between the processors. Like bandwidth, the speed of the backplane affects the time to transfer data between processors. When CORBA is used to communicate between processors within the same server, the time delays between processing stages are a function of the speed of the backplane. A server with a high backplane speed can communicate with much larger packets while not requiring an IP-type protocol (such as CORBA's Internet Inter-ORB Protocol), and therefore is far more efficient than a communications link with the same rate. In these cases, the efficiency of the ORB Core becomes the limiting factor.

So what? Look for a CORBA with an efficient broker, and one that is designed to adapt the protocol to the mode of communication.

Number of Servers — Normal Hashing or Perfect Hashing?

The more typical CORBA transaction is between two machines, a client and a server. Today's C2 systems often have several servers optimized for different operations: a communications server, a mission server, a database server, and a presentation server. The architecture may also include a combination of thin clients and thick clients. These architectures are referred to as N-tiered architectures.

In N-tier architectures, it is quite often found that the transaction process is further distributed, such that the ORB transaction can occur over three or more machines, for example: client, presentation server, and data server. The time delays occurring during the transaction are then a function of the number of servers the transaction is spread across. A higher number of servers complicates matters for the broker. Logical address resolution can become more difficult.

The broker uses a process known as normal hashing to dynamically calculate the route to the server. In a statically configured architecture, the designer can choose to short-cut this process by using a technique referred to as perfect hashing [4]. This process uses pre-calculated lookup tables to determine the route. Be cautioned though, this technique eliminates one of the principal advantages of CORBA: the flexibility to dynamically choose which of several servers will perform the method invocation as a way of doing load balancing. If the reduced flexibility is acceptable, be aware that the use of perfect hashing also increases the maintainability cost by requiring recalculation of the hashing tables every time the architecture is changed.

So what? If you have a fixed number of servers with fixed addressing, look for an ORB that allows perfect hashing.

Questions for the CORBA ORB Salesman

Armed with this new information about the factors to consider when selecting a CORBA ORB, you are ready to confront the ORB salespeople. Here are some sample questions you could ask:

- When I tested your competitor's product using a 7216 Byte packet size, we recorded an average one-way transaction time of 9.3241msec. Can you beat that?
- Do I have to use normal hashing or can I use perfect hashing?
- My entire system will be coded in Ada 95. What speed improvement can I expect using static invocation instead of dynamic invocation?

Here are some responses that would indicate you have a sales representative who knows the product and understands the implications of the alternatives:

- Was this a simple data structure or a complex one? What speed processor were you using? How many processors? How far apart were they? What was the speed of the backplane?
- Yes we support perfect hashing, but do you realize that you may get a minimal performance gain at the cost of less flexibility and more maintenance?
- The improvement will only be significant if you are using large-grained complex record types, and you run the risk of future software failure given an operating system upgrade on portions of the architecture.

Conclusions

Performance and flexibility are old rivals in computer architecture. Usually, design decisions made to achieve flexibility are detrimental to performance and vice versa. All of the decisions facing the architect therefore come down to how to best balance the needs of both goals.

Philosophically, it can easily be seen that any length of chain has a weakest link. Likewise, there is always a bottleneck in any process. One of the key roles of the software system architect is to understand, be able to detect, and manipulate the location of the bottleneck. Manipulating the location of the bottleneck is relatively straightforward: adding or taking away

processors here or bandwidth there. In the case of CORBA for C2 systems, the architect must do the following:

- Understand the architectural and performance needs of the system.
- Understand the dimensions of variability within that envelope to select the critical factors for comparison purposes.
- Apply the above to the selection of a CORBA ORB implementation that can be effectively optimized for that architecture.

What quickly became evident during this research was that in the age of gigaflop computers, the time it takes to send a message across a LAN, approximately 1 msec, is considered an eternity. Since these communications are controlled by, facilitated by, and in most cases are conducted on behalf of the middleware, its performance becomes paramount.◆

References

1. Croak, T.J. "Application of Capacity Planning Techniques as Architectural Design Decision Aids for 3-Tier and N-Tier Software Architectures." DCS Dissertation. Colorado Technical University, 2000.
2. Seetharaman, K. "The CORBA Connection." Communications of the ACM 41.10 1998.
3. Schmidt, D.C. "Evaluating Architecture for Multithreaded Object Request Brokers." Communications of the ACM 41, (Oct. 1998): 54-60.
4. Schmidt, D.C. "Principles and Patterns of High-Performance and Real-Time Distributed Object Computing." ACM Symposium on Principles of Distributed Computing, 1997.
5. Gokhale, A. and D.C. Schmidt, "Measuring the Performance of Communication Middleware on High-Speed Networks." ACM SIGCOMM Computer Communication Review 26.4 (1996): 306-317.
6. Gokhale, A.S., and D.C. Schmidt. "Measuring and Optimizing CORBA Latency and Scalability Over High-Speed Networks." IEEE Transactions on Computers 47.4 1998: 391-413.
7. Tanenbaum, A.S. Distributed Operating Systems. Englewood Cliffs, N.J: Prentice Hall xvii, 614, 1995.
8. Kleinrock, L., "The Latency/Band width Tradeoff in Gigabit Networks." IEEE Communications 30.4, 4 Apr. 1992: 36-40.
9. Gokhale, A., and D.C. Schmidt. "The Performance of the CORBA Dynamic Invocation Interface and the Dynamic Skeleton Interface Over High-Speed ATM Networks." IEEE GLOBECOM '96. London, 1996.
10. Schmidt, D.C., et al. "A High-Performance Endsystem Architecture for Real-Time CORBA." IEEE Communications 14.2, (1997): 72-77.

About the Author

Thomas J. Croak is currently chief scientist of the Joint Missile and Air Defense unit of Computer Sciences Corporation. Dr. Croak is a principal investigator analyzing compliance of Battle Management Command, Control and Communications Systems with Department of Defense standards for architecture, services infrastructure, and design philosophy. Dr. Croak is retired from the Air Force, and among his positions he was the senior software engineer of the Air Force and managed the Air Force Software Technology for Adaptable and Reliable Systems (STARS) program for DARPA. Croak earned his doctorate in computer science from Colorado Technical University.

Computer Sciences Corporation
1250 Academy Park Loop, Suite 240
Colorado Springs, CO 80910
Phone: (719) 572-2588
E-mail: tcroak@csc.com

COMING EVENTS

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

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

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

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

March 19-21, 2002
Federal Office Systems Exposition 2002
Washington D.C.
www.fose.com

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

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

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

Predicting Staff Sizes to Maintain Networks

Dr. Lon D. Gowen
The MITRE Corporation

MITRE completed a three-month study to assess the state of the practice in staffing levels for maintaining a computer-networking infrastructure (CNI). The state of the practice was determined by looking at technical papers on the subject, conducting organizational and technical-expert surveys, and looking at software models that attempt to predict staffing levels. There were very few quantitative heuristics available in the literature; however, the data did show that typical CNIs have a 1:42 ratio of support staff to users. That is, one full-time equivalent of CNI staffing per 42 users for a typical CNI. This number can vary, up or down, by 17 percent or more depending on the details of the CNI. The Department of Defense, as well as the private sector, can use the results of this study to predict initial CNI support levels, to support their current level of staffing, or to justify an increase or decrease in staffing. Additionally, this paper breaks down CNI support into four major areas: systems administration, hardware maintenance, help desk, and configuration management, and provides ratios for predicting each of them within a typical CNI.

Having the appropriate manpower to maintain a given computer-networking infrastructure (CNI) is an important factor to consider since only 26 percent of a local access network's total cost of ownership (TCO[1]) is hardware, while the remaining 74 percent is labor [1]. Of the 74 percent for labor, typically 43 percent is for end-user operations, 17 percent for technical support, and 14 percent for administration [1]. Other common reasons for having an accurate, up-to-date figure deal with budgeting, reliability, and quality. If an organization's CNI staffing levels are too high, then it wastes resources. If the staffing levels are too low, then response times, reliability, and end-product quality suffers; overtime is too high; and workers leave for a better working environment (since at present, the demand is considerably greater than the supply).

Every organization within the Department of Defense (DoD) has to estimate staffing levels (manpower) for maintaining their CNI. This activity takes place on a regular basis for most organizations whether part of the DoD or the private sector. While not a fascinating or appealing topic of research for many people, manpower-sizing predictions are a critical part of planning. Therefore, at the request of its DoD sponsors, MITRE conducted a three-month study to try to determine the current state of the practice in CNI staffing levels. The focus was primarily on the private sector. However, the study also looked at some DoD-based data.

> "Therefore, even if the data accurately portrays the state of the practice, it may not portray the optimum, since the state of the practice may not be optimal."

This article is a sanitized version of the study's full report [2]. Due to the full report's sensitive nature and critical views in certain areas, this article maintains the anonymity of informational sources. Such an approach was necessary to obtain honest data.

For purposes of this study, MITRE determined the state of the practice by collecting information from the following sources: recent technical papers (most within 24 months), organizations currently supporting CNIs, technical experts currently working in the field, and current modeling tools.

Current CNI Staffing Practices

This section lists the data that MITRE collected from the four sources mentioned previously. After discussing the data from these four areas, a section removes the outliers to show how well the data tightens up. There are some people who will have problems with dropping the outliers for statistical reasons; however, the whole purpose of dropping the outliers is not to present any kind of statistical proof. Instead, it is merely to show the effect such changes have on the averages and standard deviations of the remaining data. Such comparisons are very useful to certain organizations.

Table 1 is a list of the data MITRE collected. The next few sections reference this data in more detail; however, there are a few things worth mentioning. First, as the table shows, the deviations are too

Table 1: *CNI Staffing Data Collected*

Type of Data	Source of Data	Number of Users Per FTE of CNI Support			
		Systems Administration	Help Desk	Hardware Maintenance	Configuration Management
Technical Papers	1. Lucent INS [3]	155.2	113.8	284.5	853.5
	2. Gartner #1 [4]	106.7	77.6	106.7	
	3. Gartner #2 [5-6]	247.8	60.0		
	4. PC Week [7]		86.0		
	5. IDC [8]		99.0		
Org. Surveys	6. DoD	103.3	110.7	106.9	442.9
	7. Private A	80.0	80.0	80.0	
	8. Private B	71.0		71.0	
Technical Expert Surveys	9. DoD Sector	426.8	81.3	213.4	284.5
	10. Private Sector	227.6	136.6	162.6	227.6
COTS Modeling Tools	11. Run A	92.1	376.0	305.9	388.0
	12. Run B	80.8	199.4	193.1	292.8
Mean (Average)		159.1	129.1	169.3	414.9
Standard Deviation		112.9	90.2	86.6	228.4
Percent Standard Deviation		71.0	69.9	51.1	55.1

loose. Dropping a few outliers within the four areas of CNI support can tighten them. The numbers, however, still show useful similarities. For example, the averages between systems administration, help desk, and hardware maintenance are relatively close to each other but considerably smaller than configuration management. Second, the statistical means for all four areas seem reasonable and in general agreement. Third, the standard deviations (as a percent of the mean) are very high. When combining data from the four areas (see Table 2), the percentage drops drastically. While MITRE did not investigate this drop, there are two potential reasons: an inadequate understanding of one or more of the four areas by some or all of the sources, and a different operational definition of these terms.

Each ratio in Table 1 represents how many users one full-time equivalent (FTE) of CNI staffing can support for a given area. For example, Lucent's paper [3] recommends one help-desk FTE per 113.8 users. The shorthand for such a ratio, in this article, is 1:113.8.

Technical Papers

Of the 29 papers reviewed, only six papers [3-8] contained sufficient information to be useful for predicting CNI support levels. MITRE collected the six papers into five groups (see Table 1) combining the two Gartner papers. Worth noting on the Gartner papers is that their data on systems administration differs between these papers by a significant amount: 1:106.7 versus 1:247.8. The papers did not explain the reasons for this difference.

As Table 1 shows, the help-desk area receives the most research. Almost every source of data has recommendations for help-desk staffing, and the mathematical mean of their recommendations is 1:87.3 (one help-desk staff for every 87.3 users) with a standard deviation of 20.5 users. Therefore, depending on the CNI's environment, the typical number of help-desk staff can range from 1:66.8 up through 1:107.8.

With three data points, systems administration is the next most heavily discussed area among the sources. The mathematical mean of the ratios is 1:169.9 with a standard deviation of 71.7 users. One advantage of the technical paper data is that companies often consider it more accurate than other sources of data. Therefore, when other sources of data start to show similarities to the technical papers, they tend to confirm each other's validity.

	COTS Model Run A	Lucent Paper	Gartner Paper #1	Gartner Paper #2	DoD Org. Survey	DoD Expert Survey	Private Sector Expert Survey	COTS Model Run B	Mean	Std. Dev.
Users per FTE	51.6	50.2	31.6	48.2	33.0	43.8	44.9	38.5	42.1	7.3

Table 2: *Summary of Composite Ratios*

However, relative to other areas, the technical papers ignore both hardware maintenance (HM) and configuration management (CM). Only two papers contained hardware maintenance recommendations, and only one paper contained configuration management recommendations.

> "If an organization's CNI staffing levels are too high, then it wastes resources. If the staffing levels are too low, then response times, reliability, and end-product quality suffers; overtime is too high; and workers leave for a better working environment."

Organizational Survey

The organizational surveys represent data from existing organizations – some from the private sector and one from the Department of Defense (DoD). Unfortunately, in soliciting participation, none of the private-sector organizations were willing to participate openly. So MITRE submitted the survey anonymously to a different set of private-sector organizations in order to gain some unofficial information. MITRE obtained a few responses, but only two of them had enough clients and servers to be useful for this study. In general, the private-sector data has limited application since MITRE obtained data from small CNIs and obtained only two somewhat useful responses. As for the DoD, MITRE ran into the same problem with the exception of one very large DoD organization, which was willing to share its information. Since this study focuses on private-sector data, one data point here was sufficient.

Table 1 also summarizes the data MITRE collected from its organizational surveys. The mean systems administration (SA) ratio (1:84.8) and the standard deviation (16.7 users) are much larger (i.e., more FTEs) than are those of the technical papers. The data seems to show a large disconnect between the technical papers and actual practice for SA. MITRE did not investigate potential causes of this difference, but one possibility is that the research centers are overly optimistic. Another possibility is that this data does not accurately reflect what organizations (in general) are actually doing (i.e., since the sample space is so small, it is not accurately showing the state of the practice). The numbers for help desk (HD) need no comment, since they are in general agreement. As for HM and CM, the ratios, again, are larger than those of the technical papers. Again, MITRE did not investigate the reasons for this difference but the same possibilities exist.

From Table 1, one can see how closely the ratios for SA, HD, and HM are to each other for each of the organizations. The private sector explains that they view the three areas as having overlapping talent because data points are such small CNIs. For the DoD data set, the organization has such specialized systems that they require a large number of SAs, thus pushing the SA ratio close to the other two ratios. Without some compelling need (such as the previous examples), an organization would not have as many SA staff as HD staff.

Technical–Expert Surveys

The technical-expert surveys represented best guesses at how experts might staff a sample CNI. For this survey, MITRE used a CNI containing approximately

Type of Data	Source of Data	Number of Users Per FTE of CNI Support			
		Systems Administration	Help Desk	Hardware Maintenance	Configuration Management
Technical Papers	1. Gartner #1 [4]	106.7	77.6	106.7	
	2. Gartner #2 [5-6]	247.8	60.0		
	3. PC Week [7]		86.0		
	4. IDC [8]		99.0		
Org. Surveys	6. DoD	103.3	110.7	106.9	442.9
	7. Private A	80.0	80.0	80.0	
	8. Private B	71.0		71.0	
Technical Expert Surveys	10. Private Sector	227.6	136.6	162.6	227.6
COTS Modeling Tools	12. Run B	80.8	199.4	193.1	292.8
Mean (Average)		131.0	106.2	120.0	321.1
Standard Deviation		74.2	44.3	48.0	110.4
Percent Standard Deviation		56.6	41.7	40.0	34.4

Table 3: *The Data Without Outliers*

100 servers, 1,000 clients, and 1,100 users. One expert from the private sector and one from the DoD answered the survey. Per the agreement on the survey, both respondents remain anonymous.

Table 1 (see page 22) summaries the data from the two expert-opinion responses. The largest deviation between the two sets of answers is in the SA area, where the DoD expert's estimate is almost twice the private-sector experts estimate. There are also some drastic differences between the intra-organizational ratios. For example, the DoD response shows a clear spread between all four areas of CNI support with SA at the top of the graph and HD at the bottom. While the private-sector response is not as drastic, there is still a larger spread than found in the organizational survey.

Modeling

The modeling results represent data collected by taking the same scenario as the technical-expert surveys and running it through one of the well-known COTS modeling tools. Again, tool and vendor are anonymous.

Table 1 contains the data from two separate runs of the model: Run A views the CNI from a better light than run B (more details on this below). SA is very close between the runs. However, the other areas have a much larger deviation as the table shows. These differences are due to how the two sets of inputs characterized the scenario's CNI with respect to best practices and complexity, which are essential input parameters to the COTS model in question. Run A characterized the scenario's CNI as more advanced with respect to best practices than run B. Run A also characterized the scenario's CNI as less complex than run B. The two runs provide some insight into how these parameters affect the modeling tool and thus affect the staffing levels, which the model predicts.

Note that the tool's values for HD are significantly different from all other sources of values for HD. With a HD ratio of 1:376, the author believes the tool is modeling more of a customer-service center rather than a true help desk. The same holds true for HM. Since the vendor has no official tool validation, this issue may be a software error. Although this research strived to eliminate any differences between definitions, there may be a disconnect between the tool's terminology and those used in this article. That is, what the tool considers part of the HD support, this article may consider to be in some other category. These mapping issues are always a source of potential differences. However, these differences go away when combining the data as Table 3 shows.

> "... the help-desk area receives the most research ... and the mathematical mean of their recommendations is 1:87.3 ... with a standard deviation of 20.5 users."

The Outliers

As the previous section mentioned, there appears to be some obvious outliers in the data. This section removes some of those outliers merely to show the effects on the data – both numerically and visually – since some organizations find such information useful.

Table 3 removes three outliers from the data set. The table drops the data from the Lucent paper due to its very small ratios for HM (1:284.5) and CM (1:853.5). The ratio of one CM person per 853 users is significantly different from all other data points for this area. Next, the table drops the DoD experts data, since it had a very small SA ratio (1:426.8) relative to all other data points. Lastly, the table drops the values from run A of the COTS modeling tool since it had very low ratios for HD, HM, and CM relative to the other sources.

Composite View of the Data

To remove potential differences between how the sources used terms (such as SA, HD, HM, and CM) and to provide an easy metric for predicting staffing sizes for CNI support, this section combines the data producing an overall FTE-to-user ratio. The author picked users (versus something like servers or clients), because most research in the field uses this same unit of measure (i.e., FTEs per number of users). In some cases, such as systems administration, logic dictates that a different unit of measure (e.g., FTEs per number of servers) is best; however, since the common unit of measure is users, the composite figures use it. In addition, all previous values use this unit of measure as well.

Table 2 contains the composite figures, but only for those sources of data that contained heuristics in more than one of the four areas. For example, the IDC paper as well as the *PC Week* paper referenced only help-desk staffing, so using these figures in a composite chart are not appropriate or useful. This table also ignores the two private-sector organizational surveys due to their small CNI size. The remaining eight sources of data provide CNI staffing ratios with a mean of 1:42.1, and whose standard deviation is just 7.3 – significantly better than the deviations from the non-composite ratios.

Issues

Before concluding, the reader should be aware of the issues MITRE encountered that affected the research. Some of these issues listed in the accompanying sidebar are specific to this particular study, but most of them are generic issues associated with labor studies involving CNI support. Despite these issues, however, the author believes that the resulting data, as a whole, gives a realistic and accurate picture of CNI staffing levels – both generically and specifically – since there is general agreement among the four areas of source data. That is, the technical papers (while having some outliers) are in agreement with the organizational data, the technical experts, and the modeling data from the COTS tool. Since writing this article, another DoD organization (of about 500 technical employees) has commented that its CNI staffing is approximately 1:42, which further supports the conclusions above. Another supporting factor is that after removing the obvious outliers, the deviations tighten up significantly. Nevertheless, some caution is appropriate since there is always a chance that the data just happen to agree.

Lastly, the data may not reflect optimal staffing levels: There is no oracle to tell us the optimum. Therefore, even if the data accurately portrays the state of the practice, it may not portray the optimum, since the state of the practice may not be optimal.

Conclusions

The method for determining support levels for CNIs is still an art, not a science as many would like. The lack of public information is partly due to the proprietary nature of company information. Despite the problems with collecting CNI staffing information, the data set as a whole appears accurate and useful, since there is general agreement among the four sources of data. That is, the technical papers, despite having some outliers, are in agreement with the organizational data, the experts' opinions, and the modeling data from the COTS tool. Another supporting factor is that after removing the obvious outliers, the deviations tighten up significantly, and the composite data is very tight for the newness of the industry.

Nevertheless, some caution is appropriate since there is always a chance that the data just happen to agree. Additionally, the data may not reflect optimal staffing levels, and there is no oracle to tell us the optimum. Therefore, even if the data accurately portrays the state of the practice, it may not portray the optimum, since the state of the practice may not be optimal.

All charts in this report focus on user-based ratios for determining support levels (FTEs). However, there are other ratios: for example, those based on the number of servers and clients. When using these ratios, therefore, one must ensure an accurate census before trying to estimate staffing levels. If one uses the user-based ratios, then that person or group must ensure an accurate accounting of users in the targeted organization beforehand.

Lastly, while these research findings focus on the private sector, they have application to any CNI. The most applicable ratio is the average overall FTE ratio of 1:42; that is, one FTE of CNI support for every 42 users with a standard

Research Issues to Consider

Following are some of the issues MITRE encountered that affected the research. Some of these issues are specific to this particular study, but most are generic issues associated with labor studies involving computer-networking infrastructure (CNI) support.

Time Constraints

MITRE limited this study to a three-month effort: January 2000 through March 2000. Within this timeline, there were further tradeoffs dealing with how much time to spend in each area of interest vs. the thoroughness of the analysis. For example, how much time to spend on technical papers vs. organizational surveys vs. expert-opinion surveys vs. modeling.

Technical Articles

While staffing levels are important organizational concerns, there are surprisingly very few technical papers on the subject. Of the technical papers that do exist, only a small number (six according to this study) discuss algorithms for determining FTEs with regard to CNI staffing. Of the six papers that do discuss algorithms, most focus only on a subset of the four major areas of CNI staffing.

Organizational Surveys

Another surprising outcome from this study was that very few companies were willing to share their CNI data.

Modeling Tools

A few companies claim to have modeling tools for calculating CNI staffing levels. The costs for the presumably better COTS modeling tools are quite high; therefore, MITRE used only one of the leading COTS tools in its study. Unfortunately, as the author found out during the research, the company who developed this COTS tool did not independently validate it, therefore, providing no confidence that it computed reasonable or accurate results. Also, some of the model's inputs, which should be essential factors in determining FTEs, are for "informational use only" according to the tool's manufacturer[2].

Mapping Data

Some of the data from the technical articles and the COTS tool required normalization to ensure that the data were in agreement (i.e., that the research counted apples as apples and oranges as oranges). Everyone seems to have slightly different definitions for the four primary areas of CNI support, which makes studying this area extremely difficult. Early on, MITRE learned that trying to make the areas of study too fine would prevent certain people from wanting to participate and would take too much time; therefore, MITRE kept the granularity at a high level (i.e., simple).

deviation of seven users. For example, one environment might have a ratio of around 1:35 (i.e., more support staff), while another environment would be 1:49 (fewer support staff). The deviation is about plus-or-minus 17.3 percent of the mean ratio. The HD ratios should also have close applicability to other domains, since the HD area received a lot of attention in the literature and seems to have strong agreement within both the literature and the surveys.

Of the remaining three CNI support areas – HM, SA, and CM – both the HM and SA ratios should provide rough estimates to other domains, while other domains may have trouble using the CM ratio. The state of the practice is very unclear with respect to CM, which is why applying the recommended CM ratio may be difficult and inaccurate for other domains. The state of the practice for HM and SA is more thorough but still not as solid as HD. Therefore, when applying HM and SA, other domains may need to allow for a wider variance than they would for HD.

The author hopes this article will be helpful to many DoD and non-DoD organizations trying to wrestle with this difficult and costly problem. Hopefully other organizations, because of the difficulties MITRE encountered, will share information more freely in the future. Lastly, the author encourages colleagues in the DoD and private sector to pass along any CNI staffing data whenever and wherever possible. While MITRE collected all of the technical articles they could find, the author would appreciate hearing about any significant references that our searches may have missed, i.e., anything not listed in the references section.◆

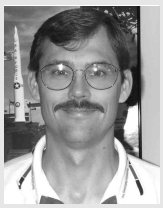
References

1. Weinberg, Neal. "TCO Tall Tales." Network World 7 June 1999.
2. Gowen, Lon. "Computer-Networking Infrastructure Manpower Study." The MITRE Corporation, Document MP-00B0000018. Apr. 2000 (not in the public domain).
3. "Network Operations Center Staffing Plan." Lucent International Network Services, Feb. 1999.
4. Silver, M. "The Staffing Connection to PC/LAN TCO." Gartner Group. 1 May 1998.
5. Cappuccio, D. "The Metrics of LAN Staffing – A Top-Down View." Gartner Group. Nov. 1996.
6. Apfel, A., K. McGee, J. Pultz, A. Schoeller, M. Zboray, and Ardito C. Smith, eds. "Selection Criteria for Tomorrow's Enterprise Networking Professionals." Gartner Group. 8 July 1998.
7. Plotnick, Neil. "Getting Your People Power in Perspective." PC Week, 12 Apr. 1999.
8. Kavanagh, Kelly, Tom Oleson, and Chris Hoffman, eds. "A Model for Determining Help-Desk Staffing." IDC Government. July 1998.

Notes

1. TCO is a term for which there is no "accepted industry standard"; however, the term usually includes just what its name says – all costs associated with owning a piece of hardware, including the support and maintenance.
2. An excellent area for research, therefore, would be 1) to compare as many of these models against each other as possible and 2) to determine their accuracy (i.e., attempt some sort of validation). Currently, there are no analyses in the literature (that the author could find) for any of these models.

About the Author



Lon D. Gowen, Ph.D., is a lead staff engineer for The MITRE Corporation. He currently works on a MITRE contract at the U.S. Strategic Command, Offutt AFB, Neb. Gowen has doctorate and master's degrees from Arizona State University with areas of interest in software engineering, software languages, and embedded systems. He has a bachelor's degree with a double major in computer science and applied mathematics from the University of Nebraska. Gowen was a Distinguished Visitor for the Institute of Electrical and Electronics Engineers Computer Society for three years. He also teaches the systems and software safety classes for the NASA Safety Training Center.

The MITRE Corporation
1004 Lincoln Road, PMB 327
Bellevue, NE 68005
Phone: (402) 294-2689
Fax: (402) 294-1264
E-mail: gowenld@mitre.org

Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity

Hillel Glazer
Entinex, Inc.©

Many process-oriented software developers (some of whom use the CMM) think of Extreme Programming (XP) as a "seat-of-the-pants" development method. Many high-speed cutting-edge developers (whether they use XP methods or not) see CMM as a cumbersome unnecessary impediment to developing software quickly. This is the result of a myth: Software development speed must be sacrificed when following a process-disciplined approach (such as CMM). This myth is defeated when we look at two realities: The CMM is tailorable, and XP is disciplined. An alternate look at the realities helps open up a new possible approach so that these methods can work together. This article puts forth ideas to bridge the gap between the two sides using the suggested approach, and concludes that process discipline can be achieved without sacrifice to the speed of development.

In the quest for *better, faster, and cheaper,* many companies are wrestling with a methodology problem convinced that their choice is between agile development vs. stable processes. People on both sides of the mat are sure that the two are perpetual adversaries. This article puts an example from each side in the match: Representing agile development will be Extreme Programming (XP); and representing stable processes will be the Capability Maturity Model® (CMM®). It follows a train of thought that seeks to dispel the adversarial myth by looking at its possible origins then building a bridge into reality.

The Myth

The misconception among many commercial software developers is that process discipline in software development (such as the CMM) is incompatible with fast-moving development processes such as XP. A similar misconception among many process-oriented people – CMM or otherwise – is that developing software quickly is tantamount to chaos. If these two views persist, they will keep excellent development teams from realizing the benefits of structured process improvement, and likewise keep larger organizations from looking at alternative development methods. They will be forever locked in a perpetual wrestling match.

Let's face it, whenever someone says CMM, most people think of big, lumbering, cumbersome, bureaucratic paperwork and with good reason. When people think of the places that have typically applied CMM to their organization, or when they look at the place that developed the guide – the Department of Defense (DoD) and their contractors – this could be an apt description, often followed by a sense of dread and loathing.

A quick look at XP may help frame the discussion. From Don Wells' [1] comprehensive Web site on the subject, we can learn that XP has well-defined rules and practices that can be summarized into four main areas: planning, designing, coding, and testing. In the June 2001 issue of CROSSTALK, Leishman [2] discussed the differences between the traditional and the XP development life cycles, while Duncan [3] did an excellent job of expanding on the requirements aspect of software planning. Figure 1, courtesy of Wells, depicts a typical XP project. In Figure 1, readers seeing unfamiliar terms such as *spike* and *system metaphor*, or the unorthodox placement of *acceptance test* or *test scenarios* are encouraged to investigate the referenced materials for more information.

XP is an exercise in iteration. The four XP rules areas are not a sequence for the entire project in one shot through. They contain activities that occur with each iteration. Code is developed by pairs of programmers, tested, and integrated in very small increments. Not only are the requirements gleaned from the user *stories* (much like use cases), but the customer is intimately involved with what and when code is implemented based on the progress of the development and the planning results. Furthermore, XP has rules that govern what *small increment* really means. Planning also includes what many would call project estimating, tracking, and controls, as well as changes to how future XP cycles will apply the experience gained from the previous iterations.

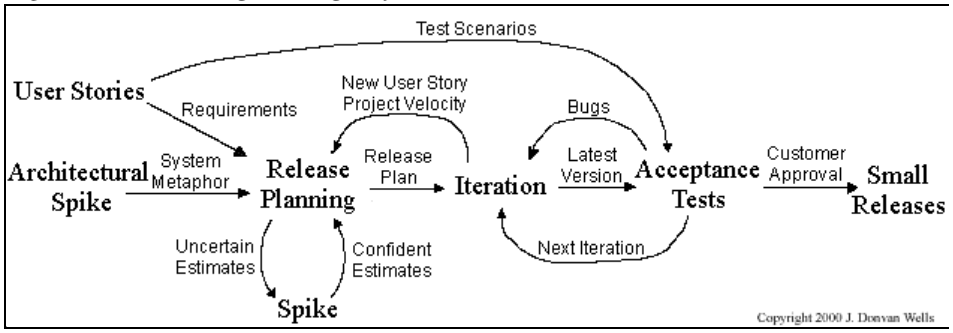
Designing and coding are distinct activities in XP. However, they occur along with testing in very tight yet simple formation. XP does not have a coding standard, except to specify that there must be one. There are several other philosophical and practical aspects to XP based on the XP creator's experience such as when code is to be *reused*, the details of designs, the timing of functionality, the characteristics of the development team, and how to ensure a closed-loop traceability between testing and design.

To be sure, given the compact time frames and immediacy of customer access, one can guess that XP is not intended for large and/or extremely complex projects spread across several locations with no single customer voice. Some say 20 developers would be a big team.

The Reality

The truth is that the CMM – and process

Figure 1: *Extreme Programming Project*



® Capability Maturity Model and CMM *is registered in the U.S. Patent and Trademark Office.*

discipline in general – do not have to send a chill down the developer's spine. One of the most overlooked aspects built right into the CMM is the fact that it is meant to be tailored to the organization. Another significant but often-overlooked facet is the definition of *maturity* as it applies to software development. The definitive text on the CMM, Paulk, et al [4], refers to immature organizations as those whose "processes are generally improvised." And that among immature organizations "even if a software process has been specified, it is not rigorously followed or enforced."

One thing we can see about XP, even from this brief explanation, is that it is not improvised. As any XP developer will tell you, it will not work unless the XP methodology is followed. In fact, looking at what makes a process mature is simply that it is (again, thanks to Paulk) "explicitly defined, managed, measured, controlled, and effective."

Why then is there such an impasse between the goals of agile, creative, and nimble development and the goals of process improvement? Could it all be related back to the same basic sticking points found in most situations? Could it be a simple matter of defining terms and expectations? What if the problem were as simple as that of confusing the *how* of development with the *what* of development? The *pre*scriptive vs. the *de*scriptive.

The Recipe vs. the Menu

A few words may be appropriate here to further explain the previous paragraph. Many standards, frameworks, and methodologies developed by government and industry are very rigorously defined. Like a recipe, they prescribe what to do and how to do it: measure three cups flour, beat in two eggs, grease a 9-inch x 11-inch x 1.5-inch pan, etc. They are full of verbs and adverbs.

On the other hand, a menu describes items that are listed: appetizer: salad or paté … ; soup: cream of mushroom or tomato … ; entrée: baked salmon or roasted chicken … ; dessert: chocolate brownies or vanilla ice cream … Menus are mostly nouns and adjectives.

From a menu you can tell a lot about an establishment. You can tell what their strengths are; you can tell what ingredients they like to use; in most cases you can tell whether you might find something to fit your appetite. Among better establishments, you are likely to find similar characteristics from menu to menu.

The CMM is more like a menu. It does not tell you how to develop software, or how to manage your software development. It simply lists those items found on the menu where good software products are served.

Given the history of most standards, in an industry with more than enough recipes, a menu is a challenging mental switch for some people to make. As a result, many think they are being told to cook, when all they are being told is to dine.

The Suggestion

How does this apply to XP vs. CMM? Let us say that the friction shows up due to a misinterpretation of terms. For example CMM'ers looking at XP see an undisciplined software management and improvement methodology, and XP developers see CMM as a too rigid development methodology. Well, there is the possible source of the answer!

> "If XP is viewed as a development methodology, what is to keep a software management methodology such as the CMM from being there at the same time?"

What if we chose to distinguish XP as a software *development* methodology and CMM as a software *management* methodology? What if XP and CMM were not in any way working at cross purposes. What would we find if we looked closely at the difference between development and management? Could the two be viewed as complementary – even mutually supportive of one another? Would that get us closer to solving the problem? I think so.

To help understand the difference between development and management methodologies, we will look to the hardware world for an example. Hardware can be designed and manufactured in any one of several ways. We will call these the development methodologies. The design can be made on paper or by using computer-aided design (CAD) systems. The manufacturing can also be by hand or can employ any number of automation systems at various steps in the production process. Other aspects of hardware production are the tools and tool control, inspection, inventory control, materials ordering, environmental controls, organizational needs, and so on. These latter aspects can be called management methodologies.

The development and management methodologies, therefore, are distinct disciplines. While the two are not completely decoupled, one does not dictate the other. Obviously the management methodologies must complement and support the development methodologies. They must work together to achieve business goals. A desired state is that they are each optimized to work in the same business and operations strategy models. However, fundamentally, whether you draw design blueprints by hand or by CAD is not dictated by how you control the flow of material through the plant.

The Bridge

In the software world, the CMM does not care what development methodology you use. It does not say that the Waterfall [5] model is better than the Spiral [6] model. Beyond that, it does not even say which life cycle or development models to choose. If XP is viewed as a development methodology, what is to keep a software management methodology such as the CMM from being there at the same time? Taken a step further, if the CMM is viewed as a management methodology for software process improvement, we can completely erase any *forced* divorce between CMM and XP.

In fact, as a development methodology, XP goes a very long way toward having a development team behave as quite a mature software process. Contrary to the perception among many organizations, as a development process, XP can be described as follows:

- Disciplined.
- Not an automatic solution to getting projects done better, faster, cheaper.
- Dependent upon constant communication within the development team and with the customer.
- Packaged to include many of the hard-taught lessons learned from many years of practical development experience.

As a result, the XP development methodology Rules and Practices almost explicitly mirror all but the Subcontract Management, and Quality Assurance CMM Level 2 Key Process Areas (KPAs).

Of these last two, if subcontracting exists on XP projects it would have to be addressed, but if not, then it can be tailored out. The last remaining item is soft-

ware quality assurance (QA). I am sure many readers familiar with XP think I am insane – seeing that with all the testing and iterations in XP, QA was not included among those KPAs satisfied by XP. This is due to another of the misunderstandings in engineering (not just software) that I will briefly address. QA is not quality control (QC).

On the Side

To put it simply, QC is testing. QC is definitely a major player in the XP methodology. QC is a step in any of the development methodologies. In XP, QC shows up all over the place in the coding and testing areas. In fact, QC is built into the planning and designing phases of XP before coding begins. The resulting code (when programming in the XP method) comes from having coded the unit tests first after understanding the component requirements. There is also a lot of QC (we hope) in every development shop. But what QC *is not*, is QA.

What is different between QA and QC? In a nutshell, QA is *process* oriented and QC is *product* oriented. *Testing*, therefore is product oriented and thus is in the QC domain. Testing for quality is not *assuring* quality, it is *controlling* it.

QA makes sure you are doing the right things, the right way. QC makes sure the results of what you've done are what you expected. Some people would prefer we redefine these as *product QA* and *process QA*. An approach that I could endorse, but that is not at issue here.

Nonetheless, while there is a lot of obvious QC in XP – testing – QA is not that far out of the XP Rules and Practices either. There is a lot of wisdom built into the XP Rules and Practices that when followed, are the fixing's for predictably high quality output. In fact, some could argue that preplanning the unit tests and then coding the software is a unique approach to QA. The key ingredients that would add the right QA flavor to XP are management visibility, independent review and audit, and assurance of the application of standards and the development process.

In other words, most XP projects that truly follow the XP Rules and Practices could easily and quickly be assessed at CMM Level 2 if they could demonstrate having a process for the following:

- Ensuring that the XP Rules and Practices are taught to new developers on the project.
- Ensuring that the XP Rules and Practices are followed by everyone.
- Escalating to decision makers when the XP Rules and Practices are not followed and not resolved within the project.
- Measuring the effectiveness of the XP Rules and Practices.
- Providing visibility to management via appropriate metrics from prior project QA experience.
- Knowing when the XP Rules and Practices need to be adjusted.
- Having an independent person doing the above.

The Resolution

If there is one thing developers have likely understood by now, it is that there is no *silver bullet* – either in development or management methodologies. *A* road to better, faster, and cheaper is not *the* road to better, faster, and cheaper. This article in no way suggests that XP is appropriate for all projects, or for any organization, or is without developmental disciplinary shortcomings. I am suggesting that there is a workable solution to organizations pursuing the use of dynamic, highly agile development methodologies (such as XP) within the context of process discipline (such as CMM).

> "If an XP team decided to add quantitative management (perhaps even statistical techniques) to provide more efficient real-time feedback, it could probably achieve Level 4 process capability."

With the understanding that XP is a software *development* methodology and that CMM is a software *management* methodology, and with people who can tell the difference, these two methodologies cannot only co-exist, but they can generate a mutually supportive environment, profitable company, and reliable product.

In this example, by documenting a project's approach to XP, or even closely following one of the many existing documented approaches, then by introducing the QA KPA, projects are already very close to CMM Level 2. With a little more work at the organizational level, CMM Level 3 is not far off. In an article on the Institute of Electrical and Electronics Engineers' computer.org web site Paulk [7] considers "… XP to be another example of a good software process (or philosophy), at least within the proper context, that would satisfy many Software CMM Level 2 and 3 goals." He also adds, "if an XP team decided to add quantitative management (perhaps even statistical techniques) to provide more efficient real-time feedback, it could probably achieve Level 4 process capability."

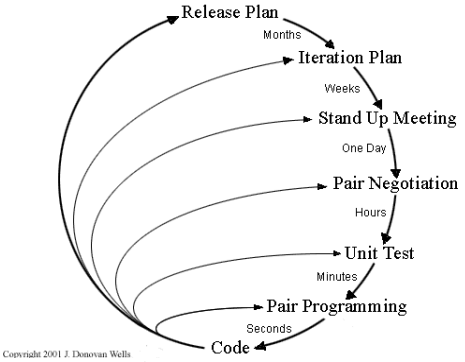


Figure 2*: Release Iteration in XP*

Another Wells' [1] diagram in Figure 2 provides an idea of the detailed development release process. The macro-cycle depicted here offers insight into opportunities for several CMM KPAs such as project planning, project tracking and oversight, organization process definition, intergroup coordination, and others.

Ron Jeffries [8] wrote a quick article outlining the activities performed on a project that used XP and how they related to the CMM's upper four levels. While a very cursory sample, it does convey that there is some interest in seeing the two methodologies work together.

While the list of projects experimenting with or transitioning to XP is as dynamic as the methodology itself, DoD projects are likely to be few among them. As mentioned, XP is intended for small teams of programmers. If DoD projects can be broken down into smaller projects and integration of these components can be tightly managed, then perhaps even these projects can try XP.

At publication, this author personally only knew of one organization giving serious thought to developing software using XP within the CMM process framework. However, although specifics of the effort were proprietary, the prognosis of successfully pinning the myth to the mat is very positive.

The Conclusion

There are many corporate and technical leaders looking to find effective paths

toward better, faster, and cheaper. There has long been the perception that while CMM managed organizations may achieve the *better*, the jury is still out on *cheaper*, and *faster* is clearly not readily evident. Especially when going from Level 1 to Level 2.

Projects thinking of using XP in organizations already assessed against the CMM are encouraged to shed the myth that they could *lose* their CMM rating. Organizations that use XP on their projects wanting to fulfill the intent of the CMM's KPAs are encouraged to shed the myth that they will be *bogged down* with the burden of dead trees. I posit that a symbiotic relationship exists to be found between the speed of agile development methodologies such as XP, and the direction of process improvement management methodologies such as CMM.

The first step is to understand why your processes do or do not fulfill the intent of CMM. Then plot the path of how to make your processes what you need them to be. The intent of the CMM is what you need to demonstrate. If you are effectively using a development methodology like XP, you are already nearly there. All you need to do is prove it.

One path to better, faster, and cheaper can be found outside the development myth in the peaceful coexistence of agile programming and structured processes and process improvement.◆

References

1. Wells, J. Donovan. "Extreme Programming: A Gentle Introduction." <www.ExtremeProgramming.org>.
2. Leishman, Theron. "Extreme Methodologies for an Extreme World." CROSSTALK, June 2001: 15-18.
3. Duncan, Richard. "The Quality of Requirements in Extreme Programming." CROSSTALK, June 2001: 19-22, 31.
4. Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis, eds. The Capability Maturity Model: Guidelines for Improving the Software Process. Software Engineering Institute. Addison-Wesley Longman, 1994.
5. Royce, W. W. "Managing the Development of Large Software Systems." Proceedings of IEEE WESCON. Aug. 1970.
6. Boehm, Barry. "A Spiral Model of Software Development and Enhancement." ACM SIGSOFT Software Engineering Notes. Aug. 1986.
7. Paulk, Mark. "XP from a CMM Perspective." IEEE Computer Society. Dynabook, 2001. <www.computer.org/seweb/Dynabook/PaulkCom.htm>.
8. Jeffries, Ron. "Extreme Programming and the Capability Maturity Model." 1 Jan. 2000. <www.xprogramming.com /xpm ag/xp_and_cmm.htm>.

About the Author

Hillel Glazer is the principal consultant of Entinex, Inc. He brings a broad spectrum of experience in process engineering and management. He is a student of the evolution of process-centered design, development and production and has followed the progress of Total Quality Management, Integrated Product and Process Development, ISO 9000, and the Capability Maturity Model from their emergence and introduction at the Department of Defense to their subsequent migration to the private sector. The focus of his career is on the issues of product integrity and technology management. He specializes in the management-driven engineering principles of quality, operations, risk, requirements, productibility, configuration, and project management. In merging these disciplines with business and operations strategies he emphasizes the importance of thoroughly planned and integrated process management. He has successfully adapted and evolved these disciplines across the Internet, software, and manufacturing industries.

Entinex, Inc.
1516 Castle Cliff Place
Silver Spring, MD 20904
Phone: (301) 384-4203
Fax: (240) 465-0062
E-mail: hillel@entinex.com

WEB SITES

Object Management Group

www.omg.org

The Object Management Group (OMG) is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Its membership roster, about 800 strong, includes virtually every large company in the computer industry, and hundreds of smaller ones. OMG's best-known specifications include CORBA, OMG IDL, IIOP, the OMA, and Domain Facilities in industries such as healthcare, manufacturing, telecommunications, and many others, UML, the MOF, and CWM. All of OMG's specifications may be downloaded without charge.

Distributed Objects & Components

www.cetus-links.org/oo_distributed_objects.html

This site of general information on distributed objects and components is part of the Cetus Links network. Cetus Links offers quick access and a comprehensive overview of tens of thousands of interesting pages about object-orientation and component-orientation that exist on the Internet. The Cetus Links can be regarded as an index to Internet addresses (http, ftp, and mailto) about object-orientation and component-orientation.

Distributed Object Computing with CORBA Middleware

www.cs.wustl.edu/~schmidt/corba.html

This site features mini-tutorials, including an overview of CORBA, research, on-line specification, related papers, tools, the ACE ORB (TAO), and CUJ and C++ report columns. It also describes the contents of the series of C++ Network Programming books written by Douglas C. Schmidt and Steve Huston.

IEEE Computer Society

http://computer.org

With more than 100,000 members, the Institute of Electrical and Electronics Engineers (IEEE) Computer Society claims to be the world's leading organization of computer professionals. Founded in 1946, it is the largest of the 36 societies of the IEEE. The society is dedicated to advancing the theory, practice, and application of computer and information processing technology. The site features listings of conferences, journals, technical committees, standards working groups, and more, to promote an active exchange of information, ideas and technological innovation among its members.

There's No Shame In Saying, 'I Don't Know!'

A few months ago, I was asked to write this month's BackTalk. I readily agreed. (I LOVE to see my name in print – and the only way I've found to see it often is to write an article myself.) However, I had no real clue what to write about. Luckily, as the deadline for the article approached, Associate Publisher Elizabeth Starrett sent me an advance copy of the articles in this month's CROSSTALK.

After reading the great line up of articles she had assembled for the month, (hey – she brings chocolate to our reviewers' meetings, so I am always nice to her), this BackTalk practically wrote itself.

First, go back and read the publishers' note. Go ahead. Next, understand that Beth referred to the fact that she would have been greatly helped in a previous job if she had had the knowledge contained in this issue of CROSSTALK back then. What she was saying is that she didn't know that there was additional knowledge out there that she could use. In other words, she didn't know that she didn't know enough. The way I used to envision it, you either know something, or you don't know something. However, I now see that you can either know or not know something, and you can also know or not know whether you know or not. (Read the sentence again – it will eventually make sense.)

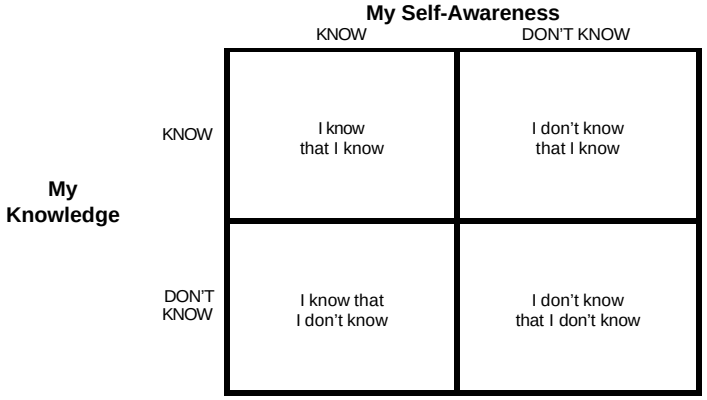


Figure 1: *Knowledge Dimensions*

So, there are two dimensions of knowledge: The first dimension is what I know; the second dimension is my awareness of my knowledge. Figure 1 shows the four possible combinations of knowledge and self-awareness. Given a specific topic, your knowledge fits into one of the quadrants above.

Now the problems in communications become clear. I (of course) am in the "I know that I know" group. I am reasonably well educated. (Have I ever mentioned that I went to Texas A&M?) This is the best quadrant in which to belong – awareness that you are knowledgeable about a subject.

If you are not lucky enough to be among the "I know that I know," group, then I suppose the next best option is to be in the "I know that I don't know" quadrant. You lack knowledge – but are aware of your ignorance. You are teachable. You can admit to yourself and others that you don't know everything, and you are willing to learn.

Unfortunately, not all of us are so self-aware. Some of us belong to the "I don't know that I know" quadrant. You have knowledge, but are unable to either apply or use the knowledge you have. You've wasted your education.

And, coming in last and least, is the "I don't know that I don't know" quadrant. Here is where communication becomes a problem. A person who fits into this group is ignorant about a topic, but isn't even aware that they are ignorant. Of course, being unaware that they are clueless, they wander around in blissful ignorance. Unfortunately, they seldom wander around in blissful ignorance in silence. They become self-proclaimed experts, ready to share their opinions to anybody ready to listen, thus making life miserable for those around them. Because they are totally unaware they are clueless about a topic, they are sure that they are experts, and frequently refuse to listen or learn from others.

Now, here's the scary part. If I think that I am an "I know that I know" person, and I'm wrong, well, that makes me a "I don't know that I don't know."

What's the cure for being an "I don't know that I don't know?" Unfortunately, often there isn't one. Self-proclaimed experts are hard to cure. Sometimes, they are even hard to recognize. I have listened to people who have never written a line of code lecture me about which programming language to use. I have listened to people that never managed a project explain how to create a schedule and how to gather requirements. Academic theory is one thing, but until you've managed a project, designed a large system, or written a real-time program, you just "don't know!"

Just having a college degree does not make you an expert in software development. You have to do it to truly understand it. Doctors, after graduating from medical school, cannot practice medicine. They have to complete a residency to show that they have practical knowledge along with theoretical knowledge. Why do we think we are different? Doctors who are true experts are still said to practice medicine. They continue to learn as they perform their craft. Again, why should we be any different?

To become real practitioners, we have to practice for a while to become qualified. The first step on the road to becoming an expert is saying, "I know that I don't know" about some topic. There is no shame in saying, "I know that I don't know." We all have gaps in our knowledge. We cannot be experts in everything. We need to recognize the gaps in our knowledge, and learn from those who "know that they know." Find the real experts, and learn from them. And if you think you are one, maybe it's time to look one more time.

I never thought I would be quoting poetry in a BackTalk, but the following verse definitely applies:

> O wad some power the giftie gie us –
> To see oursels as ithers see us!"
> ("O would some power the gift to give us –
> To see ourselves as others see us!")
> – From Ode to a Louse
> *Robbie Burns*

I am not sure why Robbie Burns wrote an ode dedicated to a louse – but I am sure you can come up with louses on your own that this ode applies to.

– David A. Cook
Software Technology Support Center
david.cook@hill.af.mil

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

CrossTalk / TISE
7278 4th Street
Bldg. 100
Hill AFB, UT 84056-5205

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